Dynamic Programming Algorithm for Smart Mobile House Management

Sonny Lazuardi Hermawan (13511029)[1]
Computer Science/Informatics
School of Electrical Engineering and Informatics
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13511029@std.stei.itb.ac.id

***Abstract*— Smart house is now possible by the support of current technology. People now are interacting more and more with digital devices in daily life. Smart house enable us to control all of our electric devices from mobile device in our hand anywhere and anytime. Broomlight is a smart house mobile management that enable us to manage electric device in our house through a mobile application connected to the internet. Broomlight has three main components to operate: controller hardware, mobile application, and server. There are some problems that make people to use smart house such as lack of time to manage home because people now are very busy, expensive electricity usage cost because of unmonitored electricity usage, no warning of dangerous threats or disaster happen to our house. This paper is focused on feature of a power management plan for our house based on priority from user. The feature can give suggestion to user about what electrical devices should be turned on so that user get the highest priority devices first but still under the budget for electric usage of user. The algorithm used in this paper is dynamic programming, specifically for knapsack problems.**

***Index Terms*— Smart House, Dynamic Programming, Mobile Application, Power Management, Cost Budget, Power Priority.**

I. Introduction

Nowadays, people are interacting more and more with digital devices in daily life. There are some people that even can't undergo a day without digital devices such as hand phone, TV, and others. Technology becomes more handy and mobile. Many digital devices now can be accessed remotely such as TV, and air conditioner. Those digital devices have their own controller. How if we can control all devices in one place?

Smart house technology is one of realization of home automation ideals using a specific set of technology. First "home computer" was an experimental system in 1966. Smart house project was initiated in 1980 by National Research Center of the National Association of Home Builders (NAHB) with the cooperation of a collection of major industrial partners. With the development of electric technology, smart house is now possible.

This idea of smart house is an interesting topic because it has some advantages that home occupants can't do when they are on office or other places. It is also because people are becoming busier and mobility of people is increasing. Some of these problems may happened in our daily life. First, people now has many things to do and they are busy to do work, activities that make them leave their home for a long time. They sometimes forget to turn on the light when day turn into night and may open opportunity for criminals to break their home. And also how if you have a lot of light in your house and you have to turn on all of it in night, and turn off all of it in the morning. You will take a lot of time and energy for doing that every day. Second, we sometimes get an expensive bill, or spend a useless cost for electricity usage because it is not monitored. We don't really know the detail about the electricity usage and what kind of electricity that we use in one month. Third, when we leave our home, we don't know if something bad happen to our house. Our house can be on fire or being robbed and we don't know it and cannot prevent it to happen.

II. Application

Broomlight is a smart house mobile management that enable us to manage electric device in our house through a mobile application connected to internet. Broomlight has three main components to operate.

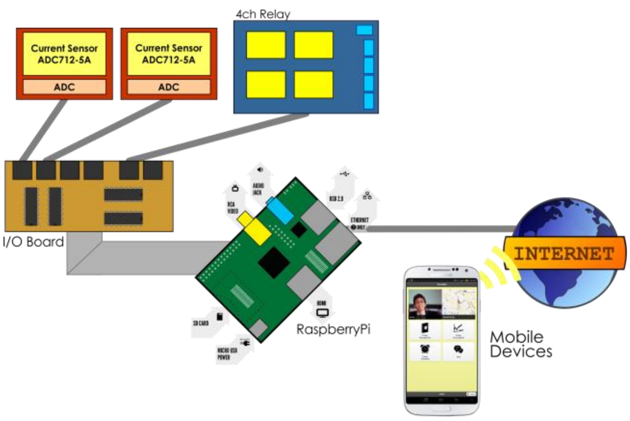

Figure 1. Scheme of Broomlight system

1. Controller Hardware
   Controller hardware of Broomlight is a microcomputer that can control and turn on or turn off electric devices that are connected to the hardware. The example of electric devices that can be connected to the controller is light bulb, TV, dispenser, AC, computer, and so on. Controller hardware is also connected to internet.

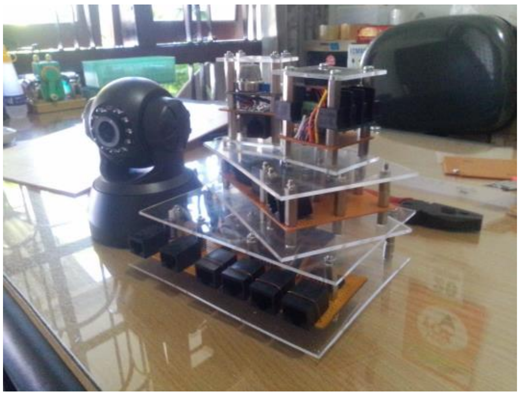

Figure 2. Hardware Controller Broomlight

2. Mobile Application
   Broomlight has a mobile application for Android, BlackBerry, and Windows Phone. This mobile application is a hybrid application from web application that is converted to mobile application using IBM Worklight. This mobile application also need an internet connection and login to Broomlight server.

3. Server
   Broomlight has the server to connect the controller hardware and mobile application. After connected, user can turn on or turn off electric device from their mobile application. To connect we need a login authorization so that we can prevent other people to control our house.

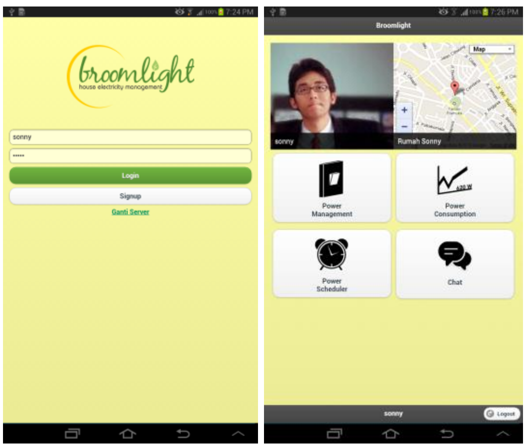

Figure 3. Display of Login dan Home screen on Broomlight Mobile App

One of the feature of Broomlight is to control (turn on and turn off) electricity device using those three components. But, we need a better application to control and monitor electricity usage of our electric devices. One of the important feature is electricity usage management plan that give us a flexibility to control and monitor our electricity device.

This application has a sensor to detect the power usage of an electric device. It uses current detection sensor to get how much currents that electric device uses and convert it to power usage. The conversion is based on the following formula.

$$P_{(kW)} = \sqrt{3} \times PF \times I_{(A)} \times V_{L\text{-}L(V)} / 1000$$

PF: power factor
I: current
V: voltage.

Broomlight is running on node.js server and use socket.io for communication between server and mobile application. The development toolkit for this application is a full-stack javascript development for server and mobile application. Mobile application is ported from web application to mobile application using IBM Worklight. The controller hardware use Java application running on Raspberry Pi Linux OS.

III. Electricity Usage Cost

In Indonesia, the electricity cost is based on PLN (Perusahaan Listrik Negara) rule. The cost is an average calculation of electricity usage.

Cost = Rp. 864,2,- per kwh.
Or equal to
864,2 / 1000 = Rp. 0,8642,- per Watt.

For example, lights bulb in a house have the power of 5 Watt ~ 220 Volt, and averagely turned on for 12 hour/day:

Usage per day:

= ((5 Watt / 1000) x 12 hour) x Rp. 864,2,-
= (0,005 KWh x 12 hour) x Rp. 864,2,-
= 0,06 KWh x Rp. 864,2,-
= Rp. 51,85,-

Usage per month:
= Rp. 51,85,- x 30
= Rp. 1.555,56,-

IV. Scope and Goal

The scope of smart house that will be explained in this paper is an electric device controller that can be accessed from a mobile application. We can turn on or off our electric device from a mobile application. The problem that arise from controlling our electric device is the monitoring and planning of electricity usage. User should be know how much cost they spend for electric device usage and the application should suggest what are the electric device that can be turn on that give a minimal cost.

The application should help us to manage and monitor the electricity usage that fit into the user-defined criteria. Users can give budget of power usage and give some criteria on each electric device such as time used for this electric device and how much electric current that will be used for each electric device.

So, the goal of this paper is make an algorithm that give profit to user by minimizing the electricity usage cost under the budget for cost limit.

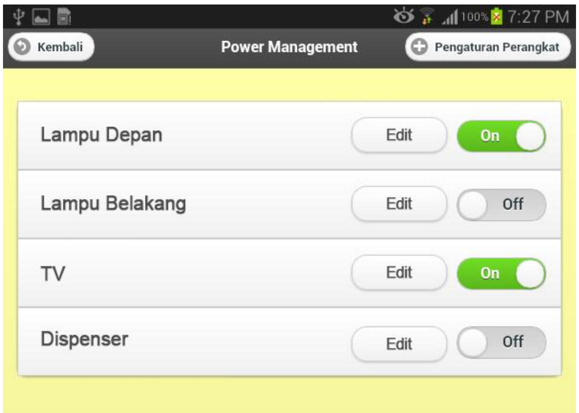

Figure 4. Power Management Panel

For each electric device we has switch to turn off or turn on the light. The additional feature that will be the goal is power management plan to give suggestion what are the electric devices should we turn on. In order to get the suggestion, we need some information for each electric device. First, how much time that the electric device will be turned on, and the power for that electric device (already calculated from sensor). Second, what is the priority of electricity device to be turned on. Lastly, application need user input of the cost that want to be spent for the electricity usage per day.

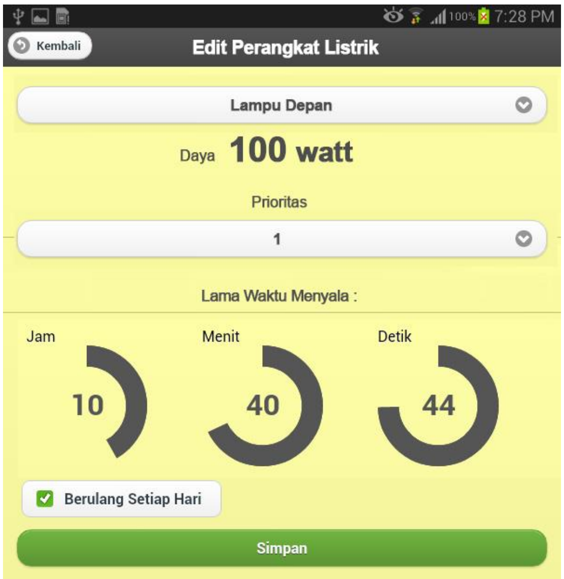

Figure 5. Setting for an electric device

First, user will input the name of electric device. Second, the priority of the electric device. The priority is a range of number from 1 to 10. 1 means very important, and 10 means not important (in the calculation priority is inversed 1 means not important and 10 means very important). Lastly, user will input the time for the electric device to be turned on.

V. Dynamic Programming

Dynamic programming solve problems by combining the solutions to subproblems. (“Programming” in this context refers to a tabular method, not to writing computer code). Dynamic programming applies when the subproblems overlap – that is, when subproblems share subsubproblems. A dynamic programming algorithm solves each susbsubproblem just once and then saves its answer in a table, thereby avoiding the work of recomputing the answer every time it solves each subsubproblem. Dynamic programming usually used for optimization problems. Such problems can have many possible solutions. Each solution has a value and we wish to find a solition to the problem, as opposed to the optimal solution. [2]

At glance, dynamic programming looks similar to Divide and Conquer algorithm because the characteristics is dividing problems into smaller problems. But, the main differences between divide and conquer is the definition of subproblem and the solving strategy. Dynamic programming design will be very useful when the sub problem that is overlapped grow exponentially as a function of input.

VI. Analysis

The problem above is an optimization problem in which we are trying to get the best solution of the electric devices that need to be turned on. The problem is similar with the knapsack problem. In knapsack problem, we have a knapsack which has capacity of weight to fill. We also have some stuff that will be taken that have their own profit and weight. The goal of knapsack problem is getting the profit as high as possible but still meet the knapsack capacity of weight.

Knapsack problem or rucksack problem is a problem in combinatorial optimization: Given a set of items, each with a mass and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to given limit and the total value is as large as possible. It derives from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items.

Similar to knapsack, power management plan also have the similar case. We have a budget for electric usage cost per day (equal to knapsack capacity). We also have some electric device which has power and time that will be turned on (equal to stuff that will be taken). For comparison of this two problem, take a look at the following tables.

Knapsack Problem
n = 3
M = 5 kg

Stuff	w_i (kg)	p_i (rupiah)
1	2	6500
2	3	8000
3	1	3000

Table 1. Knapsack Problem

Power Management Problem
n = 3
M = Rp 500 (budget per day)
Cost = Power Usage x 0.8 rupiah
Priority = 1 (not important) – 10 (very important)

Device	Power (w)	Time (h)	Power Usage (wh)	Cost (Rp)	Priority (1-10)
Front Light	50	5	250	200	6
TV	125	3	375	300	8
Back Light	25	3	125	100	3

Table 2. Power Management Problem

From the table above we can see the similarity of knapsack problem and power management problem as optimization problem.

VII. Problem Solving

For the simple example of power management problem, we use the problem in table 2, we have

1. Step (k) is the process turning on electric device (there are 4 steps).
2. Status (y) is the remaining budget to pay the electricity usage cost after turn on electric device in previous step.

From the step-1, we turn on electricity device no 1 (Front Light) for every probability of cost until the maximum cost below the budget limit.

When we turn on electric device in step k, budget now is y - w_k. To spend the remaining budget, we apply the principle of optimality based on the optimum value from the previous step for remaining budget $y - w_k$ ($f_{k-1}(y-w_k)$).

Next, we compare the value of priority from the electric device in step k (pk) added by $f_{k-1}(y-w_k)$ with the priority filling only $k-1$device, $f_{k-1}(y)$.

If $pk + f_{k-1}(y - w_k)$ is less than $f_{k-1}(y)$, then electric device k will not be turned on. But, if it is bigger, then the electric device k will be turned on.

Recurrent relation for the problem is

$f_0(y) = 0$, y = 0, 240, 320, …, M (base)
$f_k(y) = -\infty$, y < 0 (base)
$f_k(y) = \max\{f_{k-1}(y), p_k + f_{k-1}(y-w_k)\}$, (recurrent)
$k = 1, 2, \ldots, n$

$f_k(y)$ is the priority optimum from the problem in step k for the remaining budget of y.
$f_0(y) = 0$ is the value for budget is full (no cost) with the remaining budget y.
$f_k(y) = -\infty$ is the value for negative remaining budget. Optimum solution for the problem is $f_n(M)$.

Below is the process of dynamic programming to solve the problem in table 2.

Before doing dynamic programming, to make it easy, we simplify the cost by dividing by 100, and we also divide the budget by 100. So we get the following table

n = 3
M = 5 (* 100 rupiah, budget per day)

Device	Cost (* 100 Rp)	Priority (1-10)
Front Light	2	6
TV	3	8
Back Light	1	3

Table 3. Power Management Problem

Step 1:
$f_1(y) = \max\{f_0(y), p_1 + f_0(y-w_1)\}$
$= \max\{f_0(y), 6 + f_0(y-2)\}$

y	Optimum Solution			
	$f_0(y)$	$6+f_0(y-2)$	$f_1(y)$	(x1*, x2*, x3*)
0	0	$-\infty$	0	(0, 0, 0)
1	0	$-\infty$	0	(0, 0, 0)

2	0	6	6	(1, 0, 0)
3	0	6	6	(1, 0, 0)
4	0	6	6	(1, 0, 0)
5	0	6	6	(1, 0, 0)

Step 2:

$f_2(y) = \max\{f_1(y), p_2 + f_1(y\text{-}w_2)\}$
$\quad\;\; = \max\{f_1(y), 8 + f_1(y\text{-}3)\}$

y	Optimum Solution			
	f_1(y)	8+f_1(y-3)	f_2(y)	(x1*, x2*, x3*)
0	0	8+(-∞)=-∞	0	(0, 0, 0)
1	0	8+(-∞)=-∞	0	(0, 0, 0)
2	6	8+(-∞)=-∞	6	(1, 0, 0)
3	6	8+0=8	8	(0, 1, 0)
4	6	8+0=8	8	(0, 1, 0)
5	6	8+6=14	14	(1, 1, 0)

Step 3:

$f_3(y) = \max\{f_2(y), p_3 + f_2(y\text{-}w_3)\}$
$\quad\;\; = \max\{f_2(y), 3 + f_2(y\text{-}1)\}$

y	Optimum Solution			
	f_2(y)	3+f_2(y-1)	f_3(y)	(x1*, x2*, x3*)
0	0	3+(-∞)=-∞	0	(0, 0, 0)
1	0	3+(-∞)=-∞	0	(0, 0, 0)
2	6	3+0=3	6	(1, 0, 0)
3	8	3+6=9	9	(1, 0, 1)
4	8	3+8=11	11	(0, 1, 1)
5	14	3+8=11	14	(1, 1, 0)

From the process above we get the following result: Optimum solution X = (1, 1, 0) with $\sum p = f = 14$. It means that we can turn on the Front Light and TV to get the optimal result. The process above can also be represented by the following pseudo code:

```
function DP(input W: integer, input
wt: array of integer, val: array of
integer, n: integer) → integer
{function DP returns the optimal value
of priority for the solution of power
management,
W is the budget,
wt is the cost for each electric
devices,
val is the priority value for each
electric devices,
n is the number of electric device}

  i : integer
  w : integer
  K : array of array of integer

  for i from 0 to n do
    for w from 0 to W do
      if w = 0 or i = 0 then
        K[i][w] ← 0 {basis}
      else if wt[i-1] <= w then
        K[i][w] ← max(val[i-1] + K[i-1]
+ K[i-1][w-wt[i-1]], K[i-1][w])
{rekurens}
      else
        K[i][w] ← K[i-w][w]

  → K[n][W];
```

Dynamic programming for power management problems takes O(n.w), with the detail as follows: O(n.w) times to fill C-table, which has (n+1).(w+1) entries, each requiring O(1) time to computer. O(n) time to trace solution, because tracing process starts in row n of the table and moves up 1 row at each step.

The example problem above is consist of only three electric devices. We can also add more devices to make the calculation more complicated. But, dynamic programming is only have complexity of O(n.w), it is a lot better than brute force that has complexity of O(2^n).

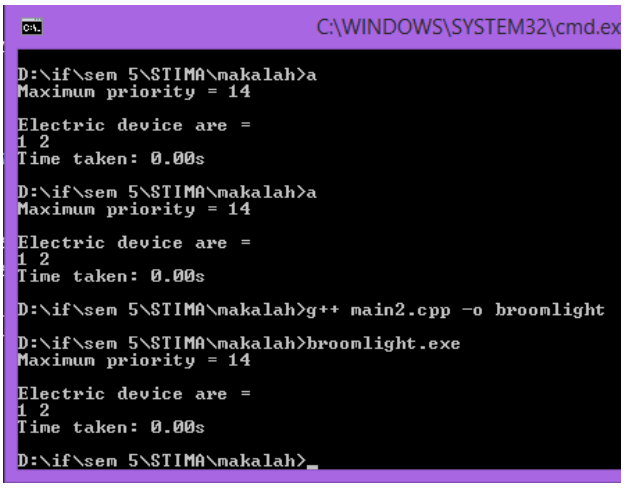

Figure 6. The result displayed on console

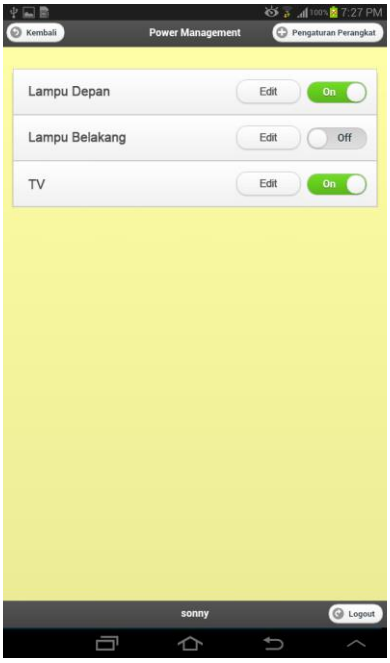

Figure 6. The result of power management suggestion

The Figure 6 is showing the application interface from an Android smartphone running on OS JellyBean. The code is originated from C++ to Javascript so that it can be run on the node.js server.

VIII. CONCLUSION

The result of the dynamic programming algorithm always give optimum solution for the problem. The complexity of dynamic programming for the power management problem is O(n.w). It is a lot of faster than brute force algorithm with the complexity of O(2^n).

Knapsack problem can be applied to the problem of power management because it has the similar characteristics.

IX. ACKNOWLEDGMENT

I want to say thanks for lecturer of Strategi Algoritma, Mrs. Masayu and Dr. Ir. Rinaldi for the patience to teach. I also thanked my parents for the support to me. Thanks to Kelvin because he give me inspiration to apply knapsack's dynamic programming into power management problem. Thank you also for Evan and Habibie. Evan for the help in making the hardware, and Habibie that always remember me and coding for socket.io and node together. I also want to thank to IDE team for their support to me.

REFERENCES

[1] Munir, Rinaldi, *Diktat Kuliah IF3051 Strategi Algoritma*, Bandung, Program Studi Teknik Informatika ITB, 2013.
[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms 3^{rd}*, 2Massachusetts: MIT Press, 2009.
[3] http://www.rapidtables.com/convert/electric/Amp_to_kW.htm, 19 Desember 2013
[4] http://listrikdirumah.com/cara-menghitung-biaya-listrik-perangkat-elektronik/, 19 Desember 2013
[5] http://michaelbluejay.com/electricity/cost.html#kilowatt, 19 Desember 2013
[6] http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Dynamic/knapsackdyn.htm, 20 Desember 2013

PERNYATAAN

I hereby declare that the paper is my own writing, not an adaptation, nor translation from another person‟s paper, and not a form of plagiarism.

Bandung, 18 Desember 2013

Sonny Lazuardi Hermawan
(13511029)