Report Number: WUCS-79-7

1979-07-01

One Dimensional Optimization on Multiprocessor Systems

Authors: Mark A. Franklin and N. Soong

This paper presents a straightforward approach to determining how best to utilize an MIMD multiprocessor in the solution of one dimensional optimization problems involving continuous unimodal functions and nongradient search techniques. A methodology is presented which allows one to consider a variety of speedup functions which may occur in parallel function and systems evaluation. It is shown how the best of two parallel optimization strategies can be determined for a given accuracy, number of processors and speedup function.

ONE DIMENSIONAL OPTIMIZATION ON MULTIPROCESSOR SYSTEMS

M. A. Franklin and N. Soong

WUCS-79-7

Department of Computer Science

Washington University

St. Louis, Missouri 63130

July 1979

This work has been supported in part by
the National Science Foundation Grant MCS78-20731

ABSTRACT

This paper presents a straightforward approach to determining how best to utilize an MIMD multiprocessor in the solution of one dimensional optimization problems involving continuous unimodal functions and nongradient search techniques. A methodology is presented which allows one to consider a variety of speedup functions which may occur in parallel function and systems evaluation. It is shown how the best of two parallel optimization strategies can be determined for a given accuracy, number of processors and speedup function.

Key Words: Parallel Processing, Multiprocessor Systems, Optimization, Search Techniques, Parallel Numerical Methods, Unimodal Function Optimization

- - - - - - - -

*This work has been supported in part by National Science Foundation Grant MCS 78-20731.

ONE DIMENSIONAL OPTIMIZATION OF MULTIPROCESSOR SYSTEMS

by

M. A. Franklin and N. Soong

I. INTRODUCTION

The advent of large and very large scale semiconductor integration techniques has significantly reduced the cost of digital logic. One result of this has been a growing interest in obtaining enhanced computational power through the use of multiprocessor computer systems [1, 2, 3]. This enhanced computational power is needed since many of the engineering, social, economic and environmental systems currently being studied are highly complex. The computer costs associated with studying such systems can be high. Often they require the solution of large numbers of nonlinear equations, many are stochastic in nature and all require numerous computer execution runs for system model verification and analysis.

This paper examines the question of what strategies to follow when faced with a system or function optimization problem to be solved in an MIMD (multiple instruction stream--multiple data stream) computer environment [4]. The goal is to select the lower cost of two well defined one dimensional optimization strategies. The objective function to be optimized is assumed to be continuous and unimodal, although the continuity assumption can be relaxed without much difficulty. The MIMD computer is assumed to consist of N_c identical processors and a general interprocessor communications network [5]. The optimization methods used are of the direct search, nongradient variety.

A number of optimization questions have already been considered in the parallel processing environment. For instance, work has been done on the question of optimal searching and sorting of files [6, 7]. Other work has centered on the numerical properties of various parallel function optimization algorithms [8, 9, 10, 11].

This paper considers a fundamental strategy alternative available when performing one dimensional function optimization studies on an MIMD machine.

One strategy involves parallel evaluation of the function to be optimized, and the use of a sequential optimization scheme. An alternative strategy involves sequential evaluation of the function to be optimized, and the use of a parallel optimization scheme. The remainder of this paper formalizes these alternative strategies and presents a procedure for selecting the better of the two. Section II that follows defines the problem and specifies the alternatives available. Section III considers the general problem of optimal strategies, and three special cases. Section IV concludes the paper and indicates some related problems.

II. THE PROBLEM

A. The System To Be Optimized

Consider the general objective function $F(\alpha)$ which is to be optimized with respect to a scalar parameter α where α is restricted to the finite range $[0,R_o]$. The function F is taken to be a continuous unimodal function of α, although the continuity property can be relaxed without much difficulty. Note that F may be quite complicated, and may involve solution of a system of operator equations such as differential equations.

Assume that a parallel algorithm is available so that F can be evaluated in an MIMD environment. Such parallel algorithms have been explored for example for both systems of differential equations [12, 13, 14] and partial differential equations [15, 16], systems of algebraic equations [17] as well as other types of systems [1].

Associated with such parallel algorithms is a speedup function, S_p. S_p relates the solution speed in the single computer case to the solution speed with N_c computers present (i.e. an MIMD machine). If $t_p(N_c)$ is the time it takes to evaluate the function with N_c computers, then the speedup $S_p = t_p(1)/t_p(N_c)$. Obtaining the speedup function is a difficult matter and is dependent on an understanding of the function to be optimized, and a detailed analysis of appropriate parallel evaluation algorithms. The references cited above contain some typical approaches to obtaining S_p and the analysis that follows assumes that such S_p curves are available.

The optimization problem concerns finding a value of the parameter α such that the general objective function, $F(\alpha)$ is within a range $[0,R_N]$ of the true optimum. This if α^* is the parameter value at which $F(\alpha)$ is optimized, the problem is to find an α such that

$$|\alpha-\alpha^*| \leq R_N \tag{1}$$

B. Two Optimization Procedures

The optimization procedures considered are in the class of nongradient search techniques. These lend themselves to one dimensional optimization problems, and are often key elements upon which multidimensional gradient approaches are based. In addition these techniques have the property that

the rate of convergence to the optimal is not dependent on the characteristics of the function to be optimized. The result is that a quantitative comparison can be made between alternative optimization strategies on a function independent basis. Finally it should be noted that there are many functions which are only piecewise smooth (e.g. piecewise linear functions) and are often not efficiently solved using gradient and other higher order methods. The non gradient techniques discussed next are applicable to these functions even though for presentation simplicity, the derivations assume the presence of continuous functions.

Two types of nongradient searches are considered; sequential and parallel or simultaneous. For both the goal is to find an $\alpha, \alpha \in [0, R_o]$, which is within the optimality range R_N. That is we want to reduce the original α interval from R_o to R_N. Define β to be the ratio of R_N/R_o.

It is well-known that the "best" sequential search from the point of view of minimizing the number of function evaluations for a given β is the Fibonacci search (18). With this search procedure, the optimality or uncertainty range after N_s function evaluations (R_N), the original range of the α parameter (R_o), and the N_s^{th} Fibonacci number (F_{N_s}) are related as follows:

$$\beta = \frac{R_N}{R_o} = \frac{1}{F_{N_s}}$$

$$= \sqrt{5}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{N_s+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{N_s-1}\right]^{-1} \qquad (2)$$

The subscript S indicates that a sequential search is involved. For $N_s >> 1$, equation 2 can be simplified to the Golden Section search and N_s solved for explicitly.

$$N_s \simeq (.324 - \ln\beta)/.481 \qquad (3)$$

Thus given the parameters initial and final ranges the number of function evaluations required can be obtained.

The second type of search procedure to be considered is a parallel or simultaneous search. Here N_c function evaluations are produced in parallel, each one using a different α parameter value. Successive groups of function evaluations are performed over successively smaller parameter ranges, until the parameter is within the acceptable optimization range (R_o). A simple approach which is optimal in the Fibonacci or minimax sense [9, 10] consists of

successive divisions of R_o into N_c+1 intervals with function evaluations being performed at the N_c internal points. Using the unimodal property of the function involved, a new α interval containing the optimal α is selected at each iteration, with N_p iterations being necessary for a given β. If N_c is taken to be an even integer then the following equation holds:

$$\beta = \frac{R_N}{R_o} = \left[\frac{2}{N_c+2}\right]^{N_p} \tag{4}$$

Solving for the number of iterations results in equation 5.

$$N_p = \frac{\ln \beta}{\ln 2 - \ln(N_c+2)} \tag{5}$$

C. Computer Implementation and Optimization Strategy Selection

Two main strategies are to be compared. These relate to whether parallel or sequential function evaluation algorithms and optimization schemes are to be used. The two strategies are illustrated in Figure 1.

In Figure 1a, the N_c processors of the MIMD machine work together using a parallel algorithm for evaluating the function. A single function evaluation is thus obtained for a single parameter value. The results of this single evaluation are used by the sequential Fibonacci optimization scheme to select a successor parameter value, and the function is then evaluated again. This continues for N_s cycles until the parameter value is obtained which satisfies the given β requirements.

The multiple processors in this case speed up the optimization process by permitting the implementation of a parallel function evaluation algorithm. Thus speed of solution is directly related to the speedup curve, S_p vs N_c, which results from the implementation of the parallel algorithm on the particular MIMD machine used. For this case the solution or evaluation method is parallel, while the optimization method is sequential. The total time to perform the optimization ($T_{p/s}$) is given below.

$$T_{p/s} = N_s * [t_p(N_c) + t_s'] \tag{6}$$

$t_p(N_c)$ is the time it takes to perform the function evaluation with the parallel algorithm utilizing N_c processors. t_s' is the time it takes to perform calculations related to the Fibonacci search procedure. Substituting the value for N_s from (3), and noting that for most practical situations $t_p(N_c) >> t_s'$ one obtains:

$$T_{p/s} = (.324-\ln\beta)*t_p(N_c)/.481 \tag{7}$$

In terms of the speedup function Sp:

$$\frac{T_{p/s}}{t_p(1)} = \frac{.324-\ln\beta}{.481*S_p(N_c)} \tag{8}$$

The second approach is illustrated in Figure 1b. Here a sequential function evaluation scheme is used with the N_c processors simultaneously evaluating the function for a different value of α. The N_c values obtained are used by a parallel optimization scheme to select a new group of N_c parameters concentrated now over a smaller subinterval of the parameter range. The function is again evaluated with each processor now handling one of the new parameter values selected. This continues for N_p cycles until a parameter value is obtained which satisfies the given β requirements.

The multiple processors in this case speed up the optimization process by allowing for N_c separate simultaneous function evaluations. Contrasted with the previous method the evaluation method is sequential and the optimization scheme is parallel. The total time to perform the optimization ($T_{s/p}$) is:

$$T_{s/p} = N_p*[t_p(1) + t_p'] \tag{9}$$

$t_p(1)$ is the single processor time to evaluate the function, while t_p' is the time it takes to perform calculations related to the parallel search procedure. Now t_p' involves comparing at most N_c adjacent function values, to select the next optimal subinterval. $t_p(1)$ on the other hand may involve the time associated with solving a set of equations. Thus for most cases of interest $t_p(1) >> t_p'$, and substituting the value for N_p from (5), one obtains:

$$\frac{T_{s/p}}{t_p(1)} = \frac{\ln\beta}{\ln 2-\ln\,(N_c+2)} \tag{10}$$

The question to be resolved is which of these two strategies is better from the point of view of requiring less time to perform the optimization. The strategies will be referred to as the P/S (i.e., parallel evaluation/sequential optimization) and S/P (i.e., sequential evaluation/parallel optimization) strategies. The next section shows how their selection is related to the function and speedup curve, to the optimality requirements (β), and to the number of processors present (N_c).

III. OPTIMAL STRATEGIES

A. General Approach

For a given speedup function it is clear from [8] and [10] that the strategy selection process involves two principal variables, β and N_c. The selection process can be understood by examining the curve obtained when $T_{p/s}$ is equated to $T_{s/p}$ as given below.

$$\frac{\ln\beta}{\ln 2 - \ln(N_c+2)} = \frac{.324 - \ln\beta}{.481*S_p(N_c)} \tag{11}$$

This in turn can be solved explicitly for β. Thus:

$$\beta = \ln^{-1}\left[\frac{.324}{1+Q}\right] \tag{12}$$

where:

$$Q = \frac{.481*S_p(N_c)}{\ln 2 - \ln(N_c+2)}$$

The remainder of this paper examines how this curve segments the strategy space into regions where either $T_{p/s} \geq T_{s/p}$ or $T_{s/p} < T_{p/s}$. The shape of these regions will differ for differing speedup functions. The sections to follow consider three example speedup functions and just what optimization strategy to follow for these cases. The approach is straightforward and can be used when other speedup functions are encountered.

B. Logarithmic Speedup Functions

Certain parallel algorithms appear to have a logarithmic speedup as the number of processors available increases as given in formula below.

$$S_p = 1 + A*\ln N_c \tag{13}$$

A is a constant determined by the parallel algorithm used.

An example of this is given in reference 12. In this situation, function evaluation involves solution of a set of first order differential equations. One solution method considered, the Equation Segmentation method, is based on partitioning the set of differential equations and allocating equation subsets to the N_c available computers. Speedup is achieved by performing the derivative evaluations associated with each subset in parallel. Information is exchanged between computers at each integration step. A test of this method using a benchmark of six relatively small problems yielded average speedup

for N_c = 1, 2, 4 and 8, of S_p = 1, 1.74, 2.82 and 4.09. With these results, a least-squares fit to (13) yields an A value of 1.407. Similar logarithmic like speedup curves have been observed by Raskin [15] for certain systems of partial differential equations.

Obtaining the decision curve is a simple matter of substituting S_p from (13) into (12), and for a given value of A and successive values of N solving for β. The results for A = 1.407 are plotted in Figure 2 together with other values of A. As indicated, the decision curve descends monotonically, eventually crossing the horizontal axis at about 25 computers. The space is neatly segmented into two regions. To the left of the curve the P/S strategy should be used while to the right the S/P strategy should be used. For instance if 16 processors were available and a β of .001 was acceptable, then the best strategy would be to use a parallel function evaluation scheme with the sequential Fibonacci search algorithm (i.e., P/S strategy). On the other hand if the number of processors were raised to 32 the S/P strategy would be preferable. Note that since in most practical situations a β less than .05 will be sought, using 25 processors as a single decision point will yield near optimal strategies.

The reasons for the decision space segmentation as described above relate to the interaction of two items. The first is the relative efficiencies of the two search algorithms in terms of the total number of function evaluations performed for a given β. In these terms the Fibonacci algorithm is the most efficient procedure. The second is the shape of the speedup curve. When the number of processors is small, additional processors have a large effect on the speedup. This coupled with a more efficient search algorithm make the P/S strategy more effective for relatively small numbers of processors. As the number of processors gets larger, however, the diminishing returns effect of the speedup curve comes into play. Now increases in the number of processors has negligible effect on parallel function evaluation speed, thus the times associated with the P/S strategy stabilize and there is no gain associated with having more processors. On the other hand the S/P strategy is always able to utilize additional processors since each additional processor provides for an additional simultaneous function evaluation with an added parameter value. The interplay of these items results in the family of curves of Figure 2.

C. Linear Speedup Functions

Parallel algorithms with near linear speedup as a function of the number of processors are possible and have been reported in the literature. An example of this is reported in reference 13 and discussed further in reference 12. Given that $S_p = 1$ when $N_c = 1$, the linear speedup function follows:

$$S_p = 1 + m*(N_c-1) \tag{14}$$

where m is the slope of the linear function.

The decision curve is obtained by substituting S from (14) into (12), and for a given value of the slope m and values of N_c solving for β. The results for a sequence of m values are given in Figure 3. The m = .927 curve corresponds to a differential equations algorithm discussed (12).

Unlike the logarithmic case, the decision curve here monotonically increases with the number of computers. For any particular m value, to the right and below the decision curve implies that the P/S strategy should be followed, while to the left and above the curve indicates the S/P strategy would be preferable. For the case m = .927, any practical β value would result in a parallel solution method combined with Fibonacci search as the better strategy independent of the number of processors present. For lower values of m, however, the strategy would depend on N_c. For m = .258 and β = .001, for instance, if $N_c < 13$, the S/P strategy should be used with the P/S strategy being used otherwise.

The explanation of these curves again relates to the interplay between search algorithm efficiencies and speedup curve properties. Note first that for high values of the slope and a reasonable β range ($\beta < .1$), the P/S strategy would be selected independent of N_c. This is because the higher efficiency of the Fibonacci algorithm combines with an effective use of processors for parallel function evaluation (i.e., high m) to reinforce each other. The result is that the P/S strategy is the better one.

The situation changes as the slope becomes low. Here processors are used less effectively in parallel function evaluations with different parameter values. At some point therefore a region is introduced where the S/P strategy is the advantageous one. For such low m values, however, the S/P strategy will give way to the P/S strategy as N_c increases. The reason for this is that the slope, though relatively small, results in a linearly increasing speedup with

higher N_c and is therefore linearly effecting solution speed in the P/S method. On the other hand, increases in N_c with the S/P strategy effect solution speed only in a logarithmic manner.

D. Composite Speedup Functions

The final speedup curve of interest is a combination of the logarithmic and linear curves. Consider a situation where two parallel solution methods are available, one having a logarithmic, and the other having linear speedup functions. In certain situations it may be possible to combine the two methods to achieve a further enhanced parallel solution method. Such a method would take advantage of the rapid logarithmic rise in speedup which occurs with a relatively small number of processors. However, instead of being subject to asymptotic behavior of the logarthmic curve, it would maintain a linear speedup as the number of processors gets large. Such a composite method, for instance, might be used to solve large sets of differential equations. Initially the set of equations would be partitioned into tightly coupled subsets representing perhaps naturally close physical subsystems. These subsets of equations would then be allocated to groups of computers where, within the group, a linear speedup solution method would be utilized.

Given that the total number of computers available is N_c, let N_c^* be the number of problem partitions, or computer groups. For simplicity assume that each group consists of an equal number of processors. This assumption will generally hold if the partitioning process results in an equivalent amount of computational work to be performed by each processor group.

If S_{lo} and S_{li} are the logarithmic and linear speedup functions, respectively, then the overall speedup curve for the composite method is defined by:

$$S_p = S_{lo}(N_c^*) * S_{li}\ (\lfloor N_c/N_c^* \rfloor)$$

or

$$S_p = (1+A \ln N_c^*) * (1+m(\lfloor N_c/N_c^* \rfloor - 1)) \qquad (15)$$

The brackets $\lfloor\ \rfloor$ indicate that an integer number of computers must be present in each computer group. Notice that the number of computer groups, N_c^*, must still be specified. Since P/S strategy performance will improve as S_p increases, N_c^* must be chosen to maximize S_p for given values of A, m and N_c.

Once N_c^* has been calculated, the optimized S_p can be used to obtain the decision curve by substituting into equation (12). The results are plotted in Figure 4. The general shape of the curves corresponds to what one might expect from a superposition of the logarithmic and linear decision curves of Figures 2 and 3. For small values of N_c the logarithmic speedup dominates, while for large values of N_c the linear speedup dominates. One interesting feature that results is that for values of m below a certain cutoff (i.e., $m < .2$ with $A = 1.407$). The decision space is partitioned into three regions. Thus for $m = .100$, $A = 1.407$ and $\beta = .1$, below ten processors and above sixty four processors the P/S strategy is best, while between these two processor limits the S/P strategy should be used. The explanation for this behavior directly follows the previous discussion under logarithmic and linear speedup functions.

IV. SUMMARY AND CONCLUSIONS

This paper presented a straightforward approach to determing how best to utilize an MIMD multiprocessor system in the solution of one dimensional optimization problems involving continuous unimodal functions and nongradient search techniques. A methodology was presented which allows one to consider a variety of speedup functions which may occur in parallel function and systems evaluation. The best of two optimization strategies can be simply determined for a given accuracy, number of processors and speedup function. The first strategy involved parallel evaluation of the function to be optimized, and a sequential Fibonacci optimization scheme (P/S). The second involved sequential evaluation of the function to be optimized, and a parallel optimization scheme (S/P). Graphs were provided so that the best of the two strategies could be selected for several common speedup functions. Further work along these lines with multidimensional problems and gradient search techniques appears possible but difficult.

Other problems exist which are directly analogous to the one considered. One relates to DO LOOP implementation on MIMD machines where each iteration through the loop corresponds to an increase in some performance measure. The loop can be implemented either by replication on the available processors, or by parallel evaluation of the tasks contained in the loop. An example of this is the class of statistical problems where multiple runs are necessary to achieve a given level of statistical convergence and the function or system involved can be solving parallel fashion. In these situations a decision similar to the P/S , S/P one discussed above can be examined.

13. Miranker, W. L. and Liniger, W. M.
"Parallel methods for the numerical integration of ordinary differential equations"
Math. Comput., 21 (1967), 303-320

14. Heller, D.
"A Survey of Parallel Algorithms in Numerical Linear Algebra"
Carnegie-Mellon Univ. Tech. Report; NTIS # AD-A024 792

15. Raskin, L.
"Performance Evaluation of Multiple Processor Systems"
Ph.D. Thesis, Dept. of Computer Science, Carnegie-Mellon Univ. (Aug. 1978)

16. Baudet, G. M.
"The Design and Analysis of Algorithms for Asynchronous Multiprocessors"
Ph.D. Thesis, Dept. of Computer Science, Carnegie-Mellon Univ. (April 1978)

17. Miranker, W. L.
"Parallel methods for solving equations"
Math. and Comp. in Simulation, XX, 2 (June 1978), 93-101

18. Kiefer, J.
"Sequential Minimax Search for a Maximum"
Proc. Am. Math. Soc., 4 (1953), 502-506

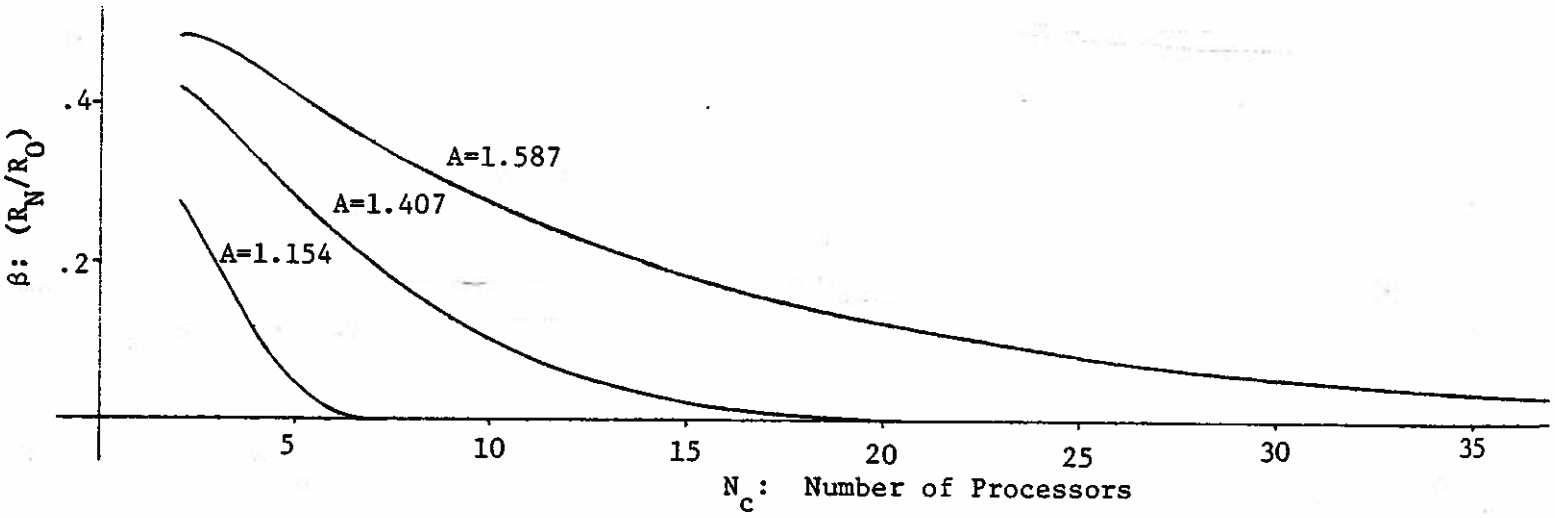


Fig. 2: Decision Space for Logarithmic Speedup

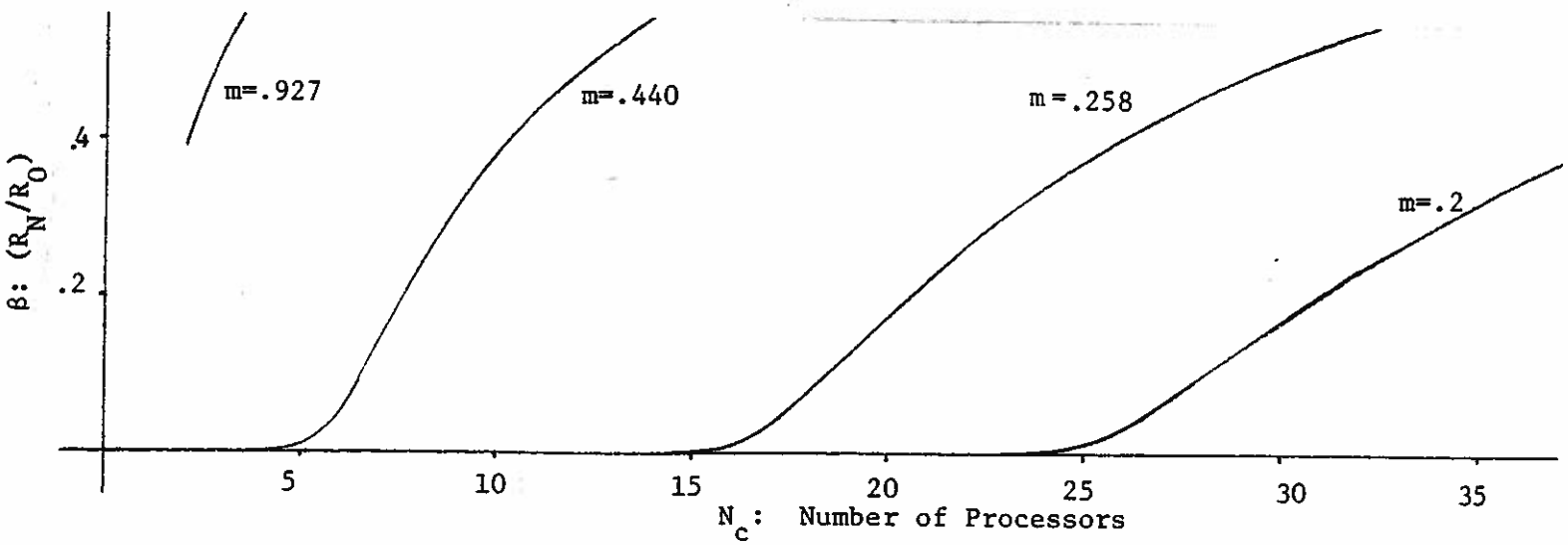


Fig. 3: Decision Space for Linear Speedup

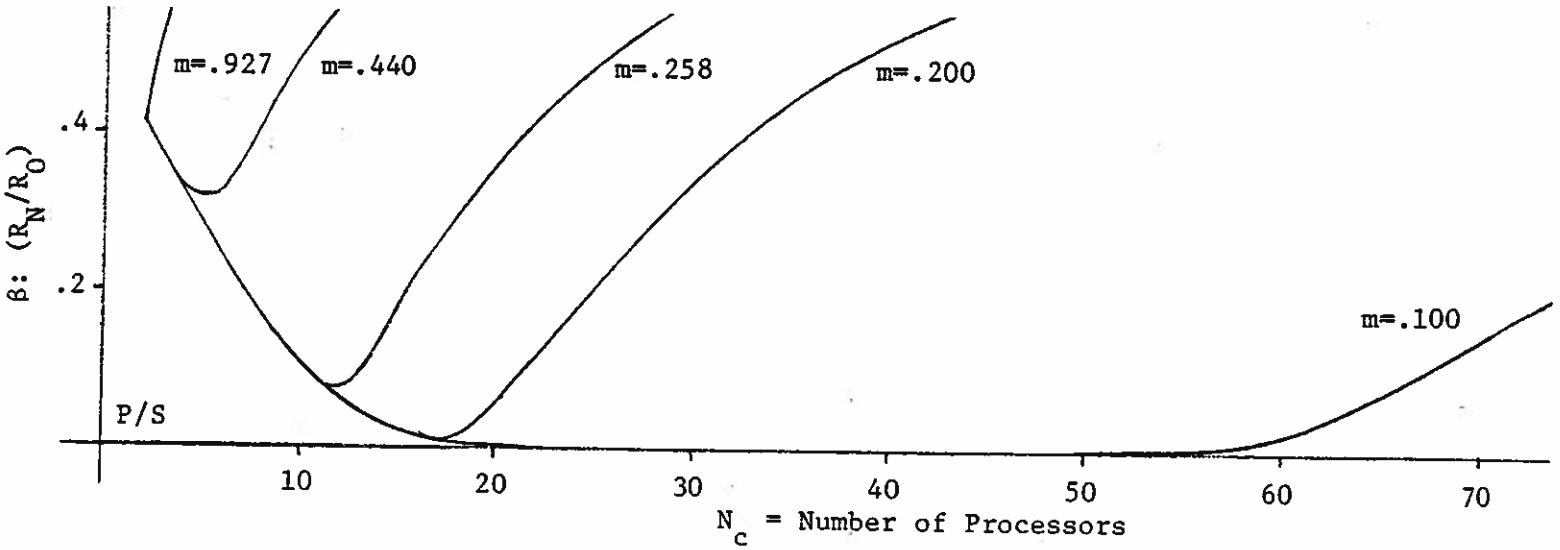


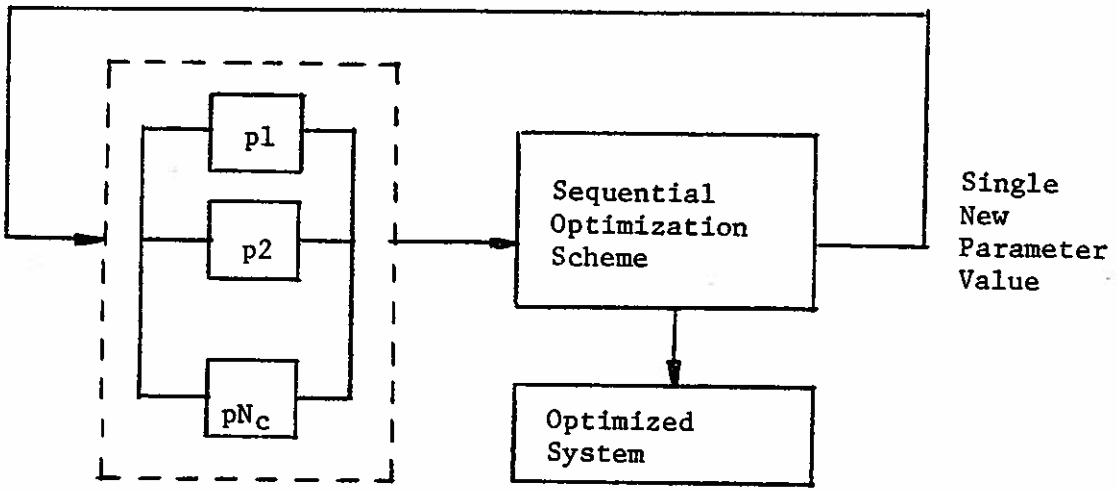
Fig. 4: Decision Space for Composite Speedup A=1.407

Figure 1: Alternative Parallel Optimization Schemes

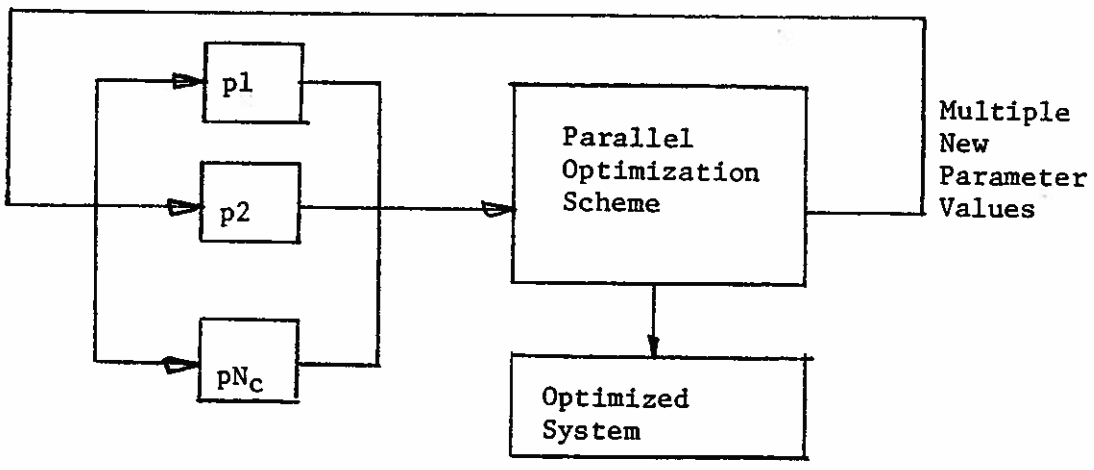
Figure 2: Decision Space for Logarithmic Speedup

Figure 3: Decision Space for Linear Speedup

Figure 4: Decision Space for Composite Speedup A=1.407



(a) Parallel processors work together to get single solution using a single parameter value.



(b) Parallel processors work separately, each obtaining a single solution with differing values.

Figure 1: Alternative Parallel Optimization Schemes.

References

1. Poole, W. G. Jr. and Voigt, R. G.
"Bibliography 35-Numerical Algorithms For Parallel and Vector Computers"
ACM Computing Reviews, 15, 10 (Oct. 1974), 379-388

2. Grooms, D. W. (Ed.)
"Parallel Processors: A Bibliography With Abstracts - Period 1/1964 - 1/1976"
National Tech. Info. Service (NTIS) #PS-76/0022 (Jan. 1976)

3. Kuck, D. J.
"A Survey of Parallel Machine Organization and Programming"
ACM Computing Surveys, 9, 1 (March 1977), 29-59

4. Flynn, M. J.
"Very High Speed Computing Systems"
Proc. of the IEEE, 54, 12 (Dec. 1966), 1901-1909

5. Anderson, G. A. and Jensen, E. D.
"Computer Interconnection Structures: Taxonomy, Characteristics and Examples"
ACM Computing Surveys, 7, 4 (Dec. 1976), 197-212

6. Hwang, K. and Yao, S. B.
"Optimal Batched Searching of Tree Structured Files in Multiprocessor Computer Systems"
JACM, 24, 3 (July 1977), 441-454

7. Weller, D. L.; Davidson, E. S. and Feng, J.
"Optimal Searching Algorithms For Parallel-Pipelined Computers"
in Parallel Processing, Springer-Verlag (1975), 291-305

8. Berlin, V. G.
"Parallel Randomized Search Strategies"
Autom, and Remote Control, 33, 3 (March 1972), 398-403

9. Auriel, M. and Wilde, D. J.
"Optimal search for a maximum with sequences of simultaneous function evaluations"
Manage. Sci., 12 (1966), 722-731

10. Karp, R. M. and Miranker, W. L.
"Parallel Minimax Search for a Maximum"
J. Combinatorial Theory, 4 (1968), 19-35

11. Chazan, D. and Miranker, W. L.
"A nongradient and parallel algorithm for unconstrained minimization"
Siam J. Control, 8 (1970), 207-217

12 Franklin, M. A.
"Parallel Solution of Ordinary Differential Equations"
IEEE Trans. on Computers, C-27, 5 (May 1978), 413-420