SU-SEL 77-002

PERFORMANCE BOUNDS FOR PARALLEL PROCESSORS

by

Ruby Bei-Loh Lee

Technical Report No. 125

November 1976

DIGITAL SYSTEMS LABORATORY

STANFORD ELECTRONICS LABORATORIES

STANFORD UNIVERSITY • STANFORD, CALIFORNIA

SU-SEL 77-002

PERFORMANCE BOUNDS FOR PARALLEL PROCESSORS

by

Ruby Bei-Loh Lee

November 1976

Technical Report No. 125

Digital Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305

The work described herein was supported by the Joint Services Electronics Program under Contract No. N00014-75-0601.

SU-SEL 77-002

Digital Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305

Technical Report No. 125

November 1976

PERFORMANCE BOUNDS FOR PARALLEL PROCESSORS

by
Ruby Bei-Loh Lee

ABSTRACT

A general model of computation on a p-parallel processor is proposed, distinguishing clearly between the logical parallelism (p^* processes) inherent in a computation, and the physical parallelism (p processors) available in the computer organization. This shows the dependence of performance bounds on both the computation being executed and the computer architecture. We formally derive necessary and sufficient conditions for the maximum attainable speedup of a p-parallel processor over a uniprocessor to be $Sp \leq \min(\frac{p}{\ln p}, \frac{p^*}{\ln p^*})$, where $\ln p$ approximates H_p, the pth. harmonic number. We also verify that empirically-derived speedups are $O(\frac{p^*}{\ln p^*})$. Finally, we discuss related performance measures of minimum exeuction time, maximum efficiency and minimum space-time product.

The work described herein was supported by the Joint Services Electronics Program under Contract No. N00014-75-C-0601.

TABLE OF CONTENTS

1. INTRODUCTION

The purpose of this paper is toestimate the maximum attainable speedup of a computation, executing on a computer organization, within a given technology. The questions we ask are: What is the minimum execution time, and hence the maximum speedup, of a given computation, assuming unlimited computer resources? Then, given a particular type of computer organization, we ask what aspects of a computation affect the raw speed of this computer and to what extent?

The computer organization we wish to consider is one where p identical processors operate in parallel on different instructions. This belongs to the general class of MIMD (Multiple Instruction Multiple Data) organizations [l]. We wish to study the speedup attainable by varying the number, p, of processors used. The basis of comparison is always that of the uniprocessor where p=l. Since the parameter of concern is the number of processors, p, we will assume that there are unlimited supplies of memories and input-output devices so that the computation is always processor-bound, with no delays due to memory faults and interference, or device interrupts.

There are many motivations for considering the parallel processor organization. The first is that it has immediate practical appeal. The rapidly decreasing cost of LSI microprocessors makes it economically feasible to consider using a whole army of processors within the computer architecture to speedup a computation, even at reduced efficiency of each component processor. No longer is the processor the hallowed CPU (Central Processing Unit), or the most valuable resource which has to be utilized with the greatest efficiency. Of course, an accpetable level of efficiency has to be obtained, but more importantly, we need to find out what sort of speedup is in fact possible by increasing the number of processors, even if we ignore the problems of control and communication which must accompany the cooperation and competition between these processors. This brings us to our second basic motivation for studying the parallel processor organization.

Controversial Views

In fact, quite a lot of controversy and "folklore" has built up around the issue of speedup bounds for parallel processors. It is clear that the nature of the computation will limit the maximum performance of the parallel processors, but to what extent?

Amdahl [8] suggested that the amount of strictly serial operations inherent in a computation point to a uniprocessor approach to computing, if some "acceptable" level of efficiency is to be achieved. The model that he used, however, was very simple where the number of processors that could be used at any time was either 1 or p. In this paper, we take the more general approach where any number of processors between 1 and the maximum number p, may be used simultaneously.

Another view, which has come to be known as "Minsky's Conjecture", suggests that the speedup is proprotional to $\log_2 p$ in most cases [6,7]. Flynn [1] has supported this view for SIMD (Single Instruction Multiple Data) organizations, and has proposed an explanation for it based on a special kind of nested branching degradation in the program. It is not clear whether such kind of branching degradation does in fact occur in programs, and if so, how common this is. Perhaps what is more important is that we must somehow specify, in terms of the parameters of a general model of computation, the conditions under which certain speedup bounds are the maximum (or minimum, or average) attainable. Then, by empirical observations of program behavior, we can see if such conditions are indeed met. This is the approach that we take, in this paper, to the problem of finding speedup bounds for parallel processors.

If we do find that, "in all but a finite number of exceptions", the speedup is proportional to $\log_2 p$, then the parallel processor organization is obviously not a very effective speedup mechanism. For example, to achieve an order of magnitude speed improvement, 3 orders of magnitude more processors have to be used, with an efficiency of only 1% of the uniprocessor. Fortunately, the rather extensive empirical experiments of Kuck, et al [3,4,5], show that the attainable speedup is almost always better than $\log_2 p$.

Kuck, et al [3,4,5,] have written a fairly sophisticated program analyser which accepts, as input, an ordinary program written for execution on a uniprocessor, and turns it into a program suitable for execution on a system with p^* parallel processors, where p^* is the maximum number of processors which can be simultaneously used in the converted program. The point, of course,is to minimize the execution time of the converted program, since the maximum speedup is inversely proportional to the minimum execution time. Based on experiments using this analyser, Kuck has proposed the following observations [3]:

"For many ordinary Fortran programs (with $T_1 \leq 10{,}000$), we can find p such that

(1) $T_p = \alpha \log_2 T_1$, for $2 \leq \alpha \leq 10$

and (2) $p \leq \frac{T_1}{.6 \log_2 T_1}$

such that

(3) $S_p \geq \frac{T_1}{10 \log_2 T_1}$ and $E_p > .3$ "

Here T_1 is the time taken by the uniprocessor, T_p is the time taken by p parallel processors, S_p is the speedup and E_p is the efficiency.

We notice three points: he has chosen to express speedup in terms of T_1, the time taken by a uniprocessor, rather than p, the number of processors used. This is because he almost always uses a system with p^* parallel processors, where p^* is the maximum number of processors that can be simultaneously used, according to the result of his program analyser.

The second point of difference is that he considers the lower bound for speedup, whereas in this paper, we are interested in the upper bound.

The third point is that he has not given any theoretical proof of his empirically-derived formula.

However, his experimental results form the main source of raw data, with which one may compare any speedup bounds obtained by non-empirical methods, like the mathematical derivations used in this paper. Also, we have used essentially the same definitions (see next section) of the performance measures of T_p, S_p and E_p. It was indeed gratifying to discover that the upper bound for speedup, $S_p < \frac{p}{\ln p}$, that we found by mathematical observations agreed very well with Kuck's experimental results.

Footnote: ln is the natural logarithm (base e) function, differing from the logarithm to any other base, by only a constant, since $\log_a x = (\frac{1}{\log_b a}) \cdot \log_b x$

2. PROBLEM DEFINITION

The problem is to find the bounds on the performance improvement of a p-parallel processor over a uniprocessor, for a given computation.

A Model of Computation on a P-Parallel Processor

A computation is a sequence of steps. At each step s, a finite number, k, of instructions may be simultaneously executed. Step s is then said to contain k parallel processes. The relevant parameters for any given computation are:

(i) p^* : the maximum number of parallel processes contained in any step of the computation

(ii) $\{r_i, 1 \le i \le p^*\}$: The probability that i parallel processes are contained in a single step, $\sum_{i=1}^{p^*} r_i = 1$.

(iii) Tl : the time taken to execute the whole computation by a uniprocessor (equivalent to the total number of instructions executed).

A p-parallel processor is a computer organization with p identical processors, each of which is capable of executing one instruction (not necessarily the same type of instruction) per time-unit. A time-unit is defined to be the amount of time taken by a uniprocessor (or l-parallel processor) to execute one instruction, and each instruction is assumed to take the same amount of time for execution. Any number of processors, from 1 to p, may execute simultaneously in a time-unit. The relevant parameters for any given p-parallel processor are:

(i) p : the maximum number of parallel processors in the computer organization

(ii) $\{q_i, 1 \le i \le p\}$: the probability that i parallel processors are simultaneously used in a time-unit, $\sum_{i=1}^{p} q_i = 1$.

The execution of a computation on a p-parallel processor consists of a mapping from steps in the computation to time-units in the computer organization,

and a corresponding mapping of the probabilities $\{r_i, 1 \leq i \leq p^*\}$ inherent in the computation to $\{q_i, 1 \leq i \leq p\}$ for the p-parallel processor. If $p \geq p^*$, the mapping is clearly:

$$q_i = \begin{cases} r_i, & 1 \leq i \leq p^* \\ 0, & p^*+1 \leq i \leq p \end{cases}$$

Hence, for $p \geq p^*$, the number of time-units taken for execution is equal to the number of steps originally present in the computation, and the computation is said to be executing at maximum speed.

If $p < p^*$, the mapping is not unique, and we will assume that the "optimal mapping" has been used, i.e., the resulting values of q_i correspond to the minimum execution time on the given p-parallel processor.

Definition of Performance Measures

(1) T_p is the number of time-units taken by a p-parallel processor to execute a given computation.

(2) S_p is the speedup of the p-parallel processor over the uniprocessor, for the same computation:

$$S_p = \frac{T_1}{T_p} = \frac{\text{execution time on uniprocessor}}{\text{execution time on p-parallel processor}}$$

(3) E_p is the efficiency of the p-parallel processor

$$E_p = \frac{S_p}{p}$$

(E_p compares the actual execution bandwidth, S_p, to the maximum possible bandwidth, p).

(4) ST_p is the space-time product for executing a given computation on a p-parallel processor:

$$ST_p = pT_p$$

(The relative space-time product, $\frac{ST_p}{ST_1}$, is inversely proportional to the efficiency, E_p, and can be said to measure the "cost" of using the p-parallel processor compared with a uniprocessor.)

Immediate Results

Fact 2.1: The execution time taken by a p-parallel processor, for a given computation, is

$$T_p \geq T_1 \sum_{i=1}^{p} \left(\frac{q_i}{i}\right)$$

Fact 2.2: The speedup of a p-parallel processor, for a given computation, is

$$S_p \leq \frac{1}{\sum_{i=1}^{p} \left(\frac{q_i}{i}\right)}$$

Fact 2.3: For any given computation, the absolute lower bound on the execution time is

$$T_{p^*} \geq T_1 \sum_{i=1}^{p^*} \left(\frac{r_i}{i}\right)$$

Fact 2.4: For any given computation, the absolute upper bound on the speedup attainable is

$$S_{p^*} \leq \frac{1}{\sum_{i=1}^{p^*} \left(\frac{r_i}{i}\right)}$$

Fact 2.5: $1 \leq S_p \leq \min(p, p^*)$

The above results are stated without proof since they are derived directly from the definitions.

We note that Facts 2.1 and 2.2 are architecture-dependent bounds, whereas Facts 2.3 and 2.4 are computation-dependent bounds. The inequalities in Facts 2.1 to 2.4 may be replaced by equalities, if we do not insist that T_p be expressed as an integral number of time-units. Also, in Facts 2.1 and 2.2, we can implicitly take care of the probability that all processors are idle if we intepret q_1 as the probability that 0 or 1 processors are used in a time-unit.

Fact 2.5 illustrates that, in general, the performance is limited by both the architecture and the computation being executed.

3. DERIVATION OF TIGHTER UPPER BOUND FOR SPEEDUP

Typical Speedup Ratios

It seems clear that the nature of the computation (or "program behavior") will limit the actual speedup obtainable by using a p-parallel processor. It is therefore instructive to review some of the best speedup ratios obtained in the past, for various specialized types of computations. Such typical speed-up ratios are on the order of [9-13,3]:

(i) $k_1 p$: matrix computations

(ii) $k_2 \frac{p}{\log p}$: sorting
tridiagonal linear systems
linear recurrence relations
polynomial evaluation
evaluation of arithmetic expressions without division

(iii) $k_3 \log p$: searching ordered list (actually $\log_2(p+1)$)

(iv) k_4 (independent of p) : some nonlinear recurrence relations
some compiler routines

In each case, the k_i are some constants, and the p often refer to the maximum number of logical parallel processes within the computation, rather than to the maximum number of physical parallel processors actually present in the computer system. Hopefully, we have eliminated this kind of confusion in our model, by using p^* for the maximum number of processes, and p for the maximum number of processors.

We note the following points:

(1) The types of computations which have a linear speedup, proportional to $k_1 p$ for $k_1 \leq 1$, are rather rare. They tend to have large amounts of inherent iterative structure, acting on disjoint domains.

(2) Since it is clear from Fact 2.5 that in fact $S_p \leq p$ for all computations, the upper bound of $k_1 p$ does not give us any more information.

(3) That the speedup of computations can be independent of p is also clear from Fact 2.5 ($S_p \leq p^*$) and Fact 2.2, where

$$S_p \leq \frac{1}{\sum_{i=1}^{p} \frac{q_i}{i}} \leq \frac{1}{q_1} \quad \text{, since each term is non-negative.}$$

It is interesting to note that the speedup is limited by the reciprocal of the proportion of time that at most 1 processor is used, i.e., by the inherent serial nature of the computation. This is essentially similar to Amdahl's argument in favor of uniprocessors, and also agrees well with empirical data from various sources. For example Flynn [2] cites statistics that 16.5% of the instructions executed in "General Technical" type computations are conditional (nonresolvable) branch instructions. Also, studies on the Atlas machine in England [22] indicate that conditional branch instructions form about 10% of all instructions executed. Since it is often hypothesized that conditional branches introduce inherent serialism into computations, we could say that empirical data suggest that $q_1 \geq 0.1$, agreeing with Amdahl's "private statistics". Hence, this implies that $S_p \leq 10$. In fact, Kuck's [3] experimental results for the value of S_p, averaged over a large class of different types of computations is $S_p \sim 9.8$. Fortunately, however, the variance is large, and many individual types of computations have speedups greater than this.

This processor-independent upper bound of $S_p \leq \frac{1}{q_1}$ is even more sobering when we consider that ql includes not only the probability that 1 processor is used, but also the probability that all processors are idle. Hence, if the system's resources are not well-balanced, e.g., delays due to memory faults dominate the execution time, then the probability that all processors will be idle will be very high, jacking up the value of q_1, irregardless of the number, p, of physical parallel processors. Hence $S_p \sim O(\frac{1}{q_1})$, for all p, a most gloomy proposition. Intuitively, this is clear:

Observation 3.1:

(i) $S_p \leq \frac{1}{q_1}$, for all p. If q_1 is large, then $S_p \sim \frac{1}{q_1} \sim 1$, since $0 < q_1 \leq 1$, and increasing the number of processors will not speed up the computation.

(ii) q_1 will be large if the computation is not processor bound and/or the computation is highly serial in nature.

(4) On the more optimistic side, we note that many types of computations can achieve higher speedups. Of the two remaining typical speedup

ratios, $O(\frac{p}{\log p})$ is achieved by more types of computations than the lower value of $O(\log p)$. This suggests that we examine the bound $\frac{p}{\log p}$ more closely.

Intuitive Motivation of the Expression $S_p < \frac{p}{\ln p}$

It is clear from Fact 2.5 that all computations executing on all p-parallel processors, have a maximum speedup of $O(p)$.

It is our intent to try to extablish the next tighter bound $O(\frac{p}{\log p})$, and show the conditions under which this is the maximum attainable speedup. We will first try to motivate the discussion by giving the following graphical or physical interpretation to the crucial expression, $\sum_{i=1}^{p} \frac{q_i}{i}$, in both the T_p and S_p bounds (see Facts 2.1 to 2.4).

Let $f(i)= \frac{1}{i}$, for $i=1,2,...,p$. Then since q_i are probabilities which sum to 1, the expression $\sum_{i=1}^{p} \frac{q_i}{i}$ is clearly the weighted average, or mean, of the function. The graph of f(i) versus i is plotted in Figure 1, for $p=6$.

If we give each value of f(i) equal weight, then $q_1=q_2=...=\frac{1}{p}$ and $\sum_{i=1}^{p} \frac{q_i}{i} = \frac{1}{p}\left(\sum_{i=1}^{p} \frac{1}{i}\right) = \frac{H_p}{p}$, where H_p is the pth. Harmonic number, defined as follows:

Definition 3.1 : $H_p = \sum_{i=1}^{p} \frac{1}{i}$, called the pth harmonic number.

The expression $\frac{H_p}{p}$ will be fundamental in establishing our speedup bound. It is "well-known" in mathematical circles (see e.g., [21]) that

Fact 3.1: $H_p = \ln p + \gamma + \frac{1}{2p} - \frac{1}{12n^2} + \frac{1}{120n^4} - \varepsilon$

where $\gamma = 0.57721...$, called "Euler's constant"

$0 < \varepsilon < \frac{1}{252n^6}$

Hence, $H_p = \ln p + \gamma + O(1)$

$> \ln p$

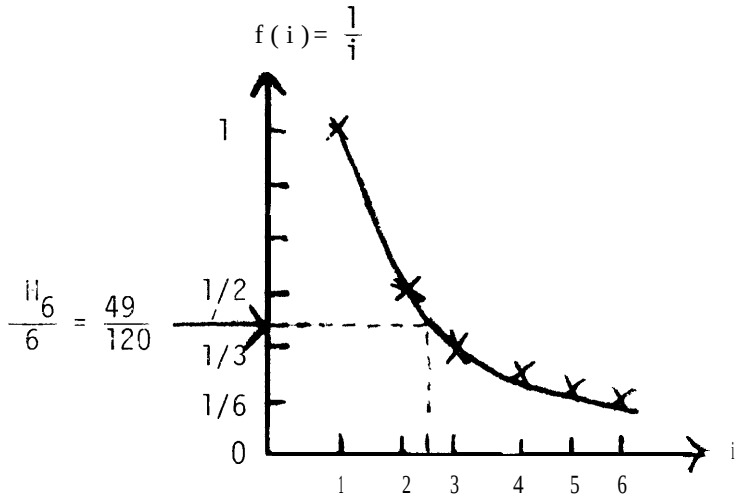

Figure 1: GRAPH OF $f(i)= \frac{1}{i}$ VERSUS i

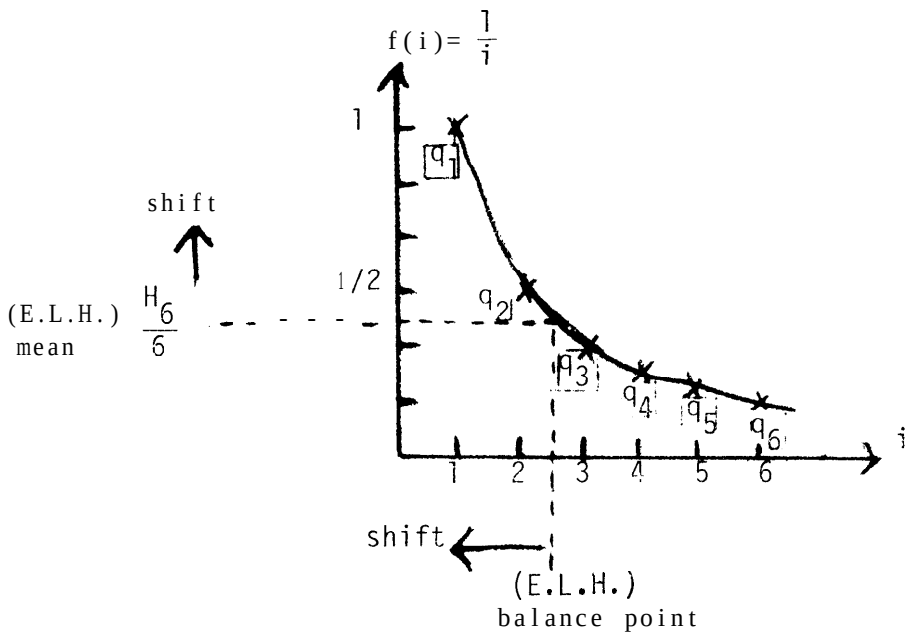

Figure 2: GRAPH OF $f(i)= \frac{1}{i}$ VERSUS i, WITH WEIGHTS q_i

dote : (1) $\sum_{i=1}^{6} q_i = 1$

(2) In the Equally-Likely Hypothesis (E.L.H.), we have

$q_1=q_2=\ldots=q_6=\frac{1}{6}$, with mean = $\frac{H_6}{6}$ and "balance-point" = $\frac{5}{H_6}$ as shown.

Next, we note that f(i) is a strictly decreasing function, so that if we assign the "weights" q_i such that the smaller values of i get larger values of q_i, then the "balance-point" of the curve f(i) will shift to the left, and the mean of f(i), viz, $\sum_{i=1}^{p}\left(\frac{q_i}{i}\right)$, will be greater than the equally-weighted case, $\frac{H_p}{p}$. (The reverse situation causes $\sum_{i=1}^{p}\left(\frac{q_i}{i}\right) \leq \frac{H_p}{p}$). See Figure 2.

In terms of the p-parallel processor, "assigning larger values of q_i to smaller values of i" simply means that "the probability that a smaller number of processors is used in a time-unit is greater than the probability that a larger number of processors is used". This seems to be a very general condition that is usually satisfied in practice. We will formalise these ideas in the next section.

Theorectical Derivation

In Lemma 3.1, we consider the "Equally-Likely Hypothesis" to derive $S_p < \frac{p}{\ln p}$. The fundamental theorem (theorem 3.1) then specifies precisely, the necessary and sufficient conditions on the computation for the upper bound $S_p < \frac{tpo}{\ln p}$ hold.

In corollaries 3.1 and 3.2, we show some less general, but sufficient conditions for the upper bound to hold. These sufficient conditions are easier to check for in a given computation, and lend more easily to physical interpretation.

In corollary 3.3, we show that the upper bound for S_p need not necessarily decrease when we use less physical processors, p, than the maximum number of logical processes, p*, present in the computation. This is due to the probability distribution of the actual number of processors used, as indicated by the necessary and sufficient conditions given in Theorem 3.1.

Finally, in corollaries 3.4 to 3.6 we summarize our results, and plot them in Figures 5 to 7.

Lemma 3.1 If $q_1=q_2=\ldots=q_p$ and $\sum_{i=1}^{p} q_i = 1$, then $S_p \leq \frac{p}{H_p} < \frac{p}{\ln p}$

Proof: $S_p \leq \frac{1}{\sum_{i=1}^{p} \frac{q_i}{i}}$ by Fact 2.2

$= \frac{1}{\frac{1}{p} \sum_{i=1}^{p} \frac{1}{i}}$ since $q_1=q_2=\ldots=q_p$ and $\sum_{i=1}^{p} q_i=1$

$= \frac{p}{H_p}$ by definition of H_p

$< \frac{p}{\ln p}$ since $H_p > \ln p$, by Fact 3.1 □

Theorem 3.1 (Necessary and Sufficient Conditions for $S_p < \frac{p}{\ln p}$)

$$S_p \leq \frac{p}{H_p} < \frac{p}{\ln p} \quad \text{iff} \quad \sum_{i=1}^{p} \frac{q_i - \frac{1}{p}}{i} \geq 0, \quad \sum_{i=1}^{p} q_i = 1$$

Proof: $S_p \leq \frac{p}{H_p} \rightarrow \frac{1}{\sum_{i=1}^{p} \frac{q_i}{i}} \leq \frac{p}{H_p} \rightarrow \sum_{i=1}^{p} \frac{q_i}{i} \geq \frac{H_p}{p}$

$$\rightarrow \sum_{i=1}^{p} \frac{q_i - \frac{1}{p}}{i} \geq 0$$

$$\sum_{i=1}^{p} \frac{q_i - \frac{1}{p}}{i} \geq 0 \rightarrow \sum_{i=1}^{p} \frac{q_i}{i} \geq \frac{H_p}{p} \rightarrow S_p \leq \frac{1}{\sum_{i=1}^{p} \frac{q_i}{i}} \leq \frac{p}{H_p}$$

That $\frac{p}{H_p} < \frac{p}{\ln p}$ follows from Fact 3.1 □

Hence, we have shown that under certain general circumstances, the maximum speedup for a p-parallel processor is less than $\frac{p}{\ln p}$, and we have found necessary and sufficient conditions for this to be true.

The condition $\sum_{i=1}^{p} \frac{q_i - \frac{1}{p}}{i} \geq 0$, may be interpreted as follows:

Let $\varepsilon_i = q_i - \frac{1}{p}$

Let $A = \sum_{\varepsilon_i > 0} \frac{\varepsilon_i}{i}$, $B = \sum_{\varepsilon_i < 0} \frac{\varepsilon_i}{i}$

Then $A \geq B$

In words, this means that for all those i, where the probability of using i processors is greater than $\frac{1}{p}$, then the sum of these diferences (ε_i), weighted by $\frac{1}{i}$, is the term A. B is the sum of the differences (ε_i), weighted by $\frac{1}{i}$, where the probability of using i processors is less than $\frac{1}{p}$. The condition requires the positive sum, A, to be greater than the negative sum, B.

For example, since the function $f(i)=\frac{1}{i}$, $i=1,2,\ldots,p$, is a strictly decreasing function, then if the q_i are also non-increasing, i.e., $q_1 \geq q_2 \geq \ldots \geq q_p$, then clearly the above condition is satisfied, and $S_p < \frac{p}{\ln p}$.

This is stated in the following corollary:

Corollary 3.1 If $q_1 \geq q_2 \geq \ldots \geq q_p$ and $\sum_{i=1}^{p} q_i = 1$, then $S_p < \frac{p}{\ln p}$

Proof: If $q_1 \geq q_2 \geq \ldots \geq q_p$ then $\exists k$ such that

$$\begin{cases} q_i \geq \frac{1}{p} & \text{for } 1 \leq i \leq k \\ q_i < \frac{1}{p} & \text{for } k+1 \leq i \leq p \end{cases}$$

Also, since $\sum_{i=1}^{p} q_i = 1$, we have $\sum_{i=1}^{k} (q_i - \frac{1}{p}) = -\sum_{i=k+1}^{p} (q_i - \frac{1}{p}) \geq 0$

$$\therefore \sum_{i=1}^{p} \frac{q_i - \frac{1}{p}}{i} = \sum_{i=1}^{k} \frac{q_i - \frac{1}{p}}{i} + \sum_{i=k+1}^{p} \frac{q_i - \frac{1}{p}}{i}$$

$$\geq \left(\sum_{i=1}^{k} q_i - \frac{1}{p} \right) \cdot \left[\frac{1}{k} - \frac{1}{k+1} \right]$$

$$\geq \left(\sum_{i=1}^{k} q_i - \frac{1}{p} \right) \cdot \left[\frac{1}{k(k+1)} \right]$$

≥ 0, since each term is positive. □

We note that Lemma 3.1, or the equally-likely hypothesis is just the special case of Corollary 3.1, when $q_1 = q_2 = \ldots = q_p$.

From the proof of Corollary 3.1, it is clear that we do not need the qi to be decreasing with increasing i. All that is required is that there is a k for which the values of q_i, for $i \leq k$, are greater than or equal to $\frac{1}{p}$, and the values of q_i, for $i > k$, are less than $\frac{1}{p}$. This leads to the

following corollary which gives a sufficient condition for $S_p < \frac{p}{\ln p}$ that is more general than that of $q_1 \geq q_2 \geq \ldots \geq q_p$.

Corollary 3.2 If $\exists k$ such that $\begin{cases} q_i \geq \frac{1}{p}, \text{ for } 1 \leq i \leq k, \text{ and} \\ q_i < \frac{1}{p}, \text{ for } k+1 \leq i \leq p \end{cases}$

$$\sum_{i=1}^{p} q_i = 1, \text{ then } S_p < \frac{p}{\ln p}$$

Proof: same as for corollary 3.1

Actually, the tightest upper bound that we have derived in Theorem 3.1 is that $S_p \leq \frac{p}{H_p}$ iff $\sum_{i=1}^{p} \frac{q_i - \frac{1}{p}}{p} \geq 0$. Also we know from Fact 2.4 that the maximum speedup attainable for a given computation (assuming unlimited processors, i.e., $p \geq p^*$) is $S_{p^*} \leq \frac{p^*}{H_{p^*}}$. For $p < p^*$, we intuitively expect a decrease in the speedup. However, we can show that, in fact, the upper bound of the speedup for $p < p^*$ need not necessarily be decreased.

Corollary 3.3 $\exists\, p < p^*$ such that $S_p > \frac{p}{H_p}$ when $S_p \leq \frac{p}{H_p}$ for $p \geq p^*$

Proof: Let $q_i, 1 \leq i \leq p^*$ be such that $S_p \leq \frac{p^*}{H_{p^*}} \leq \frac{p}{H_p}$ for $p \geq p^*$.

Let a_i, $1 \leq i \leq p$, be the probability that i processors will be used when $p < p^*$.

Let $a_i = q_i = \frac{1}{p^*}$, for $1 \leq i \leq p-1$, and $a_p = \sum_{i=p}^{p^*} q_i = \frac{p^*-p+1}{p^*}$

Now, $\sum_{i=1}^{p} \frac{a_i}{i} = \sum_{i=1}^{p-1} \frac{1}{ip^*} + \frac{p^*-p+1}{p \cdot p^*} = \frac{1}{p^*} \left(H_{p-1} + \frac{1}{p} + \frac{p^*-p}{p}\right)$

$$= \frac{1}{p^*} \left(\frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{p}\right) + \frac{1}{p}$$

$$< \frac{1}{p} \left(\frac{1}{2} + \ldots + \frac{1}{p}\right) + \frac{1}{p} = \frac{H_p}{p}$$

$$\therefore S_p = \frac{1}{\sum_{i=1}^{p} \frac{a_i}{i}} > \frac{p}{H_p}$$

□

It should be noted that we have used the expression $\frac{p}{\ln p}$ as an approximation to the (tighter) upper bound of $\frac{p}{H_p}$, because $\ln p$ is a more well-tabulated and easily-recognized function than H_p. In fact, $H_p \to \ln p$ as $p \to \infty$. But for small values of p, especially of $p \leq 8$, we should really use the actual upper bound of $\frac{p}{H_p}$, since $\frac{p}{\ln p}$ shows some anomalous behavior which is not exhibited by $\frac{p}{H_p}$, e.g., $\frac{p}{H_p}$ has a minimum point at $p=3$, whereas $\frac{p}{H_p}$ is a strictly increasing function. Also, for p=1, $\frac{p}{\ln p}$ is undefined whereas $\frac{p}{H_p} = 1$.

The results of this chapter are summarized in the following figures and corollaries:

Figure 3 shows the speedup, S_p, versus p, for one fixed computation (i.e., p^* is fixed at 36, p is variable). We note that a maximum speedup of an order of magnitude ($S_p \leq 10$) is attainable when $p^*=36$, in cases (i) and (ii). In any case, the speedup cannot be greater than p^*. Hence, no improvement in speed can result from using more physical processors, p, than the maximum number of logical parallel processes, p^*, in the computation.

Figure 4 shows the speedup, S_p, versus p^*, for one fixed p-parallel processor (i.e., p is fixed at 16 processors, p^* is variable). This time, the speedup for different computations depends on whether the computation has more inherent parallelism, p^*, than the computer system, with a fixed number p of parallel processors, or not.

Figure 5 considers the maximum attainable speedup, S_{p^*}, of each computation, over all possible computations. Hence, in each case, we choose a computer organization with $p=p^*$ processors, we see that S_p^* is limited by either of $\ln p^*$ or p^*.

We summarize the attainable speedup regions for one computation:

Corollary 3.4: For any given computation, one of the following is true:

(i) $S_p \leq \min\left(\frac{p}{\ln p}, \frac{p^*}{\ln p^*}\right)$ iff $\sum_{i=1}^{p} \frac{q_i - \frac{1}{p}}{i} > 0$

(ii) $S_p \leq \min\left(p, \frac{p^*}{\ln p^*}\right)$ if $\sum_{i=1}^{p} \frac{q_i - \frac{1}{p}}{i} < 0$ but

$$\sum_{i=1}^{p^*} \frac{r_i - \frac{1}{p^*}}{i} \geq 0$$

(iii) $S_p \leq \min(p, p^*)$ otherwise

We have established the fact that the speedup depends on both the architecture and the computation. However, we often wish to express speedup bounds solely in terms of the architecture in order that we may evaluate the performance of a particular architectural feature, e.g., the number of parallel processors available, p. The next corollary gives computation-independent speedup bounds:

Corollary 3.5

(i) $S_P \leq \frac{p}{P - \ln P}$, iff $\sum_{i=1}^{p} \frac{q_i - \frac{1}{p}}{i} \geq 0$

(ii) $S_p \leq p$, otherwise.

Similarly, we can express the maximum speedup of any given computation, in the following purely computation-dependent speedup bounds:

Corollary 3.6

The maximum speedup of a given computation, assuming sufficient physical parallel processors $(p \geq p^*)$ is:

(i) $S_p \leq \frac{p^*}{p - \ln p^*}$, iff $\sum_{i=1}^{p^*} \frac{r_i - \frac{1}{p^*}}{i} \geq 0$

(ii) $S_p \leq p^*$, otherwise.

Comparison with Empirical Results

Kuck, et al [3-5] have written a sophisticated program analyser which turns ordinary Fortran programs written for uniprocessors into the format of a "computation" as defined in our model of computation on a p-parallel processor. They have then carried out experiments on a wide variety of programs to observe the speedup attainable by ordinary programs when a finite but unlimited number of parallel processors are available on which to execute these programs. In terms of parameters in our model, they have chosen $p=p^*$ processors, where p^* is determined for each program by their program analyser.

The eighty-six programs they have analysed are summarixed into seven categories, as can be seen in Table 1. The average over these 7 categories is the entry labelled "ALL". As this work comprises perhaps the major experimental effort in the area of determining speedup ratios of programs on parallel processor systems, we cannot ignore these results. Hence we use these experimental results as raw data by which we compare our theoretically derived upper bound for speedup of $O(\frac{p^*}{\ln p^*})$. Kuck's results are ploted in figure 5, the graph of S_{p^*} versus p^*, and tabulated in Table 1. As can be seen, there is good agreement between Kuck's experimental results and corollary 3.6. Only 2 of the 7 categories exceed the $\frac{p^*}{\ln p^*}$ bound, viz, NUME and EIS. On examining these two types of computations in greater detail, we find that the two points they share in common are:

(1) they have the two highest average number of nested DOs

(2) they have the two highest average number of Assignment-Statement Blocks inside DO loops.

NUME contains standard numerical analysis programs and EIS are eigenvalue programs. We note that both these are matrix-type computations with a large amount of inherent parallelism.

They have a large number of iterative loops acting on disjoint elements in matrices. In the earlier section on "Typical Speedup Ratios", We have already mentioned that such types of computations have maximum speedup ratios of $k_1 p$, or linear with the number of processors available.

Since NUME and EIS consist of 18 out of a total of 86 programs studied, we can say that about 80% of the programs have speedups less than $\frac{p^*}{\ln p^*}$.

	p^*	T_1	T_{p^*}	S_{p^*} empirical	$S_p > \frac{T_1}{10 \log_2 T_1}$ Kuck	$S_p \leq \frac{p^*}{\ln p^*}$ Lee
GPSS	14	30	12	3.2	0.61	5.30
DYS	19	224	146	4.9	2.87	6.45
TIME	23	174	22	7.1	2.34	7.34
MISC	39	274	32	8.4	3.38	10.65
NUME	51	654	77	20.7	6.99	12.97
JAN	62	357	48	12.1	4.21	15.02
EIS	82	896	208	22.6	9.14	18.61
ALL	37	310	63	9.8	3.75	10.25

Table 1: Comparison with Kuck's Experimental Results

The first 4 columns $(p^*, T_1, T_{p^*}, S_{p^*})$ are taken directly from [3], and form the "raw data".

The 5th column gives Kuck's hypothesis for the lower bound for S_p.

The 6th column gives my theoretical upper bound of $S_p \leq \frac{p^*}{\ln p^*}$

The values in column 4 are plotted in Figure 5.

4. RELATED PERFORMANCE MEASURES

Based on the definitions in Chapter 2, we can obtain related performance bounds corresponding to corollaries 3.4, 3.5. and 3.6 for each of execution time $T_{p'}$ efficiency E_p, and space-time product $ST_{p'}$ for a p-parallel processor.

We will use the following notation:

Condition A: $\sum_{i=1}^{p} \frac{q_i - \frac{1}{p}}{i} \geq 0$

Conditon B: $\sum_{i=1}^{p} \frac{q_i - \frac{1}{p}}{i} < 0$ but $\sum_{i=1}^{p^*} \frac{r_i - \frac{1}{p}}{i} \geq 0$

We will state only those results corresponding to corollary 3.4, where the dependence of the performance bound on both the architecture and the computation is shown.

Minimum Execution Time

Theorem 4.1

For any given computation, one of the following is true:

(i) $T_p \geq T_1 \cdot \max\left(\frac{\ln p}{p}, \frac{\ln p^*}{p^*}\right)$ iff condition A

(ii) $T_p \geq T_1 \cdot \max\left(\frac{1}{p}, \frac{\ln p^*}{p^*}\right)$ if condition B

(iii) $T_p \geq T_1 \cdot \max\left(\frac{1}{p}, \frac{1}{p^*}\right)$ otherwise

In Figure 6, we plot T_p versus p for a given computation (p^* fixed at 36). In Figure 7, we plot T_{p^*} versus p^*, for all possible computations.

From both figures, we can see that it is no use increasing the number of physical processors used beyond p^*, since the minimum execution time for the computation has a lower bound of either $(\frac{\ln p^*}{p^*})T_1$ or $(\frac{1}{p^*})T_1$, depending only on the probability distribution $\{r_i, 1 \leq i \leq p^*\}$ of the computation, independent of the computer system.

Maximum Efficiency

Theorem 4.2

For any given computation, one of the following is true:

(i) $E_p \leq \min(\frac{1}{\ln p}, \frac{p^*}{p.\ln p^*})$ iff condition A

(ii) $E_p \leq \min(1, \frac{p^*}{p.\ln p^*})$ if condition B

(iii) $E_p \leq \min(1, \frac{p^*}{p})$ otherwise

In Figure 8, we show E_p versus p for one fixed computation (i.e, p^* given). We note that the efficiency drops rapidly as we increase p. For case (i), with $p^*=36$, we achieve an order of magnitude speed improvement with a maximum efficiency of 0.28. Under any circumstances, there is no value in increasing p beyond p^*, since the maximum efficiency deteriorates at the rate of $\frac{1}{p}$, while the maximum speedup remains constant. (See also Figure 3)

In Figure 9, we show E_{p^*} versus p^* for all possible computations. This is the maximum attainable efficiency when $p=p^*$ processors are used, giving the maximum speedup, S_{p^*}. (See also Figure 5)

Minimum Space-Time Product

Theorem 4.3

For any given computation, one of the following is true:

(i) $ST_p \geq T_1 \cdot \max(\ln p, \frac{p}{p^*}\ln p^*)$, iff condition A

(ii) $ST_p \geq T_1 \cdot \max(1, \frac{p}{p^*}\ln p^*)$, if condition B

(iii) $ST_p \geq T_1 \cdot \max(1, \frac{p}{p^*})$ otherwise

In Figure 10, we plot ST_{p^*} versus p^*. For case (i), we see that the performance of the p-parallel processor, measured in terms of the relative space-time product (normalised by T_1), increases at least as $\ln p^*$.

5. CONCLUSIONS

We have defined a general model of computation on a p-parallel processor, and isolated performance measures by which we may evaluate the performance improvements, if any, of p-parallel processor systems over uniprocessor systems. We have shown how the performance depends on both the computer architecture and the computation, and distinguished clearly between the number of physical parallel processors available (p) and the maximum number of logical parallel processes (p^*) inherent in the computation.

We then derived necessary and sufficient conditions, $\sum_{i=1}^{p} q_i - \frac{1}{p} \geq 0$ and $\sum_{i=1}^{p} q_i = 1$, under which:

(i) speedup, $S_p \leq \min(\frac{p}{\ln p}, \frac{p^*}{\ln p^*})$

(ii) execution time, $T_p \geq T_1 \cdot \max(\frac{\ln p}{p}, \frac{\ln p^*}{p^*})$

(iii) efficiency, $E_p \leq \min(\frac{1}{\ln p}, \frac{p^*}{p} \cdot \frac{1}{\ln p^*})$

(iv) space-time product, $ST_p \geq T_1 \cdot \max(\ln p, \frac{p}{p^*} \cdot \ln p^*)$

In each case, $\ln p$ is an approximation for the pth. harmonic number H_p.

Despite the many different views that exist on the potential performance improvements of parallel processor systems over uniprocessor systems, the upper speedup bound, $S_p \leq \min(\frac{p}{\ln p}, \frac{p^*}{\ln p^*})$, has never been established before. Furthermore, comparison with the extensive experimental results of Kuck et al indicate that empirical speedups obtained are indeed $O(\frac{p^*}{\ln p^*})$.

Acknowledgement: I would like to thank my adviser, Professor Michael Flynn, who first introduced me to this subject.

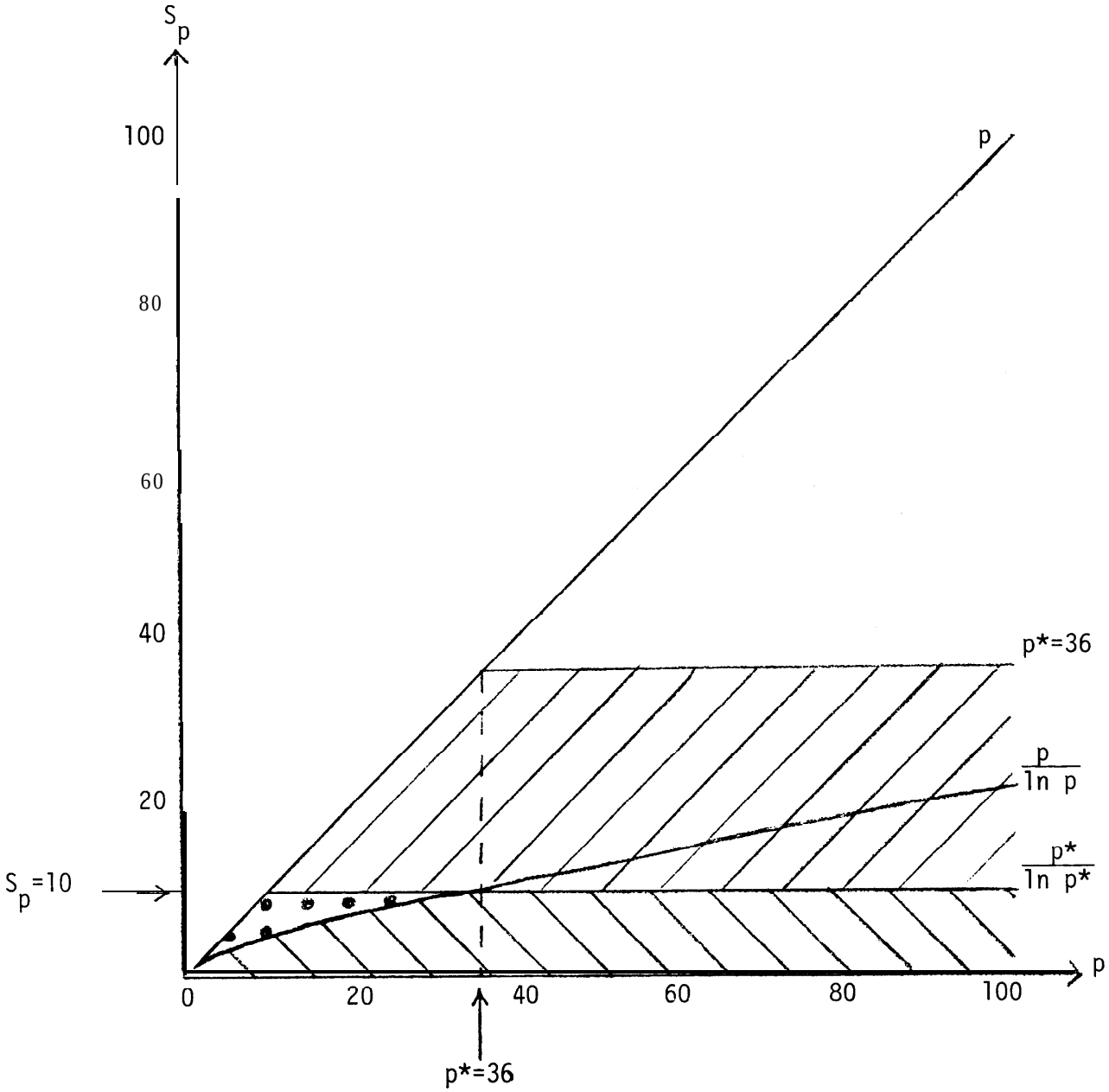

Figure 3: S_p Versus p (p*=36)

The Attainable Speedup Regions, for a Given Computation:

(i) $S_p \leq \min\left(\frac{p}{\ln p}, \frac{p^*}{\ln p^*}\right)$ iff $\sum_{i=1}^{p} q_i - \frac{1}{p} \geq 0$

(ii) $S_p \leq \min\left(p, \frac{p^*}{\ln p^*}\right)$ if $\sum_{i=1}^{p} q_i - \frac{P}{p} \frac{1}{< \text{Obut}_i} \sum_{i=1}^{p^*} r_i - \frac{1}{p^*_i} \geq 0$

(iii) $S_p \leq \min(p, p^*)$ otherwise

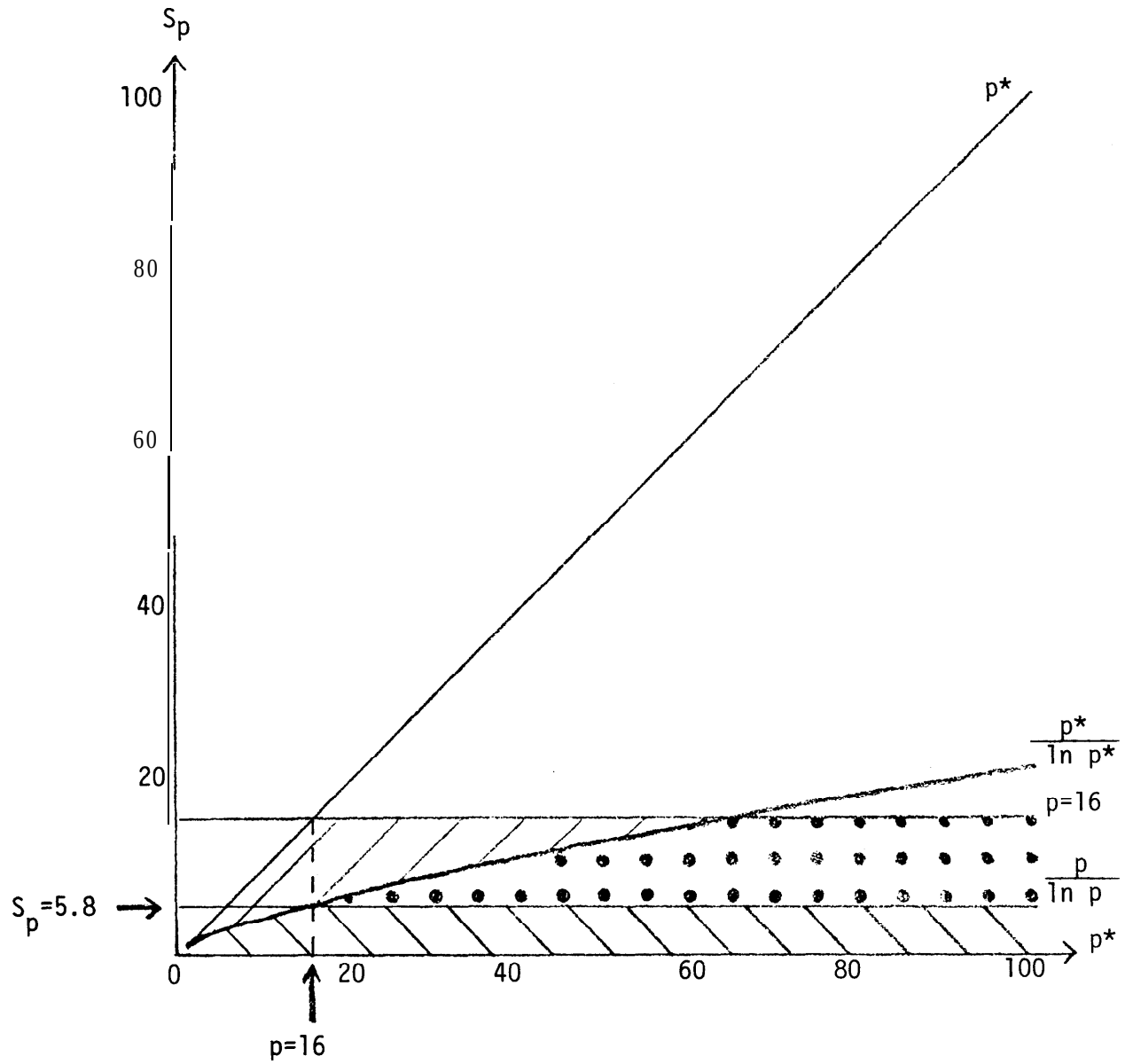

Figure 4: S_p Versus p* (p=16)

The Attainable Speedup Regions, for a Given p-Parallel Processor:

(i) $S_p \leq \min\left(\frac{p}{\ln p}, \frac{p^*}{\ln p^*}\right)$

(ii) $S_p \leq \min\left(p, \frac{p^*}{\ln p^*}\right)$

(iii) $S_p \leq \min(p, p^*)$

} conditions as in Figure 3

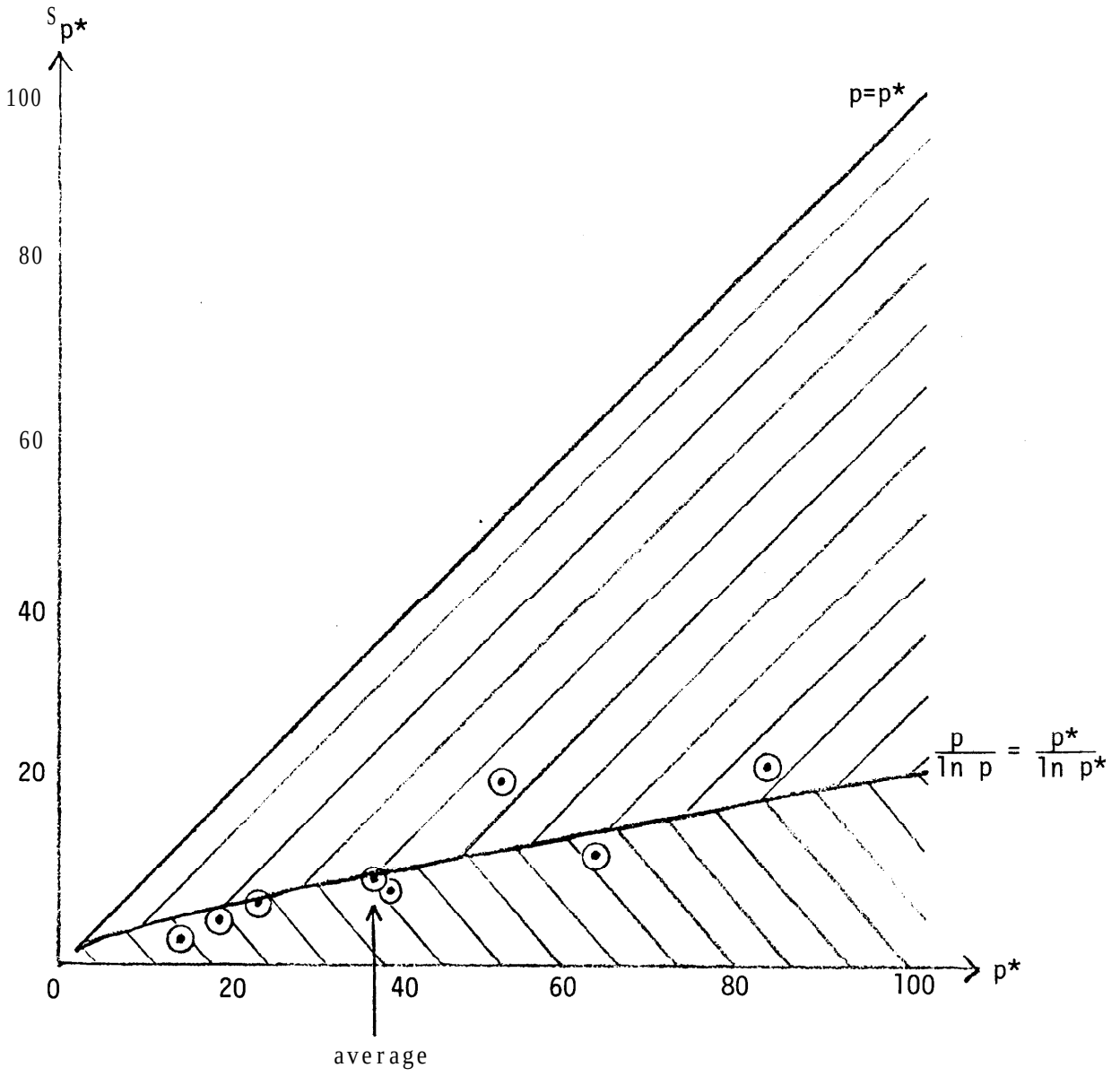

Figure 5: S_p^* Versus p* ($p=p^*$)

The Maximum Attainable Speedup Regions, for all Computations:

(i) $S_p^* \leq \frac{p^*}{\ln p^*}$, if $\sum_{i=1}^{p^*} \frac{r_i - \frac{1}{p}}{i} \geq 0$

(ii) $S_p^* \leq p^*$, otherwise

⊙ Kuck's experimental results

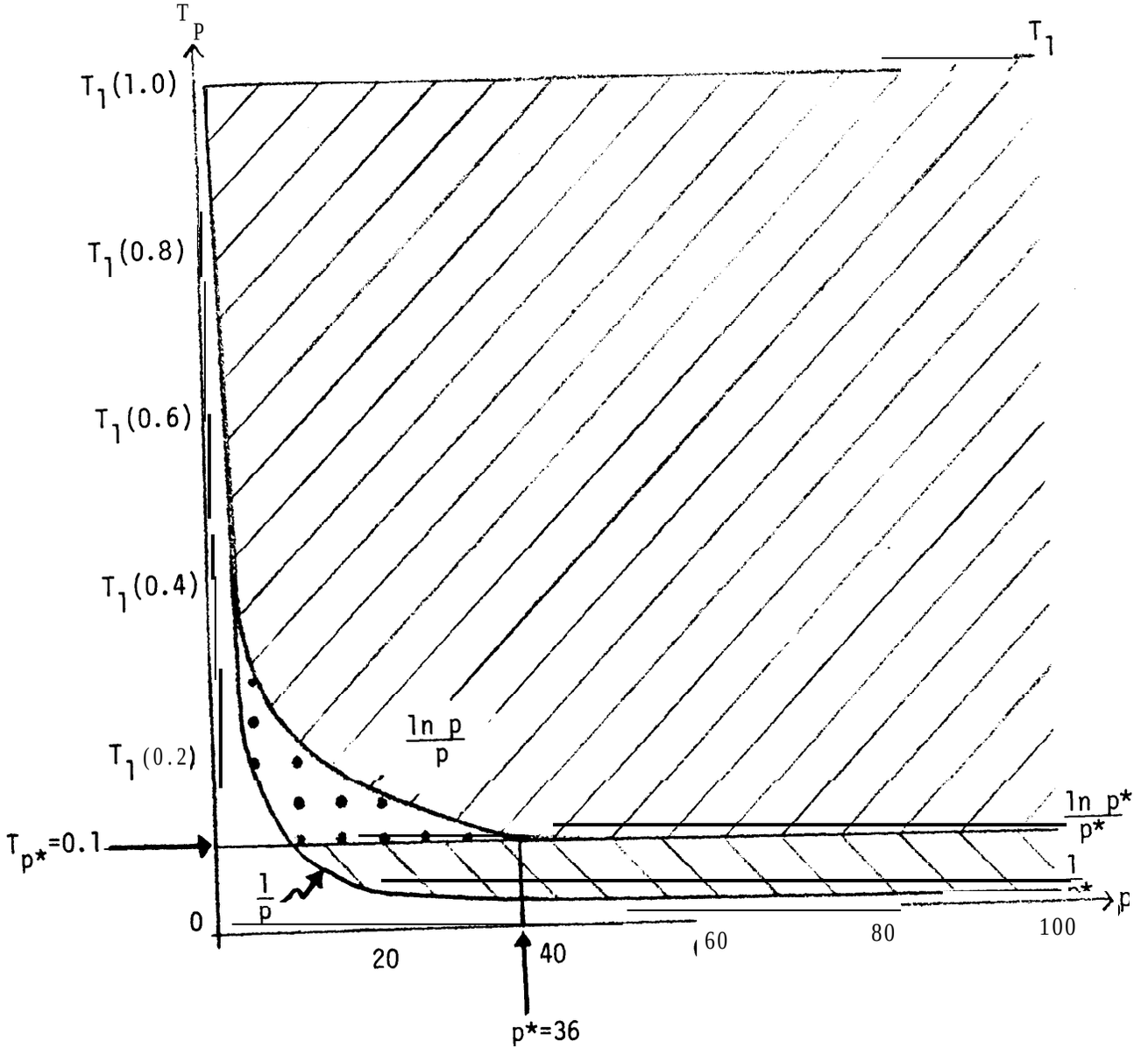

Figure 6: T_p Versus p ($p^*=36$)

(i) $T_p \geq T_1 \cdot \max\left(\frac{\ln p}{p}, \frac{\ln p^*}{p^*}\right)$

(ii) $T_p \geq T_1 \cdot \max\left(\frac{1}{p}, \frac{\ln p}{p^*}\right)$

(iii) $T_p \geq T_1 \cdot \max\left(\frac{1}{p}, \frac{1}{p^*}\right)$

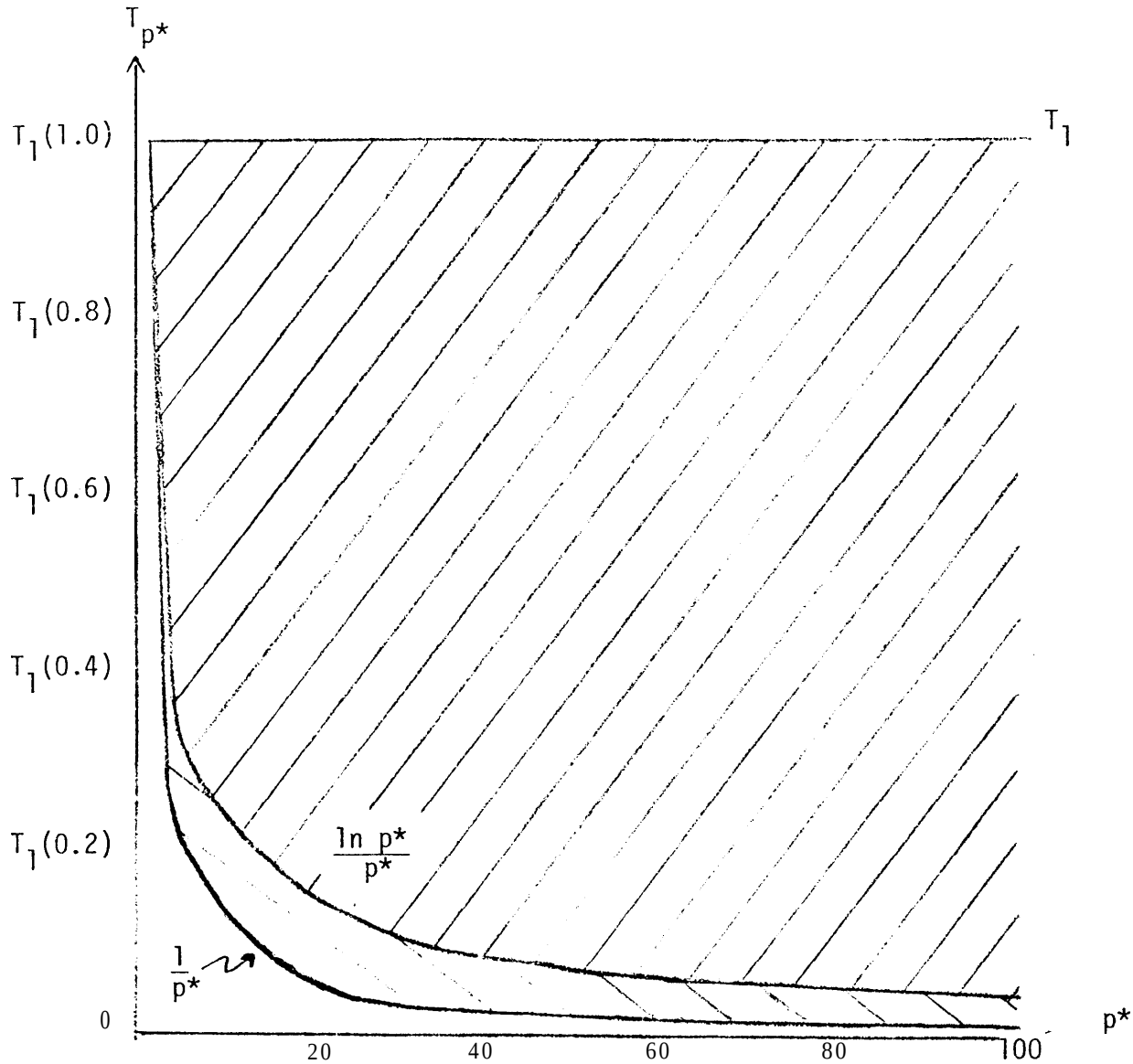

Figure 7: T_{p^*} Versus p^* $(p=p^*)$

(i) $T_{p^*} \geq T_1 \cdot \frac{\ln p^*}{p^*}$, if $\sum_{i=1}^{p^*} r_i - \frac{1}{p^*} \geq 0$

(ii) $T_{p^*} \geq T_1 \cdot \frac{1}{p^*}$, otherwise

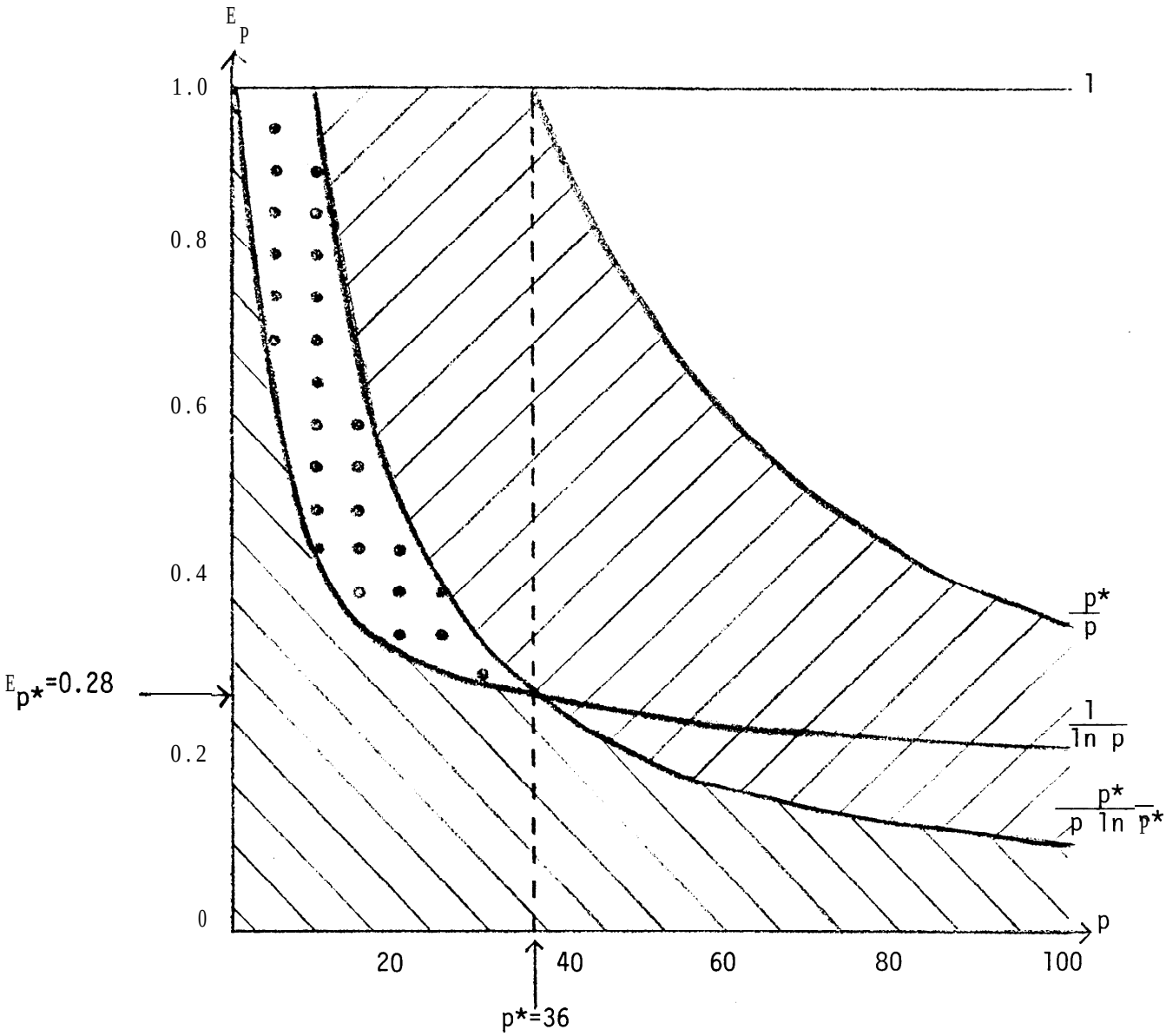

Figure 8: E_p Versus $p(p^*=36)$

The Attainable Efficiency Regions for a given computation:

(i) $E_p \leq \min\left(\frac{1}{\ln p}, \frac{p^*}{p \ln p^*}\right)$

(ii) $E_p \leq \min\left(1, \frac{p^*}{p \ln p^*}\right)$

(iii) $E_p \leq \min\left(1, \frac{p^*}{p}\right)$

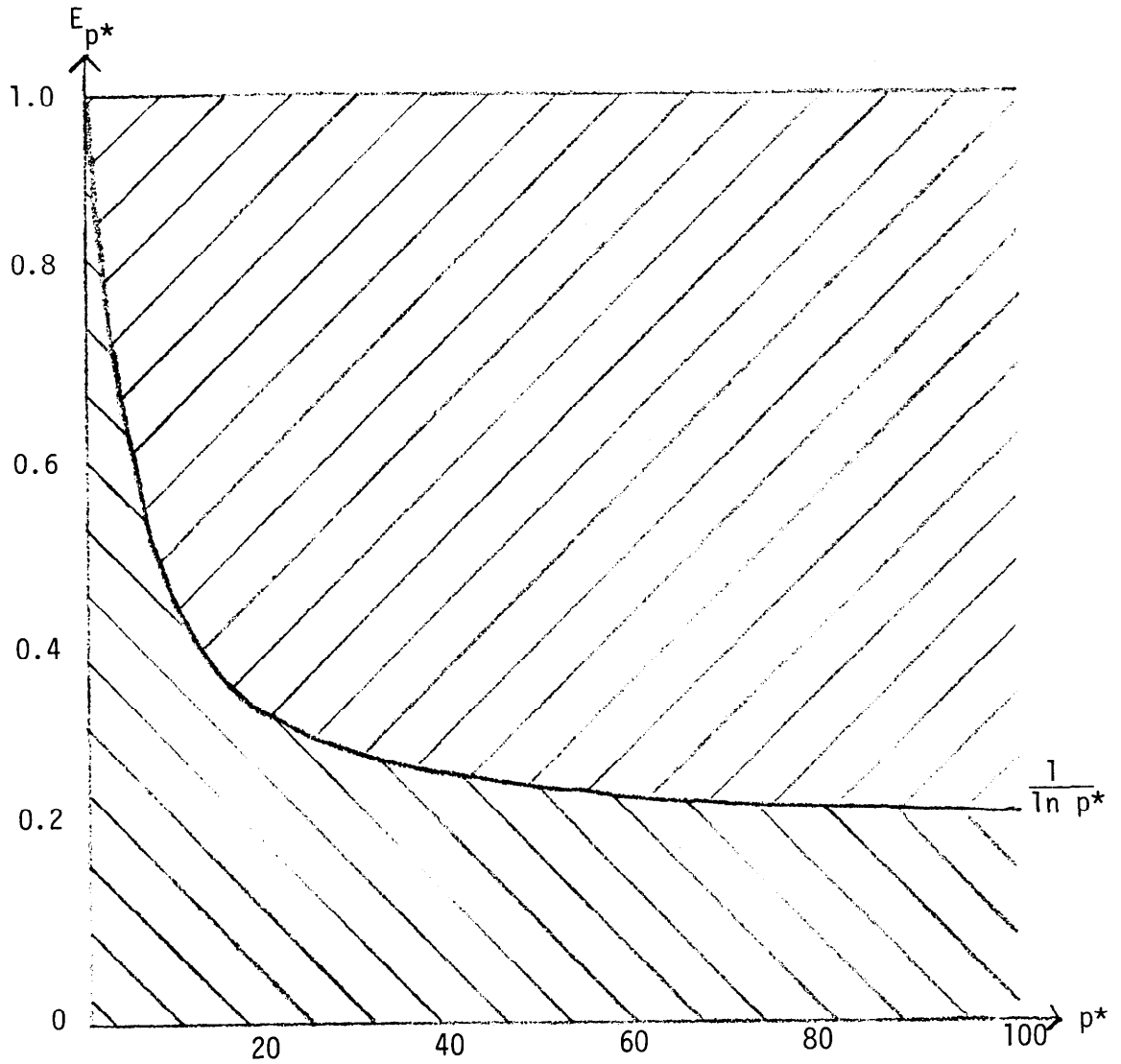

Figure 9: E_{p^*} Versus p^* $(p=p^*)$

The maximum Attainable Efficiency Regions, for all computations:

(i) $E_{p^*} \leq \frac{1}{\ln p^*}$, if $\sum_{i=1}^{p^*} r_i - \frac{1}{i} \frac{1}{p^*}$

(ii) $E_{p^*} \leq 1$, otherwise

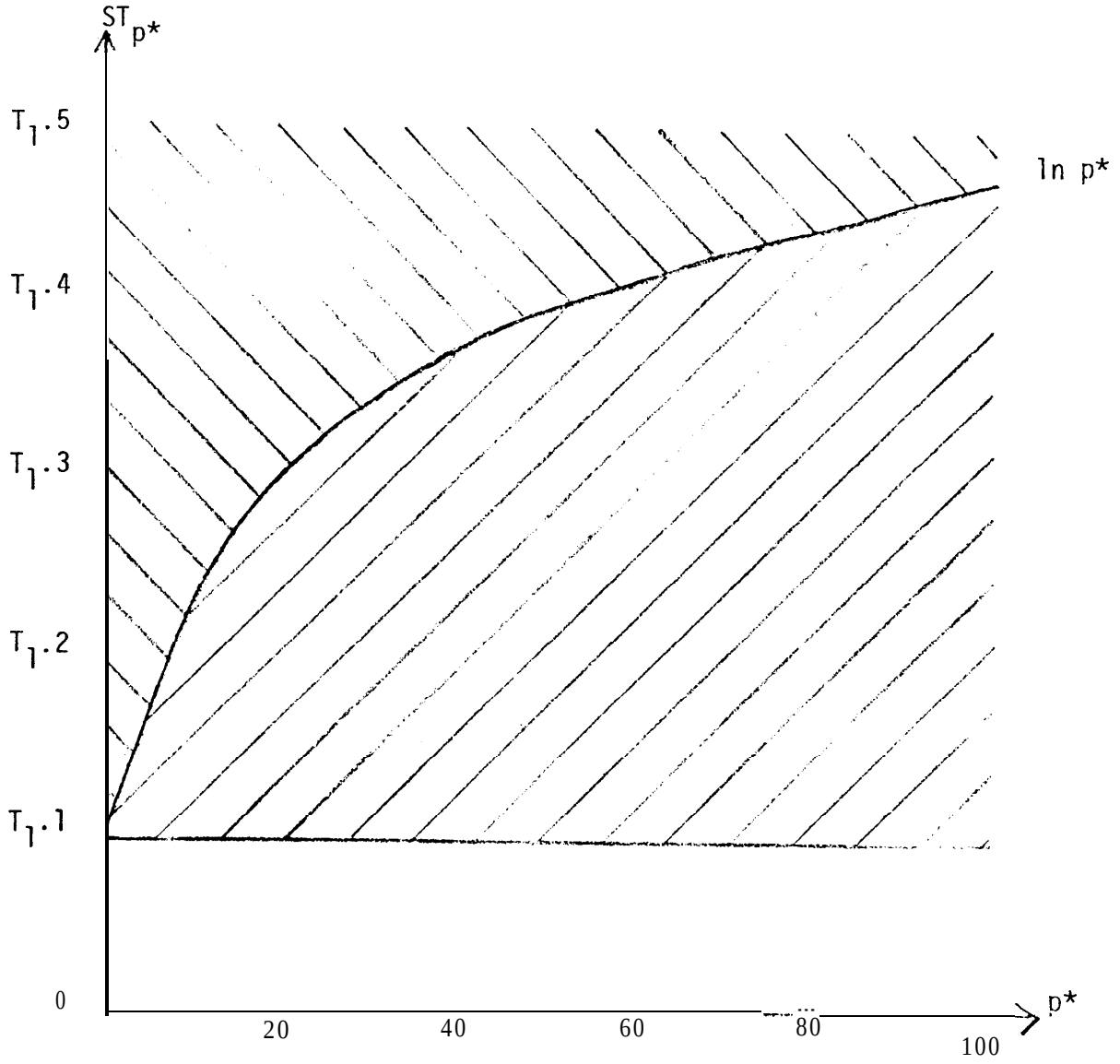

Figure 10: ST_{p^*} Versus p* (p=p*)

(i) $ST_{p^*} \geq \ln p^* > 1$ iff $\sum_{i=1}^{p^*} \frac{r_i - \bar{r}_{p^*}}{i} < 0$

(ii) $ST_{p^*} \geq 1$, otherwise

BIBLIOGRAPHY

1. Flynn, M. "Some Computer Organizations and their Effectiveness", IEEE TC Sept. 1972.

2. Flynn, M. "Trends and Problems in Computer Organizations", Information Processing 1974, North-Holland.

3. Kuck, D., et al, "Measurements of Parallelism in Ordinary Fortran Programs", 1973 Sagamore Computer Conference on Parallel Processing.

4. Kuck, D., Muraoka, Y., and Chen, S.C., "On the Number of Operations Simultaneously Executable in Fortran-like Programs and Their Resulting Speed-up", IEEE TC, 1972.

5. Kuck, D., "Parallel Procesing of Ordinary Programs", Department of Computer Science, University of Illinois, November 1975.

6. Minsky, M., and Papert, S., "On Some Associative, Parallal, and Analog Computations", from "Associative Information Techniques", 1971.

7. Minsky, M., "Form and Content in Computer Science", JACM 17, April 1970.

8. Amdahl, G. M., "Validity of the Single Processor Approach to Achieveing Large Scale Computing Capabilities", Proc AFIPS Conf. 1967.

9. Stone, H., "Problems of Parallel Computation", from "Complexity of Sequential and Parallel Numerical Algorithms", 1973 Academic Press.

10. Brent, R. P., "The Parallel Evaluation of Arithmetic Expressions Without Division", IEEE TC, May 1973.

11. Maruyama, K., "On the Parallel Evaluation of Polynomials", IEEE TC, Jan.1973.

12. Karp and Miranker, "Parallel Minimax Search for a Maximum", J. of Comb. Theory 4, 1968, Pg. 19-35.

13. Kogge, P. and Stone, H., "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations", Stanford University Tech. Report 25, March 1972.

14. Ramamoorthy, C.V. and Gonzalez, M. J., "A Survey of Techniques for Recognizing Parallel Processable Streams in Computer Organizations, AFIPS Conf. Proceedings, FJCC, 1969.

15. Bernstein, A. "Analysis of Programs for Parallel Processing", IEEE TC 1966.

16. Chen, T.C., "Unconventional Superspeed Computer Systems", Proc. AFIPS, SJCC 1971.

17. Davis, E.W. ,Jr., "Concurrent Processing of Conditional Jump Trees", COMPCON 72, IEEE Computer Society Conf. Proc. 1972.

18. Yu, P., "Some Notes on the Log Conjecture for Parallel Processors" Stanford University, DSL Tech. Note 90, June 76.

19. Riseman, E. and Caxton, C., "The Inhibition of Potential Parallelism by Conditional Jumps", IEEE TC, December 1972.

20. Caxton, C. and Riseman, E., "Percolation of Code to Enhance Parallel Dispatching and Execution", IEEE TC, December 1972.

21. Knuth, D., "The Art of Computer Programming", Vol.I, Addison Wesley, 1969.

22. Ibbett, R. N., "The MU-5 Instruction Pipeline", The Computer Journal, Vol. 15, No.1, 1971.