An Algorithmic Solution to the Minimax Resource Allocation Problem with Multimodal Functions

by

Aida Dharmakadar

Thesis submitted to the Faculty of
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTERS OF SCIENCE

in

Computer Science and Applications

APPROVED: [illegible]
E. Haddad, Chairman

William May
W. May

Richard E. Schneider
R. Schneider

September, 1993
Blacksburg, Virginia

c.2

LD
5655
V855
1993
D523
c.2

An Algorithmic Solution to the Minimax Resource Allocation Problem with Multimodal Functions

by

Aida Dharmakadar

Committee Chairman: Emile Haddad
Computer Science

(ABSTRACT)

An algorithmic approach is developed for solving the minimax continuous resource allocation problem with multimodal cost functions. Unlike previous research in the same area which developed solutions for the same problem by imposing restrictions on the cost functions, such as the assumptions of monotoniciy or convexity, this approach is applicable to problems with multimodal functions with a finite number of local extrema. Another significant advantage demonstrated by this approach is that it provides *all* the optimal solutions to the problem; in contrast to previous algorithms which provided a single optimal solution. When a further level of optimization using a second objective function is desired, one needs the entire set of optimal solutions as provided by the procedures of this thesis.

Acknowledgements

I would like to thank my thesis advisor, Professor Emile Haddad, and the members of my thesis committe, Professor William May and Professor Richard Schneider for all their guidance, support, and valuable comments.

I would also like to express special thanks to Dr. William Frakes and Ms. Linda Kramp, who provided me with the administrative support and facilities I needed to complete this thesis.

I certainly appreciated the generosity of Ms. Martha Cannon and Ms. Adaora Onyia, the Computer Science Graduate Assistants 1993-1994, for allowing me to use their computer facilities (in addition to room and snacks).

Table of Contents

List of Figures

1. Introduction and Background

The problem of resource allocation is one of the most widely investigated topics in the area of mathematical optimization because of its broad applicability to different classes of real world problems. The basic idea is that, given some type of resource whose total amount is N, we want to partition and allocate the total resource over n activities to minimize some objective function, F, whose value represents the "cost" of the allocation. This is formally stated as follows:

$$\begin{array}{ll} \text{Minimize} & F(x) = F(x_1, x_2, \ldots, x_n) \\ \text{subject to} & \sum_{i \in I} x_i = N, \text{ where } I = \{1, 2, \ldots, n\} \end{array}$$

where each nonnegative x_i is the amount of resource assigned to activity i.

In the above problem, if the resource is continuously divisible, then each x_i can be any real non-negative number, i.e., the domain of the problem is continuous. However, when the elements of the resource are of the discrete type, each x_i is modeled as an integer variable. The continuous divisibility versus discrete resource may seem to entail only minor differences in solving the corresponding problem. However, as can be seen from various research results in this area, the procedures to solve discrete and continuous resource allocation problems can be significantly different. One obvious basic difference is that the discrete problem can be exactly solved by exhaustive enumeration, while the continous problem cannot.

A simple variant of the above problem formulation arises when the objective function F(x) is to be maximized instead of minimized. This would be the case when F(x) represents some "profit" or "gain" rather than cost of allocation. As stated in

[IbKa88], there is no significant difference between the solutions of the two problem formulations because maximizing F(x) is essentially the same as minimizing -F(x).

In practical problems, each x_i is usually assigned some upper bound u_i representing a ceiling of the amount of resource that can be allocated to the ith activity. In general, a lower bound l_i may be also prescribed for the variable x_i. The problem now becomes

Minimize $F(x) = F(x_1, x_2, ..., x_n)$

subject to $\Sigma x_i = N$, where $l_i \leq x_i \leq u_i$

Clearly, for this problem to have a solution, we need to have the following condition satisfied by the given values of l_i and u_i :

$\Sigma l_i \leq N \leq \Sigma l_i$.

The wide applicability of resource allocation problems can be seen from the various types of *objective functions* represented by F(x). The common used ones are:

1 *Separable*: Minimize $F(x) = F(x_1, x_2, ..., x_n) = \sum_{i \in I} f_i(x_i)$, where each f_i is a function associated with each x_i. References [ChCo58], [EGP76], [Sa70], [Zi82], [FeZi83], and [Ko71] provide examples of different applications using this objective function and proposed different approaches to solve them.

2 *Fair*: Minimize $F(x) = F(x_1, x_2, ..., x_n) = g(\max\{f_i(x_i)\}, \min\{f_i(x_i)\})$, where g(u,v) is nondecreasing with respect to u, and nonincreasing with respect to v. Some applications and solutions of this problem can be found in [FKI88] and [KIM85].

3 *Minimax*: Minimize $F(x) = F(x_1, x_2, ..., x_n) = \max\{f_i(x_i)\}$. References that deal with the *Minimax objective function* will be described below.

This paper develops an algorithm to solve the Minimax continuous resource allocation problem. The previous research has provided algorithmic solutions to this problem by imposing strict constraints on the behavior of each f_i: each f_i is restricted to be either nondecreasing ([IbKa88]), or quasi convex ([Ze81] and [Ic84]). Furthermore, these algorithms only provide *one* optimal solution, not all of the valid optimal solutions, i.e., they only provide sufficient, but not necessary, characteristics for optimality. This paper lifts the restrictions of monotonicity and convexity. Furthermore, the procedures developed in this paper provide *all* the solutions to the Minimax problem. We shall assume that each f_i contains no constant interval (horizontal segment). Allowing a horizontal segment of a function f_i will not cause significant changes in the approach to the solution.

We now formally define this allocation problem:

Minimize $\quad F(x) = F(x_1, x_2, ..., x_n) = \max \{f_i(x_i)\}, \ i \in I = \{1, 2, ..., n\}$

subject to $\quad \Sigma x_i = N, \ x_i \in [l_i, u_i], \ l_i < u_i$

where each f_i is continuous over its domain D_{f_i}, and, for each D_{f_i}, there is no interval $[a_i, b_i]$, $a_i < b_i$, such that for all $x_i \in [a_i, b_i]$, $f_i(x_i) = c$, for some constant c. Each function f_i may be given as closed-form analytical expression of x_i or as a table of values listing f_i versus x_i.

2. Preliminaries

In this section, we will consider an example of the resource allocation problem with the objective function given above. We will examine one of the basic properties of F(x), provide definitions and notations, and generate a useful theorem for this resource allocation problem .

One property of F(x) is the connectedness of its range. [Ru91] states that if two functions f and g are continuous over their corresponding domains, then $\max\{f, g\} = \sup(f, g)$ is continuous, which implies that $R_{\max\{f,g\}}$ is connected, since the range of a continuous function is connected. Based on this, if we let $f^{[2]} = \max\{f_1, f_2\}$, we get that $f^{[2]}$ is continuous. If we then let $f^{[3]} = \max\{f^{[2]}, f_3\}$, again we conclude that $f^{[3]}$ is continuous. In general, if we recursively define $f^{[k]} = \max\{f^{[k-1]}, f_k\}$, where $k \in \{2, 3, ..., n\}$ and $f^{[1]} = f_1$, we conclude that $f^{[n]} = \max\{f^{[n-1]}, f_n\} = \max\{f_1, f_2, ..., f_n\} = F(x)$ is continuous, which further implies that $R_{f^{[n]}} = R_{F(x)}$ is connected.

We now introduce the definitions and notations used in this thesis. Since we will frequently manipulate the set of indices $I = \{1, 2, ..., n\}$ in the remainder of the paper, the following are provided for convenience:

Definition 1:

Define I^k, $k \in \{1, 2, ..., n\text{-}1\}$, *to be a proper subset of I with k elements, e.g., an example of* I^3 is $\{3, 5, n\}$.

Definition 2:

I_k, $k \in \{1, 2, ..., n\text{-}1\}$ *is defined to be the proper subset of I whose elements are* $\{1, 2, ..., k\}$.

Let us now consider an example using the figure below:

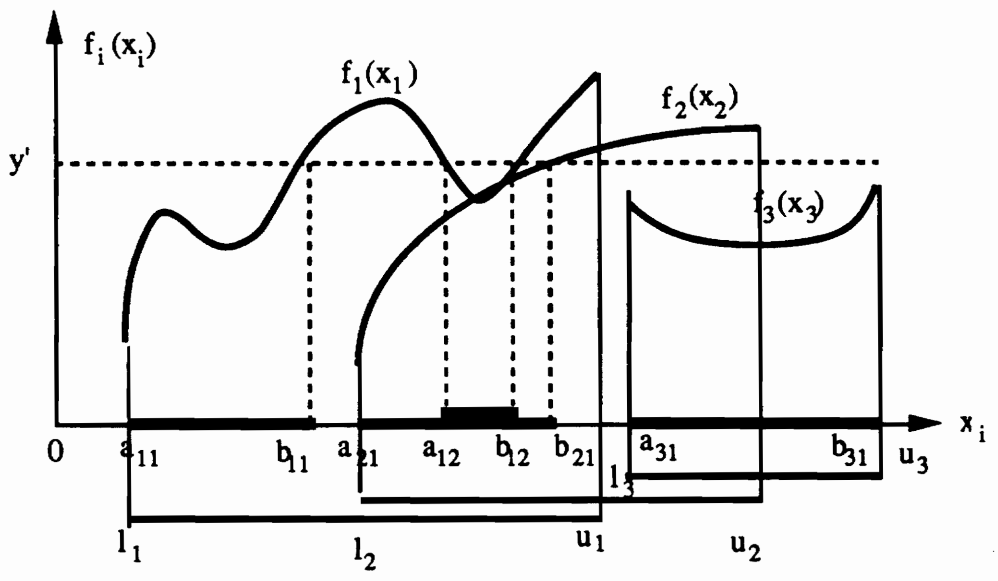

Figure 1. Relationship Among y', $\{f_i\}$, and F(x)

The horizontal axis in the above figure represents x_i, and the vertical axis, y, represents the corresponding $f_i(x_i)$. In this case, I = {1, 2, 3}. The horizontal line segments below the axis represent the domain of each f_i or the interval constraint on x_i: $[l_i, u_i]$. y' is an independent variable measured along the vertical axis. We will determine whether y' is attained by F(x) for some given N. Formally, we will determine if there exists some x' such that $\sum_{i\in I} x_i' = N$ and $\max\{ f_i(x_i')\} = y'$. We will examine the possible values of x'. Note that since $\max\{ f_i(x_i')\} = y'$, we have $f_i(x_i') \le y'$, and thus the only possible values of each x_i' are the points in $[l_i, u_i]$, in which $f_i(x_i') \le y'$. Clearly, if y' is not attained by any f_i, then y' is not attained by F(x), i.e., if y' is attained by F(x), then it must be in the union of the ranges of all f_i's, given in the following definition:

Definition 3:

Let $\mathbb{R} = \bigcup_{i \in I} f_i(x_i) = \bigcup_{i \in I} R_{f_i}, \forall x_i \in D_{f_i}$, *i.e., the union of all values that can be attained by each* f_i *but not exceeding* $\max\{F(x)\} = \max\{f_i(x_i)\}\ \forall x_i \in D_{f_i}$.

Using the above notation, the next definition provides a more convenient means to describe the possible values of each x_i,:

Definition 4:

Given $y' \in \mathbb{R}$ *such that* $y' \leq \max\{F(x)\}$, *let* $I_{ij}(y')$ *denote an interval* $[a_{ij}, b_{ij}] \subset D_{f_i}$, $a_{ij} \leq b_{ij}$, *such that* $\forall\ x_{ij} \in (a_{ij}, b_{ij})$ $f_i(x_{ij}) < y'$, *but* $f_i(a_{ij})$ *and* $f_i(b_{ij})$ can be *less than or equal to* y'. *Intuitively,* I_{ij} *is an interval of* f_i *whose range* $f_i(x_{ij})$ *does not exceed* y'. *Furthermore, each element in between* a_{ij} *and* b_{ij} *will have its functional value stricly less than* y'; *only a boundary is allowed to have its functional value to equal* y'. *We will denote the collection of such intervals corresponding to the index i as the set* $\{I_{ij}\}$. *For practical purposes, let* I_i *denote the array corresponding to the sorted elements of set* $\{I_{ij}\}$. *Unless otherwise indicated,* I_{ij} *is a shorthand notation of* $I_{ij}(y')$.

In the above figure, we have

$I_1 = [I_{11} = [a_{11}, b_{11}], I_{12} = [a_{12}, b_{12}]]$

$I_2 = [I_{21} = [a_{21}, b_{21}]]$

$I_3 = [I_{31} = [a_{31}, b_{31}]]$, (here a_{31} and b_{31} are equal to l_3 and u_3, respectively)

Therefore, in this case, the possible values of x_1', x_2', and x_3' are points taken from the intervals $\{[a_{11}, b_{11}], [a_{12}, b_{12}]\}$, $\{[a_{21}, b_{21}]\}$, and $\{[a_{31}, b_{31}]\}$, respectively. If we are able to find some combination of these points which sums to N, then it seems that we can conclude that y' is attained by F(x). Furthermore, instead of looking only at the combination of *single points* that will sum to N, we can observe the properties of a combination of *intervals* that can generate a combination of single points that will sum up to N, since an interval consists of an infinite number of points (unless the upper and lower limits are equal). The following definition provides a notion of a combination of intervals:

Definition 5:
Let $[I_{ij_i}]_{i=1}^{k}$ *denote a vector of intervals where* $I_{ij_i} \in \{I_{ij_i}\}$, *and* $i \in I_k$, i.e., $[I_{ij_i}]_{i=1}^{k}$ *is a possible combination of intervals taken from the indices* $\{1, 2, .., k\}$. *Unless otherwise indicated,* $[I_{ij_i}]$ *is a shorthand notation of* $[I_{ij_i}]_{i=1}^{n}$.

Note that the subscript j_i is used instead of a plain j to avoid any possible misinterpretation that the combination of the interval is taken by using only the j-th interval of the set of intervals I_i.

Observe that whenever the summation of the lower bounds of a combination of intervals does not exceed N and the sum of the upper bounds is not less than N, i.e., N is sandwiched between these two sums, then there is an infinite number of combinations of points whose sum is equal to N that can be taken from this combination of intervals. We can therefore conclude that once a combination of intervals with this property is found, then the constraint $\sum_{i \in I} x_i' = N$ is satisfied. However, we might not be able to conclude that y' is attained by F(x); since $F(x) = \max\{f_i(x_i)\}$, and thus y' has to be attained by at least one value $f_i(x_i')$. It is possible that we obtain some combination of x_i''s that will sum to N, but all of them actually result in $f_i(x_i')$ being strictly less than y'. The following theorem, however, assures that we can determine if a given y' (with the condition in Definition 4) is attained by F(x) only by checking if the property $\sum_{i \in I} x_i' = N$ is satisfied:

Theorem 1:

y' *is attained by* $F(x) = \max\{f_i(x_i')\}$ *if and only if there exists a vector of intervals* $[I_{ij_i}]$ *such that* $\sum_{i \in I} a_{ij_i} \leq N \leq \sum_{i \in I} b_{ij_i}$.

<u>Proof</u>:

<u>Sufficiency:</u>

Given a combination of intervals $[I_{ij_i}]$ with the above condition, the following provides a scheme to find some x' such that $\sum_{i \in I} x_i' = N$, where each x_i' is taken from each I_{ij_i} :

1 Let $x_i' = a_{ij_i} \quad \forall\, i \in I$

2. If $\sum_{i \in I} a_{ij_i} = N$, then we find $x' \in D_{F(x)}$. Otherwise, let m be the smallest element of $B \subseteq I$, where $B = \{\, k \in I : \sum_{i=1}^{k} b_{ij_i} + \sum_{i=k+1}^{n} a_{ij_i} \geq N \}$. Note that $B \neq \emptyset$ because $n \in B$. Now if we let $x_i' = b_{ij_i} \ \forall\, i \in \{1, 2, ..., k\text{-}1\}$, let $x_i' = a_{ij_i} \ \forall\, i \in \{k+1, k+2, ..., n\}$, and let $x_k' = N - \sum_{i \neq k} x_i'$, then we get that $\sum_{i \in I} x_i' = N$, which impliest that $x' \in D_{F(x)}$.

Now that we have concluded that $[I_{ij_i}]$ generates x', we need to show that y' is in fact in the range of F(x). Since $\forall\, i \in I$, I_{ij_i} consists of elements whose functional values are less than or equal to y', we know that $f_i(x_i') \leq y'$. If $\max\{f_i(x_i')\} = F(x') = y'$, then clearly y' is attained by F(x). If $\max\{f_i(x_i')\} = F(x') < y'$ then we know that a value smaller than y' is attained by F(x). Since y' $\leq$ max{F(x)} and the range of F is connected, this implies that y' must be attained by F(x).

Necessity:

First note that proving

y' is attained by F(x) implies that there exists a vector of intervals $[I_{ij_i}]$ with the above condition (*)

is equivalent to proving

If there does not exist any vector of intervals with the above condition, then y' is not attained by F(x) (**)

and thus instead of proving (*), we can prove (**) to show the necessity of the theorem.

If there does not exist any vector of intervals with the above condition, then each vector/combination of intervals will not hold the property $\sum_{i \in I} a_{ij_i} \leq N \leq \sum_{i \in I} b_{ij_i}$. Now, assume, to the contrary, that it is still possible to have y' attained by F(x), i.e., there exists some x' such that F(x') = y'. Since F(x') = y', we have that $\sum_{i \in I} x_i' = N$. Since y' = $\max\{f_i(x_i')\}$, $\forall i \in I$, $f_i(x_i') \leq y'$. This implies that $\forall i \in I$, x_i' must belong to some interval I_{ij_i} by the definition of I_{ij}, i.e., $\forall i \in I\ \exists\ a_{ij_i}, b_{ij_i}$ such that $a_{ij_i} \leq x_i' \leq b_{ij_i}$. Since $\forall i \in I$, $a_{ij_i} \leq x_i'$, and $\sum_{i \in I} x_i' = N$, we conclude that $\sum_{i \in I} a_{ij_i} \leq N$. Analogously, since $\forall i \neq k$, $b_{ij_i} \geq x_i'$, we conclude that $N \leq \sum_{i \in I} b_{ij_i}$. We are therefore able to find some combination of intervals $[I_{ij_i}]$ such that $\sum_{i \in I} a_{ij_i} \leq N \leq \sum_{i \in I} b_{ij_i}$, which is a contradiction, and thus y' can not be attained by F(x).

We have now found a scheme to determine whether a given y' is attained by F(x). If we repeatedly decrease this y' value (with some fixed step-size Δ) and determine, using the above theorem, whether or not the current y' is attained by F(x), we will be able to find the smallest y' among our sample values of y' that were attainable by F(x). The next step is to decide whether or not this y' is actually $y^* = \min\{F(x)\}$. Since the range of F(x) is connected, if we repeatedly decrease a value y' with some decrement Δ and determine if the current y' is attained by F(x), and stop the procedure only when we obtain a y' that was not attained by F(x), we can conclude that the previous y' value was actually $\min\{F(x)\} = y^*$. The value of y* is therefore the previous attainable y'. We can use this simple procedure of decrementing y' with some Δ repeatedly to find the correct y*. This is not recommended, however, since it will be computationally expensive. We need to use a step-size small enough to reduce any error tolerance ε, but a small step-size implies more trials to finally find some y' that is not attained by F(x). Alternatively, this paper uses the procedure of binary search on some part of the range of F(x) to find the value y*, which is described in the following chapter.

3. The Algorithm

This chapter presents the detailed procedure for solving the minimax resource allocation problem. The solution consists of determining the minimum value y* of F(x) as well as the value(s) x* of x for which this minimum is attained:

$$y^* = \min\{F(x)\} = F(x^*)$$

3.1 Overview of the Procedure

We first carry out a preliminary step to make the domain of each f_i as restricted as possible, based on the values of N and the lower and upper limits of the other domains. Once this is performed, we can precede to the main steps of the algorithm. The basic idea is to perform a binary search with the use of different values of y' . In the beginning, we will let y' be some (positive) number chosen in the manner described later in this paper. Once the first y' , denoted by FirstY', is selected, we trace each f_i from l_i until u_i. For each i, we keep track of all intervals I_i. The next step is to check if the condition mentioned in Theorem 1 holds for at least *one* combination of the intervals. If so, we conclude that this y' is attained by F. In this case, we replace y' by $\frac{\text{MaxMinF} + y'}{2}$, where MaxMinF is the smallest possible value that can be attained by F(x), the calculation of which will be explained later. Otherwise, we replace y' by $\frac{y' + \text{FirstY}'}{2}$. Starting from this point we repeatedly check if the new y' is attained by F and update y' in the following manner:

1 If y' is attained by F and y' > MaxMinF, let the new y' be $\frac{\text{MaxMinF} + y'}{2}$, and we update SmallestAttainY', the smallest attainable y' so far. We also keep track of $[\{I_{ij}(\text{SmallestAttainY'})\}]$

2 If y' is attained by F but y' = MaxMinF then we conclude that y* = MaxMinF has been found, let $[\{I_{ij}(x^*)\}] = [\{I_{ij}(y')\}]$, and exit the loop

3 If y' is not attained by F and the difference between the current y' and SmallestAttainY' is greater than some predefined error-tolerance ε, then we let the new y' be $\frac{y' + \text{SmallestAttainY'}}{2}$.

4 Otherwise, (if y' is not attained by F and |current y' - SmallestAttainY| $\leq \varepsilon$), then we conclude that y*, within the given error-tolerance ε, has been found (which equals SmallestAttainY') and exit the loop

After exiting the loop, i.e., after determining the value of y*, we precede to determine all possible x* such that F(x*) = y*. To do this, we need to find *all* combinations of $[x_i^*]$ that result in y*. This seems to be either an impossible or a very costly procedure, since each interval consists of an infinite number of points. However, as we will later see, each combination of $[x_i^*]$ that will result in the minimum solution will have certain characteristics, and thus the number of possible combinations that need to be checked can now be reduced by evaluating only those combinations that have these properties.

The following sections will describe each of the steps of the algorithm in greater detail.

3.2 Restricting the Domain of Each f_i

This is a preliminary step aimed at determining if the domain of each f_i can be restricted to an interval $[\lambda_i, \upsilon_i]$ which may be smaller than the given domain $[l_i, u_i]$, i.e.,

$[\lambda_i, \upsilon_i] \subseteq [l_i, u_i]$. This is useful in reducing, in addition to the work needed to trace each f_i, the number of intervals I_{ij} that need checking, because each domain has now possibly been shortened. Note that although lower and upper bounds are specified for each f_i, the actual possible values that can be assigned to each x_i may not actually be from the whole interval. A simple example is when $b_i > N$; then clearly we will not be able to assign to x_i any value greater than N. Let us consider the following example:

Let $N = 5$, $n=3$, and the domains of f_1, f_2, and f_3 be [1, 5], [2,4], and [1,6], respectively. Note that for any x_j, the largest value that it can attain is when each other x_i takes the smallest value on its interval, i.e., its lower bound. Analogously, the smallest value that x_j can attain is when each other x_i takes the largest value on its interval. In this case, then, the largest possible value that x_1 can attain is when $x_2 = 2$ and $x_3 = 1$. Since $N = 5$, and $\sum_{i \in I} x_i = N$, the largest possible value that can be assigned to x_1 is $5 - (2+1) = 3$, and thus we need to shorten [1,5] to [1,3]. Using a similar argument, we conclude that [2, 4] should be shortened to [2, 5-(1+1)=3], and [1, 6] should be shortened to [1, 5-(1+2)=2]. The next step is to check if it is possible to shorten each interval further, by increasing the lower bound of each x_i. The lowest value of x_i occurs when the other x_j's are the largest possible. Thus the smallest value that x_1 can attain is $5 - (3+2) = 0$. However, since $l_1 = 1$ for x_1, the smallest value that x_1 can attain remains 1. If we apply the same procedure to the other indices, we find that the lower bounds of the intervals are in fact the lowest possible value that each x_i can attain. From this example, we can generalize the smallest possible value, λ_i, and and the largest possible value, υ_i, of each x_i as $\lambda_i = \max\{l_i, N - \sum_{j \neq i} \upsilon_j\}$ and $\upsilon_i = \min\{u_i, N - \sum_{j \neq i} \lambda_j\}$. Note that the use of λ and υ in the summation symbol means that the values of λ and υ are not independent, one is calculated from the result of the other. If we first calculate υ_i, then each λ_i is in fact l_i, and vice versa. Reference [Ha89a] describes this conclusion, together with its rigorous proof, in more detail.

3.3 Choosing the First y'

The first y' chosen in implementing the algorithm is fairly significant. Note from the outline of the algorithm, that in order to determine whether or not y' is attained by F(x), we only need to find *one* combination of intervals $[I_{ij_i}]$ such that $\sum_{i \in I} a_{ij_i} \leq N \leq \sum_{i \in I} b_{ij_i}$. Once a combination is found, we conclude that y' is attained by F(x). However, to determine that y' is not attained by F(x), we may need to check all possible combinations of intervals to conclude that there does not exist any combination that will satisfy the above condition.

There are many ways to choose the first y'. The simplest way is probably to pick any value large enough to ensure that it is attained by F(x), but not too large so that it will not exceed max{F(x)}. The method proposed here is more costly, but it ensures that y' will be attained by F(x) in the first step, which implies that this first y', denoted as FirstY', will not be greater than max{F(x)}, which is a condition needed in Definition 4. Note that the following procedure assumes that $\sum_{i \in I} l_i \leq N \leq \sum_{i \in I} u_i$, because otherwise it means that the domain of F(x) is empty. The idea of the procedure is similar to the idea of finding the x' used in the proof of Theorem 1, except that it uses the largest element instead of the smallest element of B.

```
procedure GetFirstY
begin
   Let $x'_i$, $i \in \{1, ..., n\}$ be $l_i$.
```

```
If $\sum_{i\in I} x_i' = N$ then y' = max{$f_i(x_i')$}   [A y' is found]
    else if $\sum_{i\in I} x_i' < N \le \sum_{i\in I_{n-1}} x_i' + u_n$, then we know that the correct $x_n'$ is
    between [$l_n$, $u_n$], so we let $x_n'$ = $N - \sum_{i\in I_{n-1}} l_i$ and output y' =
    max{$f_i(x_i')$}
else
    Let $x_n' = u_n$
    Let j := n-1
    while $\sum_{i\ne j} x_i < N$ do
        If $\sum_{i\in I} x_i' = N$ then y' = max{$f_i(x_i')$}   [A y' is found]
        else if $\sum_{i\in I} x_i' < N \le \sum_{i\in I_{j-1}} x_i' + u_j + \sum_{i\in I-I_j-\{j\}} x_i'$, then we know that the
            correct $x_j'$ is in [$l_j$, $u_j$], so we let $x_j'$ = $N - \sum_{i\ne j} x_i'$ and
            output y' = max{$f_i(x_i')$}
        else
            $x_j' = u_j$
            j := j - 1
        end if
    end while
end if
end
```

3.4 Tracing the Functions {f_i} to Obtain [{I_{ij}}] - The First Step

The following procedure can be used to obtain the set of intervals for each index i such that $\forall x_{ij} \in I_{ij}, f_i(x_{ij}) \leq y'$. The variable MinF[i] and MaxF[i] correspond to the minimum and maximum values attained by each f_i respectively. They are used for the following reasons:

1 Once all [MinF[i]] and [MaxF[i]] are found, we will find MaxMinF = max{MinF[i]}. Let us assume that MaxMinF was attained by some $f_j(x_j)$. Since $\forall x_j \in D_{f_j}, \text{MinF}[j] \leq f_j(x_j)$, and since we need to take x_i from each index i in I to sum to N, any combination of x_i will not result in $F(x) = \max\{f_i(x_i)\} < f_j(x_j)$, which implies that F(x) can never be less than MaxMinF. This information is useful in performing the binary search on the minimum values of y: we avoid attempting to find a solution to y' < MaxMinF, i.e., we will use MaxMinF as the smallest bound in our binary search. To decrease the magnitude of error caused by the use of StepSize, once MaxMinF is found, for each index whose minimum value equals MaxMinF, we can use a smaller StepSize on its neighborhood (whose length is twice the original StepSize) and attempt to find the smallest value in each neighborhood. This will be our new MaxMinF.

2 Note that if we use the above procedure to obtain the first y', it is guaranteed that we will be able to find such y'. This y', which we later call FirstY', can now be used as the largest bound in our binary search. The next step in the binary search will be to set $y' = \frac{\text{MaxMinF} + y'}{2}$. We now need to find all $[\{ I_{ij_i} \}]$. The standard procedure is to trace each function. However, if for some index m, MaxF[m] < y', we know that the entire function is actually located below y', which implies that the set of intervals corresponding to this index consists of only one element, $[l_m, u_m]$. If this is the case, there is no need to trace the function.

The following procedure is to be executed for each index i. Note that a linked list is used to store the set of intervals corresponding to each index, since we do not know in advance how many intervals there are for each index. If static memory is not a problem, the use of arrays is advisable, as arrays are easier to access in the steps for determining whether y' is attained by F(x).

```
procedure InitTraceFunction
Inputs:  i, y'
Outputs: Linked-List-of-Intervals[i], NIntervals[i], MinF[i], MaxF[i]
Note:    Linked list is a doubly linked list
begin
   x' := l[i];
   Linked-List-of-Intervals[i] := nil;
   MinF[i] := f[i](x');
   MaxF[i] := f[i](x');
   while x' <= u[i] do
      while (tempf := f[i](x')) > y' and x' <= u[i] do
         x' := x' + StepSize;
            if tempf > MaxF[i] then MaxF[i] := tempf;
      end while
      if f[i](x') <= y' then
         Create a new interval with a := x';
      end if
      while (tempf := f[i](x')) <= y' and x' <= u[i] do
         x' := x' + StepSize;
         if tempf < MinF[i] then
            MinF[i] := tempf;
            MinX[i] = x'
         end if
      end while
      if f[i](x') > y' then
         Insert the new interval with b := x' at the end of list
   end while
end;
```

The next step is to execute the following procedure:

```
procedure GetMaxMinF
begin
   MaxMinF = MinF[1]
   for i = 2 to n do
      if MinF[i] < MaxMinF then
         MaxMinF := MinF[i]
         Remove list of previous MaxMinF indices, and create a new one
            consisting of i
      else if MinF[i] = MaxMinF then
         insert i into list of MaxMinF indices
      end if
   Decrease StepSize
   For each index in the list of MaxMinF do
```

```
    Retrace   f[i](x[i])   in   the   neighborhood   [MinX[i]-
       PrevStepSize,MinX[i]+PrevStepSize] and find new MinX[i]
  MaxMinF = min{MinX[i]} {This can be changed to max{MinX[i]} if it is
     deemed that the risks of missing y* is very small}
```

3.5 Tracing the Functions $\{f_i\}$ to Obtain $[\{I_{ij}\}]$ - The Latter Steps

The following procedure is used to obtain the set of intervals for each index i such that $\forall x_{ij} \in I_{ij}, f_i(x_{ij}) \le y'$. The first step for each index is always to check if the entire function is located below y' to possibly reduce the number of calculations. Otherwise, the standard step will be used. It is possible to trace the entire function in a similar way with the previous procedure, but this will be redundant. This procedure takes into consideration the following information:

1 Whenever it is determined that the previous y' was not attained by F(x), i.e., we need to increase y', the only portion of the curve that need to be evaluated will be those which were not included in the intervals corresponding to the previous y'. The procedure will therefore use each a_{ij} and evaluate only f_i to the left of a_{ij}, starting from a_{ij} itself, until at some point f_i exceeds y'. The new a_{ij} will be the last value of x_i whose evaluation did not exceed y'. We continue this tracing until $a_{ij} < b_{ij-1}$, adding any possible intervals between a_{ij} and b_{ij-1}. Similarly, the procedure will then evaluate f_i to the right of b_{ij}, until at some point f_i exceeds y'. Again, the new b_{ij} will be the last value of x_i whose evaluation did not exceed y'. This time, we do not need to continue tracing f_i until $b_{ij} < a_{ij+1}$, because any new intervals will have been found later, when we traced f_i to the left of the next a_{ij}. Note that in performing this procedure we may actually obtain some new $a_{ij} < b_{i,j-1}$, or a new $b_{ij} > a_{i,j+1}$, such that two intervals are now joined into one. It is therefore necessary to later determine if

we can join more than one interval by checking the existence of an overlapped portion of the function in consecutive intervals.

2. Similarly, when we need to decrease y', the only portions of the curve that need to be evaluated will be those which were included in the intervals corresponding to the previous y'. In this case the procedure evaluates only f_i to the right of a_{ij}, until at some point f_i is equal or less than y', and obtain a new a_{ij}. We continue this tracing until $a_{ij} > b_{ij}$, and find new intervals in between. The new b_{ij} will be obtained in an analogous manner. In this case, we check if for each j, $a_{ij} > b_{ij}$, because this implies that this interval does not actually exist, and should therefore be deleted in the final output. This case is simpler than the previous one, because here we simply need to check the same index j for each pair of a_{ij} and b_{ij}.

```
procedure TraceFunction
Inputs:  i, y', direction, NIntervals[i], Linked-List-of-Intervals[i]
Outputs: Linked-List-of-Intervals[i], NIntervals[i]
begin
   if MaxF[i] < y' then
      NIntervals[i] := 1;
      Linked-List-of-Intervals[i] := {[l[i],u[i]]}
   else
      if direction = 'u' then  {need to increase y' - going up}
         for k := 1 to j do {expand each interval}
            while a > l[i] and a >= Previous b and f[i](a) <= y' do
               a := a - StepSize;
            while a > l[i] and a >= Previous b do
            Perform a similar tracing procedure to the previous one,
            adding any new intervals between a and Previous b to the
            current list of intervals
         while b <= u[i] and b >= Next a and f[i](b) <= y' do
            b := b + StepSize;
      end for
      {Now join intervals, if possible}
      TempInterval := last(Linked-List-of-Intervals[i])
      while PrevInterval ≠ nil and PrevInterval^.b > a do
         a := PrevInterval^.a; {combine 2 intervals into 1}
         Remove PrevInterval from list;
         NIntervals[i] := NIntervals[i] - 1;
      end do
   else {need to decrease y'}
      TempInterval := first(Linked-List-of-Intervals[i])
      while TempInterval ≠ nil do
         while a < b and f[i](a) >= y' do {shrink each interval}
            a := a + StepSize;
         if a < b then {interval is not empty - yet}
            Find any possible new intervals in a similar procedure to
               the previous one for the portion of curve between a
               and b and insert new intervals to list of intervals
            while a < b and f[i](b) >= y' do
               b := b - StepSize;
            if a > b then
```

```
                  Remove TempInterval from Linked-List-of-Intervals[i]
                  and let the new TempInterval be the next interval
                  Decrease NIntervals[i]
               else
                  TempInterval := NextInterval
               end if
            else
               Remove TempInterval from Linked-List-of-Intervals[i] and
               let the new TempInterval be the next interval
               Decrease NIntervals[i]
            end if
         end do
      end if {direction}
   end if {MaxF[i]}
end
```

3.6 Determining If y' Is Attained by F(x')

We next check whether y' is actually attained by F(x). The simplest case is when at least one of the index has an empty $\{I_{ij}\}$. Otherwise, we continue with the standard procedure. This can be done by checking for the existence of a combination of intervals $[I_{ij_i}]$ that satisfies the property stated in Theorem 1. If none such combination exists, then we conclude that y' is not attained by F(x), otherwise, if we find *one* combination of intervals with this property, then we conclude that y' is attained by F(x). For convenience, the procedures in the remainder of the paper will use the following definition:

Definition 6:

Define $v_1 \mid v_2$ *to mean that two vectors* v_1, v_2 *are concatenated such that* v_1 *is followed by* v_2.

Procedure CheckYAttained is executed for each given y', after the sets of intervals $[\{ I_{ij_i} \}]$ have been found for all indices i in I. It first checks whether the summation of the $\{ a_{i1} \}$ of the first interval from each index exceeds N. If so, then clearly y' is not attained by F(x). Similarly, it also ensures that the summation of the $\{b_{i1}\}$ of the last interval of each index is not less than N, because otherwise y' is not attained by F(x). In the remainder of the paper, we will assume that the sets of intervals that need to be considered will have their redundant elements already eliminated, i.e., the intervals whose $\{a_{ij}\}$ are greater than N, which can be implemented with a simple procedure. Once this is done, it calls procedure GetNextComboInt (Get the Next Combination of Intervals) repeatedly until either a desirable combination of intervals is found or until there are no more possible intervals to be checked, which is indicated by the value of k = 0. Before executing this procedure, it initializes the current pointers j_i's of indices $\{1, 2, \ldots, n-1\}$ to 1, i.e., it indicates that the first combination of intervals that need to be checked is the combination of the first intervals of all indices $\{1, 2, \ldots, n-1\}$. The next step is to check if there exists an interval corresponding to index n that satisfies $\sum_{i \in I_{n-1}} a_{i1} + a_{nj_n} \leq N \leq \sum_{i \in I_{n-1}} b_{i1} + b_{nj_n}$, i.e., an interval that will make the current combination of $[I_{ij_i}]$ satisfy the condition in Theorem 1. This is performed in a binary search with the use of j_n: it first checks the middle interval of Intervals[n], and depending on the value of N and the current combination of $[I_{ij_i}]_{i=1}^{n-1}$, it will either check the intervals on the left or right of the middle interval. This is repeatedly done on the index n until a conclusion is found: whether or not there exists any interval on the n-th set of intervals corresponding to the index n that will satisfy the above property. For convenience, we will call the indices {1, 2, ..., n-1} as the fixed indices since in performing the binary search on the set of intervals corresponding to the index n, we only move the pointer j_n while keeping the other pointers j_i, i in {1, 2, ..., n-1} fixed. We then repeat the same procedure for every possible combination of $[I_{ij_i}]_{i=1}^{n-1}$. If we find some combination of $[I_{ij_i}]_{i=1}^{n-1}$ with some I_{nj_n} whose a_{nj_n} results in $\sum_{i\in I} a_{ij_i} > N$, then we know that the current I_{nj_n} cannot be used to satisfy the combination $[I_{ij_i}]_{i=1}^{n-2}|[I_{n-1j_{n-1}+1}]$, i.e., the same combination of I_{ij_i}'s from indices {1, 2, ..., n-2}, with the n-1th I_{ij_i}, $[I_{n-1j_{n-1}+1}]$, the next I_{ij_i} of $I_{n-1j_{n-1}}$, because it will surely result in a summation of a_{ij_i}'s greater than N. And this is true for each other a_{nk}, where $k > j_n$. This is why the array Range[i] is used: to indicate the last possible index of intervals of index i to be checked. The use of Range[i] is therefore to indicate that we need not check the intervals located on the right of Range[i] because it will not provide a solution. The next step is then to set Range[n] to be j_n-1, the index of the interval located immediately on the left side of I_{nj_n}, the interval whose a_{nj_n} causes the sum $\sum_{i\in I} a_{ij_i}$ to exceed N. Now, by repeatedly increasing the pointer to the n-1th interval, j_{n-1}, we will either (1) reach the last interval of the n-1th index, or (2) encounter some j_{n-1} that results in $\sum_{i\in I_{n-2}} a_{ij_i} + a_{n-1,j_{n-1}} + a_{n1} > N$, i.e., some interval $I_{n-1j_{n-1}}$ whose $a_{n-1,j_{n-1}}$ will cause the summation of a_{ij_i}'s of the current combination to exceed N regardless of which interval we choose from the set of intervals corresponding to index n. In this case, we will change Range[n-1] to j_{n-1}-1 because we know that any future combination of intervals from indices {1, 2, ..., n-2} will never need to use the intervals on the right of $I_{n-1,j_{n-1}-1}$, because that will surely result in $\sum_{i\in_n} a_{ij_i} > N$.

Since there are now no possible combinations that can be generated from the current $[I_{ij_i}]_{i=1}^{n-2}$, we will change the current combination of $[I_{ij_i}]_{i=1}^{n-2}$. This is performed by increasing j_{n-2} and letting $j_{n-1} = 1$. We then repeat the above procedure with the new $[I_{ij_i}]_{i=1}^{n-1}$, where the new $[I_{ij_i}]_{i=1}^{n-1}$ equals the previous $[I_{ij_i}]_{i=1}^{n-3} \| [I_{n-2,j_{n-2}+1}] \| [[I_{n-1,1}]$, i.e., the same combination of intervals from indices $\{1, 2, ..., n\text{-}3\}$, with the n-2th interval being the next interval of I_{n-2,j_n-2}, and the pointer to the last fixed interval, n-1, initialized back to 1, because we now need to check a different set of combinations of intervals. Now that j_{n-1} has been changed, we need to set the new Range[n], because the previous value of Range[n] is only valid for the last combination of intervals, in which j_{n-1} is located toward the end of the set of intervals corresponding to the index n-1. To avoid missing any possible desired combination, we will let Range[n] be the largest index k such that $\sum_{i \in I_{n-1}} a_{ij_i} + a_{nk} \le N$, i.e., the largest possible value of a_n that will not make the summation of the current combination exceed N. Similarly, we will also set Range[n-1] : it will be the greatest index k such that $\sum_{i \in I_{n-2}} a_{ij_i} + a_{n-1,k} + a_{n1} \le N$. Once these are set, we again perform a binary search on the nth set of intervals.

To generalize the above steps, if we previously let k=n-1, we next decrement k whenever j_k reaches Range[k] or whenever $\sum_{i \in I_k} a_{ij_i} + \sum_{i \in \bar{I}_k} a_{i1} > N$, i.e., whenever a_{kj_k} is the largest possible to sum to N (Note that this setting of new Range[i]'s is again based on the idea of attainable values of each x_i's ([Ha89a])). We then set j_i, where $i \in \{k+1, k+2, ..., n\text{-}1\}$ to 1 again, to start generating new combinations for the new $[a_{ij_i}]_{i=1}^{k}$. We also set Range[i] to be the largest k_i possible to sum to N for the current combination of $[a_{ij_i}]_{i=1}^{k}$, i.e., the largest k_i such that $\sum_{p \in I_k} a_{pj_p} + \sum_{m \in \bar{I}_k - \{i\}} a_{m1} + a_{ik_i} \le N$. We repeat this procedure of decreasing k and increasing j_k until either a desirable combination of intervals is found, in which case we conclude that the current y' is attained F(x); or until k=0, in which case the exhaustive procedure of finding all possible combinations is terminated and we conclude that the current y' is not attained by F(x).

```
procedure CheckYAttained
begin
    If any of the index has empty {$I_{ij}$} then
      YAttained := false
    return;
    YAttained := false
    If not ($\sum_{i \in I} a_{i1}$ > N or $\sum_{i \in I} b_{i,NIntervals[i]}$ < N) then
        k := n-1
        for i := 1 to n-1 do
            $j_i$ = 1
            Range[i] := NIntervals[i]  {in the beginning we assume that all
                intervals can be a part of a desirable combination}
        end do
        while (not YAttained) and (k > 0) do
            GetNextIntCombo
        If YAttained then
            Replace [{Iij}] corresponding to the previous SmallestYAttain
                with the new [{Iij}]
    end if
end
```

```
procedure GetNextIntCombo;

var CurrIndex, k: integer;

begin
   if $\sum_{i\in I_{n-1}} a_{ij_i} + a_{n1} \le N$ and $j_{n-1}$ <= Range[n-1] {there is at least one more
      interval to be checked on the set of intervals of index n-1} then
      Increase $j_{n-1}$ {check the next interval}
      Perform binary search on Intervals[n], from $n_1$ to $n_{Range[n]}$        (*)
      If found [I[n][$j_n$] such that $\sum_{i\in I} a_{ij_i} \le N \le \sum_{i\in I} b_{ij_i}$ then
         YAttained := true
         SmallestAttainY' := y'
         [I(SmallestAttainY')] := [I(y')]
      end if                                                             (**)
   else
      k := n-1
      repeat
         if $\sum_{i\in I_k} a_{ij_i} + \sum_{i\in \bar{I}_k} a_{i1} > N$ then {if the rest of intervals will not
            generate a solution, then we will not check them}
            Decrease k
      until k = 0 or $\sum_{i\in I_k} a_{ij_i} + \sum_{i\in \bar{I}_k} a_{i1} \le N$ or YAttained := true
      If k > 0 then
         Increase $j_k$
         for i := k+1 to n-1 do
            $j_i$ := 1
            Let Range[i] := the largest j such that $\sum_{p \in I_k} a_{pj_p} + \sum_{m \in \bar{I}_k - \{i\}} a_{m1} + b_j \le N$
        end for
      end if
      Repeat Steps (*) through (**)
   end if
end
```

3.7 The Final Step: Determining the Correct Combinations of x_i^*

By repeating the procedure CheckYAttained for different y' values chosen in the manner described in the steps of the algorithm, we will finally obtain the value of $y^* = \min\{F(x)\}$. We now need to identify all possible combinations of intervals that will satisfy Theorem 1 to find all the possible solutions x^* such that $y^* = F(x^*)$. We also need to identify the exact solutions, i.e., the combinations of single points (or single points and intervals). Since there are an infinite number of points in an interval, this objective of generating all combinations of the single points seems to be either impossible or very costly to perform. However, as the following theorems will show, the number of combinations of single points that need to be checked can be reduced by only checking the ones that satisfy certain properties that a minimum solution must have. The following definitions are needed in the subsequent theorems.

Definition 7:

If there exists some $c_i > 0$, *such that* $y_i' = f_i(x_i') < f_i(x_i) \; \forall \; x_i$ *in* $[x_i' - c_i, x_i' + c_i]$, *then* y_i' *is called a local minimum value of* f_i, *and* x_i' *is referred to as a local minimum point* *of* f_i. *Note that if* x_i *is a boundary point of the domain interval* $[l_i, u_i]$, *then* $f_i(x_i)$ *is either a minimum or a maximum on some interval* $[x_i, x_i + c_i)$ *if* x_i *is a lower bound, or on some interval* $[x_i - c_i, x_i]$ *if* x_i *is an upper bound. In such a case, when* x_i *is a minimum, we refer* x_i *and* $f_i(x_i)$ *as a* ***boundary minimum point*** *and a* ***boundary minimum value*** *respectively.* In the remainder of the paper, we shall assume, unless otherwise indicated, that whenever it is stated that f_i is increasing at x_i, then it is implied that f_i is increasing on $[x_i - c_i, x_i + c_i]$ for some positive c_i.

Definition 8:

A ***regular value*** $f_i(x_i)$ is any value $f_i(x_i)$ *that is neither a local minimum value nor a value evaluated at a boundary. Similarly, a* ***regular point*** x_i *is any* x_i *that is neither a local minimum point nor a boundary. Note that this includes a local maximum point* x_i. *We shall observe later that a local maximum point of* f_i *plays no significant role in finding a solution to* $y^* = \min\{F(x)\}$.

3.7.1 Solutions of the First Type: Regular Points

As stated in the following theorem, a solution x^* with all regular points will have equal functional values with all the functions being either all increasing or all decreasing on their corresponding points. We shall refer to such an x^* as a solution of the *First Type.*

Theorem 2:

If x^* *is a solution to* $y^* = \min\{F(x)\}$ *and each* x_i^* *is a regular point, then the following properties hold:*

1. f_i's *are either all increasing or all decreasing at* x_i^*'s
2. $\forall\, i, j \in I,\ f_i(x_i^*) = f_j(x_j^*)$

Proof of 1:

Let $[x_i^*]$ be a regular solution to $y^* = \min\{F(x)\}$, and suppose, to the contrary, that it is possible to have a situation in which there exist some two indices k, l such that f_k is increasing on $[x_k^*-c_k, x_k^*+c_k]$ and f_l is decreasing on $[x_l^*-c_l, x_l^*+c_l]$. We will obtain some x^{**} which results in $F(x^{**}) < F(x^*)$, i.e., a contradiction to the hypothesis that $F(x^*)$ is the minimum value attained by F. Note that $F(x^*) = \max\{f_i(x_i^*)\}$. For convenience, we rank the values of $f_i(x_i^*)$ in nonincreasing order $f_{i_1}(x_{i_1}^*)$, $f_{i_2}(x_{i_2}^*)$, ..., $f_{i_n}(x_{i_n}^*)$. There are two cases to consider:

1 $\max\{f_i(x_i^*)\} = f_{i_1}(x_{i_1}^*)$ is unique:

1.1 If f_{i_1} is decreasing on $[x_{i_1}^*-c_{i_1}, x_{i_1}^*+c_{i_1}]$ for some $c_{i_1} > 0$, then take some f_{i_m} that is increasing (or a local maximum) on $[x_{i_m}^*-c_{i_m}, x_{i_m}^*+c_{i_m}]$. Note that we will always be able to find this function because of the assumption that there exist some two indexes k, l such that f_k is increasing on $[x_k^*-c_k, x_k^*+c_k]$ and f_l is decreasing (or a local maximum) on $[x_l^*-c_l, x_l^*+c_l]$. We then let $x_{i_1}^{**} = x_{i_1}^* + d$ and let $x_{i_m}^{**} = x_{i_m}^* - d$, where d is a positive number satisfying $f_{i_1}(x_{i_1}^*+d) < f_{i_1}(x_{i_1}^*)$ and also $f_{i_m}(x_{i_m}^*-d) < f_{i_1}(x_{i_1}^*)$. Since $f_{i_1}(x_{i_1}^*)$ is the unique maximum value among $f_i(x_i^*)$, $f_{i_1}(x_{i_1}^*) > f_k(x_k^*)$ for all $k \neq i_1$, which implies that d will always be found because f_{i_1} is decreasing on $[x_{i_1}^*-c_{i_1}, x_{i_1}^*+c_{i_1}]$ and f_{i_m} is increasing (or a local maximum) on $[x_{i_m}^*-c_{i_m}, x_{i_m}^*+c_{i_m}]$ so there must be some positive number $x_{i_1}^{**}=x_{i_1}^*+d$ in $(x_{i_1}^*, x_{i_1}^*+c_{i_j}]$ and some $x_{i_m}^{**}=x_{i_m}^*-d$ in $[x_{i_m}^*-c_{i_m}, x_{i_m})$ such that $f_{i_1}(x_{i_1}^*) > f_{i_m}(x_{i_m}^{**})$. Since f_{i_1} is decreasing on $[x_{i_1}^*, x_{i_1}^*+c_{i_1}]$, we know that $f_{i_1}(x_{i_1}^{**}) < f_{i_1}(x_{i_1}^*)$. If we let all other $x_i^{**} = x_i^*$, then $F(x^{**}) = \max\{f_i(x_i^{**})\} = \max\{f_{i_1}(x_{i_1}^{**}), f_{i_2}(x_{i_2}^{**}), f_{i_m}(x_{i_m}^{**})\}$, all of which are less than $f_{i_1}(x_{i_1}^*)$. And thus we conclude that $F(x^{**}) < F(x^*)$, a contradiction to $F(x^*)$ being the minimum of F(x).

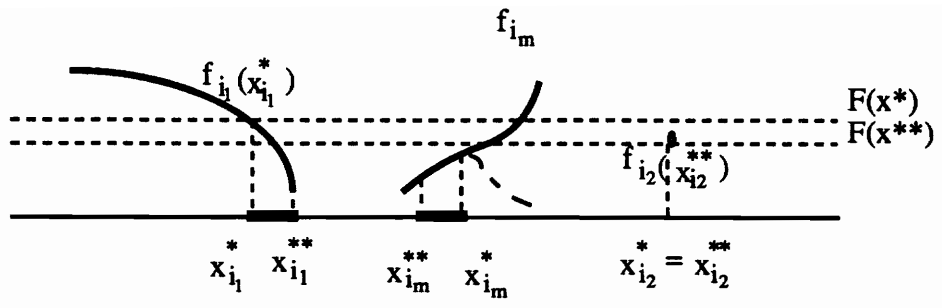
f_{i_m}
$f_{i_1}(x^*_{i_1})$
$F(x^*)$
$F(x^{**})$
$f_{i_2}(x^{**}_{i2})$
$x^*_{i_1}$ $x^{**}_{i_1}$
$x^{**}_{i_m}$ $x^*_{i_m}$
$x^*_{i_2} = x^{**}_{i_2}$

Figure 2. A Hypothetical Illustration of x* Consisting of Both Increasing and Decreasing {f_i} with Unique Maximum Regular Point

From the figure, we can observe that, if we replace the portion of the solid curve to the right of $x^*_{i_m}$ of the present f_{i_m} with the dashed curve to the right of $x^*_{i_m}$, then we can easily conclude that the case where $x^*_{i_m}$ is a local maximum does not result in any significant difference.

1.2 The proof for the case where $f_{i_1}(x^*_{i_1})$ is increasing is analogous.

1.3 From Figure 3 below, we can also observe that when $f_{i_1}(x^*_{i_1})$ is a local maximum, a contradiction is also reached, regardless of the behavior of $f_{i_m}(x^*_{i_m})$.

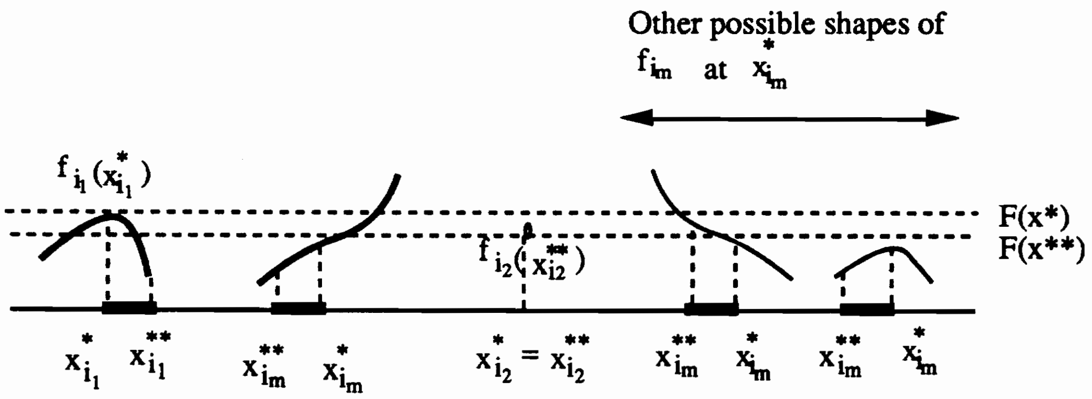
Other possible shapes of f_{i_m} at $x^*_{i_m}$
$f_{i_1}(x^*_{i_1})$
$F(x^*)$
$F(x^{**})$
$f_{i_2}(x^{**}_{i2})$
$x^*_{i_1}$ $x^{**}_{i_1}$
$x^{**}_{i_m}$ $x^*_{i_m}$
$x^*_{i_2} = x^{**}_{i_2}$
$x^{**}_{i_m}$ $x^*_{i_m}$
$x^{**}_{i_m}$ $x^*_{i_m}$

Figure 3. A Hypothetical Illustration of F(x*) with the Maximum Value Being A Local Maximum

2 If $\max\{f_i(x_i^*)\}$ is not unique, say, up to some index k, then there are two cases to consider:

2.1 Each of the maximum-attaining functions are either all increasing or decreasing on some interval $[x_{i_j}^*-c_{i_j}, x_{i_j}^*+c_{i_j}]$. Without loss of generality, we assume that they are all decreasing on this interval (the case where they are all increasing is analogous). Consider some function f_m that is increasing on $[x_m^*-c_m, x_m^*+c_m]$. Again, this function can always be found because of the assumption of the existence of two functions, one increasing and the other decreasing. Since $f_m(x_m^*)$ is less than all of these maximum values, there exist some positive numbers d, $d_{i_1}, d_{i_2}, \ldots, d_{i_k}$, such that the summation of $\sum_{j=1}^{k}(x_{i_j}^* + d_{i_j}) + x_m^* - d = \sum_{j=1}^{k} x_{i_j}^* + x_m^*$, and $f_{i_j}(x_{i_j}^*+d_{i_j}), j \in \{1,\ldots,k\}$, are all less than $f_{i_1}(x_{i_1}^*)$ and $f_{i_m}(x_{i_1}^* - d) < f_{i_1}(x_{i_1}^*)$. If we let $x_{i_j}^{**}$, $j \in \{1,\ldots,k\}$ be $x_{i_j}^* + d_{i_j}$, x_m^{**} be x_m^* - d, and all other x_i^{**} be x_i^*, then, using a similar argument to that of 1.1, we reach a contradiction since $F(x^{**}) = \max\{f_{i_1}(x_{i_1}^{**}), f_{i_2}(x_{i_2}^{**}), \ldots, f_{i_k}(x_{i_k}^{**}), f_{i_{k+1}}(x_{i_{k+1}}^{**}), f_{i_m}(x_{i_m}^{**})\}$, each of which is less than $f_{i_1}(x_{i_1}^*)$.

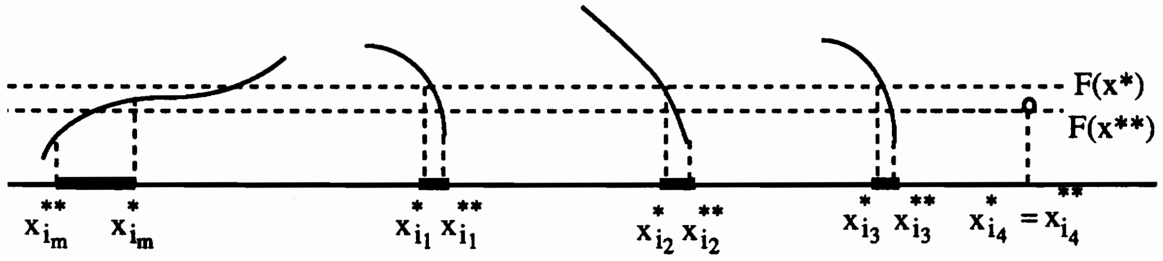
F(x*)
F(x**)
$x_{i_m}^{**}$
$x_{i_m}^{*}$
$x_{i_1}^{*}$ $x_{i_1}^{**}$
$x_{i_2}^{*}$ $x_{i_2}^{**}$
$x_{i_3}^{*}$ $x_{i_3}^{**}$
$x_{i_4}^{*} = x_{i_4}^{**}$

Figure 4. A Hypothetical Illustration of F(x*) Consisting of Both Increasing and Decreasing Values with All Decreasing Maximum Values

2.2 There are m and k-m maximum values $\{f_{i_j}(x_{i_j}^*)\}$ that are decreasing and increasing on $[x_{i_j}^*-c_{i_j}, x_{i_j}^*+c_{i_j}]$, respectively. Using a similar argument to that in 2.1, we come to the conclusion that there exist some positive numbers d_{i_1}, $d_{i_2}, \ldots, d_{i_m}, d_{i_{m+1}}, \ldots, d_{i_k}$, such that $f_{i_l}(x^*_{i_l} + d_{i_l})$, $l \in \{1, \ldots, m\}$, are all less than $f_{i_l}(x^*_{i_l})$, and also $f_{i_l}(x^*_{i_l} - d_{i_l})$, $l \in \{m+1, \ldots, k\}$ are all less than $f_{i_l}(x^*_{i_l})$, and $\sum_{l=1}^{m}(x^*_{i_l} + d_{i_l}) + \sum_{l=m+1}^{k}(x^*_{i_l} - d_{i_l}) = \sum_{l=1}^{m} x^*_{i_l} + \sum_{l=m+1}^{k} x^*_{i_l}$. Now, applying a similar argument to that of 1.1 by letting $x^{**}_{i_l} = x^*_{i_l} + d_{i_l}$, $l \in \{1, \ldots, m\}$, and letting $x^{**}_{i_l} = x^*_{i_l} - d_{i_l}$, $l \in \{m+1, \ldots, k\}$, and letting all other $x^{**}_i = x^*_i$, we again reach a contradiction.

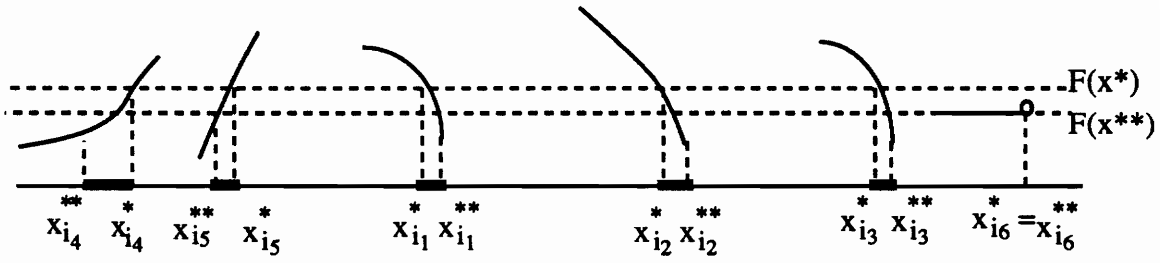
F(x*)
F(x**)
$x^{**}_{i_4}$ $x^*_{i_4}$ $x^{**}_{i_5}$ $x^*_{i_5}$ $x^*_{i_1}$ $x^{**}_{i_1}$ $x^*_{i_2}$ $x^{**}_{i_2}$ $x^*_{i_3}$ $x^{**}_{i_3}$ $x^*_{i_6} = x^{**}_{i_6}$

Figure 5. A Hypothetical Illustration of F(x*) Consisting of Both Increasing and Decreasing Values with Both Increasing and Decreasing Maximum Values

Therefore, if F(x*) is the minimum value attained by F, then either all f_i's increase on $[x^*_i - c_i, x^*_i + c_i]$ or all $\{f_i\}$ decrease on $[x^*_i - c_i, x^*_i + c_i]$.

Proof of 2:

Assume, to the contrary, that it is possible to have a condition in which x* is a non-boundary solution to y* = min F(x) and there exist m maximum-attaining f_i's, and n minimum-attaining functions f_i's. By the first property of a minimum solution, all f_i's are either all increasing or all decreasing on $[x^*_i - c_i, x^*_i + c_i]$. Without loss of generality, we assume that all of the functions are decreasing (the case where they are increasing on the intervals are similar). Again, using a similar scheme to the Proof of 1, Case 2.1, we come to the conclusion that there exist some positive numbers d_l, where l are the indices of the maximum values, and, d_k, where k are the indices of the minimum values, such that $f_l(x_l^* + d_l) < f_l(x_l^*)$ and $f_k(x_k^* - d_k) < f_l(x_l^*)$, and the summation of $\sum(x_l^* + d_l) + \sum(x_k^* - d_k) = \sum x_l^* + \sum x_k^*$. Now, applying a similar argument to that of 1.1 by letting $x_l^{**} = x_l^* + d_l$, and letting $x_k^{**} = x_k^* - d_k$, and letting all other $x_i^{**} = x_i^*$, we again reach a contradiction. Therefore, all $f_i(x_i^*)$'s must be equal.

3.7.2 Solutions of the Second Type

As shown in the following theorem, a solution x* with some boundaries and regular points will have equal values of regular $\{f_i(x_i^*)\} \geq$ boundaries $\{f_{j(x_j^*)}\}$, and consists of either all upper bounds and increasing regular $\{f_i(x_i^*)\}$, or all lower bounds and decreasing regular $\{f_i(x_i^*)\}$.

Theorem 3:

Let x* *be a solution to* $y^* = \min\{F(x)\}$ *with some boundaries* $\{x_j^*\}$ *such that* $\{x_j^*\}$ *are not boundary minimum points and* $f_j(x_j^*)$'s *do not exceed* $\max\{f_i(x_i^*)\} = f_k(x_k^*)$ *for some regular* x_k^*. *Then the following properties hold:*

1 *For all regular points* $x_k^*, x_l^*, f_l(x_l^*) = f_k(x_k^*)$, *and are either all increasing or all decreasing.*

2 2.1 *If all regular values* $\{f_k(x_k^*)\}$ *are increasing then for each boundary* $x_j^*, x_j^* = u_j$

  2.2 *If all regular values* $\{f_k(x_k^*)\}$ *are decreasing then for each boundary* $x_j^*, x_j^* = l_j$

<u>Proof:</u>

There are two cases to consider:

1 None of the maximum values are boundary values, i.e., for all boundaries $x_j^*, f_j(x_j^*) < f_k(x_k^*)$. This implies that $\max\{f_i(x_i^*)\}$ is attained by regular $\{f_k(x_k^*)\}$. Now, applying a similar argument to the proof of the previous theorem to the regular $\{f_k(x_k^*)\}$, we will obtain some $F(x^{**}) < F(x^*)$ unless all maximum regular values are equal to each other and are either all increasing or all decreasing on x_k^*, because we will always be able to obtain some regular values $\{f_k(x_k^{**})\}$ that are less than $\{f_k(x_k^*)\}$. Thus Part (1) of the theorem is proved. To prove Part 2 of the theorem, we shall analyze the case where all regular functions are increasing (the other case is similar). Assume, to the contrary, that it is possible to have a lower bound $x_j^* = l_j$, $f_j(x_j^*) = f_j(u_j) < f_k(x_k^*)$ where $f_k(x_k^*)$ is increasing. If for each boundary x_j^* we take some $x_j^{**} > x_j^*$ and for each regular (increasing) x_k^* we take some $x_k^{**} < x_k^*$ such that $f_j(x_j^{**}) < f_k(x_k^*)$ and $f_k(x_k^{**}) < f_k(x_k^*)$ and $x_k^{**} + x_j^{**} = x_k^* + x_j^*$, we will obtain some $F(x^{**}) < F(x^*)$ if we let each other x_i^{**} equal x_i^*, and thus if all regular $\{f_k(x_k^*)\}$ are increasing, then all boundaries $\{x_j^*\}$ must be upper bounds. Using a similar argument to the case where all regular values are decreasing, we conclude that all boundaries must be lower bounds. Therefore, this type of solution consists of all regular values being equal to each other and either all regular values are increasing combined with upper bounds, or all regular values are decreasing combined with lower bounds.

2 There exists at least one boundary x_j^* such that $f_j(x_j^*) = f_k(x_k^*)$. There are 2 cases to consider:

2.1 $\{x_j^*\}$ are either all upper bounds or all lower bounds:

Assume, to the contrary, that it is possible to have regular x_k^*, x_l^* such that f_k is decreasing on $[x_k^*\text{-}c_k, x_k^*\text{+}c_k]$ and f_l is increasing on $[x_l^*\text{-}c_l, x_l^*\text{+}c_l]$. Since the case in which all x_j^*'s are lower bounds are analogous, here we will analyze the case when all x_j^*'s are upper bounds. Now, if we take some $x_j^{**} < x_j^*$, $x_l^{**} <$

x_l^*, and $x_k^{**} > x_k^*$ such that $x_j^* + x_l^* + x_k^* = x_j^{**} + x_l^{**} + x_k^{**}$, then we will obtain $\max\{f_i(x_i^{**})\} < \max\{f_i(x_i^*)\} = \min F(x)$, a contradiction. Thus all non-boundary $f_i(x_i^*)$ must be either all increasing or all decreasing on $[x_i^*-c_i, x_i^*+c_i]$. We now prove that all regular values must be equal to each other. Assume, to the contrary, that it is not the case that all regular functions are equal to each other, i.e., there exists some regular index m such that $f_m(x_m^*) < f_k(x_k^*)$. Based on the previous proof, we know that all regular values must be either all increasing or all decreasing. We examine the case when they are all increasing. This implies that there exist some regular $x_m^{**} > x_m^*$ and some $x_k^{**} < x_k^*$ such that $f_m(x_m^{**}) > f_m(x_m^*)$ and $f_k(x_k^{**})<f_k(x_k^*)$. Since $f_m(x_m^*) < f_k(x_k^*)$, we will be able to find this x_m^{**} and x_k^{**} such that $f_m(x_m^{**}) < f_k(x_k^{**})$ and $f_k(x_k^{**}) < f_k(x_k^*) = F(x^*)$. If we let each other $x_i^{**} = x_i^*$, then we will obtain some $F(x^{**}) = \max\{f_m(x_m^{**}), f_k(x_k^{**})$, some value $f_j(x_j^{**}) = f_j(x_j^*) < f_k(x_k^*)\} < F(x^*)$, and thus we reach a contradiction to the assumption that not all regular values are equal to each other. We now prove that if all regular values are increasing, then all boundaries must be upper bounds. So far, we know that all regular values $f_k(x_k^*)$ are all increasing and are all equal to each other, and for each boundary x_b^*, $f_b(x_b^*) \leq f_k(x_k^*)$. Assume, to the contrary, that it is possible to have $x_b^* = l_b$. If $f_b(x_b^*) = f_k(x_k^*)$, then $f_b(x_b^*)$ is a local minimum boundary value, which is later discussed in Type 3 Solution. Otherwise, we will be able to take some $x_b^{**} > x_b^*$ and some $x_k^{**} < x_k^*$ such that $f_k(x_k^{**}) < f_k(x_k^*)$ and $f_b(x_b^{**}) < f_k(x_k^*)$. If we let each other $x_i^{**} = x_i^*$, we will obtain $F(x^{**}) = \max\ \{f_k(x_k^{**}), f_b(x_b^{**})$, some boundary value $f_{b_1}(x_{b_1}^*) < f_k(x_k^*)\} < F(x^*)$, another contradiction. If we analyze the case where all regular values are decreasing in a similar manner, we will reach to the conclusion that all regular values must be equal to each other and if all regular values are decreasing, then all boundaries must be lower bounds. We can conclude that all regular values must be equal to each other and either we have a combination of all regular increasing values with upper bounds, or all regular decreasing values with lower bounds.

2.2 There exist both upper bounds and lower bounds, call them x_u^* and x_l^* : We will immediately reach a contradiction by taking some $x_u^{**} < x_u^*$, and $x_l^{**} > x_l^*$. This will result in a condition in which the maximum of the boundaries values are less than $f_k(x_k^*)$. Since we have both upper and lower bounds, we can increase or decrease any regular values appropriately by increasing or decreasing x_k^* and compensate the difference by either reducing x_u^* or increasing x_l^* , to finally obtain $F(x^{**}) < F(x^*)$. Therefore, if x^* is a solution to $y^* = \min \{F(x)\}$, then we cannot have both upper and lower bounds occur simultaneously. Assume, without loss of generality, that all boundaries are upper bounds. We need to show that in this case, all regular functions must be increasing. Suppose it is possible to have some regular index k such that $f_k(x_k^*)$ is decreasing. We will then be able to find some $x_k^{**} > x_k^*$ and some boundary $x_b^{**} < x_b^*$ such that $f_b(x_b^{**}) < f_b(x_b^*)$ and $f_k(x_k^{**}) < f_b(x_b^*)$ since $f_k(x_k^*) \leq f_b(x_b^*)$. Again, we will obtain some $F(x^{**}) < F(x^*)$, another contradiction, and thus all regular values must be increasing if all boundaries are upper bounds. We now need to show that all regular values must be equal to each other. Assume, to the contrary, that it is not the case that all regular values are equal to each other, i.e., there exists some regular index m such that $f_m(x_m^*) < f_k(x_k^*) \leq f_b(x_b^*)$. Then there exists some $x_m^{**} > x_m^*$ and some $x_k^{**} < x_k^*$ and some $x_b^{**} < x_b^*$ such that $f_m(x_m^{**}) < f_b(x_b^*)$, $f_k(x_k^{**}) < f_b(x_b^*)$, and $f_b(x_b^{**}) < f_b(x_b^*)$ and $x_m^* + x_k^* + x_b^* = x_m^{**} + x_k^{**} + x_b^{**}$. Again, if we let each other $x_m^{**} = x_i^*$, we will obtain some $F(x^{**}) = \max\{f_m(x_m^{**}), f_k(x_k^{**}), f_b(x_b^{**})$, some boundary value

$f_{b_1}(x^{**}_{b_1}) = f_{b_1}(x^*_{b_1}) < f_b(x^*_b))\} < F(x^*)$, which is another contradiction. Thus if all boundaries are upper bounds then all regular values must be increasing and equal to each other. Applying the same argument to the case in which all boundaries are lower bounds, we conclude that if all boundaries are lower bounds then all regular functions must be decreasing and equal to each other. And since we have proved that we cannot have a combination of x* that includes both lower and upper bounds, we can further conclude that all regular values of x* must be equal to each other and are either all increasing or all decreasing. If they are all increasing, then the boundaries must be all upper bounds, otherwise, the boundaries must be all lower bounds.

3.7.3 Solutions of the Third Type

As shown by the following theorem, solutions of the Third Type are characterized by F(x) being attained by at least one local (boundary) minimum value of some f_j

Theorem 4:

Let x^* *be a solution to* $y^* = \min\{F(x)\}$. *If* $\max\{f_i(x^*_i)\}$ *is attained by at least one local minimum value* $f_m(x^*_m)$ *or one boundary minimum value* $f_b(x^*_b)$, *then if all of the following conditions hold:*

1 *For all local minimum points* $x^*_{m_1}$, $x^*_{m_2}$, $f_{m_1}(x^*_{m_1}) = f_{m_2}(x^*_{m_2})$, *and*

2 *For all regular* x^*_i, x^*_j, $f_i(x^*_i) = f_j(x^*_j) = f_m(x^*_m)$, *and*

3 *Either:*

3.1 *All regular* f_i*'s are increasing on* x^*_i *and if for all boundaries* x^*_b *such that* $f_b(x^*_b) \leq F(x^*)$, $x^*_b = u_b$

or

3.2 *All regular* f_i's *are all decreasing on* x_i^* *and if for all boundaries* x_b^* *such that* $f_b(x_b^*) \leq F(x^*)$, $x_b^* = l_b$,

then such x^* *is unique in its neighborhood, otherwise, there are an infinite number of such* x^*, *unless there exists exactly one other* $q \neq m$ *such that* $f_q(x_q^*) < f_m(x_m^*)$.

<u>Proof:</u>

We will first show that when all of the above conditions hold, x* is unique in the neighborhood. We will show that we cannot find any x** in the neighborhood of x* such that $\sum_{i\in I} x_i^{**} = N$ and $\max\{f_i(x_i^{**})\} = F(x^{**}) = F(x^*)$. Since $\sum_{i\in I} x_i^{**}$ has to equal N = $\sum_{i\in I} x_i^*$, we need to find at least two indices p, q such that $x_p^* > x_p^{**}$ and $x_q^* < x_q^{**}$ to maintain the sum of N. We consider three cases:

1 Either p or q is equal to any possible local minimum index m: then $f_p(x_p^{**}) > f_p(x_p^*) = f_m(x_m^*)$ because $f_m(x_m^*)$ is a local minimum. Since $f_m(x_m^*) = F(x^*) < f_m(x_m^{**}) \leq F(x^{**})$, x** cannot be a minimum solution if p or q is chosen from the possible local minimum indices.

2 p is a regular index: To obtain $f_p(x_p^{**}) < f_p(x_p^*)$, we need to have f_p increase on x_p^*, so we need to choose the situation in which the regular functions are all increasing. We will need to find an appropriate index q such that $x_q^{**} > x_q^*$ to obtain the sum N. Since all the regular functions are increasing and are equal to themselves, any index q obtained from a regular index will result in $f_q(x_q^{**}) > f_q(x_q^*) = F(x^*)$, and thus we can only choose q from the boundary indices. However, since all the boundaries are upper bounds, we will never be able to find such x_q^{**}.

3 p is a boundary in which $f_p(x_p^*) \leq F(x^*)$: Since $x_p^{**} < x_p^*$, this has to be an upper bound, which further implies that all of the other boundaries x_b^* that satisfy $f_b(x_b^*) \leq F(x^*)$ are also upper bounds. We can therefore only choose q from the regular

intervals, which are all increasing, according to the conditions given. Since $f_q(x_q^*) = F(x^*)$, $x_q^{**} > x_q^*$, and f_q is increasing on x_q^*, $f_q(x_q^{**}) > f_q(x_q^*) = F(x^*)$. Again we are not able to find such x^{**}.

Therefore, such x^{**} does not exist, i.e., x^* is unique in its neighborhood.

To show that there will be an infinite number of x^* otherwise, we observe from the figure below that when any one (or more) of the conditions above does not hold, the number of such x^* is infinite regardless of whether or not the other conditions are satisfied, unless there is exactly one index $q \neq m$ such that $f_q(x_q^*) < f_m(x_m^*)$. Since the proofs will be very similar to one another, only one case is discussed. We can observe all the situations that may occur if the first property is not held from Figure 6. The way to observe the figure is by including the curves corresponding to indices m_1 and m_2, and finding all possible combinations of any types of curves for the other indices that will constitute x^*.

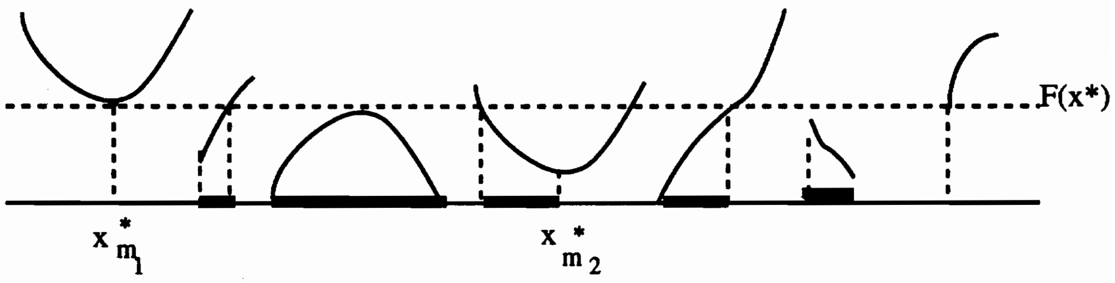

Figure 6. A Hypothetical Illustration of F(x*) Attained by At Least One Local Minimum with the Local Minima Not Equal to One Another

Note that the exceptional case in which the theorem will fail is when there is only one other index $q \neq m$ such that $f_q(x_q^*) < f_m(x_m^*)$. This is so because we need to maintain the condition $\sum_{i\in I} x_i^{**} = N$. When there is only one such index, we would not be able to find a combination of intervals (each of which consists of an infinite number of points) among the non minimum index to still satisfy $\sum_{i\in I} x_i^{**} = N$ and $\max\{f_i(x_i^{**})\} = f_m(x_m^*)$. Otherwise, we will have more than one interval whose functional values of the elements are less than or equal to $f_m(x_m^*)$. We will then be able to find an infinite number of combinations of x_i^*'s that result in $\sum_{i\in I} x_i^*$ taken from these intervals and still obtain $y^* = \max\{f_i(x_i^{**})\} = f_m(x_m^*)$.

Now that we have identified the conditions that a minimum solution can have from the the three types of possible solutions above, we shall next show that these three are the only possible types of a minimum solution. Since $F(x) = \max\{f_i(x_i)\}=f_k(x_k)$ for some k, k may not be unique, we will consider all types $f_k(x_k^*)$ for a given minimum solution x^*:

1 x_k^* is a local minimum: then we have the condition assumed by Theorem 4, and thus this type of solution is included in Type 3 Solution.

2 Any of the x_k^*'s is a boundary: there are three cases to consider:

2.1 If x_k^* is a minimum boundary, then we again have the condition assumed by Theorem 4, and thus this type solution is included in Type 3 Solution

2.2 Otherwise, if there exists any regular x_j^* such that $f_j(x_j^*) = F(x^*)$, then we have the condition assumed by Theorem 3, and thus this type of solution is included in Type 2 Solution.

2.3 Otherwise, if $F(x^*)$ is attained only by boundary values, then if the maximum attaining boundary values consist of both lower and upper bounds, then, using a similar proof to the one used in Theorem 3 part 2.2, we will reach a contradiction to the assumption that x^* is a minimum solution, and thus the boundary values must be either all lower bounds or all upper bounds. We now assume that all boundaries are upper bounds. Since F(x*) is attained by a boundary only, call this x_b^*, we need to check the possible types of functions whose values are less than F(x*). If any of the smaller values of $f_i(x_i^*)$ is regular, then, applying the proof used in Theorem 3, part 2, we know that we cannot have a combination of x_i^* that consists of both upper bounds and decreasing functions, and thus any regular functions in this case will be increasing. We will reach a contradiction by taking some $x_b^{**} < x_b^*$ and some regular $x_k^{**} < x_k^*$ which results in some $F(x^{**}) < F(x^*) = f_b(x_b^*)$, and thus we cannot have the case in which F(x*) is attained by a boundary only when there are regular values. The only exception is when each $x_i^* = u_i$ or each $x_i^* = l_i$. But in this case, this combination is the possible combination that will sum up to N.

3 x_k^*'s are all regular: then we have the condition assumed in Theorems 2 and 3, and thus this type of solution has also been covered.

4 Any of the x_k^*'s is a local maximum: This case has been discussed briefly in Type 3 Solution; a contradiction results whenever F(x*) is attained by any local maximum because we can always increase (or decrease) x_k^* appropriately to decrease $f_k(x_k^*)$ and thus obtain some other $F(x^{**}) < F(x^*)$.

Since we have looked at all possible types of $\max\{f_i(x_i^*)\}$, and in each case it is concluded that the type of solutions have been covered, the three types of solutions are the only types of solutions for x*. The exception is the case in which the only combination of x_i that can result in N is either all upper bounds or all lower bounds, in which the domain of F(x) consists of only one element. This is implicitly covered by

Type 2 Solution, i.e., when there are no regular values, then all the values will be either upper boundary values or lower boundary values.

We can now eliminate the number of possible combinations that need to be checked; we will only consider those that satisfy the above properties. We can now proceed to the next step of the algorithm.

3.8 Classifying $[\{I_{ij}(y^*)\}]$

From the above types of minimum solutions, we observe that a possible solution can be either a combination of points or a combination of both points and intervals. We will therefore need to categorize each interval corresponding to each index: whether it represents a single point, i.e., its length is less than ε, or it is a regular interval. This is done simply by checking, for every set of intervals corresponding to each index i, which of its elements have the lengths less than ε. The results are stored in two arrays for each index: one for single-point intervals and the other for regular intervals. The use of arrays is to increase efficiency, which now becomes a significant issue since we need to find (or access) *all* combinations of solutions. Due to its simplicity, the exact procedure is omitted. We continue to the next step of the algorithm by observing the characteristics of a minimum solution stated in the previous theorems.

3.9 Analyzing and Generating Each Type of Solution

The following sub-sections outlines the approach to generate all the solutions from the combinations of intervals.

3.9.1 Type 1 Solutions - Regular Points

Since Type 1 involves only regular $f_i(x_i^*)$'s, the only possible intervals that will contain a solution x^* will be all I_{ij} such that $|I_{ij}| \geq \varepsilon$, i.e., intervals that theoretically contain more than one point, because a point implies that it is a local minimum. In addition, since all $f_i(x_i^*)$'s are equal to each other and since none of the f_i's has any constant (horizontal) segment, the only possible x_i^*'s along the intervals will be the limits of the intervals, since these limits are the only x_i^*'s that can result in $f_i(x_i^*) = y^*$ (recall that all the points in between are strictly less than y^*). Furthermore, since $f_i(x_i^*)$'s have to be either all increasing or all decreasing, x^* in this situation can only be a combination of either all a_{ij_i}'s or all b_{ij_i}'s. Since the procedure of checking which points will sum to N involves some predefined error-tolerance ε, we will introduce the following definition:

Definition 9:

For any two given expressions e_1, e_2, *define* $e_1 \overset{\varepsilon}{=} e_2$ *to mean* e_1 and e_2 *are equal within some error tolerance* ε, *or, more precisely*, $|e_1 - e_2| < \varepsilon$

And thus, in our procedure we need only find combinations of all a_{ij_i}'s such that $\sum\limits_{i \in I} a_{ij_i} \overset{\varepsilon}{=} N$ and all b_{ij_i}'s such that $\sum\limits_{i \in I} b_{ij_i} \overset{\varepsilon}{=} N$. Note that once a combination of the lower limits of some intervals has been determined to result in N, we can no longer use the same combination of intervals. It is feasible to keep track of all the combination of a_{ij_i}'s that result in N so that in the next step of the procedure, we will know that the b_{ij_i}'s taken from the same combination of intervals will not result in N. However, if there are many possible solutions, than this idea may actually be more computationally expensive, because now, for every combination of b_{ij_i}'s, we need to check if its corresponding a_{ij_i}'s belong to this set of known solutions, and the checking implies that we need to observe the elements in the set of known solutions. The procedure used in this thesis does not keep track of the known solutions with the justification that the overall number of possible combinations of b_{ij_i}'s that actually have the corresponding a_{ij_i}'s belong to the set of known solutions will be small, and thus a simple evalution for every combination of points may actually have a better performance.

3.9.2 Type 2 Solutions - Regular and Boundary Points

The possible combinations of this type are somewhat similar to the ones of Type 2. Since the characteristics of this type of solutions are that the values of all regular $f_i(x_i^*)$'s are equal to each other and the functions corresponding to these values are either all increasing or all decreasing, we can again use a similar scheme to (2) to generate the possible combinations of the regular x_i^*'s. The next step is to consider the boundaries. Since a solution x^* of this type can only consist of either all increasing regular x_i^*'s and upper bounds, or all decreasing regular x_i^*'s and lower bounds, we know that the only possible combinations that need checking are all b_{ij_i}'s with upper bounds, and all a_{ij_i}'s with lower bounds, that will sum to N. Furthermore, since all upper bounds are also classified as b_{ij_i}'s, and all lower bounds are classified as a_{ij_i}'s, the final procedure to obtain combinations of this type turns out to be exactly the same as the procedure used for Type 2 Solutions.

3.9.3 Type 3 Solutions - Regular, Boundary, Local Minimum Points, and Intervals

For convenience, we will consider all the cases of solutions of Type 3 separately, as they can result in several combinations of different types of points and intervals. For a given combination of intervals, we can obtain either a finite or an infinite number of solutions. The following gives all the types of possible combinations:

1 Finite Solutions for A Given Combination of Intervals:

1.1 All of the points are local minimum points: It is possible (although very unlikely) that x* consists of all local minimum points whose values are equal to each other. To find these points, we check the intervals from all indices whose lengths are less than ε and check if $\sum_{i\in I} a_{ij_i} \overset{\varepsilon}{=} N$ or $\sum_{i\in I} b_{ij_i} \overset{\varepsilon}{=} N$.

1.2 n-1 points are local minimum points, 1 is obtained from an interval: To find these points, we find $\binom{n}{n-1}$ combinations of local minimum points. We sum these points, and check if N - $\sum_{i\in I^{n-1}} a_{ij_i}$ or N - $\sum_{i\in I^{n-1}} b_{ij_i}$ is in one of the intervals belonging to the index not included in the n-1 points (within some error-tolerance ε).

2 (Possibly) Infinite Solutions for A Given Combination of Intervals

Recall that, for a Type 3 solution, a given combination of intervals results in a unique solution only when all regular $f_i(x_i^*)$'s are equal to each other and are either all increasing or all decreasing. Furthermore, when some boundaries are involved, the non-local minima x_i^*'s will also have the same characteristics as Type 2 Solutions, and thus we can use the same procedure as before to solve for regular and boundaries x_i^*'s. The first step is thus to obtain all combinations of a_{ij_i}'s, i in I_k, from intervals whose lengths are less than ε, and obtain $\frac{1}{2}\sum_{i\in I^k} (a_{ij_i} + b_{ij_i})$, i.e., the sum of the midpoints of the intervals whose lengths are less than ε. The reason that a midpoint is chosen is to reduce any possible error of ε for each index. By taking the midpoints, it is hoped that the amount of error willl be reduced. The next step is to obtain $\sum_{i\in I-I^k} b_{ij_i}$ (or $\sum_{i\in I-I^k} a_{ij_i}$) from non local minima intervals. If $\sum_{i\in I-I^k} b_{ij_i} < N$ (within some error-tolerance ε) or if $\sum_{i\in I-I^k} a_{ij_i} > N$, then the current combination of the intervals will not generate a solution. However, if $\left| \sum_{i\in I-I^k} b_{ij_i} - (N - \frac{1}{2} \sum_{i\in I^k} (a_{ij_i} + b_{ij_i})) \right| < \varepsilon$, i.e., if the sum of the upper bounds of the regular intervals is equal to $\frac{1}{2} \sum_{i\in I^k} (a_{ij_i} + b_{ij_i})$ within some error-tolerance ε, or if $\left| \sum_{i\in I-I^k} a_{ij_i} - (N - \frac{1}{2} \sum_{i\in I^k} (a_{ij_i} + b_{ij_i})) \right| < \varepsilon$ (the lower bounds), then the current combination of the intervals will generate a unique x_i^*. If $\sum_{i\in I-I^k} a_{ij_i} < N < \sum_{i\in I-I^k} b_{ij_i}$, then the current combination of the intervals will generate an infinite number of such x_i^*. The output for this combination of intervals can be simply the intervals themselves that are later shortened to reduce redundancy. The following describes the generation of the possible combinations in a greater detail:

2.1 A Unique Solution for A Given Combination of Intervals:

k points, $0 < k < n-1$, are local minimum points, n-k points are obtained from regular intervals, where n-k points are either all a_{ij_i}'s or all b_{ij_i}'s, which implies that either the rest of the points are all regular increasing points and upper bounds, or all regular decreasing points and lower bounds: To find these points, we find $\binom{n}{k}$ combinations of local minimum points, and $\frac{1}{2} \sum_{i\in I^k} (a_{ij_i} + b_{ij_i})$, i.e., the sum of the midpoints of all the single-point intervals. The next step is to find $\binom{n}{n-k}$ combinations of either all a_{ij_i}'s or all b_{ij_i}'s from the intervals taken from indices $I - I_k$ such that $\left| \sum_{i\in I-I^k} a_{ij_i} - (N - \frac{1}{2} \sum_{i\in I^k} (a_{ij_i} + b_{ij_i})) \right| < \varepsilon$ or $\sum_{i\in I-I^k} b_{ij_i} - (N - \frac{1}{2} \sum_{i\in I^k} (a_{ij_i} + b_{ij_i})) \Big| < \varepsilon$, i.e., we attempt to find either all a_{ij_i}'s or all b_{ij_i}'s from the regular intervals that will sum up to $N - \frac{1}{2}\sum_{i\in I^k}(a_{ij_i} + b_{ij_i})$, within some error-tolerance ε.

2.2 An Infinite Number of Solutions for A Given Combination of Intervals:

Once it is determined that a combination of intervals does not hold any of the properties described in (2.1), we will check if the given combination of intervals actually provide an infinite number of solutions. Note that this type of solutions will be obtained only when the number of minimum points does not exceed n-2, because otherwise there is only possible value of the last x_i^* that can satisfy $\sum_{i\in I} x_i^* = N$. The number of minimum points cannot be less than 1 either, because otherwise we will obtain Type 2 solutions. Thus the possible number of minimal points are {1, ..., n-2}. We will use a similar procedure to (2.1) to determine the possible combination of intervals. Again, the first step is to find all $\binom{n}{n-k}$ possible combinations of k minimal points, and obtain $\frac{1}{2}\sum_{i\in I^k}(a_{ij_i} + b_{ij_i})$. The next step is to check if the given combination of intervals does provide a solution by ensuring that neither $\sum_{i\in I-I^k} a_{ij_i} > N - \frac{1}{2}\sum_{i\in I^k}(a_{ij_i} + b_{ij_i})$ nor $\sum_{i\in I-I^k} b_{ij_i} < N - \frac{1}{2}\sum_{i\in I^k}(a_{ij_i} + b_{ij_i})$. However, if $\sum_{i\in I-I^k} a_{ij_i} < N - \frac{1}{2}\sum_{i\in I^k}(a_{ij_i} + b_{ij_i}) < \sum_{i\in I-I^k} b_{ij_i}$, then we can conclude that an infinite number of solutions exist. If we observe these regular intervals that will produce an infinite number of solutions, we will note the similarities between these intervals and the intervals representing the domain of each f_i. We know that all possible combinations of the regular intervals I_{ij_i}, $[a_{ij_i}, b_{ij_i}]$, that satisfies $\sum_{i\in I-I^k} a_{ij_i} < N - \frac{1}{2}\sum_{i\in I^k}(a_{ij_i} + b_{ij_i}) < \sum_{i\in I-I^k} b_{ij_i}$ will generate an infinite number of solutions. However, it is not always the case that all $x_{ij_i} \in [a_{ij_i}, b_{ij_i}]$ can be included in the solutions. The boundaries a_{ij_i}'s and b_{ij_i}'s may not be possible to be used for a possible x_i^*. Let us observe an example (for simplicity, we will drop the notion of ε): Let $N - \frac{1}{2} \sum_{i \in I^k} (a_{ij_i} + b_{ij_i}) = 10$, and let the regular intervals be [1, 5], [2, 3], and [1, 4]. We will check if all the points on the intervals can be used to generate a solution. Now, for any x_i^* obtained from any interval, note that the smallest value it can attain is when the other x_i^*'s taken from the other intervals are the largest they can attain on those intervals, i.e., the upper limits, b_{ij_i}'s. For example, the smallest x_i^* that can be taken from [1, 5] is when the other x_i^*'s are 3 and 4, which sums up to 7. Since we need to sum the points from the intervals to $N - \frac{1}{2} \sum_{i \in I^k} (a_{ij_i} + b_{ij_i}) = 10$, the smallest possible x_i^* taken from [1, 5] is thus 10 - 7 = 3. Applying this procedure to all the intervals, we obtain the shortened intervals [3,5] from [1,5] and [2,4] from [1,4]. The interval [2,3], however, is not changed, since the summation of the other b_{ij_i}'s is 9, which, when subtracted from 10, gives us 1 as the lower limit of [2,4]. Once we adjust the a_{ij_i}'s from the regular intervals, we will need to adjust the b_{ij_i}'s. Analogously, the largest value that x_i^* can take from any interval is when each of the other xi*' s takes the smallest value on its interval, a_{ij_i}'s. Let us observe what happens when we attempt to lower the b_{ij_i}'s by assigning all the other x_i^*'s with a_{ij_i}'s. In this case, the largest x_i^* that can be taken from [3,5] is when the other x_i^*'s are 2 and 2, which sums up to 4. From this calculation, we observe that the largest x_i^* can be is 6. But since b_{ij_i} is already less than 6, the interval [3,5] remain the same. For the interval [2,4], the sum of the other a_{ij_i}'s is 3+2 = 5. Again since 4 is already less than 5, we do not shorten the interval [2,4]. It will be understood from the intervals that any possible combination of x_i^* taken from these intervals that will sum to N is a solution. Thus our final results will consist of all the minimal points and the (possibly shortened) intervals. Each of the regular intervals I_{kj_k} will become

$$[\max\{a_{kj_k}, N - \tfrac{1}{2}\sum_{i\in I^k} a_{ij_i} + b_{ij_i}) - \sum_{i\in I-I^k-\{k\}} b_{ij_i}\},$$
$$\min\{b_{kj_k}, N - \tfrac{1}{2}\sum_{i\in I^k} a_{ij_i} + b_{ij_i}) - \sum_{i\in I-I^k-\{k\}} a_{ij_i}\}]$$

where $N - \frac{1}{2}\sum_{i\in I^k} a_{ij_i} + b_{ij_i}) - \sum_{i\in I-I^k-\{k\}} b_{ij_i}$ is the smallest value that a_{kj_k} can be to obtain the sum of N, and $N - \frac{1}{2}\sum_{i\in I^k} a_{ij_i} + b_{ij_i}) - \sum_{i\in I-I^k-\{k\}} a_{ij_i}$ is the largest value that b_{kj_k} can attain to obtain the sum of N. Note than once we have found the new lower bounds for each f_i, we will use these new lower bounds to calculate the new upper bounds for each f_i (or vice versa).

3.10 Generating Combinations of x_i^* - The Combined Procedure

3.10.1 Types 1, 2 and 3 (Partial) Finite Solutions:

The following procedure, which is used repeatedly, can be used to generate the possible (finite) combinations of solutions of Types 1, 2, and 3 (with the conditions mentioned in the Theorem). Since the procedure finds all possible combinations of a's and b's only, it is appropriate to be used to find Types 1 and 2 solutions. The conditions discussed for finite solutions for Type 3 are basically that the solution consists of minimum points and either all regular increasing values and upper bounds or all regular decreasing and lower bounds. The conditions of having all regular increasing values and upper bounds or all regular decreasing and lower bounds are the same with Types 1 and 2 solutions. We now need only consider the significance of the minimum points to this solution. Since a minimum point is represented by a (very) short interval, which theoretically represents a single point, we can assume that a_{ij} taken from an interval whose length is less than ε is essentially the same as b_{ij} taken from the same interval. If we can relax the restriction of using the midpoints of single-point intervals and instead use only a_{ij}'s or b_{ij}'s of these intervals, the previous method has actually covered the type of solutions in which all other x_i^*'s that are not the local minima have to be either all increasing and upper bounds, or all decreasing and lower bounds. The procedure below describes it in a more detailed manner. It has the same major idea with the procedure used to determine if some y' is attained by F(x). The only difference is that we call it to check (almost) all of the intervals exhaustively, not only when we find one solution.

Once the value of y* is determined, Procedure InitComboSol (Initialize the Combination of Solution) is executed. It will be called twice: the first time to find all the possible solutions consisting of a_{ij_i}'s only and the second time to find all the possible combinations of b_{ij_i}'s. Note that once a combination of intervals whose a_{ij_i}'s sum up to N has been found, we will no longer be able to use the b_{ij_i}'s from the same combination of intervals to generate N. It is possible to store all the combinations of a's in the first call of InitComboSol that can be referenced later when we check the combination of b's, i.e., if a combination of intervals has been stored in this list, then we will not attempt to check if the b's corresponding to the same combination of intervals will sum up to N. This method, however, is not suggested when there are many possible solutions, because the time it takes to check if a certain combination of intervals is in the list may be expensive, in which a simple checking every time may actually require fewer calculations. Since the procedure called by InitComboSol to find the combination of all a_{ij_i}'s or all b_{ij_i}'s is essentially the same, the procedure below only provides the method when we need to find the combination of a_{ij_i}'s. The following paragraph also explains the steps that are performed by using a_{ij_i}'s.

Similar to the procedures CheckYAttained and GetNextComboInt, procedure InitComboSol calls procedure GetNextComboSol (Get the Next Combination of Solution) repeatedly until there are no more possible intervals to be checked, which is indicated by the value of $k = 0$. It first initializes the current pointer j of each index to the first interval to 1. One difference between GetNextComboInt and GetNextComboSol is that, instead of finding an n-th interval which results in a combination of intervals that satisfies $\sum_{i \in I} a_{ij_i} \leq N \leq \sum_{i \in I} b_{ij_i}$, GetNextComboSol checks if there exists an n-th interval that satisfies $\sum_{i \in I_{n-1}} a_{i1} + a_{nj_n} = N$. If we find a combination of a_{ij_i}'s that sums to N, then the current combination is a solution. Recall from Theorem 1 that we conclude a given y' is attained by F(x) if the condition $\sum_{i \in I} a_{ij_i} \leq N \leq \sum_{i \in I} b_{ij_i}$ is satisfied, even though all elements of the intervals have functional values strictly less than y', due to the connectedness of the range of F(x). This time, it is enough to conclude that any possible combination of a_{ij_i}'s or b_{ij_i}'s that satisfies $\sum_{i \in I} a_{ij_i} = N$ or $\sum_{i \in I} b_{ij_i} = N$ is a solution, because, again, all $f_i(a_{ij_i})$'s and $f_i(b_{ij_i})$' are less than or equal to y*. If any one of them is equal to y*, then clearly the current combination is a solution. Otherwise, this implies that we obtain some combination of points, x', with F(x') < y*, which is a contradiction, since y* is a minimum. This hypothetical value of F(x) must have been obtained in the previous step of the algorithm. Therefore, once such combination is found, we know that it is a solution and output it.

Very similar to GetNextComboInt, finding the correct n-th interval is performed in a binary search manner with the use of j_n: it first checks the middle intervals of Intervals[n], and depending on the value of N and the current combination of $[a_{i1}]_{i=1}^{n-1}$, it will either checks the intervals on the left or right of the middle interval. This is repeatedly done on the index n until a conclusion is found: whether or not there exists any a_{nj_n} that will sum to N. The same procedure is performed for every possible combination of $[a_{ij_i}]_{i=1}^{n-1}$. If a desirable a_{nj_n} is found then we know that a_{nj_n} cannot be used to satisfy the combination $[a_{ij_i}]_{i=1}^{n-2} | [a_{n-1,j_{n-1}+1}]$, i.e., the same combination of a's from indices {1, 2, ..., n-2}, with the n-1th a, $a_{n-1,j_{n-1}+1}$, the next a of $a_{n-1,j_{n-1}}$, because it will surely result in a summation of numbers greater than N. And this is true for each other a_{nk}, where $k > j_n$. So, in this step we will again use the array Range[i] to indicate the last possible index of intervals of index i to be checked. So our next step is to set Range[n] to be j_n-1, the index of the interval located immediately on the left side of I_{nj_n} - the interval whose a_{nj_n} satisfies the summation to equal N.

What if we are not able to find the desirable a_{nj_n}? Note that with the use of j_n, whenever we fail to find such a_{nj_n}, the pointer j_n will now be located at some k, where all elements in I_{nk} are strictly greater than any possible values of a desirable a_{nj_n} for the given combination $[I_{ij_i}]_{i=1}^{n-1}$. In this case, therefore, we again let Range[n] be j_n-1. This is another difference with GetNextComboInt. In GetNextComboInt, we will only set Range[n] = j_n-1 when we find some j_n such that $\sum_{i\in I} a_{ij_i} > N$ (not when we find a desirable combination of intervals, i.e., some interval I_{nj_n} such that $\sum_{i\in I} a_{ij_i} \le N \le \sum_{i\in I} b_{ij_i}$). This is since the lengths of intervals vary and thus having one combination of intervals $[I_{ij_i}]_{i=1}^{n-2} | [I_{n-1,j_{n-1}}] | [I_{nj_n}]$ satisfy the inequality $\sum_{i\in I} a_{ij_i} \le N \le \sum_{i\in I} b_{ij_i}$ does not imply that the combination $[I_{ij_i}]_{i=1}^{n-2} | [I_{n-1,j_{n-1}+1}] | [I_{nj_n}]$ will not satisfy the inequality. For example, if we let N = 8, and $\{[I_{1j}]\} = \{[1, 3], [4, 5]\}$ and $\{[I_{2j}]\} = \{[3,8]\}$, both [1, 3] and [4, 5] can be paired with [3,8] to satisfy the inequality. Therefore, we would only set Range[n] when we encounter some I_{nj_n} that results in $\sum_{i\in I} a_{ij_i} > N$ (otherwise, we did not change Range[n] at all). When dealing with exact points and equality, however, finding some a_{nj_n} that results in $\sum_{i\in I} a_{ij_i} = N$ will necessarily imply that $\sum_{i\in I_{n-2}} a_{ij_i} + a_{n-1,j_{n-1}+1} + a_{nj_n} >$ N, because all a_{ij_i}'s, $i \neq n-1$, remain the same, but $a_{n-1,j_{n-1}+1} > a_{n-1,j_{n-1}}$, and thus the new summation will certainly exceed N.

Now, by repeatedly increasing the pointer to the n-1th interval, j_{n-1}, we will finally finish checking the last interval of the n-1th index or we will encounter some $a_{n-1,j_{n-1}}$ such that $\sum_{i \in I_{n-1}} a_{ij_i} + a_{n1} > N$. GetNextComboSol will then increase j_{n-2}, and let $j_{n-1} = 1$, and repeat the above procedure with the new $[I_{ij_i}]_{i=1}^{n-1}$, where the new $[I_{ij_i}]_{i=1}^{n-1}$ equal the previous $[I_{ij_i}]_{i=1}^{n-3} | [I_{n-2,j_{n-2}+1}] | [[I_{n-1,1}]$. Again we need to set the new Range[n], and let it be the greatest index k such that $\sum_{i \in I_{n-1}} a_{ij_i} + a_{nk} \le N$. Once this is set, we again perform a binary search on the n-th set of intervals.

To generalize the previous procedure by letting n-1 be some index k, the next step is basically to decrement k whenever j_k reaches Range[k] or whenever $\sum_{i \in I_k} a_{ij_i} + \sum_{i \in \bar{I}_k} a_{i1} > N$, i.e., whenever a_{kj_k} is the largest possible to still sum to N. We then set j_i, where $i \in \bar{I}_k$, to start generating new combinations for the new $[a_{ij_i}]_{i=1}^{k}$. We also set the range Range[i] to be the largest k_i possible to still sum to N for the current combination of $[a_{ij_i}]_{i=1}^{k}$, i.e., the largest k_i such that $\sum_{p \in I_k} a_{pj_p} + \sum_{m \in \bar{I}_k - \{i\}} a_{m1} + a_{ik_i} \le N$. We repeat this procedure of decreasing k and increasing j_k until at some point k=0, in which the exhaustive procedure of finding all possible combinations of solutions is terminated.

```
procedure InitGetComboSol
begin
   If not ($\sum_{i \in I} a_{i1}$ > N or $\sum_{i \in I} b_{i,NIntervals[i]}$ < N) then
      k := n-1                                                    (*)
      for i := 1 to n-1 do
         $j_i$ = 1
         Range[i] := NIntervals[i]
      end do
      j[n] := (NIntervals[n]+1) div 2
      Range[n] := NIntervals[n]
      while k > 0 do
         GetNextComboSol with a's                                    (**)
      Repeat (*) through (**) with b's
   end if
end


procedure GetNextCombo;
var CurrIndex, k: integer;
begin
   if $\sum_{i \in I_{n-1}} a_{ij_i} + a_{n1} \leq N$ and $j_{n-1}$ <= Range[n-1] , i.e, there is at least
      one more interval to be checked on the set of intervals of index
      n-1, then
      Increase $j_{n-1}$ {check the next interval}
      Perform binary search on Intervals[n], from $n_1$ to $n_{Range[n]}$   (***)
      If found, then
            Output combination
      end if
      Decrease Range[n]                                                (****)
   else
      k := n-1
      repeat
```

```
        if $\sum_{i\in I_k} a_{ij_i} + \sum_{i\in \bar{I}_k} a_{i1}$ > N then {if the rest of intervals will not
           generate a solution, then we will not check them}
           Decrease k
     until k = 0 or $\sum_{i\in I_k} a_{ij_i} + \sum_{i\in \bar{I}_k} a_{i1} <$ N or j[k] < Range[k]
     If k > 0 then
        Increase $j_k$
        if k < n-2 then  {we had to decrease k at least twice}
           for i := k+1 to n-1 do
              $j_k$ := 1
              Range[i] := the largest $k_i$ such that $\sum_{p\in I_k} a_{pj_p} + \sum_{m\in \bar{I}_k-\{i\}} a_{m1} + a_{ik_i} \le N$
           end for
     end if
     Repeat Steps (***) through (****)
  end if
end
```

3.10.2 Type 3 Solutions with only One Index Whose Functional Value Is Less than $f_m(x_m)$

This is when the classication of intervals plays a significant role. The idea is that this type of solutions consists of only all minimum points except one. Thus we do not need to execute the procedure when the number of indices of x_i^*'s that are minimum points is less than n-1. Otherwise, the basic idea is to find all possible combinations of n-1 indices, and find the last one that can be attained from any point in a regular interval. To avoid obtaining the last point from an interval whose length is less than ε, in which we have a solution that consists of all local minimum points of the functions, we will only find a regular interval that will result in a combination of intervals in which the summation of all a_{ij_i}'s is strictly less than N and N is strictly less the summation of b_{ij_i}'s. This is since this case is already covered by the previous procedure. We are now left with the condition where the last x_i^* is exclusively between the lower and upper bounds of an interval, which can be regular interval or a boundary interval. The basic idea of the procedure however, is similar to the previous procedures. The only difference is that we will check different combinations of n-1 indices for the minimum points: in the beginning, we will generate all the possible combinations of n-1 minimum points (after eliminating ones that are greater than N) by taking the indices of {1, 2, .., n-1}, and search for the correct point in the correct regular/boundary interval on the n-th index. We attempt to perform a binary search on the n-th interval looking for an interval that satisfies $a_{nj} < N - \sum_{i \in I^{n-1}} a_{ij_i} < b_{nj}$ (Note that here $I^{n-1} = I_{n-1}$). If found, we will output the n-1 points and $N - \sum_{i \in I^{n-1}} a_{ij_i}$ as the current combination of solution. The next step is to generate all possible combinations of n-1 minimum points taken from the indices {1, 2, ..., n-2, n} and search for the correct point in the correct regular/boundary interval on the n-1th index, i.e., the missing index from the set of minimum indices. If in the previous procedure, for any index k, we ignore the intervals starting from the interval with the condition that $\sum_{p \in I_k} a_{pj_p} + \sum_{m \in \bar{I}_k - \{i\}} a_{m1} + a_{ik_i} > N$, here we ignore the intervals starting from the interval with the condition $\sum_{p \in I^k} a_{pj_p} + \sum_{m \in \bar{I}^k - \{m\} - \{i\}} a_{m1} + a_{ik_i} > N$, where m is the missing index among {1, 2, ..., n}, i.e., the index whose value is taken from some *open* interval (a_{mj_m}, b_{mj_m}). We will omit the exact procedure of this step since the idea of the generating all these possible combinations is very similar to the previous one. The following provides a scheme to reduce the number of combinations that need to be checked before the actual procedure is executed.

```
procedure ReduceComboOfMinPoints(MissingIndex:integer)
begin
   for i := 1 to MissingIndex-1 do
      j := 1
      while MinPoint[i][j] < N - Summation(MinPoint[i][NMinPoints[i]] of
         other indices - b[MissingIndex][NRegIntervals[i]] and j <=
         NMinPoints[i] do
         discard MinPoint[i][j] from possible checks
         j := j + 1
      end do
      if there is no more MinPoint left for index i then
        return
      end if
   end for
   for i := MissingIndex+1 to n do
      [Repeat the same procedure]
   for i := 1 to MissingIndex-1 do
      j := NMinPoints[i]
      while MinPoint[i][j] > N-Summation({New}MinPoint[i][1] of other
         indices - a[MissingIndex][1] and j >= 1 do
         discard MinPoint[i][j] from possible checks
         j := j - 1
      end do
      if there is no more MinPoint left for index i then
        return
      end if
   end for
   for i := MissingIndex+1 to n do
      [Repeat the same procedure]
   end for
end procedure
```

3.10.3 Type 3 Infinite Solutions

The basic idea of the procedure is to find different combinations of single-point intervals and regular intervals. The type of output that is produced no longer consists of single points only but a combination of both points and intervals. Again we can use the same idea with the previous procedure to produce different combinations of single-point intervals; the only difference is now we find the combinations of fewer than n-1 indices. Also, whereas in the previous procedure we used the first interval of the missing index to determine the largest possible a_{ij_i} for each single-point intervals, here we need to find the summation of all the first a_{ij_i}'s of the missing indices. The method proposed here basically uses the method implemented to determine if y' is attained by F(x), but instead of finding a combination of intervals whose sum of a_{ij_i}'s and b_{ij_i}'s will sandwich N, here we need to find a combination of intervals whose sum of a_{ij_i}'s and b_{ij_i}'s will sandwich N-the sum of all minimum points. Also, once a combination is found, we will immediately output this combination; there is no need to check if y* is actually attained by this combination of intervals because we already know that y* is attained by the minimum points. Once a combination is found, we will shorten the intervals, if possible, which will become the final output.

4. An Illustration of the Algorithm

This chapter provides an illustration of the major steps of the algorithm. For simplicity, each f_i is given graphically . Unless the exact points are indicated, each figure is approximately drawn to scale. This is to eliminate the detailed steps of tracing each f_i. The parameters are as follow:

$n = 4$, i.e., $I = \{1, 2, 3, 4\}$

$N = 21$

$D_{f_1} = [2, 10]$, $D_{f_2} = [2, 6]$, $D_{f_3} = [7, 10]$, $D_{f_4} = [1, 5]$

$\varepsilon = .25$

To illustrate the main idea of the algorithm, we will avoid the preliminary step: the step in which we restrict the domain of each function f_i. The execution of the algorithm is given in the following paragraphs:

1 Finding the First Y'

$\forall x_i', x_i' = l_i \Rightarrow x_1' = 2,\ x_2' = 2,\ x_3' = 7,$ and $x_4' = 1$

$\Rightarrow \sum_{i \in I} x_i' = \sum_{i \in I} l_i = 2 + 2 + 7 + 1 = 12 < 21 = N$ (combination not yet found)

Is $\sum_{i \in I} x_i' = 12 < 21 = N \leq \sum_{i \in I_3} x_i' + u_4 = 2 + 2 + 7 + 5 = 16$?

No $\Rightarrow x_4' = u_4 = 5,\ j := n - 1 = 4 - 1 = 3$

$\sum_{i \in I} x_i' = 16 < N \Rightarrow$ need to enter the while loop

Is $\sum_{i \in I} x_i' = 16 < N \leq \sum_{i \neq 3} x_i' + u_3 = 2 + 2 + 5 + 10 = 19$?

No $\Rightarrow x_3' = u_3 = 10,\ j := j - 1 = 3 - 1 = 2$

$\sum_{i \in I} x_i' = 19 < N \Rightarrow$ need to enter the while loop

Is $\sum_{i \in I} x_i' = 16 < N \leq \sum_{i \neq 2} x_i' + u_3 = 2 + 10 + 5 + 6 = 23$?

Yes $\Rightarrow x'_2 = N - \sum_{i \neq 2} x'_i = 21 - (2 + 10 + 5) = 21 - 17 = 4$

So one possible x' is (2, 4, 10, 5). From Figure 7, we find that $f_1(2) = 7$, $f_2(4) = 6$, $f_3(10) = 3$, and $f_4(5) = 1.5$, and thus **FirstY'** = F(x') = max{7, 6, 3, 1.5} = 7

SmallestAttainY = y' = 7

2 From Figure 7, we also obtain **MaxMinF** = max{min{$f_1(x_1)$} = 2.5, min{$f_2(x_2)$ = 2, min{$f_3(x_3)$} = 3, min{$f_4(x_4)$} = 1.5} = **3.**

3 Let y' = 7 by $\dfrac{\text{MaxMinF} + y'}{2} = \dfrac{3+7}{2} = 5$

4 From Figure 8, we obtain $[\{I_{ij}(5)\}] = [\ \{[5, 8], [8, 10]\}, \{[2, 3]\}, \{[9.5,10]\}, \{[1,5]\}]$

There are 2 combinations of intervals: [[5, 8], [2, 3], [9.5, 10], [1, 5]] and [[8, 10], [2, 3], [9.5, 10], [1, 5]]. Following the algorithm, we will first examine the first combination:

a. $\sum_{i \in I} a_{ij_i} = 5 + 2 + 9.5 + 1 = 17.5 \leq N \leq \sum_{i \in I} b_{ij_i} = 8 + 3 + 10 + 5 = 26$ (first property)

b. Can we find any a_{kj_k}, b_{mj_m}, $k \neq m$, such that $f_k(a_{kj_k}) = f_m(b_{mj_m}) = 5$?

Yes, let k = 1, and m = 2, and thus y' = 5 is attained by F(x)

$\Rightarrow$ SmallestAttainY = y' = 5

c. Replace y' = 7 by $\dfrac{\text{MaxMinF} + y'}{2} = \dfrac{3+5}{2} = 4$

5 From Figure 9, we obtain $[\{I_{ij}(4)\}] = [\ \{[7, 7], [9, 10]\}, \{[2, 2.5]\}, \{[9.8,10]\}, \{[1,5]\}]$

There are 2 combinations of intervals: [[7, 7], [2, 2.5], [9.8, 10], [1, 5]] and [[9, 10], [2, 2.5], [9.8, 10], [1, 5]]. Following the algorithm, we will first examine the first combination:

a. $\sum_{i \in I} a_{ij_i} = 7 + 2 + 9.8 + 1 = 19.8 \leq N \leq \sum_{i \in I} b_{ij_i} = 7 + 2.5 + 10 + 5 = 24.5$ (first property)

b. Can we find any a_{kj_k}, b_{mj_m}, $k \neq m$, such that $f_k(a_{kj_k}) = f_m(b_{mj_m}) = 4$?

Yes, let k = 1, and m = 2, and thus $y' = 4$ is attained by F(x)

$\Rightarrow$ SmallestAttainY = $y' = 4$

c. Replace $y' = 4$ by $\frac{\text{MaxMinF} + y'}{2} = \frac{3+4}{2} = 3.5$

6 From Figure 10, we obtain $[\{I_{ij}(3.5)\}] = [\ \{[9.7, 10]\}, \{[2, 2.3]\}, \{[9.9,10]\}, \{[1,5]\}]$.

We examine this only combination:

a. $\sum_{i \in I} a_{ij_i} = 9.5 + 2 + 9.9 + 1 = 22.4 > N$ (fail the first property - y' is not attained by F(x)

b. Is $|\text{SmallestAttainY} - y'| \le \varepsilon$?

c. No $\Rightarrow$ Replace $y' = 3.5$ by $\frac{y' + \text{SmallestAttainY}}{2} = \frac{3.5+4}{2} = 3.75$

7. From Figure 11, we obtain $[\{I_{ij}(3.75)\}] = [\ \{[9.5, 10]\}, \{[2, 2.4]\}, \{[9.85,10]\}, \{[1,5]\}]$.

We examine this only combination:

a. $\sum_{i \in I} a_{ij_i} = 9.5 + 2 + 9.85 + 1 = 22.35 > N$ (fail the first property - y' is not attained by F(x)

b. Is $|\text{SmallestAttainY} - y'| \le \varepsilon$?

c. Yes $\Rightarrow y^* = \text{SmallestAttainY} = 4$

8 We know that $[\{I_{ij}(4)\}] = [\ \{[7, 7], [9, 10]\}, \{[2, 2.5]\}, \{[9.8,10]\}, \{[1,5]\}]$. We now need to find all combinations of solutions. We first classify each interval corresponding to each index:

a. SinglePoint[1] = {[7, 7]}, RegularIntervals[1] = {[9, 10]}

SinglePoint[2] = $\varnothing$, RegularIntervals[2] = {[2, 2.5]}

SinglePoint[3] = $\varnothing$, RegularIntervals[3] = {[9.8, 10]}

SinglePoint[4] = $\varnothing$, RegularIntervals[4] = {[1, 5]}

b. Find regular and boundary solutions:

The only combination of regular intervals is [[9, 10], [2, 2.5], [9.8, 10], [1, 5]].

Check if $\sum_{i \in I} a_{ij_i} = N$: $\sum_{i \in I} a_{ij_i} = 9 + 2 + 9.8 + 1 = 21.8$ (no)

Check if $\sum_{i \in I} b_{ij_i} = N$: $\sum_{i \in I} b_{ij_i} = 10 + 2.5 + 10 + 5 = 27.5$ (no)

c. Find minimum solutions:

The only combination of intervals with SinglePoint interval is [[7, 7], [2, 2.5], [9.8, 10], [1, 5]]. Since[7,7] is the only minimum point, the only possible x_k^* is 7. For this x_k^* (k = 1) we check for both finite and infinite solutions. First we obtain $N - a_{kj_k} = 14$

1 $\sum_{i \neq 1} a_{ij_i} = 2 + 9.8 + 1 = 12.8 < N - a_{kj_k}$

2 $\sum_{i \neq 1} b_{ij_i} = 2.5 + 10 + 5 = 17.5 < N - a_{kj_k}$

Since $\sum_{i \neq 1} a_{ij_i} < N - a_{kj_k} < \sum_{i \neq 1} b_{ij_i}$, there is an infinite number of solutions. For all $i \neq 1$, x_i^* is in I_{i1}.

d. Restrict each interval solution:

$a_{21} = \max\{ a_{21}, N - a_{kj_k} - \sum_{i \neq 1,2} b_{ij_i} \} = \max\{2, 14 - (10 + 5) = -1\} = 2$

$a_{31} = \max\{ a_{31}, N - a_{kj_k} - \sum_{i \neq 1,3} b_{ij_i} \} = \max\{9.8, 14 - (2.5 + 5) = 6.5\} = 9.8$

$a_{41} = \max\{ a_{41}, N - a_{kj_k} - \sum_{i \neq 1,4} b_{ij_i} \} = \max\{1, 14 - (2.5 + 10) = 1.5\} = 1.5$

$b_{21} = \min\{ b_{21}, N - a_{kj_k} - \sum_{i \neq 1,2} a_{ij_i} \} = \min\{2.5, 14 - (9.8 + 1) = 3.2\} = 2.5$

$b_{31} = \min\{ b_{31}, N - a_{kj_k} - \sum_{i \neq 1,3} a_{ij_i} \} = \min\{10, 14 - (2 + 1) = 11\} = 10$

$b_{41} = \min\{ b_{41}, N - a_{kj_k} - \sum_{i \neq 1,4} a_{ij_i} \} = \min\{5, 14 - (2 + 9.8) = 2.2\} = 2.2$

e. The solutions for $y^* = F(x)$ are all x^* such that $x_1^* = 7$, and for each $i \neq 1$, x_i^* is any combination of points taken from [2, 2.5] for i = 2, [9.8, 10] for i = 3, and [1,5, 2.2] for i = 4 that satisfy $\sum_{i \neq 1} x_i^* = 14$

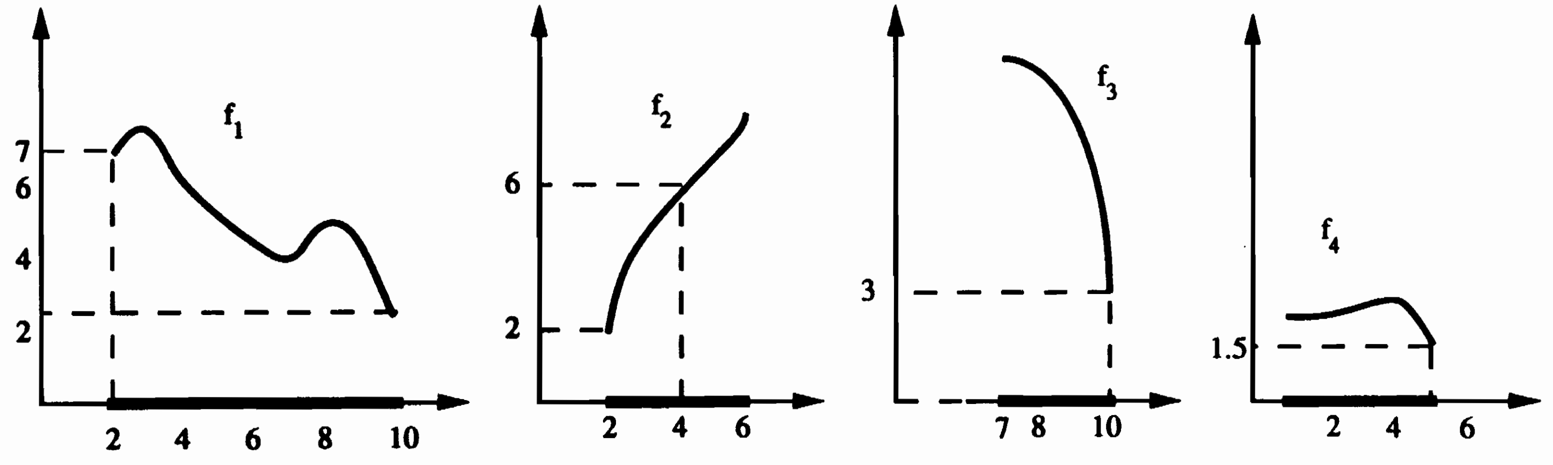

Figure 7. Finding FirstY' and MaxMinF

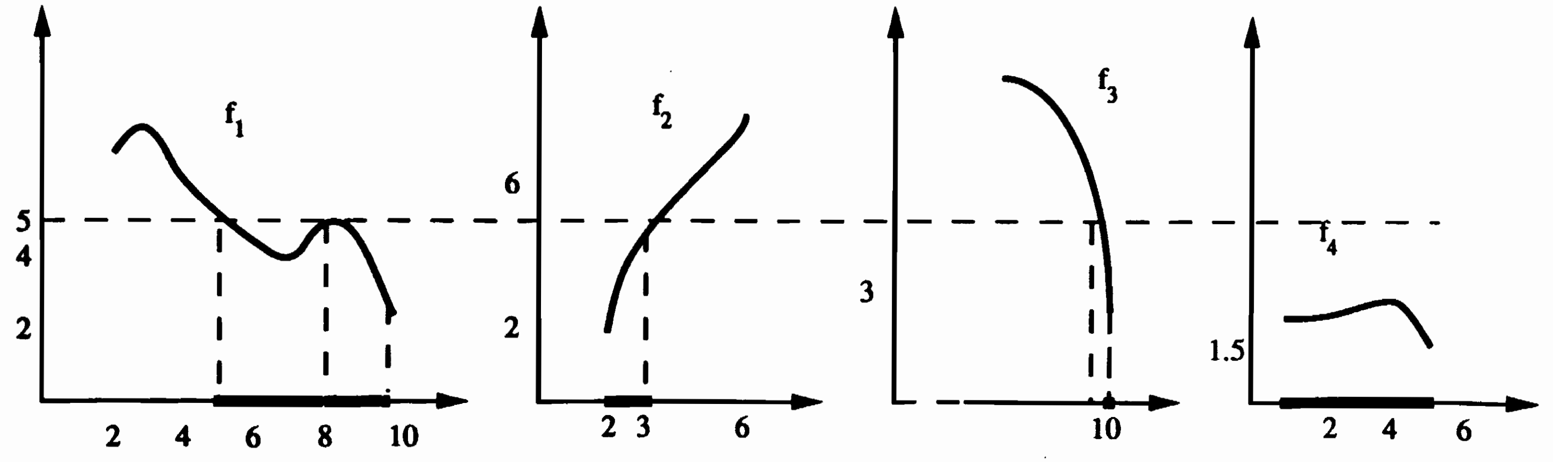

Figure 8. Determining $[\{I_{ij}(5)\}]$

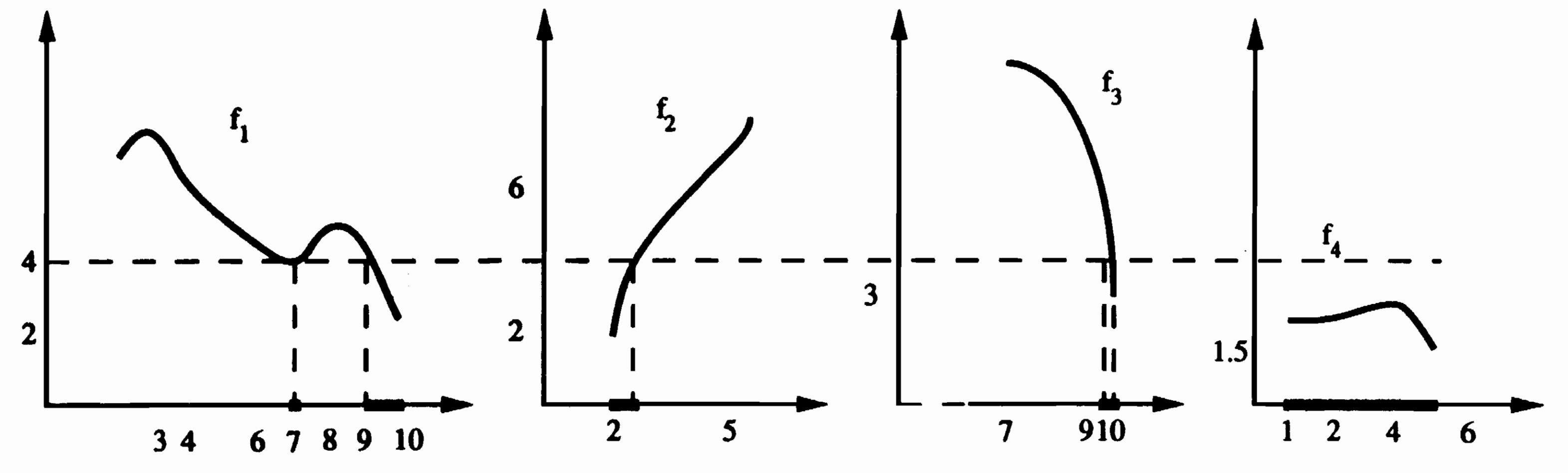

Figure 9. Determining $[\{I_{ij}(4)\}]$

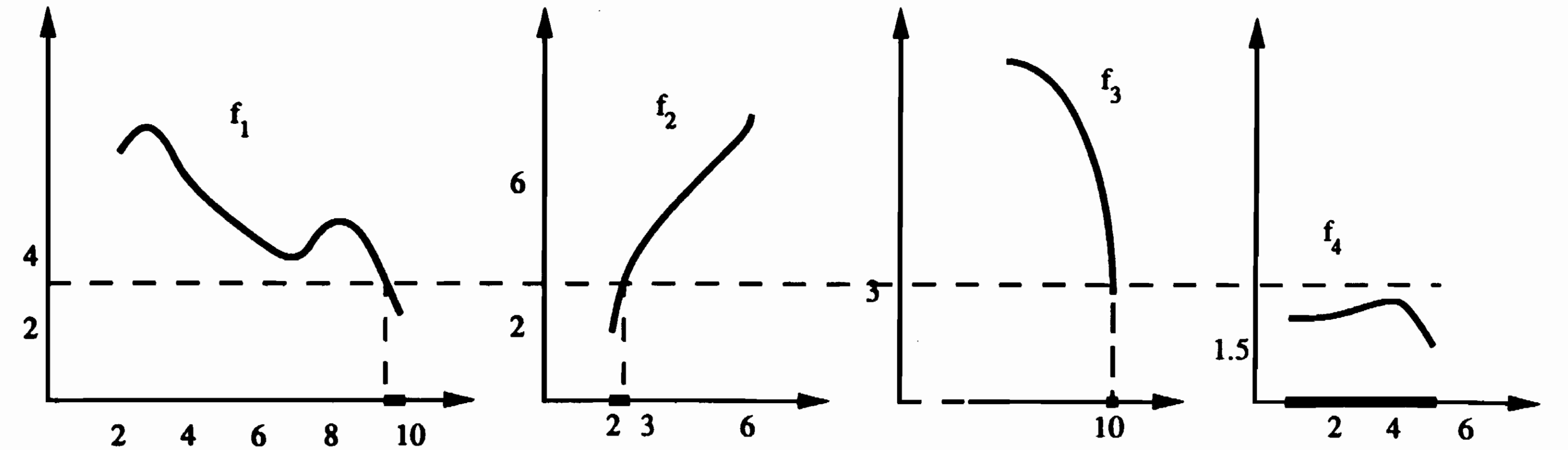

Figure 10. Determining $[\{I_{ij}(3.5)\}]$

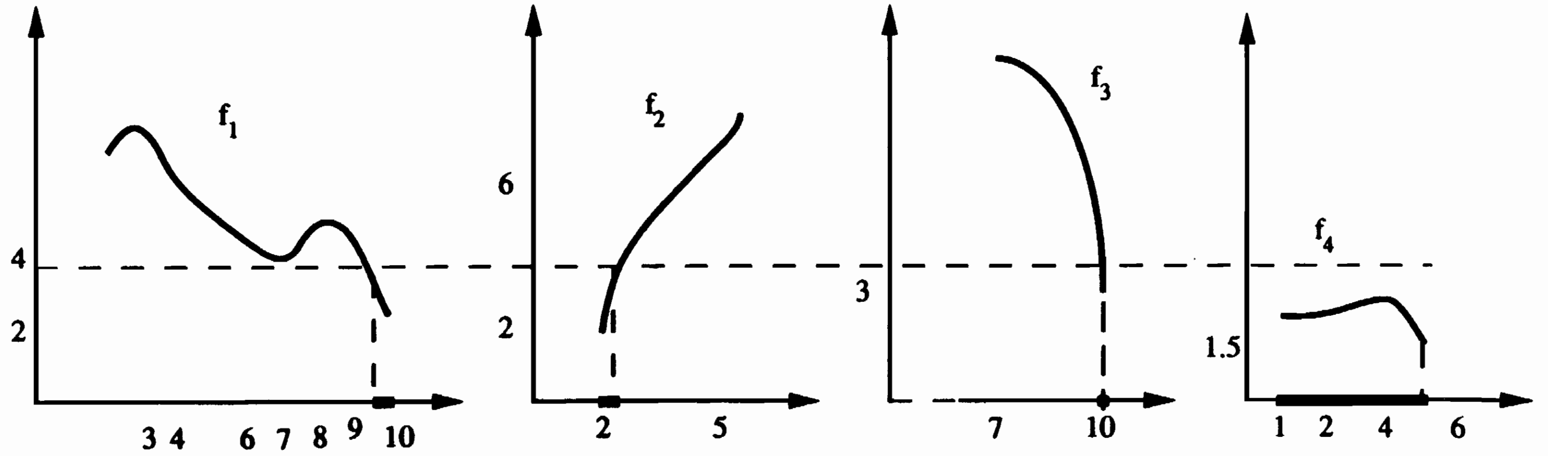

Figure 11. Determining $[\{I_{ij}(3.75)\}]$

5. Conclusions and Further Research

We have developed an algorithm to find all solutions x^* to the equation $y^* = \min\{F(x)\} = \min\{\max\{f_i(x_i)\}\}$ for continuous variable x_i's satisfying $\sum_{i \in I} x_i = N$, where each f_i can be multimodal.

[IbKa88] described a method to solve an integer resource allocation problem which is based on the continuous counterpart, called continuous relaxation. These results might suggest that further research to solve the integer version of this minimax resource allocation problem which is based on the methods presented in this thesis may be worth investigating. It is also possible that any future approach to solve the integer version might be completely independent of the continuous one. [IbKa88], however, states that the complexity of the integer version is NP-hard.

Because of the limitations of space and time, this work did not address a number of details related to the solution of problems described briefly in the following paragraphs:

1 In many practical situations, it might be sufficient to obtain only *one* solution to the problem. In this case, we can modify the procedures for finding all the solutions to terminate once a solution is found. The paper provides the method to find *all* solutions for both theoretical and practical reasons. One of the theoretical objectives is to emphasize the significance of the theorems - that the above solutions are the only possible ones, which signifies both sufficiency and necessity of the characteristics of x^*. A major practical purpose is to enable further levels of optimization. Given all possible x^*, we can formulate a second objective function to be minimized over the domain of all possible x^*. Clearly, this can only be performed when x^* is not unique.

When there are an infinite number of solutions, i.e., when the solutions include Type 3 Infinite Solutions, further levels of optimization will be more feasible. We might wish to find a lexicographically optimal solution, such as one proposed in [Fu80].

2 The procedure used in determining $[\{ I_{ij_i} \}]$ for a given y', i.e., by tracing each f_i for every increment Δ allows us to have a more general specification of f_i in which $f_i(x_i)$ is not given as a clsoed form analytical expression of x_i: $f_i(x_i)$ can now be given in tabulated form. We still, however, need to ensure that $f_i(x_i)$ is never constant on any interval. If this is not the case, then some of the procedures used will need some modifications. The possible values for regular intervals will no longer consist of combinations of only either all a_{ij_i}'s or all b_{ij_i}'s, but they can also consist of combinations of constant intervals, such that any point in between the limits of the intervals may be a possible x_i^*.

It is possible to specify each f_i piece-wise over a number of intervals in its domain. If each of the portion of this function is strictly monotone, then finding the intervals $\{I_{ij}\}$ on each monotone portion of the curve can be efficiently accomplished with a binary search. Computational efficiency can be improved further when the inverse function of each monotone portion is known, such as logarithmic, exponential, or linear functions. In such a case, tracing the function f_i is not necessary.

3 Tracing each f_i to find intervals whose functional values are less than or equal to y' using a given Step-size carries with it the risk of missing a very short I_{ij}. One way to reduce the possibility of such error is to retrace, using a smaller Step-size, each portion of each function whose functional values are very close to y'. Note that this refinement of the method does not completely eliminate the possibility of missing such interval. However, this is a common problem encountered in numerical approaches to find a solution.

4 When there is an infinite number of solutions, i.e., when the output includes some combination of intervals, we can give more specific outputs by actually producing combinations of numbers generated from these intervals: we take a (finite) number of points from each interval with the use of some increment Δ. When the algorithm is required to generate exact points, this mechanized method may be a better approach.

The above options and alternatives of the algorithm, however, were not implemented in this thesis to maintain its simplicity and general applicability. There are certainly other possible ways to improve the algorithm, which depend on the particular application and the types of inputs and functions involved.

6. References

[ChCo58] A. Charnes and W. Cooper. "The Theory of Search: Optimal Distribution of Effort". *Management Science*, 5 (1958), 44-49.

[EGP76] E. Elton, M. Gruber, and M. Padberg. "Simple Criteria for Optimal Portfolio Selection". *J. Finance*, 31 (1976), 1341-1357.

[FeZi83] A. Federgruen and P. Zipkin. "Solution Techniques for Some Allocation Problems". *Mathematical Programming*, 25 (1983), 13-24.

[FKI88] S. Fujisage, N. Katoh, and T. Ichimori. "The Fair Resource Allocation Problem with Submodular Constraints" (To appear in *Mathematics of Operations Research*).

[Fu80] S. Fujishige. "Lexicographically Optimal Base of A Polynomial with Respect to A Weight Vector". *Mathematics of Operations Research*, 5 (1980), 186-196.

[Ha89a] E. Haddad. "Attainable Resource Allocation Bounds". TR89-24, *Virginia Polytechnic Institute and State University* (June 1989).

[Ha89b] E. Haddad. "Minimax Resource Allocation with Continuous Variables: the Definitive Solution". TR 90-1, *Virginia Polytechnic Institute and State University* (February 1990).

[IbKa88] Toshihide Ibaraki and Naoki Katoh. Resource Allocation Problems: Algorithmic Approaches. Cambridge: The MIT Press, 1988.

[Ic84] T. Ichimori. "On Min-Max Integer Allocation Problems". *Operations Research*, 32 (1984), 449-450.

[KIM85] N. Katoh, T. Ibaraki, and H. Mine. "An Algorithm for the Equipollent Resource Allocation Problem". *Mathematics of Operations Research*, 10 (1985), 44-53.

[Ko71] P. Kotler. Marketing Decision Making: A Model Building Approach. New York: Holt, Rinehart and Winston, 1971.

[LuSm86] H. Luss and D. R. Smith. "Resource Allocation among Competing Activities: A Lexicographic Minimax Approach". *Operations Research Letters*, 5 (1986), 227-331.

[Ru91] William H. Ruckle. Modern Analysis: Measure Theory and Functional Analysis with Applications. Boston: PWS-Kent Publishing Company, 1991.

[Sa70] S. Schaible. A Survey of Fractional Programming, in S. Schaible and W.T.Ziemba, (eds.). Generalized Concavity in Optimization and Economics. New York: Academic Press, 1981.

[Ze81] Z. Zeitlin. "Integer Allocation Problems of Min-Max Type with Quasi Concave Separable Functions". *Operations Research*, 29 (1981), 207-211.

[Zi82] H. Ziegler. "Solving Certain Singly Constrained Convex Optimization Problems in Production Planning". *Operations Research Letters*, 1 (1982), 246-252.

7. Vita

Aida Dharmakadar

6721 Kenyon Drive

Alexandria, Virginia 22307

703-765-8031(H) 703-765-8458(Fax) 703-558-3364(W)

E-mail: dharmaka@goliat.cs.vt.edu

Education:

M.S. in Computer Science & Applications, September 1993,
Virginia Polytechnic Institute & State University,
Northern Virginia Graduate Center, Falls Church, Virginia.
3.9/4.0 Cumulative GPA.

B.S. in Computer Science and Mathematics, May 1992,
George Mason University, Fairfax, Virginia.
4.0/4.0 GPA in both majors; 3.9/4.0 overall.
Graduated with Highest Distinction.

Courses included in both schools:
Information Storage & Retrieval, Systems Simulation, Operating Systems, Analysis of Algorithms, Artificial Intelligence, Data Base Management, Systems Programming, Computer Architecture, Comparative Programming Languages, Data Structures, Digital Electronics, Computer Algebra, Numerical Analysis, Applied Mathematics,

	Probability & Applications, Discrete Mathematics, Elementary & Advanced Calculus, Theoretical Linear Algebra, and Matrix Algebra.
Languages:	Assembly Language for the IBM PC, BASIC, C, C++, COBOL, Fortran, GPSSH, JCL, LISP, Maple, Matlab, Pascal, Prolog, and SIMSCRIPT
Operating Systems:	IBM Mainframe (TSO), Unix, VMS, DOS, Macintosh
Honors/Affiliations:	Awarded the Mathematics Achievement Award 1992 (GMU) Outstanding Computer Science Junior 1991 (GMU) Alpha Chi College Honor Society Golden Key National College Honor Society Mathematical Association of America
Experience:	<u>Programmer:</u> September 1993 - present. Management Systems & Technology American Management Systems *Duties:* Write and modified COBOL programs to produce reports and to update IRS data base. <u>Graduate Assistant:</u> August 1992 - August 1993. Department of Computer Science

Virginia Polytechnic Institute & State University

Duties: Assist faculty members with their research and teaching responsibilities, provided instructions and explanation on Unix/C.

Programmer:

June 1992 - August 1992.

Computer Services & Operations

George Mason University

Duties: Wrote and modified COBOL and Culprit programs to produce reports and to update GMU database.

Tutor & Grader :

September 1990 - August 1992.

Department of Mathematics

George Mason University.

Duties: Assisted students with difficulties in mathematics, graded exams and quizzes.

References: supplied upon request

[illegible]