OPTIMUM SELECTION OF SYSTEM IMPLEMENTATIONS WITH A WEIGHTED SUM OBJECTIVE FUNCTION

University of Texas
NGR-44-012-144

INTRODUCTION

Present day technology is characterized by increasingly complex systems consisting of a variety of different components operating as a unit to satisfy some processing requirement. Due to this very complexity, design of these systems is often an exceedingly difficult problem. Typically the system is broken down into simpler subtasks each of which can then be implemented more or less independently of the others. Simulation of both the subtask implementations and that of the overall system can also be a considerable aid in the design process. Quite often however, even with the aid of simulation, the breakdown of the system into smaller parts results in some desirable characteristics of the total system being ignored or difficult to obtain. Characteristics of this type that usually trade off among one another might be cost, weight, power consumption and, where meaningful, throughput time as in a primarily serial system based on or consisting of a digital computer for example. The techniques whose preliminary development are described in this report are intended to be applied to this type of problem. They are envisioned as applicable design aids in decisions concerning combination software and hardware, and pure hardware implementation of serial systems. These methods are by no means meant to be complete design methods in themselves but only to supplement existing ones.

Specifically, given a system to implement that has been broken down into subtasks and given various implementations with differing

cost/performance specifications of these subtasks, how can a designer choose among the implementations available to him in order to accomplish the overall system objective as well as to optimize a chosen characteristic? This characteristic must be expressable as a weighted sum in terms of the subtasks for the techniques reported on to apply.

Prior work, according to literature surveyed to date, appears to be limited to the paper of Chandy and Ramamoorthy [3] who investigated selection of memory hierarchies for a digital computer system in order to minimize average access time. Their problem is considerably different from that considered here but the general attributes are similar.

EXPLICIT PROBLEM STATEMENT

The techniques we hope to develop have a minimal throughput objective in mind but their application is by no means inextricably bound to this goal.

Two types of problems are considered, one being a subset of the other, and we give a specific possible application of each before general formulation. The first example is a problem commonly found in computer programming while the second is an illustration of how the problem of determining tradeoffs among joint hardware, software system design could be treated.

Type I

A computer program consisting of a main program calling several subroutines is required to fit in a memory of limited capacity. From deterministic or statistical analysis it is known with what frequency relative to the others that each subroutine is called. Suppose that for some or all of these subroutines there are several different implementations available each with differing execution times and memory requirements.

The Type I problem is that of selecting among the implementations available such that the total program throughput is minimized subject to the limited memory space. With this example in mind the general Type I problem statement is as follows.

Type I Index Set Formulation

Given: 1) A frequency n-tuple

$$F = \{f_1, f_2, \ldots f_i \ldots f_n\} \; ; \; f_i \in R \text{ (the real numbers)}, \; f_i \geq 0$$

2) A set of subtask execution times

$$T = \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_i \\ T_n \end{bmatrix} = \begin{bmatrix} \{t_{11}, t_{12}, \ldots t_{1m_1}\} \\ \{t_{21}, t_{22}, \ldots t_{2m_2}\} \\ \vdots \\ \{t_{i1}, t_{i2}, \ldots t_{im_i}\} \\ \{t_{n1}, t_{n2}, \ldots t_{nm_n}\} \end{bmatrix} \quad ; \; t_{ij} \in R \quad t_{ij} \geq 0$$

m_i is the number of implementations available for the ith subtask.

3) Corresponding to T a set of subtask implementation costs

$$C = \begin{bmatrix} \{c_{11}, c_{12}, \ldots c_{1m_1}\} \\ \{c_{21}, c_{22}, \ldots c_{2m_2}\} \\ \vdots \\ \{c_{i1}, c_{i2}, \ldots c_{im_i}\} \\ \{c_{n1}, c_{n2}, \ldots c_{nm_n}\} \end{bmatrix} \qquad c_{ij} \in R \quad c_{ij} \geq 0$$

4) A total cost constraint

$$C_T \in R$$

Then we seek an index set n-tuple

$$K = \{k_1, k_2, \ldots k_i \ldots k_n\}; \quad k_i \in Z \text{(the positive integers)}$$

$$\text{minimizing} \sum_{i=1}^{n} f_i t_{ik_i}$$

subject to

$$\sum_{i=1}^{n} c_{ik_i} \leq C_T.$$

Note. 1) The subscripts (i,j) are such that i refers to the subtask (the system then consists of n subtasks) and j refers to the jth available implementation of the ith subtask.

2) The possible number of system implementations is

$$\prod_{i=1}^{n} m_i.$$

3) There may be some fixed cost, C_F, associated with the complete system which case let a new $C'_T = C_T - C_F$

be used. In the software problem for example, memory taken up by the main program not including the subroutines could be considered a fixed cost. Of course if there were also several implementations of the main program available it could be considered a subtask itself with the appropriate usage frequency.

Type I Integer Programming Formulation

The Type I problem can also be given an integer programming formulation using bivalent (1 or 0) variables.

We seek $X = \{x_{ij} = 1 \text{ or } 0 / 1 \le i \le n; \quad 1 \le j \le m_i\}$

$$\text{minimizing} \quad \sum_{i=1}^{n} \sum_{j=1}^{m_i} x_{ij} \cdot f_i \cdot t_{ij}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \sum_{j=1}^{m_i} x_{ij} \cdot c_{ij} \le C_T$$

$$\sum_{j=1}^{m_1} x_{ij} = 1$$

$$\vdots$$

$$\sum_{j=1}^{m_n} x_{nj} = 1$$

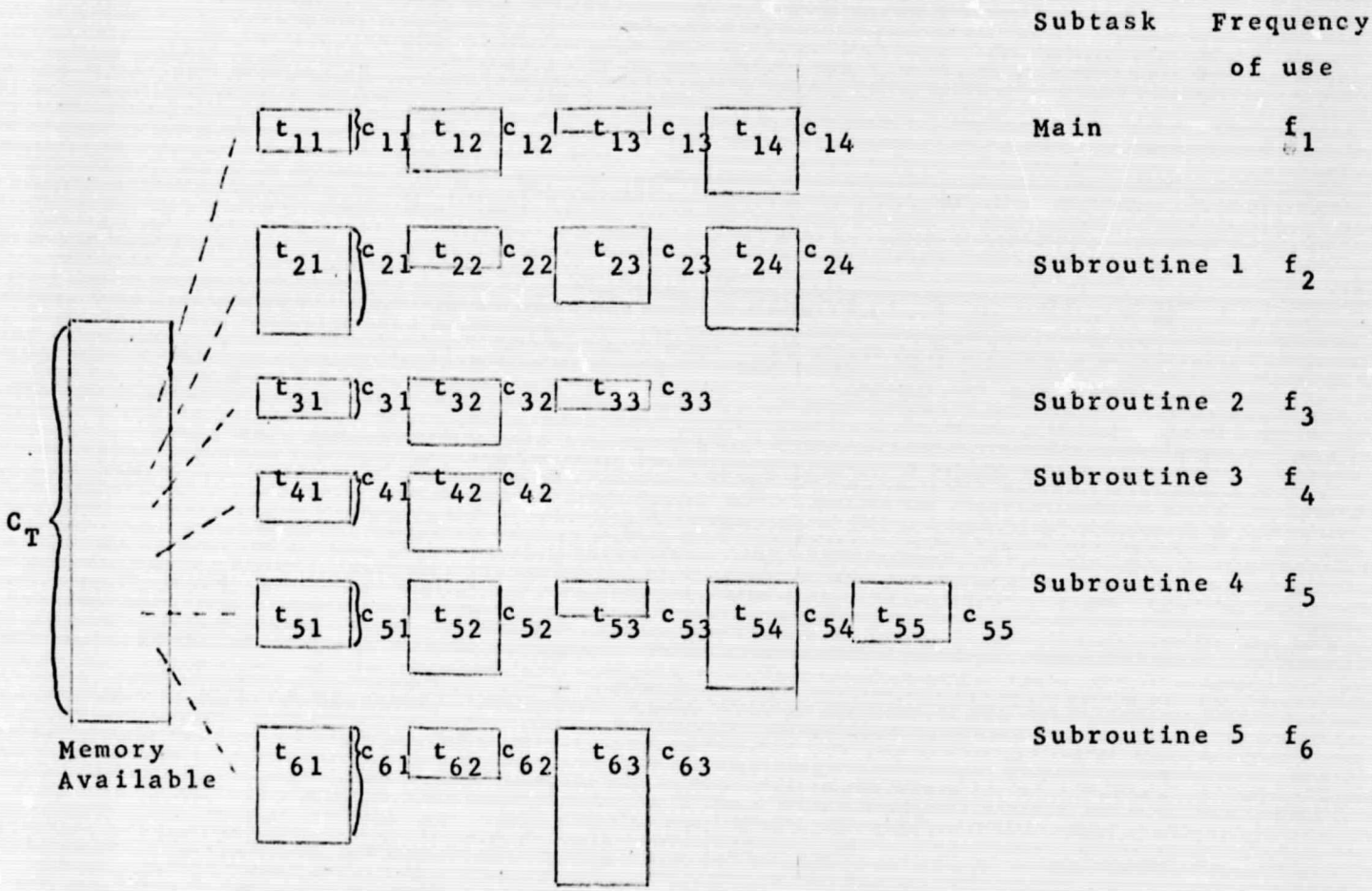
Note. 1) Constraints [2, n+1] force one and only one implementation to be selected from the set of implementations available for a given subtask.

2) In terms of the index set formulation an optimal solution X satisfies

$$x_{ij} = 1 \quad \text{if } k_i = j$$
$$= 0 \quad \text{otherwise.}$$

Type II

Consider the design of a digital computer system when choices must be made to distribute instruction implementation among various subsystems. Three such identifiable subsystem types might be hardwired logic, microprogramming and software for example. Typically some of these subsystems could be expected to perform only a subset of the total instruction set (clearly the case for software) thus necessitating use of several of the subsystems to implement the computer. Any subsystem might have costs peculiar to it; costs that are incurred or required even before any of the actual instructions are implemented in the subsystem. These costs are termed set-up costs and should be taken account of in any design tradeoffs. For example if the preceding was treated as a Type I problem, ignoring set-up costs, one result might be a solution containing a single instruction implementation via microprogramming. Modification



Available Implementations with
cost(memory required) and times(execution time) indicated.

CONCEPTUAL ILLUSTRATION OF THE TYPE I SOFTWARE PROBLEM

Note the close resemblance to the integer programming knapsack problem. In fact the Type I problem can be stated as a transportation problem but we are interested in structure considerations at the moment as well as solution techniques.

of the Type I formulations to include set-up costs is however relatively easy.

Type II Index Set Formulation

Given: 1) A frequency n-tuple

$$F = \{f_1, f_2, \ldots f_i \ldots f_n\};\ f_i \in R$$

2) A nxm set of subtask execution times

$$T = \begin{bmatrix} t_{11}, t_{12} \ldots t_{1m} \\ \vdots \\ t_{i1}, t_{i2} \ldots t_{im} \\ \vdots \\ t_{n1}, t_{n2} \ldots t_{nm} \end{bmatrix} ;\ t_{ij} \in R$$

3) Corresponding to T a nxm set of implementation costs

$$C = \begin{bmatrix} c_{11}, c_{12} \ldots c_{1m} \\ c_{i1}, c_{i2} \ldots c_{im} \\ c_{n1}, c_{n2} \ldots c_{nm} \end{bmatrix} ;\ c_{ij} \in R$$

4) A m-tuple of set-up costs

$$S = \{S_1, S_2 \ldots S_j \ldots S_m\};\ S_j \in R \wedge 1 \leq j \leq m \wedge S_j \geq 0$$

5) A total cost constraint

$$C_T \in R$$

Then we seek an n-tuple index set

$$K = \{k_1, k_2, \ldots k_i \ldots k_n\} \; ; \; k_i \in Z$$

minimizing $\sum_{i=1}^{n} f_i, t_{ik_i}$

s.t $\sum_{i=1}^{n} c_{ik_i} + \sum_{j \in K} S_j \leq C_T$.

Note. 1) The subscripts (i,j) are such that i refers to the subtask and j refers to the subsystem.

2) If for a given i there is no jth implementation then set $t_{ij} = \infty$ and $c_{ij} = 0$.

3) With Note 2) above Notes 2) and 3) of the Type I problem still apply.

Type II Integer Programming Formulation

Change of the Type I bivalent integer programming formulation to handle the Type II problem is also possible but somewhat tricky. It is accomplished by extending the single variable set-up cost technique commonly used in operations research (See Hillier and Lieberman [1, p. 564] for example) to include variable summations. Introducing additional bivalent artificial variables $Y = \{y_j / 1 \leq j \leq m\}$ we have

Determine X, Y

$$\text{minimizing } \left\{ \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} \cdot f_i \cdot t_{ij} + \sum_{j=1}^{m} y_j \cdot S_j \right\}$$

s.t

$$\sum_{i=1}^{n} x_{i1} - M \cdot y_1 \leq 0$$

$$\vdots$$

$$\sum_{i=1}^{n} x_{im} - M \cdot y_m \leq 0$$

$$\sum_{j=1}^{m} x_{1j} = 1$$

$$\vdots$$

$$\sum_{j=1}^{m} x_{nj} = 1$$

$$\sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} \cdot c_{ij} + \sum_{j=1}^{m} y_j \cdot S_j \leq C_T$$

Note. 1) $M > n$ is an arbitrary constant.

2) To understand the strategy behind the use of the artificial variables, Y, observe that for a specified j a non-zero condition for $\sum_{i=1}^{n} x_{ij}$ forces $y_j = 1$ to satisfy the jth constraint. If $\sum_{i=1}^{n} x_{ij} = 0$, the constraint no longer specifies a value for y_j but then the minimization operation of the objective function forces $y_j = 0$ as desired.

GENERAL DISCUSSION

Before continuing we should like to mention several points concerning the problems.

In particular there is an implicit assumption made concerning the interconnection of subtask implementations to form the total system. The assumption is that interconnection costs are negligible or at least reasonably constant among all the possible implementation interconnections. In effect we are saying that the system may be synthesized in "assembly line" fashion by arbitrarily choosing one implementation per subtask and then assembling them to form the system. Widely varying interconnection costs can make the approach we are considering essentially useless. Some of this difficulty can possibly be alleviated, when certain implementations for several subtasks interconnect more or less naturally but incur excessive interconnection costs when used with other implementations, by considering them as a subsystem in the Type II manner.

Another encouraging aspect in this vein is that current device technology, in particular LSI if and when it begins to make the impact promised by its supporters, requires a "functional partitioning" approach to design in order to counteract such problems as pin limiting and maintenance. This partitioning is close to what our "assembly line" assumption requires. Linhardt and Miller [9], for example, discuss architecture required for LSI and specify a digital computer structure consisting of "Instruction Execution Units" tied into a general communications bus with each of these units responsible for one instruction or a similar set of instructions.

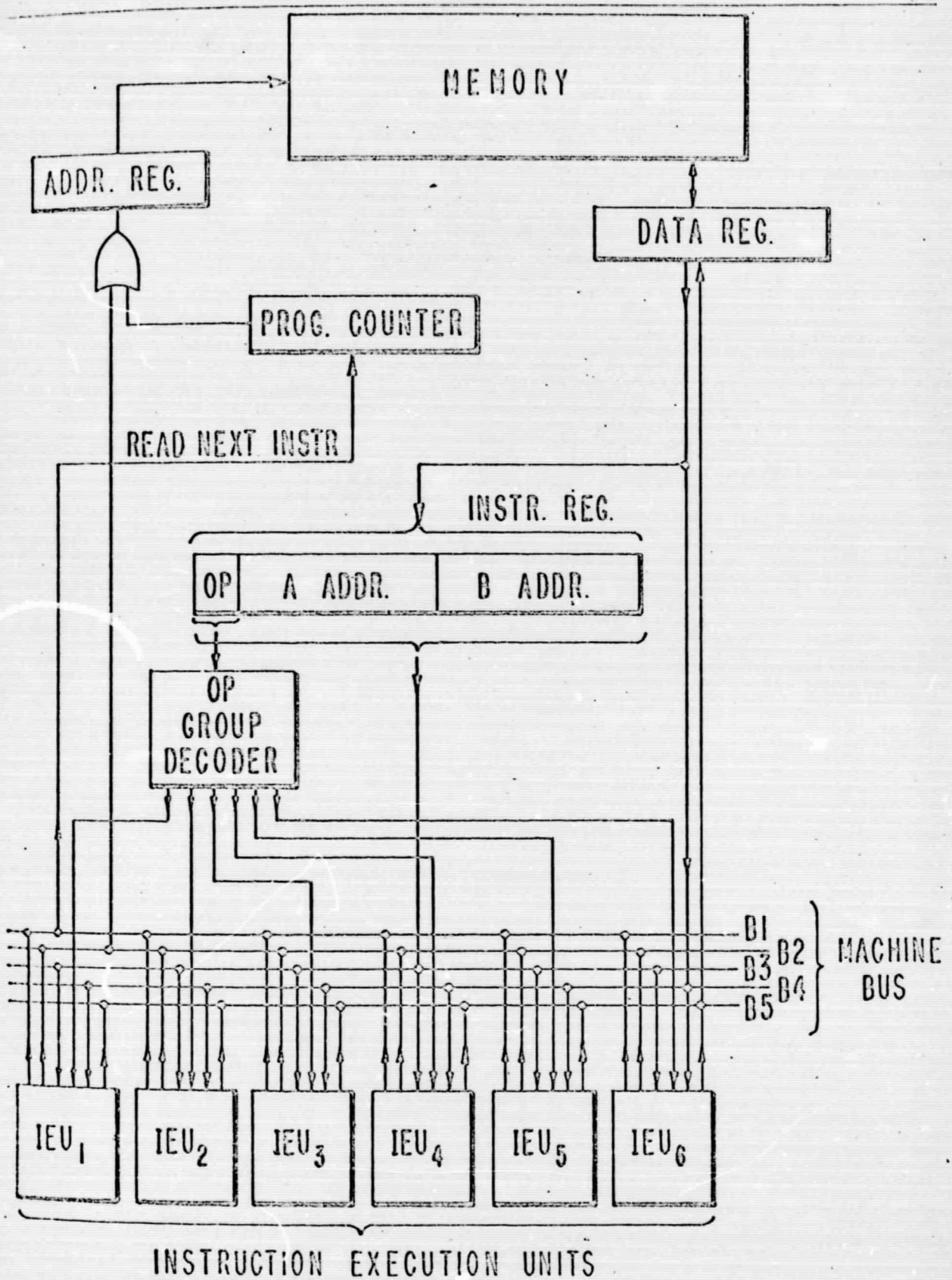


Figure 1--General block diagram of a register machine. Five parts of the machine bus are: B1-read next instruction flag; B2-memory address; B3-instruction word; B4-data word from memory; and B5-data word to memory

In addition to determining solutions, certain sensitivity information is also desirable such as the effect on the optimal solution value of the objective function from changes in C_T. This type of information is best presented in a plot of the objective function values vs C_T (See Figure 3). Other important sensitivity information is the effect on the solution from parameter changes in T, C, S and F. For practical usefulness this information is vital as much of the required information may not be determinable to an accurate degree.

Again from a practical standpoint we would like to be able to apply our techniques to fairly complex systems possibly involving several hundred subtasks. This could easily result in from a few hundred to a thousand variables in the integer programming formulation. For general integer programming algorithms this is completely impractical. In the bivalent case, various techniques developed by Hammer, Balas and Glover[6,1,4] are considerably more efficient. However there are indications that these techniques would also be strained in handling several hundred variables [5, p. 178] even though problems of Type I and II are restricted forms of the general bivalent problem (except for the one implementation/subtask requirement there could be $2^{\sum_{i=1}^{n} m_i}$ possible system implementations

instead of $\prod_{i=1}^{n} m_i$.).

Consequently for the purposes of this preliminary report the integer programming formulations are given only as "standards of comparison" and we shall concentrate instead on a structural analysis. At this early time in the research effort we are only able to give a few results on the structure of the Type I problem.

TYPE I ANALYSIS

Introductory Concepts

Certain concepts from abstract algebra will be required for the remainder of this report, particularly the idea of a lattice. Consequently we briefly review some necessary definitions. For a more complete presentation see the definitive work on lattice theory by Birkhoff[2] or a shorter but eminently more readable presentation by Rutherford [8].

Definition*: Given a set S and a binary relation $\geq$ on S such that:

1) $\geq$ is reflexive, i.e. $\forall x \in S \quad x \geq x$

2) $\geq$ is antisymmetric, i.e. $\forall x y \in S x \geq y \wedge y \geq x \Rightarrow x = y$

3) $\geq$ is transitive, i.e. $\forall x,y,z \in S \quad x \geq y \wedge y \geq z \Rightarrow x \geq y$

Then $(S,\geq)$ is said to be a <u>partially ordered set</u> or <u>poset</u>. The relation $\geq$ is commonly called an <u>inclusion</u> or <u>order</u> relation and with $\geq$ the binary relations $\leq$, $>$, $<$ can be

*Unless indicated otherwise most of these definitions follow Rutherford.

Introduced

$x \leq y \iff y \geq x$

$x > y \iff y \geq y \wedge x \neq y$

$x < y \iff y \leq y \wedge x \neq y$

Note that $x \not\geq y$ does not necessarily imply that $x < y$.

Continuing observe that if $(S,\geq)$ is a poset then $(S,\leq)$ is also. The structure $(S,\leq)$ is termed the dual of $(S,\geq)$. Especially observe that any theorem valid for posets is also valid for their duals.

Definition If an element $\varphi \in (S,\geq) \ni \varphi \leq x \ \forall x \in S$ then φ is the null element of $(S,\geq)$. Similarly if an element $I \in (S,\geq) \ni I \geq x \ \forall x \in S$ then I is termed the universal element of $(S,\geq)$. Note that φ (if it exists) and I (if it exists) are unique due to the antisymmetric property and that $\varphi \in (S,\geq) = I \in (S,\leq)$ and $I \in (S,\geq) = \varphi \in (S,\leq)$.

Definition $(S,\geq)$ is called a chain if $\forall x, y \in S \ x \geq y \vee y \geq x$.

Definition** By x covers Y it is meant that $x > y \wedge x > z > y$ is not satisfied by any $z \in S$.

Let $S' \subset S$

Definition***Let $I \in S'$ then S' is termed a covered set if every $x' \in S'$ (except I) is covered by at least one other member $y' \in S'$.

** Birkhoff

***We confess to this one. For our requirements it has certain useful computational qualities.

Definition An element $x \in S$ is said to be an upper bound (u.b.) of S' if $x \geq x' \ \forall x' \in S'$. Similarly an element $y \in S$ is said to be a lower bound (l.b.) of S' if $y \leq y' \ \forall y' \in S'$.

Definition An upper bound $x \in S$ of S' is said to be the least upper bound (l.u.b.) of S' if every upper bound x^* satisfies $x^* \geq x$. Similarly an element $y \in S$ is a greatest lower bound (g.l.b) of S' if y is a lower bound and every y^* satisfies $y^* \leq y$. Both the l.u.b. and g.l.b. are unique if they exist again due to the anti-symmetrical property.

Definition A lattice $\mathcal{L}$ is a poset such that every pair of elements possesses an l.u.b. and a g.l.b. Upon occasion we will follow common practice and denote the l.u.b. of x and y by $x \cup y$ and the g.l.b. x and y by $x \cap y$.

Definition A sublattice $\mathcal{L}' \subset \mathcal{L}$ is a subset $\ni x \in \mathcal{L}' \wedge y \in \mathcal{L}' \Rightarrow x \cap y \in \mathcal{L}' \wedge x \cup y \in \mathcal{L}'$.

Definition A function $f: \mathcal{L} \to R$ is said to be a valuation of $\mathcal{L}$ if $f(x) + f(y) = f(x \cap y) + f(x \cup y)$. Further if $x > y \Rightarrow f(x) > f(y)$ the valuation is termed positive.

Definition A lattice $\mathcal{L}$ is said to be metric if there exists a positive valuation on the elements of $\mathcal{L}$.

Definition A lattice $\mathcal{L}$ is said to be distributive if $\forall x, y, z \in \mathcal{L}$

$(x \cap y) \cup (y \cap z) \cup (z \cap x) = (x \cup y) \cap (y \cup z) \cap (z \cup x)$.

Structure Analysis

Let L (representing a system implementation) be an n-tuple with each component bounded by m_i, i.e.

$$L = (\ell_1, \ell_2, \ldots \ell_i \ldots \ell_n) \ni \ell_i \in Z \wedge 1 \leq \ell_i \leq m_i$$

Then $T(L) = (t_{1\ell_1}, t_{2\ell_2}, \ldots t_{i\ell_i} \ldots t_{n\ell_n})$ is termed a time set.

And $C(L) = (c_{1\ell_1}, c_{2\ell_2} \ldots c_{i\ell_i} \ldots c_{n\ell_n})$ is termed a cost set.

The objective function can now be represented as a real valued function

$$O_f(F, T(L)) = \sum_{i=1}^{n} f_i \cdot t_{i\ell_i} = O_f(L) \text{ for fixed } F, T.$$

Also a real valued cost function can be given

$$C_f(C(L)) = \sum_{i=1}^{n} c_{i\ell_i} = C_f(L) \text{ for fixed } C.$$

Theorem I Consider two system implementations L^1, L^2 identical except that $\ell_i^1 \neq \ell_i^2$ for some i; $1 \leq i \leq n$. If

$t_{i\ell_i}^1 > t_{i\ell_i}^2$ has corresponding costs

$c_{i\ell_i}^1 \geq c_{i\ell_i}^2$

then $k_i \neq \ell_i^1$

for any C_T.

Proof Clearly $C_f(L^2) \leq C_f(L^1)$. Thus whenever $c_{i\ell_i}^1$ is part of a cost set that satisfies $C_f(L^1) \leq C_T$ then since $C_f(L^2) \leq C_f(L^1) \leq C_T$ so must the L^2 cost set. Consequently as $O_f(L^2) < O_f(L^1)$ the

L^2 time set results in a minimal objective function value for the L^1, L^2 pair and as long as $t_{i\ell_i^2} \in T_i$, ℓ_i^1 can never be part of an optimal solution (violates theorem of optimality). Thus $k_i \neq \ell_i^1$ Q.E.D.

Clearing away notational fog, Theorem I simply states that a subtask implementation costing more and not performing as well as another implementation of the same subtask need not be considered for an optimal solution. Observe however that this does not necessarily imply that $k_i = \ell_i^2$.

Theorem II Let $\min\{T_i\} = \min\{t_{i1}, t_{i2} \ldots t_{im_i}\} = t^*_{im_i^*}$. Then $k_i = m_i^*$ for some C_T^*.

Proof: $C_T^* \geq \sum_{i=1}^{n} C_{im_i^*}$ is the required C_T^*.

Theorem II says that the best performing subtask implementation will always belong to an optimal solution containing only "best" implementations if we can afford it.

From Theorems I and II we have the following:

Corollary (Monotonicity)

$\forall\, T_i \;\; \exists\, T_i^* \subseteq T_i$ satisfying

i) T_i^* can be ordered $\}$

$t_{i1^*} > t_{i2^*} > \ldots > t_{im_i^*}$

And $c_{i1^*} < c_{i2^*} < c_{im_i^*}$ $\}\, 1 \leq m_i^* \leq m_i$

ii) $t_{ik_i} \in T_i^*$ if a solution exists for a given C_T.

The monotonicity corollary is an extension of Theorem I across an entire set of implementations for a subtask. Theorem II allows a convenient starting point for the pairwise checking required for the well ordering.

Eventually we would like to add additional cost functions with constraints (say j of them) $C_f^j(L) \leq C_T^j$ similar to $C_f(L) \leq C_T$ or $C_j^1(L) \leq C_T^1$ in use. The effect of additional cost constraint equations on Theorem I is to require that $c_{i\ell_i^1}^j \geq c_{i\ell_i^2}^j \; \forall j$ before ℓ_i^1 can be ignored. For if $c_{i\ell_i^1}^k < c_{i\ell_i^2}^k$ for any k, then assuming all other $C^j(L^1)$ and $C^j(L^2)$ would both satisfy their constraints there exists a constraint value for C_T^k such that $C^k(L^1)$ satisfies it but $C^k(L^2)$ does not. This weakens the monotonicity corollary. However that is a subject for future worry.

Returning to the problem at hand, note that the mapping $H_i; (1^*, 2^*, \ldots m_i^*) \rightarrow (1, 2, \ldots m_i)$ resulting from the ordering should be recorded to allow reference back to the original subscript designations. Also we note in passing that the corollary and two theorems bear an interesting resemblance to the "derived hierarchy" theorem of Chandy and Ramamoorthy [3, p. 513].

From this point on it will be assumed that

$$T = \begin{bmatrix} T_1^* \\ \vdots \\ T_n^* \end{bmatrix} \quad \text{with each } T_i^* \text{ ordered}$$

according to the monotonicity corollary.

Definition The set of all possible system implementations is

$$\mathcal{L} = \{(\ell_1, \ell_2, \ldots \ell_i \ldots, \ell_n) \mid \ell_i \in Z \wedge 1 \leq \ell_i \leq m_i^*\}$$

The order of $\mathcal{L} = \prod_{i=1}^{n} m_i^*$ and clearly $K \in \mathcal{L}$ where the elements of $K = k_i \in (1^*, 2^*, \ldots m_i^*)$.

Theorem III $(\mathcal{L}, \geq)$ where $L^1 \geq L^2; L^1, L^2 \in \mathcal{L}$ iff $\ell_i^1 \geq \ell_i^2 \; \forall i; 1 \leq i \leq n$ is a lattice.

Proof: $\mathcal{L}$ is a poset by virtue of its elements being composed of members of Z. Now let

$$L^* = [\max(\ell_1^1, \ell_1^2), \ldots, \max(\ell_n^1, \ell_n^2)] \; \forall \; L^1, L^2 \in \mathcal{L}.$$

We claim $L^* = L^1 \cup L^2$.

Clearly L^* is an upper bound of L^1, L^2. Suppose L^* is a u.b. but not the l.u.b.. Then $\exists$ some $L^{**} < L^* \ni L^{**} \geq L^1 \wedge L^{**} \geq L^2$. This implies $\exists$ at least one $\ell_i^{**} \geq \ell_i^1 \wedge \ell_i^{**} \geq \ell_i^2 \ni \ell_i^{**} < \max(\ell_i^1, \ell_i^2)$. Assume $\max(\ell_i^1, \ell_i^2) = \ell_i^1$. Then $\ell_i^{**} \geq \ell_i^1 \wedge \ell_i^{**} < \ell_i^1$ contradiction. Using a similar argument for $\max(\ell_i^1, \ell_i^2) = \ell_i^2$ shows $L^* = L^1 \cup L^2$. Letting

$$L^1 = [\min(\ell_1^1, \ell_1^2), \ldots \min(\ell_n^1, \ell_n^2)] \text{ for the g.l.b.}$$

and arguing dually completes the proof.

The usefulness of Theorem III results from

Theorem IV

1) $C_f(L)\ \forall\ L \in \mathcal{L}$ is a positive valuation on $(\mathcal{L}, \geq)$

2) $O_f(L)\ \forall\ L \in \mathcal{L}$ is a positive valuation on $(\mathcal{L}, \leq)$

Proof:

1) We must show that $\forall\ L^1, L^2 \in \mathcal{L}$

i) $C_f(L^1) + C_f(L^2) = C_f(L^1 \cup L^2) + C_f(L^1 \cap L^2)$

ii) if $L^1 > L^2$ then $C_f(L^1) > C_f(L^2)$

i) $$C_f(L^1 \cup L^2) + C_f(L^1 \cap L^2) = \sum_{i=1}^{n} c_{i \max(\ell_i^1, \ell_i^2)} + \sum_{i=1}^{n} c_{i \min(\ell_i^1, \ell_i^2)}$$

$$= \sum_{i=1}^{n} c_{i\ell_i^1} + \sum_{i=1}^{n} c_{i\ell_i^2} \qquad \ell_i^1 \geq \ell_i^2$$

$$= C_f(L^1) + C_f(L^2)$$

$$= \sum_{i=1}^{n} c_{i\ell_i^2} + \sum_{i=1}^{n} c_{i\ell_i^1} \qquad \ell_i^2 \geq \ell_i^1$$

$$= C_f(L^1) + C_f(L^2)$$

Q.E.D. i)

ii) Given the monotonicity corollary this follows easily since the members of T and F are nonnegative.

2) Requires for $(\mathcal{L}, \geq)$ that

i) $O_f(L^1) + O_f(L^2) = O_f(L^1 \cup L^2) + O_f(L^1 \cap L^2)$

ii) If $L^1 > L^2$ then $O_f(L^1) < O(L^2)$

both of which use essentially the same arguments as in 1) thus completing the proof.

Before discussing Theorem IV further we state the following.

Theorem V $(\mathcal{L}, \geq)$ is a metric lattice. Follows directly from 1) in Theorem IV.

Theorem VI $(\mathcal{L}, \geq)$ is a distributive lattice. (Not proved although straightforward).

It can be seen that $(\mathcal{L}, \geq)$ contains both φ and I. In fact $\varphi = \{1,1,1,\ldots,1\}$ an n-tuple of 1's and $I = \{m_1^*, m_2^*, \ldots, m_n^*\}$. Note that I corresponds to the "best" system implementation referred to in Theorem II.

Again clearing away noise, Theorem IV's significance is that by a relatively easy ordering process that can be done independently on each subtask we are able to determine a considerable amount about objective and cost function value relationships essentially independent of T and C member values. Note also that we are able to determine immediately whether a feasible solution exists by checking to see if $C_T \geq C_f(\varphi)$. An example system implementation lattice with objective and cost function values is given in Figures 1 and 2.

We now attempt to restrict the search for optimal solutions in $\mathcal{L}$. Define the set $D \subseteq \mathcal{L}$ recursively

$D(1) = (m_1^*, m_2^*, \ldots, m_n^*) = I \in (\mathcal{L}, \geq)$

$D(N+1) = L \in \mathcal{L}$) $\{O_f(L) - O_f(D(N))\} \geq 0$ and minimum with $\{C_f(L) - C_f(D(N))\} < 0$ if and $|C_f(L) - C_f(D(N))|$ maximum. In other words, there are several having $\min \Delta O_f$ choose one with $\max |\Delta C_f|$. See Figures 2 and 3 for examples of D.

Theorem VII $L \in D$ iff L is an optimal solution for some C_T. (Not proven)

Roughly, since the first member of D is optimal for some $C_T (C_T \geq C_f(I))$ a minimum positive decrease in objective function value in moving to the next member of D with a lower cost implies that member is also optimal for some value of C_T, etc.

Note that $\varphi(\mathcal{L}, \geq) \in D$ also and in fact is the "last" member of D. The D set itself is important not only because it contains all and only optimal solutions but because in doing so it is a complete description of the sensitivity of the problem with respect to C_T. By taking advantage of $\mathcal{L}$'s lattice structure it is possible, given D(N) to calculate its successor D(N+1) by examining only a subset of $\mathcal{L}$. However calculating the entire D set appears to involve interrogating the entire lattice. Consequently calculation of the members in some neighborhood around the optimal solution to obtain sensitivity information would be more practical.

Another item of interest, the order of D, is dependent on values of T and C whereas $\#(\mathcal{L})$ is a function only of the m_i. Empirical experience suggests that #(D) is however proportional to the number of "levels" in $\mathcal{L}$ (a level is all $L \in \mathcal{L} \ni \sum_{i=1}^{n} \ell_i = P; n \leq P \leq \sum_{i=1}^{n} m_i$, see figure 1 for example), at least for lattices in which the number of subtasks is large compared with the average number of implementations of subtasks. In one system with n = 7 and $m_i = 2$, $1 \leq i \leq n$ the #(D) = 10 vs #(L) = 128. This is interesting as this latter condition would be typical of practical applications.

The next question is whether or not D possesses a useful structure. From a computational standpoint the chain and covered set structures would be desirable but as clearly shown by the example of Figure 3 this is not the case. We suspect that D may be a sublattice but have not been able to prove or disprove this contention.

Finally note that we may form a D' set on $(\mathcal{L}, \leq)$ corresponding to D on $(\mathcal{L}, \geq)$ and that these two sets, with identical elements but with reverse indices, behave in a fashion analogous to duality in mathematical programming.

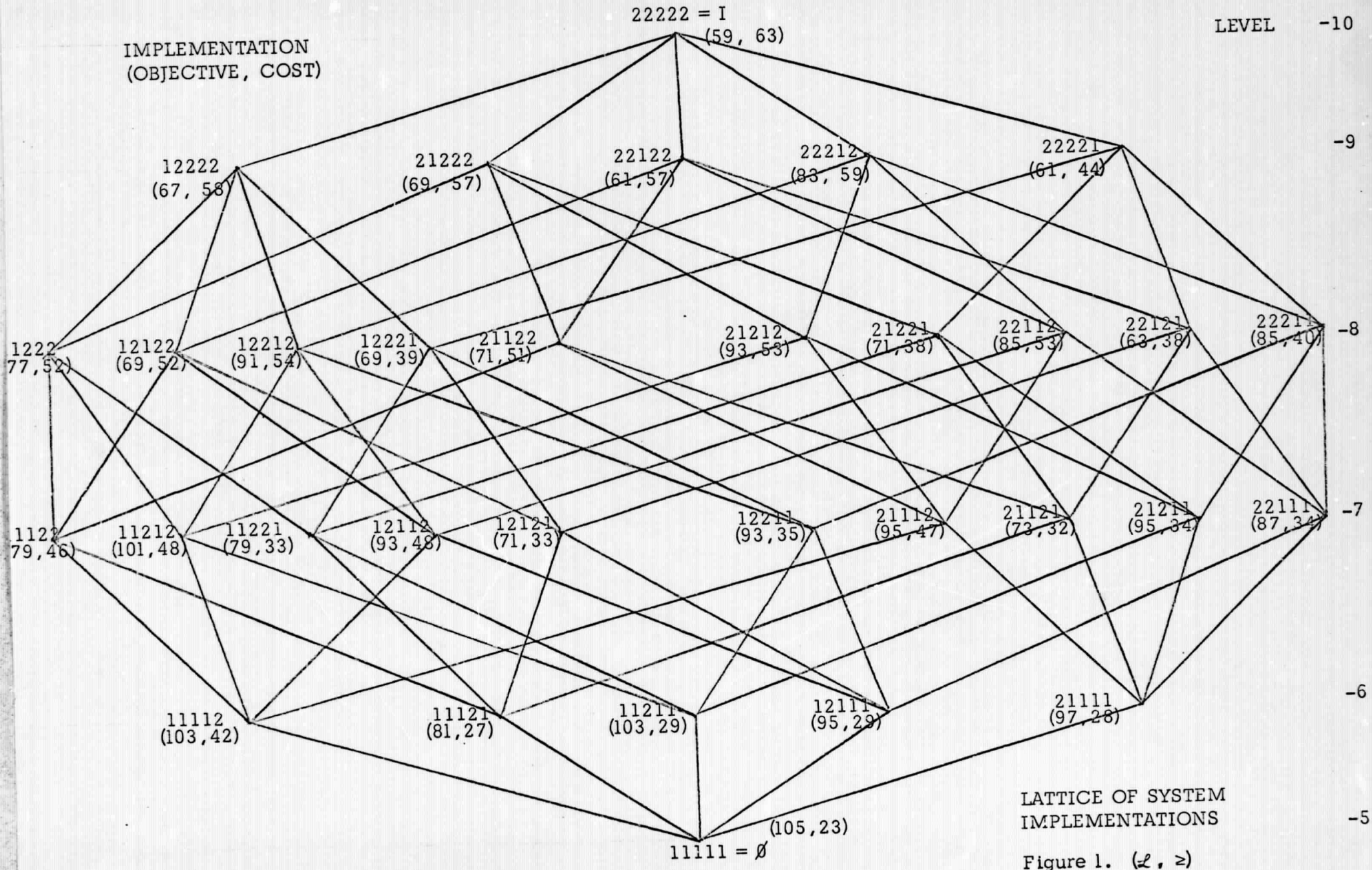
$$D'(1) = (1,1,\ldots 1) = I \in (\mathcal{L}, \leq) = \varphi \in (\mathcal{L}, \geq)$$

$$D'(N+1) = L \in \mathcal{L} \ni \{C_f(L) - C_f(D'(N))\} \geq 0 \text{ and minimum with}$$

$$\{O_f(L) - O_f(D'(N))\} < 0 \text{ and } |O_f(L) - O_f(D'(N))| \text{ maximum.}$$

With these two definitions the members of D as ordered by their recursive indices move from the "best" but typically nonfeasible solution, $I \in (\mathcal{L}, \geq) = \varphi \in (\mathcal{L}, \leq) = D(1)$ toward successively less best and more feasible solutions while members of D' move from the feasible but typically less than optimal solution, $I \in (\mathcal{L}, \leq) = \varphi \in (\mathcal{L}, \geq) = D'(1)$ toward optimality. This is exactly the manner in which solutions under dual and primal techniques are supposed to behave in mathematical programming.

This has been only a preliminary report giving research progress after about two months on the problem. The final goal is, of course, computer algorithms for solution of both Type I and II problems as well as determination of desired sensitivity information.



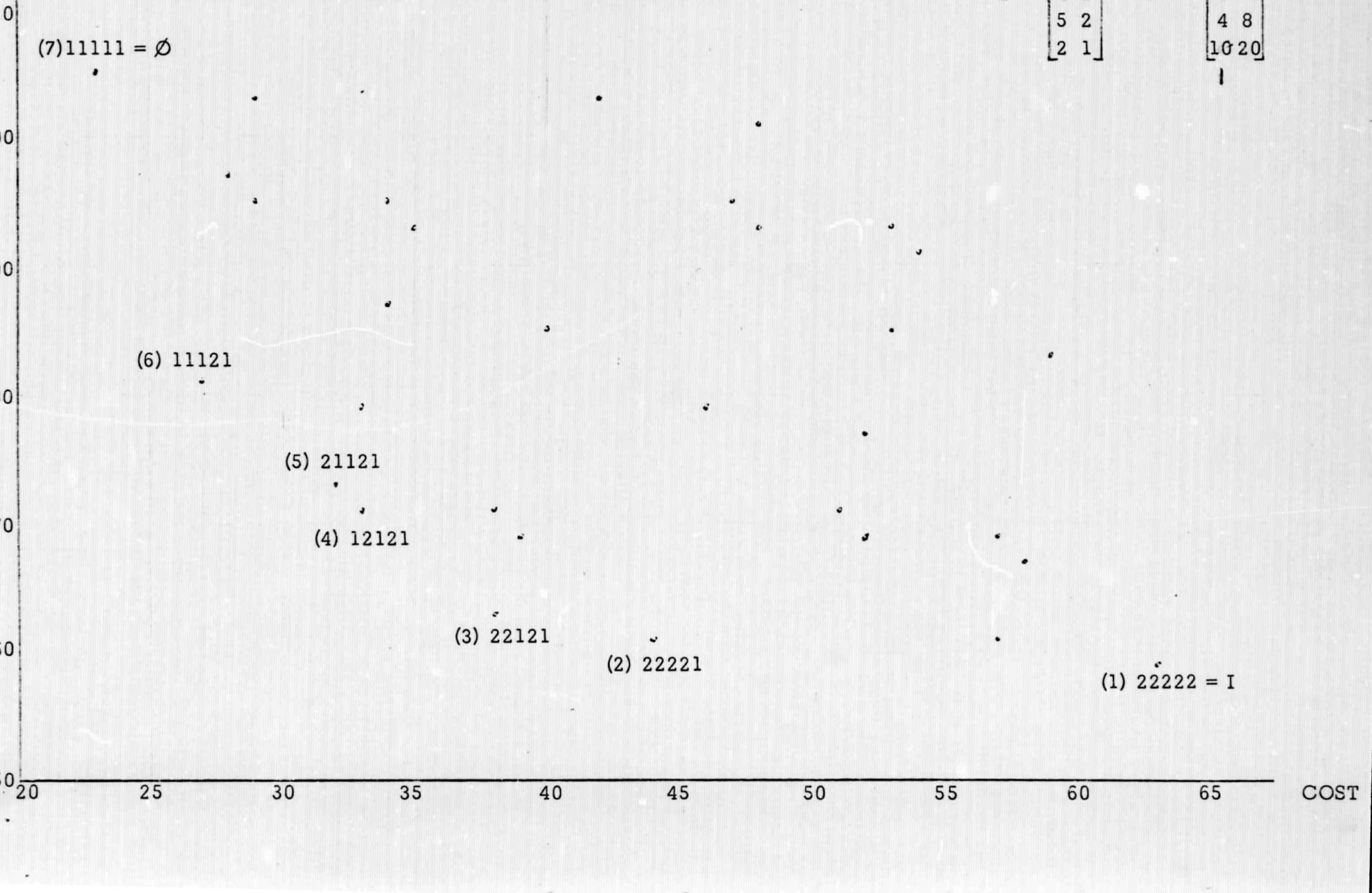
LATTICE OF SYSTEM IMPLEMENTATIONS

Figure 1. $(\preceq, \geq)$

FIGURE 2)
OBJECTIVE VS. COST
FOR THE LATTICE OF
FIGURE 1). WITH F, T, AND C
AS SHOWN. MEMBERS OF D ARE INDICATED BY ( ).

$$F = [4, 5, 1, 8, 2]$$

$$T = \begin{bmatrix} 3 & 1 \\ 9 & 7 \\ 4 & 2 \\ 5 & 2 \\ 2 & 1 \end{bmatrix} \qquad C = \begin{bmatrix} 4 & 9 \\ 10 & 16 \\ 4 & 10 \\ 4 & 8 \\ 10 & 20 \end{bmatrix}$$



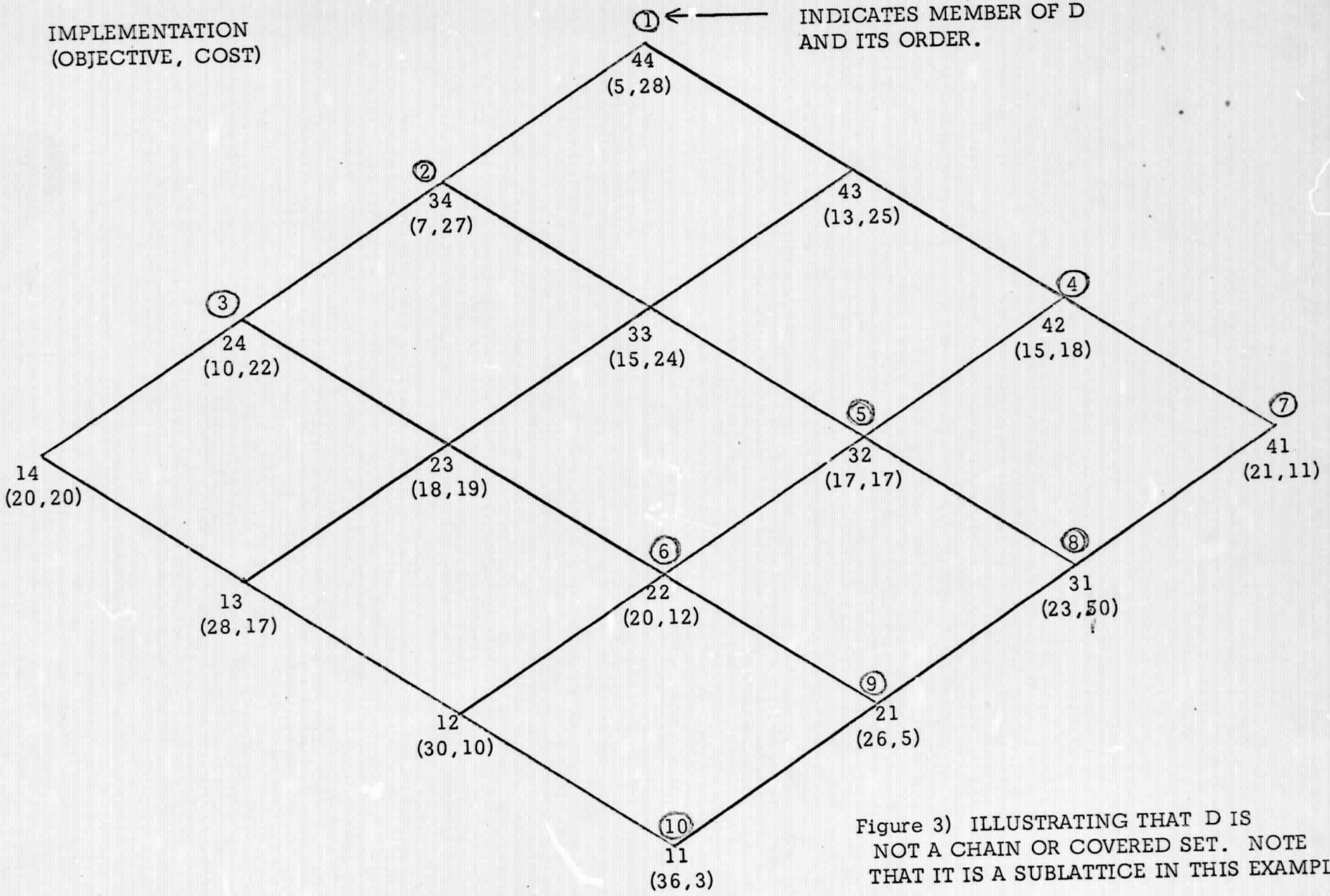


Figure 3) ILLUSTRATING THAT D IS NOT A CHAIN OR COVERED SET. NOTE THAT IT IS A SUBLATTICE IN THIS EXAMPLE.

REFERENCES

1. Balas, E., "An Additive Algorithm for Solving Linear Programs with Zero-One Variables", Operations Research 13, pp. 517-545, 1965.

2. Birkhoff, G. D., Lattice Theory, American Mathematical Society Colloquium Publication #XXV, 1948.

3. Chandy, K. M. and C. V. Ramamoorthy, "Optimization of Information Storage Systems", Information and Control 13, pp. 509-520, 1968.

4. Glover, F., "A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem", Operations Research 13, pp. 879-919, 1965.

5. Graves, G. and A. Whinston, "A New Approach to Discrete Mathematical Programming", Management Science 15 #3, pp. 177-190, 1968.

6. Hammer, P. L. and S. Rudeanu, Boolean Methods in Operations Research, Springer Verlag, New York, 1968.

7. Hillier, F. S. and G. J. Lieberman, Introduction to Operations Research, Holden-Day Inc., San Francisco, 1968.

8. Rutherford, D. E., Introduction to Lattice Theory, Hafner Publishing Company, London, 1965.