Acta Cybernetica **16** (2004) 427–441.

Notes on the properties of dynamic programming used in direct load control

Isto Aho*

Abstract

We analyze a dynamic programming (DP) solution for cutting overload in electricity consumption. We are able to considerably improve the earlier DP algorithms. Our improvements make the method practical so that it can be used more often or, alternatively, new state variables can be added into the state space to make the results more accurate.

1 Introduction

The shortage of electricity may cause a supplier to use direct load control (DLC); the supplier may turn off the electricity from some of its customers or may start generators to meet the demand. The goal is to minimize the losses by buying the minimum amount of electricity from other suppliers to cover the demand after DLC. A typical example of a control group is a residential appliance with electricity heaters or air conditioners. When the supplier controls the devices, a consumption peak called payback appears after the control period when the devices go back to their normal state [1, 17].

Different solution methods for DLC include, e.g., DP [1, 7, 8, 18], linear programming (LP) [12, 14], heuristics [1, 3], enumerative methods [8], and hybrids of LP and DP [13]. Objectives include load reduction minimization [7, 8, 18], peak load minimization [12, 13], production cost minimization [8, 18], and profit maximization [14]. Some of the decisions can be left to the customers [9]. DLC is often combined with unit commitment and economic dispatch, and the applied methods include, e.g., DP [4, 5, 6, 11] and evolutionary strategies [10]. See also [19].

Our method is related to that of [1, 7, 8, 14]. The present work improves the results of [1] by focusing on DP. In our model, (small) electricity suppliers group their customers based on the payback behavior: the payback shapes and other properties to be presented are for groups. Thus, the suppliers end up with a small number of controllable and prioritized groups. Our objective is between load reduction minimization and peak load minimization, and it differs from the objectives presented in the literature. We assume that the suppliers mainly buy

*Dept. of Computer and Information Sciences, University of Tampere, Tampere, Finland, Isto.Aho@uta.fi

the electricity they distribute. The above assumptions are realistic in Finland and in some other European countries with small suppliers.

Purchase transactions of electricity and own production give optimum level to be resold at each hour, while load over the optimum level has high price. If demand is higher than our predefined level, we want to cut (expensive) “over loads”. We also use purchasing and reselling prices (time-of-use rates). Our solution can use different objectives and, e.g., energy storages of [15, 16], without major modifications.

Sections 2 and 3 describe the model, optimization problem and the DP solution. The main results concern the “wait states” and “alternative states” (state variables) needed in DP. We build up a hierarchy of DP solutions so that it is possible to choose between fast and inaccurate and slow but more accurate methods. Section 4 shows how the number of wait states can be decreased to be about half of the number used in [1]. State space can also be decreased with the multi-pass DP of [18], but then one should be able to relax some constraints. Section 4 includes also the use a fourth state variable (alternative states). Tests are reported in Section 5.

2 Control, payback, restrictions and goal function

The model given below is a slight simplification of the model used in [1]. Table 1 contains relevant symbols used in this work.

An *interval* $[a, b]$ is the set $a, a+1, \ldots, b$ $(a < b)$ of integers. The *length* of an interval $[a, b]$ is $b - a + 1$. A *clipping situation* $\mathbf{s}$ is a vector $s_0, s_1 \ldots, s_N$ $(N > 0)$ of reals representing the difference between electricity demand, and electricity production and purchases in time interval $[0, N]$. The domain $[0, N]$ is called the *optimization interval* and values s_i are called either *overload* or *underload*. Overload represents a situation where the demand is higher than combined production and

Table 1: Used symbols.

Clipping situation	$\mathbf{s} = [s_0, s_1, \ldots, s_N]$	Set of controls	$\mathbf{C}$
Prices	$\mathbf{p} = [p_0, p_1, \ldots, p_N]$	Control capacity	C^c
Revenue	$\mathbf{r} = [r_0, r_1, \ldots, r_N]$	Control length	C^l
Length of hour	h^l	Resting time	C^r
Time interval or control	$[a, b]$	Minimum control length	C^m
Loss of incomes	$R(\mathbf{s})$	Maximum control length	C^M
Optimal control plan	$R^*(\mathbf{s})$	Maximum control times	C^T
Dynamic forward recursion	$R'(\mathbf{s}, S', k+1)$	Control time	C^t
Stage change	$R''(\mathbf{s}, S', S, k+1)$	Length of payback	P^l
Wait state	W	Amount of payback[a]	P^c
Alternative state	A	Impact of a control	$I([a, b], \mathbf{s})(k)$
State (3 variable)	$S = (C^t, W, C^l)$	Impacts of all controls	$I'(\mathbf{C}, \mathbf{s})$
State (4 variable)	$S = (C^t, A, W, C^l)$		

[a]Amount of payback corresponds to capacity explaining the c-superscript.

electricity purchases ($s_i \geq 0$), and underload represents a situation where combined production and purchases of electricity is above the level of consumption ($s_i \leq 0$). The element i of optimization interval $[0, N]$ is called a *time point*. The phrase "time point i" is also used for the interval $[i, i]$.

A time point i with overload has a positive real p_i called the *price* (buying price of electricity). Similarly, a time point i with underload has a positive real r_i, the *revenue* (selling price of electricity). The *overload interval* is an interval $[a, b] \subseteq [0, N]$ with overload at every time point $i \in [a, b]$. The clipping situation is partitioned into *hours* $0 = a_1, a_2, \ldots, a_{n+1} = N$, of equal length, i.e., $a_{i+1} - a_i = a_i - a_{i-1}$ $(2 \leq i \leq n)$. The length $a_{i+1} - a_i$ of an hour is denoted by h^l. Hour i refers to the interval $[a_i, a_{i+1} - 1]$. Overloads (underloads), revenues and prices do not change during an hour, because of the electricity trading system. Thus, we have $s_j = s_{j+1}$, $p_j = p_{j+1}$, and $r_j = r_{j+1}$, where $j \in [a_{i-1}, a_i - 1]$ $(2 \leq i \leq n)$.

The total loss is

$$R(\mathbf{s}) = \sum_{i \in [0,N]} K(i, s_i), \quad \text{where} \quad K(i, s_i) = \begin{cases} -p_i s_i, & \text{if } s_i \geq 0, \\ r_i s_i, & \text{otherwise.} \end{cases} \tag{1}$$

Note that $K(i, s_i) \leq 0$. If there is underload, we lose income (revenue) and if there is overload, we have to pay some extra. We count the money lost, so its best possible value is 0.

A *group* is used to decrease the overload with a *control* made for interval $[a, b]$ by turning electricity off (an auxiliary generator corresponds to a group). The *controlling capacity* of a group, denoted by C^c, is the amount the group can decrease the load in an hour. The hours $[a_i, a, a_{i+1}, \ldots, a_{j-1}, b, a_j]$, where $a_i \leq a < a_{i+1}$ and $a_{j-1} < b \leq a_j$ (and $a < b$), are affected by a control. *Control amount* is the product of the controlling capacity C^c and of the control length $b - a + 1$.

Function $P^l : \mathbb{N} \to \mathbb{N}$ maps the control length $b-a+1$ to the length of a payback and function $P^c : 2^{\mathbb{N}} \times \mathbb{N} \to \mathbb{R}$ describes the amount of the payback of control $[a, b]$ at time i. We always have $P^c([a, b], i) \geq 0$, where $i \in [b + 1, b + P^l(b - a + 1)]$, and otherwise $P^c([a, b], i) = 0$. Further, "in practice" we have

$$\sum_{k \in [b+1, b+P^l(b-a+1)]} P^c([a, b], k) \leq C^c(b - a + 1)$$

meaning that a payback does not exceed the control amount.

Next we show the impact of a control $[a, b]$ and its payback to clipping situation $\mathbf{s}$ as a function I (functions I_1 and I_2 used in I are defined below). The hours to be affected are $[a_i, a, b, a_j, b + P^l(b - a + 1), a_l]$. By function

$$I\big([a, b], \mathbf{s}\big)(k) = \begin{cases} s_k, & \text{if } 0 \leq k < a_i, \quad \text{or} \quad a_l \leq k \leq m, \\ s_k + I_1\big([a, b]\big)(k), & \text{if } a_i \leq k < a_{j-1}, \\ s_k + \big(I_1 + I_2\big)\big([a, b]\big)(k), & \text{if } a_{j-1} \leq k < a_j, \\ s_k + I_2\big([a, b]\big)(k), & \text{if } a_j \leq k < a_l \end{cases} \tag{2}$$

($0 \leq k \leq m$) we obtain the control amount of a control $[a,b]$ into the clipping situation. The first line leaves the unaffected hours as they are and the second line calculates the effects of control. The third line is for both the control and payback calculation, and the fourth line calculates the effects of payback. By $I([a,b],\mathbf{s})$ we mean the new clipping situation obtained after control $[a,b]$.

The effects of control part having no payback are calculated with I_1:

$$I_1\big([a,b]\big)(k) = \begin{cases} -C^c(a_{i+1}-1-a+1)/h^l, & \text{if } a_i \leq k < a_{i+1}, \\ -C^c, & \text{if } a_{i+1} \leq k < a_{j-1}, \\ -C^c(b-a_{j-1})/h^l, & \text{if } a_{j-1} \leq k < a_j. \end{cases} \tag{3}$$

The second line is for the hours between the starting and stopping hours, if any. The first and the third lines handle the hours where the control starts and stops. These hours may have partial control (in contrast to a full control lasting the whole hour). Controls decrease the overload. Paybacks are calculated with I_2:

$$I_2\big([a,b]\big)(k) = \begin{cases} \displaystyle\sum_{k'=b+1}^{a_j-1} \frac{P^c([a,b],k')}{h^l}, & \text{if } \ a_{j-1} \leq k < a_j, \\ \displaystyle\sum_{k''=a_{k'}}^{a_{k'+1}-1} \frac{P^c([a,b],k'')}{h^l}, & \text{if } \ j \leq k' \leq l-1 \\ & \text{and } \ a_{k'} \leq k < a_{k'+1}, \\ \displaystyle\sum_{k'=a_{l-1}}^{a_l-1} \frac{P^c([a,b],k')}{h^l}, & \text{if } \ a_{l-1} \leq k \\ & \qquad < b + P^l(b-a+1). \end{cases} \tag{4}$$

The payback starts in the first line, and in the third line we calculate the last moments having payback. (Both may involve partial hours.) The second line calculates the payback for hours, where each time point will get some payback. The payback increases the overload.

It would simplify formulas (2)–(4) a bit if we were not to hourly even out the effects. Another alternative is to let the overloads and underloads vary within the hours and even out the loads when calculating the results. If the control starts and stops in the same hour, we cannot directly apply (2). In this situation we calculate the effects for the first hour with

$$s_k - C^c(b-a+1)/h^l + \sum_{k'=b+1}^{a_j-1} P^c([a,b],k')/h^l, \text{ where } a_{j-1} \leq k < a_j, \tag{5}$$

and the rest of the payback is calculated with the second line of I_2. If the payback starts and stops in the same hour, we have to make a correction similar to (5). Energy storage capability is similar to payback: energy storing appears before control while payback appears after the control.

Figure 1 shows two examples of a control. The vertical lines indicate hours. The dotted line is a clipping situation without control and the straight line is a clipping situation with control. The left picture shows the advantage of a control: payback can “move” the overload to the next hour where the overload is cheaper.

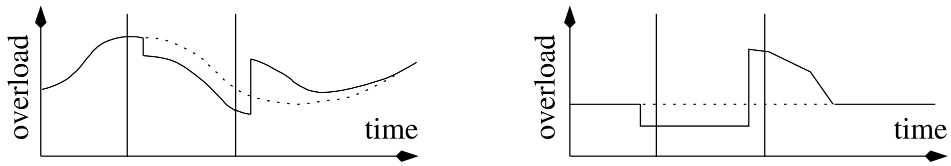

Figure 1: Realistic and theoretic clipping situation with a control.

The combined effect of all controls $\mathbf{C}$ can be calculated recursively by the function

$$I'(\mathbf{C},\mathbf{s}) = \begin{cases} I'\big(\mathbf{C} - [a,b], I([a,b],\mathbf{s})\big), & \text{if } [a,b] \in \mathbf{C} \ (\neq \emptyset), \\ \mathbf{s}, & \text{if } \mathbf{C} = \emptyset. \end{cases} \tag{6}$$

When all the effects of controls have been calculated, we can use (1) to find out the value of the new clipping situation.

Next we look at the restrictions. First, the controls must be separate such that for all $[a,b], [c,d] \in \mathbf{C}$ we have

$$[a,b] \neq [c,d] \Rightarrow [a,b] \cap [c,d] = \emptyset. \tag{7}$$

Further, during *resting time* it is not allowed to start a new control. Function $C^r : \mathbb{N} \to \mathbb{N}$ is increasing and it maps the length of a control to the length of a resting time. So, for all $[a,b], [c,d] \in \mathbf{C}$ we have

$$[a,b] \cap [c,d] = \emptyset \Rightarrow [b+1, b + C^r(b-a+1)] \cap [c,d] = \emptyset. \tag{8}$$

Note that a new control can be started even if the payback still occurs, provided that the resting time does not overlap with the new control. Usually, the resting time is used to prevent a new control to start in the beginning of the payback, when the need for extra electricity is the largest. If we start a new control at the end of a payback, the change in the payback of the new control is so small that it is not usually taken into account.

We also need the *minimum* and *maximum* control times C^m and C^M, respectively, which bound the length of control as

$$C^m \leq b - a + 1 \leq C^M. \tag{9}$$

Sometimes we also restrict the number of control times $\sum_{[a,b]\in\mathbf{C}} 1$ by C^T, a positive integer.

We can suppose that at every time point k the price factor p_k is (much) larger than the revenue r_k. By making controls we can affect the clipping situation, so the optimization problem can be given in the form

$$\max_{\mathbf{C}} \sum_{k=0}^{N} R(k, I(\mathbf{C},\mathbf{s})) \tag{10}$$

with restrictions (7)–(9).

3 A DP solution

The problem (10) can be solved with DP by using two state variables corresponding to the control times and control length [1, 8, 7, 19]. The tests in [1] indicated that it is possible to add at least one state variable to improve the results. In this work we use the state variables *control time* C^t, *wait state* W and *control length* C^l. As a result, we have a slower but more accurate system than those with two state variables. The state variables are defined in finite integer domain.

In the state space we need the control length C^l, so that DP can form the optimal control length and at the same time fulfill the restriction (9). In addition to control length, C^l also contains the resting time. Without the control times, DP complying with conditions (7)–(9) would find only one control. With these three state variables we have one state of stage $k \in [0, N]$ as a triple $(C^t, W, C^l)(k)$. The phrase "stage k" refers to a time point. A system in state (C^t, W, C^l) is defined to be the C^tth control of length C^l with W time points delay before its start. Our tests demonstrate that the three variable solution does not give the optimal solution, if the the group has payback (see Section 5).

In practice we have to determine upper bound for the number of wait states W. Theorem 2 gives an upper bound for W when the group does not have payback. If the group does have payback, we assume that W can have $h^l - 1$ (h^l is the length of an hour) different values. We also test other possibilities, see Section 5.

We denote $S = (C^t, W, C^l)$ and $S' = ({C'}^t, W', {C'}^l)$. The variables with primes are "new" ones and the plain variables are "old", when we form the connections between the "new" stage $k+1$ and the "old" stage k. Function

$$R''(\mathbf{s}, S', S, k+1) = \begin{cases} 0, & \text{when (14)–(17)}, \\ -P', & \text{when (18)}, \\ R(I([k - C^t, k], \mathbf{s})) - R(\mathbf{s}), & \text{when (19)}, \\ -\infty, & \text{otherwise}, \end{cases} \tag{11}$$

gives the change in the value, when moving from state (C^t, W, C^l) of stage k into state $({C'}^t, W', {C'}^l)$ of stage $k+1$. The first line is used when the value does not change. The second line is used, when we start a new control and the third line is applied, when we make a decision about the best control. The cost of making a control is denoted by P'. The last line is used with every other values of the variables S and S'. They are impossible since they do not have any reasonable real world interpretation.

The dynamic forward recursion equation is

$$R'(\mathbf{s}, S', k+1) = \max_S \big(R'(\mathbf{s}, S, k) + R''(\mathbf{s}, S', S, k+1)\big) \tag{12}$$

and

$$R'(\mathbf{s}, (C^t, W, C^l), 0) = \begin{cases} R(\mathbf{s}), & \text{when } C^t = W = C^l = 0, \\ -\infty, & \text{otherwise.} \end{cases} \tag{13}$$

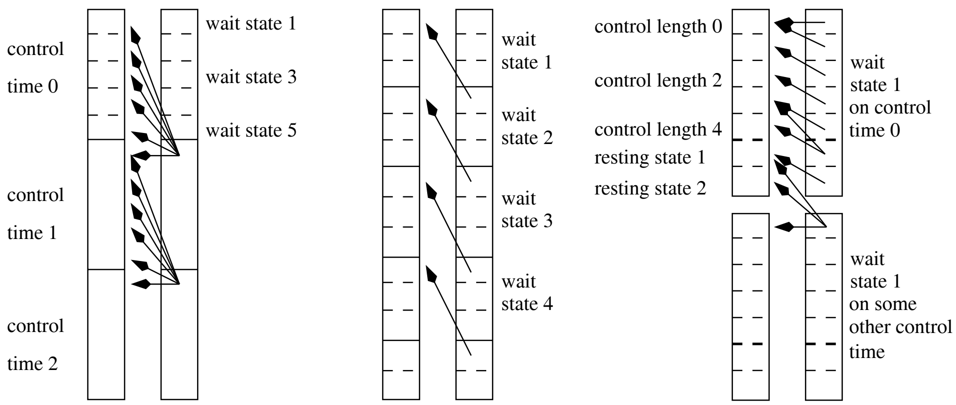

Figure 2: The structure of the state space. Alternative states are similar to control times, except that they do not choose the best control plan available.

When

$$\begin{aligned} &C'^t = C^t + 1, \quad W' = C'^l = 0, \quad 0 \le W \le h^l - 2, \\ \text{and} \quad &C^r(C^m) + C^M - 1 \le C^l \le C^r(C^M) + C^M - 1, \end{aligned} \tag{14}$$

the control at stage k and in state (C^t, W, C^l) has stopped, which in turn increases the amount of control times by one ($C'^t = C^t + 1$). The next control starts in state $(C'^t, 0, 0)$ at stage $k+1$. We add $C^M + 1$ both for minimum and maximum resting times, because C^l indexes the control length and the resting time (see Figure 2), and because the states corresponding to the resting time are located next to the control lengths (i.e., after state $C^M - 1$). Note, that we restrict the number of wait states by $h^l - 2$ (wait states can get $h^l - 1$ different values).

Moreover, it is possible that "old" optimal plan at stage k does not change (or be better) when moving into stage $k+1$, and so

$$C'^t = C^t, \quad W' = C'^l = 0, \quad \text{and} \quad W = C^l = 0. \tag{15}$$

This is the only case with conditions (14) and (19), where DP (recursion formula (12)) can make decisions about the path. If two paths give the same result, DP (12) chooses the one with a later control. This does not have any impact on the result, but in practice we usually want to do the controls as late as possible.

Figure 2 shows the state structure. Conditions (14) and (15) are shown on the left. There we have several states, from which we choose the maximum. When

$$C'^t = C^t, \quad W' = W + 1, \quad \text{and} \quad C'^l = C^l = 0, \tag{16}$$

we "move some information from the past" to new stage $k+1$. With this information we can check what result can be achieved, if we choose the best path W stages ago instead of some other control plan with the last control started in the interval

$[k-W,k]$. Figure 2 shows this in the middle, where a control time is depicted. When

$$C'^{t} = C^{t}, \quad W' = W, \quad C'^{l} = C^{l} + 1, \quad \text{and} \quad (C'^{l} \neq 1,\ C'^{l} \neq C^{M} + 1), \tag{17}$$

we increase the control length (the control started C^l time points ago). When

$$C'^{t} = C^{t} \quad W' = W, \quad \text{and} \quad 1 = C'^{l} = C^{l} + 1, \tag{18}$$

a new control starts. In this situation we add the cost of control P' to the result. Finally, when

$$C'^{t} = C^{t} \quad W' = W, \quad C'^{l} = C^{M} + 1, \quad \text{and} \quad C^{m} \leq C^{l} \leq C^{M}, \tag{19}$$

we can calculate the impact of a legal control on a clipping situation. Figure 2 shows conditions (17)–(19) in the right.

For each state (C^t, W, C^l) and for each stage $k > 0$, we save the connection pointing to some state of the previous stage. The connections form a *path*. When we have the values

$$R'(\mathbf{s}, (C^t, W, C^l), N)$$

with appropriate values in C^t, W and C^l, we can form the control plan $\mathbf{C}$ by traversing the path formed by the connections. The path is optimal with respect to the state space used (but not with respect to the problem). Note that functions R' and R'' in equations (11)–(13) comply with the conditions (7)–(9).

4 The properties of the state space

Next we study the properties of the dynamic recursion formula (12)–(13). Consider stage i. A *local* control for state $(C^t+1, 0, 0)$ is a control formed at control time C^t+1, stopped at stage $j > i + C^r(C^m)$, and using the control plan formed at stage i for state $(C^t, 0, 0)$. Stages $k > i$ do not belong to the local control, provided that we do not use the control plan of state $(C^t, 0, 0)(i)$ at stage k. This means that the wait states are not considered when forming a local control.

In the next theorem we suppose that all references to wait states have been omitted from the conditions (14), (15), (18) and (19).

Theorem 1. *State space (C^t, C^l) finds, for each stage i, the best local control following stage i.*

Proof. Consider the controls starting after stage i from state $(C^t, 0)$ and using control plan $\mathbf{C}$ determined by i and $(C^t, 0)$. The conditions (14) and (15) determine the best control for state $(C^t+1, 0)(j)$, according to the equation (12). The condition (14) gives the maximum because of the conditions (18) and (19). □

Corollary 1. *State $(1,0,0)(N)$ gives the best control plan having one control.*

Note that the length and the amount of payback do not have any consequences in the case of Theorem 1. State space (C^t, C^l) gives sub-optimal results [1], which can be improved with wait states (still being sub-optimal).

Let $\mathbf{C}_i, \mathbf{C}_{i+1}, \ldots, \mathbf{C}_a$ be the control plans of the control time C^t and the stages (time points) $i, i+1, \ldots, a$, respectively, i being the first time point of an hour. In the next theorem we show that it is enough to choose between the control plans $\mathbf{C}_i, \mathbf{C}_{i+1}, \ldots, \mathbf{C}_a$, when forming a control $[a, b]$. This refers to the situation in condition (14) of DP (12), where we check, how well the control plans of states $(C^t, 0, 0)(i), (C^t, 0, 0)(i+1), \ldots, (C^t, 0, 0)(a)$ work with the control starting at time point a.

Intuitively, the next theorem is based on the property that if the last control of some control plan stops with resting time in the "previous hour", it will not have any impact on the controls in the "present hour".

Theorem 2. *Suppose there is no payback. Suppose further that we start a new control from stage a for control time C^t, which will stop at stage b locally maximizing the control plan $\mathbf{C}_a$. Let i be the first moment of the hour containing a. Now it is enough to choose (with the wait states) from the set of control plans $\mathbf{C}_i, \mathbf{C}_{i+1}, \ldots, \mathbf{C}_a$, when forming a new control $[a, b]$.*

Proof. We show that it is not necessary to reach time points earlier than the start of the present hour. Consider situations where it is possible to choose between control plan $\mathbf{C}_{i-n}$ of time point $i-n$ $(n \geq 1)$, and control plans $\mathbf{C}_i, \mathbf{C}_{i+1}, \ldots, \mathbf{C}_a$, when forming $[a, b]$. Since DP (12) chooses always the maximum, the result of $\mathbf{C}_j$ does not decrease when j increases. Thus, the result of $\mathbf{C}_i$ is at least that of $\mathbf{C}_{i-n}$. If we choose some of the control plans $\mathbf{C}_{i-n}, \ldots, \mathbf{C}_{i-1}$ to be used with a control that starts at a, we obtain at least as good result with the control plan $\mathbf{C}_i$. $\square$

Note that the absence of payback and the fact that the resting time is coded into the state space are crucial here. It follows from Theorem 2 that we need one wait state at the first time point of an hour, two at the second time point and finally $h^l - 1$ at the last time point of an hour (h^l is the length of an hour). In other words, we need on the average $(h^l - 1)/2$ wait states at each time point. (In [1] we used $h^l - 1$ wait states at each time point.)

In general, the state space (C^t, W, C^l) does not achieve the optimal result when the length of payback is non-zero (a sample case is given in Section 5). We need at least one more state variable to be able to form a better path [2, pp. 30–34]. With variable A we check the non-maximum paths according to (12) for the three state variable system.

A *local alternative* of stage i is a control which stops at the stage i including the resting time and which is not chosen into the control plan. A three variable system chooses the best alternative among several, as shown in the left side of Figure 2. We set this to be the alternative state one. In the alternative state two, we choose the second best path from the control time C^t for the first state of control time $C^t + 1$. The third alternative state uses the third best path found so far and so on.

Alternative states can be easily implemented. When looking for the best path, we can cater the required amount of paths to find the Ath path. Moreover, the solution (a path) given by the condition (15) has to be checked when we are looking for the number of the best paths. The starting configuration (13) and the transition conditions (14)–(19) work with alternative states without major modifications. The initial solution is calculated only for the first alternative state and the conditions (14)–(19) work inside an alternative state just as in the case of the three variable system.

5 Tests

We first checked that Theorem 2 gives twice as fast method as the old DP. After that, we wanted to see whether paybacks affect the solutions given by the new three state variable DP. When we saw that paybacks do affect the results, we tried to improve the accuracy by increasing the number of wait states.

Intuitively, the wait states starting at point a can only "look up" that far (to point a) later on at the moment b. So if a is the beginning of an hour, we "do not see" into the previous hour at moment b and cannot make a choice between earlier control plans. When there is no payback, the number of wait states equaling to the number of time points after a start of an hour is enough, because a control finished in the previous hour does not affect the present hour to be cut. However, when we are using payback, we need to be able to look further into the previous hours in order to increase the accuracy. By adding c wait states, we directly reach earlier moments. We are tempted to think that increasing c will improve the solution. Our tests, however, show that while this is mostly true, there are exceptions.

In the tests we used a group shown in the upper left corner in Figure 3. We tested the group with 5 different clipping situations. Tests 1–4 could be clipping situations occurring in reality. They are similarly shaped and contain "morning and afternoon" consumption peaks. The shapes are at different levels giving tests

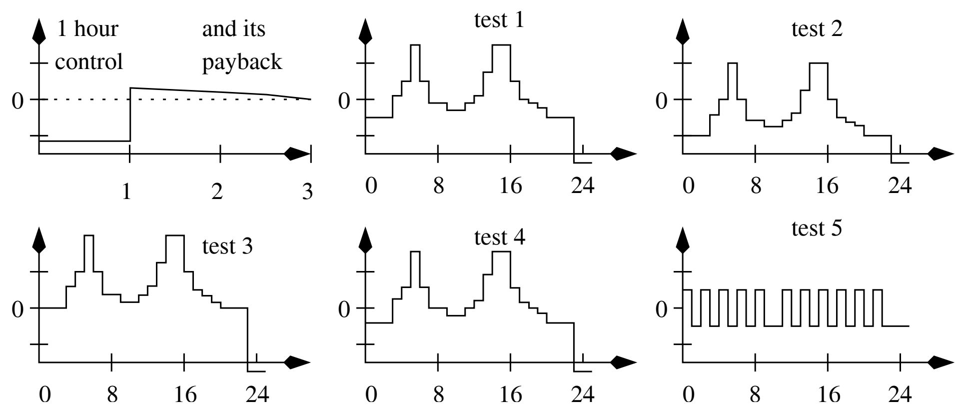

Figure 3: Payback used in the tests and the test cases.

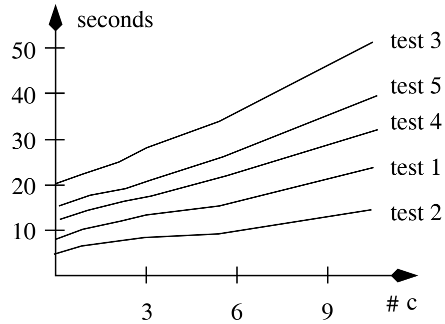

Figure 4: Running times.

of different difficulty. Test 5 is artificial. When load curve is above 0, we have overload that should be cut off. One tick stands for 1 MW. (Horizontal axis are for time.)

Each hour is discretized into twelve time points. The group had 1.2 MW control capacity (i.e., 0.1 for five minutes). The payback is two hours long, the minimum control length is 30 minutes and the maximum control length is one hour. One MW overload costs −99 000 and underload −900. Resting time is 10 minutes.

Figure 4 contains running times (in seconds) for the old DP solution and for DPs with $c = 0, 1, 2, 3, 5$ and 11 (horizontal axis). Moreover, $h^l = 12$. We see that the running times increase almost linearly on the number of waits states. Table 2 contains the results for the tests used in Figure 3. Here we see that, in general, the results improve if the number of wait states is increased.

In the second test, however, we see that increasing the number of wait states has decreased the result between DPs with $c = 3$ and $c = 5$ (a locally better solution is worse). This somewhat non-intuitive result follows from the fact that our DP solutions do not fulfill the optimality principle usually stated for dynamic programming solutions [2, p. 16]. Reason for this is that we cannot guarantee that optimal solution at stage i entails optimal solution for the rest of the case. Payback may affect later hours and decisions. This information should be available at the moment when we are deciding the length of a control.

We did not use any alternative states in the test series reported in Table 2 and

Table 2: Solutions without alternative states.

Test	old DP	$c = 0$	$c = 1$	$c = 3$	$c = 5$	$c = 11$
1.	−144 478	−144 478	−144 478	−144 478	−144 478	−144 478
2.	−15 603	−15 662	−15 662	**−15 662**	**−15 778**	−15 603
3.	−368 665	−371 296	−370 238	−368 665	−368 665	−368 665
4.	−174 490	−175 668	−175 668	−175 668	−175 668	−174 490
5.	−9 860	−9 860	−9 860	−9 860	−9 860	−9 860

Table 3: The best solutions with alternative states.

Test	old DP	$c = 0$	$c = 1$	$c = 3$	$c = 5$	$c = 11$
1.	**−144 478**	−144 478	−144 478	−144 478	−144 478	−144 478
2.	−15 603	−15 662	−15 662	**−15 487**	**−15 487**	**−15 487**
3.	−368 373	−369 069	**−368 046**	−368 162	−368 278	−368 373
4.	−174 490	−175 668	−175 543	**−174 398**	**−174 398**	−174 490
5.	−8 956	−8 336	−8 351	−8 186	−7 742	**−7 538**

Figure 4. We run the same test series using 2, 5, 10 and 15 alternative states. Table 3 contains the best solutions found among all the test series. Results were improved in most of the cases, sometimes over 10%.

Test 5 informatively demonstrates how results improve as the number of wait states or alternative states (or both) increases. The results and running times are shown in Figures 5 and 6. We also tried DPs with 30 and 100 alternative states. DP with 15 alternative states gives $-8\,336$, with 30 states $-7\,872$, and with 100 states $-7\,214$, which is better than the solution given by 15 alternative and 11 additional wait states (see Table 3). Our conclusion is that a clipping situation with many overload intervals most likely benefits from the use of alternative states.

We also studied how alternative states and wait states together improve the results and how they affect the running times. The left hand side of Figure 5 contains the results for different alternative state amounts (1, 2, 5, 10 and 15). As the number of wait states is increased, the results improve in general. There are exceptions where few wait states do worse than DP with $c = 0$ (see lines for A5 and A15). The number of additional wait states used is irrelevant for this instance, when we used only one alternative state.

On the right side the same data is plotted for five different additional wait state amounts as well as for the old DP system. We conclude that the number of alternative states is much more crucial for the results than the number of wait states. Alternative states also improve the results of DP system with fixed amount

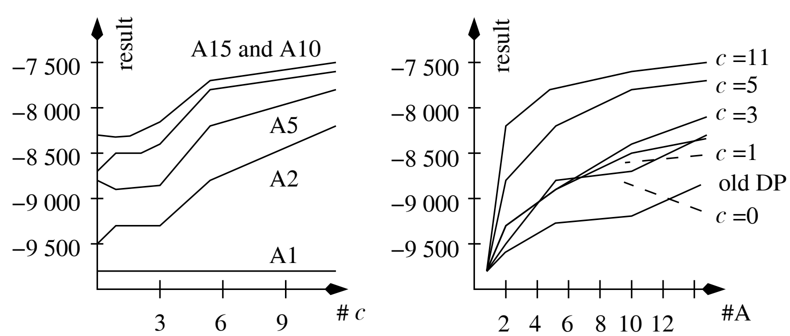

Figure 5: Wait and alternative states, results for test 5.

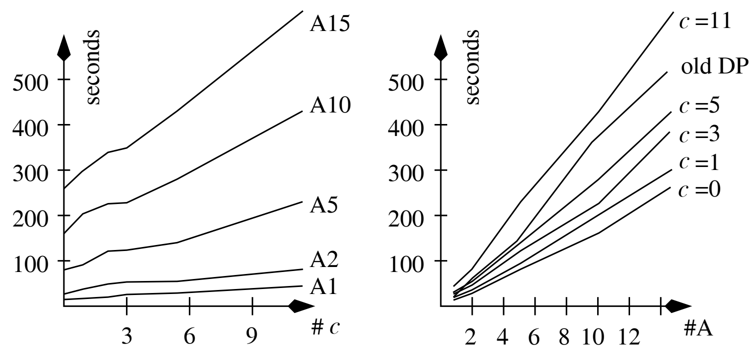

Figure 6: Wait and alternative states, running times for test 5.

of wait states (in old DP $c = h^l - 1$).

Figure 6 contains the running times for test 5. Execution decelerates almost linearly as the number of alternative or wait states is increased. The number of wait states in one hour is $\sum_{a=0}^{h^l-1}\left(c + (a \mod h^l)\right)$ (where $c = 0, \ldots, h^l - 1$ and a is a moment). Hence, the increase of c by one gives $h^l - 1$ additional wait states for one hour. This is proportionally less than the increase brought by the increase of the number of alternative states by one, which is the number of used wait states in one hour. This explains why the increase of the number of alternative states decelerates more the running times than the increase of the number of wait states.

6 Conclusions

In this work we have analyzed the properties of the state space of a dynamic programming problem arising in direct load control, and quicker optimization algorithms are formed without sacrificing the accuracy of the results when payback is not used. Moreover, we have found practical ways to improve the results by increasing the state space when payback is used. We have described (sub-optimal) solutions for three and four state variables. If the result accuracy is not crucial, one can drop wait states away, arriving to a faster two state variable solution of [8].

There are still open problems concerning the properties of state variable C^t. They seem to behave in a way that enables us to reduce the number of states used (for details, see [1]). Moreover, we conjecture that the control length C^l has properties, by which we can further speed up the algorithms.

If there is enough time, it is possible to add a new state variable, called alternative state. With four state variables we achieve even better results, as is shown in our tests. Most of the time, additional wait states as well as additional alternative states improve the results. Hence, one can choose between fast inaccurate, and accurate but slow solutions. Similar trade can be made between two, three and four variable state spaces. The alternative states seem to improve the results also in the cases occurring in production systems.

Acknowledgment

The author is grateful to Erkki Mäkinen for his time and comments and to an anonymous referee whose valuable comments improved the quality of this work.

References

[1] Isto Aho, Harri Klapuri, Jukka Saarinen, and Erkki Mäkinen. Optimal load clipping with time of use rates. *International Journal of Electrical Power & Energy Systems*, 20(4):269–280, May 1998.

[2] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 1996.

[3] R. Bhatnagar, J. Latimer, L.J. Hamant, A.A. Garcia, G. Gregg, and E. Chan. On-line load control dispatch at Florida Power & Light. *IEEE Transactions on Power Systems*, 3(3):1237–1243, August 1988.

[4] R. Bhatnagar and S. Rahman. Dispatch of direct load control for fuel cost minimization. *IEEE Transactions on Power Systems*, PWRS-1(4):96–102, November 1986.

[5] K. Bhattacharyya and M.L. Crow. A fuzzy logic based approach to direct load control. *IEEE Transactions on Power Systems*, 11(2):708–714, May 1996.

[6] Douglas W. Caves and Joseph A. Herriges. Optimal dispatch of interruptible and curtailable service options. *Operations Research*, 40(1):104–112, January–February 1992.

[7] Wen-Chen Chu, Bin-Kwie Chen, and Chun-Kuei Fu. Scheduling of direct load control to minimize load reduction for a utility suffering from generation shortage. *IEEE Transactions on Power Systems*, 8(4):1525–1530, November 1993.

[8] Arthur I. Cohen and Connie C. Wang. An optimization method for load management scheduling. *IEEE Transactions on Power Systems*, 3(2):612–618, May 1988.

[9] I.M. El-Amin, A.R. Al-Ali, and M.A. Suhail. Direct load control using a programmable logic controller. *Electric Power Systems Research*, 52(3):211–216, 1999.

[10] A.J. Gaul, E. Handschin, W. Hoffmann, and C. Lehmköster. Establishing a rule base for a hybrid es/xps approach to load management. *IEEE Transactions on Power Systems*, 13(1):86–93, February 1998.

[11] Yuan-Yih Hsu and Chung-Ching Su. Dispatch of direct load control using dynamic programming. *IEEE Transactions on Power Systems*, 6(3):1056–1061, August 1991.

[12] C.N. Kurucz, D. Brandt, and S. Sim. A linear programming model for reducing system peak through customer load control programs. *IEEE Transactions on Power Systems*, 11(4):1817–1824, November 1996.

[13] Jean-Charles Laurent, Guy Desaulniers, Roland P. Malhamé, and François Soumis. A column generation method for optimal load management via control of electric water heaters. *IEEE Transactions on Power Systems*, 10(3):1389–1400, August 1995.

[14] Kah-Hoe Ng and Gerald B. Sheblé. Direct load control — a profit-based load management using linear programming. *IEEE Transactions on Power Systems*, 13(2):688–695, May 1998.

[15] B. Rautenbach and I.E. Lane. The multi-objective controller: a novel approach to domestic hot water load control. *IEEE Transactions on Power Systems*, 11(4):1832–1837, November 1996.

[16] Paresh Rupanagunta, Martin L. Baughman, and Jerold W. Jones. Scheduling of cool storage using non-linear programming techniques. *IEEE Transactions on Power Systems*, 10(3):1279–1285, August 1995.

[17] Katsuyuki Tomiyama, John P. Daniel, and Satoru Ihara. Modeling air conditioner load for power system studies. *IEEE Transactions on Power Systems*, 13(2):414–421, May 1998.

[18] Deh-chang Wei and Nanming Chen. Air conditioner direct load control by multi-pass dynamic programming. *IEEE Transactions on Power Systems*, 10(1):307–313, February 1995.

[19] Allen J. Wood and Bruce F. Wollenberg. *Power Generation, Operation and Control*. John Wiley & Sons, 1984.

Received November, 2002