

AD659379



LAMBDA

REPORT 3

AN OPTIMIZATION STUDY OF
BLAST SHELTER DEPLOYMENT

David L. Mitchell

VOLUME III

Appendix H - BLAST - The Computer Program

September 1, 1966

OCD Contract PS-66-113, Work Unit 1632A

144

LAMBDA REPORT 3

AN OPTIMIZATION STUDY OF BLAST
SHELTER DEPLOYMENT

by David L. Mitchell

VOLUME III: APPENDIX H - BLAST - THE COMPUTER PROGRAM

September 1, 1966



PREPARED UNDER
INSTITUTE FOR DEFENSE ANALYSES
SUBCONTRACT 138-5

FOR

OFFICE OF CIVIL DEFENSE
OFFICE OF THE SECRETARY OF THE ARMY
WASHINGTON, D.C. 20310

CONTRACT OCD-PS-66-113
WORK UNIT 1632A

THIS REPORT HAS BEEN REVIEWED IN THE OFFICE OF CIVIL DEFENSE AND
APPROVED FOR PUBLICATION. APPROVAL DOES NOT SIGNIFY THAT THE
CONTENTS NECESSARILY REFLECT THE VIEWS AND POLICIES OF THE OFFICE
OF CIVIL DEFENSE.

LAMBDA CORPORATION
1501 WILSON BOULEVARD
ARLINGTON, VIRGINIA 22209

THIS DOCUMENT HAS BEEN APPROVED FOR PUBLIC RELEASE AND SALE.
ITS DISTRIBUTION IS UNLIMITED.

Best Available Copy

TABLE OF CONTENTS

	<u>Page</u>
I. Introduction	1
A. Generation of Defense	2
B. Evaluation of Defense	6
C. The Bounding Procedure	7
II. Use of the Program	9
A. Basic Program Operation	9
B. Catalog of Run Options	12
C. Glossary of Variables in Common	43
D. Input Data Sequence Definitions	51
E. Values of Important Program Constants	62
III. Sample Deck	63
A. Basic Data Deck	63
B. Shelter Input Deck	72
C. Sample Run Deck #1	83
D. Sample Run Deck #2	92
E. Sample Run Deck #3	98
F. Sample Run Deck #4	101
G. Sample Run Deck #5	104
IV. Subroutine Descriptions	107

I. INTRODUCTION

The basic tool for the blast shelter study is a computer program, BLAST. This program performs three functions which operate independently, but are contained in the same program so that subroutines and data may be shared and so that any common parts will be consistent.

The first function of the program is to generate a blast shelter defense posture of a specified cost level for whatever population data is supplied, and then to calculate the fatalities which that posture permits for a range of attack levels. The second function of the program is to compute, for any cost level specified, upper and lower bounds for the optimum defense performance at each attack level - the bounding procedure. The third function is to evaluate independently the cell model by comparing results using the cell model with results using a separate laydown method - the targeting model. Since the targeting model is used quite separately from the rest of the program, that portion of the program is discussed in Appendix B.

The program itself has the same three divisions with a driver program which controls the selection of the program to be run. If the driver data is read in, parameters are set, and then the portion of the program containing the desired program (called an overlay) is read in. The desired program is executed using subroutines in the main portion of the program plus the proper overlay.

The next three sections describe how to use the program in detail. The first describes the options and parameters to be set, the second gives sample data decks for the different runs and the data sequences for all possible types of runs, and the third describes the operations of the subroutines.

A. GENERATION OF DEFENSE

The program is designed to find a defense which minimizes fatalities, subject to a cost constraint. Each possible defense for each individual cell is tested under an attack optimized against that cell with that defense posture; and it is this optimum kill that is minimized by the program for a given cost. How the program does this is described later in this Appendix. The purpose here is to discuss the parameters which may be used to vary that defense.

Aside from the population data and the shelter and hardness data, only three parameters concerning the defense need to be set. The first is TCOST, the cost of the nationwide posture. The second is IFILL, which is the shelter occupancy or filling mode. IFILL = 1 is an optimal filling mode; i.e., fill shelters starting with the hardest until either people or shelters run out, and assign any remaining population to the lowest shelter category. IFILL = 2 is a different filling mode, which assumes that some specified fraction of the population is alerted. Those alerted fill shelters in proportion to the number of spaces available; that is, if the shelters are underfilled, each shelter is filled to the same percentage.*

To understand the third parameter, we need to give consideration to what choices may be made for the defense on each cell.

*A third possibility for IFILL, for use with balanced defense, has no interpretation as a filling mode; rather it was the most convenient method for implementing balanced defense.

The defense options are a number (15 or less as the program stands) of local shelter mixes available for use on each cell. Each shelter mix defines the percentage of the population in that cell to be assigned each type of shelter. The entire nationwide defense consists, then, of a set of numbers (from 1 to 15) specifying which shelter mix is assigned to each cell. The point of this discussion is that since there are two populations considered in this model (night and day), it must be specified which population is to be used for each cell in determining how many shelters is meant by some defense option*. The rule for selection of the population, IDPOP, to be used across the nation serves as a basis for determining the number of shelters to be assigned to each cell. Of the many possible rules, four are available in this program:

- IDPOP = 1 in each cell, select nighttime (resident) population:called NIGHT
- = 2 in each cell, select daytime population:called DAY
- = 3 in each cell, select whichever is larger, night or day:called MAX
- = 4 in each cell, select whichever is smaller, night or day:called MIN

Although TCOST, IFILL, and IDPOP are the only three parameter choices which need to be made concerning the defense, it must be observed that if the program is to minimize fatalities in an optimized attack, some assumptions need to be made about the aims of the optimized attack. Note that this is not the same as specifying an entire attack in advance; this is merely specifying the goals to be achieved in the attack optimization. There are three major assumptions to be made.

The first assumption about the attack is an assumed time of attack. This choice is made by setting the variable IAPOP. The values for IAPOP are 1,2,3 and 4 with

*For example, if defense option 11 is 70% of the population supplied 100 psi shelters, and if cell 58 has 100,000 people in the day and 53,000 at night, we need to know whether defense option 11 means assign 70,000 shelters or 37,100 shelters.

the same definitions as for IDPOP. In contrast to IDPOP, of course, the attacker cannot really choose a mixed rule for selecting population, such as values 3 or 4; his only choice is to select a time of attack, night or day. However, these "mixed rules" for IAPOP generate defenses which may be of interest; hence, they are included as possible values. To create a defense, one must set, in addition to other parameters, both IDPOP and IAPOP. Selections for these parameters are written in the form MAX/DAY, meaning the choice for IDPOP is 3 and for IAPOP is 2.

The second assumption about the attack concerns the objective of the attacker. The attacker optimizes his payoff, which may be either population or industry, or any linear combination of the two. The program can handle up to 10 of these linear combinations, and these are referred to as the attack objectives, each of which is defined by a population multiplier POPMULT, and an industrial multiplier, VINDMULT. Then, in making the shelter deployment, the defense may assume any one of those attack objectives or he may assume any number of them with a weight or confidence attached to each. This confidence is defined in an array WDAOB, the weight the defense assigned to each attack objective. See the sample run decks for examples of these attack objectives.

The third assumption the defense must make about the attack concerns the attack level, or total number of equivalent megatons expected to be delivered. A defense optimized against a few megatons is vastly different from one optimized against 10,000. There is, however, a problem with this assumption. The basic population data and weapon effects model is constructed on the basis of separate and independent cells. Only in this way can a defense be found to minimize the maximum number of fatalities

using the double Lagrange technique; hence, any assumptions made about the attack level must be in a separable form also. Instead of assuming an attack level, the double Lagrange technique assumes a lambda to determine the shelter allocation. This lambda is the attacker's marginal or minimum required fatalities per megaton. The attacker attacks each and every cell in an amount so that the marginal number of fatalities per megaton equals lambda. The double Lagrange method does insure that the defense obtained is best for whatever attack level results from the lambda used.

The program is more flexible than the strict double Lagrange technique. In order to obtain a defense which is good against a range of attack levels, the defender is allowed to assume a number of lambdas (DLAM's) and to assign to each a weight (WDLAM's). This corresponds to finding a defense which tries to be good against a number of attack levels, since each lambda corresponds to some attack level. Or the defense may choose a number of lambdas and try to minimize the fatalities at the worst lambda--the minimax selection of defense.

As to exactly how the lambdas are chosen--a small amount of trial and error with the aid of an evaluation or a bounding run provides sufficient information about what lambdas correspond to what attack levels, and the exact values of the lambdas appear not to be important. As for what mixed lambda combination (lambdas plus weights) or minimax combination results in good stabilized defenses, no simple procedure was found. It may be useful to experiment with small data first, and to apply a good combination to the large-scale data. Or perhaps a few trials with the large-scale data may find a sufficiently good combination. It was found that two lambdas for either the mixed lambda or minimax technique seemed to be sufficiently flexible and much easier to handle

than a larger number of lambdas.

One last statement about these assumptions--the justification for any defense is in the evaluation. Any method of finding a defense including writing down random numbers is valid if it gives desired results in the evaluation. All the above parameters with the exception of TCOST and IFILL need no justification besides that test, and hence one need not worry, for instance, if the attacker really will have a mixed attack objective, if that assumption gives good results against the various possibilities of attack.

B. EVALUATION OF DEFENSE

When a nationwide defense has been computed or entered into memory from cards, the next step is to evaluate the defense. For each of a number of attack levels (total delivered weapons in terms of one-megaton equivalents), the program finds the optimum attack against the nationwide defense and computes the payoff (combination of population and industry) and the population and industry killed. Most of the same parameters need to be set as for generation of the defense; however, the interpretation is different--they are now actual objectives attributed to the attacker. The selection of these objectives and parameters is independent of what was assumed by the defense.

The first parameter is the actual time of attack, which is set in IAPOP, but which differs from the previous use of IAPOP, which was only for the purposes of generating the defense. Only two realistic choices exist for actual time of attack: IAPOP = 1 for nighttime attack and IAPOP = 2 for daytime attack. In order to evaluate a defense, both times of attack are usually used in turn.

The second parameter is the attacker's objective--the relative weighting of population and industry killed. The attack objectives are defined by one or more values in

the arrays POPMULT and VINDMULT, each combination corresponding to one attack objective. Only the relative size of POPMULT and VINDMULT matters here, and not the absolute values.

Also, for another evaluation of a defense one might vary IFILL from optimal to fractional filling modes. (See the previous section.) Other evaluations might be made for softer shelters (via the PERTURB option), for variations in the definition of unsheltered population, or for different height of burst selections.

C. THE BOUNDING PROCEDURE*

For any size defense budget, we would be interested in knowing to what level the defense can hold fatalities. When this minimum is determined for a number of attack levels, the result can be called the optimal curve. It represents (for some set of parameters) the lowest level of fatalities the defender can achieve against an optimized attack of known total size. We cannot actually compute the optimal curve, but we can find upper and lower bounds for it, and these can be made as close together as desired for some expenditure of computation time. The method for the computation is described in Appendix A.

There are not many parameters to set for the procedure; cost, of course, is the important parameter. IDPOP and IAPOP need to be set; the actual time of attack will always be the same as the assumed time of attack. Any attack objective may be

*The bounding procedure described here was originally conceived as a method for checking solutions generated by the double Lagrange procedure to see if they were spurious. However for a defense which is to be stabilized rather than optimized for a single set of parameters, a solution is not spurious if it gives desired results. A standard of performance is still needed, and that standard is provided by the bounding procedure.

selected, but since the payoff to defender and attacker must be the same, the only attack objective which will indicate population fatalities is a pure population objective. All other parameters which need to be set have to do with program operation and have no larger significance.

II. USE OF THE PROGRAM

A. BASIC PROGRAM OPERATION

There is considerable freedom allowed in the operation of the program, as can be seen by considering Figure 1, a flow chart of the basic structure of the program. The option selector merely selects the operations to be performed; all input, output, and computation is performed in the option portions of the program.

Although all the options are shown in the flow chart having equal status, they vary drastically in how much they perform. Some options merely set a single constant and others may compute for an hour or more. As each option is completed, control returns to the option selector and the next option card is read. Normal termination is by reading a STOP card; however, the program will also terminate if an option card is read which the option selector does not recognize.

Because of the way the program is constructed, any sequence of options is allowed. After each option card, the data sequence for that option is read and that option is then performed. Hence the data deck structure is as shown in Figure 2: the basic data deck, followed by a sequence of option cards each with its data deck, terminated by a STOP card. The data sequences for each option may be large (see for example the shelter input deck, which is the INPUT option) or the particular option may not read any cards at all. The data sequences for all the options are specified in Section II D of this Appendix.

The typical mode of operation is as shown in Figure 3. This mode of operation is simply a special case of the pattern of Figure 2. The shelter input

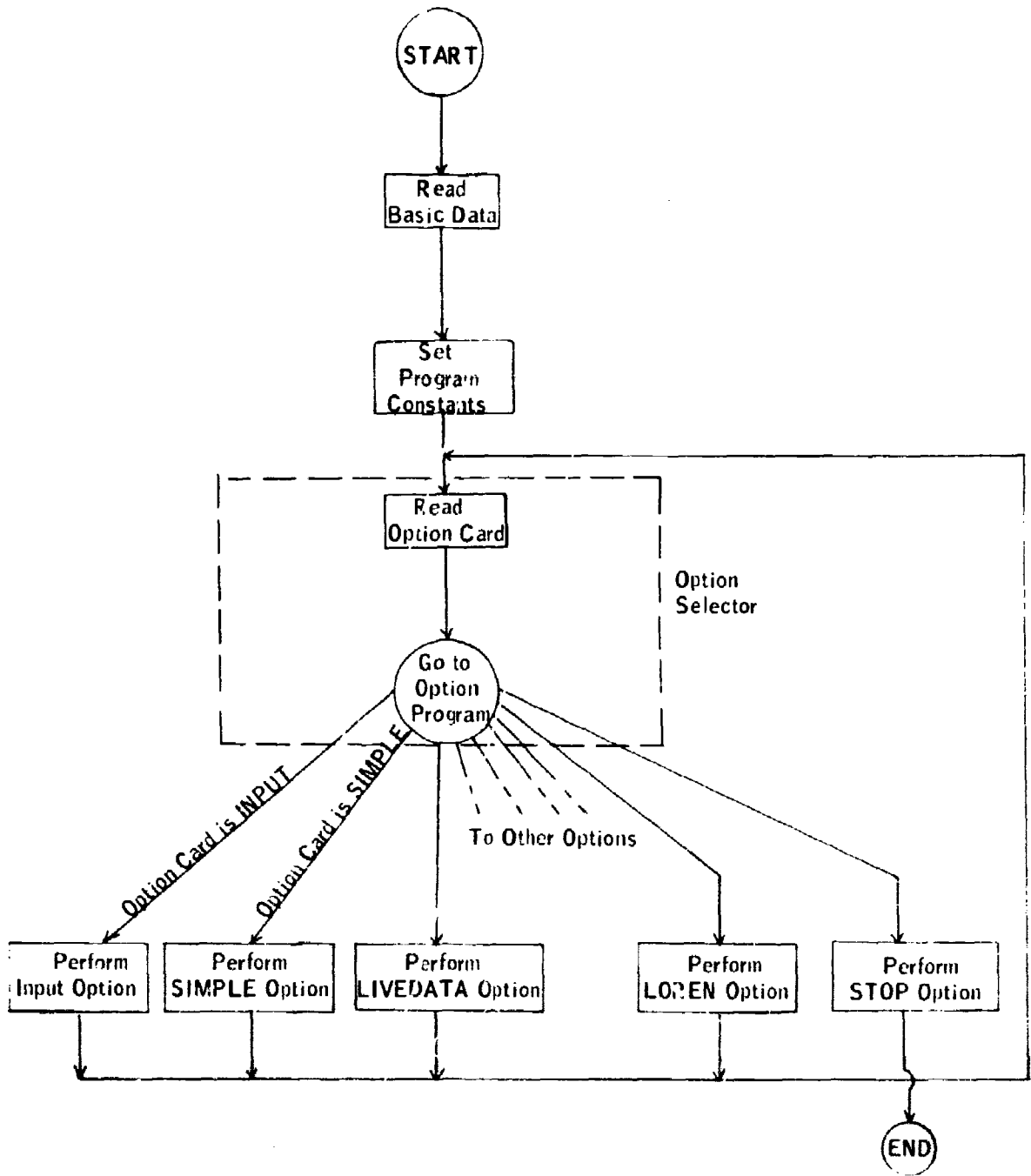


Figure 1. Flow Chart of Program BLAST

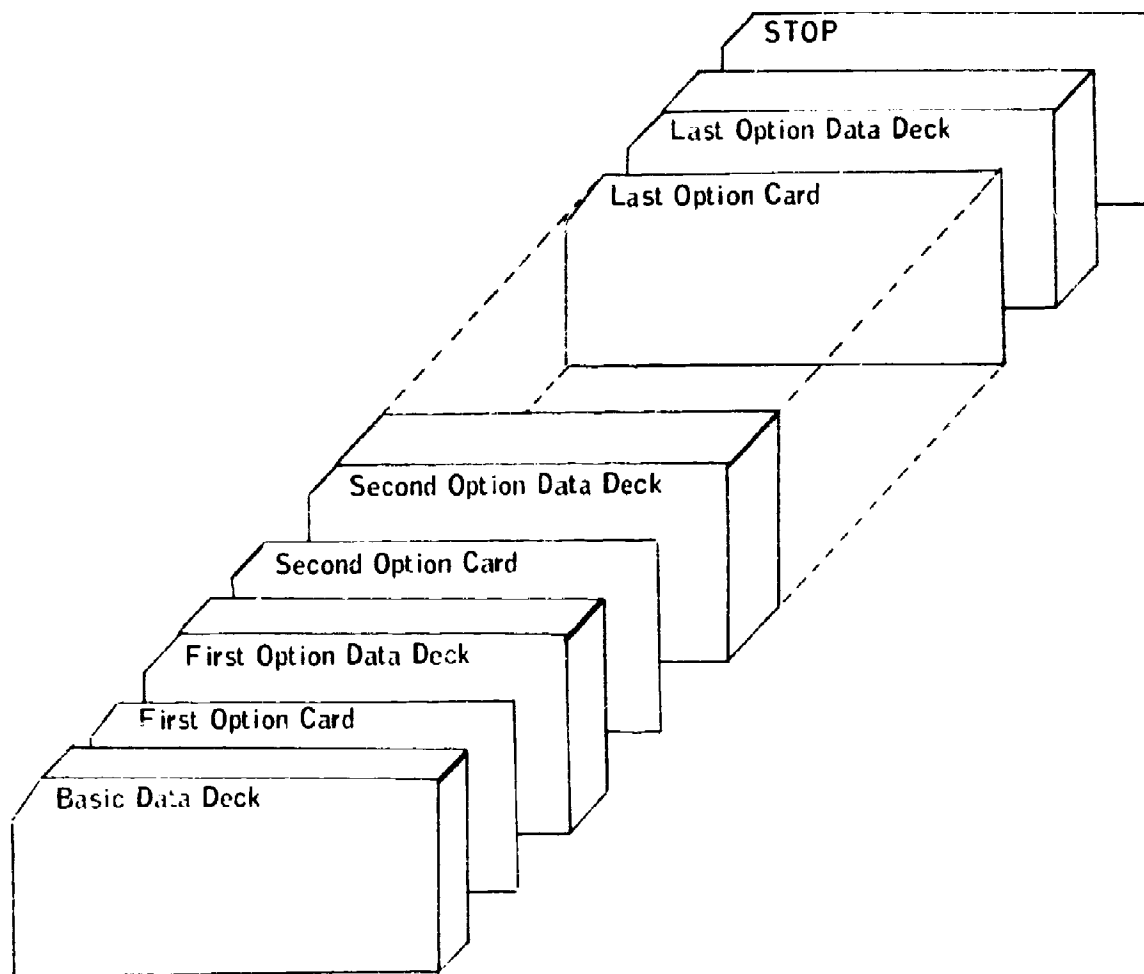


Figure 2. Data Deck Structure

deck is an **INPUT** card followed by an **INPUT** option data sequence (an example of a shelter input deck is given in the Sample Data Deck section). The run deck is a sequence of option cards, each followed by its own option data deck (examples of run decks are shown in the Sample Data Deck section). And, as always, the **STOP** card terminates the deck.

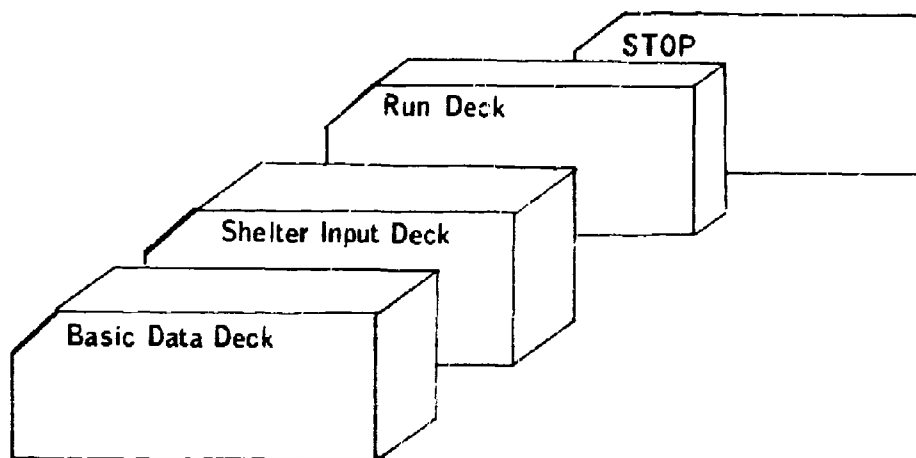


Figure 3. Typical Operation Sequence

As can be seen, it is the options and the option cards which control whether the program performs large computations or small (or even no) computations. Hence the next two sections define what the options do and what data is required. The point of view will be strictly as a user; the subroutines are discussed in Section IV.

B. CATALOG OF RUN OPTIONS

1. Primary Computation Options

Run Type SIMPLE

This is the basic option of the program -- it allows one to generate a defense and to evaluate it. There are many parameters to be set before entering **SIMPLE** where the defense is generated. Then the defense will be generated in **SIMPLE** on the basis of whatever values were chosen for the parameters. If the program is directed to go ahead and evaluate the defense, the same parameters will be used for the evaluation. If the defense is not evaluated immediately, after

generating the defense in the SIMPLE option control returns to the option selector, where the parameters may be reset and an evaluation performed.

Parameters which are to be set before entering SIMPLE are the cell population data, the population bases IDPOP and IAPOP, the shelter data, the shelter filling mode (IFILL), and the lethal areas, heights of burst, the attack weapon density quanta^{*}, and the attack levels for the evaluation portion of SIMPLE. The population data is set in the basic data deck or in run type LIVEDATA; run type OPTIONS is used to set IDPOP, IAPOP, and IFILL; the shelter data, lethal areas, and heights of burst are set in the basic data deck or in run type INPUT; and the attack density quanta and attack levels are set in the basic data deck. In addition, two decisions concerning the generation of the defense need to be made before entering SIMPLE--first, if a punch output of the defense is desired option PUNCH should be entered, and second, if minmax lambda stabilization is desired option MINIMAX should be entered.

In run type SIMPLE, the following parameters are set explicitly:

TCOST	Shelter program cost
NDLAM	Number of defense lambdas
DLAM's	Lambdas assumed by the defense
NAOB	Number of attack objectives
POPMULT's	Population multipliers of the attack objectives
VINDMULT's	Industry multipliers of the attack objectives

*Recall that in the defense optimization every defense is attacked optimally; hence these attack parameters must be set.

For description of precisely how the above parameters may be input see the Data Sequence Definition section. Since no weights on the attack objectives (WDAOB's) have been read, the program uses whatever has been stored previously. (As set initially in the program constants, WDAOB places the entire weight for the defense on the first attack objective; i.e., the defense assumes the first attack objective unless otherwise set -- subsequently in SIMPLE or in run type ATTOBJ.)

At this point the program will generate the defense and, if it has been called for by option PUNCH, it will retain the defense on a punched card deck.

The defense is now fixed and fully defined and may, for instance, be plotted on maps. The defense is defined by one of the 15 defense options for every cell plus the split cell information for the one cell which must be split to achieve exact cost closing, together with IDPOP which defines the population basis to determine the number of shelters in each cell. The chosen defenses are stored in the memory in packed form along with the cell population data. The next step is to evaluate the defense.

The next card to be read by option SIMPLE determines whether the defense will be evaluated immediately or control will be returned to the option selector. If control is returned to the option selector then the defense which is defined in memory may be evaluated later using option EVAL, or it may be analyzed using OUTPUT, or used for time phasing studies in TIMEFAZE, or the run may simply be terminated. If the defense is to be evaluated immediately, the program will evaluate the defense for each of the NAOB attack objectives, printing the result after each evaluation.

The evaluation consists of optimizing the offense using a one-sided Lagrange multiplier maximization, adjusting the lambdas in the maximization until sufficiently accurate calculation is made of the fatalities for each of several attack levels. The offense parameters required are the density quanta, the attack levels, and the attack objectives, and these have already been set.

After each evaluation a print like Figure 4 is made, identifying the run recording the parameters and giving the results of the evaluation. The first line of Figure 4 identifies this as a run type SIMPLE. The key number in the next line is assigned during the time the punched card deck is being made. The same number is assigned to both the printout and the punch deck so that there is no confusion about matching the two*.

The next block defines the parameters used to determine the defense--the population bases IDPOP and IAPOP, the shelter filling mode, the cost for the shelter program (TCOST), and the mu found which achieves that cost. Next the lambdas assumed by the defense are given together with either the weights for the mixed lambda case or the statement that minmax stabilization was used. The last parameter of the defense is a listing of all attack objectives which have nonzero weights (WDAOB), i.e., all attack objectives considered by the defense.

The next block defines the parameters used by the offense--the attack density quanta and the attack objective used by the offense for this evaluation. Next the payoffs or fatalities are given for each attack level. The attack level is given in equivalent megatons. The payoff killed is the fraction of the total

* The number assigned is the first five digits of the fraction of the split cell which is assigned to the primary defense for that cell. This number is unique to each run.

payoff killed by the attacker, or the sum of the population killed and the industry killed, weighted by the population and industry multipliers used by the offense.

THIS DEFENSE HAS BEEN ASSIGNED KEY NUMBER 7729

DEFENSE SET USING THESE PARAMETERS**

POPULATION BASIS FOR SHELTERS IS IDPOP = 1 -- RES. POPULATION
 ACTUAL POPULATION AT ATTACK IS IAPOP = 2 -- DAY POPULATION
 SHELTER FILLING MODE IS IFILL = 1 -- OPTIMAL
 TARGET COST FOR SHELTER PROGRAM IS 15.00 BILLION
 DEFENSE ARRIVED AT TARGET COST USING MU = 0.0026485
 DEFENSE WEIGHTS ATTACKER'S LAMBDA'S AS FOLLOWS.
 LAMBDA'S = 40000 2500
 WEIGHTS = 0.6500 0.3500
 DEFENSE WEIGHTS ATTACKER'S OBJECTIVES AS FOLLOWS
 POPMULT INDMULT WEIGHTS
 1.000 0.000 1.000

OFFENSE SET USING THESE PARAMETERS**

11 ATTACK DENSITY LEVELS ARE 0.00 0.02 0.05 0.10 0.25 0.50
 1.00 2.00 4.00 8.00 16.00
 ATTACK OBJECTIVE IS IAOB = 1 -- POPMULT = 1.000, INDMULT = 0.000

PAYOFFS (IN MILLIONS) FOR PROGRAM COST 15.00 BILLION

ATTACK LEVEL	PAYOFF KILLED	PEOPLE KILLED	INDUSTRY KILLED	LAMPD:
20.0	0.10	21.34	16.29	412069.60
50.0	0.14	29.67	27.10	203491.80
100.0	0.18	37.17	36.73	115695.37
200.0	0.22	45.91	46.56	71600.26
500.0	0.29	61.26	58.60	43462.83
1000.0	0.39	81.48	66.10	35072.80
2000.0	0.52	108.70	75.34	19355.03
5000.0	0.66	141.95	85.68	7518.00
10000.0	0.80	167.07	89.84	3354.71

TAIL CELLS DEFENSE AND ATTACK

CELL NO.	DEFENSE	ATTACK LEVELS									
3123	4	1	1	1	2	2	2	2	4	5	
3124	4	1	1	1	1	2	2	2	4	5	
3125	4	1	1	1	1	2	2	2	3	5	
3126	4	1	1	1	1	1	1	2	2	4	
3127	2	1	1	1	1	1	2	2	3	3	
3128	1	1	1	1	1	1	2	2	3	3	
3129	1	1	1	1	1	1	1	2	2	3	
3130	1	1	1	1	1	1	1	2	2	3	
3131	1	1	1	1	1	1	1	1	2	3	
3132	1	1	1	1	1	1	1	1	2	2	

Figure 4. Sample of SIMPLE Output

In this case since the attack objective is pure population, the payoff killed is the fraction of total population killed. The population killed is in millions for a total of some 209,000,000; the industry killed is in billions of dollars of manufacturing value added for a total of \$97,000,000,000.

The lambda column gives the lambdas found for each attack level in the evaluation with no scale factor. This gives the marginal return in payoff for an additional megaton of attack. Thus, at attack level of 5000 megatons an additional megaton adds approximately 7500 fatalities.

The final block of the SIMPLE print indicates what happens in the tail cells--the low density aggregated cells. For each cell the defense option is given together with the attack quantum for each attack level considered in the evaluation. For example, in the figure, tail cell number 3128 was undefended (defense option 1) and for attack level 5000 it received .05 megatons per square mile (attack quantum 3).

After making this print, the evaluation is repeated for the remaining attack objectives, and a similar printout is made for each. After that a card is read to see if new WDAOB's are desired. If so, then the WDAOB's are read, and a new defense is generated and evaluated for all attack objectives. Because of running time, this last option was not used on the large-scale data runs in this study. When option SIMPLE terminates, control returns to the option selector.

There are no error exits or messages for option SIMPLE; only two abnormalities may occur: if the defense is unable to spend all its target cost for

the initial mu's, then it saturates at some cost and evaluates the defense for that cost. The achieved cost is printed along with a statement that the defense saturated in the block of the SIMPLE print which defines the defense parameters. Secondly, if in the evaluation the offense is not able to achieve the highest attack levels with lambdas greater than 1.0, then the number of attack levels is reduced and the highest attack level is dropped. No message occurs, but this truncation is obvious in the printout.

When the SIMPLE option card is read, control is transferred to a portion of a driver program called OVLYDAVE, the first overlay of the program. At that point subroutines in the main portion of the program and in the first overlay are available for use. After the data is read in OVLYDAVE, the program then calls DEFOPTG to generate the defense, DEFPUNCH if the punch output is desired, EVAL to perform the evaluations, and ANSPRIN to print the results of each evaluation. For discussion of the methodology of these subroutines refer to the section on each subroutine.

Run Type EVAL

This is the option which evaluates any defense stored in the cell data in memory, no matter whether computed or input. This enables one to evaluate a defense for new attack objectives, time of attack, or filling mode.

EVAL itself reads no data and sets no parameters. The entire defense, all objectives, and all other parameters must be set before entering option EVAL. (Because of program construction these parameters will always be set to some value, but they

must be set to desired values). The following parameters must be properly set before EVAL:

1. The cell population data together with the defense to be evaluated (JDEF defined for every cell)
2. The shelter data, lethal areas, heights of burst
3. IDPOP is presumably already set to the proper basis
4. Shelter filling mode (IFILL)
5. Attack weapon density quanta
6. Attack levels
7. Time of attack (IAPOP)
8. Attack objective (POPMULT, VINDMULT, and IA0B)

With the above parameters set, when the program reads the EVAL option card, optimized defenses will be generated for each attack level, and the result will be printed out. As in SIMPLE, the procedure is the Lagrange multiplier method. When the defense is evaluated, the result is printed in the same form as the SIMPLE output (Figure 4), except that the key number is not assigned. The parameters for the defense are whatever remain for those variables, and these may or may not be meaningful. The time of attack (IAPOP) is indicated in the second line of the first block. The offense parameters are those which have been used for the optimization, and the results are printed as before. Option EVAL terminates after making the printout, and control returns to the option selector.

There are no error exits or messages from option EVAL; as in SIMPLE, if the attack cannot achieve the highest attack level, that attack level is dropped. The

EVAL option is performed from OVLYDAVE and uses only two subroutines, EVAL and ANSPRIN.

Run Type BALDEF

This option enables one to compute and evaluate "balanced" defenses for given cost levels. The procedure uses both an upper hardness limitation and a lower hardness cutoff*.

Before entering BALDEF one should set the population data, IDPOP, and IAPOP; shelter data may be entered in option BALDEF or previously. The data sequence section indicates the parameters to be set to define the balanced defense shelter options; the variables have the same meaning as in the INPUT sequence. After the initial shelter data are set, one reads in the costs and attack levels to be evaluated. The final parameters are PSIMIN, the lower hardness cutoff, PSIMAX, the upper hardness cutoff, and WMIN, the lethal area corresponding to WMIN.

All shelter cost data is computed in subroutine BDCELL using an analytic approximation to the cost data, and hence any changes must be made to the function. The function used for FINAL shelters is:

$$\text{Cost per person} = 112 \text{ and } 13.5 \sqrt{\text{PSI}} ;$$

the degree of fit is shown in the section on balanced defense.

* Lambda Research Memorandum 2 indicates the effect of cutoffs and has a general discussion of balanced defense.

The output after the evaluation has the same format as the SIMPLE printout, and all the quantities have the same meaning. The program then prints the computed lambda value; this lambda gives the hardness of any cell according to the equation

$$\text{Lethal area} = \frac{\lambda}{\text{Population Density}} .$$

After the balanced defense is computed, it can be evaluated changing any parameters desired. The exception is that the filling mode must remain at the artificial value of IFILL = 4.

This option is performed in OVLYDAVE in subroutine BALDEF2, and the evaluation is performed in EVAL.

Run Type BOUND

This option, which computes upper and lower bounds for the optimal curve as a function of attack levels, is the implementation of the boundary procedure described in the methodology section of the main report. For any specified cost the bounds can be generated as close to each other as desired; the payment for the accuracy is in computation time.

Before entering BOUND, the two population bases IDPOP and IAPOP should be set. Here the actual attack time is the same as the assumed attack time. The attack objective may be set, but pure population is the only attack objective which gives the answer in population fatalities; the attack objective will already normally be set to pure population.

There are several modes of operation available with run type BOUND; however, only three were found at all useful; the others are described briefly in the data sequence

definition section. All the modes require the following data:

- MODESW This is the switch which selects the mode of operation.
- FMUS 20 mus are used for the boundary procedure. These mus should be chosen carefully so that resultant costs are close to the desired budget level.
- TCOST This is the target cost for deployments.
- HSAT This is the total possible payoff (for pure population objective, it is total population).

One of the keys to getting good accuracy from this procedure is to choose the mus properly. If most of the 20 mus can be selected in a region where they will give costs close to TCOST, then the cost closing will be quite good and the bounds will be good. As many as possible should be concentrated in the region of interest; however, some mus should be available at extreme values so the cost will always be bracketed. Previous bound runs and defenses generated provide information for the mu selection.

With the above parameters required for all modes, we now discuss them individually.

Mode 1. This mode finds upper and lower bounds for the optimal curve by sweeping through all lambdas, starting from some point in the middle for efficiency. The procedure continues until it has swept through all lambdas from 10.0 to 1,000,000.0. Two additional parameters are required:

- DLAM(1) - This is the initial lambda used to compute the initial point and begin the sweep.
- EPS - This is a parameter controlling the separation between the lambdas chosen, and hence controls computation time and accuracy. Approximately, it is the distance between upper and lower bounds.

After reading the data, the program computes the initial point from the initial lambda. It then computes 20 lambdas and does one pass through all the cells, finding solutions of

the Lagrangian those lambdas with defenses of cost approximately TCOST. The lower and upper bound theorems are applied to the solutions, and the bounds are computed. A new set of lambdas is chosen to extend the range, and the process continues until the bounds are generated.

Figure 5 shows the initial point used for a bound run, the payoff, weapon expenditure, cost, lambda, and mu. Figure 6 shows the output after the second pass through the cells. The 20 lambdas are shown, together with the mus giving the closest cost, and the

```

INITIAL POINT IS--
BND DAT
-----
      BKILL      RATE      GCOST      DLAM      BMU
-----
1  0.901+017  1759.35  1.570+010  2.000+004  2.591-003
2  0.901+017  1759.35  1.570+010  2.000+004  2.591-003
3  0.901+017  1759.35  1.570+010  2.000+004  2.591-003
-----
      MSAT      EPS      ULSAVE      WILAS
-----
2.095+008  2.000+006      32  1.000+006
-----
      MODESW      NERU      RUTV      ALEVMAX
-----
1              0              0              0.00
-----
      FMAX      RMIN
-----

```

Figure 5. Output for Run Type BOUND

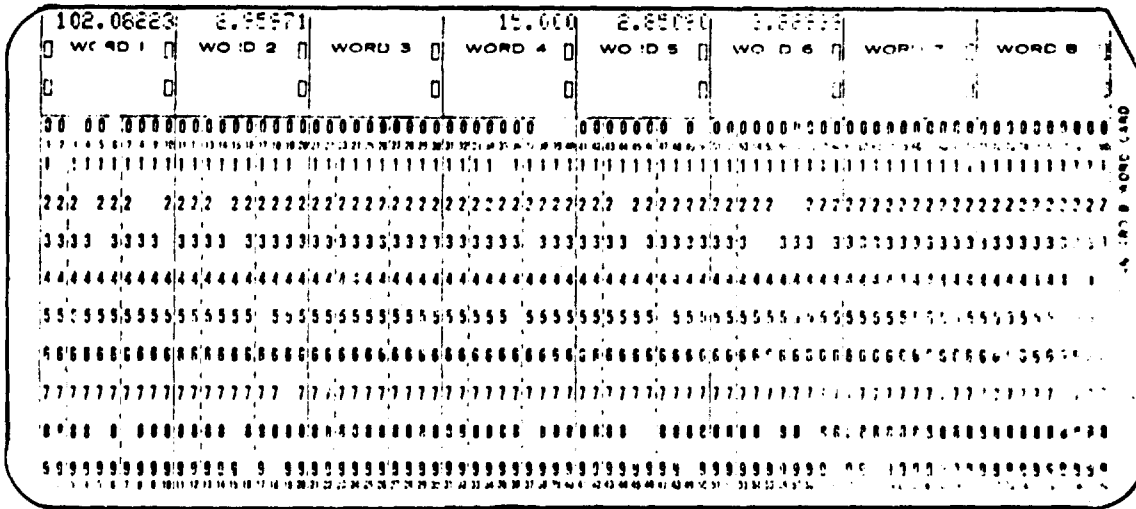
```

      JCH02      DLAM      FMAX      CUMKILL      CUMWEP      CUMCOST      TCOST      VNTCEPT      SATWEP
-----
1  13  2.851+003  2.730+003  1.297+008  1.125+004  1.806+010  1.500+010  1.040+008  3.628+004
2  13  3.220+003  2.730+003  1.318+008  9.019+003  1.571+010  1.500+010  1.021+008  3.326+004
3  13  3.514+003  2.730+003  1.257+008  8.759+003  1.664+010  1.500+010  9.849+007  3.071+004
4  13  4.006+003  2.730+003  1.286+008  8.949+003  1.578+010  1.500+010  9.491+007  2.858+004
5  13  4.410+003  2.730+003  1.279+008  8.444+003  1.538+010  1.500+010  9.149+007  2.675+004
6  13  4.521+003  2.730+003  1.215+008  7.729+003  1.645+010  1.500+010  8.817+007  2.516+004
7  13  5.241+003  2.730+003  1.257+008  8.057+003  1.549+010  1.500+010  8.478+007  2.379+004
8  13  5.670+003  2.730+003  1.204+008  6.798+003  1.474+010  1.500+010  8.165+007  2.254+004
9  13  6.104+003  2.730+003  1.104+008  6.753+003  1.515+010  1.500+010  7.866+007  2.141+004
10 13  6.556+003  2.730+003  1.119+008  6.371+003  1.500+010  1.500+010  7.584+007  2.038+004
11  0  3.769+004  3.190+004  0.000+000  0.000+000  9.320+007  1.500+010  7.283+007  1.948+004
12 -1  0.550+003  3.190+004  0.000+000  0.000+000  1.637+008  1.500+010  1.529+007  5.555+003
13  8  3.789+004  2.210+003  4.448+007  8.415+007  1.500+010  1.500+010  1.295+007  5.214+003
14  8  4.145+004  2.210+003  3.723+007  6.296+002  1.443+010  1.500+010  1.094+007  4.978+003
15  8  4.329+004  2.210+003  3.873+007  5.999+002  1.336+010  1.500+010  9.226+006  4.630+003
16  7  4.609+004  2.090+003  3.042+007  5.411+002  1.600+010  1.500+010  7.678+006  4.379+003
17  7  4.695+004  2.050+003  3.176+007  5.368+002  1.532+010  1.500+010  6.139+006  4.154+003
18  7  5.129+004  2.090+003  3.244+007  5.285+002  1.443+010  1.500+010  4.597+006  3.951+003
19  6  5.477+004  1.870+003  2.178+007  3.429+002  1.528+010  1.500+010  3.483+006  3.748+003
20  6  5.770+004  1.870+003  2.146+007  3.225+002  1.442+010  1.500+010  2.505+006  3.543+003
21  5  6.077+004  1.500+003  1.738+007  2.530+002  1.474+010  1.500+010  1.697+006  3.410+003
22  4  6.384+004  1.070+003  1.192+007  1.795+002  1.505+010  1.500+010  1.049+006  3.245+003

```

Figure 6. Lagrangian Solutions for Pass Number 2

payoff, weapon expenditure, cost, TCOST, and for each bounding line the intercept and saturation point (H^0 and A^0 in the methodology section). After each pass the bounding lines are computed and the lower bound lines are punched onto cards. For example, for lambdas number 1 and 2 in Figure 6, the following card was punched:



The first number is the intercept of the lower bound line for zero attack level (in millions); the second is the slope of the line (in thousands). The third number is TCOST, the fourth and fifth are the two lambdas used (in thousands). Now, this bounding line is the lower bound for points on the optimal curve with slope between the two lambdas; the entire lower bound is the lower envelope of all such lines. We will use this fact when it comes to fill in lambdas (mode 5). And lastly, the upper and lower bounds generated (values in millions) are shown in Figure 7 .

The only error return occurs when the defense saturates in the computation of the initial point. An error message is printed and the run terminates.

FOR TCOST = 15.00 BILLION, IDPOP = 2, IAPOP = 2

ATTACK LEVEL	LOWER BOUND	UPPER BOUND
20,0	1.33	1.57
50,0	3.25	3.83
100,0	5.43	7.48
200,0	12.87	13.87
500,0	29.26	30.54
1000,0	43.23	49.52
2000,0	70.60	72.81
5000,0	103.13	107.58
10000,0	131.68	134.39
20000,0	150.89	153.83
50000,0	168.92	171.91
100000,0	177.42	182.76

Figure 7. Upper and Lower Bounds for Lagrangian Solutions

Mode 2 is exactly the same as Mode 1, except that the initial point is read in, not computed. This point is used only for lambda selection and not in the bound computation itself, so if a Lagrangian solution is approximately known, it would be useful to use this mode. The following additional parameters are required:

- BKILL(2) - Initial payoff
- BWEP(2) - Initial weapon expenditure
- BCOST(2) - Initial cost (this replaces TCOST)
- BLAM(2) - Initial lambda (this replaces DLAM(1))
- BMU(2) - Initial mu

Outputs are exactly as in Mode 1.

Mode 5. It may often be useful to specify lambdas for the bound generation, rather than having them computed. This mode performs one pass with lambdas which are input and computes the bound from solutions from those lambdas.

The chief use for this mode is to improve the accuracy of a Mode 1 or Mode 2 run. To do this, suppose that for some portion of a run, the error is unacceptably high. The

procedure is as follows:

1. Select two lambdas from the previous run which define the inaccurate section.
2. Select 18 lambdas spread between those two limits. The 20 lambdas thus found will be the lambdas (DLAM's) for the new run.
3. From the printout of the mus chosen for lambdas in the bracket, select 20 new mus. It should be possible to have very good cost matching for these closely chosen mus.
4. Perform mode 5 run
5. Upper bound at each point is now the lower of the upper bounds from the two mus.
6. To find the lower bound, recall that we must have bounding lines which represent the entire lambda range. Thus, we discard those output cards from the first run which cover the lambda range between the two limit lambdas chosen above, and we substitute the output cards from the second run (which cover the same range). The composite deck still covers the entire range, but the accuracy will be improved*. The lower bound is then the lower envelope of the bounding lines.

The bounding procedure is performed in OVLAY3 with subroutine PPEBND as a monitor and input program. The lambda selection is done in PICKLAM, the Lagrangian solutions are found in DEFOPTG, and the bounds are computed in BNDSET.

Run Type STATCOST

This option implements the generation of defenses under the constraint of allocating the defense budget to states by population and optimizing each state individually for its portion of the budget. All the same parameters are set in option STATCOST as in SIMPLE; in fact, the two data sequences are identical (they are read from the same part of the program). The only difference is that instead of a single optimization in DEFOPTG in the SIMPLE option, the STATCOST option does the optimization in subroutine

* It is possible to use all Lagrangian solutions in the critical range to find new bounding lines using the theorem, but the above procedure is simpler and quite adequate.

STATCOST; however, subroutine STATCOST also does evaluations, and that necessitates changes to the run deck if extra evaluations are not to be performed. First, however, we will see what STATCOST does.

The data input for STATCOST is precisely the same as option SIMPLE-- all the variables have the same interpretation with the exception that TCOST is now to be divided among the states and each state optimized individually. Figure 8 displays the first portion of the STATCOST print, and it shows, for instance, that Connecticut's share of the total budget is \$160 million. Connecticut, and every other state, is then optimized in DEFOPTG for the allotted amount of money. For each state the optimization is performed using those cells which have been explicitly stored in memory (and hence meet the population density requirement).

Now for the mus used in the program, it may not be possible to spend the entire amount allocated to a state on the cells which represent that state*. This is indicated in Maine, for instance. When that happens, a message is printed, and the defense is set for whatever amount can be spent. This means that there will be some money unspent out of the initial budgeted amount. What can be, and has been done, is to redo the calculation using a slightly higher initial cost to bring the total amount spent to the desired amount. This added amount to be budgeted can be found as the unspent money multiplied by the ratio of the initial total budget to the money allocated to the unsaturated states. This will then give the proper amount spent on the second pass if no new states saturate. As shown in the figure, it took a budget of 10.38 billion to achieve 10 billion spent.

* This saturation is with respect to the mus and in consideration of the attack which is assumed; i.e., if a state is attacked very lightly even without shelters, it may not be possible for it to spend its budgeted amount.

TOTAL BUDGET OF \$ 10.38 BILLION ASSIGNED TO STATES AS FOLLOWS

CONN	--	\$ 100.0 MILLION		
MAINE	--	\$ 52.9 MILLION	SATURATED--CANNOT SPEND MORE THAN \$	42.8 MILLION
MASS	--	\$ 299.2 MILLION		
NEW HAM	--	\$ 34.0 MILLION	SATURATED--CANNOT SPEND MORE THAN \$	34.5 MILLION
N J	--	\$ 380.3 MILLION		
N Y	--	\$ 904.4 MILLION		
R I	--	\$ 46.5 MILLION		
VER	--	\$ 21.0 MILLION	SATURATED--CANNOT SPEND MORE THAN \$	11.3 MILLION
DEL	--	\$ 72.8 MILLION		
D C	--	\$ 34.6 MILLION		
NY	--	\$ 160.0 MILLION		
MD	--	\$ 190.2 MILLION		
OHIO	--	\$ 252.7 MILLION		
PENN	--	\$ 607.8 MILLION		
VA	--	\$ 439.0 MILLION		
MT VA	--	\$ 62.1 MILLION	SATURATED--CANNOT SPEND MORE THAN \$	52.3 MILLION
ALA	--	\$ 172.5 MILLION		
FLA	--	\$ 369.3 MILLION		
GA	--	\$ 216.4 MILLION		
MISS	--	\$ 114.1 MILLION	SATURATED--CANNOT SPEND MORE THAN \$	40.5 MILLION
N CAR	--	\$ 244.4 MILLION	SATURATED--CANNOT SPEND MORE THAN \$	178.9 MILLION
S CAR	--	\$ 125.3 MILLION	SATURATED--CANNOT SPEND MORE THAN \$	59.2 MILLION
TENN	--	\$ 145.4 MILLION		
ILL	--	\$ 397.4 MILLION		
IND	--	\$ 241.2 MILLION		
MICH	--	\$ 450.8 MILLION		
MINN	--	\$ 192.0 MILLION		
WIS	--	\$ 223.7 MILLION		
ARK	--	\$ 86.7 MILLION	SATURATED--CANNOT SPEND MORE THAN \$	59.8 MILLION
LA	--	\$ 193.1 MILLION		
NEW MEX	--	\$ 63.0 MILLION		
OKLA	--	\$ 127.3 MILLION		
TEXAS	--	\$ 364.8 MILLION		
COL	--	\$ 106.1 MILLION		
IDA	--	\$ 140.0 MILLION	SATURATED--CANNOT SPEND MORE THAN \$	125.9 MILLION
KAN	--	\$ 116.4 MILLION		
MO	--	\$ 225.1 MILLION		
NEB	--	\$ 72.0 MILLION		
N DAK	--	\$ 31.0 MILLION	SATURATED--CANNOT SPEND MORE THAN \$	15.2 MILLION
S DAK	--	\$ 33.1 MILLION	SATURATED--CANNOT SPEND MORE THAN \$	6.1 MILLION
WYO	--	\$ 19.8 MILLION	SATURATED--CANNOT SPEND MORE THAN \$	11.9 MILLION
ARIZ	--	\$ 194.8 MILLION		
CAL	--	\$ 1073.0 MILLION		
HAWAII	--	\$ 41.2 MILLION		

Figure 8. Sample of STATCOST Output

After finding the defenses for the 51 states, subroutine STATCOST also punches out the defense (with no tail cells) and then evaluates the defenses two ways, both with IAOB = 1. The first way is with no tail cells; i.e., an evaluation of the cells stored in memory. The second way is with the tail cells included, but undefended. This second way is what is useful for comparison with unconstrained defenses. Answers are printed for both evaluations in exactly the same form as the SIMPLE output. After this, the evaluations normally performed in SIMPLE will be performed if the NOEVAL card is not used. Also, an additional punch deck may be obtained with the PUNCH option.

Because of the way the evaluations are performed, certain modifications must be made to the run deck if only pure population evaluations are desired. A sample is as follows:

```

LIVEDATA          10.375653
OPTIONS           2
LAPOP             2    78850.0    7885.0
ATTOBJ           2
                2    0.7    0.3
1.0              0.5           2
0.0              1.077        1.0    0.5
0.0              1.0          0.0    1.077
PUNCH            NOEVAL
STATCOST         STOP

```

This sequence will generate the defense, punch two output decks (one with tail cells, one without), and do two evaluations (one with tail cells, one without). There are no error exits or messages in STATCOST. STATCOST is executed in OVLYDAVE, and after reading data, subroutine STATCOST is called, which then makes use of DEFOPTG, DEFPUNCH, EVAL, and ANSPRIN.

Run Type TIMEFAZE

This option enables one to find a defense which is time-phased with a larger defense. Before entering TIMEFAZE the larger defense should be entered into memory by computation or by DEFSPEC. Calling option TIMEFAZE then sets that old defense up for time-phasing. Next the routine determines which combinations of defenses are excluded, as is shown in Figure 9 (the 1's are excluded combinations). To find the new defense one then uses option SIMPLE in the normal manner; those defenses excluded in the table will not be allowed when the new defense is optimized. This routine requires no data; it is executed in subroutine TIMEFAZE, which prepares DCONSTR for proper functioning.

TABLE OF EXCLUDED DEFENSES

FINAL DEFENSE	INTERMEDIATE DEFENSE														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1
4	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1
5	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1
6	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1
7	0	1	0	0	1	1	0	2	1	1	1	1	1	1	1
8	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1
9	0	1	0	1	1	1	1	1	0	1	1	1	1	1	1
10	0	1	1	0	1	1	1	1	1	0	1	1	1	1	1
11	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1
12	0	0	1	0	1	0	1	1	1	1	0	1	1	1	1
13	0	0	1	1	1	1	1	1	1	0	1	0	1	1	1
14	0	0	1	1	1	1	1	1	1	1	1	1	1	0	1
15	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Figure 9. Output for Run Type TIMEFAZE

Run Type UNSHEL

This is the option which enables one to compute unsheltered population fatalities (or in fallout shelters only, or whatever is used for defense option number 1). This option is merely a special case of the EVAL option, and almost all parameters which are set there for the attacker should be set for UNSHEL (the exception is IFILL, which, of course, has no significance here.) No data is required in UNSHEL; as in EVAL, all parameters must be set previously.

The procedure used for UNSHEL is to set all cells to JDEF = 1 (normally unsheltered defense option) and to evaluate in subroutine EVAL. The print (from ANSPRIN) is exactly as the SIMPLE printout, except that much of the information does not apply. Then a correlation printout is given which is like that in option OUTPUT, and option UNSHEL terminates.

2. Input and Parameter Options

Run Type ALERT

This option enables use of the alternate filling mode $IFILL = 2$. This mode enables one to specify a fraction of the population alerted. Only one parameter is read-- $FALERT$, the fraction alerted. Then the $IFILL$ parameter is set to 2.

Run Type ATTOBJ

This option enables one to change the attack objective. Four variables are set-- $NAOB$, the number of attack objectives; $POPMULT$'s, the population multipliers of the different objectives; $VINDMULT$'s, the industry multipliers; and $WDAOB$'s, the weight the defense assigns to the attack objectives.

Run Type DEFSPEC

This option, for all its input data sequence complexity, is really just an input procedure enabling one to specify a defense for evaluation or analysis. An input deck may be made up according to the definition in the next section for any defense desired whatever; however, the primary method of use is for reading in the defenses saved on cards via the $PUNCH$ option.

The input sequence for $DEFSPEC$ includes almost all parameters needed to define a defense and evaluate it. One that is not defined is $IAOB$, the index of the attack objective. It may be set to a desired value before using $DEFSPEC$; if not set, it normally has a value of 1.

The last card of the input sequence gives one a choice of either evaluating the defense immediately using all parameters as set or not evaluating. In the latter case one may evaluate after changing parameters, or one may have other purposes. In either case,

the printout has the same form as the ones in INPUT and SIMPLE, and the purpose is to present the parameters for the defense.

Previous to using DEFSPEC the cell data will presumably have been entered. If the number of cells in the data is not equal to the number defined in the defense, an error message is printed, but the program continues. If the number of defenses is too few, all remaining cells are set to unsheltered posture.

DEFSPEC is executed in OVLYDAVE using subroutine DEFSPEC and, if evaluation is desired, subroutine EVAL.

Run Type INPUT

This option is the primary means of entering shelter data into the program. It defines lethal areas, shelter data, and defense options, as described in the data sequence section and the sample deck section.

The printout which indicates the data that has been entered is in Figure 10. This output corresponds to the data in the sample shelter input deck for the FINAL shelters. FINAL-2 is simply a name for the defense data, as are NFSS, 10 PSI, 30 PSI, etc. names for the shelters defined in the data.

INPUT is executed from OVLYDAVE and is performed in subroutine DEFPRNT, which uses SHELINP.

Run Type LIVEDATA

This is the option which enters the population-industrial data for the 2 sq. mi. cells into memory. The data is read from a data tape whose format will be described. No parameters need be set before using LIVEDATA; they may be set before or after. However, option LIVEDATA destroys any defense stored in memory.

DEFENSE CODE FINAL-2

LISTING OF SHELTER TYPES

SHELTER TYPE	COST	SPREAD OF PSI LEVELS									
NFSS	0.0	5.4	10.0	15.0	19.5	24.0	28.5	33.0	37.5	42.0	46.5
10 PSI	141.3	0.17	0.33	0.50	0.67	0.83	1.00	1.17	1.33	1.50	1.67
30 PSI	194.3	0.33	0.50	0.67	0.83	1.00	1.17	1.33	1.50	1.67	1.83
100 PSI	278.3	0.50	0.67	0.83	1.00	1.17	1.33	1.50	1.67	1.83	2.00
300 PSI	381.0	0.67	0.83	1.00	1.17	1.33	1.50	1.67	1.83	2.00	2.17

LISTING OF POSSIBLE SHELTER ALLOCATIONS

OFFENSE OPTION	FRACTION OF POPULATION SUPPLIED EACH SHELTER TYPE												
1	ENTIRE POPULATION SUPPLIED SHELTER TYPE NFSS												
2	NFSS	0.00	100 PSI	0.20									
3	NFSS	0.50	10 PSI	0.50									
4	NFSS	0.50	30 PSI	0.50									
5	ENTIRE POPULATION SUPPLIED SHELTER TYPE 10 PSI												
6	NFSS	0.30	30 PSI	0.40	100 PSI	0.30							
7	10 PSI	0.50	30 PSI	0.40									
8	NFSS	0.30	30 PSI	0.20	100 PSI	0.40	300 PSI	0.10					
9	10 PSI	0.80	300 PSI	0.20									
10	ENTIRE POPULATION SUPPLIED SHELTER TYPE 30 PSI												
11	NFSS	0.30	100 PSI	0.70									
12	30 PSI	0.50	100 PSI	0.50									
13	ENTIRE POPULATION SUPPLIED SHELTER TYPE 100 PSI												
14	100 PSI	0.50	300 PSI	0.50									
15	ENTIRE POPULATION SUPPLIED SHELTER TYPE 300 PSI												

OUTPUT FOR RUN TYPE INPUT

LETHAL AREA TABLE

	PSI											
	5.40	10.00	19.50	14.00	35.00	42.00	52.00	117.00	141.00	175.00	375.00	490.00
0.00	17.87	8.84	4.41	4.29	2.67	2.21	1.74	0.84	0.71	0.62	0.32	0.26
2500.00	23.14	11.04	5.17	7.67	2.86	2.39	1.89	1.07	0.93	0.76	0.19	0.08
5000.00	24.56	11.44	5.30	8.16	2.96	2.48	1.96	1.13	0.93	0.71	0.07	0.00
7500.00	25.44	12.14	5.51	8.67	3.06	2.57	2.07	1.07	0.93	0.34	0.00	0.00
10000.00	26.63	13.71	6.01	10.10	3.27	2.76	2.24	0.12	0.00	0.00	0.00	0.00
15000.00	29.72	15.72	7.51	11.83	3.43	1.97	0.79	0.00	0.00	0.00	0.00	0.00
20000.00	32.32	17.62	7.83	13.07	1.16	0.49	0.00	0.00	0.00	0.00	0.00	0.00
30000.00	39.29	17.62	2.04	4.78	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40000.00	41.13	7.19	0.57	1.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100000.00	48.39	4.91	0.00	1.57	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

REVISED LETHAL AREA TABLE

	PSI											
	5.40	10.00	19.50	14.00	35.00	42.00	52.00	117.00	141.00	175.00	375.00	490.00
0.00	17.87	8.84	4.41	4.29	2.67	2.21	1.74	0.84	0.71	0.62	0.32	0.26
5000.00	29.72	15.72	7.51	11.83	3.43	1.97	0.79	0.00	0.00	0.00	0.00	0.00

DEFENSE CODE FINAL-2

SUMMARY OF DEFENSE OPTIONS

PSI LEVEL-	5.4	10.0	19.5	14.0	35.0	42.0	52.0	117.0	141.0	175.0	375.0	490.0
DEFENSE OPTION												
1	0.33	0.33	0.33									
2	0.27	0.27	0.27						0.07	0.07	0.07	
3	0.17	0.25	0.26	0.32								
4	0.17	0.17	0.17		0.17	0.17	0.17					
5		0.17	0.19	0.64								
6	0.10	0.10	0.10	0.32	0.13	0.13	0.13	0.10	0.10	0.10		
7		0.08	0.10	0.32	0.17	0.17	0.17					
8	0.10	0.10	0.10	0.07	0.07	0.07	0.07	0.13	0.13	0.13	0.09	0.09
9		0.14	0.15	0.51							0.10	0.10
10					0.33	0.33	0.33					
11								0.23	0.23	0.23		
12	0.10	0.10	0.10		0.17	0.17	0.17	0.17	0.17	0.17		
13								0.33	0.33	0.33		
14								0.17	0.17	0.17	0.29	0.29
15											0.30	0.30

Figure 10. Printout of Data Entered through Run Type INPUT

The data file must be a single-reel, binary buffered file with 900 words per physical record. The file is unlabeled and the terminal record is to be filled with the actual word 777777777777776 (octal) in all 900 words. The file is read on logical unit number 3. Each record contains data for 100 cells, each of which has the following format:

Word 1 Sector number.Format:

LLLLLLLL mmmmmmm, where

L = Latitude of lower right corner in seconds, fixed point.

m = Longitude of lower right corner in seconds, fixed point.

2 State and city code (BCD) Format:

SSCCCCC, where

S = State code, BCD

C = City code, BCD

3 City center and SMSA indicators (BCD) .Format:

00000CS where

C = city center indicator

S = SMSA indicator

4 Nighttime population, fixed point

5 Daytime population, fixed point

6 Industrial output, floating point

7 Area of sector, floating point

8 Population density of sector, fixed point

9 Number of tracts in sector, fixed point

Each cell read is tested to see if the population density is greater than MINDEN and if the industrial density is greater than MININD. If either is above the cutoff the cell is stored into memory; if not, the cell is accumulated into one of 10 "tail cells" according to its maximum population density. MINDEN and MININD are read in the basic data deck.

If sense switch 3 is on, the program gives a printout of the stored cell data in a form corresponding to the above format. This can be of two types, depending on the variable NAMEC, the last card in the basic data deck. If NAMEC is blank, all cells will be printed. If NAMEC is some city code, then only that city will be printed.

When the terminal record is recognized, unit 3 is rewound, and control returns to the option selector.

Run Type LVCTDAT

This is the option which reads in the population tract data for use in the targeting model. Again the data file is a single-reel, binary buffered file, this time with 996 words per physical record. The file is unlabeled with the actual word 777777777777776 (octal) in the terminal record. This file is read on logical unit number 4. Each record contains data for 83 tracts in the following format:

Word 1 Sector number. Format:

LLLLLLLLLmmmmmmmm where,

L = Latitude of lower right corner in seconds, fixed point

m = Longitude of lower right corner in seconds, fixed point

- Word 2 Tract latitude in degrees, floating point
- 3 Tract longitude in degrees, floating point
- 4 Standard location code (BCD)
- 5 City center and SMSA indicators (BCD). Format:
000000CS where
- C = city center indicator
- S = SMSA indicator
- 6 City code (BCD)
- 7 1965 daytime population, fixed point
- 8 1965 nighttime population, fixed point
- 9 1970 nighttime population, fixed point
- 10 1970 daytime population, fixed point
- 11 Area of sector, floating point
- 12 Industrial output, floating point

The city whose code is NAMEC, read in the basic data deck, will be that used for the data for the targeting model.

Run Type MINIMAX

This option enables one to control whether defenses are to be generated using the mixed lambda (or single lambda) procedure or the minimax procedure. Both procedures are defined in the description of subroutine DEFOPT. The minimax approach tries to limit the excess kill over what could be achieved for each of several lambdas.

Only one parameter needs to be set to change to minimax operation; that is the switch, MINIMAX = 1. It is suggested that a smaller number of mus be used and that

NNEW, the number of mus used on second and subsequent passes, be of the order of five or ten for program efficiency. The input definitions are in that section, and sample values for these are shown in Sample Run Deck #2.

Run Type OPTIONS

This run type is used to set one of the parameters IDPOP, IAPOP, IFILL, or IAOB to any value desired. For the format, see the data sequence definition; for examples see the sample run decks.

Run Type PERTURB

This option changes the lethal areas according to any desired assumed change in hardnesses, although the change should be small, since partial derivatives are used to change the areas. The parameter which determines the change is DELTA, the positive or negative fractional change in hardnesses. After reading this parameter, the lethal areas are adjusted, and a revised lethal area table is printed. PERTURB is performed in subroutine PERTURB in OVLAY3.

Run Type TRIMTAIL

This option enables one to set NCELLS to any desired value. It can be used for such things as evaluating a defense without the tail cells being included.

3. Output Options

Run Type OUTPUT

This option enables one to obtain additional information about the defense and the attack on the defense. The routine does three breakdowns--defenses vs. population density, attack vs. density, and attack vs. defense. Figure 11 shows how the population density categories are set up, with the number of people in each category. Next, in the

same figure, an important breakdown of defenses by population density. It is interesting to note, for example, that defense options 3, 5, 7, 9, 14 and 15 were never chosen. Figure 12 shows the distribution of the different attack densities over the population density brackets and over the defenses. No data is required; the option card alone calls for the print. This routine is executed in subroutines POPQUANT and CORELATE.

Run Type PUNCH

Inclusion of a PUNCH card before generating a defense enables one to obtain a card output deck. This contains all the defenses assigned over the country for future reference, and may be used as an input for DEFSPEC. No data is required for this option; the punching is executed in DEFPUNCH.

DENSITY INDEX	1	2	3	4	5	6	7	8	9
DENSITY CATEGORY	7000- 9700	9700- 13600	13600- 18900	18900- 26400	26400- 36800	36800- 51400	51400- 71700	71700 100000	100000 +ABOVE
CELLS IN CATEGORY	676	561	338	258	141	73	41	14	14
FRACTION CELLS	0.216	0.179	0.108	0.076	0.045	0.023	0.013	0.004	0.004
PEOPLE IN CATEGORY	14220175	15487800	12705085	12007420	11753189	8084343	7499988	3329109	699281
FRACTION PEOPLE	0.068	0.074	0.061	0.057	0.056	0.039	0.036	0.016	0.033

DEFENSES VS. DENSITY									
DENSITY CATEGORY-	7000- 9700	9700- 13600	13600- 18900	18900- 26400	26400- 36800	36800- 51400	51400- 71700	71700 100000	100000 +ABOVE
DEFENSE OPTION									
1	0.379	0.246	0.107	0.089	0.014	0.041	0.024	0.000	0.000
2	0.037	0.004	0.000	0.004	0.000	0.000	0.000	0.000	0.000
3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.038	0.029	0.033	0.004	0.000	0.000	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
6	0.037	0.034	0.015	0.000	0.021	0.000	0.000	0.000	0.000
7	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
8	0.000	0.012	0.000	0.004	0.000	0.014	0.000	0.000	0.000
9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
10	0.216	0.036	0.041	0.017	0.028	0.041	0.000	0.000	0.000
11	0.201	0.220	0.263	0.252	0.163	0.123	0.073	0.071	0.000
12	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
13	0.374	0.415	0.541	0.639	0.773	0.781	0.902	0.929	1.000
14	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
15	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Figure 11. Population Density Categories

ATTACK VS. DENSITY FOR ATTLEV = 100.0

DENSITY CATEGORY-	7000- 9700	9700- 13600	13600- 18900	18900- 26400	26400- 36800	36800- 51400	51400- 71700	71700 100000	100000 -ABOVE
ATTACK DENSITY									
0.00	0.916	0.807	0.855	0.777	0.695	0.479	0.268	0.214	0.000
0.06	0.094	0.194	0.145	0.218	0.277	0.274	0.195	0.214	0.000
0.13	0.000	0.000	0.000	0.004	0.028	0.247	0.537	0.500	0.071
0.25	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.071	0.714
0.50	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.214
0.75	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1.00	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1.50	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2.00	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4.00	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
8.00	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
16.00	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

ATTACK VS. DEFENSE FOR ATTLEV = 100.0

DEFENSE OPTION-	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ATTACK DENSITY															
0.00	0.800	3.000	0.000	1.034	0.000	1.000	0.000	2.333	0.000	0.825	1.868	0.000	1.316	0.000	0.000
0.06	0.479	0.000	0.000	0.019	0.000	0.000	0.000	0.000	0.000	0.118	0.000	0.000	0.110	0.000	0.000
0.13	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.038	0.000	0.000	0.043	0.000	0.000
0.25	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.011	0.000	0.000
0.50	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.003	0.000	0.000
0.75	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1.00	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1.50	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2.00	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4.00	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
8.00	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
16.00	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Figure 12

4. Auxiliary Computation

Run Type CROSSCOR

This option sorts the cells into population and industry density brackets and keeps tallies of the marginal distributions as well as the joint distribution. The parameters to set are indicated in the data definition section. The population density brackets are given in population per square nautical mile; the industry density is thousands of dollars per square nautical mile, and the marginal distributions for these brackets are included in the heading. The correlation table accumulates industry in each population density-industry density "box" and normalizes by the total amount of industry in each industry density bracket.

Run Type EXERCISE

This option enables one to obtain tuned (single lambda) defenses for a number of cost levels and lambdas, evaluating each of them for all attack objectives. More importantly, it then uses the lambda which generated the defense to find an attack against the defense, thus obtaining a "solution to the Lagrangian" in the sense of the theoretical section above. Otherwise its usefulness is mostly for small scale data in exploration on the behavior of different lambdas for different costs.

Before entering EXERCISE, all the same parameters need to be set as in SIMPLE; in addition, the attack objectives must be set (presumably using ATTOBJ). EXERCISE uses subroutine WORKOUT, which reads an array of costs, and of lambdas, and of attack levels. For each cost and defense lambda, DEFOPTG is called, and then DEFPUNCH, as well as EVAL and ANSPRIN for every attack objective. Then for every attack objective the attack using the defense lambda is obtained, in OFFOPTZ, and that "solution of the Lagrangian" is printed.

Run Type MIXLAM

This option enables one to observe the effect of changing the relative weights of two lambdas in generating defenses. The routine explores defenses plus evaluations for combinations of the two lambdas in steps. Its use is in small data analysis, for the computation time is very large otherwise. The routine first tries an (unsuccessful) method of guessing the mixed lambda combination and then, if desired, scans all possible combinations of the two lambdas in steps of .05. After each defense is found, an evaluation is performed, and an output is made which is identical to SIMPLE. For input data, see the input sequence definition section.

Run Type TABLDENS

This option tables the optimum attack densities and payoffs as for different values of ρ/λ , where ρ is the population (or payoff) density. This data is for information only--it is not used in the program. Figure 13 shows a sample table for the first defense option. For example, for ρ/λ less than .312 but greater than .102, the optimum attack density is .06 and the payoff is .615 with the height of burst being 6 (air burst). This printout is repeated for all the defense options.

OPTIMUM ATTACK DENSITIES FOR DEF = 1

ATTACK DENSITY	FOR RHO/LAMBDA LESS THAN	ATTACK PAYOFF, LOWER DENSITY	HOB	ATTACK PAYOFF, UPPER DENSITY	HCB
0.00	0.10167	0.00000	0	0.01471	0
0.06	0.31234	0.61471	6	0.81482	6
0.13	0.94059	0.81482	6	0.94229	6
0.25	5.02160	0.94229	6	0.99208	6
0.50	37.15274	0.99208	6	0.99888	6
0.75	246.07702	0.99888	6	0.99982	6
1.00	2805.03997	0.99982	6	1.00000	6
1.50	*****	1.00000	6	1.00000	6
2.00	*****	1.00000	6	1.00000	6
4.00	0.00000	1.00000	6	1.00000	1
8.00	0.00000	1.00000	1	1.00000	1

Figure 13. Sample Table for the First Defense Option

C. GLOSSARY OF VARIABLES IN COMMON

Basic

A (12)	Value (to attacker) of each payoff component
ACOST	Actual cost spent if defense saturates
ALPHA	Weight on offense cost in defense Lagrangian
AREA	Current cell area (sq. n.mi)
AREAMAX	Normalizing constant for cell area
ATTCORR(20, 15)	Frequency count for 15 defense options and 20 attack densities in CORELATE
ATTLEV(20)	Attack levels (in equivalent megatons)
ATTPAY(100)	Total attacker payoff for each current lambda
ATTPAYZ(20)	Best estimate of attacker payoff for each attack level
ATTSPRED(10,20)	Frequency count for 10 density brackets and 20 attack densities in CORELATE
BASEPOP	Base population selected by IDPOP.
BDLAM	Lambda for current balanced defense calculation
BDPSI	Shelter hardness assigned to current cell for balanced defense
CDEF (300)	Cost of optimum defense for each mu, total over cells
CDEFCL (300)	Cost of optimum defense for each mu, current cell
CDLAG (300)	Optimum Lagrangian for each mu, current cell
CELLCOST	Balanced defense cost for current cell
CITPOP (650)	City population
COSTPER (15)	Cost per person of defense option (shelter mix)

CSTSHEL (10)	Cost per shelter space of 10 shelter types.
D (20)	Possible attack density quanta (equivalent megatons/sq.n.m.)
DCOST	Defense cost in dollars for current defense option
DEFEXCL	= 1 if current defense is excluded (set by DCONSTR)
DLAM (22)	Lambdas considered by the defense
ERMEAS (20)	Error measure (fraction of attack level) for each lambda in EVAL
EXCLUDEF(15, 15)	Used in time-phasing; for each possible final defense option EXCLUDEF is 1 for each excluded intermediate defense option, otherwise 0.
FDEF (10, 15)	Fraction of population supplied one of 10 shelter types for each of 15 defense options.
FINC	Industrial value of current cell
FINDMAX	Normalizing constant for cell industrial value.
FKIL (12, 20, 15)	Fraction of 12 hardness components killed by 20 attack densities at 15 heights of burst.
FLAM (100)	Initial set of offense multipliers for evaluation
FLAMLEV (20)	Best estimate of lambda for each attack level
FLAMX(100)	Set of offense multipliers for any call to OFFOPT
FLAMZ(100)	Temporary array for new lambda generation
FMU (300)	Values for current iteration
FMUX(300)	Initial mu values
FRACP (300)	Fraction of split cells' areas on primary defense
HLAG(100)	Optimum offense Lagrangian for each lambda, current cell
HOB (15)	The 15 possible heights of burst
HOPT (100)	Optimum payoff for each lambda, current cell

IAOB	Index of attack objective
IAPOP	Population basis at attack (second and third population bases): 1 - night, 2 - day, 3 - maximum, 4 - minimum
ICBEG	Index of first cell of group to be optimized
ICEND	Index of last cell of group to be optimized
ICELL	Current cell index
ICOMP(12)	Hardness level of each (non-empty) payoff component
ICT1 (650)	Starting index in cell data for city
ICT 2(650)	Ending index in cell data for city
IDEF	Index of defense shelter option
IDENS	Attack density index
IDFOPT (300)	Index of optimum defense for each mu, current cell
IDOPT(100)	Optimum attack density index for each lambda, current cell
IDPOP	Population basis for defense option (first population basis): 1 - night, 2 - day, 3 - maximum, 4 - minimum
IFILL	Shelter filling mode: 1 - optimal, 2 - partial alert, 4 - balanced defense
IHI (20)	Index of bracketing lambda (large of two lambdas) for each attack level in EVAL
IHIDLAM	Index of lowest (variable misnamed) lambda
IHOB	Height of burst index
IHOBOPT (100)	Optimum height of burst index for each lambda, current cell
ILODLAM	Equivalent to IHIDLAM
ILSAVE	Index of closest (lower est.) mu in cost closing
ISAT	= 2 if defense cannot spend TCOST, otherwise = 1
ISPLC (300)	Cell index of split cells

IST1 (60)	Starting index in cell data for state
IST2 (60)	Ending index in cell data for state
ISW1	Setting both = 1 allows changing cell data
ISW2	
JDEF	Normalizing constants FINDMAX, AREAMAX, POPMAX
JDEFSP (300)	Defense option, current cell
JPOPDENS(10)	Secondary defense option of split cells
KDEF	Population density brackets in POPQUANT
MAPHOB (15)	Auxiliary defense option (for constraint purposes)
MINDEN	Indices of heights of burst scanned
MINIMAX	Population density cutoff for cells stored in memory
MININD	= 1 for minimax selection of cell defense Lagrangian over lambdas
MMAOB	Industry density cutoff for cells stored in memory
NAHOB	Originally for minimax over attack objectives, not now used
NALEV	Actual number of heights of burst scanned
NAMCIT (650)	Number of attack levels
NAMSTAT(60)	City names (BCD)
NAOB	State names (BCD)
NCELA(5000)	Number of attack objectives
NCELB (5000)	
NCELLS	Packed cell data, to be unpacked by RDCELL
NCIT	Number of population cells
	Number of cities

NCOMP	Number of hardness components in payoff (non-empty)
NDEF	Number of defense options
NDENS	Number of attack density quanta
NDLAM	Number of lambdas considered by the defense
NHOB	Number of height of burst options
NLAM	Number of attacker Lagrange multipliers
NLAMX	Number of current offense multipliers
NMU	Number of initial mu values
NMUX	Number of mus in current iteration
NNEW	Number of mus used in second and subsequent passes in DEFOPTG
NPOPDEN(10)	Number of cells in density bracket in POPQUANT
NPSI	Number of hardness levels
NSPLIT	Number of split defense cells
NSPLTOLD	Number of split cells in previous defense (for time-phasing)
NSTAT	Number of states
NSTYP	Number of shelter types
PAYIND(100)	Total industrial value destroyed for each current lambda
PAYINDZ(20)	Best estimate of industrial kill for each attack level
PAYPOP(100)	Total population kill for each current lambda
PAYPOPZ(20)	Best estimate of population kill for each attack level
PKILL(100)	Optimum population kill for each lambda, current cell
PKP	Daytime population of current cell
PMAX	Maximum population of current cell

POPACT	Actual current cell population
POPCOUNT(10)	Population in density bracket in POPQUANT
POPMAX	Normalizing constant for cell population
POPMULT(10)	Factor to convert population kill to attacker value
POPSH(10)	Population sheltered in 10 shelter types , current cell
POPUL(12)	Population in each hardness component
PSI(15)	The 15 possible psi levels
PSIMAX	Upper cutoff psi for balanced defense
PSIMIN	Lower cutoff psi for balanced defense
PUNCH	Setting = 1 gives punch output
RES	Nighttime (resident) population of current cell
SHLSPRED(10,15)	Frequency count for 10 density brackets and 15 defense options in CORELATE
SPACES(10)	Spaces available in 10 shelter types , current cell
SPREAD (12,10)	Fraction of population in each of 10 shelter types assured vulnerable at each of 12 hardnesses .
SPRIND (12)	Fraction of industrial value at each of 12 hardnesses
STATE	Setting = 1 gives STATCOST defense
STPOP(60)	State population
SWITCH	Variable read by option selector
TCOST	Target cost of defense investment for group of cells
TOLERR	Tolerable percent error in attack level
VALIND (12)	Industrial value in each hardness component

VINDMULT(10)	Factor to convert industrial value killed to attacker value.
VKILL(100)	Optimum industrial kill for each lambda, current cell
W(15,15)	Lethal area for each height of burst and hardness
WDAOB(10)	Weight defense assigns to each attack objective
WDLAM(15)	Weight defense assigns to each lambda
WEPEQ(100)	Optimum weapon expenditure for each lambda, current cell
WMIN	Lethal area corresponding to upper psi cutoff, balanced defense
WPEXP(100)	Total weapon expenditure for each current lambda

Glossary (continued)

Bounding Procedure

BCOST(3)	Cost at beginning, middle, and end of lambda sweep
BKILL(3)	Payoff at beginning, middle, and end of lambda sweep
BLAM(3)	Lambda at beginning, middle, and end of lambda sweep
BMAX	Upper bound values
BMIN	Lower bound values
BMU(3)	Mu at beginning, middle, and end of lambda sweep
BWEP(3)	Weapon exp. at beginning, middle, and end of lambda sweep
CUMCOST(20,20)	Cumulative cost for 20 lambdas and 20 mus
CUMKILL(20,20)	Cumulative payoff for 20 lambdas and 20 mus
CUMWEP(20,20)	Cumulative weapon expenditure for 20 lambdas and 20 mus
CURCOST(20,20)	Current cell cost for 20 lambdas and 20 mus
CURLAG(20,20)	Current cell Lagrangian for 20 lambdas and 20 mus
CURWEP(20,20)	Current cell weapon expenditure for 20 lambdas and 20 mus
EPS	Parameter controlling separation between lambdas chosen
HILAM	Maximum lambda which need be considered
HSAT	Maximum payoff possible
JJCH0Z(22)	Index of closest mu for each lambda
MODESW	Selects mode of running verification
NENO	Number of end limits reached on lambda sweep
SATWEP(22)	Weapon expenditure at saturation, line upper boundary payoff
SLOPE(20)	Slope of line lower boundary payoff

VERIFY	Setting = 1 turns on verification procedure
YCEPTLO (20)	Intercept of line lower boundary payoff
YNTCEPT (22)	Intercept of line upper boundary payoff

D. INPUT DATA SEQUENCE DEFINITIONS

This section contains the definition of the data sequences for the various run options or types and for the basic data deck; the data must, of course, appear in this form. The data for each option begins with the option card, shown at the top of each definition, and continues or branches according to the defined sequences.

The formats are all exactly as they appear in the various subroutines; the subscripts on all subscripted variables are dummy variables in the program and are shown here as I or J.

In addition to the options given here, options EVAL, LIVEDATA, LVCTDAT OUTPUT, PUNCH, STOP, TABLDENS, TIMEFAZE, and UNSHEL have no data cards except the option card itself. Option STATCOST and SIMPLE have the same data sequence, but STATCOST allocates money to states by population and then optimizes the nationwide defense state by state. Option WIGGLE has been dismantled and if used, has the same effect as a SIMPLE card.

BASIC DATA DECK INPUT SEQUENCE

<u>FORTRAN Variable</u>	<u>Format</u>	<u>Explanation</u>
NDENS	(8I10)	Number of attack density quanta.
D(I)	(8F10.2)	Density quantum levels.
NHOB, NPSI, LACOMP	(8I10)	NHOB is number of heights of burst (≤ 15). NPSI is the number of hardnesses (≤ 15). If LACOMP $\neq 0$, lethal areas are computed from lethal radii.
HOB(J)	(8F10.2)	Here read NHOB heights of burst.
PSI(I)	(8F10.2)	Here read NPSI hardnesses.
IF LACOMP $\neq 0$ skip to * in sequence.		
WJ,I)	(8F10.2)	Here read lethal areas (sq. n.mi) by HOB (on single cards) and PSI.
Skip to ** in sequence.		
* WJ,I)	(8F10.2)	Here read lethal radii (kilofeet) by PSI (on single card) and HOB.
** NAHOB	(8I10)	NAHOB is the actual number of heights of burst to be scanned (\leq NHOB). If NAHOB = NHOB skip next card in sequence.
MAPHOB(J)	(8I10)	MAPHOB(J) are the indices of the NAHOB height of burst scanned.
** NSTYP, NDEF	(2I10)	NSTYP is the number of shelter types. NDEF is the number of defense options.
FDEF (I,J)	(8F10.2)	Here read fraction of each defense option J composed of each shelter type I by defense option (on single card) and shelter type.
SPREAD(I,J)	(8F10.2)	Here read fraction of each shelter type J at each hardness I by shelter type (on single card) and hardness.
SPRIND (I)	(8F10.2)	Fraction of Industrial Value assumes to have each of NPSI hardnesses.
CSTSHEL (I)	(8F10.2)	Cost per person for each of NSTYP shelters.
FINDMAX, AREAMAX, POPMAX	(3F10.2)	Maximum industrial value, area, and population over all cells.
ICELL, RES, PKP, FIND, AREA, JDEF, KDEF	(I10, 4F10.2, 2I10)	ICELL is the cell number, RES is nighttime population, PKP is daytime population, FIND is industrial value, AREA is area, JDEF is defense option and KDEF is for constraint purposes. Continue reading these cards (cell data) until ICELL ≤ 0 .
NALEV	(8I10)	Number of attack levels (≤ 20).
ATTLEV(I)	(8F10.2)	Here read NALEV attack levels.
MINDEN, MININD	(2I10)	Lower population density and industrial density cut-offs for a cell's inclusion into memory. A cell is included if $\text{MAX} \{ \text{PKP, RES} \} / \text{AREA} \geq \text{MINDEN}$ or $\text{FIND} / \text{AREA} \geq \text{MININD}$.
NAMEC	(A8)	If NAMEC is blank, the cell printout in option LIVEDATA will list all cells; if it is a city code, the listing will be of that city only.

Data Sequence for Run Type **ALERT**

<u>Fortran Variables</u>	<u>Format</u>	<u>Explanation</u>
FALERT	(8F10.2)	Fraction of population alerted.

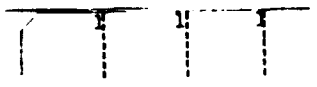
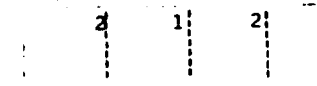
Data Sequence for Run Type **ATTOBJ**

<u>Fortran Variables</u>	<u>Format</u>	<u>Explanation</u>
NAOB	(I10)	Number of attack objectives
POPMULT(I)	(8F10.2)	Here read population kill multipliers for NAOB attack objectives.
VINDMULT(I)	(8F10.2)	Industrial kill multipliers.
WDAOB(I)	(8F10.2)	Here read weight defense assigns to NAOB attack objectives.

Data Sequence for Run Type **BALDEF**

<u>Fortran Variable</u>	<u>Format</u>	<u>Explanation</u>
DEFNAME, NEWDATA	(A8,2X,I10)	DEFNAME is any 8 character code for the balanced defense. If NEWDATA = 0 skip to *** in sequence. The following lethal area data (down to ***) needs to be specified for unsheltered posture only.
NHOB, NPSI, LACOMP	(8I10)	NHOB is number of heights of burst (= 15). NPSI is the number of harnesses (= 15). If LACOMP ≠ 0, lethal areas are computed from lethal radii.
HOB (J)	(8F10.2)	Here read NHOB heights of burst.
PSI(I)	(8F10.2)	Here read NPSI harnesses.
If LACOMP ≠ 0 skip to * in sequence.		
WJ,I)	(8F10.2)	Here read lethal areas (sq. nm) by HOB (on single cards) and PSI.
Skip to ** in sequence		
* WJ,I)	(8F10.2)	Here read lethal radii (kilofett) by PSI (on single card) and HOB.
** NAHOB	(8I10)	NAHOB is the actual number of heights of burst to be scanned (= NHOB). If NAHOB = NHOB skip next card in sequence.
MAP HOB(J)	(8I10)	MAP HOB(J) are the indices of the NAHOB heights of burst scanned.

BALDEF, Cont.

Fortran Variables	Format	Explanation
*** NSTYP = 2	(8I10)	Two shelter types -- unsheltered and variable hardness sheltered.
NDEF = 2	(8I10)	Two defense options.
SHLNAME, CSTSHEL, NHARD, XK	(A8,2X,F10.2, I10,F10.2)	Here read unsheltered data. SHLNAME is the name of the unsheltered posture. CSTSHEL is the cost per person of the unsheltered posture (usually = 0.0) NHARD is the number of hardnesses of the unsheltered posture. If NHARD = 1 then XK is the hardness of the unsheltered posture in psi and skip next card. If NHARD ≠ 1 then XK may be blank and read next card to obtain hardness spread. Hardness must appear in PSI array.
HARD (1), FRAC (1), HARD (2), FRAC (2), ...	(8F10.2)	Hardnesses of unsheltered posture in psi and fraction of population assumed to have each hardness.
SHLNAME, CSTSHEL, NHARD, XK	(A8,2X,F10.2, I10,F10.2)	Here read variable hardness data. SHLNAME is arbitrary. CSTSHEL is arbitrary or blank (it is not used). NHARD = 1. XK should be one of the hardnesses used for the unsheltered posture.
	(3I10)	These two cards define the defense options for balanced defense.
	(3I10)	
SPRIND(I)	(8F10.2)	Fraction of industrial value assumed to have each of NPSI hardnesses.
NCOST	(I10)	Number of costs for which defenses will be computed.
COST (I)	(-9PF10.2)	Here read NCOST costs (in billions of dollars).
NALEV	(I10)	Number of attack levels.
ATTLEV (I)	(8F10.2)	Here read NALEV attack levels (in equivalent megatons).
PSIMIN, PSIMAX, WMIN	(8F10.2)	PSIMIN is the lower PSI cutoff. If balanced defense assigns hardness below this level, then unsheltered posture is assigned. PSIMAX is the maximum allowable hardness for a shelter. WMIN is the lethal area corresponding to PSIMAX.

Data Sequence for Run Type

BOUND

	<u>Fortran Variables</u>	<u>Format</u>	<u>Explanation</u>
	MODESW	(I10)	This selects a mode for the bounding procedure. After initial data go to sequence for mode desired.
	NEWMU	(I10)	If NEWMU \neq 0 read new mus, otherwise skip to mode sequence desired.
	FMU(1)	(8 F10.2)	Here read 20 new mus.
Mode	1		
	DLAM(1)	(F10.2)	This mode computes initial point from a cost and a lambda, then sweeps lambdas from 10.0 to 1000000.0 to generate bounds. This is the initial lambda.
	TCOST	(F20.2)	Total cost of shelter posture (input in dollars).
	HSAT	(F20.2)	Total payoff (or population).
	EPS	(F10.2)	Parameter controlling separation between lambdas chosen. Approximately it is the distance between upper and lower bounds.
Mode	2		This mode reads in initial point, otherwise as in Mode 1.
	BKILL(2), SWEP(2), BCOST(2), BLAM(2), BMU(2)	(F20.2, F10.2, F20.2, 2F10.2)	Initial kill, weapon expenditure, cost, lambda, and mu.
	HSAT	(F20.2)	
	EPS	(F10.2)	
Mode	3		This mode reads in initial point; then lambdas are chosen to verify this initial point, and bounds are set for the attack level of the initial point. The data sequence is exactly as in Mode 2.
Mode	4		This mode computes initial point from a cost and an attack level, then verifies that point as in Mode 3.
	ATTLEV(1)	(F10.2)	Attack level for initial point.
	TCOST	(F20.2)	
	HSAT	(F20.2)	
	EPS	(F10.2)	
Mode	5		This mode reads lambdas and does one pass, computing the bounds for those lambdas.
	TCOST	(F20.2)	
	NDLAM	(I10)	Number of lambdas (1 \leq 20)
	DLAM(I)	(8F10.2)	Here read NDLAM lambdas (ascending order)
	HSAT	(F20.2)	

5.0.0.0

	<u>Fortran Variables</u>	<u>Format</u>	<u>Explanation</u>
Mode			
			6
	BKILL(2), BWEP(2), BCOST(2), BLAM(2), BMU(2)	(F20.2, F10.2, F20.2, 2F10.2)	Kill, weapon expenditure, cost, lambda, and mu.
	TCOST	(F20.2)	
	NDLAM	(I10)	Number of lambdas (≤ 20)
	DLAM(I)	(8F10.2)	Here read NDLAM lambdas (ascending order)
	HSAT	(F20.2)	
Mode			
			7
	BKILL(1), BWEP(1), BCOST(1), BLAM(1), BMU(1)	4(016,4X)	Lower attack level end of previous sweep.
	BKILL(3), BWEP(3), BCOST(3), BLAM(3), BMU(3)	4(016,4X)	Upper attack level end of previous sweep.
	TCOST, HSAT, EPS	(-9PF 10.2, OPF 20.2, -6PF10.2)	

Data Sequence for Run Type CROSSCOR

<u>Fortran Variables</u>	<u>Format</u>	<u>Explanation</u>
NBOXES, IDPOP, INORM, ICBEG, ICEND, MIN	(8I10)	<p>NBOXES is the number of correlation categories (≤ 31).</p> <p>IDPOP is the population basis = 1, 2, or 3 for NIGHT, DAY, and MAX.</p> <p>INORM is the normalization mode =1 normalized by cells in population bracket =2 normalized by cells in industry bracket =3 normalized by population in bracket =4 normalized by industry in bracket</p> <p>ICBEG is the beginning index of the group of cells; ICEND is the end.</p> <p>MIN is the lower cutoff of density brackets; all cells below this density are placed in one category.</p>

Data Sequence for Run Type

DEFSPEC

Fortran Variable	Format	Explanation
KEY, TCOST, NCELLS	(10X, I10, 20X, -9PF10.2, 10X, I10)	KEY is any integer number to identify the defense TCOST is the cost of the defense NCELLS is the number of cells included. If NCELLS \neq old NCELLS on error message is printed and remaining cells are set to unsheltered condition.
IDSPEC (I)	(40I2)	Here read NCELLS defense options, one for each cell at 40 per card. Each IDSPEC \neq NDEF except for split cells, where index is 100 - defense option.
NSPLIT	(I10)	Number of split cells.
ISPLC(I), JDEFSP(I), FRACP(I)	(2I10, F10.8)	Here read NSPLIT split cell data cards --- index of split cell, secondary defense of split cell, and fraction on primary defense.
GO or SKIP or QUIT		If this card is GO continue in sequence. If this card is SKIP skip to * in sequence. If this card is QUIT skip to *** in sequence.
IDPOP, IAPOP, IFILL, ISAT, NDLAM, NAOB, MINIMAX, MMAOB	(6I10, 2L10)	IDPOP and IAPOP are population bases, IFILL is shelter filling mode, ISAT is saturation indicator, NDLAM is number of lambdas for defense, NAOB is number of attack objectives, and MINIMAX and MMAOB deal with minimax.
TCOST, ACOST, ALPHA	(-9PF10.7, 10.7, 0PF10.8)	TCOST is the shelter cost ACOST is actual cost in case of saturation ALPHA is normally 1.0.
DLAM(I)	(8F10.2)	Here read NDLAM lambda considered by defense.
WDLAM(I), WDADB(1), POPMULT(1), VINDMULT(1), WDAOB(2), POPMULT(2)	(8F10.8)	Here read NDLAM weights for lambdas and NAOB attack objective weights, population multipliers, and industry multipliers.
GO or SKIP or QUIT		If this card is GO continue in sequence. If this card is SKIP skip to ** in sequence. If this card is QUIT skip to *** in sequence. Note that IAOB is not an input variable in this data sequence. It may be put in using the OPTIONS sequence.
NHOB, NPSI, LACOMP	(8 I10)	NHOB is number of heights of burst (≤ 15). NPSI is the number of hardnesses (≤ 15). If LACOMP \neq 0, lethal areas are computed from lethal radii.
HOB(J)	(8F10.2)	Here read NHOB heights of burst.
PSI (I)	(8F10.2)	Here read NPSI hardnesses.
If LACOMP \neq 0 skip to + in sequence.		

DEFSPEC Card

<u>Fortran Variable</u>	<u>Format</u>	<u>Explanation</u>
W(J,I) Skip to → → in sequence.	(8F10.2)	Here read lethal areas (sq. nm) by HOB (on single cards and PSI)
→ W(J,I)	(8F10.2)	Here read lethal radii (kilofeet) by PSI (on single card) and HOB.
→ → NAHOB	(8 I 10)	NAHOB is the actual number of heights of burst to be scanned (≤ NHOB). If NAHOB = NHOB skip next card in sequence.
MAPHOB (J)	(8 I 10)	MAPHOB(J) are the indices of the NAHOB height of burst scanned.
<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="border: 1px solid black; padding: 2px;">GO</div> <div style="border: 1px solid black; padding: 2px;">NEW</div> <div style="border: 1px solid black; padding: 2px;">SKIP</div> <div style="border: 1px solid black; padding: 2px;">QUIT</div> </div> <div style="display: flex; justify-content: space-around; align-items: center; margin-top: 5px;"> or or or or </div>		If this card is GO continue in sequence. If this card is NEW, insert entire INPUT sequence (without INPUT card). Note that with NEWDATA = 0 it is not necessary to repeat RDATT sequence. No more data required after that. If this card is SKIP or QUIT skip to *** in sequence.
*** NSTYP, NDEF	(2I10)	NSTYP is the number of shelter types. NDEF is the number of defense options.
FDEF (I,J)	(8F10.2)	Here read fraction of each defense option J composed of each shelter type I by defense option (on single card) and shelter type.
SPREAD (I,J)	(8F10.2)	Here read fraction of each shelter type J at each hardness I by shelter type (on single card) and hardness.
SPRIND (I)	(8F10.2)	Fraction of industrial value assumed to have each of NPSI hardnesses.
CSTSHEL (I)	(8F10.2)	Cost per person for each of NSTYP shelters.
<div style="border: 1px solid black; padding: 2px;">NOEVAL</div> <div style="border: 1px solid black; padding: 2px; margin-left: 10px;">or</div>		If this card is NOEVAL, no evaluation will be performed, otherwise DEFSPEC will evaluate the defense with parameters as set.

Data Sequence for Run Type

EXERCISE

<u>Fortran Variable</u>	<u>Format</u>	<u>Explanation</u>
NCOST	(I10)	Number of costs for which defenses will be computed.
COST(I)	(4F20.2)	Have read NCOST costs (in dollars).
NWLAM	(I10)	Number of lambdas for which defenses will be computed.
WLAM(I)	(8F10.2)	Have read NWLAM lambdas.
NALEV	(I10)	Number of attack levels.
ATTLEV(I)	(8F10.2)	Have read NALEV attack levels (in equivalent megatons).

Field Name	Format	Explanation	
DEFNAME NEWDATA	(A8, 2A, 10)	DEFNAME is any 8 character code for shelter input. If NEWDATA = 0 skip to *** in sequence.	
NHOB, NPSI, LACOMP	(8 I 10)	NHOB is number of heights of burst (≤ 15). NPSI is the number of hardnesses (≤ 15). If LACOMP $\neq 0$, lethal areas are computed from lethal radii.	
HOB (J)	(8F10.2)	Here read NHOB heights of burst.	
PSI (I)	(8F10.2)	Here read NPSI hardnesses.	
If LACOMP $\neq 0$ skip to * in sequence.			
W(J,I)	(8F10.2)	Here read lethal areas (sq. mi) by HOB (on single cards) and PSI.	
Skip to ** in sequence.			
* W(J,I)	(8F10.2)	Here read lethal radii (kilofeet) by PSI (on single card) and HOB.	
** NAHOB	(8 I 10)	NAHOB is the actual number of heights of burst to be scanned (\leq NHOB). If NAHOB = NHOB skip next card in sequence.	
MAPHOB (J)	(8 I 10)	MAPHOB (J) are the indices of the NAHOB height of burst scanned.	
*** NSTYP	(I 10)	Number of shelter types. (≤ 10)	
NDEF	(I 10)	Number of defense options. (≤ 15)	
NSTYP of these groups	SHLNAME, CSTSHEL, NHARD, XK	(A8, 2X, F10.2, 10, F10.2)	SHLNAME is the name of the shelter- any 8 characters. CSTSHEL is the cost per person of the shelter. NHARD is the number of hardnesses of the shelter. If NHARD = 1 then XK is the hardness of the shelter in psi and skip next card. If NHARD $\neq 1$ XK may be blank and read next card to obtain hardness spread. Hardness must appear in PSI array
	HARD (1), FRAC (1), HARD (2), FRAC(2), ...	(8F10.2)	Hardnesses of shelter in psi and fraction of shelter assumed to have that hardness.
NDEF of these groups	IDEF, NSHEL, KX	(3 I 10)	IDEF is the index of the defense option. NSHEL is the number of shelters included in the option. If NSHEL = 1 then KX is the index of the shelter type and skip next card. If NSHEL $\neq 1$ KX may be blank and read next card for shelter mix.
	ISHEL (1), FRAC(1), ISHEL(2), FRAC(2), ...	(4 I 10, F10.2)	Indices of shelter, types and fraction of mix composed of each.
SPRIND (I)	(8F10.2)	Fraction of industrial value assumed to have each of NPSI hardnesses.	

Data Sequence for Run Type

MINIMAX

<u>Fortran Variable</u>	<u>Format</u>	<u>Explanation</u>
MINIMAX, MMAOB, NDLM, NMLD	(2L10, 6I10)	MINIMAX = 1 will cause DEFOPT to run in minimax procedures; 0 for normal procedure. MMAOB for minimax attack objectives, not used -- should be blank or zero. If NDLM = 0 no more data is required. If NDLM is non-zero, it becomes NMU and next card is read. If NMLD is non-zero it becomes NNEW, the number of mus on the second pass in DEFOPTG.
FMU (I)	(BE10, 4)	Here read NMU mus.

Data Sequence for Run Type

MIXLAM

<u>Fortran Variable</u>	<u>Format</u>	<u>Explanation</u>
XMLAM(1), XMLAM(2)	(2F10, 2)	Two lambdas to be used for mixed-lambda combinations.
TCOST	(-9PF10, 2)	Total cost of defense posture (in millions).
EXPLORE or		If this card is EXPLORE, program will explore mixed-lambda combinations in steps of 0.05 after trying first mixed-lambda guess.

Data Sequence for Run Type

OPTIONS

<u>Fortran Variables</u>	<u>Format</u>	<u>Explanation</u>
SWITCH, IVALUE	(A8, 2X, I10)	SWITCH is examined to see if it is IDPOP, IAPOP, IFILL, or IAOB. If it is, the program sets whatever variable it is to IVALUE and reads the next card. If it is not, the run type terminates, and SWITCH is used as the next run type card.
IDPOP		
or		
IAPOP		
or		
IFILL		
or		
IAOB		

Data Sequence for Run Type

PERTURB

<u>Fortran Variables</u>	<u>Format</u>	<u>Explanation</u>
DELTA	(8F10, 2)	Fractional change in hardnesses of shelters (positive or negative).

Data Sequence for Run Type

SIMPLE

and

STATCOST

<u>Fortran Variable</u>	<u>Format</u>	<u>Explanation</u>
TCOST	(9PF10.2)	Total cost of shelter posture (input in billions)
NNDLAM	(I10)	Number of lambdas to be considered by defense (≤ 15) If NNDLAM = 0 program uses previous lambdas; skip to * in sequence.
DLAM(I)	(8F10.2)	Here read NNDLAM defense lambdas.
NWDLAM	(I10)	If NWDLAM = 0 program assigns equal weights to lambdas; skip to * in sequence.
WDLAM	(8F10.2)	Have read NNDLAM weights for defense lambdas.
*NAOB	(I10)	Number of attack objectives (≤ 10). If NAOB = 1, no more data is required.
POPMULT(I)	(8F10.2)	Here read population kill multipliers for NAOB attack objectives.
VINDMULT(I)	(8F10.2)	NAOB industrial kill multipliers. After reading this card the program generates the defense.
NOEVAL or		If NOEVAL card appears, no more data is required and this run type terminates without evaluating the defense.
**NEWDAOB	(I10)	If NEWDAOB = 0, no more data is required. Otherwise, read next card.
WDAOB(I)	(8F10.2)	Here read weights defense assigns to NAOB attack objectives. Program generates new defense and evaluates it for all attack objectives.

Return to ** in sequence.

Data Sequence for Run Type

TRIMTAIL

<u>Fortran Variable</u>	<u>Format</u>	<u>Explanation</u>
NCELLS	(I10)	NCELLS is the number of cells considered in the run. The number may be cut down to exclude tail cells, or may be set for any other purpose.

E. VALUES OF IMPORTANT PROGRAM CONSTANTS

The following variables are initially set in the program to the values indicated.

These may then be reset as desired using the input sequences.

IDPOP = 1

IAPOP = 1

IFILL = 1

IAOB = 1

NAOB = 1

NDLAM = 1

POPMULT(I) = 1.0

VINDMULT(I) = 0.0

WDAOB(I) = 1.0

DLAM(I) = 2500.0

WDLAM(I) = 1.0

TCOST = 10000000000.0

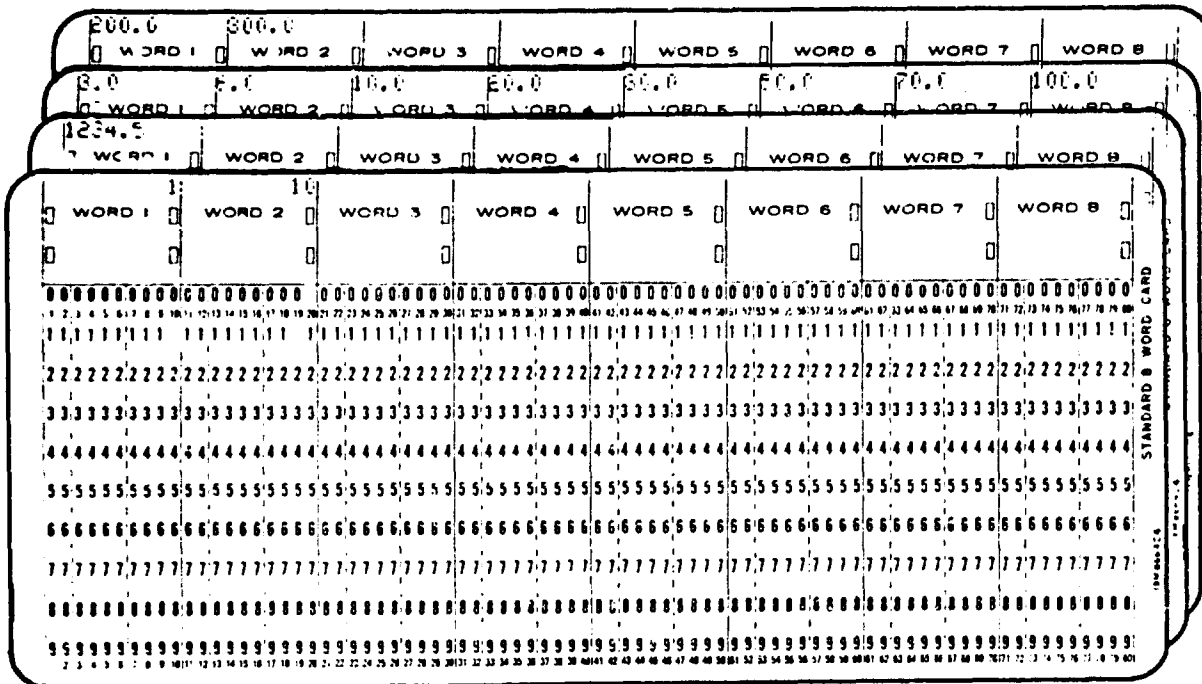
III. SAMPLE DECK

A. BASIC DATA DECK

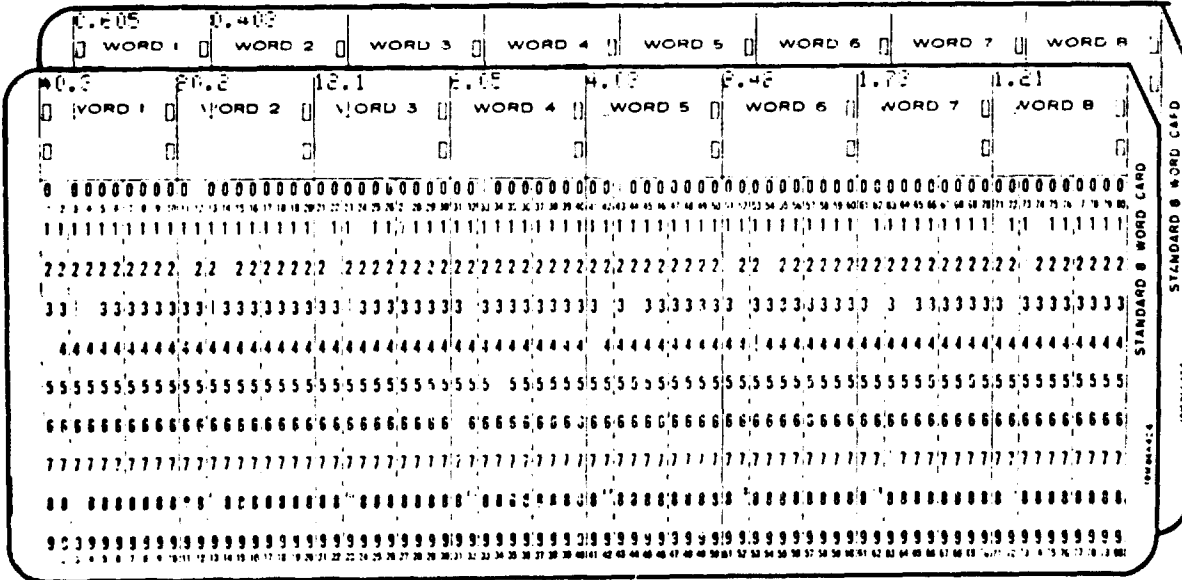
Most of the basic data deck exists only because the program grew over a period of time, and this method of reading data once and then writing over it insured that runs did not terminate because a parameter was unset. There is no reason to ever change cards on most of this data deck; the important variables (the ones which are not normally overwritten) will be noted so changes may be made.

2.0		4.0		2.0		16.0		WORD 5		WORD 6		WORD 7		WORD 8	
WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8	WORD 9	WORD 10	WORD 11	WORD 12	WORD 13	WORD 14	WORD 15	WORD 16
0.1	1.0E25	0.125	0.25	0.5	0.75	1.0	1.5	WORD 9	WORD 10	WORD 11	WORD 12	WORD 13	WORD 14	WORD 15	WORD 16
WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8	WORD 9	WORD 10	WORD 11	WORD 12	WORD 13	WORD 14	WORD 15	WORD 16
12															
WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8	WORD 9	WORD 10	WORD 11	WORD 12	WORD 13	WORD 14	WORD 15	WORD 16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9

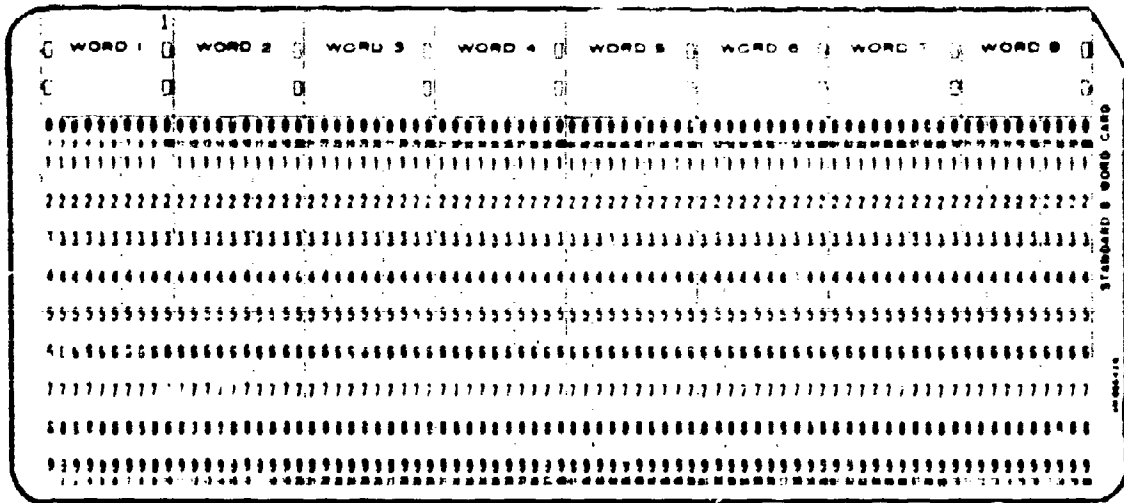
First is read the number of attack density levels and the density quantity themselves. These density levels (especially the minimum nonzero one) affect the way the density model approximates the damage effects; the values here were used on all the FINAL shelter runs except one. This data is not overwritten, and the attack density levels are set only in the basic data deck.



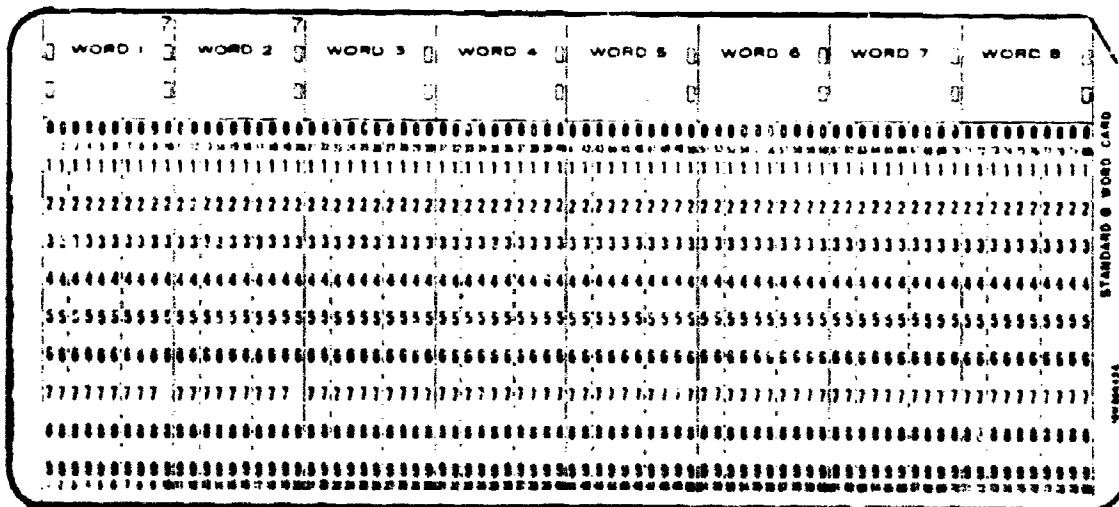
These cards indicate one height of burst (=1234.5) and 10 psi levels, and give the quantities. The hardness data and small population data in this basic data deck correspond to the data used in the analytic model in Appendix C.



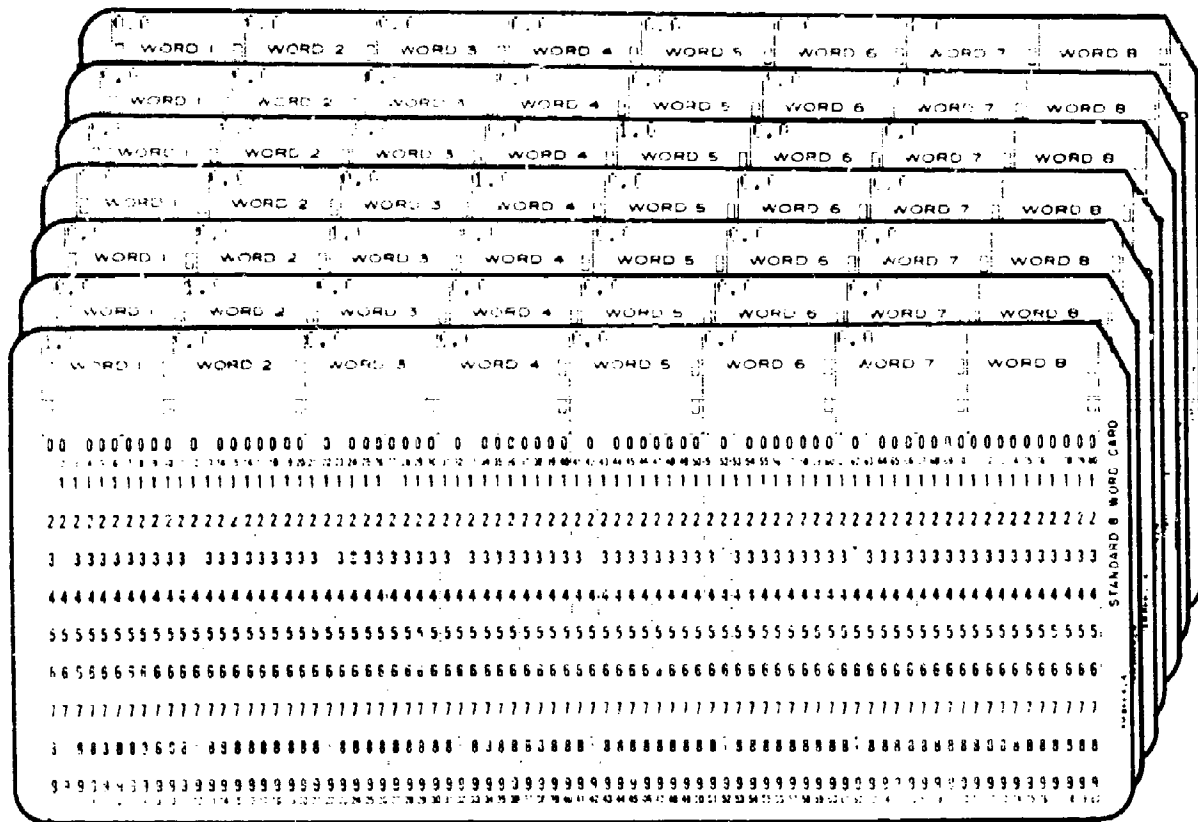
The 10 lethal areas for the simple height of burst are next (these were computed from L.A. = 121/psi).



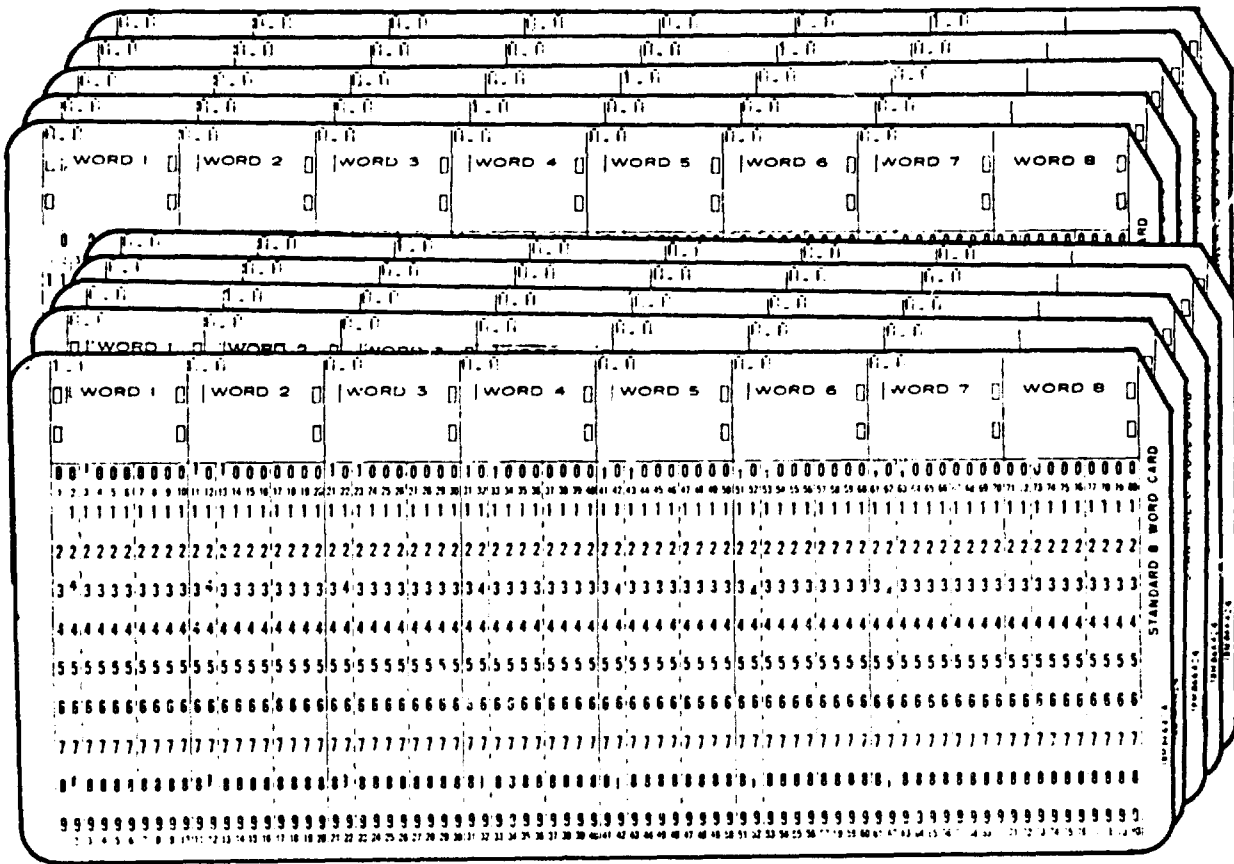
This is the actual number of heights of burst.



Here is indicated 7 shelter types and 7 defense options.



This is the FDEF array; there are 7 defense options consisting of pure deployments of the 7 shelter types.



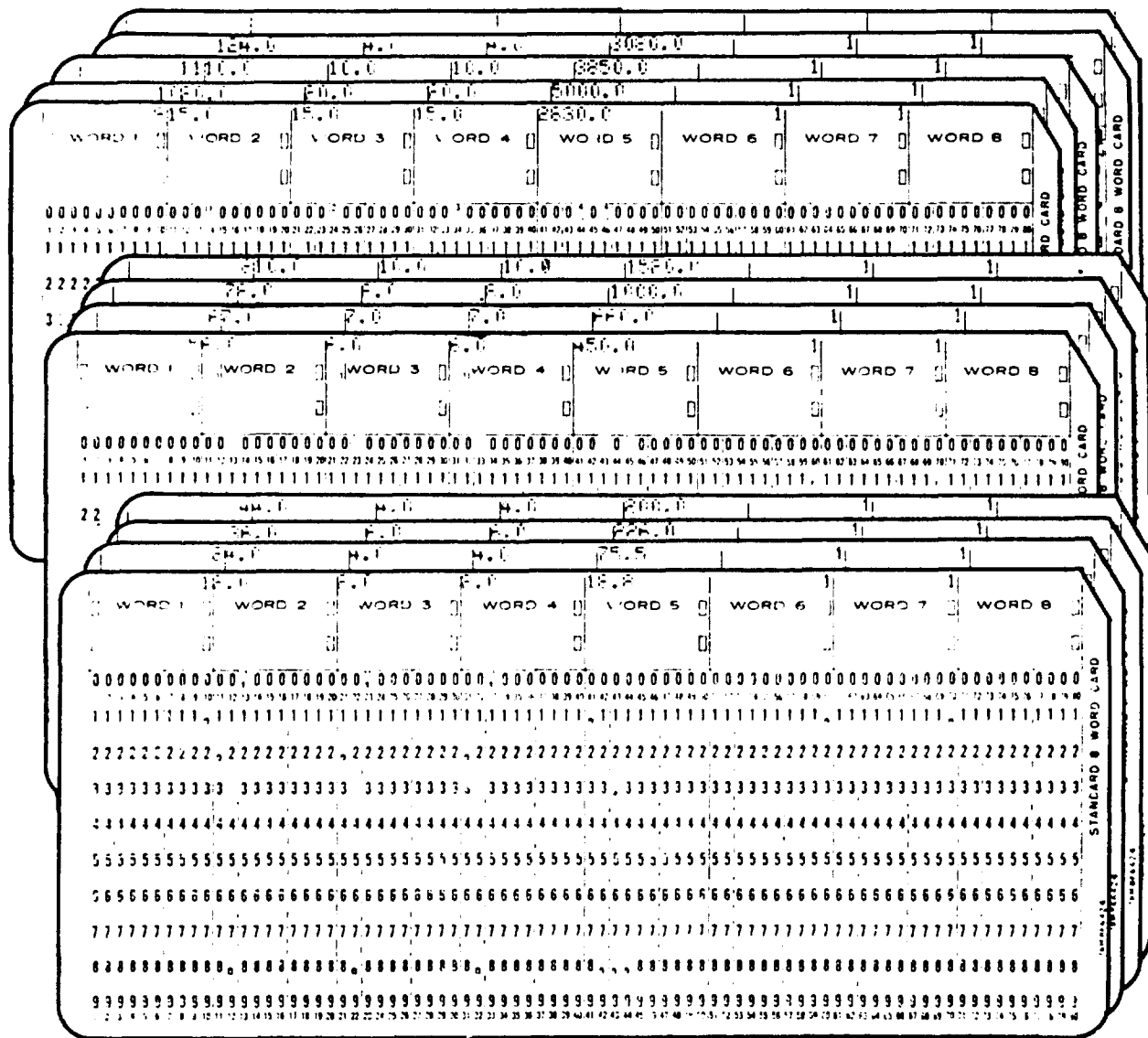
This is the SPREAD array, indicating 7 pure shelter types at 3, 10, 30, 70, 100, 200, and 300 psi.

WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111
22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222
33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333
44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444
55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555
66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666
77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777
88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888
99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999

This is the SPRIND array - the definition of industrial hardness for this basic data.

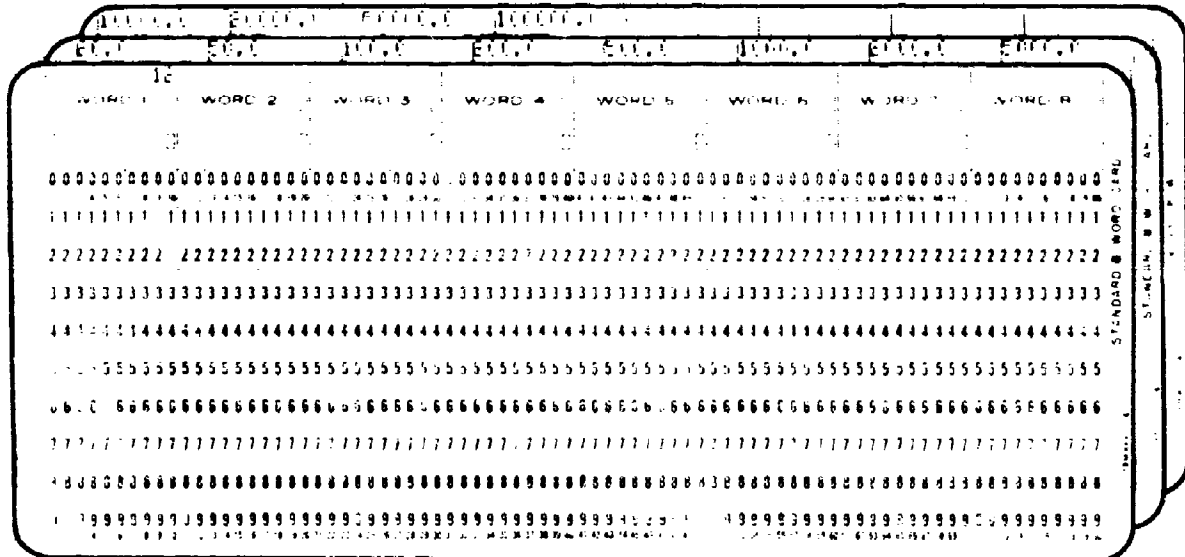
WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111
22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222
33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333
44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444
55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555
66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666
77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777
88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888
99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999

Here are the shelter costs and then FINDMAX, AREAMAX, and POPMAX for the small cell data which follows.

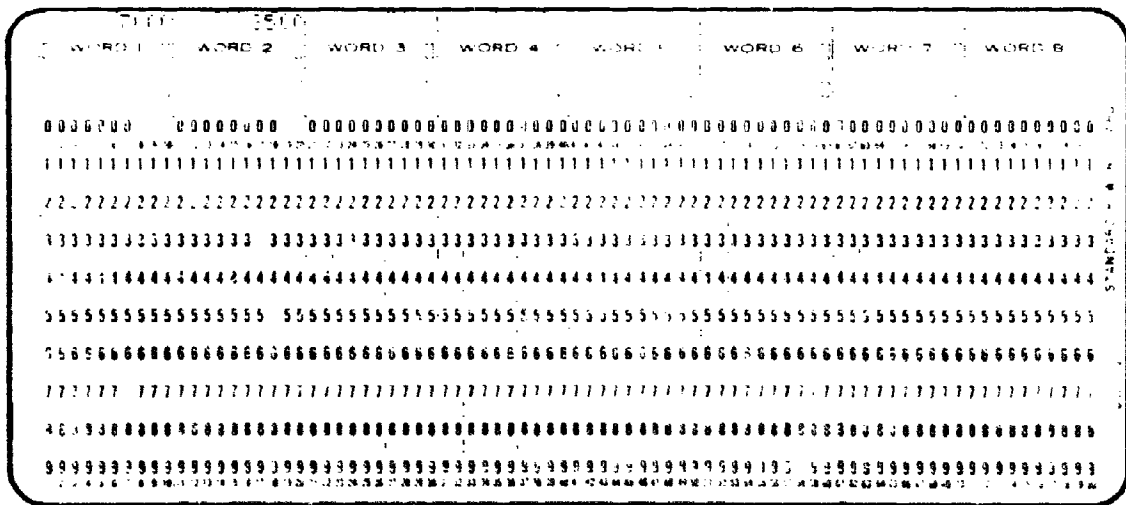


Here is the 12 cell data used in Appendix C ; this time with area in nautical miles. There are twelve cells and for each cell is read ICELL, RES, PKP, FIND, AREA, JDEF, and KDEF. The twelve are terminated by the blank card. This data is useful for checkout and exploratory runs.

All the previous data except the attack density levels is normally reset to desired lethal area, shelter, and cell data. The following cards are not normally reset and serve as parameters for all programs run.

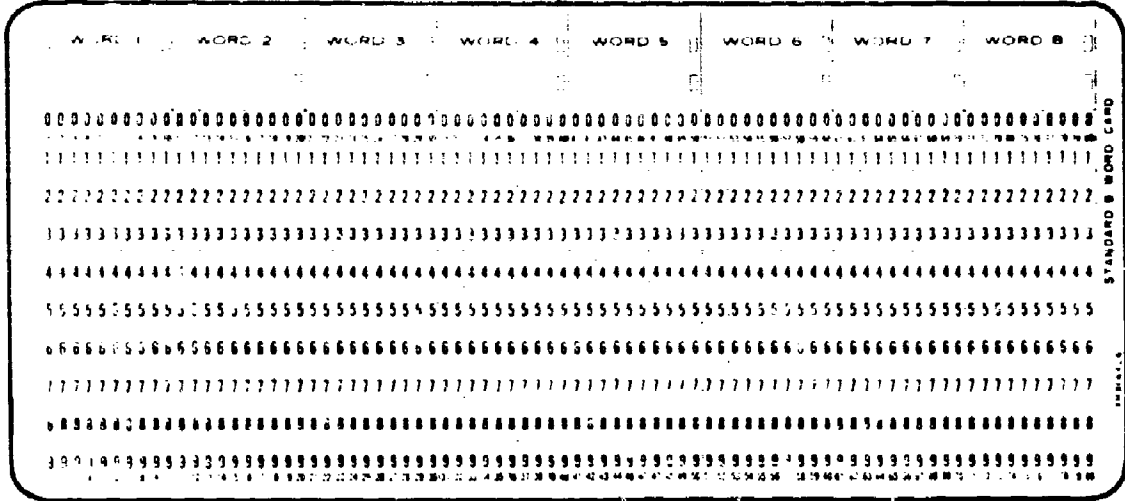


Here are the 12 attack levels (in equivalent megatons) used for evaluation and verification runs.

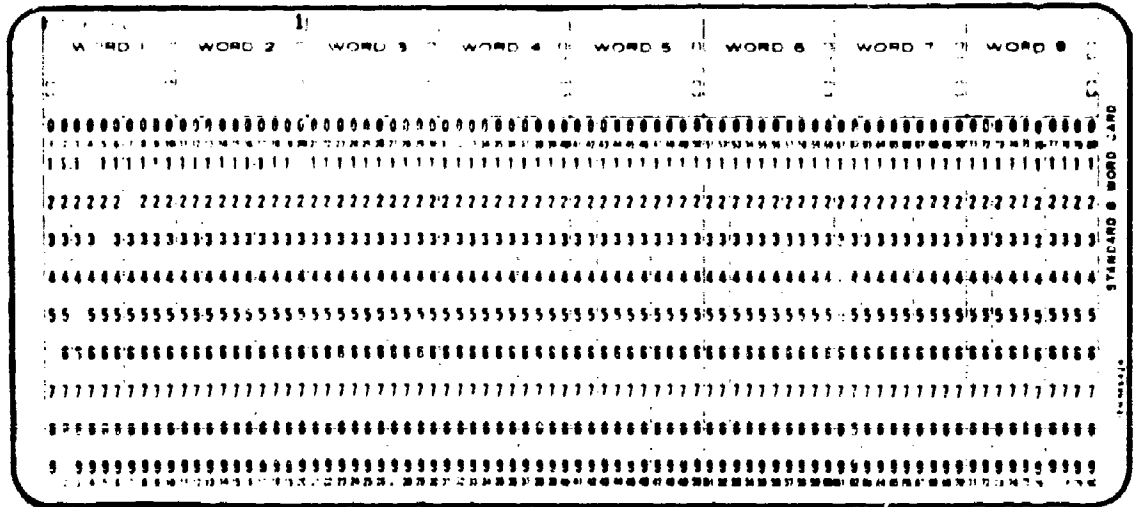


These are parameters MINDEN and MININD, the population density and industry

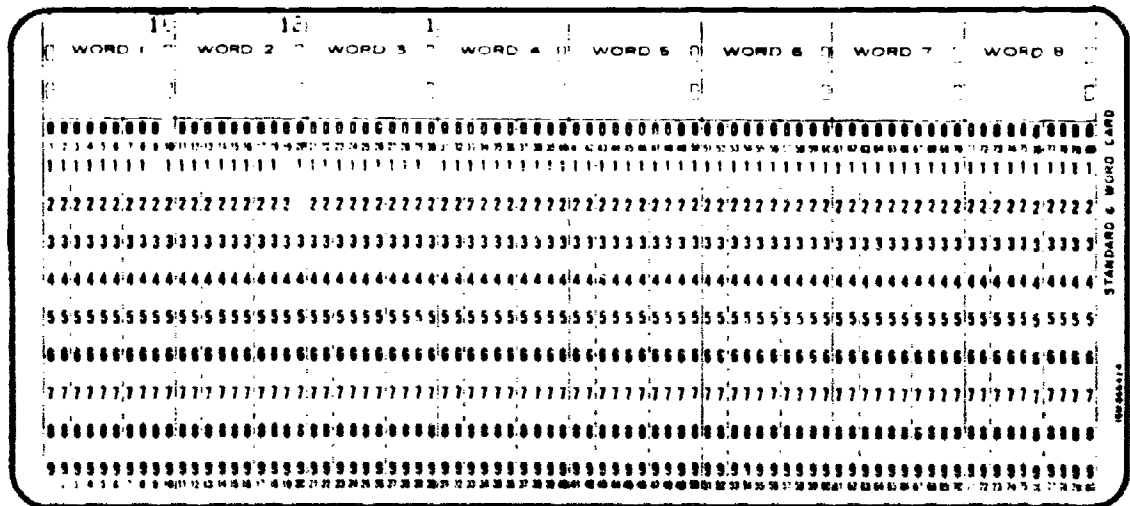
density cutoffs. When the full scale data is read, a cell is included explicitly (rather than in the tail cells) if either its population density is greater than 7,000 per sq. mi. or its industry density is greater than 3,500 per sq. mi.



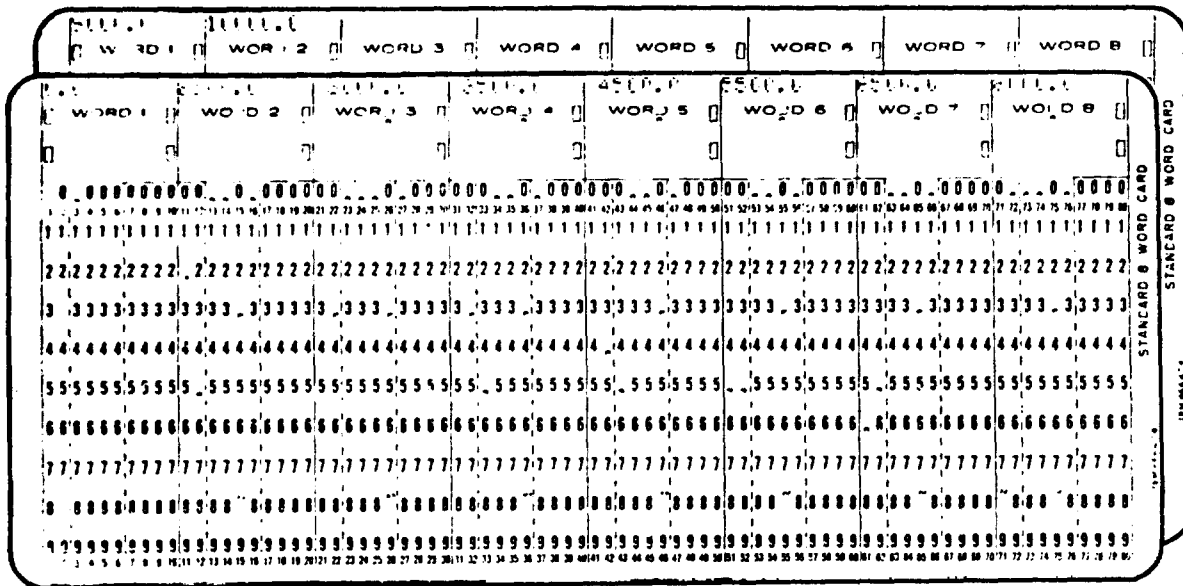
This card terminates the basic data deck. Its use is in the cell listing for option LIVEDATA -- if the card is blank all cells will be listed, if the card contains a city code (such as 22530005 for Washington, D.C.) only the cells in that city will be listed.



On this card FINAL-2 is a name for this shelter input deck. The 1 is a switch which here indicates that new hardness data will be read.



This card indicates 10 heights of burst and 12 hardness level. The 1 is a switch which indicates that lethal radii will be read in kilofeet with all heights of burst for a single hardness of a single pair of cards.



These are the 10 heights of burst in feet.

141.1	170.0	275.0	490.0	WORD 5	WORD 6	WORD 7	WORD 8
0000000000	0000000000	0000000000	0000000000	0000000000	0000000000	0000000000	0000000000
1111111111	1111111111	1111111111	1111111111	1111111111	1111111111	1111111111	1111111111
2222222222	2222222222	2222222222	2222222222	2222222222	2222222222	2222222222	2222222222
3333333333	3333333333	3333333333	3333333333	3333333333	3333333333	3333333333	3333333333
4444444444	4444444444	4444444444	4444444444	4444444444	4444444444	4444444444	4444444444
5555555555	5555555555	5555555555	5555555555	5555555555	5555555555	5555555555	5555555555
6666666666	6666666666	6666666666	6666666666	6666666666	6666666666	6666666666	6666666666
7777777777	7777777777	7777777777	7777777777	7777777777	7777777777	7777777777	7777777777
8888888888	8888888888	8888888888	8888888888	8888888888	8888888888	8888888888	8888888888
9999999999	9999999999	9999999999	9999999999	9999999999	9999999999	9999999999	9999999999

These are the 12 hardness levels, ranging from 5.4 psi to 490 psi.

141.1	170.0	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8
0000000000	0000000000	0000000000	0000000000	0000000000	0000000000	0000000000	0000000000
1111111111	1111111111	1111111111	1111111111	1111111111	1111111111	1111111111	1111111111
2222222222	2222222222	2222222222	2222222222	2222222222	2222222222	2222222222	2222222222
3333333333	3333333333	3333333333	3333333333	3333333333	3333333333	3333333333	3333333333
4444444444	4444444444	4444444444	4444444444	4444444444	4444444444	4444444444	4444444444
5555555555	5555555555	5555555555	5555555555	5555555555	5555555555	5555555555	5555555555
6666666666	6666666666	6666666666	6666666666	6666666666	6666666666	6666666666	6666666666
7777777777	7777777777	7777777777	7777777777	7777777777	7777777777	7777777777	7777777777
8888888888	8888888888	8888888888	8888888888	8888888888	8888888888	8888888888	8888888888
9999999999	9999999999	9999999999	9999999999	9999999999	9999999999	9999999999	9999999999

These are the lethal radii for a one-megaton weapon for 5.4 psi hardness and for the 10 different heights of burst. The lethal radii for the other 11 hardnesses follow in the same pattern, 10 heights of burst for a single hardness on a pair of cards, then moving to the next hardness. These are shown in Table I, and they follow immediately after the preceding two cards.

Table I . Remainder of Lethal Areas

10.2	11.4	11.6	11.95	12.7	13.6	14.4	14.4
9.2	7.6						
7.2	7.8	7.9	8.05	8.55	9.4	9.6	4.9
2.6	0.0						
8.6	9.5	9.8	10.1	10.9	11.8	12.4	7.5
6.0	4.3						
5.6	5.8	5.9	6.0	6.2	6.35	3.7	0.0
0.0	0.0						
5.1	5.3	5.4	5.5	5.7	4.3	2.4	0.0
0.0	0.0						
4.53	4.72	4.8	4.94	5.13	3.04	0.0	0.0
0.0	0.0						
3.15	3.55	3.65	3.2	1.2	0.0	0.0	0.0
0.0	0.0						
2.9	3.25	3.3	2.5	0.0	0.0	0.0	0.0
0.0	0.0						
2.7	3.0	2.9	2.0	0.0	0.0	0.0	0.0
0.0	0.0						
1.95	1.5	0.9	0.0	0.0	0.0	0.0	0.0
0.0	0.0						
1.75	1.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0						

WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9

STANDARD 8 WORD CARD
100 045424
100 045424

Here it is specified that only two of the 10 heights of burst will be scanned for attack optimization - numbers 1 and 6, or 0 and 5500 feet.

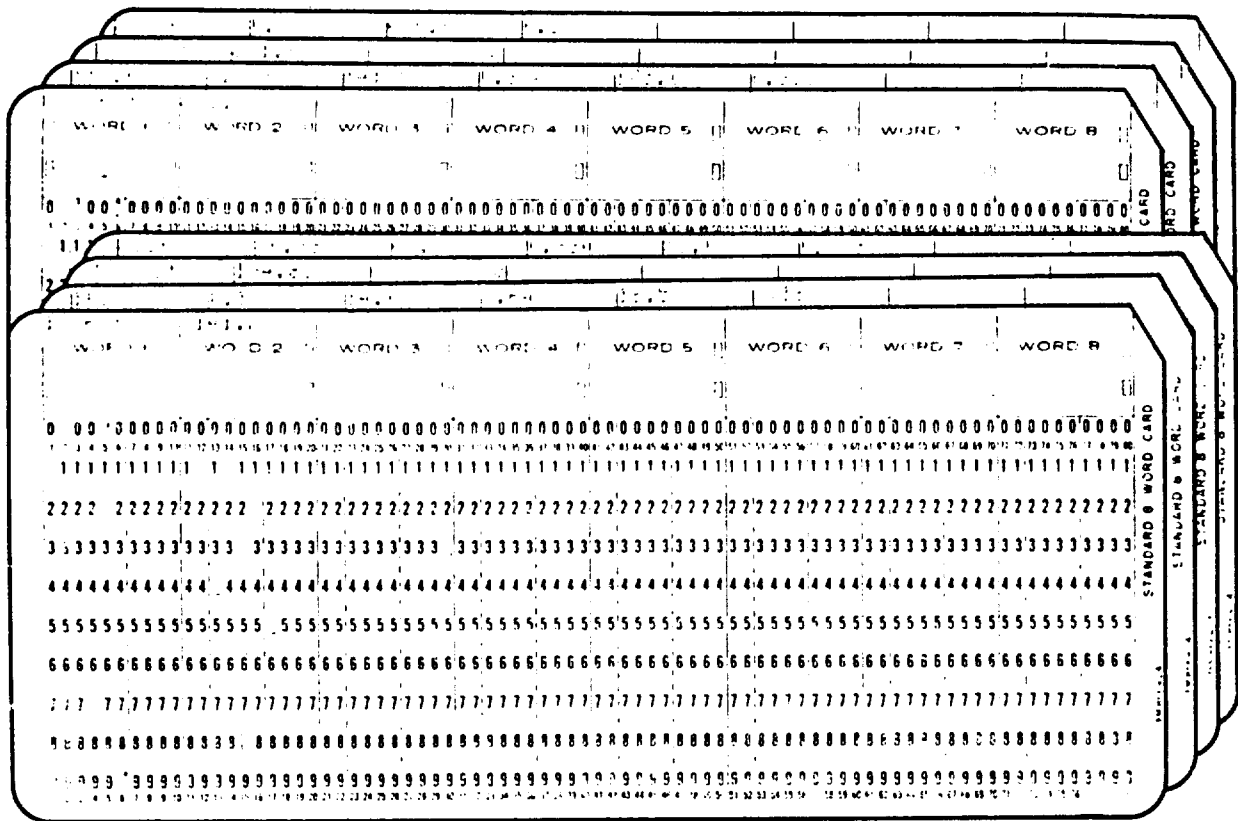
WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9

STANDARD 8 WORD CARD
100 045424
100 045424

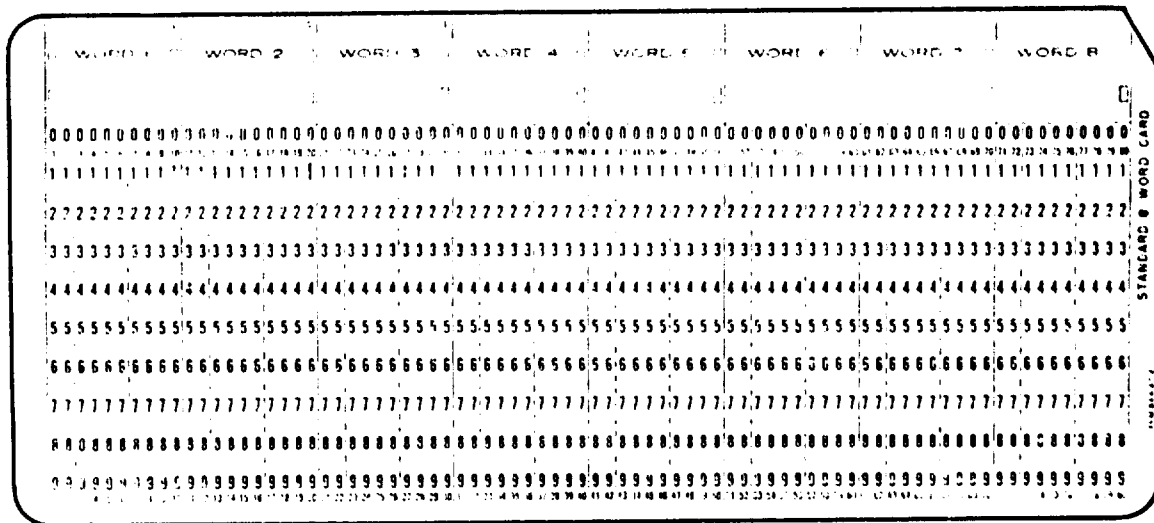
Here are indicated 5 different shelters (National Fallout Shelter Survey, 10 psi blast, 30 psi blast, 100 psi blast, and 300 psi blast) and 15 different defense options or shelter mixes.

WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8
00	00	00	00	00	00	00	00
11	11	11	11	11	11	11	11
22	22	22	22	22	22	22	22
33	33	33	33	33	33	33	33
44	44	44	44	44	44	44	44
55	55	55	55	55	55	55	55
66	66	66	66	66	66	66	66
77	77	77	77	77	77	77	77
88	88	88	88	88	88	88	88
99	99	99	99	99	99	99	99

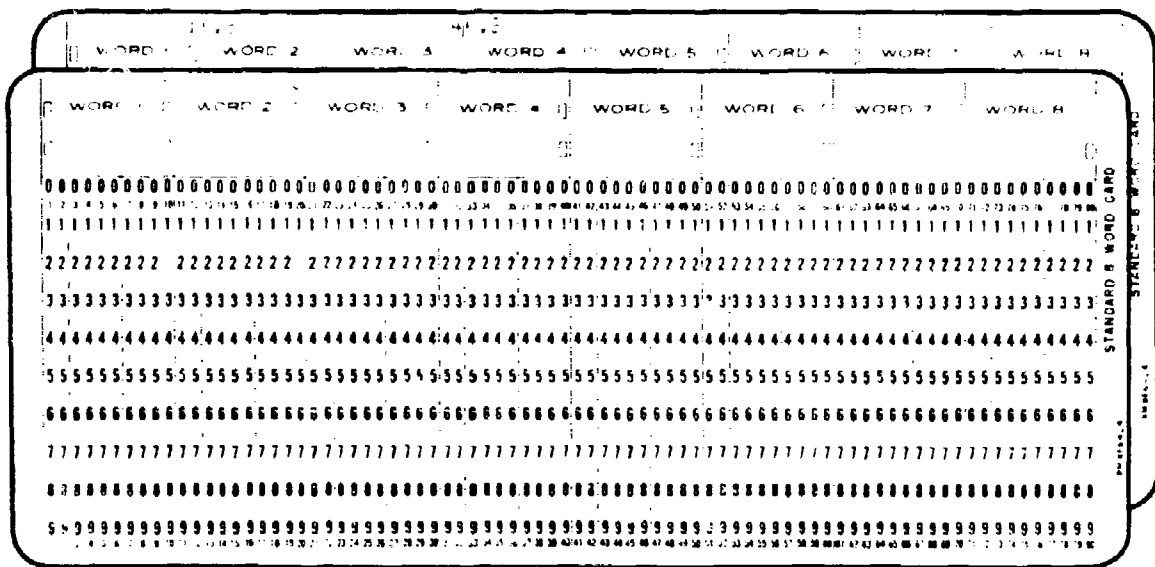
Here is the definition of the hardness spread for the NFSS shelter. The name of the shelter is NFSS, the cost is zero, and the number of hardnesses is 3. The second card specifies the 3 hardnesses and the fraction of the shelter assumed to have each hardness; here one-third each at 5.4 psi, 10.0 psi, and at 19.5 psi. The hardness specifications for the remaining four shelters follow. Note that for each of these shelters 10 psi, etc., is just a name for the shelter, and that the characteristics of the shelters are specified by the hardness spread and thence by the lethal radii previously read.



Here the shelter mix specifications begin. The first defense option is specified by:



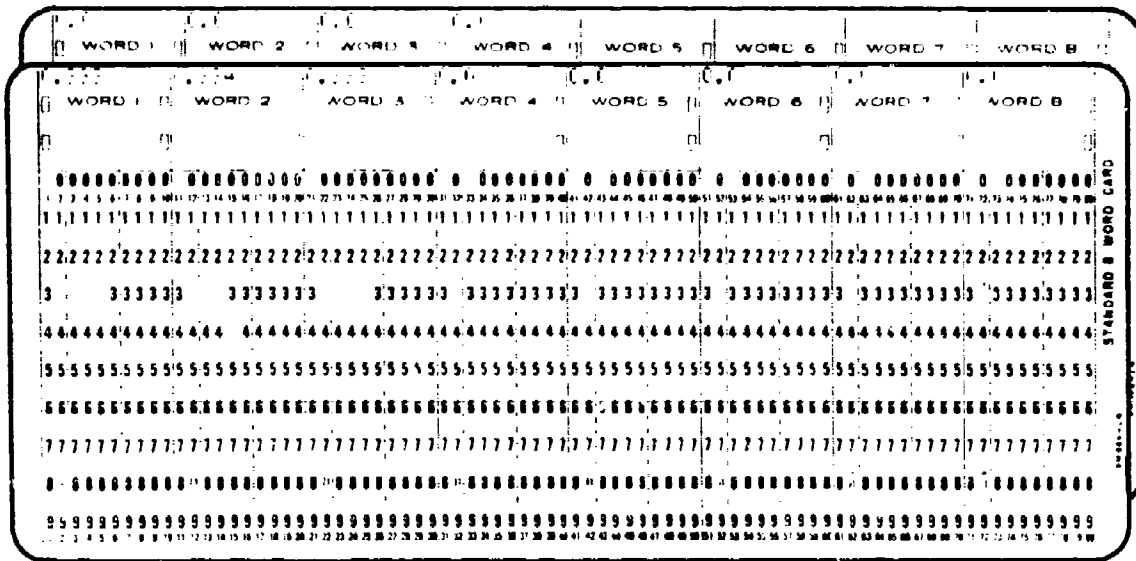
The first number 1 is the index of the defense option. The second number is the number of shelters in the mix for that defense option. If there is only one shelter in the mix, then the third number specifies which of the shelters just read is the one shelter type. Here for the first defense option there is one shelter in the mix and that shelter is the NFSS type, shelter number 1.



For the second defense option there are 2 shelter types in the mix. Whenever the number of shelters in the mix is more than one, a second card is read which specifies the mix. Here the mix is 0.8 of shelter type 1 (NFSS) and 0.2 of shelter type 4 (100 psi). The shelter mixes for the other 13 defense options are specified similarly, as shown in Table II; they follow immediately after the above two cards.

Table II. Remainder of Shelter Mixes

3	2			
10.5		20.5		
4	2			
10.5		30.5		
5	1	2		
6	3			
10.3		30.4	40.3	
7	2			
20.5		30.5		
8	4			
10.3		30.2	40.4	50.1
9	2			
20.8		50.2		
10	1	3		
11	2			
10.3		40.7		
12	2			
30.5		40.5		
13	1	4		
14	2			
40.5		50.5		
15	1	5		

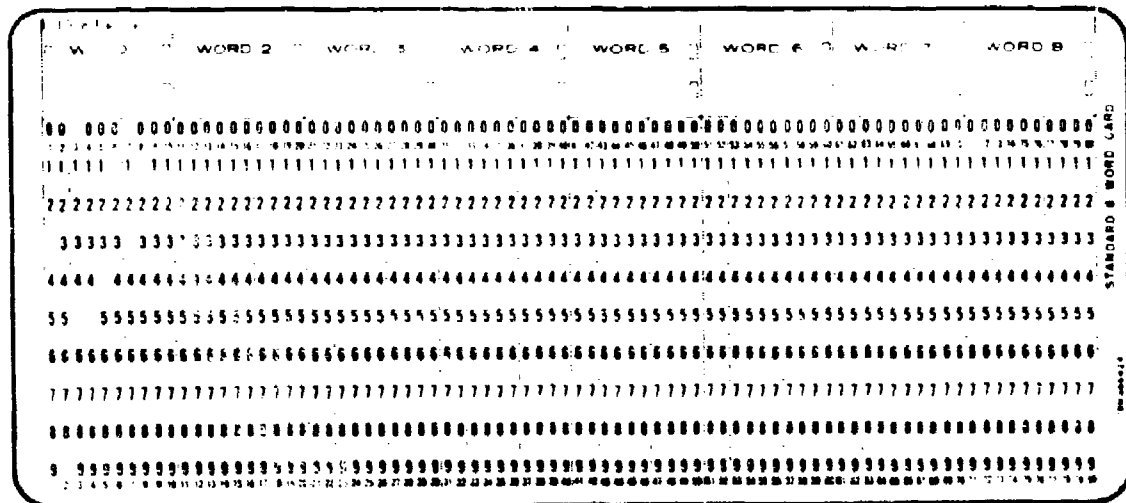


The last two cards define the hardness spread for industry by specifying the fraction of the industry which has each of the 12 hardness levels. Here industry is assumed to have the same hardness spread as the NFSS shelter type, or one-third at each of 5.4 psi, 10 psi, and 19.5 psi. These two cards terminate the data sequence for this INPUT option. Control now returns to the option selector for whatever type of run is desired.

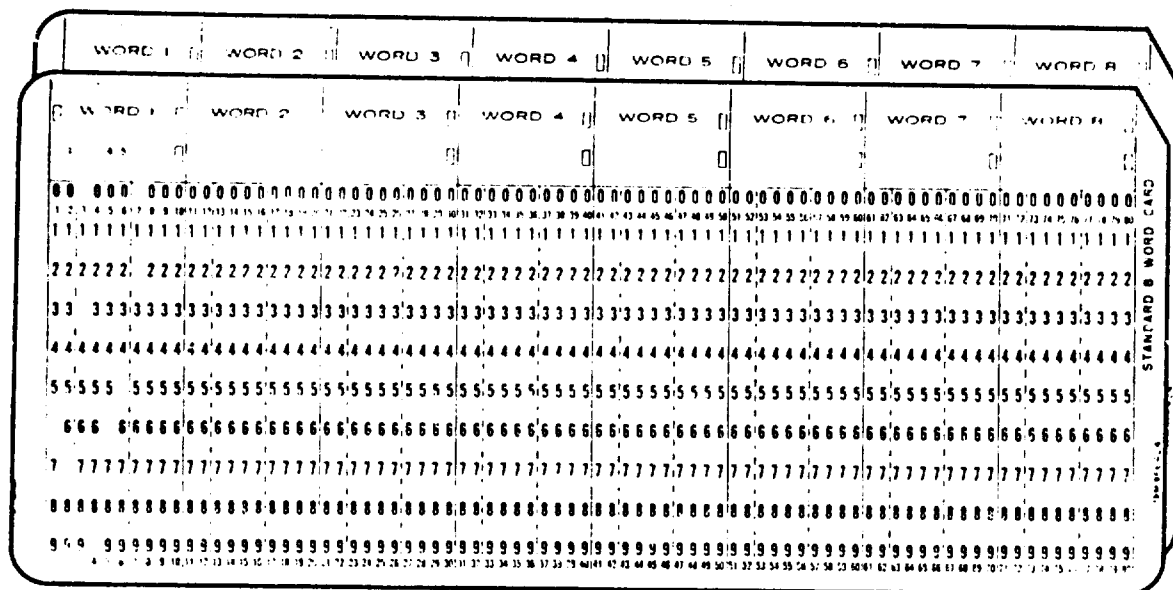
C. SAMPLE RUN DECK #1

The purpose of this run is to generate a defense using the mixed lambda technique and assuming a mixed attack objective and then to evaluate the defense twice for two different attack objectives. The defense in this sample deck is found with IDPOP = 1 and IAPOP = 2 with the budget level at 10.0 billion.

The following data sequence is put behind the basic data deck and the shelter input deck:



This card causes the program to read the population data tape. Control is then returned to the option selector.



These two cards set $IAPOP = 2$ (the program sets $IDPOP = 1$).

The first card sends control to the portion of the driver which examines the subsequent card for one of the parameters $IDPOP$, $IAPOP$, $IAOB$, and $IFILL$. If the card is one of those parameters, then the parameter is set to the value in column 20 and the next card is examined. If it is in the list, the value is set; otherwise it is assumed to be the next card for the option selector part of the program.

WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9

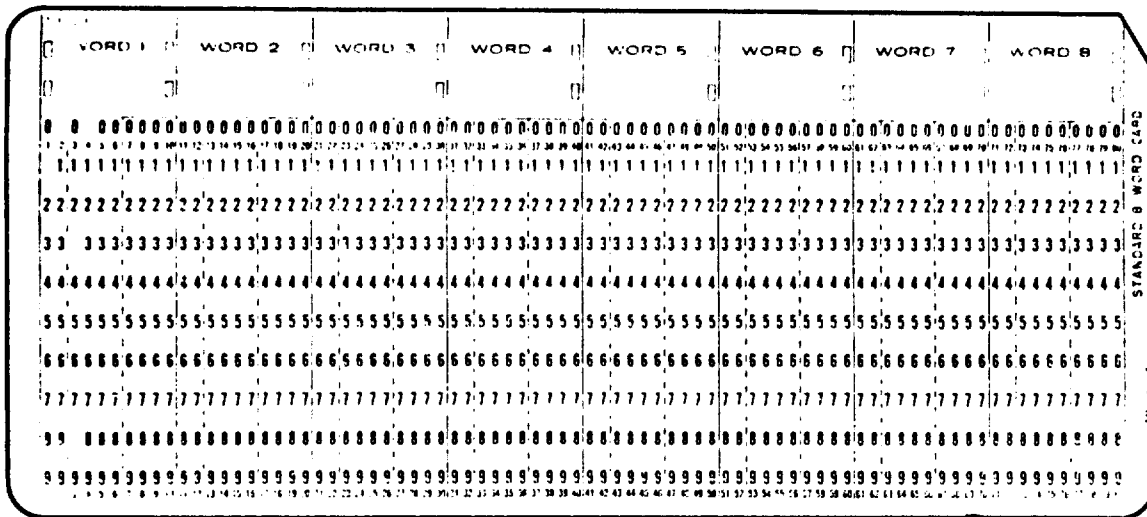
STANDARD 8 WORD CARD
IBM 8042

This card sets a switch which, after the defense is generated, causes the defense options selected to be punched onto cards for every cell. Also, the parameters and shelter data for the defense are punched. This punched deck is then a suitable input for further evaluation runs as in Sample Deck # 3. After this card, control returns to the option selector.

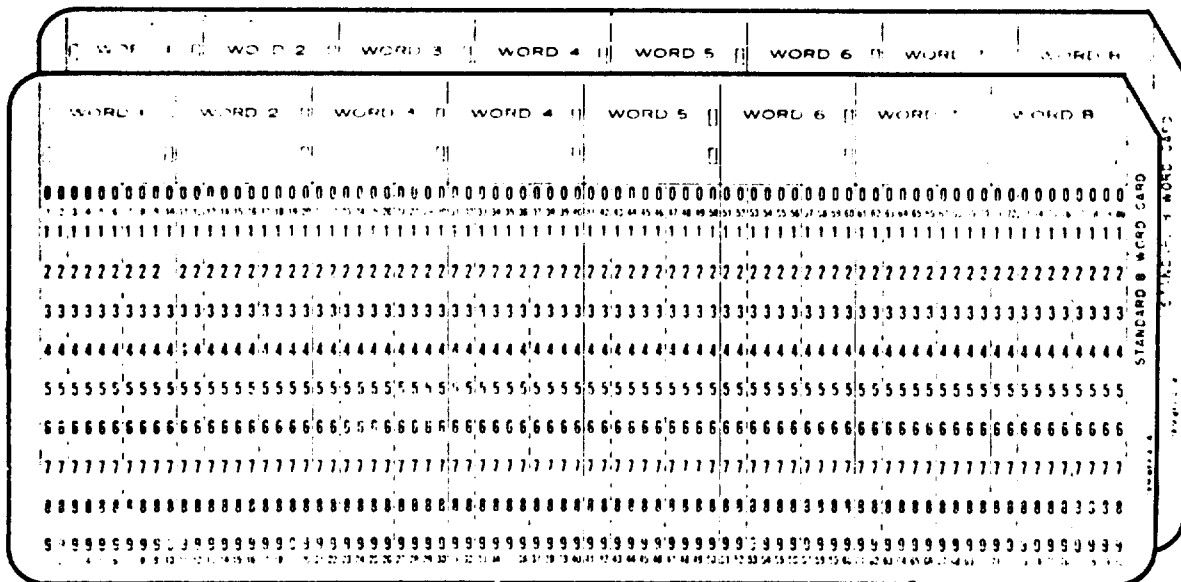
WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9

STANDARD 8 WORD CARD
IBM 8042

This card makes the program do run type SIMPLE, which is the basic run type to generate a defense, and, if desired, to evaluate it. The entire flexibility of the SIMPLE run type is described in the data sequence definition section; the sequence presented here is one possible type which has considerable usefulness.



This card sets the target cost for the defense (TCOST) to 10 billion dollars.

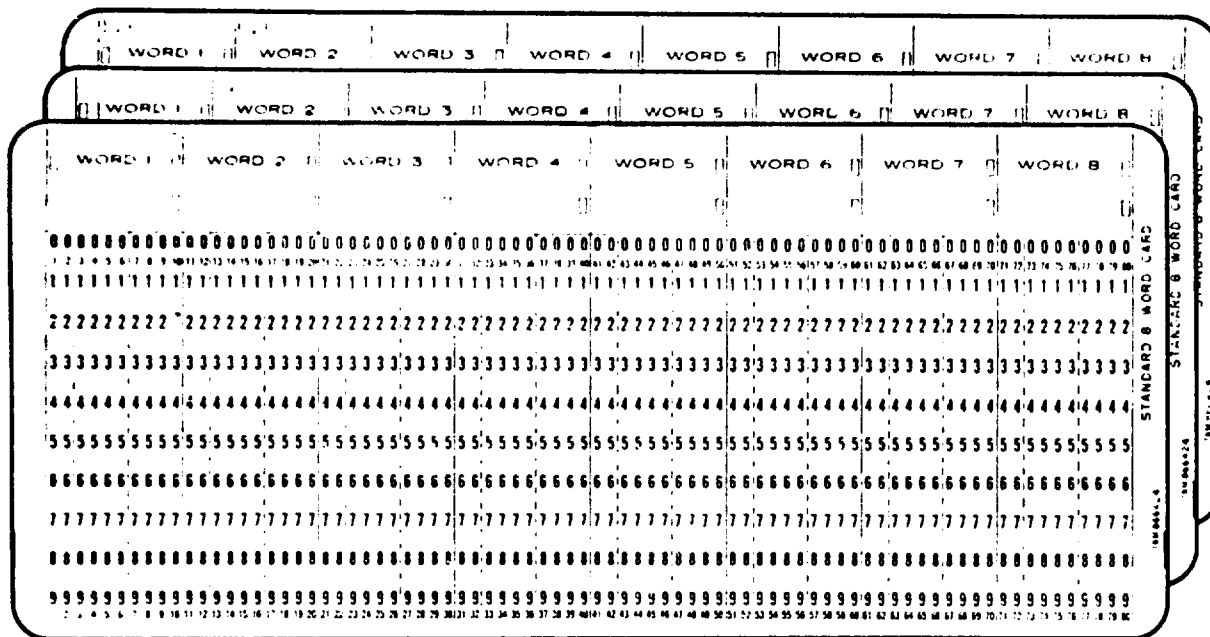


These two cards input the lambdas to be assumed by the defense (DLAM).
 The first card indicates the number of defense lambdas; the second card contains
 the lambdas, one lambda per word on the card.

WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9

STANDARD 8 WORD CARD
UNIVERSITY MICROFILMS

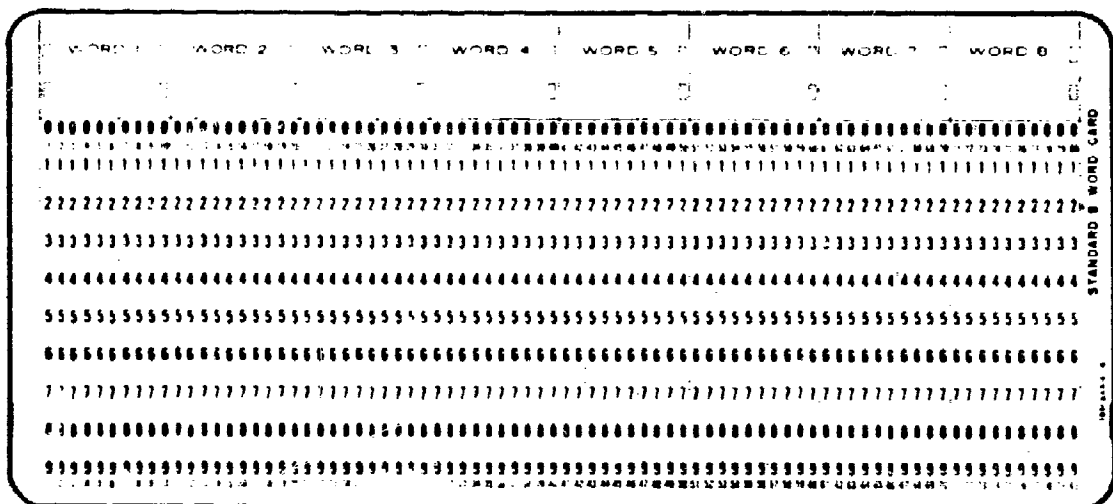
These two cards set the weights for the lambda just read. The first card may be any non-zero integer (if it is zero the program weights all lambdas equally and the second card will not appear). The second card contains the weights for the lambdas, the first for the first lambda read, etc. The mixed lambda combination just read is 78850.0 weighted at 0.6 and 7885.0 weighted at 0.4.



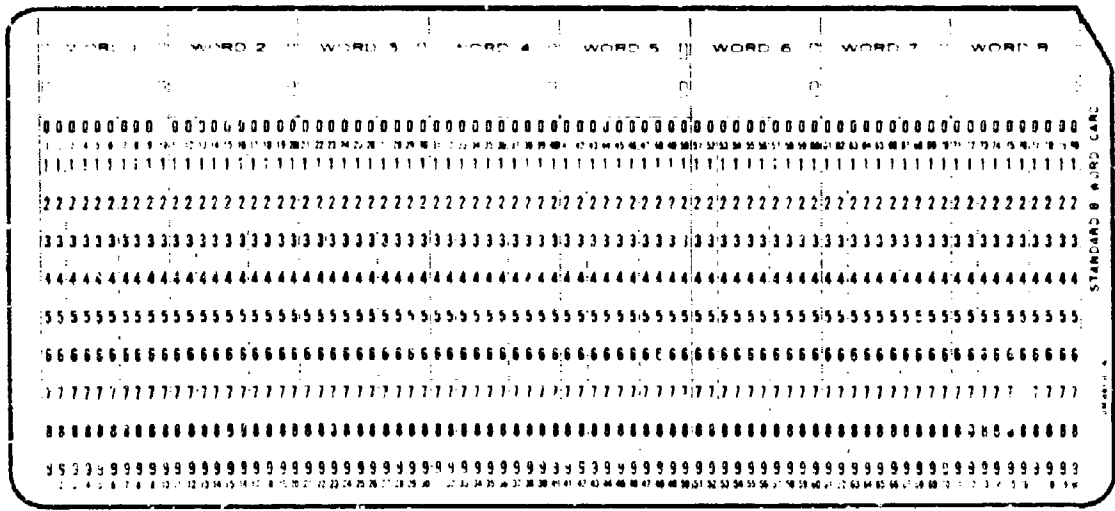
These three cards read the attack objectives used in this run. The first card is the number of attack objectives, NAOB. If $NAOB = 1$, then the run is performed using the attack objective set previously (the program sets this to pure population in the beginning). Here, two attack objectives are read. The first card gives the population multipliers for the two attack objectives; the second card gives the industry multipliers. The first attack objective is a 50/50 attack objective (equal weightings of population and industry); the second is pure population. The first weight on industry is 1.077 and not 0.5 because there is 2.154 times more total population than total industry; hence, that weight is $0.5 \times 2.154 = 1.077$.

When the program generates the defense it will assume some attack objective for the attacker. Since the weights on the attack objectives were set to 1.0, 0.0, 0.0, 0.0, ..., the defense places all its weight on the first attack objective, the 50/50 mix. Hence this defense is generated assuming a 50/50 attack objective. The second attack objective is not used in the generation of this defense.

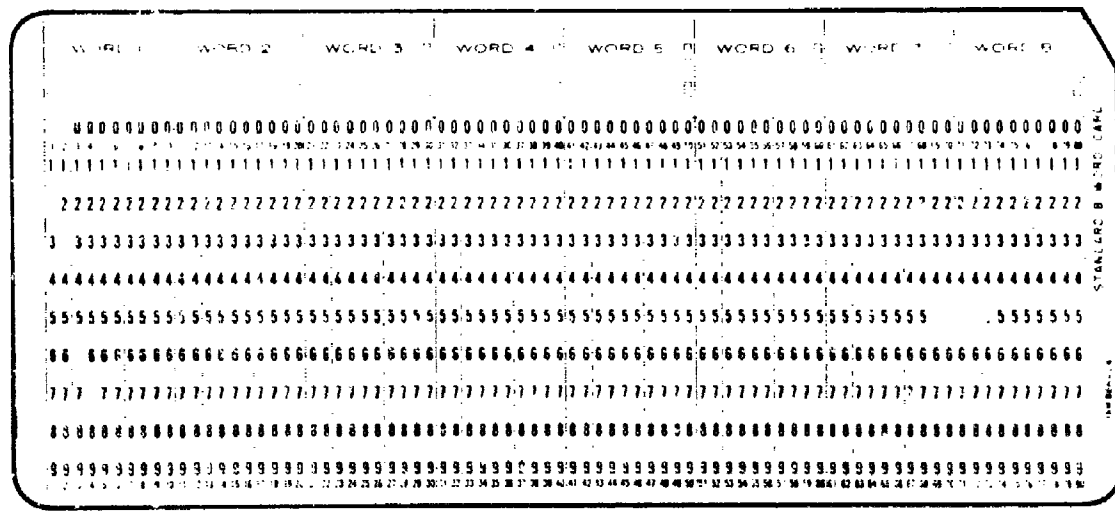
At this point in the data sequence the program will compute the defense and punch it onto cards for future use. The next card determines whether the program will immediately evaluate the defense:



If this card is anything but the key word NOEVAL, then the program will immediately evaluate the defense it has generated for all attack objectives, one at a time. There it evaluates for two attack objectives, the first 50/50, and the second pure population.



This card is a switch as to whether the program is to read a different set of attack objective weights for the defense. Here the zero answers negatively, and this card terminates the SIMPLE sequence. If the card were any other number then the program would read new attack objective weights, generate the defense and evaluate it. Control now returns to the option selector.



This card is the normal termination for the program, taking the program out of the option selector loop.

Figure 14 shows the entire data sequence for this run type, with the basic data deck and the shelter input deck indicated at the front.

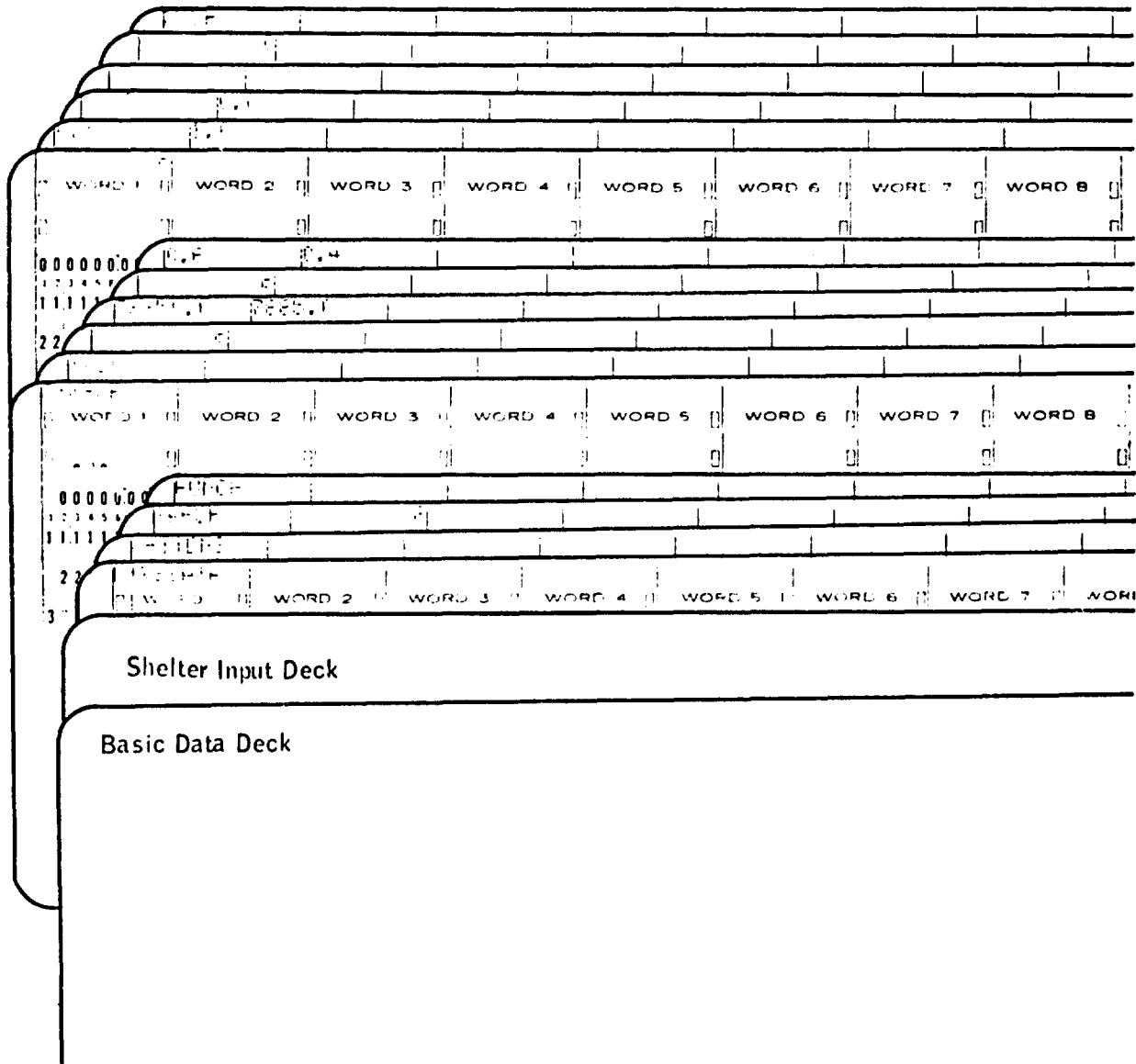
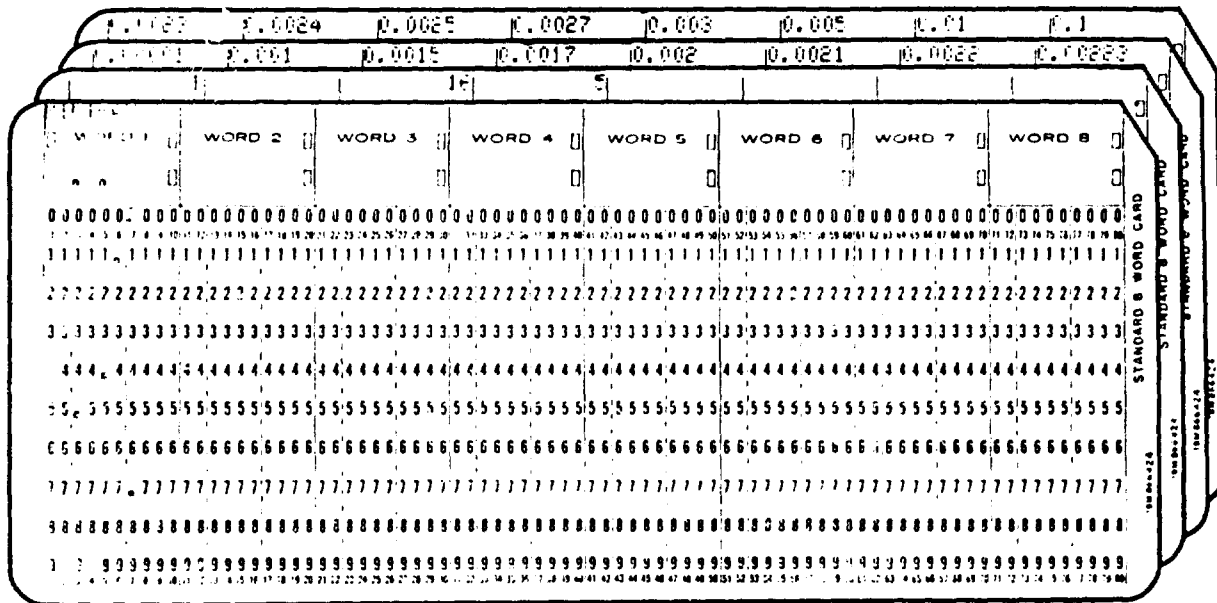
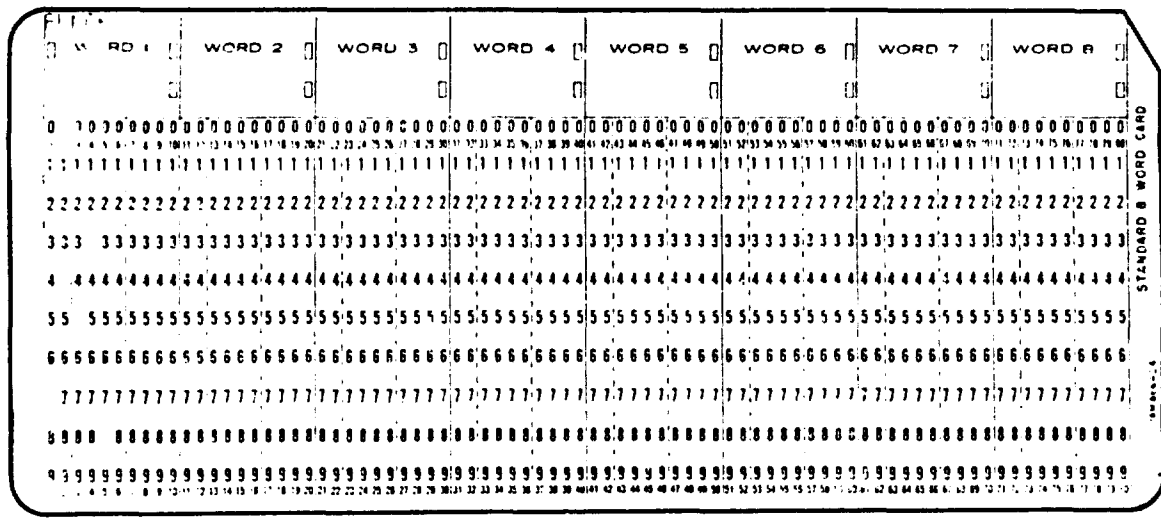


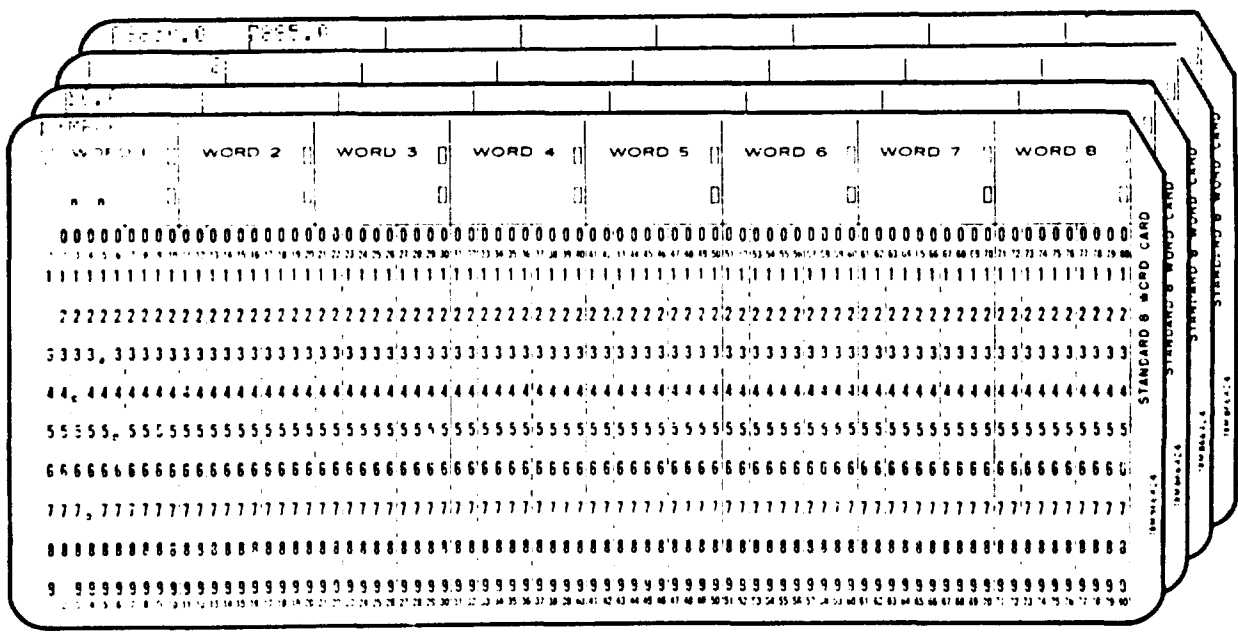
Figure 14. Entire Data Sequence for Sample Run Deck #1



The MINIMAX card here enters that option for setting up the minimax procedure. The second card sets the switch MINIMAX to 1 so the defenses will be chosen according to that procedure. The next number means that 16 new mus will be read on the following two cards. The last number sets NNEW = 5; NNEW is the number of mus used after the first pass in DEFOPTG -- the value 5 helps speed the computation better than leaving it at 50 as it would otherwise be set. The 16 mus read next are tailored to fit the 10 billion budget; they need not be selected thus, but it will help computation time to do so.



Again, this card causes the defense generated to be punched out for saving.



The SIMPLE option follows the same data sequence as always and here sets TCOST = 10 billion, and reads the two lambdas to be used.

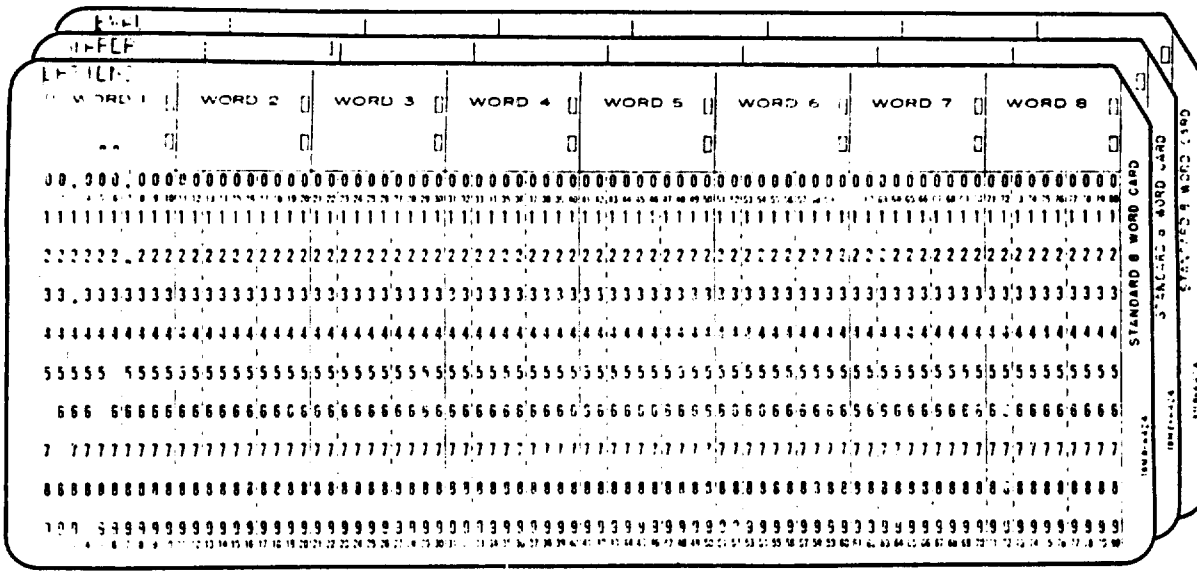
WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111
22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222
33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333
44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444
55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555
66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666
77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777
88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888
99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999

This card causes WDLAM to be set to 0.5 for each lambda (equal weight). Even though the lambda weights will not be used, the data sequence must be maintained, and this is the simplest method here.

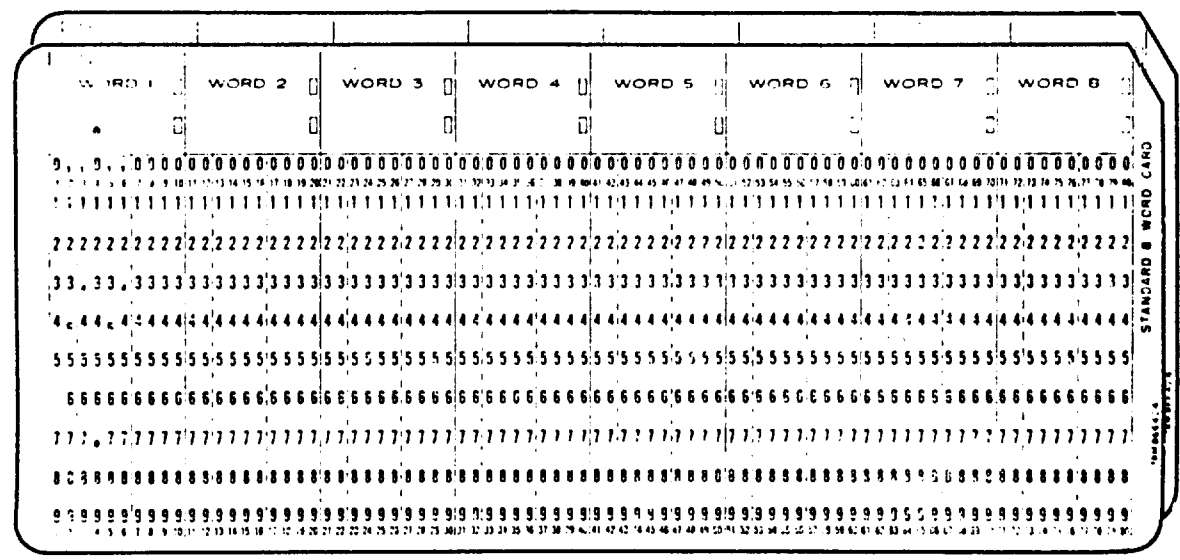
WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8	WORD 9
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111
22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222
33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333
44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444
55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555
66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666
77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777
88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888
99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999

The same three attack objective cards as for Run Deck #1.

At this point the program will generate the defense using the minimax procedure and assuming a 50/50 attack objective. It will then be punched onto cards for future use.



These cards change the time of attack to night (IAPOP = 2) and cause a new evaluation. The attack objective remains at pure population, and all other parameters stay fixed.



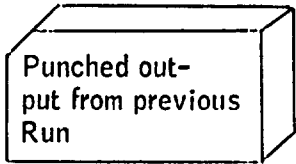
The OUTPUT card obtains the printout described in the OUTPUT section, and, finally the STOP card terminates the run.

E. SAMPLE RUN DECK #3

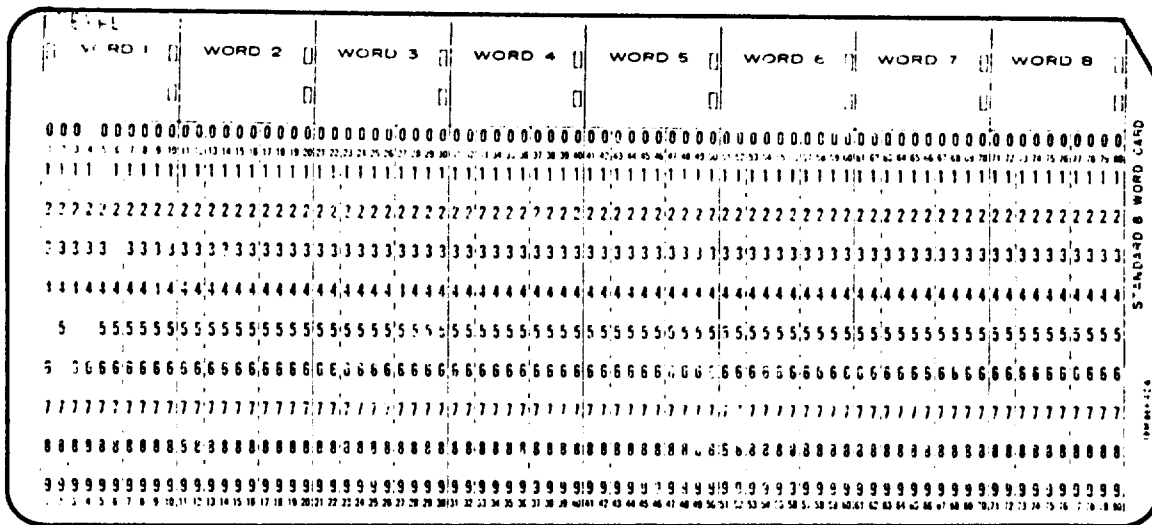
The purpose of this run is to read in a previously generated defense and to evaluate it for a 50% alerted nighttime population in a pure population attack. The defense to be evaluated is contained on a punch deck, exactly as it came out from some previous run. Since shelters are stored on that output deck, we need have only the basic data deck in front of the following cards:

DEFSPEC	WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8
LIVEDATA								
00	000	000	000	000	000	000	000	000
11	111	111	111	111	111	111	111	111
22	222	222	222	222	222	222	222	222
33	333	333	333	333	333	333	333	333
44	444	444	444	444	444	444	444	444
55	555	555	555	555	555	555	555	555
66	666	666	666	666	666	666	666	666
77	777	777	777	777	777	777	777	777
88	888	888	888	888	888	888	888	888
99	999	999	999	999	999	999	999	999

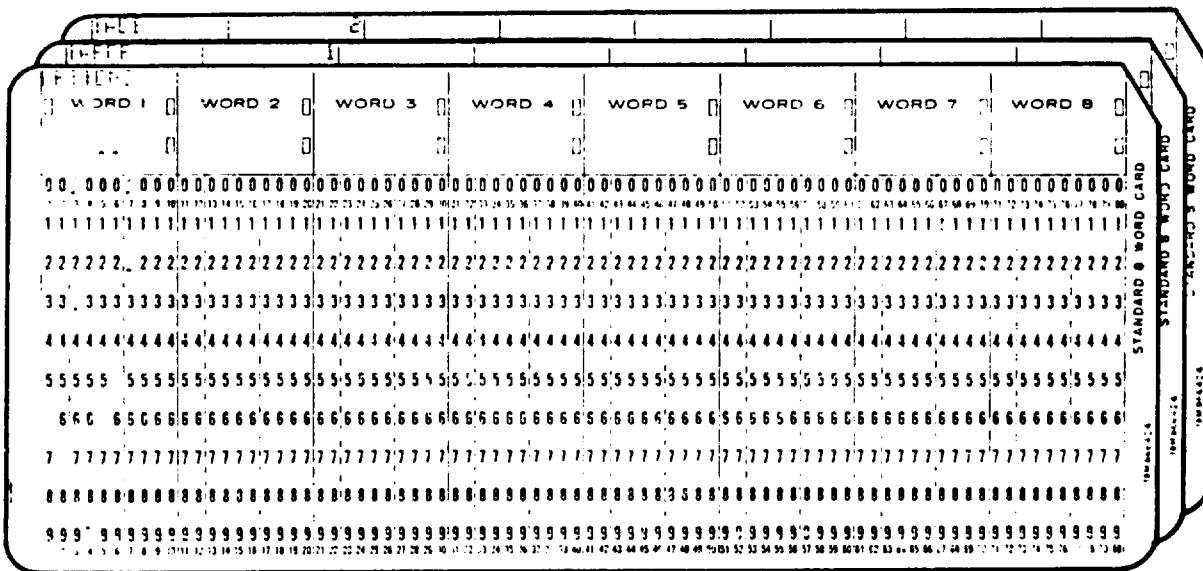
The LIVEDATA card brings in the population data, and the DEFSPEC card causes the program to go to the portion which reads the punch deck next:



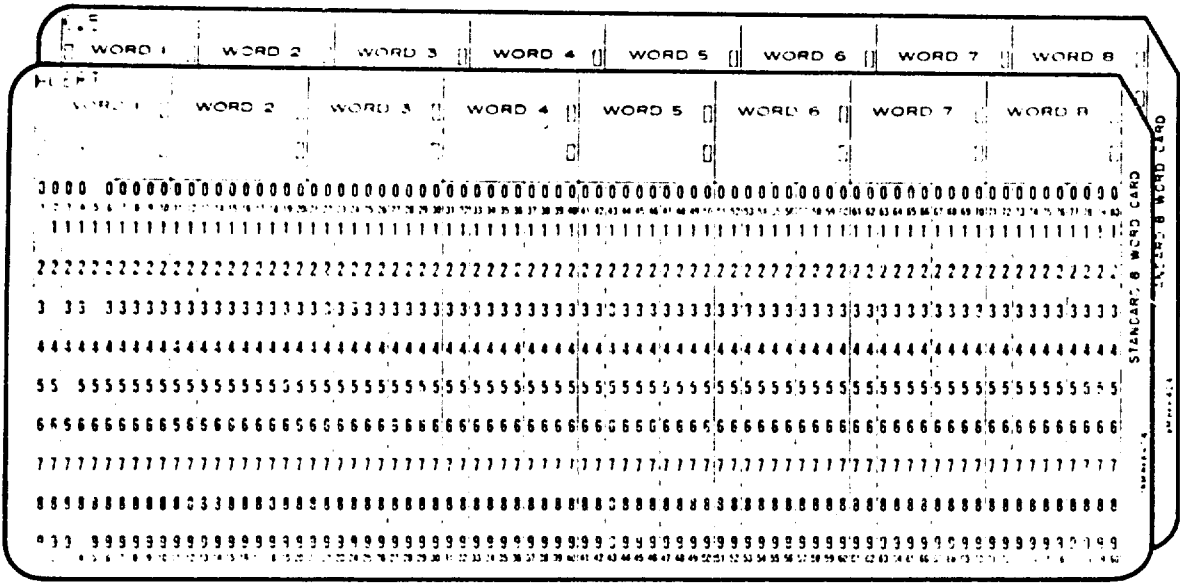
For a description of what is contained in the punch deck, see the data sequence definition for DEFSPEC.



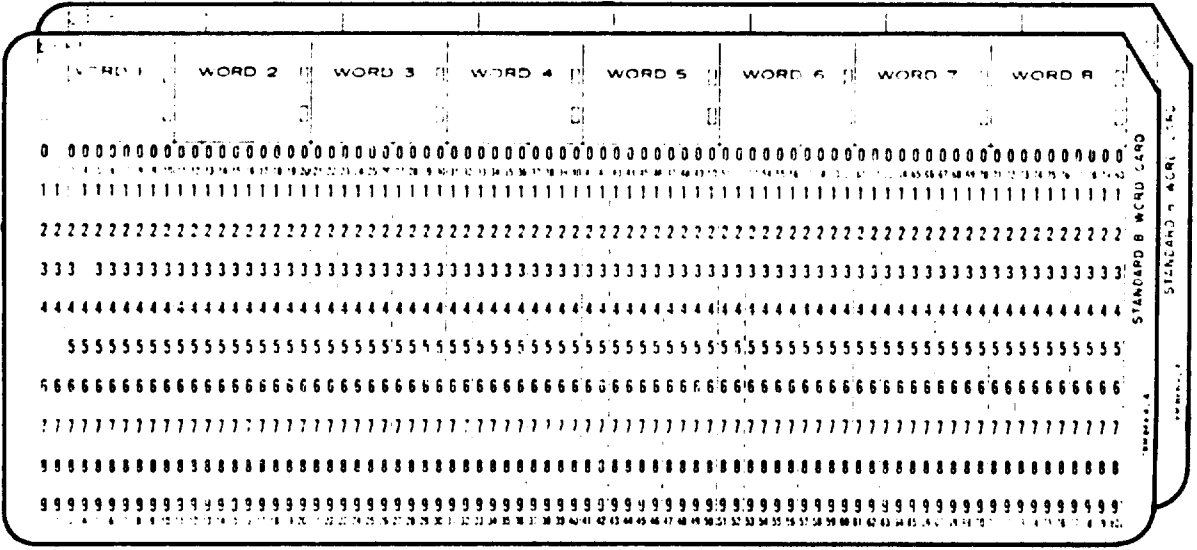
This NOEVAL card causes DEFSPEC to not evaluate the defense; otherwise, the defense would be evaluated with parameters as they were set.



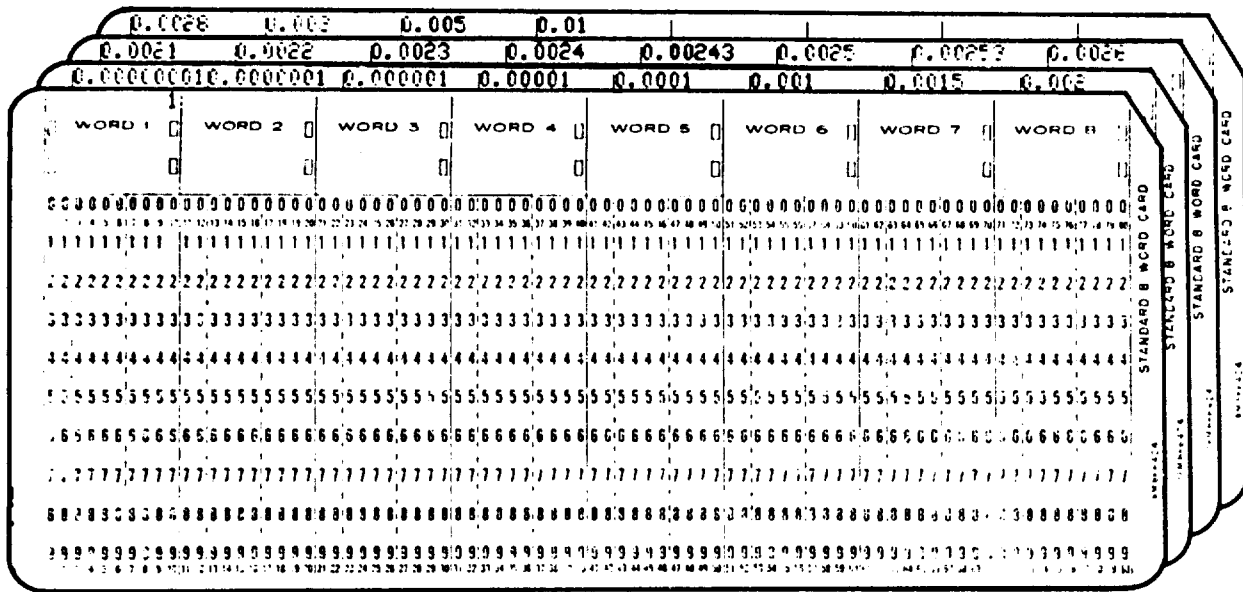
These cards set the time of attack to night and, we will suppose, the attack objective to pure population (if the defense we are evaluating is like the previous sample decks, this would be pure population). Although the punch deck saves most parameters and, indeed, saves the POPMULT and VINDMULT arrays (and, hence, the attack objectives), it does not save the particular attack objective, IAOB, leaving that to be set before the evaluation.



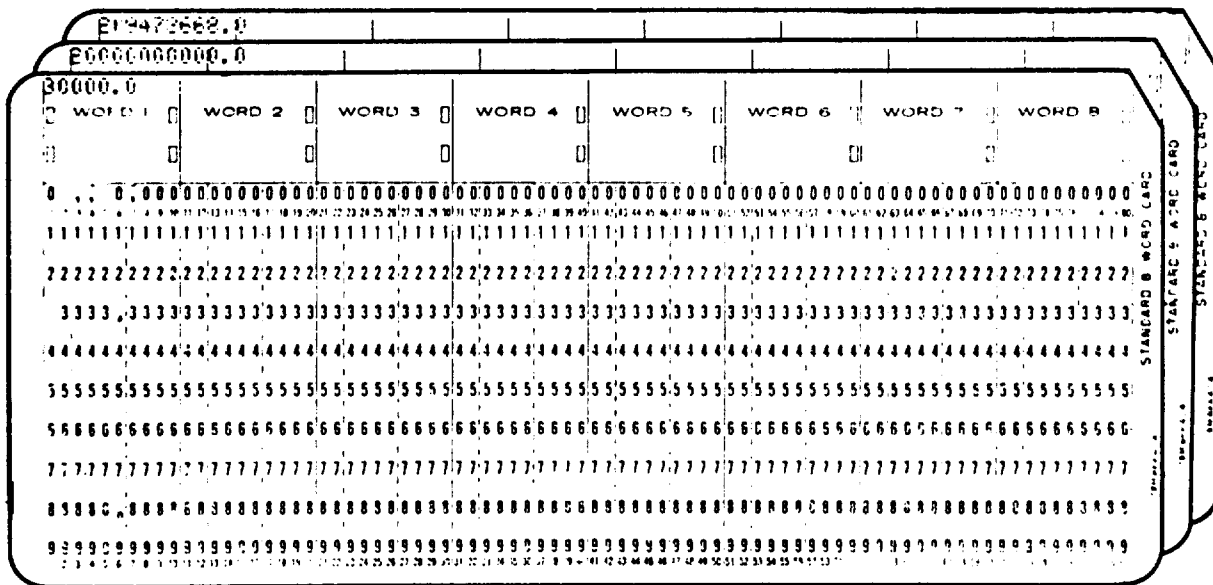
These two cards set up the fractional filling mode and define the fraction alerted - 0.5. All parameters are now set for the evaluation.



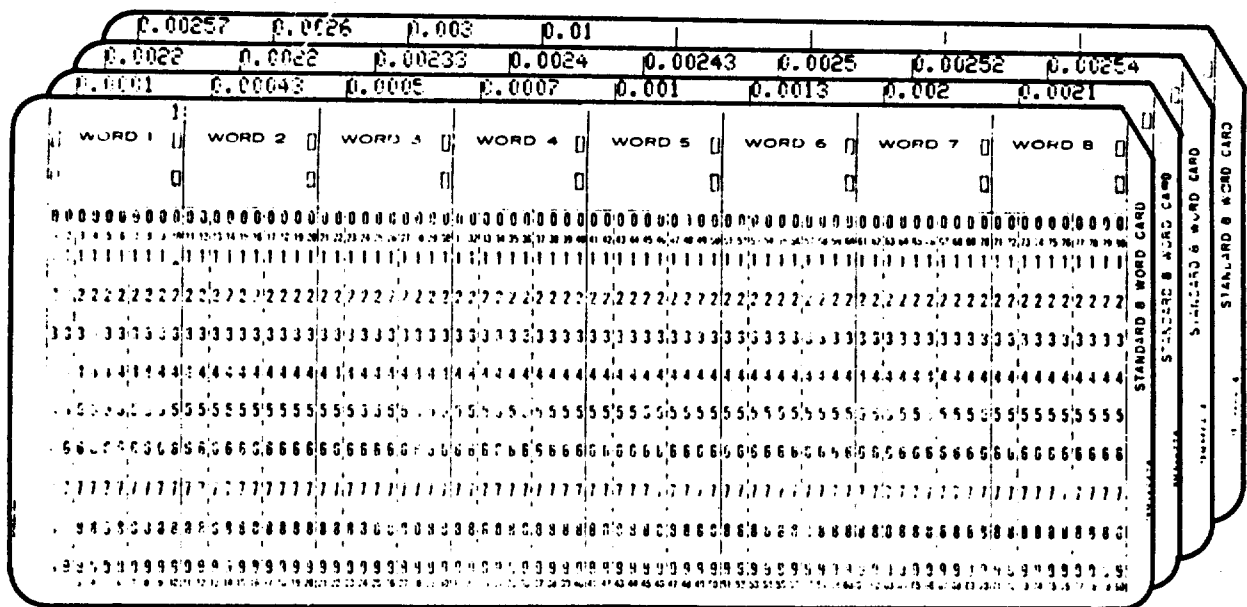
As a last step we evaluate the defense using the new parameters and conclude the run.



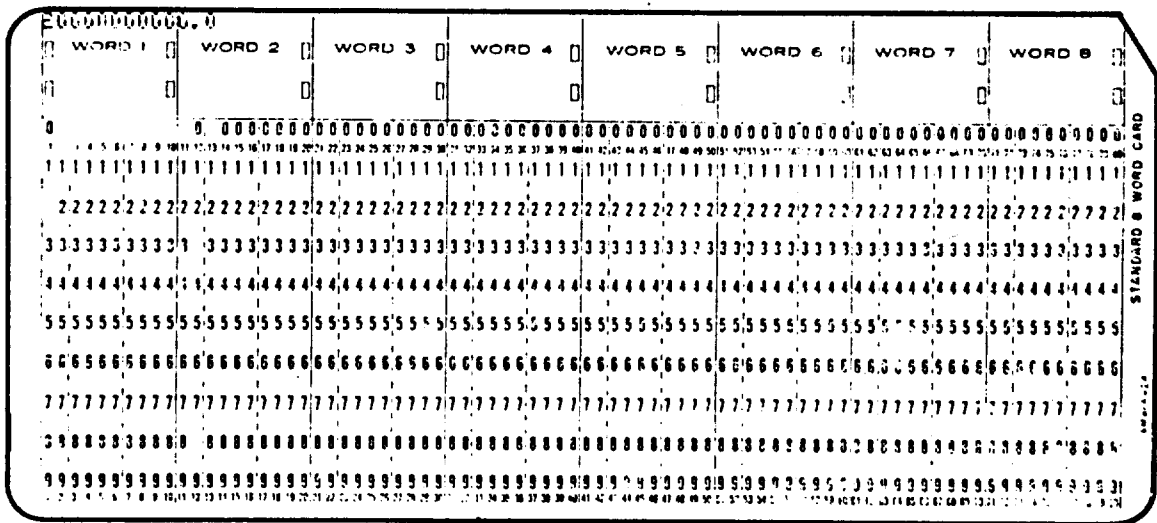
These cards read in the new mus. Notice that although the entire range is six orders of magnitude, a number of the mus are concentrated in what is the expected region for the \$20 billion level.



Here the initial lambda is 30,000, TCOST is \$20 billion, and HSAT is 209,473,668, the total daytime population (since no attack objective was set, it remains at the program setting which is pure population attack; hence, the total payoff is the total population).



Notice that this new set of mus does not cover the entire range that the previous set does. That is because from the previous computation we know what region the mus will fall, so a smaller range can be used, enabling better cost matching.



TCOST - \$20 billion again.

2000.0		4000.0		6000.0		8000.0		10000.0									
2034.69		3000.0		3500.0		4000.0		4500.0		5000.0		6000.0		7000.0		8000.0	
WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8	WORD 9	WORD 10	WORD 11	WORD 12	WORD 13	WORD 14	WORD 15	WORD 16	WORD 17	WORD 18
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111
22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222
33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333
44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444
55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555
66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666
77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777
88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888
99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999

These are the 20 new lambdas. The two ends (2034.69 and 30000.0) were lambdas used in the computation of the previous case. The object here is to fill in a number of lambdas over the same region as before to improve the accuracy. The other lambdas are simply spaced between the ends.

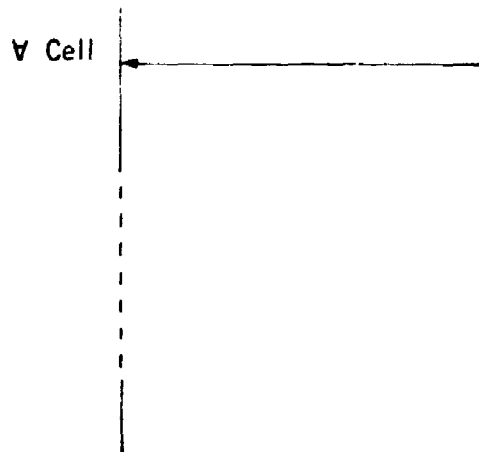
And to complete the sequence:

STOP		209473662.0															
WORD 1	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6	WORD 7	WORD 8	WORD 9	WORD 10	WORD 11	WORD 12	WORD 13	WORD 14	WORD 15	WORD 16	WORD 17	WORD 18
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111	11111111
22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222	22222222
33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333	33333333
44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444	44444444
55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555	55555555
66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666	66666666
77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777	77777777
88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888	88888888
99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999	99999999

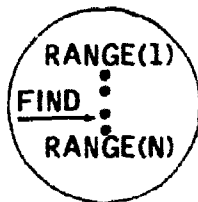
IV. SUBROUTINE DESCRIPTIONS

The program is divided into a monitor program (contained in program DRIVE and the overlay programs) plus subroutines. The discussion of these routines is contained in this section.

One brief word about the flow charts will be included here. Iterative loops will be indicated as follows:



The symbol \forall means "for every," and the example means that the indicated loop is to be performed until all cells have been exhausted. The following indicates a table lookup process of locating the variable FIND in the table RANGE:



Program DRIVE

DRIVE reads the basic data deck (in doing so it calls RDATT, DEFINP, and RDCELDT), and then reads option cards and turns control over to the proper overlay. In this capacity it acts as part of the option selector; the same function is performed in the first overlay.

Subroutine ANSPRIN

This is the basic answer printing routine; its output is discussed under Run Type SIMPLE.

Subroutine BALDEF2

This is the driver for balanced defense computation. It is similar to DEFOPTG, in that its task is to close to the proper cost for the defense, and it indeed uses similar code. The individual cell computation is done in BDCCELL. Figure 15 indicates the flow chart for BALDEF2.

This program computes the balanced defense for a number of costs, and evaluates each defense.

The defense computation is done for a range of lambdas, for here the attacker's lambda is the driving parameter for cost closing (see Appendix A).

Subroutine BDCCELL

BDCCELL is used in the balanced defense procedure and finds the hardness and cost for a given cell according to the balanced defense formulation*:

$$\text{AREAETH} = \frac{\text{LAMBDA} * \text{AREA}}{\text{BASEPOP}}$$

which assures constant return for the attacker over a number of cells. The hardness is

* See Appendix C for the formulation.

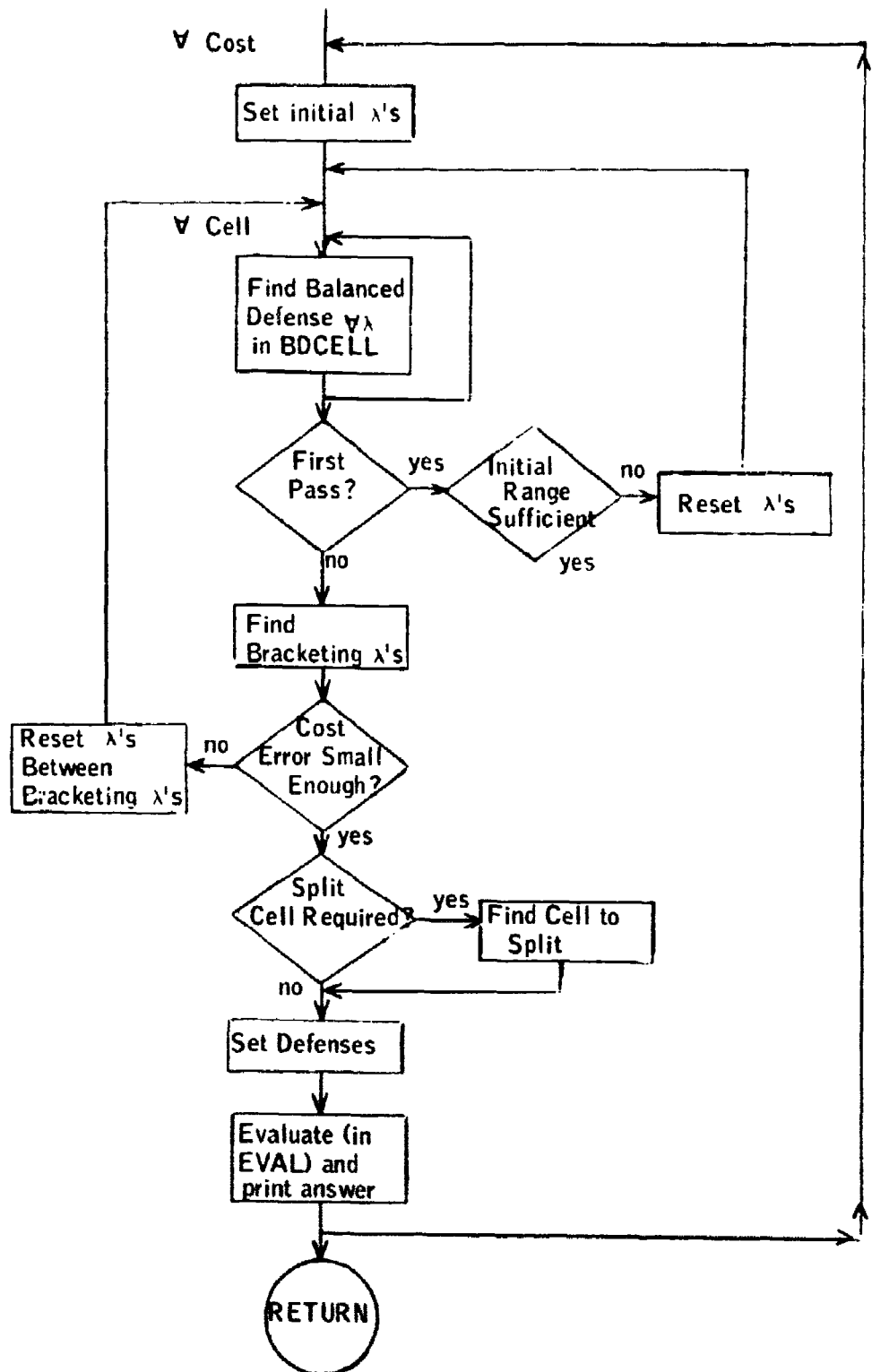


Figure 15. Flow Chart of BALDEF2

BDCELL (continued)

computed from*

$$\text{BDPSI} = \frac{8.1416}{\left[0.92491 * (\text{AREALETH})^{0.15} - 0.5\right]^3}$$

and the cost from †

$$\text{BALCOST} = 112.0 + 13.5 * \sqrt{\text{BDPSI}}$$

Saturation is checked at high psi (against WMIN) and low psi (against PSIMIN) and the highest psi is assigned in the one case and the unsheltered posture in the other (the unsheltered posture is signaled with $\text{BDPSI} = -10$), and costs adjusted accordingly.

Subroutine BNDSET

This is the portion of the bounding procedure which applies the upper and lower bound theorems. After the program has been through DEFOPTG to find the kill, weapons and cost for all the lambdas and mus, the following are performed in BNDSET:

1. For every lambda the solution with cost closest to TCOST is selected. The intercepts are then computed, and a printout is made of this information.
2. To find the upper bound the bracketing mus are used to interpolate an upper bound at cost TCOST.
3. Between each adjacent pair of lambda solutions, it is decided which lower bound theorem gives the best bound -- Pugh's primary bound, his secondary bound, or the lower bound found in this report. This problem is merely a geometric one and has been reduced to tests on the intercepts. The lower bound line is then stored.
4. From all bounding lines the current lower envelope is computed.

Subroutine COLLAPSE

This subroutine collapses the hardnesses available into a new lethal area table which has no unused hardnesses.

* From H. O. Horn, Nuclear Weapons Effects: Approximations of the Phenomena.

† An approximate analytical fit to the FINAL shelters data.

Subroutine COMFKIL

COMFKIL computes

$$FKIL(I,K,J) = 1.0 - \exp \{ -W(J,I) * D(K) \}$$

for every hardness I, attack density K, and height of burst J.

Subroutine CORELATE

This subroutine prepares and prints the following tables by sorting through all cells:

1. Distribution of shelter mixes over the population density brackets.
2. For each attack level, the distribution of attack density over the population density brackets. This is done using the lambdas remaining from an evaluation.
3. For each attack level, the distribution of attack density over the various shelter mixes, again using the lambdas.

For samples of these outputs, see the discussion of Run Type OUTPUT.

Subroutine CROSSCOR

See Run Type CROSSCOR.

Subroutine DCONSTR

DCONSTR checks to see if the current defense for the current cell is excluded and sets DEFEXCL accordingly. Normally it returns DEFEXCL = 0 (not excluded). During time-phased runs it sets

$$DEFEXCL = EXCLUDEF(IDEF, KDEF)$$

where IDEF and KDEF are the current defense and defense used for time-phasing; EXCLUDEF is computed in TIMEFAZE. If KDEF signals a split cell, DCONSTR computes the number of spaces of each shelter type and checks defense option IDEF against it; again returning DEFEXCL = 1 (excluded) or 0 (not excluded).

DCONSTR also has entry ABORT for all improper terminations of the program.

Subroutine DEFCOST

DEFCOST computes the defense cost:

$$DCOST = \sum_{I=1}^{NSTYP} CSTSHEL(I) * SPACES(I).$$

Subroutine DEFINP

DEFINP is an input routine which reads shelter parameters, defense options, and industry hardinesses. The routine directly reads FDEF, SPREAD, SPRIND, and CSTSHEL, and computes COSTPER for each defense option. Since the input is in a difficult form, this routine is used only for the basic data decks and for reading shelter data on the punch decks.

Subroutine DEFOPT

Subroutine DEFOPT optimizes the defense for a single cell for a range of current mu's (FMUX's) and fills CDLAG, the cell defense Lagrangian, IDFOPT, the optimum defense option, and CDEFCL, the cost of the current optimum defense option for each FMUX. It may operate in one of three modes - mixed lambda, mini-max lambda, and verification.

In the mixed lambda mode, the task of DEFOPT is to choose one of the 15 defense options for every FMUX so as to minimize the defense Lagrangian:

$$DLAG = \sum_{IAOB=1}^{NAOB} WDAOB(IAOB) * \sum_{I=1}^{NLAMX} WDLAM(I) * \left[PKILL(I) - \right. \\ \left. ALPHA * OFFCOST \right] + FMUX(J) * DCOST$$

where WDAOB and WDLAM are defense weights on attack objective and lambda, PKILL and OFFCOST are the population kill and offense cost returned by OFFOPT

DEFOPT (continued)

from the optimum attack for attack objective IAQB and lambda I, and DCOST is the cost of the defense option under consideration. OFFCOST is given by

$$\text{OFFCOST} = \text{FLAMX}(I) * \text{D}(K\text{DENS}) * \text{AREA}$$

where FLAMX is the lambda, D(KDENS) is the optimum attack density determined by OFFOPT, and AREA is the area of the cell. ALPHA is always 1.0 for the double Lagrangian procedure.

For each defense the population kill and offense cost is computed for an optimum attack. This is done by first calling GENSPACE, to compute the number of shelter spaces defined by the defense option; next DEFCOST is called, which calculates the cost of the defense option for that cell. The shelters are filled with the population of the cell according to the desired filling mode in FILSPACE, and PAYCOEF then puts the payoff (population and industry) into a form easily usable by OFFOPT. The optimum defense is generated by OFFOPT for a number of lambdas, and this repeated for all attack objectives so that the defense Lagrangian in the above form may be computed.

For each mu DEFOPT records the defense Lagrangian CDLAG and the defense cost CDEFCL of the cell for the optimum defense chosen for that mu. Figure 16 shows the flow chart for DEFOPT in the mixed lambda mode.

In the minimax mode DEFOPT operates as shown in Figure 17. In this mode the object is to consider several lambdas for the attacker, and to choose a defense which minimizes the worst excess of what is achieved for a lambda compared to the best that can be achieved. DEFOPT first finds the defense with the lowest Lagrangian for every lambda -- it then computes the delta or difference between the Lagrangian which each

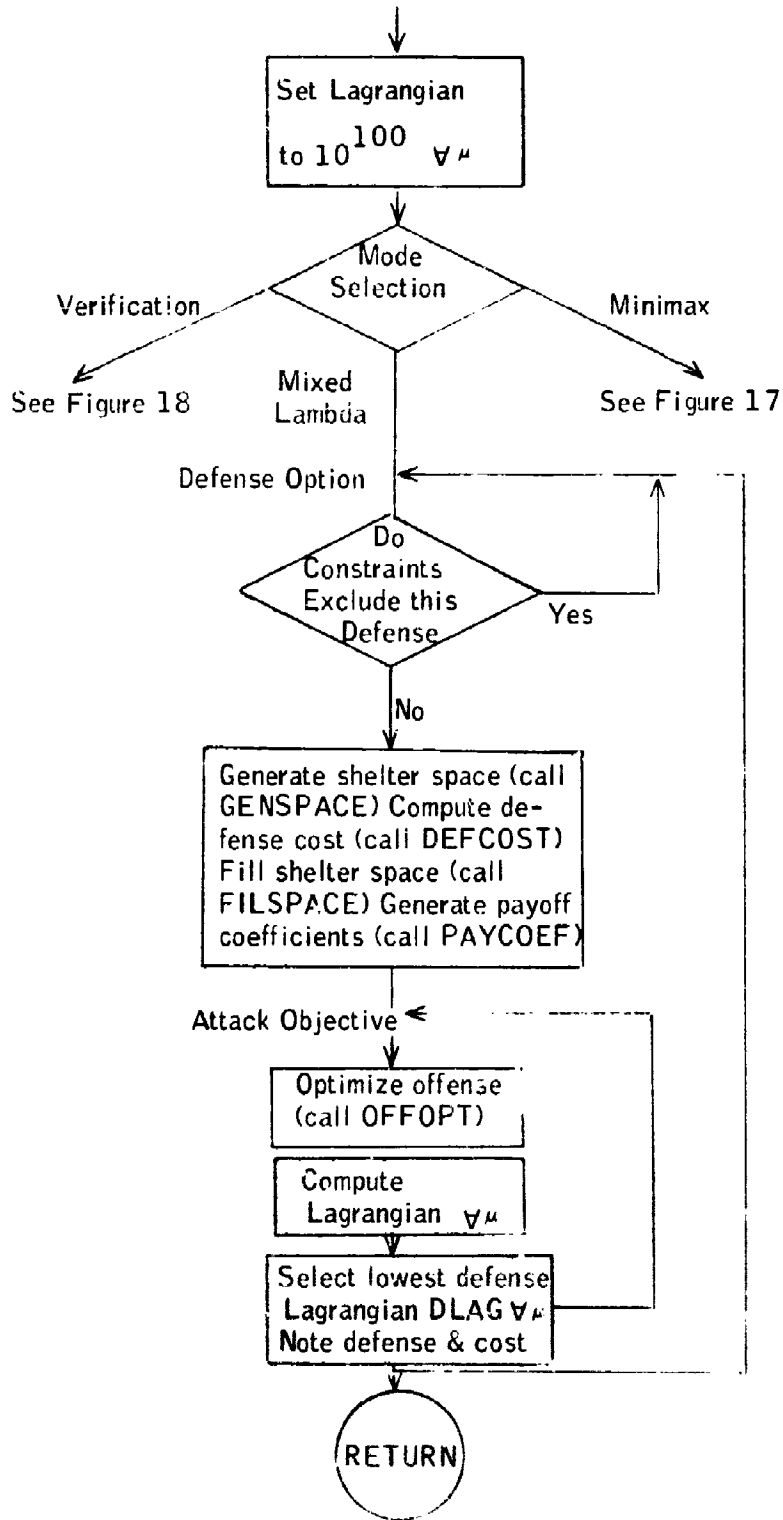


Figure 16. Flow Chart of DEFOPT, Mixed Lambda Mode

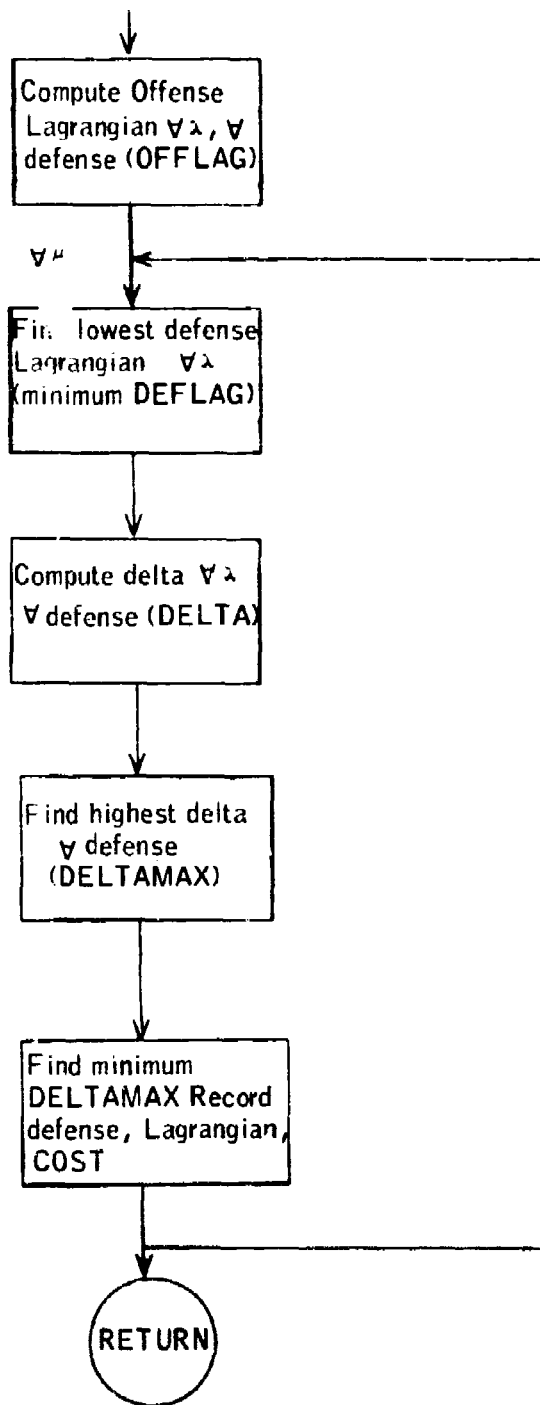


Figure 17. Flow Chart of DEFLOPT, Minimax Mode

DEFOPT (continued)

defense achieves for every lambda and the lowest Lagrangian for that lambda -- it then finds the highest delta for every defense -- and picks the defense which minimizes the highest delta. The only other computation is to precompute the offense Lagrangian for every lambda and for every defense.

DEFOPT, in this mode, picks a defense to minimize

$$\text{DELTAMAX (IDEF)} = \text{Maximum}_{\text{ILAMX}} \{ \text{DELTA (ILAMX, IDEF)} \}$$

where

$$\text{DELTA (ILAMX, IDEF)} = \text{DEFLAG (IDEF)} - \text{Minimum}_{\text{IDEF}} \{ \text{DEFLAG (IDEF)} \}$$

and

$$\text{DEFLAG (IDEF)} = \text{OFFLAG (ILAMX, IDEF)} + \text{FMUX (J)} * \text{DDCOST (IDEF)}$$

$$\text{OFFLAG (ILAMX, IDEF)} = \sum_{\text{IAOB}=1}^{\text{NAOB}} \text{PKILL (ILAMX)} - \text{ALPHA} * \text{OFFCOST}$$

This procedure is repeated for every mu until optimal defenses are found for the cell for every mu.

The verification mode of DEFOPT is shown in Figure 18. This differs from the mixed lambda case in that a number of single lambda Lagrangians are computed, and the lowest one selected. The best defenses are not saved, but only the Lagrangian, cost, and weapon expenditure. The Lagrangian which is minimized for every lambda and mu is

$$\text{CURLAG (I, J)} = \text{Minimum}_I \{ \text{HLAG (I)} + \text{FMUX (J)} * \text{DCOST} \}$$

where HLAG is computed by OFFOPT and is the maximum offense Lagrangian.

There are no abnormal returns from DEFOPT.

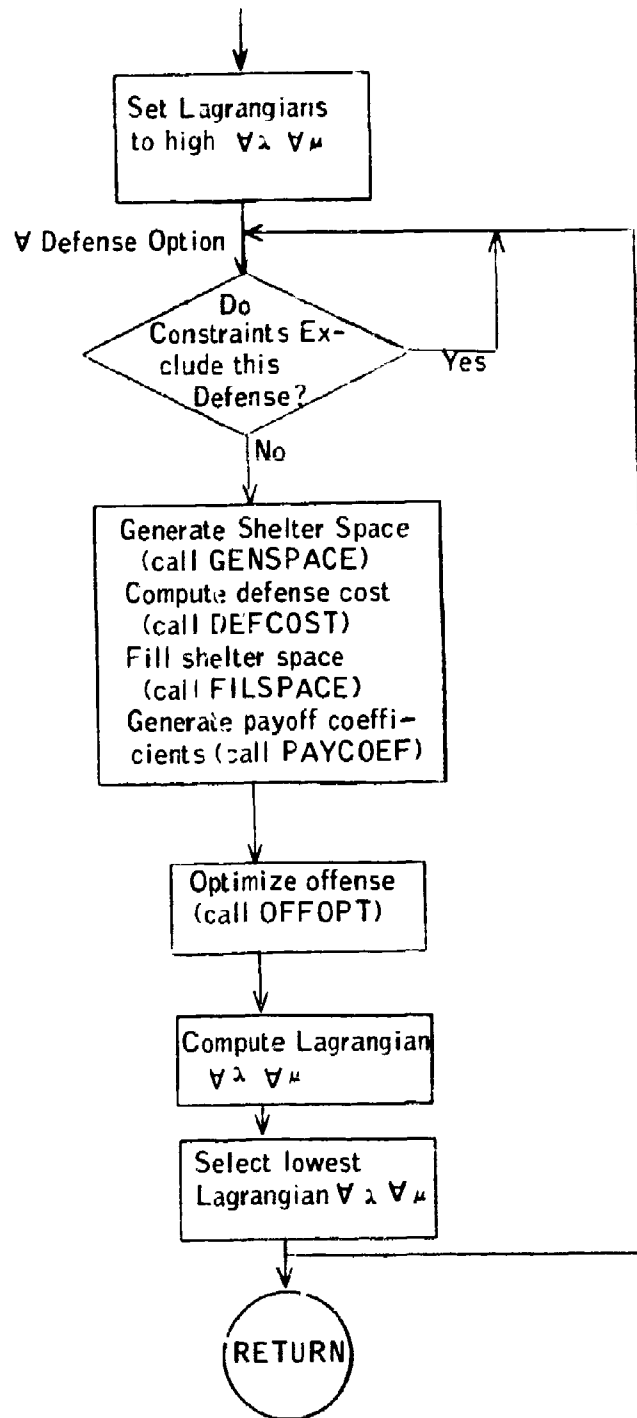


Figure 18. Flow Chart of DEFOPT, Verification Mode

Subroutine DEFOPTG

DEFOPTG has as its purpose to optimize the defense for a group of cells, indices ICBEG and ICEND, to a total defense expenditure TCOST. If it is unable to spend this amount (saturation), it spends most possible and exits with ISAT=2 (which is otherwise 1) and ACOST equal to the actual expenditure.

Actually, this is a much too lofty statement of the function performed by DEFOPTG. In order to optimize the defense for a group of cells, DEFOPTG requires the services of an extremely able assistant, subroutine DEFOPT. Recall that on a single cell DEFOPT finds the optimum defense for a number of mu's (or externally set values for the marginal number of people saved per dollar). It is the task of DEFOPTG to drive DEFOPT with a fixed set of mu's once for each cell in the group of cells, to accumulate the total cost of defenses optimized for each mu, and to pick a new set of mu's to bring the achieved defense cost closer to the target cost TCOST. DEFOPTG supplies the mu's to DEFOPT, which then returns the cost of the optimal defense found for each mu (other information is available, but DEFOPTG uses only the cost), and DEFOPTG uses this information to pick a new set of mu's after each pass of optimizing all cells. Thus we see that DEFOPTG contains only the cost closing mechanisms for adjusting the defense for the group of cells to a precise cost. Since individual cell optimization is done in DEFOPT, the various options and strategies such as expected attack levels, attacker objectives, and defense constraints are reflected in that subroutine. Actually DEFOPTG is a general routine, suitable for any separable resource allocation problem, and in no way considers the defense aspects of the problem.

DEFOPTG (Continued)

The procedure for cost closing, shown in the flow chart of DEFOPTG in Figure 19, is to find the optimal defense for each cell for a number of mu values. The initial mu's are pre-chosen program constants or input quantities; a first pass is performed, optimizing each cell for this initial set of mu's. If a cell has the same optimum defense for the entire mu range (this is unlikely on the first pass, but this is an extremely important mechanism on subsequent passes), its defense is frozen, and the cost of its defense is added to the frozen cost. After the first pass the two mu values whose costs bracket TCOST are selected and a new range of mu's is set between those two values.

All subsequent passes follow the same procedure as the first pass: each unfrozen cell is optimized for the range of mu's and the costs are noted for each mu. If the optimized defense for a cell is the same for the entire mu range, the defense for that cell is frozen and the cost of the defense is added to the frozen cost. Then after the pass is completed the bracketing mu's are found and the mu's reset between them. On the second and subsequent passes, an error quantity is checked to see if the procedure can be ended.

The desired outcome of the procedure is to have an optimal defense which is within some small amount of costing precisely, TCOST. However, it may not be possible to get sufficiently close in by this procedure. It may happen that for very small change in mu a group of cells or one sufficiently large cell may change in cost enough to make exact cost closing possible. Thus it is reasonable that the error criterion is

$$ERRZ = \frac{DLTMU}{FMULOW} * \frac{DELTC}{TCOST} \leq 0.0001$$

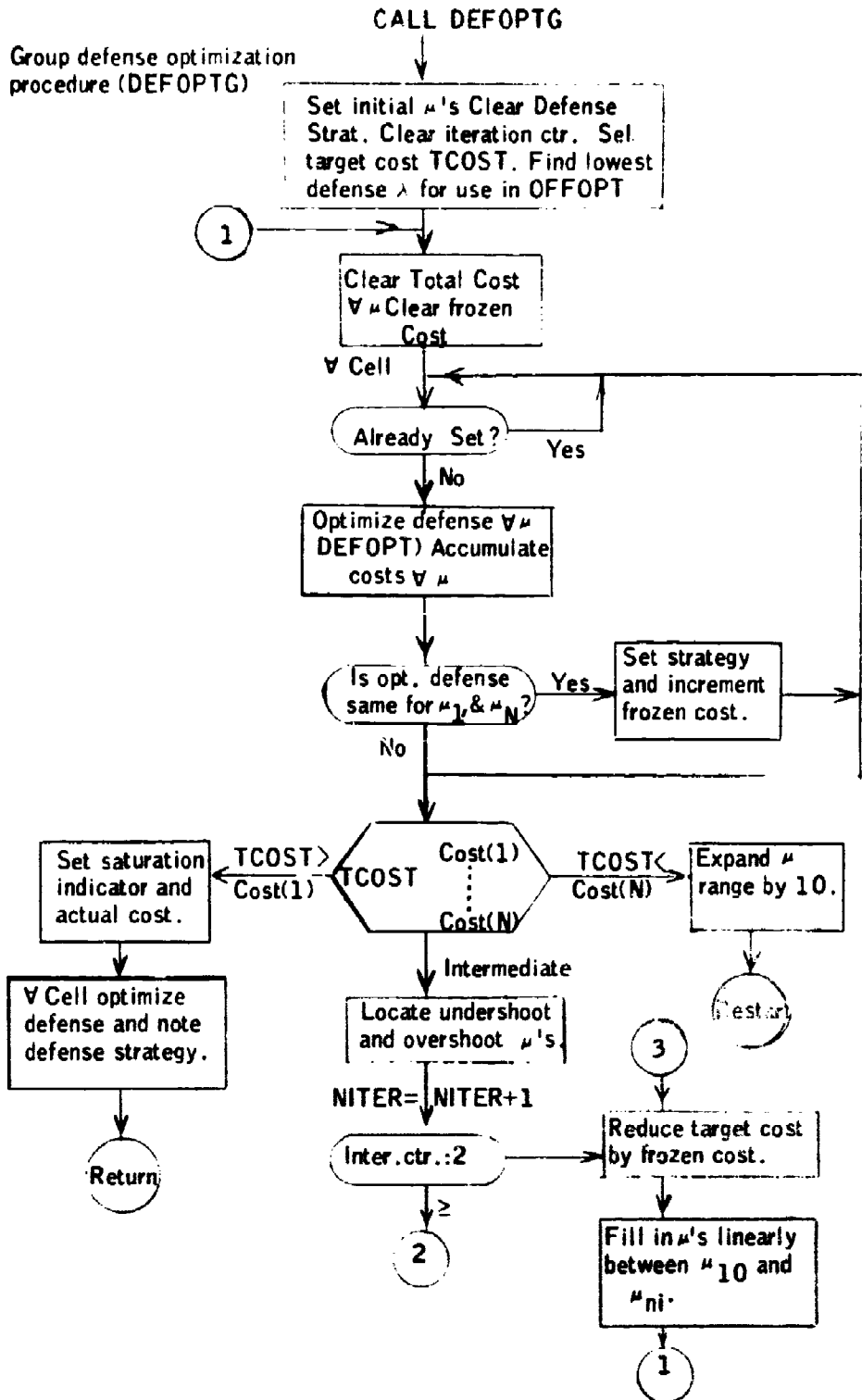


Figure 19. Flow Chart of DEFOPTG

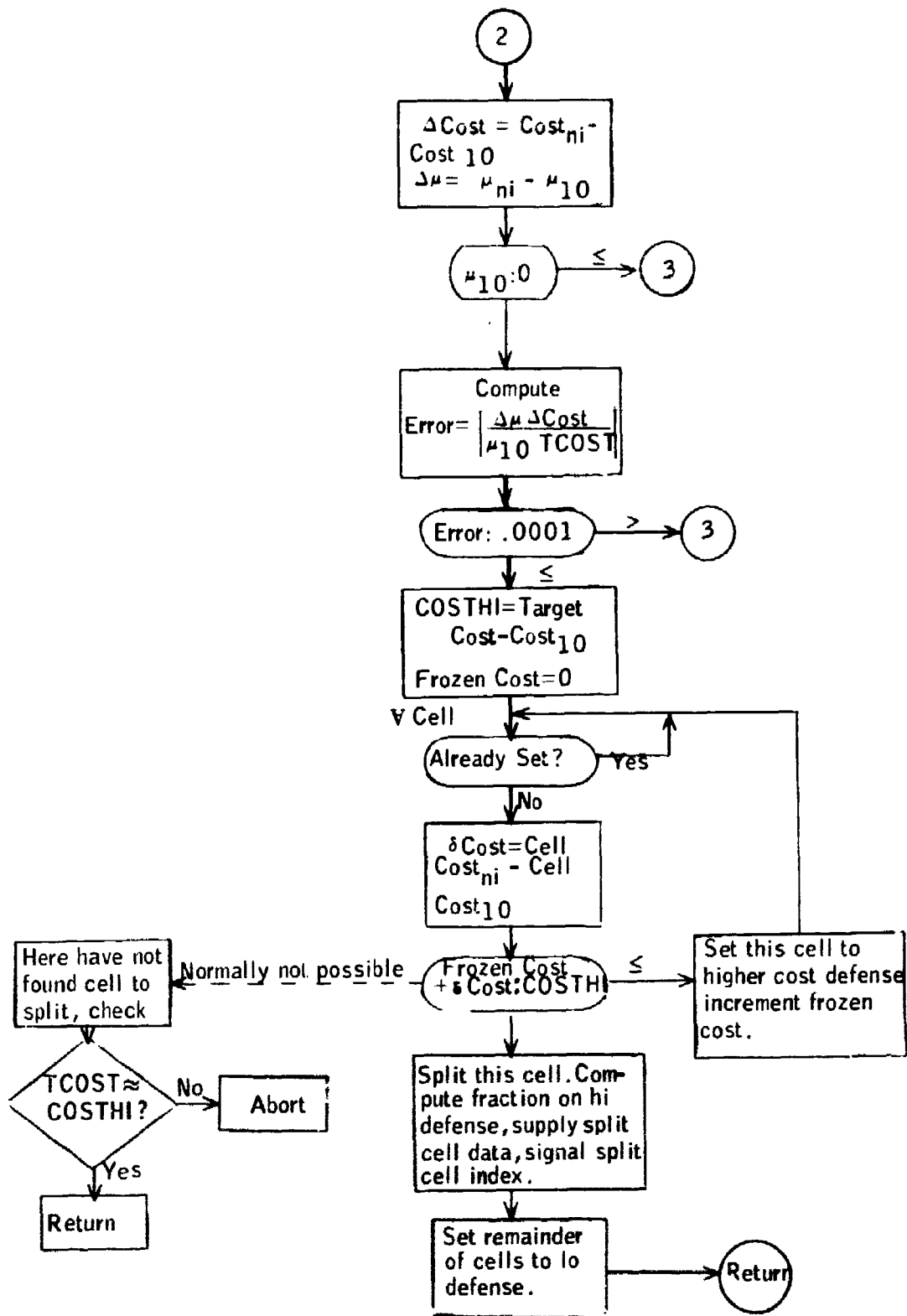


Figure 19 (continued). Flow Chart of DEFOPTG

DEFOPTG (Continued)

where $DLTMU$ is the change in μ between upper and lower values, and $FMULOW$ is the lower μ value, $DELTC$ is the fractional change in μ and $DELTC/TCOST$ is the fractional change in cost for the two values. Thus if either fractional quantity is small, i.e., if either μ closing or cost closing is close, then the iterating will cease.*

When the error criterion has been satisfied (but the cost is still not closed exactly), two μ 's have been found, one with too high a cost and one with too low a cost. Most of the cells will have the same defense for both μ 's, and will have been frozen. Since the cells which have not been frozen yet are flipped, one by one, from the lower cost defense to the higher cost defense, and the additional cost for that cell is added to the amount spent in the lower cost defense. This process of flipping cells to the higher defense is continued until the cell is reached where expended cost exceeds $TCOST$. Then this cell is split, putting part at the higher defense and part at the lower defense. Finally, all remaining cells are set to the lower cost defense (again, between choices generated by the two bracketing μ 's), and the defense is set for the entire group of cells.

If after the first pass, the costs computed for the initial μ range do not encompass $TCOST$, one of two things happen--if the computed costs are too low, the saturation indicator (ISAT) is set and all cells are set to the defense with the highest computed cost. If the computed costs are too high, the μ range is expanded

* $ERRZ$ has a more theoretical justification: $DLTMU * DELTC$ is an upper bound on possible difference in fatalities between the two points generated by the two μ 's in question (from the definition of μ), and $FMULOW * TCOST$ is a lower bound on the number of people saved by a shelter program of cost $TCOST$. Hence $ERRZ$ is an upper bound on the fractional error in fatalities from using either one of the μ 's.

DEFOPTG (Continued)

by 10 and the first pass is repeated.

For the operation of DEFOPTG in the verification procedure, see Figure 20 . In this procedure the idea is to find optimum defenses for a range of 20 lambda's and 20 mu's, and to save the total kill, weapon expenditure, and cost expenditure for each of the 400 lambda-mu combinations. For verification, exact cost closing is not required; hence this mode of operation is not iterative in DEFOPTG; the subroutine makes one pass through all the cells, accumulating totals for each lambda-mu combination, and that is all that is required for verification.

There are only two abnormal returns from DEFOPTG. One is in the verification procedure--if more than 20 lambdas or mus are used, the program aborts. The other is shown at the very end of Figure 20 : if, because of rounding perhaps, all the cells are flipped and set to the upper defense without finding the one to split, then if the upper defense cost is close to TCOST then the routine returns normally (with all cells flipped to the higher defense). If the upper defense cost is not close, the routine aborts.

Subroutine DEFPRNT

This is a print routine responsible for printing the defense when it is input; for a sample print see the discussion of Run Type INPUT.

Subroutine DEFPUNCH

This subroutine punches a defense onto cards for saving. First, a header is punched, and then the defense options, 40 to a card. Next, split cell information, and all parameters and shelter data are punched. For a list of the parameters punched see DEFSPEC in the data sequence definition section.

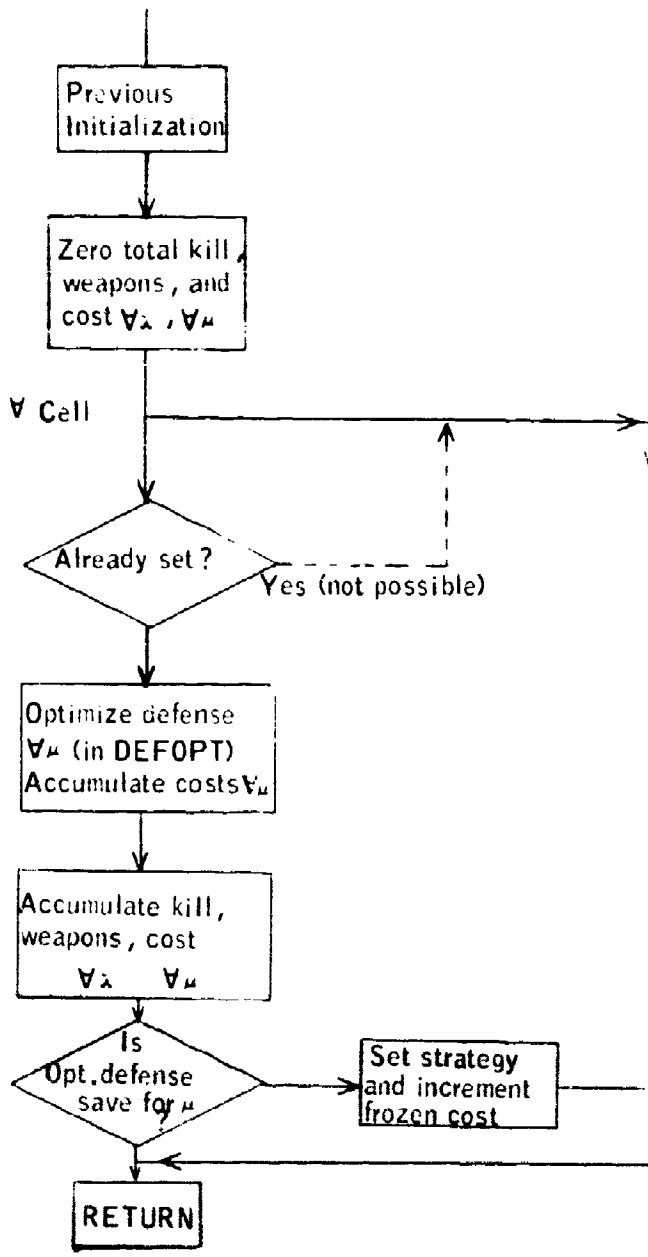


Figure 20. DEFOPTG in Verification Mode

Subroutine DEFSPEC

See Run Type DEFSPEC for functions of this subroutine.

Subroutine EVAL

The task of EVAL is to find the optimum attack against all cells for a range of attack levels and compute the attack payoff, population kill, and industry kill for each attack level. Although EVAL is used to evaluate a defense for different times of attack (IAPOP), attack objectives (IAOB), and filling modes (IFILL), these parameters are not reflected in EVAL but rather in FILSPACE and PAYCOEF, which are called from OFFOPTZ. The flow chart of EVAL is shown in Figure 21.

The portion of the evaluation task performed in EVAL is to select lambdas to drive OFFOPTZ, finally selecting lambdas which give weapon expenditures close enough to the desired attack levels. For the initial preset set of lambdas OFFOPTZ finds the optimum offense, returning weapon expenditures for each lambda. If the range of expenditures does not encompass the desired attack level range, the lambda range is increased and the initial pass is redone. When the weapon expenditure range encompasses the attack level range, the procedure is simply to find the lambdas with expenditures bracketing each desired attack level, setting lambdas linearly between the bracketing lambdas, and finding a new set of optimum attacks in OFFOPTZ. After each pass the kill at each precise attack level is estimated from the bracketing values for each, and an error bound is determined. If the error is small enough, the procedure terminates; otherwise, new lambdas are set linearly between the bracketing lambdas and a new set of optimum attacks is found in OFFOPTZ, until the error is small enough.

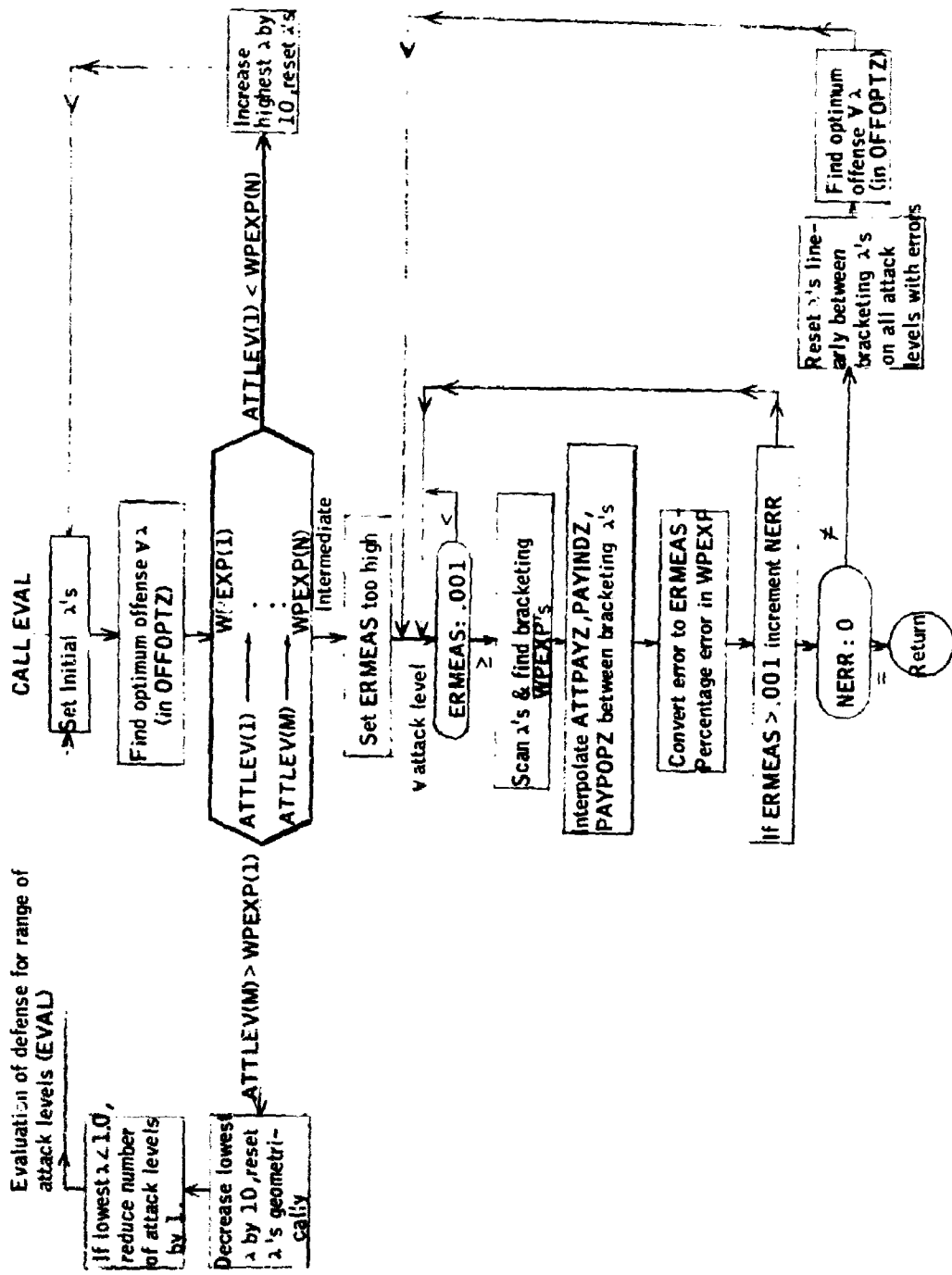


Figure 21. Flow Chart of EVAL

EVAL (continued)

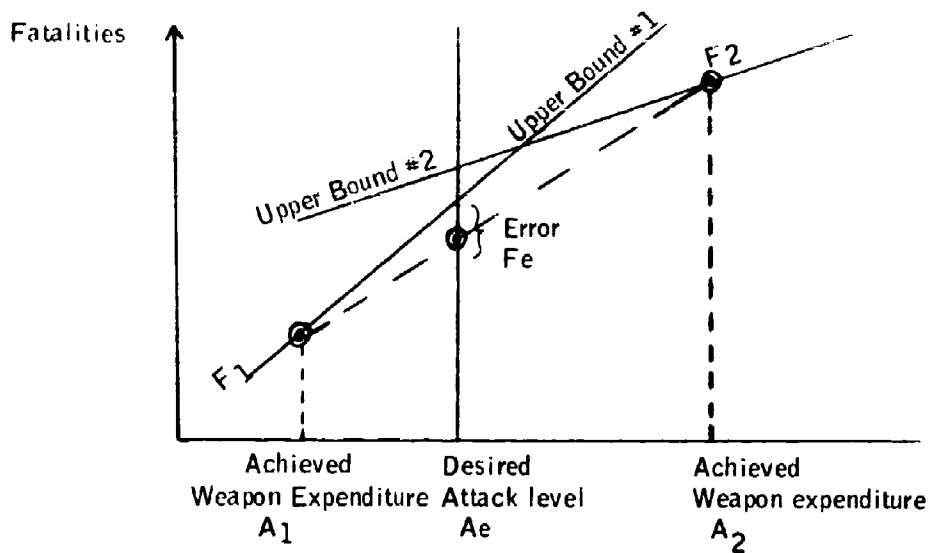


Figure 22. Estimation and Error for Kill at Desired Attack Level

Figure 22 shows the estimation procedure and error criterion for EVAL.

$F_1(A_1)$ and $F_2(A_2)$ are two fatality levels corresponding to lambdas which bracketed the desired attack level A_e . The estimate used for fatalities at A_e is the linear interpolation between F_1 and F_2 . The point F_e is a lower bound on fatalities at A_e since the payoff curve is convex*. The lines upper bound #1 and upper bound #2 have slopes of the bracketing lambdas (see the section on double Lagrange theory for why they are upper bounds), and these lines provide an upper bound for the payoff at any attack level. To provide an upper bound for the payoff at A_e , we simply take the lower of the two upper bounds. The possible error is thus as indicated, and this error is converted into an error in attack level using the lambda value of the upper bound.

* Convex because of decreasing returns and because the attacker could split attack among cells to achieve any intermediate point.

EVAL (Continued)

The decision to terminate the iteration is made on that quantity. For information purposes, the lambda value at A_e is estimated as $(F_2 - F_1)/(A_2 - A_1)$.

There is only one abnormal return from EVAL and it can occur, as it turns out, if on the first pass the lowest attack level is precisely equal (to the last bit) to the lowest weapon expenditure.

Subroutine FILSPACE

FILSPACE first determines the number of people in the cell at time of attack (from IAPOP) and then fills the shelter spaces in one of three modes -- IFILL = 1, 2, or 4 -- optimal, fractional alert, and balanced defense. The routine begins with the SPACES array and fills the population into the POPSH array.

Optimal filling mode fills the population into the hardest shelters first, working down until either population or space is exhausted. Any remaining population is placed in the unsheltered category.

The fractional alert mode assumes that some fraction of the population has received the alarm. If the alerted population is more than available space, all spaces are filled. If the alerted population is less than available space, each shelter is filled the same fraction of capacity. All unalerted population, of course, is unsheltered.

The balanced defense mode is not really a filling mode, but rather a way of implementing the variable hardness requirement for balanced defense in the evaluation process. If the cell is unsheltered, operation of FILSPACE is normal. If not, subroutine BDCELL is called which computes the proper hardness and its lethal area. FILSPACE then sets the proper kill fractions by calling COMFKIL, and operation is then normal.

Subroutine GENSPACE

GENSPACE computes the number of shelter spaces in a cell indicated by each defense option. It first finds the base population BASEPOP using IDPOP and then computes each shelter space from

$$\text{SPACES}(I) = \text{FDEF}(I, \text{IDEF}) * \text{BASEPOP}.$$

Subroutine GETCELL

This subroutine reads cell data in the form shown in option LIVEDATA.

Subroutine LAMSET

LAMSET uses FLAM(I), and FLAM(NLAM) and computes intermediate FLAM's in an ascending geometric series. LAMSET also contains a diagnostic print routine as entry LOBNDZ.

Subroutine MIXLAM

See Run Type MIXLAM for this subroutine.

Subroutine OFFOPT

OFFOPT generates the optimum attack density, height of burst, population kill, and industry kill on a single cell for a set of NLAMX attack Lagrange multipliers (FLAMX's). The flow chart is shown in Figure 23.

The basic task of OFFOPT is to find the attack density index and the height of burst index which maximizes the offense Lagrangian

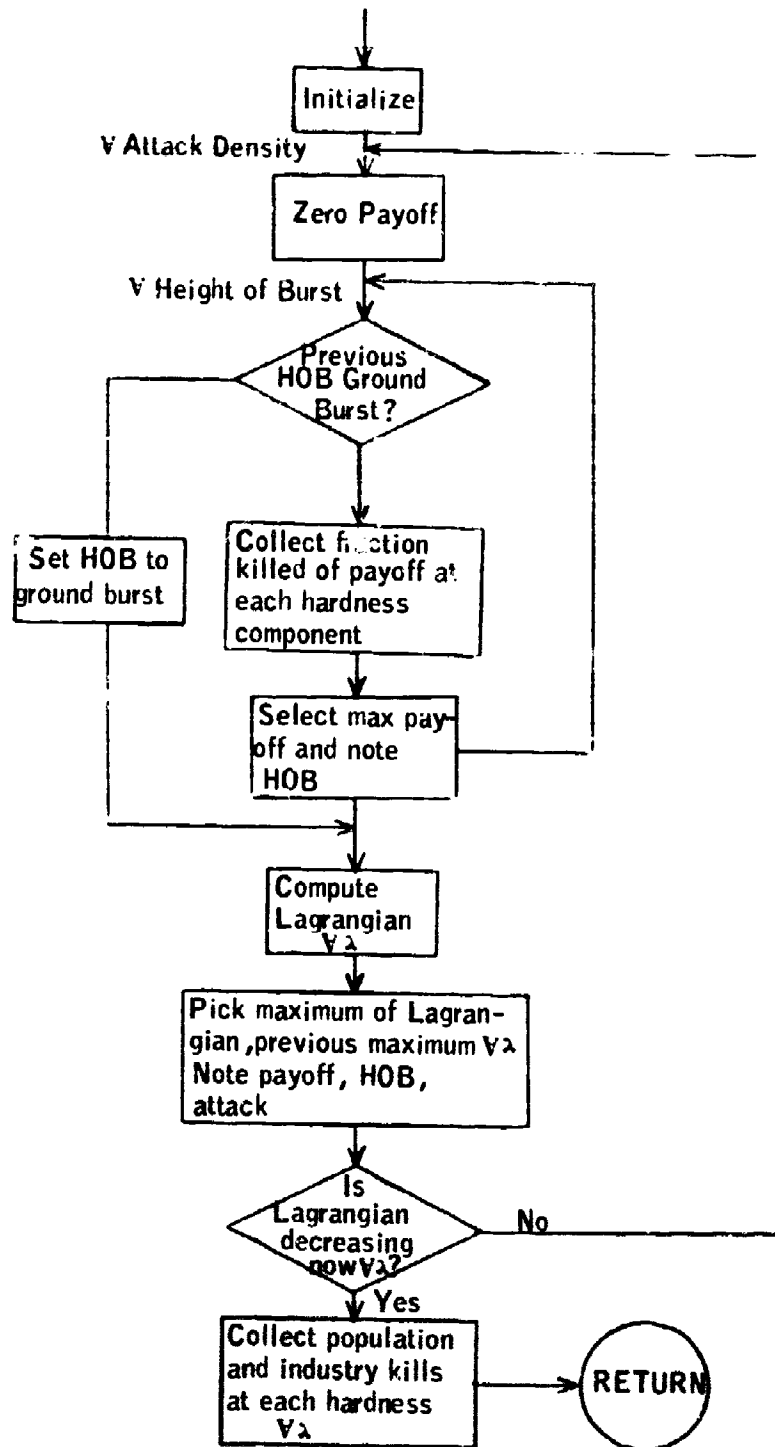


Figure 23. Flow Chart of OFFOPT

OFFOPT (Continued)

$$\text{Maximum JHOB} \left\{ \sum_{L=1}^{\text{NCOMP}} A(L) * FKIL [ICOMP(L), KDENS, JHOB] \right\} \\ - FLAMX(ILAM) * AREA * D(KDENS)$$

for every Lagrange multiplier FLAMX (ILAM). See the glossary for definition of terms. For every optimum attack so found, OFFOPT saves the optimal attack density IDOPT, height of burst IHOBPT, payoff HOPT, offense Lagrangian HLAG, population kill PKILL, industry kill VKILL, and weapon expenditure WEPEQ.

Three time saving short cuts were added to this procedure: 1) FKIL is not used as a three-dimensional array but rather as 15 two-dimensional arrays; 2) in the attack density scan, if the optimum height of burst is ground burst, all subsequent higher densities are ground burst also; 3) in the attack density scan, if for the lowest (and hence all) lambda the Lagrangian reverses and starts decreasing, all subsequent higher densities will have lower Lagrangians, and the scan is terminated.

There are no abnormal returns from OFFOPT.

Subroutine OFFOPTZ

OFFOPTZ finds the optimum attack for all cells for a range of externally set lambdas (FLAMX). OFFOPTZ is a simple routine and for a normal (unsplit) cell, the procedure is to read the cell, and next

```
CALL      GENSPACE
CALL      FILSPACE
CALL      PAYCOEF
CALL      OFFOPT
```

OFFOPTZ (Continued)

and lastly to accumulate the total attacker's payoff, population kill, industry kill, and weapons expended for every lambda. All the payoffs and kills are returned from subroutine OFFOPT.

All cells are processed in the above manner except for the split cells, of which there is usually one in a defense. (The split cell is generated by DEFOPIG for final cost closing.) OFFOPTZ processes that cell twice, weighting the kills for the primary and secondary defense by the fraction of the cell at each defense. Figure 24 shows the flow chart for the entire procedure.

There are two abnormal returns from OFFOPTZ: if a cell has defense option zero, the program aborts; and if a cell has a negative defense option (indicating a split cell), but the cell is not contained in the split cell list, the program aborts.

Subroutine PAYCOEF

PAYCOEF first computes the industrial value and population in each hardness component (using POPSH computed in FILSPACE) and stores these in array VALIND and POPUL. It then computes the value at each hardness component for the current attack objective using POPMULT and VINDMULT, and stores the attacker's value at each hardness in a compacted array A (having no zero components), using an array ICOMP to store the hardness components represented in A.

Attack generator for range of lambdas (OFFOPTZ)

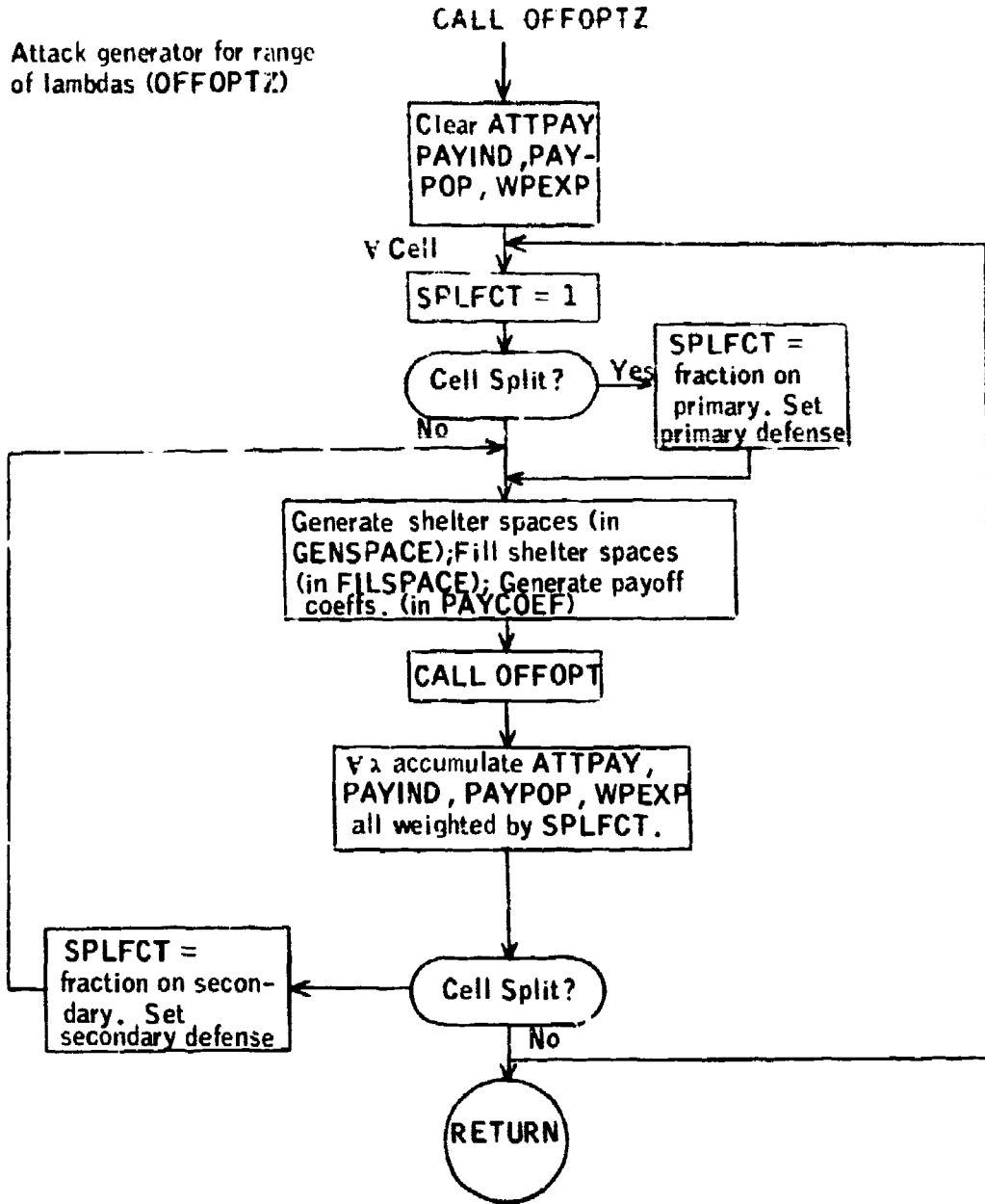


Figure 24. Flow Chart of OFFOPTZ

Subroutine PERTURB

The purpose of this subroutine is to adjust a lethal areas in response to a fractional change in hardness of DELTA. The equation used is

$$W(J,I) = W(J,I) - \text{DELTA} * \text{PSI}(I) * \frac{2 * W(J,I)}{\text{BETA} * \text{PSI}(I) - X}$$

$$\text{where } X = \frac{.00001875 * \text{HOB}(J) * \text{ALPHA}}{[0.34303 * \sqrt{W(J,I)}] \text{BETA} - 0.75}$$

where ALPHA is 8.7 or 10.0 and BETA is 2.28 or 2.9, depending on whether the lethal area is greater or less than 3.03. The above equation is found by taking partial derivatives from

$$\text{PSI} = \frac{\text{ALPHA}}{\text{DBETA}} [1.0 + \text{HOB}^2 + A(1 - .25 D^{0.75})]^*$$

where here A and D are height of burst and distance in kilofeet for a 1 kt weapon.

Subroutine PICKLAM

This subroutine is responsible for choosing the lambdas for the bounding procedure. The procedure used is exactly that described in the theoretical section on the bounding procedure. The lambda sweep begins in the middle and extends both directions; hence two sets of 10 lambdas are chosen sweeping in opposite directions from the previous ends of the sweep.

The controlling parameter for the separation between the lambdas is EPS; it enters strongly into the expressions for the new lambdas.

*From H. O. Horn, "Nuclear Weapons Effects: Approximations of the Phenomena".

Subroutine POPQUANT

This subroutine generates nine population density brackets and sorts the cells into the nine brackets, accumulating totals. It is a prelude to subroutine CORELATE; an example of the printout is the first block of print in Run Type OUTPUT.

Subroutine PREBND

This is the driver for the bounding procedure. From the flow chart in Figure 25, it can be seen that the basic cycle is

1. Select lambdas in PICKLAM
2. Find Lagrangian solutions in DEFOPTG
3. Compute bounds in BNDSET
4. Repeat

The variations on this theme are in the different modes; these modes are described in the section on run type BOUND.

Subroutine PRNTBLK

PRNTBLK evokes the entire range of diagnostic block print routines from ATT0BJZ to SPLITDAZ.

Subroutine PRTOCT

PRTOCT is an octal print routine with entries PRTINT for integer print and PRTOEC for decimal print.

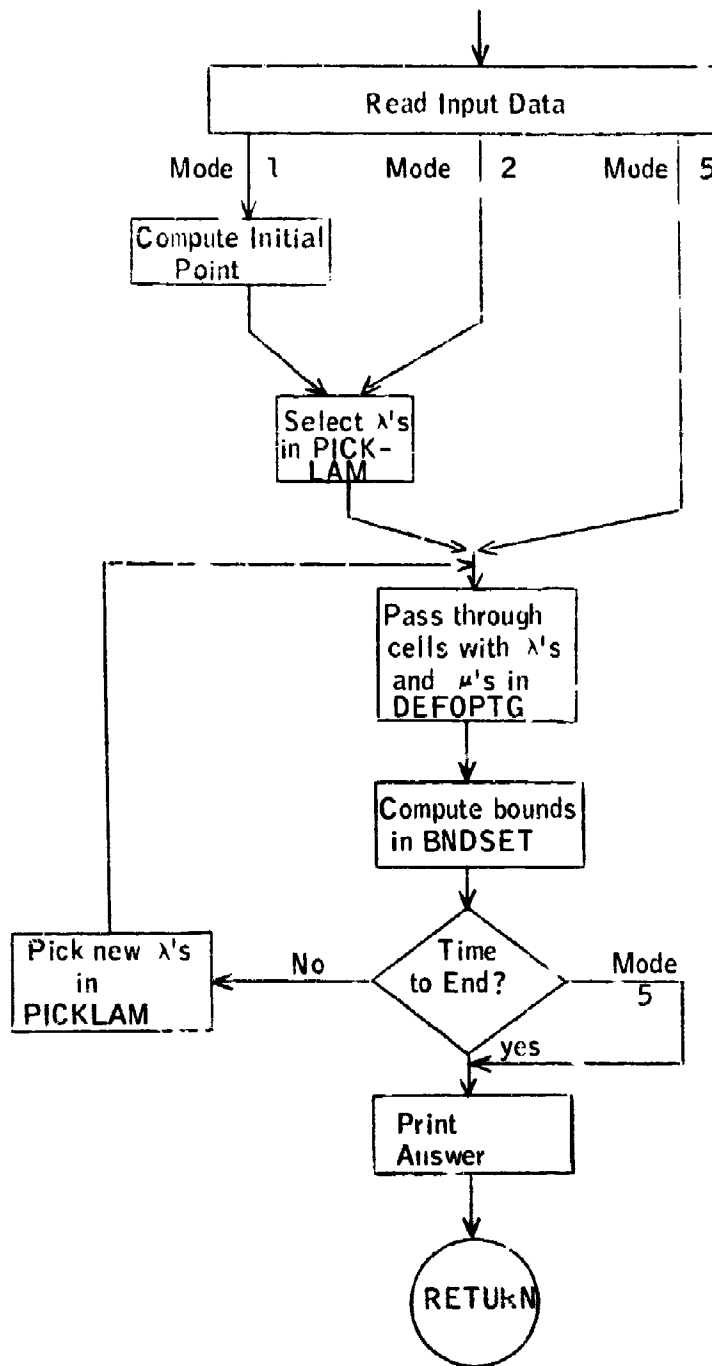


Figure 25. Flow Chart of PREBND

Function RANDNX)

RANDN is a pseudo-random number generator.

Subroutine RDATT

RDATT is the input routine which reads heights of burst, hardnesses and either lethal areas or lethal radii. It then reads the actual heights of burst to be scanned and prints both the entire lethal area table and a reduced table with heights of burst to be used. The input format is included in the INPUT data sequence definition.

Subroutine RDCELDT

RDCELDT is used in reading the basic data deck, and it reads variables FINDMAX, AREAMAX, POPMAX, and then reads the cell data for each cell.

Subroutine RDCELL

RDCELL unpacks the basic cell data for cell ICELL from storage in NCELA and NCELB and stores it in usable form in common block CURCELL.

Subroutine RDCTDAT

This subroutine reads track data in the form shown in option LVCTDAT.

Subroutine SHELINP

This subroutine reads in shelter data in the form specified in the data sequence for Run Type INPUT.

Subroutine STATCOST

See Run Type STATCOST for a discussion.

Subroutine SPLITDAZ

SPLITDAZ contains 20 of the diagnostic block print routines, each under a separate entry point.

Subroutine STCELL

STCELL packs the basic cell data for cell ICELL into NCELA and NCELB from CURCELL in a form usable by RDCELL.

Subroutine STCELLDT

This subroutine tests the population and industrial density of cells read in GETCELL and either stores the cell or accumulates it into tail cells.

Subroutine TABLDENS

The function of TABLDENS is described in that Run Type discussion. The ρ/λ breakpoint between two attack densities is

$$\rho/\lambda = \frac{D(1) - D(2)}{ATTPAY1 - ATTPAY2}$$

Where the D's are the attack densities and the ATTPAY's are the fractional kills.

Subroutine TABLE

This is a table lookup subroutine.

Subroutine TIMEFAZE

Discussed under Run Type TIMEFAZE.

Subroutine WORKOUT

For discussion see Run Type EXERCISE.

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D		
<i>(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)</i>		
1. ORIGINATING ACTIVITY (Corporate author) LAMBDA Corporation		2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED
		2b. GROUP
3. REPORT TITLE AN OPTIMIZATION STUDY OF BLAST SHELTER DEPLOYMENT, VOLUME III: APPENDIX H = "BLAST" - THE COMPUTER PROGRAM		
4. DESCRIPTIVE NOTES (Type of report and inclusive dates)		
5. AUTHOR(S) (First name, middle initial, last name) David L. Mitchell		
6. REPORT DATE September 1, 1966	7a. TOTAL NO OF PAGES 138	7b. NO OF REFS 2
8a. CONTRACT OR GRANT NO OCD-PS-66-113	8b. ORIGINATOR'S REPORT NUMBER(S) Report 3	
b. PROJECT NO Work Unit 1632A		
c. Subcontract 138-5	8c. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
10. DISTRIBUTION STATEMENT This document has been approved for public release and sale; its distribution is unlimited.		
11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Office of Civil Defense Department of the Army	
13. ABSTRACT This study examines methods of determining blast shelter deployments and of assessing their performance for a variety of nuclear attacks. The goal is not to seek a single optimal deployment, which generally requires making arbitrary assumptions on the nature and size of the attack, thus overlooking the attacker's freedom of choice after a blast shelter program has been deployed. Rather, the study seeks "stabilized" deployments which protect population almost as well as an optimal deployment, even though it is not truly optimal for any specified attack. The study examines the attacker's freedom to vary force level, time of attack, attack objective, height of burst, and targeting. A quite general and flexible computer model BLAST, based on generalized Lagrange multipliers, generates shelter deployments for the U.S. and computes their effectiveness against attacks in which these factors are varied. In BLAST the nation is considered as a collection of cells two nautical miles square, providing a detailed analysis of the offense/defense interaction. Volume I summarizes the methodology, results, and conclusions. Volume II contains technical appendixes, including the data employed. Volume III describes BLAST in detail for the analyst and programmer.		

DD FORM 1473
1 NOV 65

UNCLASSIFIED

Security Classification

UNCLASSIFIED

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Blast Shelter Deployments Civil Defense Systems Maximization of Nuclear Attack Fatalities Stabilized Shelter Deployments Shelter Vulnerability Allocating Population to Blast Shelter HOUSTON and APRIL Shelter Data BLAST-Computer Program						

UNCLASSIFIED

Security Classification