5-1-2016

Analysis of Power-Down Systems with Five States

Govind Pathak
University of Nevada, Las Vegas, govind.pathak@unlv.edu

ANALYSIS OF POWER-DOWN SYSTEMS WITH FIVE STATES

By

Govind Pathak

Bachelor of Technology, Information Technology
Guru Nanak Dev Engineering College, India
2014

A thesis submitted in partial fulfillment
of the requirements for the

Masters of Science in Computer Science

Department of Computer Science
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
May 2016

UNLV | GRADUATE COLLEGE

Thesis Approval

The Graduate College
The University of Nevada, Las Vegas

April 28, 2016

This thesis prepared by

Govind Pathak

entitled

Analysis of Power-Down Systems with Five States

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
Department of Computer Science

Wolfgang Bein, Ph.D.
Examination Committee Chair

Ajoy Datta, Ph.D.
Examination Committee Member

Laxmi D. Gewali, Ph.D.
Examination Committee Member

Venkatesan Muthukumar, Ph.D.
Graduate College Faculty Representative

Kathryn Hausbeck Korgan, Ph.D.
Graduate College Interim Dean

ABSTRACT

Analysis of Power-down Systems with Five States

By

Govind Pathak

Dr. Wolfgang Bein
Examination Committee Chair
Professor
Department of Computer Science
University of Nevada, Las Vegas

We consider a device, which has states ON, OFF and fixed number of intermediate states. In the ON state the device uses full power whereas in the OFF state the device consumes no energy but a constant cost is associated with switching back to ON. Intermediate states use some fraction of energy proportional to the usage time but switching back to the ON state has a constant setup cost depending on the current state. Such systems are widely used to conserve energy, for example to speed scale CPUs, to control data centers, or to manage renewable energy.

We analyze such a system in terms of competitive analysis and give a heuristic for finding optimal online algorithms. We then use our approach to discuss five-state systems which are widely used in practice.

ACKNOWLEDGEMENT

No endeavor is effort of an individual. I would like to express my deep sense of gratitude to my committee chair Professor Dr. Wolfgang Bein, who made it possible for me to complete my research. His astute guidance, advice and constructive criticism is the key of success of this research. Without his direction and persistent help this dissertation would not have been possible.

I am deeply indebted to Dr. Ajoy Datta and Dr. Laxmi Gewali. Without their guidance, support and good nature, pursuing computer science would have been very difficult. Not only they were available whenever I needed their help, but their willingness to discuss anything from convection to libation helped me through two important years of my life. I would also like to thank Dr. Venkatesan Muthukumar for agreeing to be on my thesis committee despite his extremely busy schedule.

I must express my very profound gratitude to my parents Mrs. Rajni Pathak and Mr. Womesh Pathak for providing me with unfailing support and continuous encouragement throughout my years of study. I would also like to thank my sister Surbhi and my brother-in law Sumit for their consistent motivation in thick and thin, without their support it all would have been a different story.

Finally, I would like to acknowledge the support given by all my friends specially Gaurav for helping me whenever I was stuck while coding, Anjali for proof reading my report and Sharath for being there when ever I was stressed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

There has been an exponential increase in the power consumption in Information Technology. This power consumption has led to two types of problems- energy consumption and temperature [1]. In case of a Xerox machine [7], for example, the device turns off after some time if the next job does not arrive in a definite amount of time. This process of the device going into the OFF state occurs so that the energy is saved and the device is not always ON waiting for the next request. Now assuming that the next job comes immediately after the machine turns OFF, in that case the cost involved in turning ON will be much higher than the cost that was saved earlier when the device turned OFF. Moreover, in such case not only the cost will be affected, but also other factors like the time is affected. If the device has to go from OFF to ON to complete the request, it will take a lot of time resulting in poor device performance. Similarly, in thermostat, the air-condition turns off when it reaches the set temperature without considering the outside temperature. Now if the outside temperature is quite high, it will make the temperature of the room high as well implying that the air-conditioner needs to cool down the room again to reach the set temperature. Again in this case, the performance of the device will be affected since the time to cool the room again will be high and also the device has to turn on again to do the same. Power down mechanisms are used in these cases to save power [6]. We seek to address these issues by introducing lower power saving states, and then comparing it with the case in which the device has the knowledge of the next request. For better understanding a detailed explanation of another problem called the paging problem is provided in the following section.

1.2 The Online Competitive Analysis

In online model, we do not know at what time the next request will arrive as the request comes one piece at a time, whereas, in offline model all the input data is completely available before the algorithm even starts. Online algorithm is considered as one of the realms in paging problem. Paging problem is one of the fundamental and significant problem in computer science. Here we have two level of memory storage the fast memory or the cache and a slow memory or virtual memory. We can only store f number of pages in fast memory whereas, s in slow memory and $f < s$. In paging algorithm, a series of requests is presented to the virtual memory pages [2]. If the requested page is present in the fast memory than it is a hit where there is no cost occurred, on the other hand, if it is in the slow memory than it must be brought to the fast memory at a unit cost. Now the algorithm will decide which of the *f* pages in the fast memory should be removed to make room for the new page.

If we know all the requests before hand, we can make better decisions. Here the optimal offline algorithm will remove the page from the fast memory whose request is farthest in the future. But, in real scenario paging decisions are made without the prior knowledge of the future requests. In cases where the pages are evicted from the fast memory without knowing which page will be requested in future, they are referred as an online paging algorithm.

For an online paging algorithm, two analysis techniques can be implemented to evaluate the performance, these are worst case and average case analysis. The problem with worst case is that it is completely uninformative and similarly, for the average case we have to propose a statistical model in order to input. Keeping these observations in mind, Sleator and Tarjan [13] proposed the concept of competitive analysis. In competitive analysis the ratio of online algorithm to optimal offline algorithm performance is compared and the worst case ratio is considered. In order for an algorithm to be considered strongly competitive it should achieve the best possible competitive ratio for a give problem. The majority of research recently done on online algorithms use competitive ratio as an evaluating standard.

A common offline algorithm is LFD (Longest Forward Search) where on each miss the page is replaced in cache that will be requested farthest out in the future. Whereas, one of the online algorithm is LRU (Least Recently Used) here on each miss the page is replaced in cache that was requested furthest in the past [2]. The competitive ratio, when we compare these two algorithm, will be 2. Below is the explanation:

Consider that we can store only two pages in fast memory. In this case we compare two different algorithms with the same requests. While implementing the LRU (Online) model, we can observe that when a new page request is arrived, the page which has most recent access will be replaced with the new page, in the picture the red colored page shows that it is restored by the algorithm and the other page is discarded.

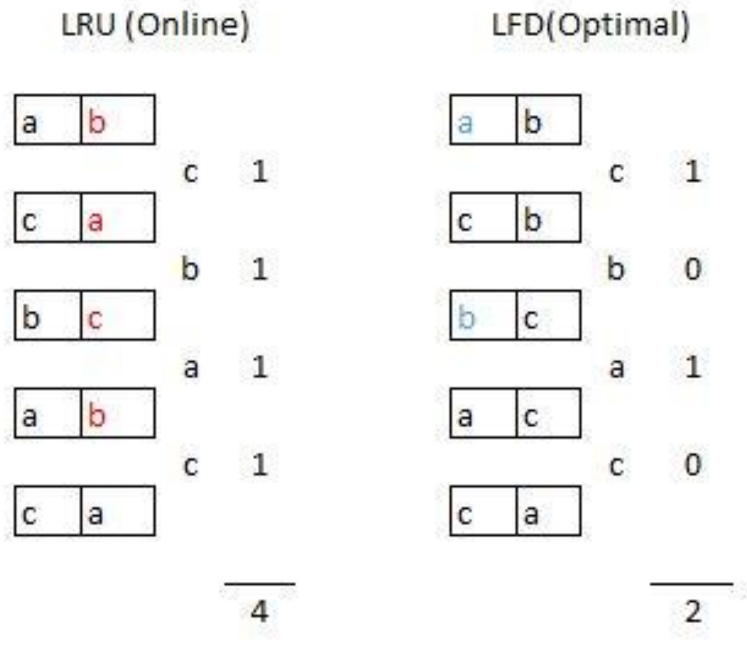


Figure 1.1 Calculating CR in Paging problem

$$CR = \frac{Cost_{LRU}}{Cost_{OPT}} = 2$$

On the other hand, in LFD (Optimal) model, as we are already aware of the future request, the pages in the fast memory are replaced in accordance with the future request. Here the blue colored page is the one which is restored and the other one is discarded.

Here whenever the page is not available in the fast memory its considered as a miss and a unit cost of 1 is charged. Whereas if the page is accessible in the fast memory than its considered as a hit and no cost is charged. In the given case we applied the LRU and LFD algorithm on same problem with identical jobs are requested. We can see that the total cost is 4 units for the online algorithm whereas if we know all the request prior to the start of the algorithm (Optimal) than the cost is 2 units. The competitive ratio at a given instance (ρ) is given as:

$$\text{CR } (\rho) = \frac{CostALG_{online}\ (\rho)}{Cost\ ALG_{Optimal}(\rho)}$$

Whereas, the CR of the algorithm will be:

$$\text{CR} = \max \text{CR}(\rho)$$

We consider the max CR as the actual CR of the algorithm as the algorithm cannot worse than that specific value. Hence CR is a very important factor in considering an online problem as it will aware us that how worst an algorithm can perform.

1.3 Previous Work

The power down problem has been discussed by Augustine et. al [4]. They describe an algorithm, which gives us competitive ratio for an ε tolerance where ε is an error parameter for which a strategy is given such that the competitive ratio will be within the ε range. Time complexity of this algorithm is O(k^2/log k) log (1/ε) where k+1 is the number of states.[4]. Further there are numerous other online problems, one of them is *BahnCard Problem.* The BahnCard is a discount card offered by the German national railway company. The pass is valid for one year and holders are given specific discount on the train tickets depending on the cost of the BahnCard. This problem is considered as online problem as the person does not know that when one will be travelling next and how many trips one will make in the year. It has been proved that the greedy algorithm application is baser in the worst case than the algorithm in which one never buy BahnCard [14].

Another example of online problem is smart grid problem. Smart grid will integrate with renewable energy sources to reduce its dependence on conventional non-renewable resources [11]. For instance, we are using electricity by generated by a wind mill and suddenly the wind stopped blowing so now we have to switch to conventional power generation sources, but let us say after some time there is breeze of wind now its our decision that whether we want to immediately switch back using power generated by the wind mill or will wait. As we do not know that for how long the wind will below it can be considered as an online problem.

The best deterministic online algorithm will remain in ON state till the energy spent is equivalent to OFF state or lower power state [4]. The algorithm will attain the optimum competitive ratio of 2 [15]. In case of randomized online algorithm, the best competitive ratio attainable is $e/(e-1)$ [15].

CHAPTER 2

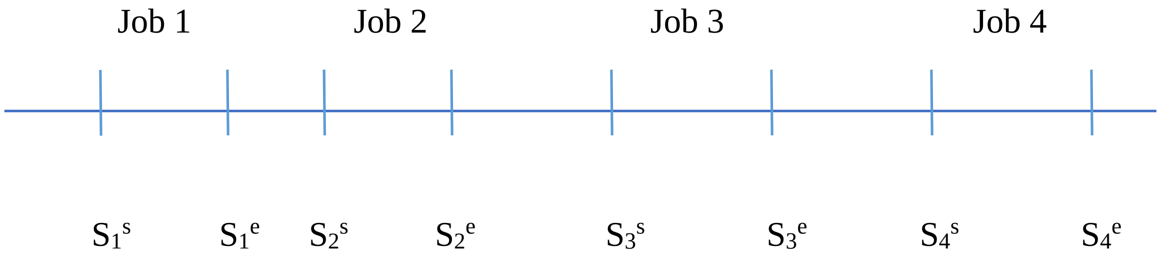
FORMULIZATION OF THE MODEL

We consider a two state system, which only consists of ON and OFF state, as well a multiple state system which have INT_k states. The energy consumed by devices depends on two factors, standby cost and switching cost. Standby cost is consumed by the system when its not OFF, whereas switching cost is the amount required to change from one state to another. When the device is in ON state the standby cost is 1 and switching cost is 0 whereas, when the device is in OFF state the standby cost is 0 and the switching cost is 1. In INT_k states the standby and switching cost factor will be between (0,1), such systems has been considered in [4,7].

State	**Standby (COST)**	**Switching (COST)**
OFF	0	1
INT_j	$a_j \in (0,1)$	$d_j \in (0,1)$
INT_i	$a_i \in (0,1)$	$d_j \in (0,1)$
ON	1	0

Table 2.1 Device with multiple INT states

Here in table it is shown the standby and the switching cost for different states. Whenever the service is requested, device will immediately change its state to ON state no matter which state it was earlier in. The standby cost of INT_j is less then INT_i where as the switching cost of INT_i is less then that of INT_j. Let S_1^s, S_2^s, S_3^s, S_4^s and S_1^e, S_2^e, S_3^e, S_4^e be non negative real number where S_1^s is the start of job 1 and S_1^e is the end of job 1. So, the device has to be in ON state from S_1^s to S_1^e.



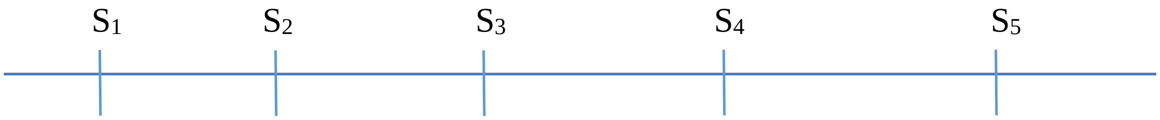
After job 1 is finished the device can be in either ON, OFF or any of the INT states until the next request comes. If the next request is immediately after the S_1^e that is, end time of job 1, it would be better if the device stays in the ON state since, going to any other state (OFF or any INT state) would be inefficient.

Let us assume that the difference between S_1^e and S_2^s is small, in that case, if the device goes to the OFF state, then the overall cost will increase as the switching cost will also be incorporated. This will not be the case if the device continues to stay in the ON state until the next job starts. On the other hand, if the difference between S_1^e and S_2^s is more then staying in the ON state will now result in high standby cost which will affect the efficiency of the device. Hence, in this case, it would be better for the device to go in the OFF state.

Power Down mechanisms can be classified as online and offline models. In an online model, the machine does not have prior knowledge of the next request [9]. On the other hand, in offline algorithm, machine knows the arrival sequence of all the next requests. For example, in online model, the device will not know the start time of Job 2 that is, S_2^s after Job 1 is finished whereas in offline model, the device will know the exact starting time S_2^s of Job 2.

The offline model is considered as optimum model since the decision to go to a particular state between the jobs can be made by comparing the overall cost for different states which includes standby and switching costs [9]. The algorithm that determines the state that the device should go into is known as the optimal offline algorithm.

Practically the device has to be in ON state between the start time and end time of any job.



If S_1^s is the start time of job 1 and S_1^e is the end time of job 1 but as we know that the device will have to be in ON state, no matter whether we are applying online or optimal algorithm as considering this cost will not make any difference, so we can only use S_1 to represent both the start time and end time of job1. Similarly, S_2 will represent the start time and end time of job2 and so on hence the modified timeline is showed above.

As we mentioned before, to measure the quality of the online algorithm we calculate competitive ratio (CR) [5,12], which is the ratio of the cost of online algorithm to the cost of offline (optimal) algorithm at a given time. The competitive ratio can be represented as,

$$\mathrm{CR} = \frac{\mathrm{Cost}_{\mathrm{ONLINE}}}{\mathrm{Cost}_{\mathrm{OFFLINE}}}$$

In my study I have proposed a five state algorithm in which I have used ON, OFF and three intermediate states, INT1, INT2, INT3.

State	Standby (COST)	Switching (COST)
OFF	0	1
INT3	a3	d3
INT2	a2	d2
INT1	a1	d1
ON	1	0

Table 2.2 Device with Three INT states

Here the value of a1 is greater than a2 and value of a2 is greater than that of a3. Likewise, the value of d3 is greater than d2 and value of d2 is greater than that of d1. We proposed an online algorithm in which the device will go to each INT state one after the other and ultimately goes to the OFF state, if the next request doesn’t come by that time. If the request comes before the device goes to the OFF state, then it will go to the ON state from its current intermediate state. Then we have compared it with the optimal algorithm and tried to find the best competitive ratio.

CHAPTER 3

POWER DOWN SYSTEMS WITH THREE STATES

In this algorithm, there is only one intermediate state which makes it a three state algorithm consisting of ON, OFF and INT state. The different values of *a* (standby cost) and *d* (switching cost) [7] for the three states are as shown below:

State	**Standby (COST)**	**Switching (COST)**
OFF	0 (a_{off})	1 (d_{off})
INT	a_i	d_i
ON	1 (a_{on})	0 (d_{on})

Table 3.1 Device with one three states

We calculate the cost for both online and offline model to obtain the best competitive ratio for the given values of *a* and *d.*

In an offline model [9], the subsequent job requests are known in advance. After finishing a job, the algorithm determines which state the device should go into before the next request comes in by comparing the costs involved in different states. The algorithm chooses the state with the minimum cost and continues to remain in the same state till the next request.

3.1 Three State Optimal and Online Model

To choose the state with the minimum cost, the device will calculate the cost in each state and than will go to the one with minimum cost from the outset. If the request come after time period *s* than if *s* is small, it is best to stay ON. If *s* is larger is may best to go the intermediate state or OFF state. We calculate value z_0 for which it is cheapest to stay in ON if it is smaller than z_0. We also calculate z_1 for which it is cheapest to stay in the intermediate state if $s > z_0$ and $s < z_1$. The time after which the device will go to different state, we refer these points as ‘z’ values. After completing the request, the device will be initially in the ON state for a certain time as the cost will be lowest for this state since immediate switching would mean more cost but there will be a certain point at which the the cost to remain in the ON state will be more than that in the INT state, that point is considered as z_0. Similarly, z_1 will represent the point at which it will be better to switch from the INT state to the OFF state because of the higher cost in INT state.

The value of z can be obtained using:

$$z_i = \frac{d_{i+1} - d_i}{a_i - a_{i+1}}$$

Where *i* can be 0 or 1 (since it is three state), also *d* and *a* are the switching and standby cost of their respective states.

To calculate the value of z_0, the value of i will be 0, so the above formula becomes:

$$z_0 = \frac{d_1 - d_0}{a_0 - a_1}$$

Substituting the values of d_0 and a_0 (ON state), the value of z_0 will be:

$$z_0 = \frac{d_1}{1 - a_1}$$

Similarly, the value of z_1 after substituting i = 1, $a_2 = 0$ and $d_2 = 1$ (OFF state), will be:

$$z_1 = \frac{1 - d_1}{a_1}$$

On the other hand, in an online model, the job sequence is not known. Hence, after finishing a job the device is unaware of the start time of the next job. So the device will stay in the ON state for a fixed interval of time which is represented using t_0. After this it goes to INT state for another fixed interval of time, t_1 and eventually to the OFF state if the next request is not received by that time. Here the cost calculated will depend on the state(s) the device goes into while waiting for the next request. If the request is received immediately after the previous job is completed, then the device will continue to stay in the ON state thus eliminating the switching cost (*d*). Otherwise, the device will stay in ON state for a fixed interval of time after which it goes to the INT state for a definite time gap. If the request is received while this time, the device will have to go to the ON state from its current INT state, implying an added switching cost involved during this transition. Similarly, if the device goes to the OFF state after the INT state while waiting, a different value of switching cost will be added to the overall cost when it goes to the ON state to start the next job. Here the total cost is cumulative, which means that if the next request comes while the device is in the INT state the total cost will be the cost while it was in the ON state plus the cost till the time it was in INT state and the switching cost from the INT state, similarly, if the next request comes when the device is in OFF state the total cost will be cost in ON state, cost in INT state and switching cost from OFF state.

3.2 Calculating Cost in Offline & Online Model

In an offline model, the cost involved [10] till the next request comes is calculated as follows:

$$\text{Cost}_{\text{offline}} = (S_1^s - S_0^e)\, a + d$$

Where S_1^s is the time at which the next request arrives or the starting time of the next request, S_0^e is the time when the machine was in ON state previously i.e. end of prior request, '*a*' is the standby cost and '*d*' is the switching cost. As described earlier, S_1^s can be rewritten as S_1 and S_0^e as S_0. Hence we can restate the formula as:

$$\text{Cost}_{\text{offline}} = (S_1 – S_0)\, a + d$$

The algorithm will calculate the cost involved in all the three states and will go to the state in which the cost is minimum.

Likewise, in the online model the basic method of calculating the cost [10] is same as the offline model but the only difference is that the cost will be calculated for only those states that the device goes into while waiting for the next request. Let us examine the following cases for better understanding.

Case 1: The request comes while the device is in the ON state. Let t_0 be the fixed amount of time the device stays in the ON state while waiting for the next request and before going to the INT state. Then, the cost is calculated using the formula:

$$\text{Cost}_{\text{online}} = (T_{o} – S_{0})\, a_{\text{on}} + d_{\text{on}}$$

Where T_0 is the time when the next request comes while the device is in the ON state, and its value is less than t_0. S_0 is the time at which the last request was completed when the machine was in the ON state. Here, the value of a_{on} which is the standby cost is 1 and the value of d_{on} which is the switching cost is 0 since the device is already in the ON state and no transition from any other state is required to start the next job. Hence, the formula for calculating the cost in this case can be rewritten as:

$$\text{Cost}_{\text{online}} = T_0 – S_0$$

Case 2: The request comes while the device is in the INT state. Let t_1 be the amount of time the device stays in the INT state while waiting for the next request and before further going to the OFF state. Then, the cost in this case can be calculated as:

$$\text{Cost}_{\text{online}} = (t_0 - S_0)\, a_{\text{on}} + (T_1 - t_0)\, a_{\text{int}} + d_{\text{int},}$$

Where t_0 is the time for which the device stays in the ON state, S_0 is the ending time of the last request, T_1 is the time when the next request comes while the device is in the INT state and its value is less than t_1. Also, a_{on} represents the standby cost for the device in the ON state, a_{int} represents the standby cost for the INT state and d_{int} represents the switching cost involved in transitioning from the INT state to the ON state for starting the next job.

Case 3: The request arrives while the device is in the OFF state. Once the device reaches this state, regardless of the arrival of next request, the cost to switch it to the ON state will remain unchanged. The cost can be calculated using the formula:

$$\text{Cost}_{\text{online}} = (t_0 - S_0)\, a_{\text{on}} + (t_1 - t_0)\, a_{\text{int}} + (t_{\text{off}} - t_1)\, a_{0\text{ff}} + d_{\text{off}},$$

where t_{off} is the time when the device reaches the OFF state and t_1 is the last time while the device was in the INT state waiting for the next request. Here the value of a_{off} is 0 since no cost is involved when the device stays in the OFF state and d_{off} is 1 as the device has to switch from OFF state to ON state, so the formula can be restated as:

$$\text{Cost}_{\text{online}} = (t_0 - S_0)\, a_{\text{on}} + (t_1 - t_0)\, a_{\text{int}} + 1$$

3.3 Calculating CR in Three States

We applied this algorithm considering the real time scenario where we assigned standby and switching cost values for the INT state. And also, we specified the time for which the device will stay in the ON state and INT state. We considered different cases and in each case we assigned different values for *a* and *d* and different time frames for which the device will stay in ON and INT state.

1. In this scenario, we consider a device where we assign values 0.3 and 0.7 to *a* and *d* respectively. As shown in the graph, the device will stay in the ON state for 0.798 units of time after which it will go to INT state and continues to stay in the same state till 1.0 unit of time. If next request does not come by this time, then it will go to the OFF state and will stay in that stay until the next request arrives.

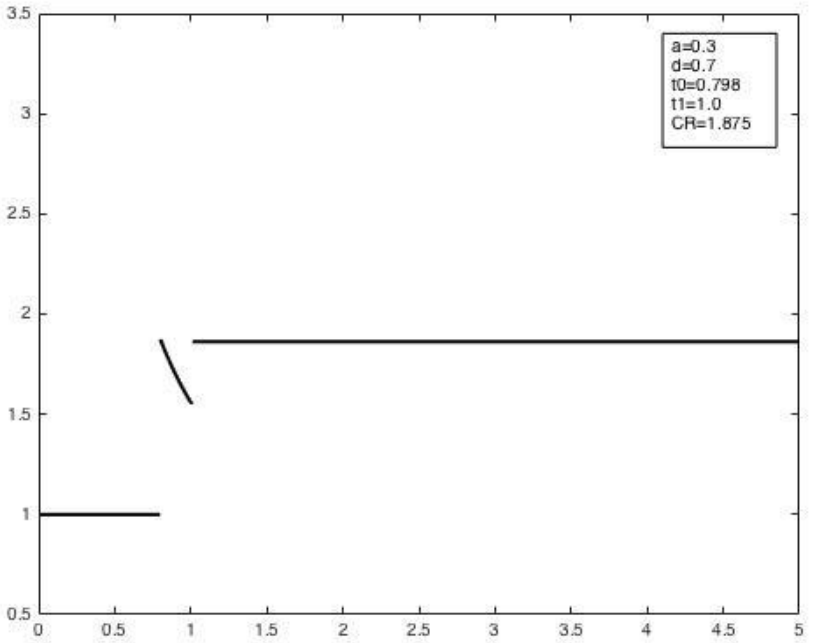


Figure 3.1 Case 1 Three States

The above graph represents the value of competitive ratio (CR) which comes out to be 1.875 using the considered values of *a, d* and the time frame for the different states of the device. Here as we can see that untill 0.798 the CR is 1, which conveys that the power in online and offline model is same. After that there is sudden rise in the CR which decreases constantly till the time reaches 1.0. After this the device goes to OFF state and the cost in both the models becomes same again as the CR in this state is constant.

2. In this scenario, the values of *a* and *d* have been changed to 0.2 and 0.8 respectively. The graph below shows that the device stays in the ON state for 0.88 units of time and then goes to the INT state until 1.0 unit of time. It then goes to the OFF state if

the request is not received and continues to stay in the same state till the next request arrives.

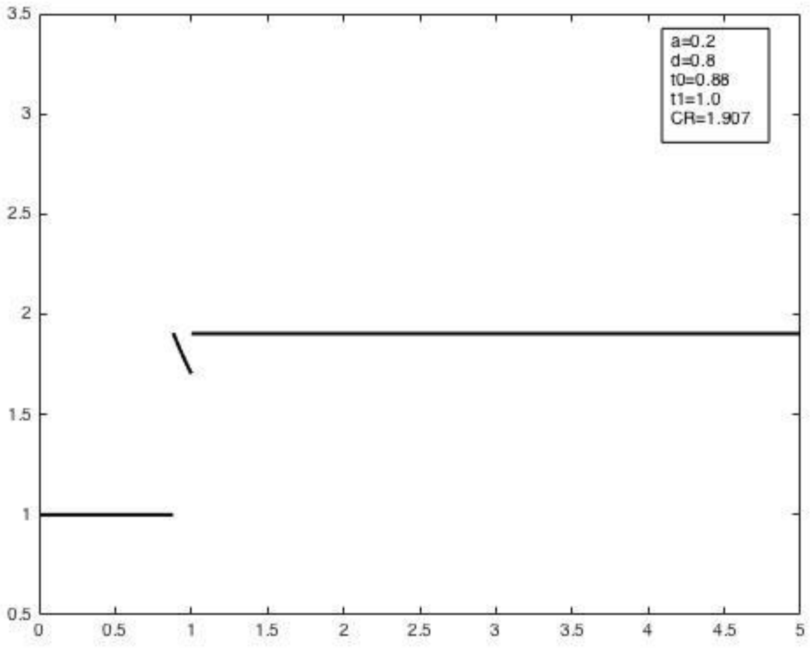


Figure 3.2 Case 2 Three States

The competitive ratio (CR) in this case comes out to be 1.907 using the values of *a* and *d* and time frames for different states. The CR is 1 till the device is in the ON state that is, for 0.88 units of time hence, depicting same power in both online and offline models. Then a sudden increase in the value of CR occurs which continues to decline till the device is in the INT state that is, till 1.0 unit of time. The device then goes to the OFF state and the cost for both the online and offline model becomes same again due to constant CR.

3. Here, the values of *a* and *d* have been changed to 0.1 and 0.9 respectively. The device in this case, will stay in the ON state for 0.944 units of time as can be seen in the graph below. It then goes to the INT state till the time reaches 1.0 unit of time and then goes to the OFF state till the next job request is received.

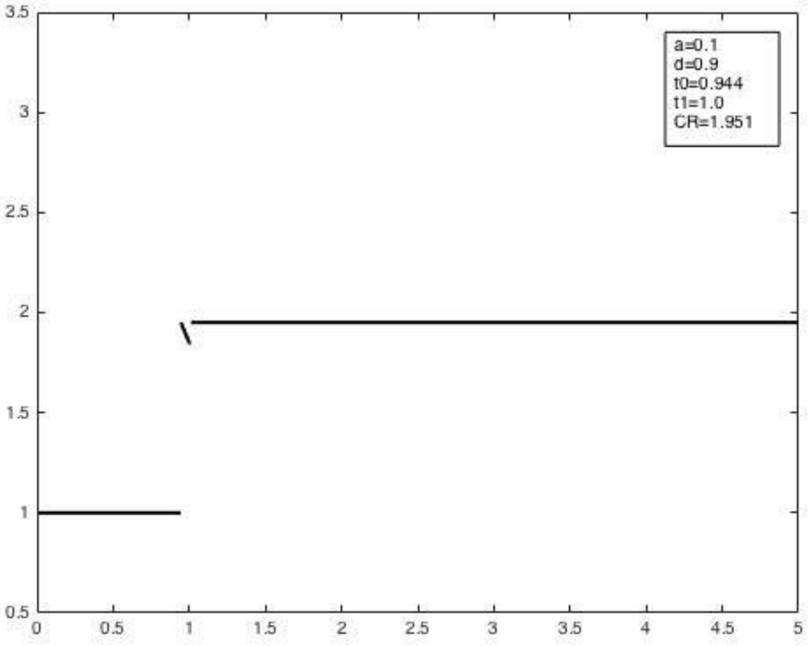


Figure 3.3 Case 3 Three States

The competitive ratio (CR) for this case is 1.951 which has been calculated using the values assigned to *a* and *d* and the different time frames for the various states the device goes into. The CR is 1 till 0.944 units of time or till the device is in the ON state. The power in this case will be the same for online and offline models. The value of CR then experiences a rapid increase after which it keeps dropping till 1.0 unit of time. The device goes to the

OFF state after this and because of constant value of the CR, the cost for online and offline models will become same again.

3.4 Theoretical Analysis of Three State Power Down:

The above cases are the theoretical explanation of the different scenarios, however, the practical approach for calculating the values of CR and obtaining the best values of *a* and *d* can be explained as follows:

To calculate the value of Competitive Ratio (CR), we assume that the sum of the switching cost and the standby cost is equal to 1 [3], that is,

$$a + d = 1$$

Let us consider two points t_0 and t_1, where t_0 is point at which the device switches from ON state to INT state and t_1 is the point of time at which the device switches from INT state to OFF state. The competitive ratio [5,12] at t_0 can be written as:

$$C_1 = \frac{Cost_{ONLINE}(\mathrm{t0})}{Cost_{OFFLINE}(\mathrm{t0})} = \frac{t_0 + \mathrm{d}}{\mathrm{t0}}$$

In the online algorithm, the device stays in the ON state till t_0 time and then powers down to the INT state from which going back to the ON state will involve a switching cost '*d*'. Whereas, in offline model, the device will stay in the ON state.

To calculate the competitive ratio at t_1, for online algorithm, the cost will be added for the ON state, INT state and the OFF state. Hence, the cost for online algorithm can be written as $t_0 + at_1 + 1$, since the device stays in ON state for t_0 time, for the INT state the device stays for t_1 time and involves the standby cost 'a' and for the OFF state as the standby cost is 0, only the switching cost is added which is 1.

For offline model, the cost will be 1 as the device will remain in the ON state therefore the C2 at point t_1 becomes:

$$C_2 = \frac{Cost_{ONLINE}(\text{t1})}{Cost_{OFFLINE}(\text{t1})} = \frac{t_0 + at_1 + 1}{1}$$

t_1 can be rewritten as, $1 - t_0$ since the device will stay in the INT state till 1.0 unit of time and it is staying in the ON state for t_0 time, so the duration of the device being in the INT state (t_1) can be stated as $1 - t_{0.}$ So the formula becomes [3],

$$C_2 = \frac{Cost_{ONLINE}(\text{t1})}{Cost_{OFFLINE}(\text{t1})} = t_0 + a(1- t_0) + 1$$

We will equate the two competitive ratios, C1 and C2, for values a = 0.5 and d = 0.5, to minimize the competitive ratio:

$$C1 = C2$$

$$\frac{\text{t0+d}}{\text{t0}} = t_0 + a\,(1- t_0) + 1 \qquad (1)$$

$$\frac{\text{t0+0.5}}{\text{t0}} = t_0 + 0.5(1- t_0) + 1$$

$$t_0 = 0.618$$

Using this value of t_0, the CR will be 1.809. The graph for the same can be represented as:

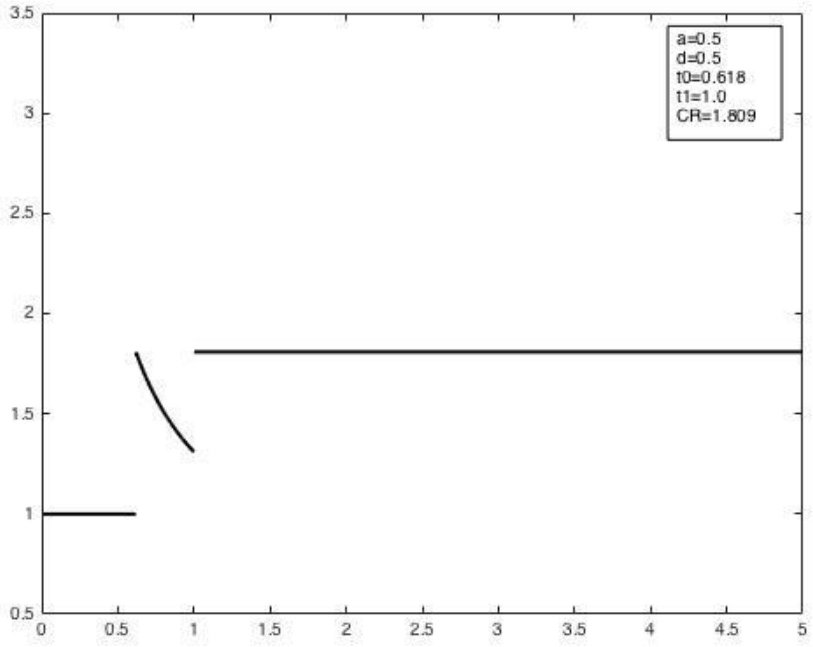


Figure 3.4 Theoratical Analysis of Three States

Now for calculating the best values of a and d, we set C1 = C2 [3],

$$\frac{t_0+\mathrm{d}}{\mathrm{t0}} = t_0 + a\,(1 - t_0) + 1$$

$$dt_0^2 + (1 - d)\, t_0 - d = 0 \qquad \text{(From } a + d = 1\text{)}$$

$$t_0 = \frac{d-1+\sqrt{5d^2-2d+1}}{2d}$$

After solving this we obtain,

$$20d^2 + 8d = 0$$

This gives us $a = 0.6$ and $d = 0.4$. Substituting the values of a and d in (1), we will get t_0=0.5 and CR = 1.8. The results are shown in the graph below:

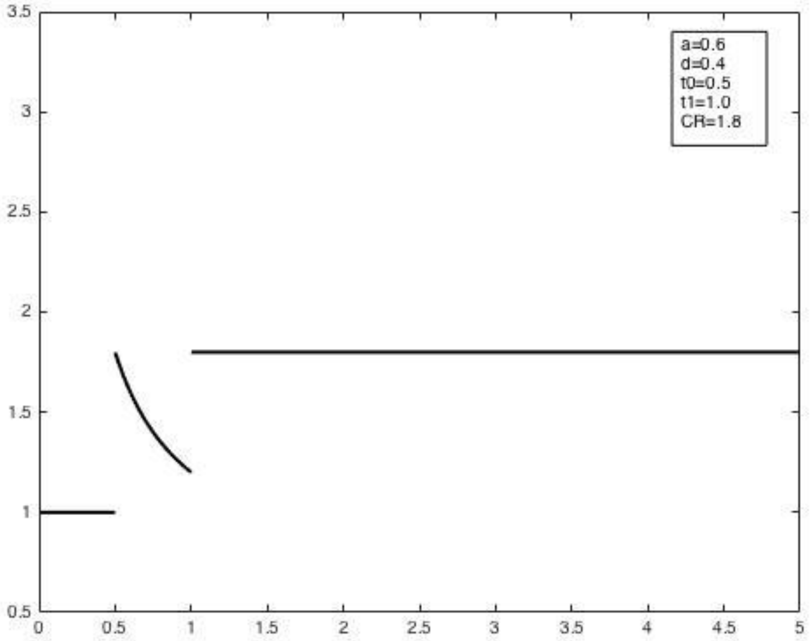


Figure 3.5 Best Case Three States

CHAPTER 4

FIVE STATE POWER DOWN SYSTEMS

In this algorithm, there is ON and OFF state similar to three state algorithm but instead of one intermediate state there are three intermediate states here which makes this algorithm a five state algorithm. The table below represents the five states and values of standby and switching cost for each of the states.

State	Standby (COST)	Switching (COST)
OFF	0	1
INT3	a3	d3
INT2	a2	d2
INT1	a1	d1
ON	1	0

Table 4.1 Device with Five states

When the device goes from INT3 to INT1 the value of standby cost will keep on increasing, whereas the value of switching cost will decrease, which means the value of a1 will be maximum among the a1, a2 and a3 and the value of d3 will be maximum among d1, d2 and d3.

Here, we first find the cost using the offline model [9] and then using the online model and finally using these values we calculate the competitive ratio.

4.1 Calculating Cost in Offline and Online models

For the offline model, we calculate the cost [10] using the formula:

$$Cost_{offline} = (S_1^s - S_0^e)\ a_i + d_i$$

Where a_i and d_i are the standby and switching costs of any of the intermediate states where i can be 1,2 or 3 or can be on or off and S_1^s is the time at which the next request arrives or the starting time of the next request, S_0^e is the time when the machine was in ON state previously i.e. end of prior request, 'a' is the standby cost and 'd' is the switching cost. As described earlier, S_1^s can be rewritten as S_1 and S_0^e as S_0. Hence we can restate the formula as:

$$Cost_{offline} = (S_1 – S_0)\ a_i + d_i$$

The algorithm will calculate the cost involved in all the three states and will go to the state in which the cost is minimum.

Similar to three state algorithm, we will calculate the *z* values here as well using the formula:

$$z_i = \frac{d_{i+1} - d_i}{a_i - a_{i+1}}$$

Where *i* can be 0,1,2 or 3 (since there are five states)

To calculate the value of z_0, the value of i will be 0, so the above formula becomes:

$$z_0 = \frac{d_1 - d_0}{a_0 - a_1}$$

Substituting the values of d_0 and a_0 (ON state), the value of z_0 will be:

$$z_0 = \frac{d_1}{1 - a_1}$$

Similarly, we can calculate the value of z_1 by substituting i with 1,

$$z_1 = \frac{d_2 - d_1}{a_1 - a_2}$$

Where a_1 and d_1 are the standby and switching cost of the INT1 state and a_2 and d_2 are the standby and switching cost of INT2 state.

Likewise, the subsequent z values will be: $z_2 = \frac{d_3 - d_2}{a_2 - a_3}$ and $z_3 = \frac{1 - d_3}{a_3}$.

Similarly, in the online model the basic method of calculating the cost is same as the offline model. The difference between the two is that the cost [10] is not calculated for all the states but for only those states that the device goes into while waiting for the next request. Let us study the following cases for better understanding.

Case 1: The request comes while the device is in the ON state. Let t_0 be the fixed amount of time the device stays in the ON state while waiting for the next request and before going to the INT state. Then, the cost is calculated using the formula:

$$Cost_{online} = (T_o - S_0)\ a_{on} + d_{on}$$

Where T_0 is the time when the next request comes while the device is in the ON state, and its value is less than t_0. S_0 is the time at which the last request was completed when the machine was in the ON state. Here, the value of a_{on} which is the standby cost is 1 and the value of d_{on} which is the switching cost is 0 since the device is already in the ON state and no transition from any other state is required to start the next job. Hence, the formula for calculating the cost in this case can be rewritten as:

$$Cost_{online} = T_0 - S_0$$

Case 2: The request comes while the device is in the INT state. Let t_1 be the amount of time the device stays in the INT state while waiting for the next request and before further going to the OFF state. Then, the cost in this case can be calculated as:

$$Cost_{online} = (t_0 - S_0)\ a_{on} + (T_1 - t_0)\ a_1 + d_{1,}$$

Where t_0 is the time for which the device stays in the ON state, S_0 is the ending time of the last request, T_1 is the time when the next request comes while the device is in the INT state and its value is less than t_1. Also, a_{on} represents the standby cost for the device in the ON state, a_1 represents the standby cost for the INT state and d_1 represents the switching cost involved in transitioning from the INT state to the ON state for starting the next job.

Case 3: The request comes while the device is in the INT2 state. The device goes to the INT2 state after INT1 state while waiting for the next request. In this case, if S^{i2} is the amount of time the device stays in this state before going to the next state and while waiting for the next request, then we can calculate the cost using:

$$\text{Cost}_{\text{online}} = (t_0 - S_0)\, a_{on} + (t_1 - t_0)\, a_1 + (T_2 - t_1)\, a_2 + d_{2,}$$

Where t_0 is the time for which the device stays in the ON state, S_0 is the ending time of the last request, t_1 is the time for which the device stays in the INT1 state, T_2 is the amount of time the device stays in the INT2 state before going to the ON state to complete the next request and its value is less than $t_{2.}$ Also, a_{on} represents the standby cost for the device in the ON state, a_1 represents the standby cost for the INT1 state, a_2 represents the standby cost for the INT2 state and d_2 represents the switching cost involved in transitioning from the INT2 state to the ON state for starting the next job.

Case 4: When the request comes while the device is in the INT3 state.

If the request is not received while the device is in the INT2 state, then it goes to the INT3 state. If S^{i3} is the amount of time that the device stays in the INT3 state, then we can calculate the cost using the formula:

$$Cost_{online} = (t_0 – S_0)\ a_{on} + (t_1 – t_0)\ a_1 + (t_2 – t_1)\ a_2 + (T_3 – t_2)\ a_3 + d_3$$

Where t_0 is the time for which the device stays in the ON state, S_0 is the ending time of the last request, t_1 is the time for which the device stays in the INT1 state, t_2 is the amount of time the device stays in the INT2 state, T_3 is the amount of time the device stays in the INT3 state before going to the ON state and its value is less than t_3. Also, a_{on} represents the standby cost for the device in the ON state, a_1 represents the standby cost for the INT1 state, a_2 represents the standby cost for the INT2 state, a_3 represents the standby cost for the INT3 state and d_3 represents the switching cost involved in transitioning from the INT3 state to the ON state for starting the next job.

Case 5: When the request comes while the device is in the OFF state. The device ultimately goes to the OFF state after the INT3 state while waiting for the next request and if the request is received while the device is in the OFF state then the cost can be calculated using:

$$Cost_{online} = (t_0 – S_0)\ a_{on} + (t_1 – t_0)\ a_1 + (t_2 – t_1)\ a_2 + (t_3 – t_2)\ a_3 + (t_{off} – t_3)\ a_{off} + d_{off}$$

Where t_0 is the time for which the device stays in the ON state, S_0 is the ending time of the last request, t_1 is the time for which the device stays in the INT1 state, t_2 is the amount of time the device stays in the INT2 state, t_3 is the amount of time the device stays in the INT3 state before going to the OFF state and t_{off} is the time at which the request comes when its in the OFF state. Also, a_{on} represents the standby cost for the device in the ON state, a_1 represents the standby cost for the INT1 state, a_2 represents the standby cost for the INT2 state, a_3 represents the standby cost for the INT3 state, a_{off} is the standby cost of OFF state and d_{off} is the switching cost of the OFF state. Here the value of a_{off} is 0 and the value of d_{off} is 1 so the formula can be restated as:

$$\text{Cost}_{\text{online}} = (t_0 – S_0)\ a_{on} + (t_1 – t_0)\ a_1 + (t_2 – t_1)\ a_2 + (t_3 – t_2)\ a_3 + 1$$

4.2 Calculating CR in Five States

We applied this algorithm to real time scenario where we assign values to standby and switching costs for all the intermediate states. And also, we specify the time for which the device will stay in the ON state and intermediate states, that is, t_0, t_1, t_2 and t_3 respectively. To assign the standby (a) and switching (d) costs for the intermediate states, we calculate their values using the formula:

$$d_i = \left(\frac{i}{4}\right)^2$$

$$a_i = 1- \left(\frac{i}{4}\right)$$

Here, the values of i can be 1, 2 or 3. For example, to calculate the values of a and d for the first intermediate state, the value of i will be 1. Hence, using the formulae above, the values of a and d will come out to be 0.75 and 0.0625 respectively.

The different values of *a* and *d* for all the states are as shown in the graph below:

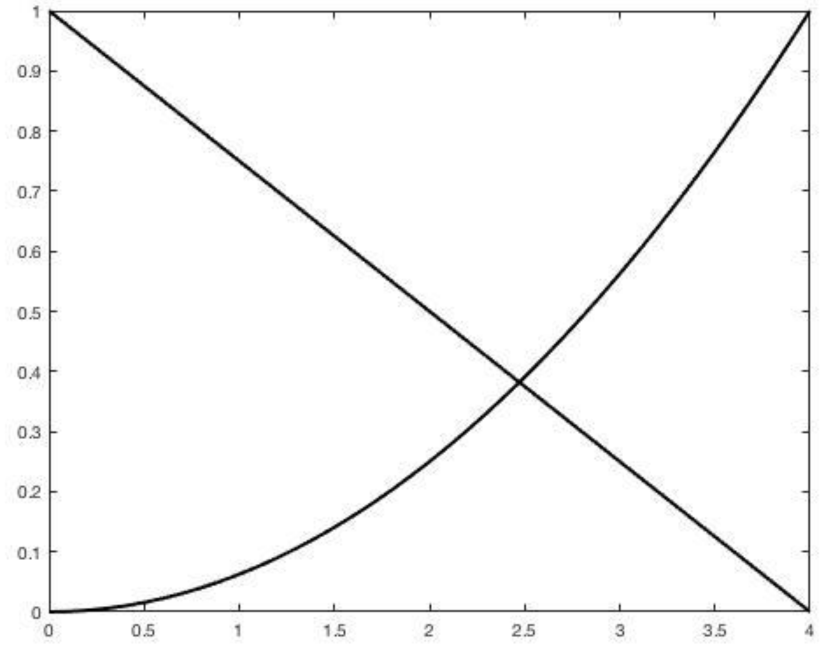


Figure 4.1 Values of a and d

We consider different cases and in each case we assign different values for *a* and *d* and different time frames for which the device will stay each of the states.

1. In this scenario, we consider a device in which we assign the calculated values to standby costs for different intermediate states as, a_1=0.75 a_2=0.50 and a_3=0.25, and

also, the values of switching costs as, d_1= 0.0625, d_2= 0.25 and d_3= 0.56 using the above formulae. We substitute these values in

$$z_i = \frac{d_{i+1} - d_i}{a_i - a_{i+1}}$$

Where *i* can be 0,1,2 or 3 (since there are five states), to get the *z* values. The z values come out to be z_1 = 0.25, z_2 = 0.75, z_3 = 1.25, and z_4 = 1.75.

The optimal graph with the above values is as shown:

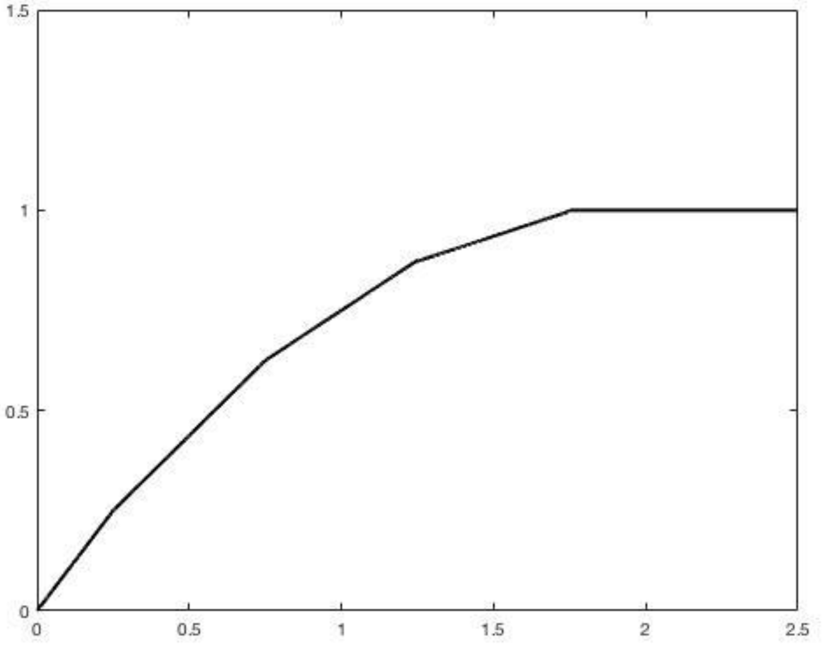


Figure 4.2 Cost in optimal algorithm

For the online model we use the following *t* values:

t_0 =0.090

t_1 =0.33

t_2 = 0.734

$t_3 = 1.75$.

When we compare the online cost with the optimal algorithm, the worst CR comes out to be 1.731, the graph is as shown below:

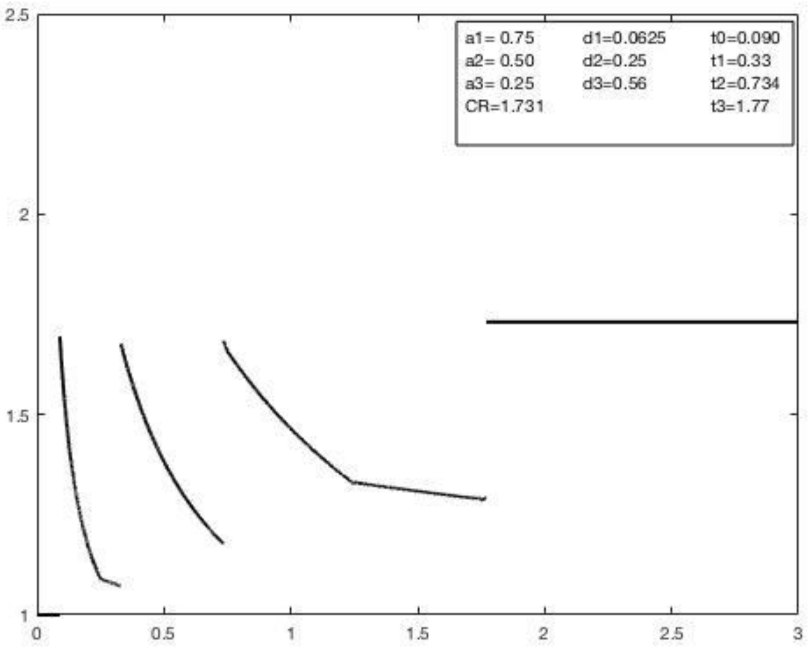


Figure 4.3 CR in Five States Case 1.

2. In this scenario, we consider a device in which we assign values to standby costs for different intermediate states as, a_1=0.95 a_2=0.90 and a_3=0.85, and also, the values of switching costs as, d_1= 0.05, d_2= 0.10 and d_3= 0.15 using the above formulae. We substitute these values in

$$z_i = \frac{d_{i+1} - d_i}{a_i - a_{i+1}}$$

Where *i* can be 0,1,2 or 3 (since there are five states), to get the *z* values. The z values come out to be $z_1 = 1$, $z_2 = 1$, $z_3 = 1$, and $z_4 = 1$

For the online model we use the following *t* values:

$t_0 = 0.05$

$t_1 = 0.0967$

$t_2 = 0.1435$

$t_3 = 0.88$

When we compare the online cost with the optimal algorithm, the worst CR comes out to be 2.01, the graph is as shown below:

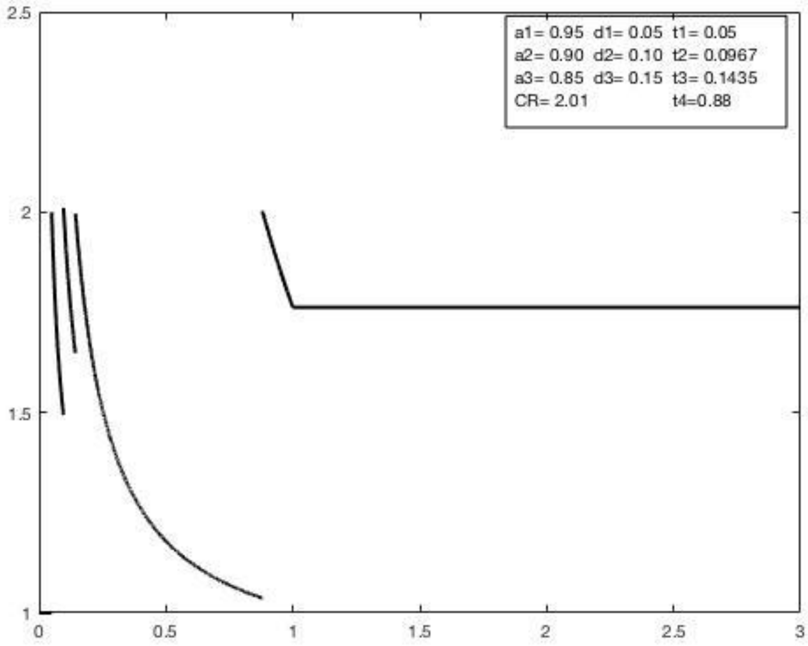


Figure 4.4 CR in Five States Case 2.

CHAPTER 5

AN EFFECTIVE SUSPEND/HIBERNATE SYSTEM

In this system, we propose a five state device in which there are ON, OFF and the following intermediate states:

- Turn off screen
- Suspend
- Hibernate

When the device is in the ‘turn off screen’ state, even though the screen of the device is turned off but the processor is still consuming the power. From this state if the device has to go to the ON state, the switching cost will be minimal as everything is running on the back-end. Considering these factors, we assign values 0.60 and 0.095 to standby and switching cost respectively.

After this, the device will switch to the ‘Suspend’ state, which has lower standby cost than that of the ‘Turn off screen’ state as all the peripherals are now switched to lower power mode. Even at low power mode the processor is still running since the device is yet to save all the work in RAM. Considering these factors, we assign values 0.25 and 0.45 to standby and switching cost respectively.

The device will then switch to the ‘Hibernate’ state which has even lower standby cost than the other two intermediate states since now the device will save all the work on RAM and then shuts down all the peripherals so that the next time the device is turned on, the users can continue from where they left. The values of a and d in this state will be 0.125 and 0.68 respectively.

The following table shows the different states and their respective standby and switching costs:

State	**Standby (COST)**	**Switching (COST)**
OFF	0 (a_4)	1 (d_4)
Hibernate	0.125 (a_3)	0.68 (d_3)
Suspend	0.25 (a_2)	0.45 (d_2)
Turn Off Screen	0.60 (a_1)	0.095 (d_1)
ON	1 (a_0)	0 (d_0)

Table 5.1 Effective Five State System

Now substituting the above values, we find the *z* values as following:

$$z_0 = \frac{d_1}{1-a_1} = 0.2375$$

$$z_1 = \frac{d_2-d_1}{a_1-a_2} = 1.0143$$

$$z_2 = \frac{d_3 - d_2}{a_2 - a_3} = 1.8400$$

$$z_3 = \frac{1 - d_3}{a_3} = 2.560$$

With all these z values the five state optimal graph will look like below:

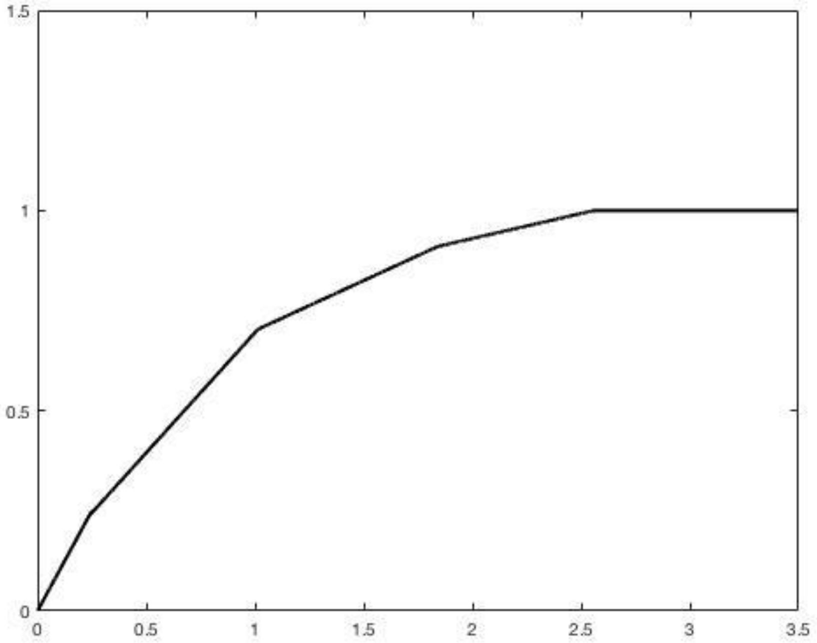


Figure 5.1 Five State Offline

From the above z values, we get the values of t using the heuristic from the previous chapter:

$t_0 = 0.1255$

$t_1 = 0.7339$

t_2 = 1.0382

t_3 = 2.5600

Using the standby cost, switching costs and t values we get the online algorithm graph as shown:

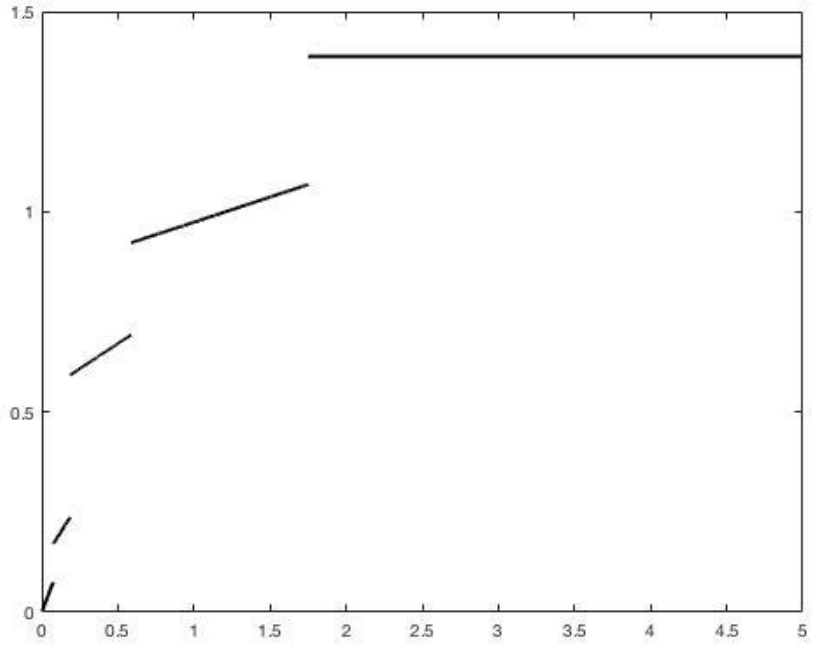


Figure 5.2 Five State Online

Each line represents respective state of the algorithm, starting from bottom right which represents ON state then Turn Off Screen state and so on. When the device reaches the OFF state the graph represents it by the constant line as at that point whenever the request comes the cost to go to the ON state is same.

In this case, the CR comes out to be 1.756.

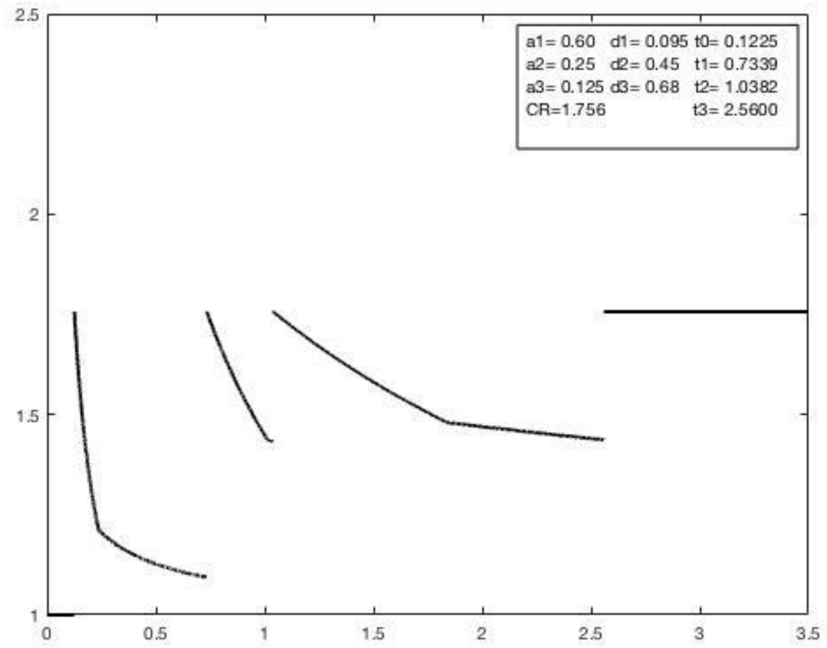


Figure 5.3 Competitive Ratio

CHAPTER 6

HEURISTIC FOR OPTIMAL ONLINE ALGORITHM

We need the best values for the times t_0 (time for which the device stays in the ON state) and t_i (time for which the device stays in the INT states for i = 1, 2 or 3). After this the device goes to the OFF state. At different states, there will be different range of values for the competitive ratio. To get the values for time t_0, t_1, t_2 and t_3 at which the competitive ratio (CR) is nearly equal for the different states we follow the pseudo code below:

1. Select the initial values of $t_{0,}$ $t_{1,}$ t_2 and t_3 such that $t_0=\frac{z_0}{3}$ $t_1=\frac{z_1}{3}$, $t_2=\frac{z_2}{3}$ and $t_3=z_3$.
2. Choose a CR value from the range of values between time t_0 and t_1 and replace the value of t_0 corresponding to the CR value selected.
3. Now pick a CR value from the range of values between t_1 and t_2 such that it is approximately equal to the value chosen in Step 2.
4. For the CR value chosen in Step 3, select the corresponding value of time t and update $t_1$with that value.
5. If no approximate value exists in Step 3, repeat Steps 2 to 3 and choose a different value for CR until Step 3 is satisfied.
6. Repeat from Step 3 to Step 5 for range t_2 to t_3 and update the values of t_2.
7. Repeat Steps 2 to 6 until we find the minimum CR.

6.1 An Example

We consider an example and try to apply these heuristics for better competitive ratio.

The value of the switching cost and standby cost are mentioned in the table below.

State	Standby (COST)	Switching (COST)
OFF	0	1
INT3	0.25	0.60
INT2	0.40	0.40
INT1	0.55	0.225
ON	1	0

Table 6.1 Values a & d for Five state Heuristics

Considering the above values of *a* and *d* we calculate the *z* values which comes out be:

$z_0 = 0.5$

$z_1 = 1.1667$

$z_2 = 1.3333$

$z_3 = 1.600$

Using the above *z* values we will now calculate the *t* values using the step 1 of the heuristic, from where the *t* values comes out to be as following.

$t_0 = 0.1666$

$t_1 = 0.3889$

$t_2 = 0.4444$

$t_3 = 1.600$

Using the above values, the CR graph looks like below:

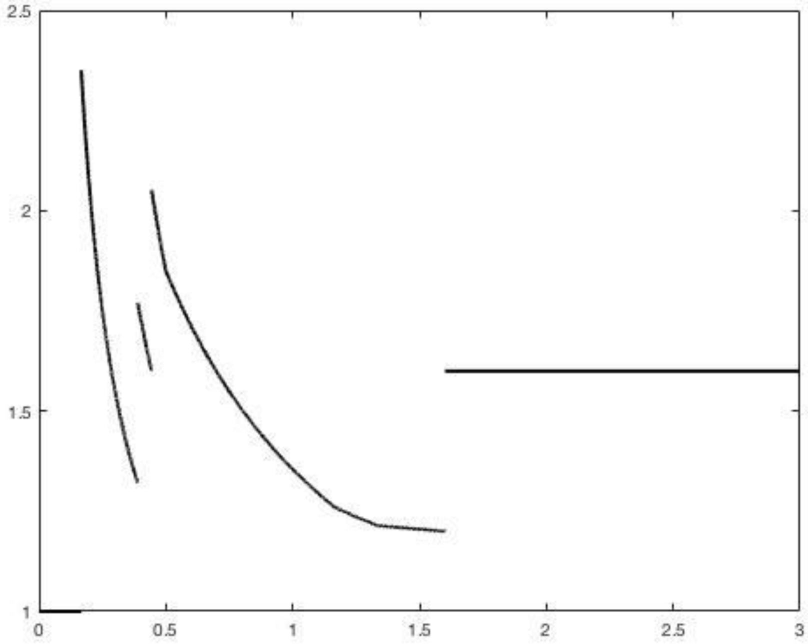


Figure 6.1 CR with initial t values.

In this case the competitive ratio comes out to be 2.35.

As we can see from the graph that the competitive ratio can be brought down as we have as there are time where the CR in each of the state can be lower, keeping that in mind that when the all the states have the equal or almost equal CR that is the best CR we can get from a given device.

Now we select a value from CR from range of values between t_0 and t_1 and then replace the value of t_0 with the corresponding time value of the CR value selected.

So we try steps 3 to 6 of heuristics and following values of *t* were selected

$t_0 = 0.2666$

$t_1 = 4019$

$t_2 = 0.5885$

$t_3 = 1.60$

Using these values of *t* the graph looks like as follows.

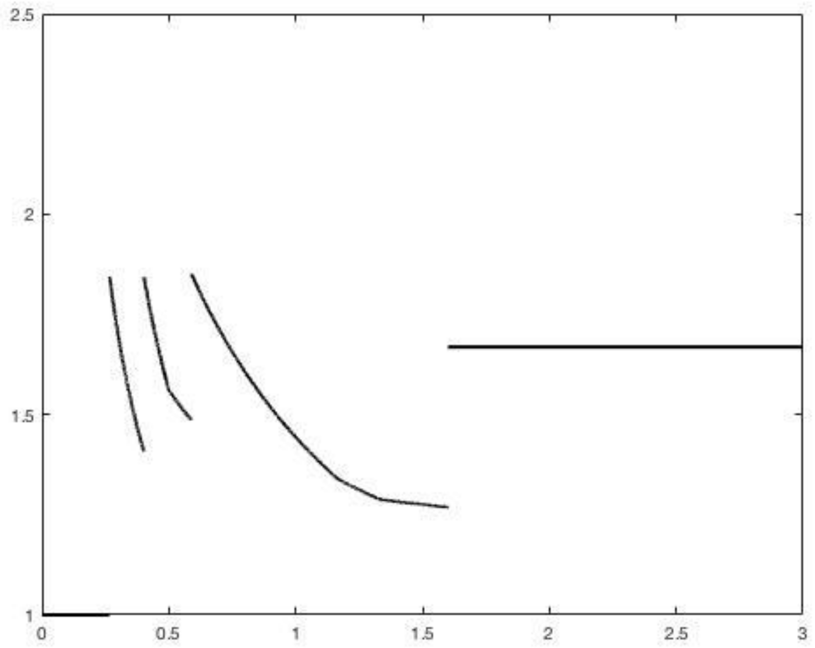


Figure 6.2 CR after applying Heuristics once.

Here the CR is 1.85. As we can see that the worst CR of each intermediate state is almost equal but the CR of the OFF state is better than that of the INT states so we will try step 7 of the heuristics which is repeating from step 2 to 6 until all the CR are approximately equal.

The t values for which the CR for each state is almost equal are:

$t_0 = 0.312$

$t_1 = 0.462$

$t_2 = 0.76$

$t_3 = 1.60$

The graph corresponding to these t values looks like below:

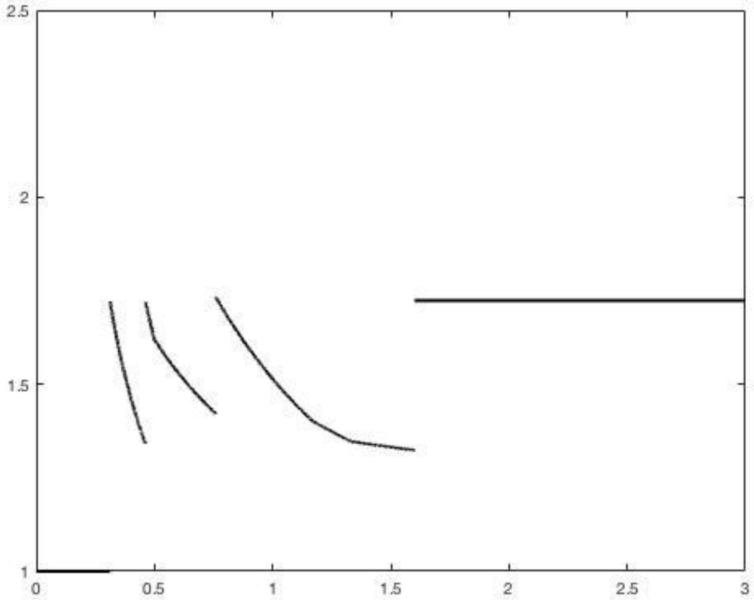


Figure 6.3 Best CR after repeating Heuristics.

As we can see that the CR in each state is roughly equal which conveys that for these specific values of *a* and *d* these are the best *t* values for which the CR comes out to be 1.732.

CHAPTER 6

CONCLUSION

Clearly, as shown by our work, a five states system behaves much better than a two state system, where the best competitive ratio is 2; thus it is useful to have intermediate states. We also note that a five states system may well be most realistic, with states like ON, Turn Off screen, Suspend, Hibernate, and OFF.

Secondly, algorithmically, our heuristic is more compact as compared to the general approach in Augustine et. al [4] where they use the ε tolerance to find the best CR and the complexity of the algorithm is $O(k^2/\log k)\log(1/\varepsilon)$ where k+1 are the number of states and which can have value of hundred as well which is not practical. We note that in practice likely very states exist, i.e. k is small. As shown five states systems can indeed be solved quite efficiently using our heuristic.

Finally, the work here is useful in the context of modeling power-down situations when auxiliary conventional power turbines must be on standby to fill in lack of supply from renewables. Here the issue is that the behavior of renewable power sources such as wind and solar is highly unpredictable. Thus our approach based on the competitive analysis model is appropriate and can give use insights. In summary, for the smart grid problem of multiple conventional power plants with multi-state power generation model [11] the presented power-down model can be efficiently addressed.

APPENDIX A

CODE FOR CALCULATING Z VALUES

```
public class Calculating {


        public static void main(String[] args){
                float[] z=new float[5];


                float[] a = {1.0f, 0.90f,0.83f,0.77f,0.0f};
                float[] d = {0.0f, 0.05f, 0.1f,0.15f,1.0f};
                DecimalFormat df = new DecimalFormat("0.00000");


                for(int j=0;j<4;j++){
                        z[j]= (d[j+1] - d[j])/ (a[j] - a[j+1]);
                System.out.println("Values of z0: "+ df.format(z[0]) );
                System.out.println("Values of z1: "+ df.format(z[1]) );
                System.out.println("Values of z2: "+ df.format(z[2]) );
                System.out.println("Values of z:3 "+ df.format(z[3]) );




                }
                        }
                }
```

APPENDIX B

FIVE STATE OPTIMAL

```
import java.io.BufferedWriter;
import java.text.DecimalFormat;

public class FiveOffOPT {
        // Hard coded value of a and d of ON and OFF state.
        static final double[] STATE_ON = {1.0f,0.0f};
        static final double[] STATE_OFF = {0.0f,1.0f};

public float[] OfflineOPT() {

        // Hard coded value of a and d of INT states.
        float a1= 0.90f, a2=0.83f, a3=0.77f ;
        float d1= 0.05f, d2=0.1f, d3=0.15f;

        BufferedWriter bw;

        // Creating an array to store 510000 values.
```

```
      float[] x_val = new float[51000];
 float[] y_val = new float[51000];

// Declaring the decimal format to get 4 places of decimal precision.

                DecimalFormat format = new DecimalFormat("0.0000");

      int x=0;

      // calculating the cost when the device is in ON state.
      // Here the i values is the z0 value.

      for(float i=0.0001f;i<0.5000f;i+=0.0001f){

                //double difference = i-0;
                double calc_on = (i*STATE_ON[0])+STATE_ON[1];
                System.out.println("Power in ON State "+format.format(calc_on));
                x_val[x]=i;
                y_val[x]=Float.parseFloat( format.format(calc_on));
                x++;
                if (i==0.075){
                          System.out.println("Value at 0.075" +calc_on);
```

```
        }

}

// calculating the cost when the device is in INT1 state.

// Here the i values used are z0 and z1.

int y=x;

for(float i=0.5000f;i<0.71429f;i+=0.0001f){


        //double difference = i-0;

        double calc_int1 = (i*a1)+d1;

        System.out.println("Power in INT1 State "+format.format(calc_int1));

        x_val[y]=i;

        y_val[y]=Float.parseFloat( format.format(calc_int1));

        y++;


}

// calculating the cost when the device is in INT2 state.

// Here the i values used are z1 and z2.

int z=y;

for(float i=0.71429f;i<0.8333f;i+=0.0001f){


        double calc_int2 = (i*a2)+d2;
```

```
System.out.println("Power in INT2 State "+format.format(calc_int2));
        x_val[z]=i;
        y_val[z]=Float.parseFloat( format.format(calc_int2));
        z++;
    }
    // calculating the cost when the device is in INT3 state.
    // Here the i values used are z2 and z3.
    int a=z;
    for(float i=0.8333f;i<1.10390f;i+=0.0001f){
        //double difference = i-0;
        double calc_int3 = (i*a3)+d3;
        System.out.println("Power in INT3 State "+format.format(calc_int3));
        x_val[a]=i;
        y_val[a]=Float.parseFloat( format.format(calc_int3));
        a++;
    }

    // calculating the cost when the device is in OFF state.
    // Here the i values used are z4 and till the time we want to find the cost.
    int b=a;
    for(float i=1.10390f;i<=5.0000f;i+=0.0001f){
        double calc_off = (i*STATE_OFF[0])+STATE_OFF[1];

        System.out.println("Power in OFF State "+format.format(calc_off));

        x_val[b]=i;

        y_val[b]=Float.parseFloat( format.format(calc_off));

        b++;

    }

    return y_val;

    }

}
```

APPENDIX C

FIVE STATE ONLINE

```
import java.io.BufferedReader;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.text.DecimalFormat;

import java.text.Format;

import java.util.Scanner;

import java.io.FileWriter;

import java.io.BufferedWriter;


public class FiveONOPT {
    public float[]  OnlineOPT(){
             // Hard coded value of a and d of ON and OFF state.
              final float[] STATE_ON = {1.0f,0.0f};
        final float[] STATE_OFF = {0.0f,1.0f};

        // inserting the t values

        float ta = 0.957f;
        float tb = 0.989f;
        float tc = 0.990f;
        float td = 1.10390f;



        float t_on=ta;
  float t_int1=tb-ta;
  float t_int2=tc-tb;
 float t_int3=td-tc;

       //Declaring the array to store 50000 values

float[] x_val = new float[51000];
    float[] y_val = new float[51000];
```

```
        Scanner sc;
        sc = new Scanner(System.in);
            FileInputStream fis;
            FileReader fr;
  FileWriter fw;
            BufferedReader br;
  BufferedWriter bw;
            float
a1=0.0f,a2=0.0f,a3=0.0f,d1=0.0f,d2=0.0f,d3=0.0f,t1=0.0f,t2=0.0f,t3=0.0f,t4=0.0f;

            //Creating a text file which will have the output for online state

            try{
                  File f = new File("/Users/govind/Desktop/PDM.txt");
   File f_output = new File("/Users/govind/Desktop/PDMout.txt");
                  fr  = new FileReader(f);
   fw = new FileWriter(f_output);
                  br = new BufferedReader(fr);
   bw = new BufferedWriter(fw);
                  String line;

            //Reading the values of a and d from file
        while((line=br.readLine())!=null){

                if(line.contains("a1")){
                line = line.replace(" ", "");
                a1 = Float.parseFloat(line.substring(line.indexOf("=")+1));
                //System.out.println("Got a1");
                line=null;
            }

        else if(line.contains("d1")){
                    line = line.replace(" ", "");
                    d1 =
Float.parseFloat(line.substring(line.indexOf("=")+1));
                    //System.out.println("Got d1");
            }

        else if(line.contains("a2")){
                        line = line.replace(" ", "");
                        a2 =
Float.parseFloat(line.substring(line.indexOf("=")+1));
            }
        else if(line.contains("d2")){
                line = line.replace(" ", "");
                d2 = Float.parseFloat(line.substring(line.indexOf("=")+1));


            }
        else if(line.contains("a3")){
                line = line.replace(" ", "");
                a3 = Float.parseFloat(line.substring(line.indexOf("=")+1));
             }
        else if(line.contains("d3")){
                    line = line.replace(" ", "");
                    d3 =
Float.parseFloat(line.substring(line.indexOf("=")+1));
                }
        }
        if(a1==0.0f||a2==0.0f||a3==0.0f||d1==0.0f||d2==0.0f||d3==0.0f){
            System.out.println("Missing values");
            System.exit(1);
        }


  int i=0;
```

```
        // Calculating the cost for 50000 values
                                 for (float conv_inp=0.0001f;conv_inp<=5.0000f ;conv_inp
+=0.0001 ){
    // Declaring the decimal format to get 4 place decimal presion.
                         DecimalFormat format = new DecimalFormat("0.0000");
                         bw.write("Current time "+ format.format(conv_inp));
                         bw.newLine();


        System.out.println("Current time "+ format.format(conv_inp));


                          float calc=0.000f;
                          if(conv_inp<=t_on){
 //Calculating Cost if request come while the device is in ON State
                             calc = conv_inp*STATE_ON[0];


                System.out.println("Cost for ONLINE:"+format.format(calc));
                          bw.write("Cost for ONLINE:"+format.format(calc));
             bw.newLine();
             }
 //Calculating Cost if request come while the device is in INT1
                          else if(conv_inp<=(t_on+t_int1)){
                             float calc_on = t_on*STATE_ON[0];
```

```
                                   float calc_int1 = ((conv_inp-t_on)*a1)+d1;

                                   calc = calc_on+calc_int1;

                                   System.out.println("Cost for ONLINE:"+format.format(calc));

                                 bw.write("Cost for ONLINE:"+format.format(calc));

                        bw.newLine();

                       }

//Calculating Cost if request come while the device is in INT2 State

                                 else if(conv_inp<=(t_on+t_int1+t_int2)){

                                   float calc_on = t_on*STATE_ON[0];

                                   float calc_int1 = t_int1*a1;

                                   float calc_int2 = ((conv_inp-(t_on+t_int1))*a2)+d2;

                                   calc = calc_on+calc_int1+calc_int2;

                                   System.out.println("Cost for ONLINE:"+format.format(calc));

                                bw.write("Cost for ONLINE:"+format.format(calc));

                        bw.newLine();

                       }

                               //Calculating Cost if request come while the device is in INT3  State

                                 else if(conv_inp<=(t_on+t_int1+t_int2+t_int3)){

                                   float calc_on = t_on*STATE_ON[0];

                                   float calc_int1 = t_int1*a1;

                                   float calc_int2 = t_int2*a2;

                                   float calc_int3 = ((conv_inp-(t_on+t_int1+t_int2))*a3)+d3;
```

```
          calc = calc_on+calc_int1+calc_int2+calc_int3;
          System.out.println("Cost for ONLINE:"+format.format(calc));
        bw.write("Cost for ONLINE:"+format.format(calc));
 bw.newLine();
 }
       //Calculating Cost if request come while the device is in OFF State
        else if(conv_inp>(t_on+t_int1+t_int2+t_int3)){

          float calc_on = t_on*STATE_ON[0];
          float calc_int1 = t_int1*a1;
          float calc_int2 = t_int2*a2;
          float calc_int3 = t_int3*a3;
          float calc_off = STATE_OFF[1];
          calc = calc_on+calc_int1+calc_int2+calc_int3+calc_off;
          System.out.println("Cost for ONLINE:"+format.format(calc));
        bw.write("Cost for ONLINE:"+format.format(calc));
 bw.newLine();
 }



System.out.println("Output :"+ format.format(calc));
```

```
          bw.write("Output :"+ format.format(calc));
            bw.newLine();
            bw.newLine();
               // Putting the values in the variable.
         x_val[i] =Float.parseFloat(format.format(conv_inp));
         y_val[i] =Float.parseFloat(format.format(calc));;
      System.out.println();
      i++;


      if (i==0.075){
        System.out.println("Value at 0.075" +calc);
       }
                      }


bw.close();


                      }
              // NO file found Exception Error handling


              catch(FileNotFoundException e){
                      e.printStackTrace();
              }
```

```
 // Input output file exception.


    catch (IOException e) {

        e.printStackTrace();

    }


return y_val;

}

    }
```

APPENDIX D

CODE FOR CALCULATING CR

```
import java.text.DecimalFormat;

public class Combine {

    public static void main(String[] args){

        FiveOffOPT off = new FiveOffOPT();

        FiveONOPT on = new FiveONOPT();

        float[]  x = off.OfflineOPT();

        float[]  y = on.OnlineOPT();

        float[] val = new float[51000];

        float[] cal = new float[51000];

        float cal1;

        float cal2;

        float cal3;

        float cal4;

        float t_val = 0.000f;

        DecimalFormat format = new DecimalFormat("0.0000");

    System.out.println("Length " +y.length);

        for(int i = 0; i<x.length; i++){
```

```
        val[i] = Float.parseFloat(format.format(t_val));

        cal[i] = y[i] / x[i];

        t_val+=0.0001f;

        System.out.println("Calc "+cal[i]);

    }
```

REFERENCES

[1]. Sandy Irani and Pruhs, P. "Algorithmic Problems in Power Management". SIGACT

News, 36(2), 63–76 (2005).

[2]. Sandy Irani and Karlin, A. "On Online Computation" In Dorit Hochbaum, editor, Approximations for NP-Hard Problems. PWS Publishing Co.,1995.

[3]. James Andro-Vasko, Wolfgang Bein, Hiro Ito, and Dara Nyknahad "Evaluation of online power-down algorithms". In Proceedings of the 12th International Conference on Information Technology (ITNG) (2015).

[4]. John Augustine, Sandy Irani, and Chaitanya Swamy. "Optimal power-down strategies". SIAM Journal on Computing, 37(5), 1499–1516 (2008).

[5]. Anna Karlin, Mark Manasse, Lyle McGeoch, and Susan Owicki. "Competitive randomized algorithms for non uniform problems". Algorithmica, 11, 542–571 (1994).

[6]. Susanne Albers. "Energy-efficient algorithms". Communications of The ACM, 53:86–96, 2010.

[7]. Wolfgang Bein, Naoki Hatta, Nelson Hernandez-Cons, Hiro Ito, Shoji Kasahara, and Jun Kawahara. "An online algorithm optimally self-tuning to congestion for power management problems". In Proceedings of the 9th

International Conference on Approximation and Online Algorithms. pages 35-48. Springer-Verlag. 2012.

[8]. Sandy Irani, Sandeep Shukla, and Rajesh Gupta. “Online Strategies for dynamic power management in Systems with multiple power-saving states”. ACM Transactions on Embedded Computing Systems, 2(3). 2003.

[9]. Sandy Irani and Gaurav Singh. “An overview of the competitive and adversarial approaches to designing dynamic power management strategies”. IEEE Transactions Very Large Scale Integration. 13(12). December 2005.

[10]. L. Benini, A. Bogio, and G. De Micheli. “A survey of design Techniques for system-level dynamic power management”. IEEE Trans. VLSI Syst., 8:299-316. 2000.

[11]. Wolfgang Bein, Bharat B. Madan, Doina Bein, Dara Nyknahad (2016) 'Algorithmic Approaches for a Dependable Smart Grid', in Shahram Latifi (ed.) *Information Technology: New Generation*, Springer International Publishing, pp. 677-687.

[12]. S. Phillips and J. Westbrook. chapter 10 of Algorithms and Theory of Computation Handbook, chapter On-line algorithms: “Competitive analysis and beyond”. CRC Press, Boca Raton, 1999.

[13]. McGeoch, L. A. and Sleator, D.D. “A Strongly Competitive Randomized Paging Algorithm”. Technical Report CMU-CS-89-122, School of Computer Science, Carnegie Mellon University 1989.

[14]. Lili Ding, Chunlin Xin, Jian Chen, (2005) “A Risk Reward Competitive Analysis of the Bahncard Problem”, in Nimrod Megiddo, Yinfeng Xu, Binhai Xu (ed.) *Algorithmic Applications in Management,* Springer International Publishing, pp. 37-45.

[15]. A. Karlin, M. Manasse, L. McGeoch, and S. Owicki, Randomized competitive algorithms for non-uniform problems, in Proceedings of the ACM–SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 1990, pp. 301–309.

CURRICULUM VITAE

Graduate College

University of Nevada, Las Vegas

Govind Pathak

Degrees:

Bachelor of Technology in Information Technology, 2014

Guru Nanak Dev Engineering College, Punjab, India.

Master of Science in Computer Science, 2016

University of Nevada Las Vegas

Thesis Title: Analysis of Power Down Systems with Five States.

Thesis Examination Committee:

Chair Person, Dr. Wolfgang Bein.

Committee Member, Dr. Ajoy K. Datta, Ph.D.

Committee Member, Dr. Laxmi Gewali, Ph.D.

Graduate College Representative, Dr. Venkatesan Muthukumar, Ph.d.