A FULLY AUTOMATIC ANALYTIC APPROACH TO BUDGET-CONSTRAINED SYSTEM UPGRADE

By

ANGELA SAI ON WONG

B.S., University of California at Los Angeles, 1983

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE STUDIES
(DEPARTMENT OF COMPUTER SCIENCE)

We accept this thesis as conforming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

March 1987

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of COMPUTER SCIENCE

The University of British Columbia
1956 Main Mall
Vancouver, Canada
V6T 1Y3

Date March 30, 1987

Abstract

This thesis describes the development of a software package to upgrade computer systems. The package, named OPTIMAL, solves the following problem: given an existing computer system and its workload, a budget, and the costs and descriptions of available upgrade alternatives for devices in the system, what is the most cost-effective way of upgrading and tuning the system to produce the optimal system throughput? To enhance the practicality of OPTIMAL, the research followed two criteria: i) input required by OPTIMAL must be system and workload characteristics directly measurable from the system under consideration; ii) other than gathering the appropriate input data, the package must be completely automated and must not require any specialized knowledge in systems performance evaluation to interpret the results. The output of OPTIMAL consists of the optimal system throughput under the budget constraint, the workload and system configuration (or upgrade strategy) that provide such throughput, and the cost of the upgrade. Various optimization techniques, including saturation analysis and fine tuning, have been applied to enhance the performance of OPTIMAL.

Acknowledgement

Words cannot quite express my gratitude for my supervisor, Dr. S. Chanson, for his inspirations, encouragement, advice and guidance during the entire period of the development of OPTIMAL. Thanks must also go to Dr. S. Vuong for being the second reader, and to friends in the Department of Computer Science here for technical discussions, proof reading and helping me on producing the final draft of this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Role of Performance Evaluation in Capacity Planning

As the computer plays an increasingly dominant role in today's world, a great deal of human and economic effort is being spent on planning computer system configurations to meet the service requirements of a given workload. This process of planning system configurations is commonly known as *capacity planning*; it is a process of continually monitoring a system and its workload, and recommending changes when one or more service requirements with respect to a workload is not being met. The recommendations may include adjusting the workload, hardware changes, operating system changes and staffing changes. Adjusting workload may include restructuring or relocating user files. Hardware changes are to reconfigure, add or replace existing hardware components. Operating system changes include adjusting the settings of various parameters. Staffing changes include hiring more staff, retraining present staff or instituting new

quality control procedures. [Conn84]

Capacity planning is usually carried out in two stages: characterization of workload and analysis of performance [Pang86]. In the first stage, the workload of some system of interest is analyzed and quantified; this is then used as input to the second stage. The results of performance analysis indicate whether the system configuration is suitable for the workload.

In the performance analysis stage, system performance is typically computed many times as modifications to system configuration or system workload are made. Different approaches, ranging from guessing, mathematical modelling, simulation, to real system emulation, have been used for performance prediction [Pang86]. Among these, performance evaluation by analytic modelling has been used extensively [Kuma80] [Lo80] [Sevc80] [Hodg82] and demonstrated to be effective tools to provide reasonable predictions. In particular, the most useful analytic modelling techniques use workload parameter values derived from a real system, and employs mathematical methods in queueing theory and operational analysis [Denn78] to predict system performance.

1.2 Motivation of Thesis

The notion of applying artificial intelligence (specifically, expert system) technology to capacity planning to increase the productivity of performance analysts has aroused

much interests [Levi85] [Maed85] [Ostr85]. In particular, an expert system which uses facts and heuristics on a domain of expertise to solve a certain problem provides a powerful tool for system performance analysis. Previously, most works have been designed for the needs of particular computer systems and utilize tools specific to these systems [Lo80] [Hodg82] [Levi85] [Ostr85] [Sala85], or consider only particular units in a computer system, such as the CPU or the disk storage system [Coop80] [Kuma80] [Orch83]. Even for [Sevc80] that considers alternative configurations to replace a system, the study was directed toward a particular case and thus is limited in its application. Furthermore, gathering the required input data and interpreting the output results in most of the suggested approaches require experience and knowledge in computer systems performance evaluation. In the realm of applying expert systems to capacity planning, the problem that we propose to solve for a given computer system is: given a budget, the current system configuration and workload, available devices on the market and their associated costs, what is the most cost-effective way of upgrading and tuning the system such that the optimal system throughput is obtained? This is the system upgrade problem that computing centre managers face from time to time when their systems saturate or approach saturation.

We have chosen system throughput rather than mean system response time as the primary performance index in our solution for the following reasons:

1. System throughput reflects the capacity of a computing system most directly.

2. For general purpose computing systems, response times vary tremendously - from a few milliseconds to hours. The mean response time is practically meaningless for evaluation purposes [Pang86].

3. Distribution of response time is expensive to compute and current estimation and measurement techniques predict throughput much more accurately than response time.

4. Under the condition that the workload remains constant, hardware upgrade strategies that maximize system throughput also optimize mean response time.

However, the approach described in this thesis can be used for other performance indices provided that the underlying system performance analysis tool is able to compute their values.

To enhance the practicality of the solution, two criteria are followed. The system under consideration (SUC) is a system already in existence, and the parameters characterizing the system configuration and workload are directly measurable from the SUC. Furthermore, unlike most currently available capacity planning systems, after the required input data have been gathered, the solution process is fully automated and no specialized knowledge in performance evaluation is required to interpret the final results.

Two intrinsic issues to consider initially are: what kind of system modelling tech-

nique should be employed and how optimal throughput should be defined. For the first issue, for the purpose of being able to apply the solution to any system in general, and the fact that analytic models have proven to be able to accurately and cost-effectively reflect the performance measures of a system [Denn78], we have opted for analytic models over simulation. For the second issue, rather than aiming at an absolute highest throughput possible for a given system, budget and set of upgrading device alternatives, we define optimal throughput to be the highest throughput per dollar spent. Furthermore, in almost all systems, depending on the availability of memory, there exists a limit on the number of processes that can be active simultaneously; such a limit is known as *memory constraint* which places an upper bound on the extent to which processing resources can be used concurrently, and thus on the system throughput [Lazo84]. So, for systems with paging and memory constraint, we further optimize the system throughput with the optimal multiprogramming level with respect to the SUC.

1.3 Brief Outline of Thesis

With the modelling technique and the optimization criterion to the upgrading problem determined, we begin with a simple backbone solution to our problem, the Baseline Heuristic. The solution is applicable to systems where each device has only one upgrade

alternative with some associated cost. The number of devices in the SUC is assumed to remain constant; i.e. only replacement of devices is considered. This Baseline Heuristic and its performance evaluation are presented in Chapter 2.

A more commonly found situation is one where each device in a system has one or more possible upgrade alternatives, and both addition and replacement of devices are to be considered in an optimal upgrade recommendation. To adapt the Baseline Heuristic to such a situation, a variation of the Baseline Heuristic, which we shall call OPTIMAL, is presented in Chapter 3. This is followed by a description of an implementation of OPTIMAL in C on a SUN 3/160 in Chapter 4. Various optimization techniques including saturation analysis, tuning, optimizing the multiprogramming level, and fine tuning are also discussed.

A conclusion of the research and possible future extensions of OPTIMAL are discussed in Chapter 5. A User's Guide to OPTIMAL and a sample session are presented in Appendix A.

Chapter 2

The Baseline Heuristic

2.1 Assumptions and Inputs

For simplicity, the SUC is assumed to be representable as a centralized single class closed queuing system model with batch or terminal workload. The possibility of extending the model to include more complex systems such as multiple job class systems will be discussed in Chapter 5.

Placement of files on devices with the objective of balancing the load across I/O devices [1] can often improve system performance [Lazo84]. Thus tuning, with the above objective, is considered whenever it is possible to move files from one I/O device to another. It is also assumed that such tuning measures do not incur additional monetary costs.

For systems with memory constraint, it is assumed that there exist two system-dependent parameters: the maximum memory size and the maximum degree of mul-

[1] In the context of tuning, only on-line secondary storage I/O devices are modelled.

tiprogramming. Homogeneity assumptions about the workload are also made, as described below. For swapping systems, it is assumed that with respect to the configuration and workload of the SUC, the mean degree of multiprogramming is measurable and the users share main memory equally. Thus the mean size of memory allocated per job is calculated by dividing the size of main memory available to users by the mean degree of multiprogramming. For paging systems, the size of memory allocated per job actually varies according to the current multiprogramming level of the SUC, but since jobs are assumed to be identical, the mean multiprogramming level of a paging system obtained over some fixed period of time is used. Furthermore for paging systems, it is assumed that the paging device can be either dedicated to paging activities, or shared between regular file I/O and paging activities. The memory reference characteristics of jobs and the page replacement policy of the operating system are assumed to interact with one another in a manner reflected by the programme lifetime function specific to the SUC [Lazo84]. It is also assumed that the operating system allocates memory on an equipartitional basis for the current number of jobs within the maximum multiprogramming level supported by the system.

The input data required by the Baseline Heuristic are the current system configuration and workload characteristics of the SUC, the available budget, a fixed unit increment on the capability of each device in the system, and the cost for such a unit increment. One system configuration characteristic is the capability of each device in

the system. The capability of a non-memory device is assumed to be characterized by the service rate, and may be expressed in various terms. For example, for the CPU, it may be in number of millions of instructions per second (MIPS), or in number of jobs per second; for I/O devices, it may be in number of jobs per second or in number of page frames per unit time. For memory, its capability may be expressed in number of kilobytes or number of page frames. Thus a unit increment for the CPU or an I/O device may be expressed as the increase in the number of jobs serviced per second, and for memory, it may be an increase in the number of page frames. Another system configuration characteristic is whether files on a device can be reallocated to another. Workload characteristics of SUC are expressed as visit ratios for I/O devices, mean number of instructions per job or visit ratio for CPU, and, the multiprogramming level for memory.

2.2 The Iterative Process

The following parameters are assumed in the Baseline Heuristic:

System Parameters:

Ncents : number of non-memory devices in the system

μ_i : service rate of device i

r : mean CPU instruction execution rate

memsize : main memory size

max-memsize : maximum main memory size supported by system

Workload Parameters:

Ncusts : number of active customers in the closed system

z : mean thinktime for transactional workload

v_i : visit ratio, i.e., mean number of service requests made per job, for device i

v : sum of visit ratios of all devices on which tuning is allowed

L : mean number of instructions executed by CPU per job

M : maximum multiprogramming level allowed in a memory-constrained system

m : mean size of memory allocated per job

$l(m) = lifetime(m)$: lifetime function for paging system

Economic Parameters

$\Delta\mu_i$: unit increment on service rate for device i

ΔM : unit increment on memory size

Δr : unit increment on the CPU instruction execution rate

ΔC_i : unit cost incurred by unit increase in the capability of device i

B : initial budget

remain : running balance of the budget kept in the heuristic

The Baseline Heuristic recommends tuning to be done initially if files are allowed to be moved across I/O devices. As mentioned earlier, balancing system load by re-assignment of files has been shown to improve system performance. An I/O device with very high utilization (close to 100%) may be a system bottleneck; improving the capability of other devices may not yield any improvement in system performance. However, if files are re-assigned from this device to others balancing their service demands, utilization of the original bottleneck will be lowered. Thus, if tuning is allowed, it should be done before upgrading. Otherwise, upgrading the original system bottleneck alone is not likely to be the most cost-effective measure to improve system throughput.

After initial tuning (if possible), the Baseline Heuristic operates as an iterative process.

Step 0 Balance load of I/O devices by reallocating files, if possible. Evaluate system to find initial system throughput t. Initialize $remain = B$.

The iterative process:

Step 1 For each device i for which upgrading is considered, $0 < i \leq Ncents$:

Step 1.1 If $remain \geq \Delta C_i$,

then if i = memory and $memsize + \Delta M \leq max\text{-}memsize$,

then increment *memsize* by ΔM,

else if i = CPU,

then increment either r by Δr, or μ_{cpu} by $\Delta\mu_{cpu}$,

where appropriate [2] ,

else increment μ_i by $\Delta\mu_i$.

Balance load of device i with other devices, if possible.

Step 1.2 Evaluate system to find current system throughput t_i with respect to the upgrade on device i.[3]

Step 1.3 Restore capability of device i to the state prior to Step 1. [4]

Step 2 Calculate the cost-effectiveness of device i, determined as the increase in throughput obtained per unit dollar spent:

$$ceff_i = (t_i - t)/\Delta C_i$$

Step 3 For device i where

$$ceff_i = \max_i\{ceff_i\},$$

upgrade device i by the corresponding increment. Balance load of device i with other I/O devices if possible.

[3]The system is modelled and evaluated analytically by a system performance analysis algorithm such as exact Mean Value Analysis.

[4]If tuning was done in Step 1.1, then visit ratios to these tuned devices should also be restored to the state in Step 1.

Step 4 Evaluate system performance and find the throughput t with respect to the upgrade in Step 3. Decrement *remain* by ΔC_i.

The iterative process halts as soon as $remain < \Delta C_i$, $\forall i$.

2.3 The Output

The Baseline Heuristic produces the following as results:

1. A recommmendation on how to distribute the given budget among the devices for which upgrading is considered.

2. The remaining budget.

3. The optimal system throughput with respect to the recommended system configuration, with the load balanced across I/O devices, if possible. [5] Note that, as discussed earlier in Section 1.2, although an absolute highest throughput, t_1, might be obtained with amount B_1 within the given budget, B, the heuristic produces a throughput, t_2, less than (or equal to) t_1, obtained with amount $B_2(\leq B_1)$. It is optimal in the sense that the improvement in throughput obtained per unit dollar with B_2 is higher than (or equal to) that with B_1.

[5] Balancing service demands on I/O devices while keeping the sum of their visit ratios constant is mathematically likely to produce non-integral visit ratios. Fine tuning of these devices is required to produce an optimal throughput that can be realized. This is further discussed in Section 4.3.

2.4 Performance of the Baseline Heuristic

To evaluate the performance of the Baseline Heuristic, a control programme, P1, was set up to enumerate all possible ways of allocating a given budget to the devices of a given system. For each such possible way, based on the allocated funds, each device is upgraded accordingly, and the system throughput is evaluated, using exact Mean Value Analysis (MVA) [Reis80]. The optimal throughput produced by P1 is thus the throughput which is maximum over all possible allocations of funds.

Two criteria of comparing the performance of the Baseline Heuristic and P1 are: the optimal throughput and the execution time required. Four hypothetical systems of different complexities have been used to test the two programmes. Their descriptions are shown in Tables 2.1 to 2.4.

- Closed interactive system without memory-constraint.
- No devices are tuned.
- $Ncusts = 15$
- $z = 50.0$s
- $B = \$20000$
- uniform unit increment = 1

Device i	μ_i	v_i	ΔC_i
CPU	300	96	\$2000
Disk 1	100	20	3000
Disk 2	120	35	4000
Disk 3	140	40	5000

Table 2.1: System 1

- Closed interactive system without memory-constraint.
- No devices are tuned.
- $Ncusts = 70$
- $z = 8.0$s
- $B = \$10000$
- uniform unit increment = 1

Device i	μ_i	v_i	ΔC_i
CPU	180	36	\$2000
Disk 1	40	10	999
Disk 2	50	13	1000
Disk 3	50	12	800

Table 2.2: System 2

- Closed interactive system without memory-constraint.
- Disks 1 and 3 can be tuned.
- $Ncusts = 70$
- $z = 8.0$s
- $B = \$10000$
- uniform unit increment = 1

Device i	μ_i	v_i	ΔC_i
CPU	180	36	\$2000
Disk 1	40	10	999
Disk 2	50	13	1000
Disk 3	50	12	800

Table 2.3: System 3

- Closed interactive system without memory-constraint.
- Disks 1 and 3 can be tuned.
- $Ncusts = 70$
- $z = 8.0$s
- $B = \$60000$
- uniform unit increment = 1

Device i	μ_i	v_i	ΔC_i
CPU	180	36	\$2000
Disk 1	40	10	999
Disk 2	50	13	1000
Disk 3	50	12	800

Table 2.4: System 4

Comparison of results in terms of execution time [6] , the optimal throughput obtained and amount spent is shown in Table 2.5.

[6]Time data was collected by the *time* system call in UNIX. User time and system time are reported as the length of time in seconds a programme spends in user mode and supervisory mode respectively; their sum contribute to the CPU time. Elapsed time is reported in hours:minutes:seconds. The comparisons were carried out and times were measured on a SUN 3/160 with an execution rate of approximately 2MIPS.

	Baseline Heuristic	P1	Conclusion
System 1			
User time(sec.)	3.3	22.3	For timing,
System time(sec.)	0.2	0.6	Baseline Heuristic
Elapsed time	00:00:03	00:00:24	is superior.
Optimal throughput	0.293115	0.293115	Same
Amount spent	$20000	$20000	Same
Throughput increase per dollar spent ($\times 10^{-6}$)	0.00405	0.00405	Same
System 2			
User time(sec.)	8.8	19.9	For timing,
System time(sec.)	0.3	1.3	Baseline Heuristic
Elapsed time	00:00:19	00:02:04	superior.
Optimal throughput	4.033241	4.033635	Differ by 0.0097%
Amount spent	$9597	$9797	
Throughput increase per dollar spent	0.000029	0.000028	Baseline Heuristic is superior.
System 3			
User time(sec.)	1.0	24.5	For timing,
System time(sec.)	0.2	1.3	Baseline Heuristic
Elapsed time	00:00:21	00:02:08	superior.
Optimal throughput	4.055627	4.059224	Differ by 0.0886%
Amount spent	$9800	$10000	
Throughput increase per dollar spent	0.0000307	0.0000305	Baseline Heuristic is superior.
System 4			
User time(sec.)	8.9	14943.8	For timing,
System time(sec.)	0.7	181.9	Baseline Heuristic
Elapsed time	00:01:50	04:14:58	far superior.
Optimal throughput	4.938146	4.942504	Differ by 0.0881%
Amount spent	$59400	$60000	
Throughput increase per dollar spent	0.0000199	0.0000198	Baseline Heuristic is superior.

Table 2.5: Comparison of Results from the Baseline Heuristic and P1 applied to Systems 1 to 4

Thus for the above examples, the improvement in throughput per dollar spent with the Baseline Heuristic is always equal to or higher than that with P1, and the optimal throughput obtained by the Baseline Heuristic is within 0-0.1% of the highest possible for each system. As for the execution time, the more complex the SUC, the better the Baseline Heuristic performs relative to P1; the improvement can be very significant, ranging from a few to more than a thousand times improvement in User time, and a few to more than two hundred times improvement in System time in the examples tested.

Chapter 3

Variation of the Baseline Heuristic

3.1 Description of a Practical Situation

A more commonly found situation is one in which each device in a system has more than one upgrade alternative, each alternative with a possibly different cost. Furthermore, the devices in the existing system may carry a positive resale value. Upgrading a system with a given budget may call for replacing existing devices with certain upgrade alternatives, as well as adding more devices to the system, completely changing the system configuration. A variation of the Baseline Heuristic, OPTIMAL, takes into consideration the above issues.

3.2 Description of OPTIMAL

3.2.1 Assumptions and Inputs

The assumptions for the Baseline Heuristic described in Section 2.1 carry over to OPTIMAL. In addition, for simplicity, it is assumed that the sets of upgrade alternatives for the devices in a given SUC are compatible with each other. Thus, for instance, each of the CPU upgrade alternatives is compatible with all of the memory upgrade alternatives. Since addition of devices is also considered, alternative j of device i added to the system as a new device is assumed to have the same upgrade alternatives as device i. A system-dependent parameter, the maximum number of devices on a given system, is also assumed.

Similar to the Baseline Heuristic, the input data required by OPTIMAL include a given budget and the existing system configuration. In addition, for each device in the SUC for which upgrading is considered, the required input data are the resale value, the number of upgrade alternatives, and for each alternative, its capability and its cost, and whether tuning is allowed for that device [1] .

[1] Applicable to non-CPU and non-memory devices only.

3.2.2 The Iterative Process

In addition to the system and workload parameters described in Section 2.2, the following are used by OPTIMAL:

System Parameters:

max : maximum number of devices supported by system

Economic Parameters:

$n\text{-}alt_i$: number of upgrade alternatives for device i

μ_{ij} : service rate of jth upgrade alternative of device i

M_j : memory size of jth upgrade alternative of the memory device

r_j : instruction execution rate of the jth upgrade alternative for the CPU

C_{ij} : cost of the jth upgrade alternative of device i

c_i : resale value of device i

B : initial budget

$remain$: running balance of the budget

Similar to the Baseline Heuristic, OPTIMAL recommends tuning to be done initially whenever possible. It then operates as an iterative process.

Step 0 Balance load of I/O devices by reallocating files if possible. Find initial system throughput t.

The iterative process:

Step 1 For each device i for which upgrading is considered, and for each upgrade alternative j of device i, $0 < i \leq Ncents$, $0 < j \leq n\text{-}alt_i$:

Step 1.1 Try replacing device i by alternative j: if $C_{ij} > c_i$ and $remain \geq C_{ij} - c_i$, or if $C_{ij} \leq c_i$, substitute capability of alternative j for device i. Balance load of device i with other devices if possible. If replacement is not possible, go to Step 1.4.

Step 1.2 Evaluate system performance and find the throughput t^r_{ij} with respect to the replacement in Step 1.1.

Step 1.3 Restore capability of device i to the state in Step 1. If tuning was done in Step 1.1, restore visit ratios of tuned devices to the state in Step 1.

Step 1.4 Try adding alternative j to system: [2]

if $remain \geq C_{ij}$,

and if $i \neq$ memory and $Ncents < max$,

or if $i =$ memory and $memsize + \Delta M \leq max\text{-}memsize$,

[2]Addition of devices is only permitted for memory and for I/O devices for which files can be reallocated.

then add a new device with capability equal to that of alternative j to the system. If i is an I/O device, move files to new device such that load is balanced across all I/O devices that can be tuned. If addition is not possible, go to Step 2.

Step 1.5 Evaluate system performance and find the throughput t_{ij}^a with respect to the addition in Step 1.4.

Step 1.6 Restore original system configuration by removing new device, and if $i \neq$ memory, restoring visit ratios to the state in Step 1.

Step 2 Calaculate the cost-effectiveness of each upgrading measure carried out in Step 1, i.e., for each device i, compute:

$$ceff_{ij}^r = (t_{ij}^r - t)/(C_{ij} - c_i)$$

and

$$ceff_{ij}^a = (t_{ij}^a - t)/C_{ij},$$

if the corresponding upgrade is possible.

Step 3 For device i and alternative j where

$$ceff_{ij}^{action} = \max_{i,j,action} \{ceff_{ij}^{action}\},\ action \in \{r, a\},$$

upgrade system by the corresponding *action*, i.e., either replacing device i by its j-th alternative, or adding alternative j to the system as a new device. Balance

load of device i with other I/O devices if possible. Increment *Ncents* if *action* is addition of device to system.

Step 4 Record system throughput t with respect to upgrade decision in Step 3 (i.e. $t = t^r_{ij}$ or t^a_{ij}). Decrement *remain* by $(C_{ij} - c_i)$ if *action* is replacement, and by C_{ij} if *action* is addition.

The iterative process terminates when the remaining budget is insufficient for any type of upgrading action for any device.

3.2.3 The Output

OPTIMAL produces the following as output:

1. A recommendation on how to distribute the given budget among the various upgrade alternatives, i.e., which devices to replace in the initial system and which devices should be added to the system.

2. The remaining budget.

3. The system throughput with respect to the recommended system configuration (after fine-tuning if possible).

Chapter 4

Implementation of OPTIMAL

4.1 System Performance Analysis Algorithms

As mentioned in Chapter 2, the SUC that is assumed is a centralized single class closed system with batch or terminal workload, and the system performance analysis algorithm that is used is the exact Mean Value Analysis algorithm first developed by Reiser and Lavenberg [Reis80] and later extended by Lazowska et al. [Lazo84] However, this is by no means a restriction on the practicality of the Baseline Heuristic and OPTIMAL. In fact, given the appropriate analytic modelling tool, other systems, for instance, multiple class systems, can also be applied to the Baseline Heuristic and OPTIMAL with minor modifications, as discussed later in Chapter 5. Such modelling tools are widely available; for instance, BEST/1 [BEST79] has been widely used in North America since 1979. QNETS [Pang86] analyzes multiple class centralized systems with memory-constraint. Another package, QNAP2 [INRI84], models and evaluates the performance of systems with varied complexities: open or closed systems with single

or multiple job class, with or without memory constraint, with local or remote transactions over a network, and non-separable [Denn78] systems with a finite number of processors linked together by a network.

Two notes on the exact Mean Value Analysis as applied to memory-constraint batch and paging systems. For memory-constraint batch systems where the total number of customers, *Ncusts*, is less than or equal to the multiprogramming level, M, of the system, rather than taking the mean throughput over all feasible populations 1,..., *Ncusts*, the system throughput is that obtained at multiprogramming level *Ncusts*, since all processes can be accommodated in main memory and there is no delay imposed outside of the multiprogramming subsystem (refer to Figure 4.1).

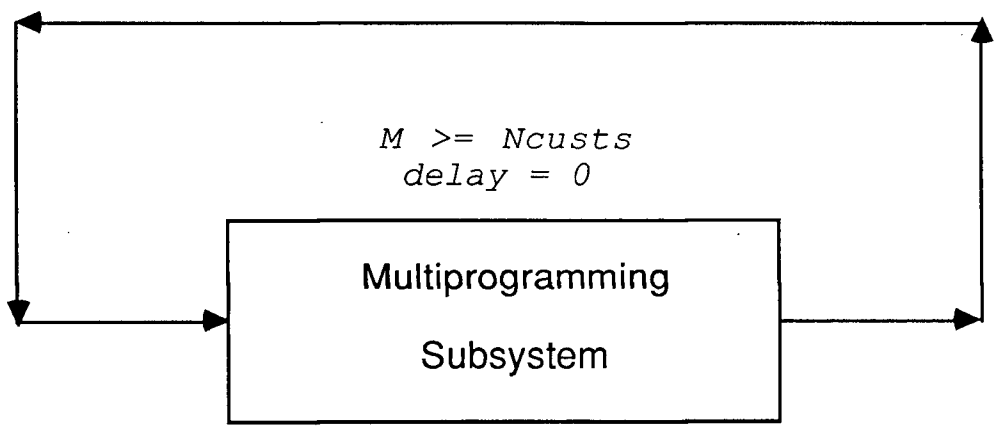

Figure 4.1: Closed Batch Model with Memory Constraint $M \geq Ncusts$

For paging systems, because the visit ratio to the paging device for paging service [1] is dependent on the current multiprogramming level, to calculate the system throughput at each feasible multiprogramming level, it is necessary to find the visit ratio and the service demand to the paging device for paging service at each of these levels. The visit ratio to the paging device for paging service at feasible population n is in turn derived from the formula:

$visit\ ratio_{pag} = service\ demand_{cpu} /\ lifetime(m)$,

where, according to the user-specified lifetime function, *lifetime(m)* yields the mean CPU time between consecutive page faults with respect to the memory allocation per job (m) when the current multiprogramming level is n (i.e., $m = memsize/n$).

4.2 The Implementation Assumptions

In addition to the assumptions outlined in Section 3.2.1 for OPTIMAL, one other major implementation assumption has been made for the sake of clarity in demonstrating the operation of OPTIMAL. The assumption is: for a given system, if files are allowed to be reallocated, then in principle they can be moved to any I/O device on which files can be reallocated. That is, when a file is moved from one device to another, the service demands of all I/O devices in the system to which files can be reallocated are also

[1] Paging devices may be shared between paging activities and regular file I/O activities.

balanced. Furthermore, only upgrade alternatives for memory and such "tuneable" I/O devices are considered to be added to the system in the iterative process of OPTIMAL. Since only single processor systems are considered in OPTIMAL, upgrade alternatives of the CPU are for replacement only, never for addition.

Other more sophisticated strategies can be incorporated into OPTIMAL by defining n closed groups of I/O devices. Moving a file from one device to another within a group results in balancing the service demands of only those devices in the group. One may also allow adding devices for which general tuning is not permitted. In this case, files from such an existing I/O device in the system are only moved to the new device to balance the service demands of these two devices only. Thus, different levels of sophistication may be built into OPTIMAL allowing a wider scope of application.

4.3 Optimization of OPTIMAL

Optimization is carried out in OPTIMAL in the following four main areas: saturation analysis and tuning, calculation of cost-effectiveness, optimizing multiprogramming level, and fine tuning.

4.3.1 Saturation Analysis and Tuning

As mentioned in Section 2.2, a saturated device in a system is not necessarily the most cost-effective device to upgrade. If the saturated device is an I/O device from which files can be moved to other I/O devices in the system balancing their service demands, then such tuning action may eliminate this primary bottleneck. Some other device whose service demand has not been affected by the tuning measure may become the most heavily utilized device in the system and hence emerges as the current primary bottleneck. However, owing to the fact that the cost-effectiveness of upgrading a device is defined as the ratio of the improvement in throughput to the cost of the upgrade, cost is also a factor in determining whether a device is the most cost-effective to upgrade. Thus, the most highly utilized device is not necessarily selected for upgrade.

4.3.2 Calculation of Cost-effectiveness

In some situations, the replacement cost of a new device may be less than or equal to the resale value of an existing device in the system. For instance, soon after a disk is purchased, a competitor manufacturer offers a huge discount on a slightly faster disk such that the resale value of the newly purchased disk is higher than or equal to the cost of the discounted disk. In such cases, after examining the replacement cost and resale value of a device, OPTIMAL upgrades this device by replacing it with the alternative, before proceeding to calculate the cost-effectiveness of the rest of the devices.

The profit made in the upgrade transaction is accounted into the budget. In situations where the capability of an upgrade alternative is equal to or even lower than that of an existing device in the system, OPTIMAL ignores this upgrade possibility and proceeds to examine the next.

4.3.3 Optimizing the Multiprogramming Level

When memory is added to memory-constrained swapping systems, the multiprogramming level of the system may be increased up to the maximum possible, based on the mean memory size allocated per job for that system (see Section 2.1). System throughput will increase with the multiprogramming level for such systems. For memory-constrained paging systems, however, since throughput is convex with multiprogramming level [Lazo84] (refer to Figure 4.2), the optimal multiprogramming level is one at which system throughput is maximized. It is important to note that the optimal multiprogramming level is a function of the current workload and system configuration, including the lifetime function that characterizes the paging activities of the particular system.

As in the case of tuning, optimization of the multiprogramming level for paging systems should be done before the iterative process of OPTIMAL begins. This is achieved by evaluating the system at each feasible population from 1 to the current

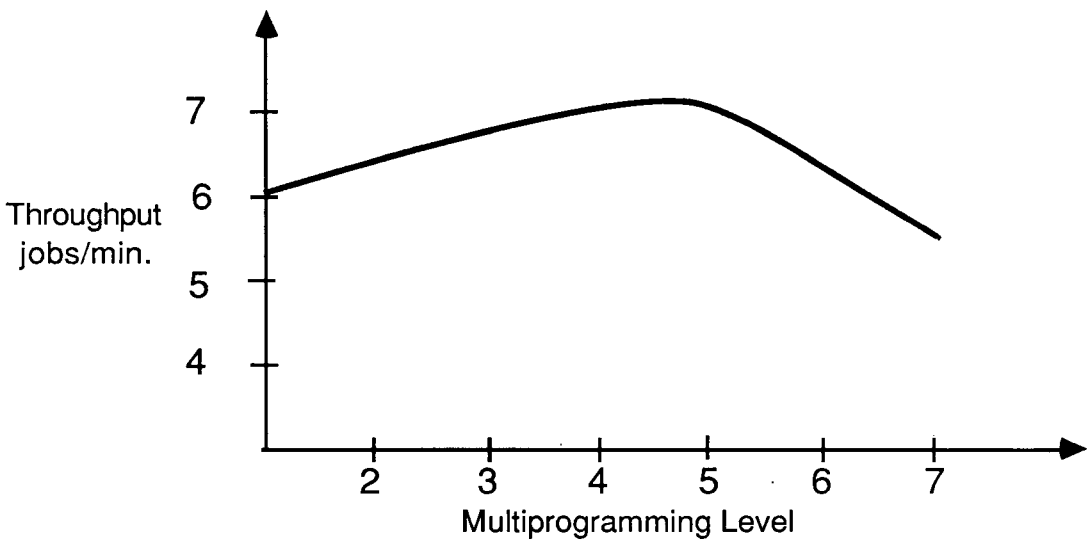

Figure 4.2: Throughput vs. Multiprogramming Level in a Paging System

multiprogramming level, but halting as soon as the throughput obtained at level $M+1$ is less than that at M ($M+1 \leq$ current multiprogramming level). The initial optimal multiprogramming level is then set at M. At every step of 1.2 and 1.5 of the iterative process, optimization is also necessary. The optimization is carried out as outlined below.

If the current optimal multiprogramming level is M when an upgrade possibility is considered, evaluate the system at each multiprogramming level $M, M+1, \ldots, max\text{-}M$, obtaining system throughputs $t_M, t_{M+1}, \ldots, t_{max-M}$. The optimal multiprogramming level is the one corresponding to the maximum throughput rate. However, the evaluation process can stop as soon as $t_{M+n} < t_{M+n-1}$ for some $n > 0$ and $M + n \leq max\text{-}M$. In the latter case, the optimal multiprogramming level with respect to the specific upgrade possibility is $M + n - 1$. Optimization is carried out in OPTIMAL so that at every step the most cost-effective upgrade measure is determined based on the *highest possible* throughput for each upgrade possibility.

4.3.4 Fine Tuning

Balancing service demands on I/O devices while keeping their service rates and the sum of their visit ratios constant may often result in non-integral visit ratios [Ferr83]. One goal of fine tuning is to obtain integral visit ratios based on the non-integral

ones such that the throughput obtained with these visit ratios is at least as high as that obtained with the non-integral visit ratios.[2] As the relationship between throughput and visit ratios is intuitively complex, it is only reasonable to try integral visit ratios in the vicinity of the non-integral ones while keeping the sum of the visit ratios constant. Since the practicability of the *final* recommendations of OPTIMAL is the main concern, fine tuning is required only in the final step of OPTIMAL, rather than after every tuning measure.

4.4 Performance of OPTIMAL

To evaluate the performance of OPTIMAL, a control programme, P2, was set up to enumerate all possible ways of allocating funds to the upgrading alternatives of a given system. Each possible way results in replacement of existing devices by upgrade alternatives and/or addition of new devices into the system. The optimal throughput produced by P2 is the throughput that is maximum over all such possibilities. Similar to the case of evaluating the performance of the Baseline Heuristic, the optimal throughput and the execution time are used as criteria for comparison between OPTIMAL and P2. However, for systems with devices having two or more alternatives that are also candidates to be added to the system, and for a sufficiently large budget, it is obvious that the number of possible ways of allocating the budget is very large.

[2]Empirical data from test examples on fine tuning have shown that throughput at least as high is achievable.

Consequently, for most test systems, OPTIMAL performs far better than P2 with respect to execution time, and thus this criterion is of less interest here. The criterion of primary concern in such systems is the optimal throughput.

Four hypothetical systems of varied complexities have been tested by both OPTIMAL and P2. Their descriptions are shown in Tables 4.1 - 4.8.

- Closed interactive system with memory-constraint.
- Memory management: swapping.
- Disks 1 and 2 can be tuned.
- *System configuration and workload:*
  - $Ncusts = 20$
  - $z = 50.0$s
  - $B = \$120{,}000$
Device i	μ_i	v_i	c_i
CPU	150	351	\$ 5000
Disk 1	50	150	10000
Disk 2	30	200	5000
- *Memory:*
  - $memsize = 16$MB
  - $M = 8$
  - $max\text{-}memsize = 32$MB
  - $max\text{-}M = 15$
  - $c_{memory} = \$5000$

Table 4.1: System 5

Device i	Capability	Cost
CPU:	-	-
Disk 1:		
1	50	$25000
2	60	30000
Disk 2:		
1	50	25000
2	60	30000
Memory:		
1	4	20000
2	6	28000

Table 4.2: System 5 Upgrade Alternatives

- Closed interactive system with memory-constraint.
- Memory management: swapping.
- Disks 1 and 2 can be tuned.
- *System configuration and workload:*
  - $Ncusts = 20$
  - $z = 50.0$s
  - $B = \$130{,}000$
Device i	μ_i	v_i	c_i
CPU	250	621	\$ 5000
Disk 1	50	150	10000
Disk 2	30	200	5000
Disk 3	40	120	4000
Disk 4	45	150	4500
- *Memory:*
  - $memsize = 12$MB
  - $M = 10$
  - $max\text{-}memsize = 32$MB
  - $max\text{-}M = 12$
  - $c_{memory} = \$9000$

Table 4.3: System 6

Device i	Capability	Cost
CPU:	-	-
Disk 1:		
1	55	$25000
2	70	40000
Disk 2:		
1	55	25000
2	70	40000
Disk 3:		
1	55	25000
2	70	40000
Disk 4:		
1	55	25000
2	70	40000
Memory:		
1	2	15000
2	4	20000

Table 4.4: System 6 Upgrade Alternatives

- Closed interactive system with memory-constraint.
- Memory management: swapping.
- Disks 1, 2 and 3 can be tuned.
- *System configuration and workload:*
  - $Ncusts = 30$
  - $z = 60.0$s
  - $B = \$130{,}000$
Device i	μ_i	v_i	c_i
CPU	250	621	\$ 5000
Disk 1	50	150	10000
Disk 2	30	200	5000
Disk 3	40	120	4000
Disk 4	45	150	4500
- *Memory:*
  - $memsize = 12$MB
  - $M = 10$
  - $max\text{-}memsize = 32$MB
  - $max\text{-}M = 12$
  - $c_{memory} = \$9000$

Table 4.5: System 7

Device i	Capability	Cost
CPU:	-	-
Disk 1:		
1	55	$25000
2	70	40000
3	80	60000
Disk 2:		
1	55	25000
2	70	40000
3	80	60000
Disk 3:		
1	55	25000
2	70	40000
3	80	60000
Disk 4:		
1	55	25000
2	70	40000
3	80	60000
Memory:		
1	2	15000
2	4	20000

Table 4.6: System 7 Upgrade Alternatives

- Closed interactive system with memory-constraint.
- Memory management: swapping.
- Disks 1, 2 and 3 can be tuned.
- *System configuration and workload:*
  - $Ncusts = 35$
  - $z = 50.0$s
  - $B = \$130{,}000$
Device i	μ_i	v_i	c_i
CPU	250	621	\$ 5000
Disk 1	50	150	10000
Disk 2	30	200	5000
Disk 3	40	120	4000
Disk 4	45	150	4500
- *Memory:*
  - $memsize = 12$MB
  - $M = 10$
  - $max\text{-}memsize = 32$MB
  - $max\text{-}M = 12$
  - $c_{memory} = \$9000$

Table 4.7: System 8

Device i	Capability	Cost
CPU:	-	-
Disk 1:		
1	55	$25000
2	70	40000
3	80	60000
4	60	30000
Disk 2:		
1	55	25000
2	70	40000
3	80	60000
4	60	30000
Disk 3:		
1	55	25000
2	70	40000
3	80	60000
4	60	30000
Disk 4:		
1	55	25000
2	70	40000
3	80	60000
4	60	30000
Memory:		
1	4	20000
2	6	30000

Table 4.8: System 8 Upgrade Alternatives

Comparison of results from OPTIMAL and P2 in terms of the optimal throughput produced and execution time are shown in Table 5.

	OPTIMAL	P2	Conclusion
System 5			
User time	2.4	42.6	For timing,
System time	0.2	12.8	OPTIMAL is
Elapsed time	00:00:14	00:21:06	slightly superior.
Optimal throughput	0.303972	0.303972	Same
Amount spent	$115000	$115000	Same
Throughput increase per dollar spent ($\times 10^{-6}$)	1.343	1.343	Same
System 6			
User time	7.1	717.0	For timing,
System time	0.1	28.8	OPTIMAL
Elapsed time	00:00:26	00:57:12	is superior.
Optimal throughput	0.259581	0.261927	Differ by 0.89%
Amount spent	$116500	$126500	
Throughput increase per dollar spent ($\times 10^{-6}$)	0.632	0.601	OPTIMAL is superior.
System 7			
User time	57.7	3672.9	For timing,
System time	1.6	377.6	OPTIMAL is
Elapsed time	00:01:15	04:34:53	superior.
Optimal throughput	0.308968	0.311382	Differ by 0.775%
Amount spent	$120500	$125500	
Throughput increase per dollar spent ($\times 10^{-6}$)	0.928	0.910	OPTIMAL is superior.
System 8			
User time	1.5	6021.6	For timing,
System time	1.2	195.4	OPTIMAL is
Elapsed time	00:01:46	13:34:21	superior.
Optimal throughput	0.328815	0.328815	Same.
Amount spent	$125500	$125500	Same.
Throughput increase per dollar spent ($\times 10^{-6}$)	1.049	1.049	Same.

Table 4.9: Comparison of Results from OPTIMAL and P2 applied to Systems 5 to 8

Thus, for the above examples, the improvement in throughput per dollar spent is always equal or higher in OPTIMAL than in P2, and the optimal throughput is within 1.0% from the highest possible for each system.

Chapter 5

Conclusion

5.1 Concluding Remarks

OPTIMAL, an automatic scheme of upgrading a given system to produce optimal system throughput per dollar, constrained by a given budget and available upgrade alternatives, has been implemented and described in the thesis. Various optimization techniques, including optimizing the multiprogramming level and fine tuning, have been used to enhance the performance of OPTIMAL. The backbone heuristic - the Baseline Heuristic - as well as OPTIMAL have been tested with sample systems of varied complexities. Both methods give throughputs within the range 0.0 - 1.0% of the absolute maximum for each system, and with equal or higher improvement in throughput per unit dollar spent as compared to that obtained with the absolute maximum throughput.

5.2 Future Extensions

One of the possible future extensions of OPTIMAL is to incorporate a more sophisticated set of system performance analysis algorithms to increase the scope of application of OPTIMAL to cover more complex systems, such as multiple job class systems, and systems with elaborate I/O subsystems and swapping devices. As suggested in Section 4.1, such algorithms are widely available as packages and can be interfaced with OPTIMAL, replacing the current simple exact MVA system performance analysis programme. The extensions can be easily accommodated as the structure of OPTIMAL is not dependent on the performance analysis algorithm used.

5.2.1 Multiple Job Class Systems

As mentioned in Section 2.1, homogeneity assumptions about the workload are made and currently only systems with single job class are considered. The reason for this is to simplify the workload so as to clearly illustrate the operation of the Baseline Heuristic and OPTIMAL. However, for systems in which the jobs being modelled are not indistinguishable but have significantly different behaviours (for instance systems with a mixture of CPU and I/O bound jobs), a multiple class model can provide more accurate results. To include the multiple job class systems as possible input systems, OPTIMAL can be modified to accept the workload characteristics of each job class, and in Steps 0, 1.2, 1.5 and 4, use exact MVA that models multiple job class systems

to obtain throughput values for each job class, which are then summed to obtain the system throughput. [Lazo84]

5.2.2 I/O Subsystems

Because of advances in VLSI technology, processor, memory and other purely electronic components of the computer system have outpaced the advancement in I/O subsystems technology. Consequently it is more common for current systems to have bottlenecks in I/O components rather than CPU or memory. A common I/O subsystem consists of various elements of varied intelligence: channels, controllers, string heads, disks. Being able to locate the bottleneck element in a highly utilized I/O subsystem is a key to improving system performance. The current implementation of OPTIMAL captures the details of an I/O subsystem in the disk service demand obtained from measurement data. Thus a more sophisticated modelling approach, such as the one suggested in [Lazo84], is necessary to account for the effect of improving each I/O subsystem component on system performance.

5.2.3 Swapping Systems

Similar to I/O subsystems, in order for OPTIMAL to examine the effect of improving the swapping device on system performance, the system performance analysis tool has

to be able to model the swapping device. A possible means has also been suggested in [Lazo84].

Bibliography

[BEST79] *BEST/1 User's Guide*, Release 5.0, BGS Systems, Inc., Lincoln, MA, Aug. 1979

[Buze78] J. Buzen, *Operational Analysis: an Alternative to Stochastic Modeling*, Proc. of International Conference on the Performance of Computer Installation, ICPCI '78, Italy, 1978

[Conn84] W.M. Conner, *An Introduction to Capacity Planning*, CMG '84 Proc. of the International Conference on the Management and Performance Evaluation of Computer Systems, San Francisco, CA, Dec. 1984, pp.586-590

[Coop80] J.C. Cooper, *A Capacity Planning Methodology*, IBM Systems Journal, Vol.19, #1, 1980, pp.28-45

[Denn78] P.J. Denning, J.P. Buzen *The Operational Analysis of Queueing Network Models*, Computing Surveys, Vol.10, #3, Sept. 1978, pp.225-261

[Ferr83] D. Ferrari, G. Serazzi, A. Zeigner, *Measurement and Tuning of Computer Systems*, Prentice Hall, 1983

[Hodg82] L.F. Hodges, *Workload Characterization and Performance Evaluation in a Research Environment*, Proc. of the 1982 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, Vol.11, #4, 1982

[INRI84] *New Users' Introduction to QNAP2*, Version 08/84, INRIA, France, 1984

[Kuma80] S.R. Kumar, R.B. Lake, C.T. Nute, *File Allocation Methodology for Performance Enhancement*, Proc. 16th CPEUG Meeting, 1980, pp.175-188

[Lazo84] E.D. Lazowska, J. Zahorjan, G.S. Graham, K.C. Sevcik, *Quantitative System Performance, Computer System Analysis Using Queueing Network Models*, Prentice Hall, 1984

[Levi85] A.P. Levine, *ESP: an Expert System for Computer Performance Management*, CMG '85 Proc. of the International Conference on the Management and Performance Evaluation of Computer Systems, 1985, pp.181-186

[Lo80] T.L. Lo, *Computer Capacity Planning Using Queuing Network Models*, Proc. of Performance '80 the 7th IFIP W.G. 7.3, International Symposium on Computer Performance Modelling, Measurement and Evaluation, Toronto, Ont., May 1980

[Maed85] A.T. Maeda, L.A. Aho, *Using Expert System Technology to Evaluate System Resource Planning Alternatives: a Management Perspective*, CMG '85 Proc. of the International Conference on the Management and Performance Evaluation of Computer Systems, 1985, pp.558-563

[Pang86] J.F. Pang, *Characterizing User Workload for Capacity Planning*, M.Sc. Thesis, U. of British Columbia, 1986

[Reis80] M. Reiser, S.S. Lavenberg, *Mean Value Analysis of Closed Multichain Queueing Networks*, JACM, Vol.27, #2, April 1980, pp.312-322

[Orch83] R.A. Orchard, *Algebraic Models for CPU Sizing*, CPEUG 83 Proc. of the Computer Performance Evaluation Users Group 19th Meeting, San Francisco, CA, Oct. 1983, pp.116-134

[Ostr85] J.M. Ostrowski, J.L. Webb, T.A. Hilton, *Implementing Expert System Technology into the Field of Computer Performance Evaluation*, CMG '85 Proc. of the International Conference on the Management and Performance Evaluation of Computer Systems, 1985, pp.245-250

[Sala80] K.O. Salawu, *A Note on Computer System Capacity Planning Through Material Requirements Planning*, CPEUG 80 Proc. of the Computer Performance Evaluation Users Group 16th Meeting, 1980, pp.199-203

[Sala85] K.O. Salawu, *Capacity Prediction for UNIX Systems*, CMG '85 Proc. of the International Conference on the Management and Performance Evaluation of Computer Systems, 1985

[Sevc80] K.C. Sevcik, G.S. Graham, J. Zahorjan, *Configuration and Capacity Planning in a Distributed Processing System*, Proc. 16th CPEUG Meeting, 1980, pp.165-171

Appendix A

OPTIMAL User's Guide

OPTIMAL [1] can be invoked [2] on-line by the command: **optimal.**

OPTIMAL consists of three phases: the *input phase*, the *processing phase* and the *output phase.* The *input phase* of OPTIMAL is a question-and-answer session; input and output are directed through the terminal.

The type of system and input parameters that OPTIMAL considers are described in Table A.1 below.

[1] The source of the current implementation requires approximately 118K, and the object 115K.

[2] OPTIMAL as running on a UNIX-based system.

System Configuration : centralized
Workload : batch or interactive
Memory Management : swapping or paging; with or without memory constraint
Input Parameters : as listed below; units are not fixed but must be consistent (e.g. *seconds* for mean thinktime, and *jobs per second* or *MIPS* for CPU capability)

System Configuration	Workload	Economic
For all systems: 1. No. of service centres 2. Paging or swapping 3. Memory-constrained or not 4. Whether tuning is allowed, and if so, which are the devices on which tuning is allowed	For all systems: 1. Mean no. of customers 2. Mean thinktime	For all systems: 1. Budget 2. For each device to be considered for upgrading, its market value, no. of upgrade alternatives available, and their capabilities and costs
For paging systems: service rates for non-CPU, non-memory devices, and mean instruction execution rate for CPU. For non-paging systems: service rates for all non-memory devices	For paging systems: visit ratios for non-CPU, non-paging, non-memory devices, and mean no. of instructions per program for CPU. For non-paging systems: visit ratios for all non-memory devices	
For memory-constrained systems: max. memory size, current memory size, maximum degree of multiprogramming supported		

Table A.1: System and Input Parameters for OPTIMAL

After the initial input session and before processing begins, adjustments to prior input values are possible. Table A.2 shows a list of adjustment codes for the corresponding adjustments. Note that adjustments for corresponding adjustment codes are carried out *as described*, except for the following where related adjustments are also prompted for:

4: if system *is* paging, then OPTIMAL also prompts for descriptions of the paging device.

5: if system *is* memory-constrained, then OPTIMAL also prompts for descriptions of the memory constraint.

9: if number of devices on which tuning [3] is allowed is > 1, then OPTIMAL also prompts for the identities of these devices.

16: if number of upgrade alternatives for a device is > 0, then OPTIMAL also prompts for descriptions of the upgrade alternatives

When no further adjustments are necessary, inputting adjustment code **20** ends the adjustment process and begins the *processing phase.*

[3] Files are allowed to be relocated among these devices.

adjustment code	adjustment
1	number of customers
2	mean thinktime
3	number of devices
4	system is / not paging
5	system is / not memory-constrained
6	instruction execution rate of CPU / service rate of a device
7	visit ratio of a device
8	paging device is / not shared
9	number of devices on which tuning is allowed
10	maximum memory size supported by system
11	maximum degree of multiprogramming supported
12	current main memory size
13	mean degree of multiprogramming
14	budget
15	market value of a device
16	number of upgrade alternatives for a device
17	lifetime function
20	no further adjustments

Table A.2: Adjustments and Corresponding Adjustment Codes for the Input Phase of OPTIMAL

The *processing phase* of OPTIMAL is a completely automated process.

At the end of the *processing phase*, OPTIMAL enters the *output phase*. Output of OPTIMAL includes:

- optimal system throughput, with or without fine-tuning
- the recommended system configuration, including:
  - number of added devices
  - for each device, its capability, visit ratio where applicable, the recommended amount to be spent on it, and the equivalent device type in the initial system (EqD) (this information is needed if the device is a new device to be added onto the system)
  - for paging systems, the recommended multiprogramming level
- the budget required and the remaining budget

A sample session of OPTIMAL, showing the *input* and *output* phases of System 9 (see Table A.3, Figure A.1, Table A.4), is illustrated below. This session attempts to illustrate most of the features of OPTIMAL.

- Closed interactive system with memory-constraint.
- Memory management: paging.
- Disks 2 and 3 can be tuned.
- *System configuration and workload:*
  - $Ncusts = 15$
  - $z = 50.0$s
  - $L = 5$ MI
  - $B = \$30{,}000$
  -

Device i	capability	v_i	c_i
CPU	1 MIPS	-	\$ 1000
Disk 1	30 jobs/s	120	2500
Disk 2	30 jobs/s	150	2500
Disk 3	40 jobs/s	160	3000
Shared paging disk	45 jobs/s	90	3000

- *Memory:*
  - $memsize = 300$ page frames
  - $M = 10$
  - $max\text{-}memsize = 1500$ page frames
  - $max\text{-}M = 12$
  - $c_{memory} = \$500$

Table A.3: System 9

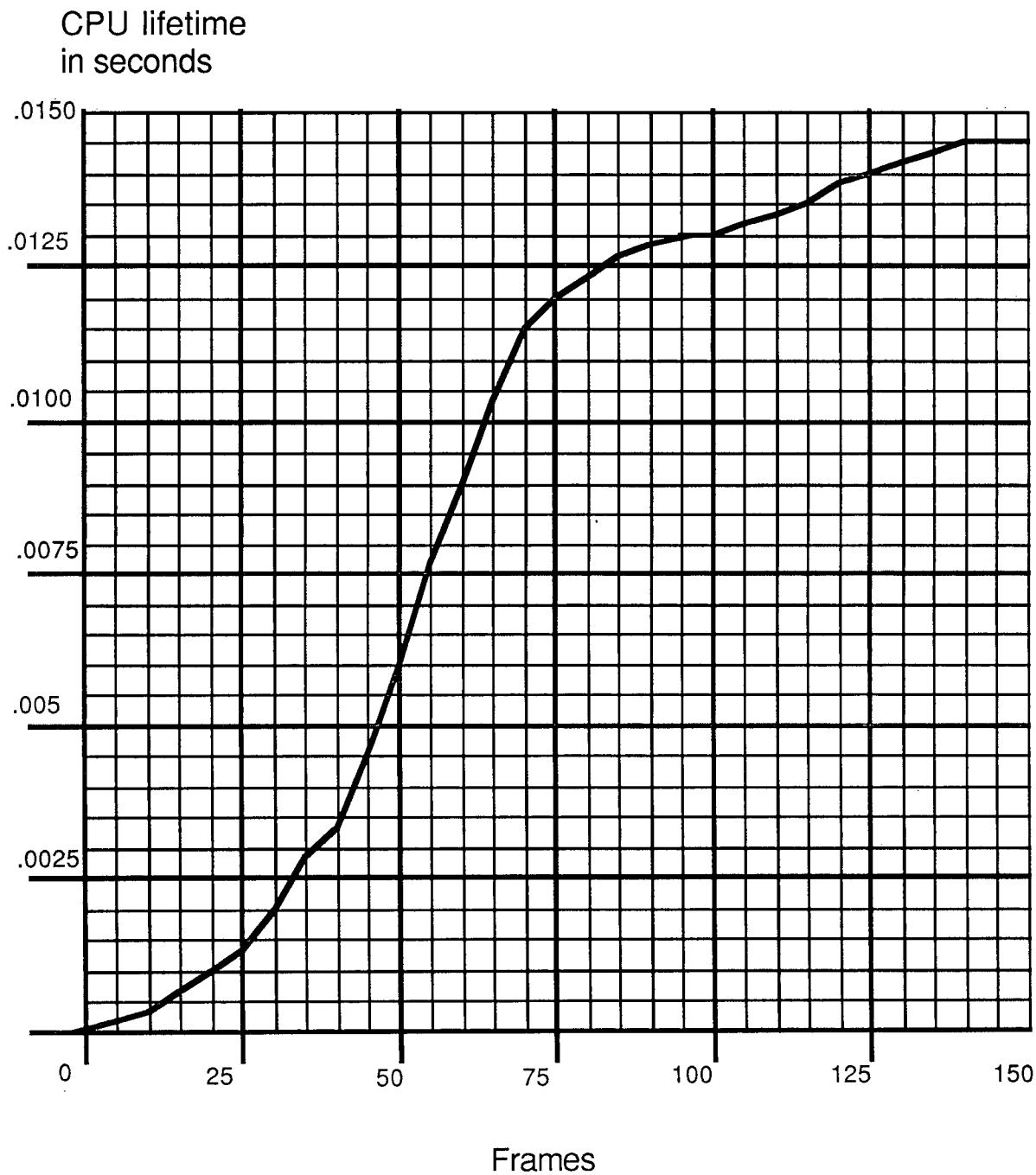

Figure A.1: Lifetime Curve Representing the Paging Activities of System 9

Device i	Capability	Cost
CPU:	2 MIPS	$4500
Disk 1:		
1	25	3000
2	40	4500
Disk 2:		
1	35	3000
2	40	4500
Disk 3:		
1	35	3500
2	50	5500
Shared paging disk:		
1	50	5500
Memory:		
1	500	2000

Table A.4: System 9 Upgrade Alternatives

>optimal [4]

```
**************************************
*    System and Workload Parameters    *
**************************************

Enter number of customers: 15

Enter average think time of customers (sec.): 40

Enter number of devices excluding memory: 5

Is there a paging device? (0 = 'no', 1 = 'yes') 1

Is memory to be evaluated? (0 = 'no', 1 = 'yes') 1

For the following, device 1 must be the CPU
and device 5 the paging device
and device 6 main memory
```

[4]Invoke OPTIMAL.

```
*********
*  CPU  *
*********

Enter mean no. of instructions to be executed per programme: 5000000

Enter mean CPU instruction execution rate: 1000000

************************
*  On-line I/O Device  *
************************

Enter visit ratio for device 2 : 120

Enter service rate for device 2 (transactions/sec.) :30
```

```
          ************************
          *  On-line I/O Device  *
          ************************

Enter visit ratio for device 3 : 150

Enter service rate for device 3 (transactions/sec.) :30

          ************************
          *  On-line I/O Device  *
          ************************

Enter visit ratio for device 4 : 160

Enter service rate for device 4 (transactions/sec.) :40

            ********************
            *  Paging Device  *
```

```
          ******************

Is paging device shared? (0 = 'no', 1 = 'yes') 0

Enter service rate for paging device (transactions/sec.) : 45

             ************
             *  Memory  *
             ************

Enter maximum memory size possible (e.g.KB, page frames): 1500

Enter maximum degree of multiprogramming supported: 12

Enter main memory size (e.g.KB, page frames): 300

Enter mean degree of multiprogramming: 10
```

```
**********************
*  Lifetime Function  *
**********************
```

The lifetime function is input by assigning the maximum abscissa (programme size), the interval on the horizontal axis between points at which the ordinates of the lifetime curve are to be given (step), and the ordinates corresponding to these points (mean time between page faults).

```
Enter maximum abscissa (e.g.KB, page frames): 150

Enter step size for lifetime function: 5

For each of the following mean memory allocation per job,
enter mean time between page faults:
0: 0
```

```
5: 0.0002

10: 0.0004

15: 0.0007

20: 0.001

25: 0.0014

30: 0.002

35: 0.00273

40: 0.0034

45: 0.004

50: 0.006
```

```
55: 0.0077

60: 0.009

65: 0.0104

70: 0.0114

75: 0.0119

80: 0.01225

85: 0.0126

90: 0.01275

95: 0.01285

100: 0.013
```

105: 0.01325

110: 0.01335

115: 0.0136

120: 0.01375

125: 0.014

130: 0.0142

135: 0.01438

140: 0.0145

145: 0.0145

150: 0.0145

```
            ************
            *  Tuning  *
            ************

Enter number of devices on which tuning is allowed: 2

Enter their device numbers,
each followed by a RETURN: 3
4

*****************************
*     Economic Parameters     *
*****************************

Enter budget (dollars): 30000

Enter market value of device 1 (dollars): 1000

Enter no. of alternatives available for upgrading device 1: 1
```

```
For alternative 1 for device 1,

enter its capability (transactions/sec.): 2000000


enter its price (dollars): 4500


Enter market value of device 2 (dollars): 2500


Enter no. of alternatives available for upgrading device 2: 0


Enter market value of device 3 (dollars): 2500


Enter no. of alternatives available for upgrading device 3: 2


For alternative 1 for device 3,

enter its capability (transactions/sec.): 35


enter its price (dollars): 3000


For alternative 2 for device 3,
```

```
enter its capability (transactions/sec.): 40

enter its price (dollars): 4500

Enter market value of device 4 (dollars): 3000

Enter no. of alternatives available for upgrading device 4: 2

For alternative 1 for device 4,
enter its capability (transactions/sec.): 35

enter its price (dollars): 3500

For alternative 2 for device 4,
enter its capability (transactions/sec.): 50

enter its price (dollars): 5500

Enter market value of device 5 (dollars): 3000
```

```
Enter no. of alternatives available for upgrading device 5: 1


For alternative 1 for device 5,

enter its capability (transactions/sec.): 50


enter its price (dollars): 5000


Enter market value of device 6 (dollars): 500


Enter no. of alternatives available for upgrading device 6: 1


For alternative 1 for device 6,

enter its capability (e.g.KB, page frames): 500


enter its price (dollars): 2000
```

```
**********************
*    Adjustments     *
**********************
```

Adjustments and corresponding adjustment codes:

```
*************************************************************************
*  1 * number of customers          * 11 * max. degree of               *
*    *                              *    * multiprogramming             *
*  2 * mean thinktime               * 12 * current main mem. size       *
*  3 * number of devices            * 13 * mean degree of               *
*    *                              *    * multiprogramming             *
*  4 * system is/ not paging        * 14 * budget                       *
*  5 * system is/ not memory-       * 15 * market value of device       *
*    * constrained                  *    *                              *
*  6 * instruction execution        * 16 * number of upgrade            *
*    * rate of CPU / service        *    * alternatives for a device    *
*    * rate of a device             *    *                              *
*  7 * visit ratio of device        * 17 * lifetime function            *
*  8 * paging device is/ not shared* 20 * no further adjustments       *
```

```
*  9 * no.of tuneable devices      *    *                          *

* 10 * max. memory size            *    *                          *

*****************************************************************

Enter adjustment code: 2


Thinktime entered is 40.000000 (sec.)

Is entered information correct? (0 = 'no', 1 = 'yes') 0


Enter thinktime (sec.): 50
```

```
Adjustments and corresponding adjustment codes:

*****************************************************************

*  1 * number of customers          * 11 * max. degree of          *
*    *                              *    * multiprogramming        *
*  2 * mean thinktime               * 12 * current main mem. size  *
*  3 * number of devices            * 13 * mean degree of          *
*    *                              *    * multiprogramming        *
*  4 * system is/ not paging        * 14 * budget                  *
*  5 * system is/ not memory-       * 15 * market value of device  *
*    * constrained                  *    *                         *
*  6 * instruction execution        * 16 * number of upgrade       *
*    * rate of CPU / service        *    * alternatives for a device*
*    * rate of a device             *    *                         *
*  7 * visit ratio of device        * 17 * lifetime function       *
*  8 * paging device is/ not shared* 20 * no further adjustments   *
*  9 * no.of tuneable devices       *    *                         *
* 10 * max. memory size             *    *                         *

*****************************************************************

Enter adjustment code: 7
```

```
Enter device number: 5


Visit ratio for device 5 entered is 0.000000

Is entered information correct? (0 = 'no', 1 = 'yes') 0


Enter visit ratio for device 5 : 90
```

```
Adjustments and corresponding adjustment codes:

*****************************************************************

*  1 * number of customers          * 11 * max. degree of          *
*    *                              *    * multiprogramming        *
*  2 * mean thinktime               * 12 * current main mem. size  *
*  3 * number of devices            * 13 * mean degree of          *
*    *                              *    * multiprogramming        *
*  4 * system is/ not paging        * 14 * budget                  *
*  5 * system is/ not memory-       * 15 * market value of device  *
*    * constrained                  *    *                         *
*  6 * instruction execution        * 16 * number of upgrade       *
*    * rate of CPU / service        *    * alternatives for a device*
*    * rate of a device             *    *                         *
*  7 * visit ratio of device        * 17 * lifetime function       *
*  8 * paging device is/ not shared* 20 * no further adjustments  *
*  9 * no.of tuneable devices       *    *                         *
* 10 * max. memory size             *    *                         *

*****************************************************************

Enter adjustment code: 16
```

```
Enter device number: 2

No. of alternatives available for upgrading device 2 entered is 0
Is entered information correct? (0 = 'no', 1 = 'yes') 0

Enter no. of alternatives available for upgrading device 2: 2

For alternative 1 for device 2,
enter its capability  (transactions/sec.): 25

enter its price (dollars): 3000

For alternative 2 for device 2,
enter its capability  (transactions/sec.): 40

enter its price (dollars): 4500
```

```
Adjustments and corresponding adjustment codes:

*****************************************************************

*  1 * number of customers          * 11 * max. degree of          *
*    *                              *    * multiprogramming        *
*  2 * mean thinktime               * 12 * current main mem. size  *
*  3 * number of devices            * 13 * mean degree of          *
*    *                              *    * multiprogramming        *
*  4 * system is/ not paging        * 14 * budget                  *
*  5 * system is/ not memory-       * 15 * market value of device  *
*    * constrained                  *    *                         *
*  6 * instruction execution        * 16 * number of upgrade       *
*    * rate of CPU / service        *    * alternatives for a device*
*    * rate of a device             *    *                         *
*  7 * visit ratio of device        * 17 * lifetime function       *
*  8 * paging device is/ not shared* 20 * no further adjustments  *
*  9 * no.of tuneable devices       *    *                         *
* 10 * max. memory size             *    *                         *

*****************************************************************

Enter adjustment code: 16
```

```
Enter device number: 5

No. of alternatives available for upgrading device 5 entered is 1
For alternative 1 for device 5,
capability entered is 50.000000, (transactions/sec.)
price entered is 5000.000000 (dollars)
Is entered information correct? (0 = 'no', 1 = 'yes') 0

Enter no. of alternatives available for upgrading device 5: 1

For alternative 1 for device 5,
enter its capability  (transactions/sec.): 50

enter its price (dollars): 5500
```

```
Adjustments and corresponding adjustment codes:

*****************************************************************

*  1 * number of customers          * 11 * max. degree of          *
*    *                              *    * multiprogramming        *
*  2 * mean thinktime               * 12 * current main mem. size  *
*  3 * number of devices            * 13 * mean degree of          *
*    *                              *    * multiprogramming        *
*  4 * system is/ not paging        * 14 * budget                  *
*  5 * system is/ not memory-       * 15 * market value of device  *
*    * constrained                  *    *                         *
*  6 * instruction execution        * 16 * number of upgrade       *
*    * rate of CPU / service        *    * alternatives for a device*
*    * rate of a device             *    *                         *
*  7 * visit ratio of device        * 17 * lifetime function       *
*  8 * paging device is/ not shared* 20 * no further adjustments   *
*  9 * no.of tuneable devices       *    *                         *
* 10 * max. memory size             *    *                         *

*****************************************************************

Enter adjustment code: 20
```

```
***Processing begins ... optimal solution is ...

                    **********************
                    *   Recommendations  *
                    **********************

*************************************************************************
* Optimal system throughput (not fine-tuned):   0.166517               *
* Optimal system throughput (fine-tuned):       0.166536               *
*-----------------------------------------------------------------------*
* Recommended system configuration:                                     *
*-----------------------------------------------------------------------*
* Number of added non-memory devices:                3                  *
* Their device numbers:                              5                  *
*                                                    6                  *
*                                                    7                  *
*                                                                       *
```

```
* Paging device is now device 8.                                    *
*********************************************************************
* Dev * EqD  *   Service Rate   *   Visit Ratio    *  Amount Spent  *
*********************************************************************
*  1  *   1  *    2000000.00    *       n/a        *      3500.00   *
*  2  *   2  *         40.00    *       120        *      2000.00   *
*  3  *   3  *         40.00    *        57        *      2000.00   *
*  4  *   4  *         50.00    *        74        *      2500.00   *
*  5  *   3  *         40.00    *        57        *      4500.00   *
*  6  *   4  *         50.00    *        72        *      5500.00   *
*  7  *   3  *         35.00    *        50        *      3000.00   *
*  8  *   5  *         50.00    *        90        *      2500.00   *
*********************************************************************
* Recommended memory size:                1300      *      4000.00   *
* Recommended multiprogramming level:        9      *                *
*********************************************************************
* Total budget required:                                  29500.00   *
* Remaining budget:                                         500.00   *
*********************************************************************
```

```
Upgrade completed.
```