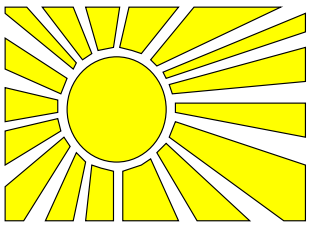
EPCC-SSP 1996

Applications of Parallel Processing in Management Science and Operational Research

D. P. Lee

Abstract

This paper examines the range of possible solution approaches within the areas of Operational Research (OR) and Management Science (MS) that could benefit from the application of High Performance Computing (HPC) utilizing a parallel processing paradigm implemented with Message Passing Interface (MPI). The results include an overview of the Infinite Horizon Discounted Markov Process (IHDMP) using Pre-Jacobi (PJ) Iteration with Porteus Bounds (PB)and a comparism of the results partially differentiated with respect to an isolated operational parameter, demonstrating how the variation of input affects the final solution set. This paper includes a graphical demonstration of the relationship between the monotonic and contraction properties of the IHDMP and the prime number system describing the Regular Rectilinear Markov Primes (RRMPS). Also described is a comparism of the computational performance of the serial and parallel algorithms.



1 Introduction

This paper deliberates upon the efficient implementation and subsequent operation of infinite horizon discouted Markov decision processes utilizing value iteration type solution algorithms. Although other algorithms are discussed and their properties expounded, here we shall be dealing primarily with the Pre-Jacobi (PJ) or optimal equation algorithm, using Porteous Bounds (PB), and the PJ numerical properties and solutions. References to the problem include [1,2,3,4] which tabulate solutions to deterministic and probalistic models relating to inventory control, [8,9,10] which include solutions to and analysis of up to six problems of a serial to parallel nature, each containing different properties relating the performance of an algorithm to the properties of the problem being solved and [6,7] which describe a detailed mathematical and algorithmic analysis of Markov Chains including a theoretical study, theorems and proofs. In this paper we consider one problem concerning the inventory control of a fixed state warehouse with *n* input parameters, holding *n*-1 values constant whilst varying 1 over a fixed range in order to elicit information regarding the relative properties of the optimum value policy and the optimum value function. The process is the computational equivalent of partially differentiating the function singularly with respect to all available parameters. We also address the parallel solution of Markov decision processes, practical examples of which are often computationally unmanageble on a serial machine, owing to memory allocation and execution time. The results indicate that parallel computors allow faster solutions and larger problem pools to be solved. Deeper theoretic and graphical investigation describes the relationship between the prime numbers and the monotonic and contraction properties of the optimum policy function created by the Markov Decision Process.

1.1 The Markov Decision Process (MDP)

The Markov decision processes which we consider can be described as follows.

The state space S = {1,2, ... |S|} is finite.
At any stage, when in state *i*, an action *k* can be chosen from the action space k_i and an immediate reward r_i^k received. future rewards are discounted with factor β ,where $0 \leq \beta < 1$.
The probability that the process will be in state *j* at the next stage is given by the transition probability p_{ij}^k
This paper only considers problems in which the action spaces are finite and r_i^k and p_{ij}^k are independent of stage.
The maximum discounted reward over an infinite horizon starting in state *i* is denoted by $V^*(i)$ is the unique solution of

$$V_i^n = \max_{k \in k_i} \left\{ r_i^k + \beta \sum_{j \in S} p_{ij}^k V_j^{n-1} \right\} \tag{1}$$

A policy is an assignment of an action to each state. A policy with discounted reward equal to V^* is termed an optimal policy and a policy δ with discounted reward V^δ is termed ε-optimal if

$$\|V^* - V^\delta\|_\infty < \varepsilon, \tag{2}$$

where

$$\|V\|_\infty = max_i(V(i)). \tag{3}$$

V^δ satisfies the above equation with k_i restricted to the assigned action for state i. Given a small tolerance, the aim is to find an ε-optimal policy and vectors of bounds V^- and V^+, such that

$$V^- \leq V^* \leq V^+ \tag{4}$$

and

$$\|V^+ - V^-\|_\infty < \varepsilon, \tag{5}$$

1.2 The Computing Envirionments

The computational results were obtained using a CrayT3D and a T800 Meiko Computing Surface.
The T800 Meiko Computing Surface contains 130 nodes. Each T800 node has at least 4 Mbytes of memory and a peak speed of around 1 Mflop. The nodes are divided into domains which range in size from 2 to 130 processors.
The Cray T3D possesses 512 Dec Alpha processors with a peak performance of 40 Gflops. The T3D has 20 Gbytes of main memory, 212 Gbytes of disc and a Multi-Terabyte mass storage device. A recent addition to the system is a 10 pe J90 with around 1 Gbyte of central memory, intended as a pre-processing and post-processing facility for the T3D.

1.3 Overview

In section 2, the algorithms generally used to solve infinite horizon discounted Markov decision processes are discussed including the stopping rules and conditions, the algorithmic selection and the method of storage. The associated problem parameters are described in section 3, followed by their relationship with the Markov Decision Process formulation in section 4. the serial implementation is described in section 5, with its parallel counterpart in section 6. Sections 7 and 8 deliniate the partitioning, efficiency and major factors affecting Markov Decision Processes respectively. The serial and parallel results are tabulted in section 10, leading to the implementation of the parallel optimisation program in section 11, the results of which are contained in section 12. Section 13 describes the final conclusions, followed by the article references.

2 Description of the Algorithms

The following algorithms are those most widely used and are the best candidates for solution methods with respect to infinite horizon discounted Markov Decision Processes because the properties of Markov Decision Processes guarantee convergence in most cases and the matrices for policies are often large and sparse, the conditions under which transition iterative methods

are often preferred for systems of linear equations. The first three algortihms have proved to be the most efficient.[10]

Pre-Jacobi (PJ) :

$$V_i^n = \max_{k \in k_i} \left\{ H_i^k(V_{n-1}) \right\} \tag{6}$$

where

$$H_i^k(v) = \left\{ r_i^k + \beta \sum_{j \in S} p_{ij}^k V_j^{n-1} \right\} \tag{7}$$

The Pre-Jacobi is also referred to as value iteration, succesive approximation or the optimal equation. The PJ is rarely used to solve systems of linear equations.

Gauss-Seidel (GS) :

$$V_i^n = \max_{k \in k_i} \left\{ \frac{r_i^k + \beta \left\{ \sum_{j<i} p_{ij}^k V_n(j) + \sum_{j>i} p_{ij}^k V_{n-1}(j) \right\}}{1 - \beta p_{ii}^k} \right\} \tag{8}$$

Unlike in PJ and Jacobi where the components of V_i^n can be calculated independently of each other, in Gauss-Seidel the components of V_i^n have to be calculated in the order of increasing state.

Successive Over-Relaxation (SOR) with relaxation parameter $\omega > 1$:

$$\omega \left\{ V_i^n = \max_{k \in k_i} \left\{ \frac{r_i^k + \beta \left\{ \sum_{j<i} p_{ij}^k V_n(j) + \sum_{j>i} p_{ij}^k V_{n-1}(j) \right\}}{1 - \beta p_{ii}^k} \right\} \right\} + (1 - \omega) V_{n-1}(i). \tag{9}$$

With ω =1, Successive Over-Relaxation is equivalent to GS. The SOR normally possesses no monotonic or contraction properties.

Jacobi (J)

$$V_i^n = \max_{k \in k_i} \left\{ \frac{r_i^k + \beta \sum_{j \neq i} p_{ij}^k V_{n-1}(j)}{1 - \beta p_{ii}^k} \right\} \tag{10}$$

Since the convergence properties of Jacobi are known to poor, it will be given no further consideration, other than for the purpose of algorithmic comparism.

2.1 Stopping Rules

In order to terminate these iterative procedures, upper and lower bounds V^{up} and V_{low}, are required. These bounds are calculated from the sequence of values contained in the Value function vector, $\{ V^n \}$. Here we will use the method of the Porteus Bounds to solve the halting problem.

2.2 The Porteus Bounds

The Porteus Bounds (PB) exploit the monotonic and contraction properties of the mappings in PJ, J and GS. Since SOR does not generally have these properties, the PB cannot be applied to this algorithm.

$$\|V^*(i) - V_n(i)\|_\infty \leq \frac{\beta}{1-\beta} \left\{ \max_{i \in S} \{V_n(i) - V_{n-1}(i)\} - \min_{i \in S} \{V_n(i) - V_{n-1}(i)\} \right\} < \epsilon \quad (11)$$

With the bounds as shown, the required convergence criterion in equation (5) is satisfied.[10] Thus, varying the input parameter epsilon $\{\epsilon\}$, will adjust the convergence and any related properties.

2.3 Selecting the Optimum Algorithm

In all the algorithms if the k_i contains a single action for all *i*, then those schemes are equivalent to the standard schemes for solving linear equations. Since the update rules in all of the above schemes are linear, the value vector V_n may be expressed in terms of V_{n-1} by V_n = MV_{n-1}. M is termed the iteration matrix for the policy which is selected at the n^{th} iteration. The convergence of the different schemes is largely determined by the eigen-structure of the iteration matrices used. The iteration matrix M for PJ has the form βP where P is the transition matrix for the current selected policy. The eigenvalues of βP are β times the eigenvalues of P and the eigenvectors are the same. Since P is a stochastic matrix it has largest eigenvalue 1 and corresponding eigenvector (1,1, ... ,1)^T (The iteration matrices in PJ are not strictly stochastic, but are equal to a scalar multiple of a stochastic matrix and possess similar properties. These results are exploited by the PB with the result that the convergence of PJ using stopping rule PB is determined by the sub-radius of the transition matrices for the policies used during the algorithm.(The sub-radius of a matrix is the modulus of the eigenvalue with 2^{nd} largest modulus). The iteration matrices of the other schemes do not possess this stochastic property and the PB are therefore less efficient.[8] For this reason we will define the solution implementation using Pre-Jacobi using Porteus Bounds. vspace0.2in

2.4 Data Structures

The probability distribution will be of form poisson and is necessarily truncated, since probabilities less than some specified tolerance ϵ are ignored and zero values are therefore created.

It is inefficient to store transition probabilities that are equal to zero. The most efficient implementations to date use 'indirect addressing'. the immediate rewards and non-zero transition probabilities are stored in a One-Dimensional array called DATA. The addressing information is contained in three One-Dimensional arrays, STATE, ACTION and MOVE.

For action k $\in k_i$ in state i $\in$S,

$$r_i^k = \text{DATA (ACTION (STATE (i+1) + k + 1))}$$

$$p^k_{i,MOVE(j)-1} = \text{DATA (j)}$$

wheres ACTION (STATE (i+1) + k + 1) +1 $\leq$ j $\leq$ MOVE (ACTION (STATE (i+1) + k + 1))

Example with CAPACITY = 2

STATE	1	5	8

ACTION	4	1	3	6	7	10	13	9	17

MOVE	2	0^{+1}	5	1^{+1}	0^{+1}	9	2^{+1}	1^{+1}	0^{+1}	12	1^{+1}	0^{+1}	16	2^{+1}	1^{+1}	0^{+1}	20	2^{+1}
DATA	r_0^0	$p^0_{0,0}$	r_0^1	$p^1_{0,1}$	$p^1_{0,0}$	r_0^2	$p^2_{0,2}$	$p^2_{0,1}$	$p^2_{0,0}$	r_1^0	$p^0_{1,1}$	$p^0_{1,0}$	r_1^1	$p^1_{1,2}$	$p^1_{1,1}$	$p^1_{1,0}$	r_2^0	$p^0_{2,2}$

S = $\{0,1,2\}$ $K_0 = \{0,1,2\}$ $K_1 = \{0,1\}$ $K_2 = \{0\}$

$P^0_{0,j}$=0 if j$\neq$0 $P^0_{1,j}$=0 if j$\neq$0 or 1 $P^0_{2,j}$=0 if j$\neq$0,1 or 2

$P^1_{0,j}$=0 if j$\neq$0 or 1 $P^1_{1,j}$=0 if j$\neq$0,1 or 2

$P^2_{0,j}$=0 if j$\neq$0,1 or 2

3 Description of the Problem Parameters

BETA Discount factor.

TOLERANCE Determines the stopping criterion.

CAPACITY Maximum number of items that can be stored.

LEAD Supply lead time = time between placing an order and receiving the goods. Assumed to be either 0 or 1 period.

FIXED Fixed order cost.

UNIT Unit cost per item ordered.

HOLD Cost of holding stock per item per period. Expressed as a percentage of unit cost. Applied to items in stock immediately before a replenishment opportunity.

STOCKOUT Cost per item of demand that cannot be met from inventory.

LAMBDA Parameter for the Poisson model of demand = Mean demand per period.

Assumption: Time between successive replenishment opportunities = 1 period.

Initially, this example has a storage capacity of 119 individual units implying 120 states. The fixed order cost is £10, unit cost £0.50, holding cost £0.035 per unit per period and stockout cost £20 per unit. The lead time is one period and the demand per period is modelled by a Poisson

distribution with mean 2.5 units per period. The discount factor, 0.95 and the tolerance 0.001 are used in the evaluation of the Porteus bounds to allow probabilities of less than 0.00005 to be ignored.

4 Relationship with General MDP Formulation

State space: S = {0,1, ... ,CAPACITY}

Action space in state i: K_i = {0,1, ... ,CAPACITY}

Immediate cost when action k is taken in state i:

with LEAD = 0:

$$r_i^k = \delta(k)FIXED+kUNIT+\beta\sum_{d=0}^{i+k-1}\frac{e^{-\lambda}\lambda^d}{d!}(i+k-d)HOLD+\beta\sum_{d=i+k+1}^{\infty}\frac{e^{-\lambda}\lambda^d}{d!}(d-i-k)STOCKOUT \tag{12}$$

with LEAD = 1:

$$r_i^k = \delta(k)FIXED+kUNIT+\beta\sum_{d=0}^{i-1}\frac{e^{-\lambda}\lambda^d}{d!}(i-d)HOLD+\beta\sum_{d=i+1}^{\infty}\frac{e^{-\lambda}\lambda^d}{d!}(d-i)STOCKOUT \tag{13}$$

Probability of being in state j next period when action k is taken in state i:

witn LEAD = 0:

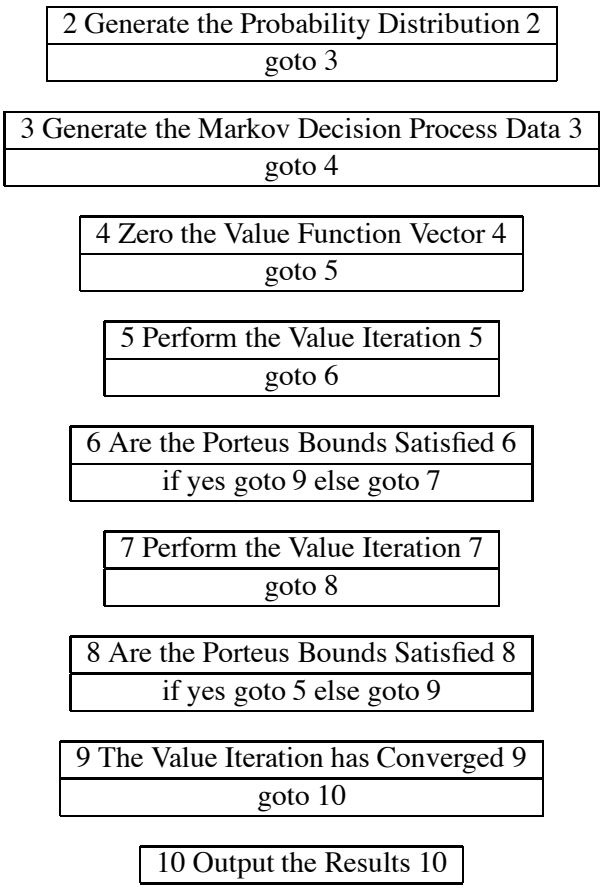
$$p_{i,j}^k = \begin{cases} \frac{e^{-\lambda}\lambda^{i+k-j}}{i+k-j} & if\ \ 0 < j \leq i+k \\ \sum_{d=i+k}^{\infty}\frac{e^{-\lambda}\lambda^d}{d!} & if j = 0 \\ 0\ otherwise & \end{cases} \tag{14}$$

with LEAD = 1:

$$p_{i,j}^k = \begin{cases} \frac{e^{-\lambda}\lambda^{i+k-j}}{i+k-j} & if\ \ k < j \leq i+k \\ \sum_{d=i+k}^{\infty}\frac{e^{-\lambda}\lambda^d}{d!} & if j = k \\ 0\ otherwise & \end{cases} \tag{15}$$

5 Implementing the Serial Program

1 Read the Problem Parameters 1
goto 2



6 Implementing the Parallel Program

The general principle used to satisfy the distribution reqirements of chain, torus and ring topologies is to partition the state space and allow efficient allocation of a specific set of states to each processor in the network. Each processor will then be responsible only for the part of the calculation which is specific to its allocated states. This method reduces both the calculation time and the amount of data which is required to be stored on any single processor. The Value Function Vector, V_n, the vector of the completed correct values with respect to the final unconditional probabilities is the only data which needs to be duplicated on every processor. This is because every processor performing a calculation involving the Value Function Vector accesses it much more often than it is required to be updated, reducing the communications time.

6.1 Distribution

Given a network of N processors and a partition $\{ P_t \}_{t=1}^{N}$ of the state space S, allocate the r^{th} set in the partition to the r^{th} processor in the network. Initially all processors have a copy of V_0 = 0. At the n^{th} iteration, (n $\geq$ 1) the r^{th} processor calculates V_n(i) $\forall$ i $\in P_r$,

$$max_{i \in P_r}(V_n(i) - V_{n-1}(i)) \; and \; min_{i \in P_r}(V_n(i) - V_{n-1}(i)) \tag{16}$$

and then sends these values to all other processors in the network so that each processor can calculate

$$max_{i \in S}(V_n(i) - V_{n-1}(i)) \; and \; min_{i \in S}(V_n(i) - V_{n-1}(i)) \tag{17}$$

Which is required for the evaluation of the Porteus Bounds. Each processor also has a copy of V_n(i) $\forall$ i $\in$ S before the start of the $(n+1)^{st}$ iteration. It is a simple matter to verify that the r^{th} processor only requires the following data:[8]

$$\left\{ \left\{ r_i^k : \forall_i \in P_r, \forall_k \in K_i \right\}, \left\{ P_{i,j}^k : \forall_i \in P_r, \forall_j \in S, \forall_k \in K_i \right\}, V_n \right\} \tag{18}$$

7 Partitions and Efficiency

The choice of partition will affect the time required for the communication in the algorithm and the load balancing. For high efficiency, the choice of the partition and the ordering of the states within each set of the partition must give a good load balancing assuming that communications overheads are small. communications overheads account for only a small part of the drop in efficiency. The major part is due to poor load balancing which in this problem remains, even when there is only one action in each state. Since the workload per action is almost constant, load balancing with one action per state can be achieved by allocating an equal number of states to each processor. Howeveer, when this was tried, there was a slight increase in solution times due to the poor initial load balancing. A random ordering of the state space resultsin a better load balance when there is only one action per state beause the maximum number of states allocated to a processor is smaller.[10]

8 Major Factors Affecting Markov Decision Processes [8]

1. Discount factor β
2. Tolerance ϵ
3. Number of states M
4. Average and variation of the number of actions per state
5. Average and variation of the number of transitions per action
6. Speed of mixing (A problem is said to be rapidly mixing if given any starting state, the probability of being in any other state n stages later converges *quickly* to a limit as n increases. The property of mixing is related to ergodicity)
7. Concentration of transition probabilities in the lower triangle of the transition matrices.
8. Closeness of the structures of the transition matrices of the non-optimal policies to that of the optimal policy.
9. Closeness of the values of the non-optimal policies to that of the optimal policy

9 Serial and Parallel Results

The values in the tables following are given in seconds and exclude the time taken to obtain the required data initially. The poor load balancing, due to the variation in the number of actions per state is the major limiting factor regarding the efficiency of PJ with PB when the number of processors used is greater than 11.[10]

N_{it}- Number of iterations

N_{pr}- Number of processors

T_{sol}- Solution time = calculation time + communication time

t_{av}- Average time per iteration = T_{sol}/N_{it}

t_{stan}- Standardised time = T_{sol}/best time for that problem

ϵ- Efficiency = ((1/N_{pr})*serial solution time)/T_{sol}Xs

ϵ_{comm}- The communications overheads = T_{comm}/T_{sol}

Serial Value Iteration

Iterations	T_{sol}	t_{av}	t_{stan}
93	57.41	0.62	7.24

Syncronous Non-Overlapped Parallel Pre-Jacobi

Processors	Iterations	T_{sol}	ϵ	ϵ_{comm}
1	93	57.41	100	0.0
5	93	11.34	101.2	0.3
11	93	5.32	98.1	1.2
25	93	2.63	87.3	4.9
49	93	1.74	67.3	13.2

10 Preliminary Conclusion

Since optimum efficiency is a major issue, noting that for 1< N_{pr} <11, the number of processors provides approximately only a 1% variation in efficiency at a cost of increasing overheads with respect to a monotonic increase in the N_{pr}. For $N_{pr} \geq 11$, the variation in efficiency is >2.2$\%$ and increases monotonically. Since on occasion we may need to investigaterelations requiring > 11 processors, and since efficiency hardly varies between 1 and 5 processors, an optimum configuration for the purposes of satisficability (a state close to optimisation), may be taken to be the use of 1 processor per parameter set. In this way, communications overheads do not affect the performance and load balancing is not an issue. With this in mind, a parallel program has been constructed to allow thevariation of any one of the 9 input parameters and to place each distinct partitioned set of 9 elements on a single processor. This provides a family of results from which the optimum solution can be chosen. In this fashion, the use of 50 processing elements on a varied stock capacity of say 120 units to 170 units would return a set of 50 unique solutions with respect to 120,121,122, ... ,170 units. This could be viewed either as a further step in optimisation if we choose the best policy and value function, or as a basis for experimentation and research into the properties of the solutions in order to provide a more manageable solution, perhaps via linear extrapolation and the production of a convergent polynomial.

11 Implementation of the Parallel Optimisation Program

In this paper, the parallel optimisation program was implemented using a chain topology, although a ring or torus might have been used. The general principle was to create a discrete set of n values, with respect to any single input parameter α say, with n-1 intervals of regular length,

$$\alpha_1, \alpha_2, ..., \alpha_i, ..., \alpha_{n-1}$$

by requesting a specified start = α_1 and end point = α_n on processor zero.
(Since we have truncation and round off errors, α_n is not necessarily equal to the specified end point). The number of intervals is created by the size of the processing environment with size = no of processing elements = ω say,

length of interval

$$= \psi = \frac{end - start}{\omega} \tag{19}$$

then

$$\alpha_1 = start \ \ and \ \ \alpha_n = \alpha_1 + \psi * \omega = actualendpoint \tag{20}$$

Each set of input parameters is related to V by

$$V_{\alpha_i} = \frac{\partial V}{\partial \alpha_i} \tag{21}$$

where $\partial \ \alpha_i$ is the change in α from α_{i-1} to α_i and

$$V = f\{\alpha, \nabla, \Im, \mho, \wp, ..., \phi\} \ say \tag{22}$$

This process describes a family of functions, one set S_i of which is distributed to processor i in the set of processors

$$S_i, \ i = \{0, 1, 2, ..., \omega - 1\}$$

via the broadcasting of every constant parameter and the sending of each independent α_i to Processor

$$i \ \in S_i$$

. The Markov process then operates independently on each processor, at the end of which we process the results of each

$$V_{\alpha_i} \ and \ V^{\alpha_i}$$

and finally gather the results to processor zero for the displaying and returning of data.

12 Results

The parallel optimisation program has been designed to deliver the functional results displayed in table 1. These can be viwed graphically, using PLOTMTV or some similar program, or read as data values directly from the files. Every function is returned with respect to the number of processors. this has allowed properties of the delivered MDP Optimum Policy and Value Function to be investigated to a high degree.

Table 1: Returning Functions

FILENAME	X-AXIS	Y-AXIS	Z-AXIS	FUNCTION
aagedp	Rank	Arithmetic Average	-	Vfn Arithmetic Average
aaverge	Rank	Arithmetic Average	-	Policy Arithmetic Average
capacty	Rank	Capacity	-	Warehouse Capacity
demand	Rank	Demand	-	Highest Demand
deviate	Rank	Standard Deviation	-	Policy Standard Deviation
gagedp	Rank	Geometric Average	-	Vfn Geometric Average
gcdmat	policy	policy	-	GCD(Policy,Policy)
globmn	Minimum	-	-	Policy Global Minimum
globmnd	Minimum	-	-	Vfn Global Minimum
globmx	Maximum	-	-	Policy Global Maximum
globmxd	Maximum	-	-	Vfn Global Maximum
iterate	Rank	Iterations	-	Number of Iterations
maximum	Rank	Maximum	-	Policy Maximum
minimum	Rank	Minimum	-	Policy Minimum
polcy3d	State	Policy	Vfn	State/Policy/Vfn
policy	State	Policy	-	State/Policy
primes	Count	State	Prime	Count/Primes/State
sum	Rank	Sum	-	Policy Sum
vary	Rank	Variable	-	DP Variable/Rank
vary1	Rank	Variable	-	Int Variable/Rank
vfn	State	Vfn	-	Value Function/Rank
vfnmax	Maximum	-	-	Global Vfn Maximum
vfnmin	Minimum	-	-	Global Vfn Minimum
vfnprod	Rank	Product	-	Vfn Product/Rank
vfnsum	Rank	Sum	-	Vfn Sum/Rank

Table 2 delineates a number of the initial returned values for comparism, where the $\odot$ symbol indicates a state of constancy. Figures 1 and 2 display the Optimal Policy and the Value function respectively, in their initial state.

Table 2: Initial State of Parameters

Processors	Iterations	Highest Demand	Policy	Value Function
1	93	11	$\odot$	$\odot$

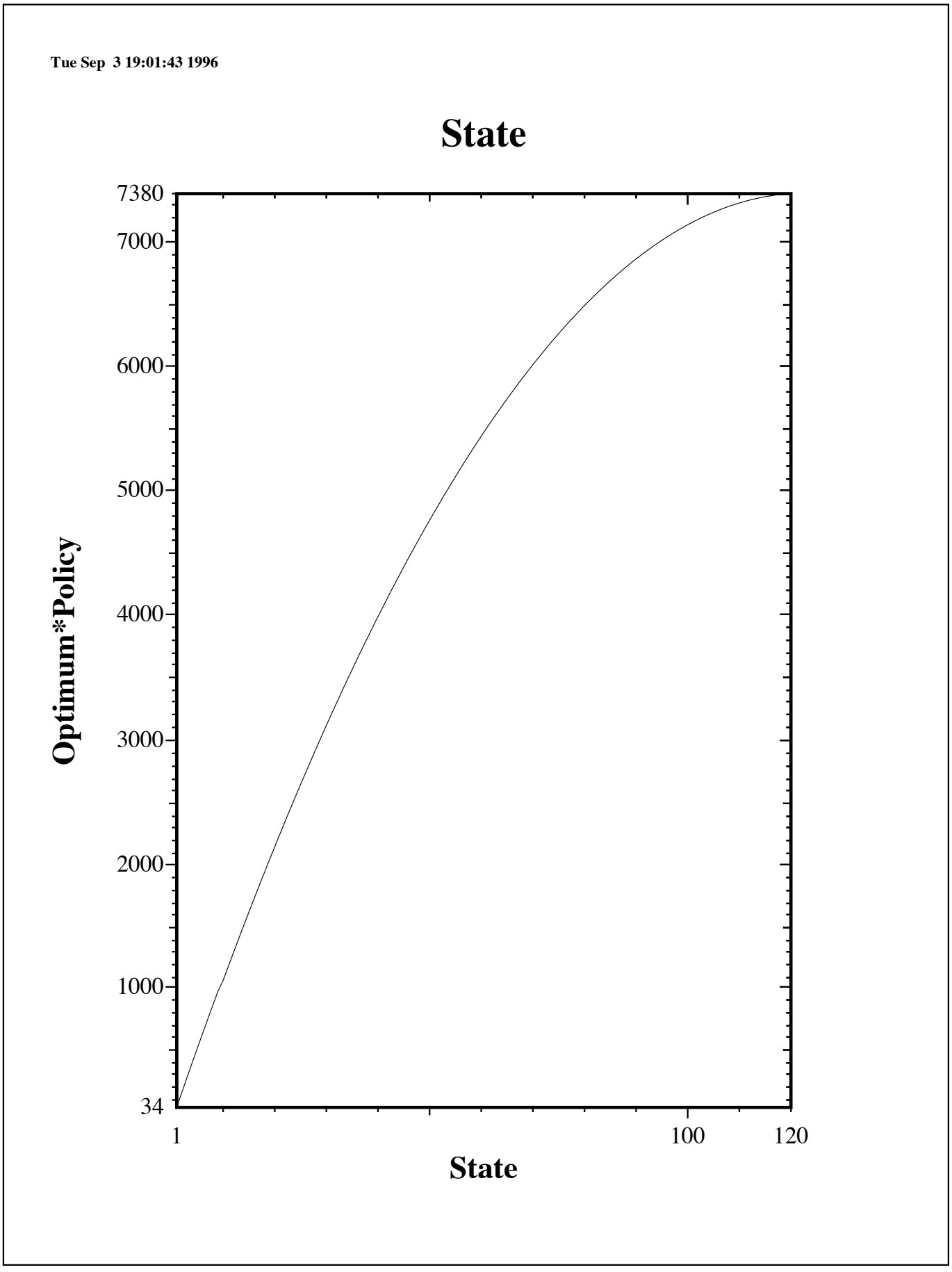


Figure 1: **Initial Optimum Policy Function**

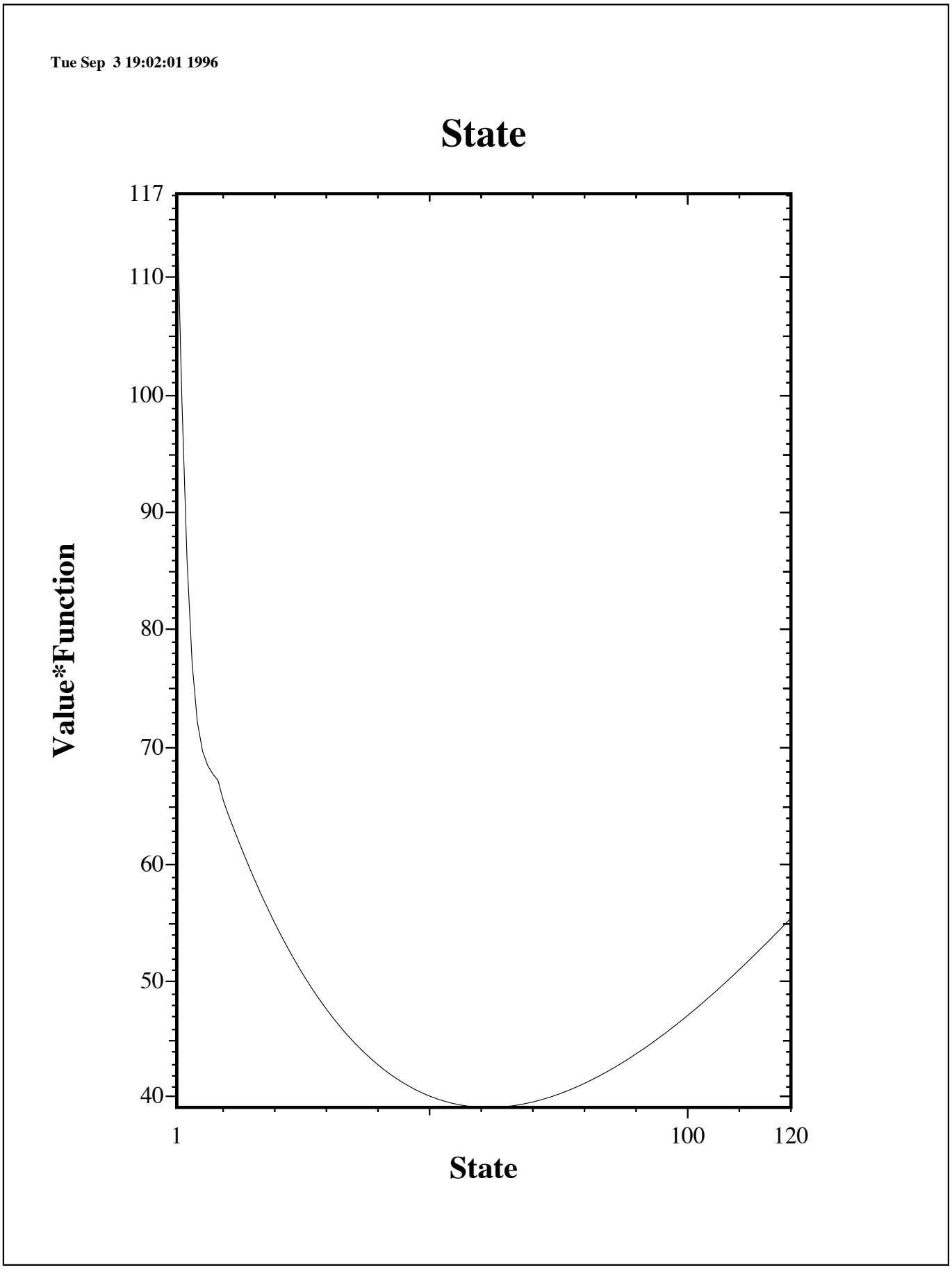


Figure 2: **Initial Value Function**

s

12.1 Example of Variation

In table 3, a number of the results for a varied discount factor β are tabulated where,

$\odot$ $\Rightarrow$ No change

δ $\Rightarrow$ Was there a change in parameter ?

$\surd$ $\Rightarrow$ Yes

$\otimes$ $\Rightarrow$ No great change, but some deviation which requires explaination or further investigation.

Table 3: Variable Discount Factor

Processor Rank	0	1	2	3	4	5
Discount β	0.9	0.91	0.92	0.93	0.94	0.95
Highest Demand	11	11	11	11	11	11
No of Iterations	61	69	77	87	103	125
δ Policy $\otimes$	$\odot$	$\odot$	$\odot$	$\odot$	$\odot$	$\odot$
δ Vfn ?	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Aaverage	4841.92	4842.05	4842.19	4842.61	4842.85	4843.14
Policy Sum	576189	576204	576221	576271	576299	576334
Vfn Sum	1212.37	1385.80	1610.98	1909.58	2325.86	2931.97

In this case, there was an irregular deviation indicated by $\otimes$ between states 1 and 10 in each returned value of Optimum Policy(See figures 3,4,5 and 6 for the initial and expanded views) occuring in the region of the singular point between the same values of states on the Value function related curve.(See figures 7 and 8).

12.2 Regular rectilinear Markov Primes

The irregularity and occasional regularity of the returning function *primes*, displaying the prime numbers contained in the optimum policy initially gave no indication of any connection or relation between the sequence of the optimum policy values and the prime numbers. However, when the Greatest Common Denominator (GCD) function was studied in its graphical form, there seemed to be some regularity (other than diagonal symmetry which is irrelevant in this case since GCD(policy,policy) is both the x-axis and the y-axis). See figure 9. Closer inspection of the graphics and the corresponding data indicated the existence of regularly spaced prime GCDs, beginning at the diagonal where the original optimum policy values appear (since GCD(A,A) = A) and extending in the x and y directions Rectilinearly, from wherever such a point was initially created. The distribution of the RRMP will necessarily form both a monotonic and a contraction property in the resulting optimum policy curve. It is interesting to note that a regular sequence does not begin until after state 10, indicating the possibility of an initial flaw in the optimisation program and/or algorithm. Figures 10,11,12 and 13 display expanded views of the RRMP for the case of the prime number 17.

13 Conclusion

The most important property to usefor optimisation of speed is the degree of mixing. When the problem is rapidly mixing, Pre-Jacobi will perform well. In terms of efficiency, PJ parallelises well but communication overheads were the major limiting factor.[10] Allocating 1 parameter set per processor is both efficient and versitile, allowing the related family structure of asymptotes, standard deviations and maxima and minima to be extrapolated in order to further optimise of to investigate fundamental properties of the Markov Decision Process. A graphic example of this has been demonstrated through the use of the greatest common denominator function, gcdmat, showing the relation between the Rectilinear Markov Primes (RMP) and the monotonic and contraction properties of PJ.[6]

References

[1] D.T. Phillips, A. Ravindran, J.J. Solberg. *Operations Reasearch Princples and Practice*.

[2] J. Kidd. *Managing with Operational Research*.

[3] Budnick, Mojena and Vollmann. *Principles of Operations Research for Management*.

[4] R.A. Howard. *Dynamic Programming and Markov Processes*.

[5] S.M. Ross. *Introduction to Stochastic Dynamic Programming*.

[6] V.S. Borkar. *Topics in Controlled Markov Chains*.

[7] W.J. Stewart. *Introduction to the Numerical Solution of Markov Chains*.

[8] T.W. Archibald, K.I.M. McKinnon, L.C. Thomas. *Serial and Parallel Value Iteration Algorithms for Discounted Markov Decision processes*.

[9] L.C. Thomas, T.W. Archibald, K.I.M. McKinnon. *On the Random Generation of Markov Decision Processes*. Working Paper, Department of Mathematics, University of Edinburgh, 1991.

[10] T.W. Archibald. *Parallel Iterative Solution Methods for Markov Decision Processes*.

Dennis Lee is an ex-R.E.M.E. Telecommunications Technician and I.E.E. Associate completing studies as a third year student for a BSc in Mathematics and Computing

Project Supervisors were:
Professor Tom Archibald and Professor Jake Ansell Department of Business Studies, Edinburgh University