# An Improved Algorithm for Single-Unit Commitment with Ramping Limits

Rogier Hans Wuijts<sup>\*†</sup>, Marjan van den Akker<sup>\*</sup> and Machteld van den Broek<sup>†</sup> \* Department of Information and Computing Sciences Utrecht University, The Netherlands {R.H.Wuijts, J.M.vandenAkker}@uu.nl <sup>†</sup> Copernicus Institute for Sustainable Development Utrecht University, The Netherlands M.A.vandenBroek@uu.nl

*Abstract*—The single-unit commitment problem (1UC) is the problem of finding a cost optimal schedule for a single generator given a time series of electricity prices subject to generation limits, minimum up- and downtime and ramping limits. In this paper we present two efficient dynamic programming algorithms. For each time step we keep track of a set of functions that represent the cost of optimal schedules until that time step. We show that we can combine a subset of these functions by only considering their minimum. We can construct this minimum either implicitly or explicitly.

Experiments show both methods scale linear in the amount of time steps and result in a significant speedup compared to the state-of-the-art for piece-wise linear as well as quadratic generation cost. Therefore using these methods could lead to significant improvements for solving large scale unit commitment problems with Lagrangian relaxation or related methods that use 1UC as subproblem.

*Index Terms*—Dynamic programming, single-unit commitment problem, polynomial-time algorithm.

## NOMENCLATURE

f ()	The cost of anothering a st time t (\$)
$J_t(p_t)$	The cost of producing $p_t$ at time $t \left(\frac{1}{MW}\right)$
$\Delta^+$	Ramp-up limit $\left(\frac{MW}{h}\right)$
$\Delta^{-}$	Ramp-down limit $\left(\frac{MW}{h}\right)$
$\overline{P}$	Maximum generation (MW)
$\underline{P}$	Minimum generation (MW)
$c_{start}$	Time independent start cost (\$)
$c_{stop}$	Stop cost (\$)
$M_{\rm down}$	Minimum down time (h)
$M_{\rm up}$	Minimum up time (h)
$S\hat{D}$	Shut-down ramp limit (MW)
SU	Start-up ramp limit (MW)
	I. INTRODUCTION

The unit commitment (UC) problem revolves around finding the least cost power generation schedule for a set of generators such that the demand is met at each time step subject to technical restrictions [1].

The single-unit commitment problem (1UC) is a special case of the UC problem in which the least cost schedule is searched for only one generator subject to its technical restrictions [2]. In this case, the generator is not required to meet a demand, but a time series of electricity prices is

given that determines how much revenue the generator can make at each time step. In this paper we will study 1UC with generation limits, minimum up- and downtime and ramping limits.

The relevance of 1UC lies in the fact that it arises as a subproblem in the Lagrangian relaxation or column generation, which have been shown to be competitive for UC [1]. For the efficiency of these algorithms, the efficiency of solving 1UC is crucial.

A solution to 1UC is a schedule that for every time step specifies whether the generator is on or off and how much power it is producing. The cost associated with power production is the generation cost and consists of the cost of operating the generator minus the revenue. The generation cost is assumed to be convex and in the often modeled as a linear, piece-wise linear or quadratic function. When 1UC is used as a subproblem in the Lagrangian relaxation this revenue corresponds to the Lagrangian multipliers.

Fan, Xiaohong Guan, and Zhai [4] solved 1UC with piecewise linear generation cost in  $O(n^3)$  time by splitting the problem in two parts. The first part of the algorithm is to define *on-periods*. A *on-period* is a subsequence of time steps of where the generator is on and must be larger or equal to the minimum up time. Every *on-period* has an optimal power output subsequence. Finding the optimal power output subsequence for each *on-period* is an optimization problem. Fan et al. solved this with a dynamic programming algorithm that recursively partitions the continuous state space by finding corner points of the cost-to-go function. In the second part, these *on-periods* weighted by the optimal economic dispatch costs, are combined in an optimal schedule by solving a shortest path problem.

Frangioni and Gentile [2] improved this method by making it more general. Their method of calculating the optimal economic dispatch works for any convex cost function. Their algorithm calculates the optimal economic dispatch of all onperiods in  $O(n^3)$  and finds the shortest path in another  $O(n^3)$ . Guan et al. [3] found that a redefinition of the commitment state space graph can speed up this second part to  $O(n^2)$ .

In 2010, Zhai, Xiaohong Guan and Gao [5] presented a similar algorithm in which IUC is also split into two parts,



but which has the advantage that it also works for non-convex piece-wise linear cost functions.

In 2018 Yongpei Guan, Pan, and Zhou [3] introduced an algorithm to solve 1UC in O(n) time. However, as a restriction it only works with convex piece-wise linear generation cost and when the ramp up and ramp down limits are equal to each other. They solved 1UC by keeping track of a finite number of points where the power production could be optimal.

#### II. Outlook

In this paper we present two algorithms for solving 1UC that improve upon previous algorithms in terms of generality, time complexity and computation time in practice. Both algorithms are based on two equivalent recurrence relations.

The idea for both relations is as follows: for each time step we define all states the generator can attain and all valid state transitions. After that we define the recurrence relation that represents for each state at each time step the value of the optimal schedule that ends in that state. However, since the amount of states is infinite as the power output is continuous we cannot solve these recurrence relations directly. Therefore, 1UC is reformulated in terms of recurrence relations on functions, later defined as  $F_t^{\tau}$  and  $H_t^{\tau}$ . These functions can be constructed by partitioning their domain as was previously done to calculate the optimal economic dispatch for onperiods<sup>1</sup>. However, instead of computing the optimal economic dispatch of an on-period beforehand  $F_t^{\tau}$  and  $H_t^{\tau}$  represent the cost of the complete 1UC schedule from time 1 up until time t.

The first dynamic programming algorithm we defined, enables the identification of superfluous functions  $F_t^{\tau}$  that are always dominated by other functions i.e. for every power output level  $p_t$  there is another function with a lower cost. If we remove superfluous functions the overall computation time can be reduced. Identification of these functions comes at a cost and has a worst case time complexity of  $O(n^3)$ . However, in practice it has little computational overhead. Moreover, we describe the conditions in which this algorithm grows linear with the amount of time steps and show these conditions are met for the instances we studied. This results in a speedup of the algorithm of Frangioni and Gentile [2].

The second dynamic programming algorithm we defined, combines subsets of on-states and thus reduces the total amount of states. At the same time, the number of functions the recurrence relation consists of diminishes, as multiple functions are replaced by their combined minimum. When  $f_t$  is piece-wise linear we can create this function by only keeping track of a finite number of points similar to the algorithm of Guan et al. [3]. However, in contrast to the algorithm of Guan et al. our algorithm, is more generic as it also works for ramp up and ramp down rates which differ from each other. Moreover, we show how to efficiently compute the values at those points. This results in an improved algorithm in terms of generality and time complexity.

In section III we formally define the 1UC problem. In section IV and section V we will state the recurrence relation that solves 1UC and section VI we define and analyse a dynamic programming algorithm, RRF+, that uses this recurrence relation. In section VII we show an equivalent recurrence relation and in section VIII we define and analyse a dynamic programming algorithm, RRH, that uses this recurrence relation when the generation cost function is piece-wise linear. At last we conclude this paper in section IX with experiments regarding our two algorithms for both piece-wise linear and quadratic generation cost and show that they significantly decrease the computation time compared to previous methods.

#### **III. PROBLEM DEFINITION**

A solution is represented as two vectors  $\mathbf{x} \in \{0,1\}^n$ and  $\mathbf{p} \in \mathbb{R}^n$ . Where  $\mathbf{x} = x_1 \dots x_n$  represents the binary commitment variables i.e.  $x_t$  is 1 when the generator is on at time t and 0 otherwise. The vector  $\mathbf{p} = p_1 \dots p_n$ represents the continuous power output variables that for each time step indicates how much power a generator provides at that time. The associated cost of a solution is determined by time dependent generation cost function  $f_t$ . Here  $f_t(p_t)$  is a function that returns the generation cost of providing  $p_t$  power at time t. If the generator is on then the production must be between the minimum generation  $\underline{P}$  and the maximum generation P. Moreover, if the generator is turned on(off), it must stay on(off) for at least  $M_{\rm up}(M_{\rm down})$  time steps. Finally, there are four ramping limits. There is a ramp-up  $(\Delta^+)$  and ramp-down  $(\Delta^-)$  constraint that limits how much two consecutive power output levels  $p_t$  and  $p_{t+1}$  can differ from each other. These limits only hold for two consecutive time steps where the generator is on. Special ramping limits apply when a generator starts or shuts down, the startup limit SU and shutdown limit SD. We assume the generator can be in any state at  $t=1^2$ . We can now define 1UC as the following mixed integer program:

$$\min c_{cycle}(\mathbf{x}) + \sum_{t \in \{1, \dots, n\}} f_t(p_t) \text{ subject to}$$
(10)

$$\underline{P}x_t \le p_t \le \overline{P}x_t, t = 1\dots n \tag{11}$$

$$p_{t+1} \le p_t + x_t \Delta^+ + (1 - x_t) SU, t = 1 \dots n - 1$$
 (12)

$$p_t \le p_{t+1} + x_{t+1}\Delta^- + (1 - x_{t+1})SD, t = 1 \dots n - 1$$
 (13)

$$\mathbf{x} \in X, p_t \in \mathbb{R}$$
 (14)

 $c_{cycle}(\mathbf{x})$  consists of the start-up  $c_{start}$  and shutdown cost  $c_{down}$  of the commitment vector  $\mathbf{x}$ . X is the set of feasible commitment vectors with respect to the minimum up- and downtime (14).

<sup>&</sup>lt;sup>1</sup>cost-to-go functions  $L^t$  in [4] and  $z_{hk}$  in [2]

<sup>&</sup>lt;sup>2</sup>Here we ignore the transition constraints and cost between t = 0 and t = 1. Alternatively, the state at t=0 can also be provided as input to the problem.

$$c(of f_t^{\tau}) = 0 \qquad t = 1$$

$$c(of f_t^{\tau}) = \min \qquad \min \qquad c(on_{\tau'+1}^{\tau'}, p_{t-1}) + c_{stern} \qquad t > 1$$

$$c(of f_t^{\tau}) = c(of f_{t-1}^{\tau-1}) \qquad t > 1$$

$$c(off_t^{\tau}) = \min\{c(off_{t-1}^{\tau-1}), c(off_{t-1}^{\tau})\} \qquad t > 1 \qquad \tau$$

$$c(on_t^{\tau}, p_t) = f_t(p_t)$$

$$c(on_t^{\tau}, p_t) = f_t(p_t) + c(off_{t-1}^{M_{\text{down}}}) + c_{start}$$

$$c(on_t^{\tau}, p_t) = f_t(p_t) + \min_{\substack{p_{t-1} \in [p_t - \Delta^+, p_t + \Delta^-]}} c(on_{t-1}^{\tau-1}, p_{t-1})$$

$$c(\_) = \infty$$

$$F_{t}^{\tau}(p_{t}) = \begin{cases} f_{t}(p_{t}) & t = 1, \underline{P} \leq p_{t} \leq \overline{P} \\ f_{t}(p_{t}) + c(off_{t-1}^{M_{down}}) + c_{start} & t > 1, \tau = 1, \underline{P} \leq p_{t} \leq SU \\ f_{t}(p_{t}) + \min_{p_{t-1} \in [p_{t} - \Delta^{+}, p_{t} + \Delta^{-}]} F_{t-1}^{\tau-1}(p_{t-1}) & t > 1, \tau > 1, \underline{P} \leq p_{t} \leq \overline{P} \\ \infty & \text{otherwise} \end{cases}$$
(9)

# **IV. FIRST RECURRENCE RELATION**

In this section we define the state space and the corresponding recurrence relation that solves it.

The state space represents the possible states of the generator for each time step. Every state at time t has a set of feasible transitions to states at time t + 1 which depends on ramping limits and minimum up and down time. It is therefore necessary to keep track of how long a generator has been on or off, until the minimum up- or downtime is reached, and to keep track of the power output  $p_t \in [P, \overline{P}]$  when a generator is on. The set of states that a generator can be in at time t is defined as:

$$S_t = \bigcup \begin{cases} \{ off_t^{\tau} \mid \tau \in \{1 \dots M_{\text{down}}\} \} \\ \{ (on_t^{\tau}, p_t) \mid \tau \in \{1 \dots \max(t, M_{\text{up}})\}, p_t \in [\underline{P}, \overline{P}] \} \end{cases}$$
(15)



Fig. 1. The commitment statespace and transitions of 7 time steps. The offstates are represented as white nodes and the on-states as black nodes.

See Figure 1 for the binary commitment variables statespace and transitions. The state  $of f_t^{\tau}$  represents the state at time

$$t = 1 \tag{1}$$

$$>1 \qquad \qquad \tau = 1 \tag{2}$$

$$t > 1 \qquad 1 < \tau < M_{\text{down}} \qquad (3)$$
  
$$t > 1 \qquad \tau = M_{\text{down}} \qquad (4)$$

1 
$$\underline{P} \le p_t \le \overline{P}$$
 (5)

> 1 
$$\tau = 1, \underline{P} \le p_t \le SU$$
 (6)  
> 1  $\tau > 1, \underline{P} \le p_t \le \overline{P}$  (7)

t =t

(8)

$$t = 1, \underline{P} \leq p_t \leq P$$
  

$$t > 1, \tau = 1, \underline{P} \leq p_t \leq SU$$
  

$$t > 1, \tau > 1, \underline{P} \leq p_t \leq \overline{P}$$
(9)

t where the generator is off for  $\tau$  time steps, but when  $\tau = M_{\rm down}$  it represents that the generator is off for at least  $M_{\rm down}$  time steps. The state  $(on_t^{\tau}, p_t)$  represents that at time t a generator is on for  $\tau$  time steps and produces  $p_t$  at time t.

We will now define the recurrence relation  $c(s_t)$  that for each state  $s_t$  returns the cost of the optimal 1UC schedule at time t that ends in  $s_t$  (1)-(8). For each state  $s_t$  this is defined as the cost of that state, the minimum of  $c(s_{t-1})$  over states  $s_{t-1}$  that can transition towards  $s_t$  and the transitions costs.

At t = 1 the generator can be in any state with only the cost of being in that state and no transition cost, (1) and (5).

For t > 1 we can formulate every possible state transition that respects ramping limits and minimum up- and downtime and define the recurrent part of the recurrence relation by (2)-(4) and (6)-(7).

There are three types of transitions to a state where the generator is off. First, we can get to the first off-state  $of f_t^1$ from an on-state  $(on_{t-1}^{\tau}, p_t)$  where the generator is on for at least  $M_{\rm up}$  time steps and produces less than the shutdown limit SD (2). Secondly we can get to the off-state  $of f_t^{\tau}$  that is off for  $\tau > 1$  time steps from an off-state that is off for  $\tau - 1$  time steps (3). Thirdly, If  $\tau = M_{\text{down}}$  we can also get to the last off-state  $off_t^{M_{\text{down}}}$  from  $off_{t-1}^{M_{\text{down}}}$  (4).

For the on-states it is almost similar. We can only get to the on-state  $(on_t^1, p_t)$ , where  $p_t$  is less than the startup limit SU, from an off-state  $off_{t-1}^{M_{\text{down}}}$  where the generator is off for at least  $M_{\text{down}}$  time steps (6). We can get to on-state  $(on_t^{\tau}, p_t)$ that is on for  $\tau > 1$  time steps from an on-state  $(on_{t-1}^{\tau-1}, p_{t-1})$ if the difference between  $p_t$  and  $p_{t-1}$  respects the ramp-up  $\Delta^+$  and ramp-down  $\Delta^-$  limit (7).

# V. Constructing $F_t^{\tau}$

Because the set of on-states is infinite, we cannot calculate the cost of every possible on-state,  $(on_t^{\tau}, p_t)$ , explicitly. Therefore we construct for every  $t \in \{1, \ldots, n\}, \tau \in$   $\{1, \ldots, \max(t, M_{up})\}\$  a function  $F_t^{\tau}$  such that  $F_t^{\tau}(p_t) = c(on_t^{\tau}, p_t)$ , the function is given by (9). Moreover  $F_t^{\tau}$  has a piece-wise nature and is convex<sup>3</sup>.

In order to solve 1UC we need iteratively determine  $c(of f_t^{\tau})$  and  $F_t^{\tau}$  for each t and  $\tau$ . Solving 1UC efficiently boils down to constructing  $F_t^{\tau}$  efficiently for all t and  $\tau$ . In this chapter we will present a method to construct  $F_t^{\tau}$  equivalent to Frangioni and Gentile [2] but in a declarative way which is in our opinion more intuitive.

Constructing the first  $F_t^1$  is easy since  $f_t$  is given as input and  $c(off_{t-1}^{M_{\text{down}}})$  is just a single value. The hard part of constructing  $F_t^{\tau}$  is taking the sliding window minimum of  $F_{t-1}^{\tau-1}$ . Suppose  $p_{t-1}^*$  is a point in  $[\underline{P}, \overline{P}]$  for which  $F_{t-1}^{\tau-1}$  is minimal:

$$p_{t-1}^* = \operatorname{argmin}_{p_{t-1} \in [\underline{P}, \overline{P}]} F_{t-1}^{\tau-1}(p_{t-1})$$

To find the value of  $\min_{p_{t-1} \in [p_t - \Delta^+, p_t + \Delta^-]} F_{t-1}^{\tau-1}(p_{t-1})$  for a given  $p_t$  three cases can be identified equivalent to those of Frangioni and Gentile [2]:

$$\min_{\substack{p_{t-1} \in [p_t - \Delta^+, p_t + \Delta^-] \\ F_{t-1}^{\tau-1}(p_t + \Delta^-) \\ F_{t-1}^{\tau-1}(p_{t-1}^*) \\ F_{t-1}^{\tau-1}(p_{t-1}^*) \\ F_{t-1}^{\tau-1}(p_t - \Delta^+) \\ F_{t-1}^{\tau-1}(p_t - \Delta^+) \\ p_t > p_{t-1}^* + \Delta^+ \\ F_{t-1}^{\tau-1}(p_t - \Delta^+) \\ F_{t-1}^{$$

In other words, if  $p_t$  can be reached from  $p_{t-1}^*$ within the ramping limits, i.e. with a subtraction smaller than  $\Delta^-$  or an addition smaller than  $\Delta^+$  then  $\min_{p_{t-1}\in[p_t-\Delta^+,p_t+\Delta^-]}F_{t-1}^{\tau-1}(p_{t-1}) = F_{t-1}^{\tau-1}(p_{t-1}^*)$ . Otherwise  $p_t$  is either smaller than  $p_{t-1}^* - \Delta^-$  or larger than  $p_{t-1}^* + \Delta^+$ . In the first case, since this function is convex, the largest feasible  $p_{t-1}$  is optimal. Which is given by  $p_{t-1} = p_t + \Delta^-$ . Similar for the second case the smallest feasible  $p_{t-1}$  is optimal. Which is given by  $p_{t-1} = p_t - \Delta^+$ . The idea is illustrated in Figure 2.

Suppose the function  $F_{t-1}^{\tau-1}$  consists of *m* piece-wise functions  $g_{t-1}^1 \dots g_{t-1}^m$  over *m* intervals:

$$[\underline{P}, p_{t-1}^1], [p_{t-1}^1, p_{t-1}^2], \dots, [p_{t-1}^{m-1}, \overline{P}]$$
(17)

$$g_{t-1}^{i}(p_{t-1}) = F_{t-1}^{\tau-1}(p_{t-1}) \qquad p_{t-1} \in [p_{t-1}^{i-1}, p_{t-1}^{i}]$$
(18)

Suppose  $p_{t-1}^*$  falls in interval k,  $p_{t-1}^* \in [p_{t-1}^{k-1}, p_{t-1}^k]$ . We now observe that  $p_t \mapsto \min_{p_{t-1} \in [p_t - \Delta^+, p_t + \Delta^-]} F_{t-1}^{r-1}(p_{t-1})$  consists of m+2 piece-wise functions  $g_t^1 \dots g_t^{m+2}$  which we can explicitly construct from the previous intervals. These intervals are given by:

$$[\underline{P}, p_t^1], \dots, [p_t^k, p_t^{k+1}], \dots, [p_t^{m+1}, \overline{P}]$$
(19)

$$p_{t}^{i} = \begin{cases} \max(\underline{P}, p_{t-1}^{i} - \Delta^{-}) & i < k \\ \max(\underline{P}, p_{t-1}^{*} - \Delta^{-}) & i = k \\ \min(\overline{P}, p_{t-1}^{*} + \Delta^{+}) & i = k+1 \\ \min(\overline{P}, p_{t-1}^{i-2} + \Delta^{+}) & i \ge k+2 \end{cases}$$
(20)

 $^{3}$ Proof is omitted for brevity but convexity is easy to see if you look at the epigraph of the new function created in (16).



Fig. 2. Visualising (16). Here the bottom axis represents how the domain of  $F_{t-1}^{\tau-1}$  (17) is partitioned and how its mapped to the domain of  $p_t \mapsto \min_{p_{t-1} \in [p_t - \Delta^+, p_t + \Delta^-]} F_{t-1}^{\tau-1}(p_{t-1})$  (19).

$$g_t^i(p) = \begin{cases} g_{t-1}^i(p + \Delta^-) & i < k \\ g_{t-1}^{i-1}(p_{t-1}^*) & i = k+1 \\ g_{t-1}^{i-2}(p - \Delta^+) & i \ge k+2 \end{cases}$$
(21)

Intervals that become  $[\underline{P}, \underline{P}]$  or  $[\overline{P}, \overline{P}]$  are redundant and can be removed. Moreover if the optimal point lies on the end point of an interval, say  $p_{t-1}^* = p_{t-1}^k$ , then the k+2st interval becomes a single point and can be removed. Also note that if  $g_{t-1}^1 \dots g_{t-1}^m$  are linear functions then the optimal point always lies on a breakpoint. Therefore if  $f_t$  is piece-wise linear only one extra interval is introduced.

## VI. ALGORITHM RRF(+)

Solving c with  $F_t^{\tau}$  is similar to the algorithm of Frangioni and Gentile [2]. The major difference is that instead of precalculating the optimal economic dispatch for on-periods we now inductively for each time step create a set of functions  $F_t^{\tau}$ . These functions represent the optimal 1UC schedule up until time t which includes the optimal economic dispatch. This has the advantage that we can identify functions that will not lead to optimality and do not have to calculate the next function  $F_{t+1}^{\tau+1}$ . This is based on the following proposition:

**Proposition VI.1.** If a function  $F_t^{\tau}$  has the property  $\forall p \in [\underline{P}, \overline{P}] F_t^{\tau}(p) > \min_{\tau' \in [M_{up}, t]} F_t^{\tau'}(p)$  then a schedule in which the generator at time t is on for  $\tau$  time steps cannot be optimal.

Proof is omitted for brevity. The idea is that for any t and  $p_t \in [\underline{P}, \overline{P}]$  all states  $(on_t^{\tau}, p_t)$  where  $\tau \in \{M_{up}, t\}$  have equivalent state transitions. Therefore the antecedent implies there exists a better schedule for every  $p_t \in [\underline{P}, \overline{P}]$ . Therefore  $F_t^{\tau}(p)$  can be forgotten and consequently all following  $F_{t+i}^{\tau+i}$ ,  $1 \leq i < n$ .

Irrelevant functions can be identified in multiple ways. One way is to trace the minimum of all  $F_t^{\tau}$  functions and mark those that are part of the minimum. After that we can remove those functions  $F_t^{\tau}$  that are not marked. The minimum can be traced by finding the function  $F_t^{\tau}$  that has the minimal value at <u>P</u>. This function is part of the minimum and if it intersects with another function at any point in [<u>P</u>, <u>P</u>] then that function



is also part of the minimum. We can find every function by iteratively finding intersections.

We call the algorithm that solves the recurrent relation (1)-(8) by constructing the  $F_t^{\tau}$  functions without removing irrelevant ones: *RRF* and the algorithm that removes the irrelevant functions *RRF*+. The whole procedure of *RRF* and *RRF*+ is outlined in Algorithm 1.

# Algorithm 1 RRF(+)

1: Initialise  $F_1^{\tau}$  and  $c(off_1^{\tau})$ 2: for all  $t \in \{2, ..., n\}$  do for all  $\tau \in \{1, \ldots, M_{\text{down}}\}$  do 3: 4: Determine  $c(of f_t^{\tau})$  with (2), (3) and (4) end for 5: for all  $\tau \in \{1, \ldots, \max(t, M_{up})\}/$ Irrelevant do 6: Determine  $F_t^{\tau}$  with (9), (19), (20) and (21) 7: 8: end for 9: (Remove irrelevant functions by finding intersections) 10: end for 11: Backtrack to get the solution

# A. Time Complexity

Let *m* be the maximum number of intervals of all  $F_t^{\tau}$  functions at any time and let *k* be the maximum number of relevant functions at any time:

$$m = \max_{t \in \{1,\dots,n\}\tau \in \{1,\dots,t\}} intervals(F_t^{\tau})$$
(22)

$$k = \max_{t \in \{1,\dots,n\}} \left| \left\{ \tau \mid \begin{array}{c} \tau \in \{1,\dots,t\}, \exists p \in [\underline{P},P] \\ F_t^{\tau}(p) \le \min_{\tau' \in \{1,\dots,t\}} F_t^{\tau'}(p) \end{array} \right\} \right| \quad (23)$$

Regarding the complexity of RRF(+), Line 7 is repeated O(nkm) times since the O(k) relevant functions have O(m) intervals. Line 9 is repeated O(n) times and has a cost of  $O(mk^2)$  since the maximum number of intersections that can occur for O(k) functions with O(m) intervals is  $O(mk^2)$ .

Therefore, the total time complexity of RRF+ is  $O(nmk^2)$ and RRF is  $O(n^2m)$  since line 9 is skipped in which irrelevant functions are removed. As the numbers k and m theoretically could both be O(n) the time complexity of RRF+ and RRFare  $O(n^4)$  and  $O(n^3)$  respectively.

However, we found that every function only has a small number of intervals when we use generator data from specific UC problems described in the literature. Moreover only a small (most of the time only one) number of functions is minimal at some point  $p_t \in [\underline{P}, \overline{P}]$ . In this case our algorithm RRF+ runs in linear time (Table I).

		$f_t$ is convex					
Author/ Name	Time	Time Detailed	Note				
Frangioni et al. [2]	$O(n^3)$						
RRF (this work)	$O(n^3)$	$O(n^2m)$	$m = \max$ intervals				
RRF+ (this work)	$O(n^4)$	$O(nmk^2)$	$k = \max$ relevant functions				
		TABLE Í					
OVERVIEW 1UC ALGORITHMS							

#### VII. SECOND RECURRENCE RELATION

In this section we present a different but equivalent recurrence relation that is based on the fact that it is only necessary to keep track of how long a generator has been on up until the minimum uptime is reached. Therefore we could reduce the state space by reformulating the states  $S_t$  (15) to  $S'_t$ :

$$S'_{t} = \bigcup \begin{cases} \{off_{t}^{\tau} \mid \tau \in \{1 \dots M_{\text{down}}\}\} \\ \{(on_{t}^{\tau}, p_{t}) \mid \tau \in \{1 \dots M_{\text{up}}\}, p_{t} \in [\underline{P}, \overline{P}]\} \end{cases}$$

$$(27)$$

See Figure 3 for the binary commitment variables statespace and transitions. Now the state  $(on_t^{\tau}, p)$  represents the state at time t where the generator is on for  $\tau$  time steps and produces  $p_t$  at time t but when  $\tau = M_{\rm up}$  it represents that the generator is on for at least  $M_{\rm up}$  time steps.

Besides reducing the state space this also introduces an additional type of state transition that can be made. Now we can also get to the last on-state  $(on_t^{M_{up}}, p_t)$  from  $(on_{t-1}^{M_{up}}, p_{t-1})$ if the difference between  $p_{t-1}$  and  $p_t$  respects the ramping limits. We define a new recurrence relation  $c'(s_t)$  similar to  $c(s_t)$ . For brevity we only show the parts of the recurrence relation that differ from  $c(s_t)$  in (24) and (25).

Again since we cannot compute the cost c'(s) for every possible state s we need to construct a function  $H_t^{\tau}$  such that:  $H_t^{\tau}(p_t) = c'(on_t^{\tau}, p_t)$  and is defined in (26). Now it is easy to see that  $\forall t \in \{M_{up} \dots n\}$ :

$$H_t^{M_{\rm up}}(p) = \min_{\tau \in \{M_{\rm up}...t\}} F_t^{\tau}(p) \qquad \forall p \in [\underline{P}, \overline{P}]$$
(28)

For any time step t,  $H_t^{M_{\rm up}}$  is the minimum of multiple  $F_t^{\tau}$  functions and this is the reason why we can remove irrelevant functions in Section VI. Effectively we are trying to find the smallest subset of  $F_t^{\tau}$  that represents  $H_t^{M_{\rm up}}$ .

We can solve 1UC either by constructing  $F_t^{\tau}$  or  $H_t^{\tau}$ . For  $F_t^{\tau}$  there exists an efficient method to construct these functions and to identify redundant functions. Constructing  $H_t^{\tau}$  for any convex function  $f_t$  is harder since we need to iteratively take the point-wise sliding minimum of two functions, one convex  $H_{t-1}^{M_{\text{down}}-1}$  and one non-convex  $H_{t-1}^{M_{\text{down}}}$  see (26).

#### VIII. ALGORITHM RRH

We will now show a way to construct  $H_t^{\tau}$  when  $f_t$  is piecewise linear by storing  $H_t^{\tau}$  at a finite set of points. When solving the recurrence relation in this way the state space becomes equivalent to the state space of Guan et al. [3]. But we found a mistake in their formulation, fixed that mistake<sup>4</sup> and made the algorithm more general. Our algorithm also works with non-equal ramping limits. Moreover, by a more in-depth analysis of the recurrence relation we make the algorithm more efficient.

When  $f_t$  is piece-wise linear we can represent  $H_t^{\tau}$  by only storing a finite set of points that must contain the optimum of  $F_t^{\tau}$  and, by a consequence of (28), of  $H_t^{\tau}$ . This set also

 $^4 {\rm For}$  example the point  $SU-\Delta$  where  $\Delta=\Delta^+=\Delta^-$  should also be included.

$$c'(of f_t^{\tau}) = \min_{p_{t-1} \in [\underline{P}, SD]} c'(on_{t-1}^{M_{up}}, p_{t-1}) + c_{stop} \qquad t > 1 \quad \tau = 1$$
(24)  
$$c'(on_t^{\tau}, p_t) = f_t(p_t) + \min_{p_{t-1} \in [p_t - \Delta^+, p_t + \Delta^-]} \min\{c'(on_{t-1}^{\tau-1}, p_{t-1}), c'(on_{t-1}^{\tau}, p_{t-1})\} \quad t > 1 \quad \tau = M_{up}, \underline{P} \le p_t \le \overline{P}$$
(25)

$$H_{t}^{\tau}(p_{t}) = \begin{cases} f_{t}(p_{t}) & t = 1, \underline{P} \leq p_{t} \leq \overline{P} \\ f_{t}(p_{t}) + c'(off_{t-1}^{M_{\text{down}}}) + c_{start} & t > 1, \tau = 1, \underline{P} \leq p_{t} \leq SU \\ f_{t}(p_{t}) + \min_{p_{t-1} \in [p_{t} - \Delta^{+}, p_{t} + \Delta^{-}]} H_{t-1}^{\tau-1}(p_{t-1}) & t > 1, 1 < \tau < M_{\text{up}}, \underline{P} \leq p_{t} \leq \overline{P} \\ f_{t}(p_{t}) + \min_{p_{t-1} \in [p_{t} - \Delta^{+}, p_{t} + \Delta^{-}]} \min\{H_{t-1}^{\tau-1}(p_{t-1}), H_{t-1}^{\tau}(p_{t-1})\} & t > 1, \tau = M_{\text{up}}, \underline{P} \leq p_{t} \leq \overline{P} \\ \infty & \text{otherwise} \end{cases}$$
(26)



Fig. 3. The new commitment statespace and transitions of 7 time steps. The off-states are represented as white nodes and the on-states as black nodes.

contains the points that are required to compute the optimum. These are the points that are computed when solving the recurrence relation, i.e. points that are on the path found by backtracking from optimal points. We use  $B^{f_t}$  to denote the set of breakpoints of the piece-wise linear cost function  $f_t$ . We use B to denote the set of all possible optimal points. We use Q to denote the set of all optimal points plus those points required to compute  $H_t^{\tau}(b)$  for  $b \in B$ . Let the set B be defined as:

1)  $\{\underline{P}, SU, \overline{P}\} \cup B^{f_t} \in B$ 2) if  $p \in B$  and  $p + \Delta^+ < \overline{P}$  then  $p + \Delta^+ \in B$ 3) if  $p \in B$  and  $p - \Delta^- > P$  then  $p - \Delta^- \in B$ 

5) If 
$$p \in D$$
 and  $p - \Delta \ge \underline{P}$  then  $p - \Delta \in D$ 

Let the set Q be defined as:

- 1)  $B \cup \{SD\} \subseteq Q$
- 2) if  $p \in Q$  and  $p \Delta^+ > \underline{P}$  then  $p \Delta^+ \in Q$
- 3) if  $p \in Q$  and  $p + \Delta^- < \overline{P}$  then  $p + \Delta^- \in Q$

All proofs regarding B and Q are in the Appendix. When we solve the recurrence relation c' we need to calculate  $H_t^{\tau}(p_t)$  for every  $t \in \{1, \ldots, n\}, \tau \in \{1, \ldots, M_{up}\}$  and  $p_t \in Q$ .

For a single function  $H_t^{\tau}$  where  $\tau > 1$  and for every  $p_t \in Q$ we need to find the point  $p_{t-1} \in [p_t - \Delta^+, p_t + \Delta^-] \cap Q$  where  $H_{t-1}^{\tau-1}(p_{t-1})$  is minimal. Finding this point can be trivially done in O(|Q|) resulting in a total time complexity of  $O(|Q|^2)$ .

However if  $\tau < M_{up}$  then  $H_t^{\tau}$  is still a convex function and the minimal point in Q is given as the argument of the minimum of (16) and can be computed in O(1).

The case where  $\tau = M_{\rm up}$  could also be improved. We can make use of the property that for two consecutive points

 $p_t, p_t' \in Q$  the set representing the interval around the points  $Q \cap [p_t - \Delta^+, p_t + \Delta^-]$  and  $Q \cap [p_t' - \Delta^+, p_t' + \Delta^-]$  shares the majority of elements. To find to minimum value for all points is finding the sliding minimum over an array. All minimal points can be found with a double-ended queue in O(|Q|). The full Algorithm *RRH* is described here:

# Algorithm 2 RRH

1: Initialise  $H_1^{\tau}$  and  $c(of f_1^{\tau})$ 2: for all  $t \in \{2, ..., n\}$  do Compute  $c(of f_t^{\tau})$  with (2), (24) and (25) 3: for all  $\tau \in \{2, \ldots, M_{up} - 1\}$  do 4: Determine  $H_t^{\tau}(p_t)$  with (26),(16) for all  $p_t \in Q$ 5: 6: end for 7: Create Dequeue D 8: for all  $p \in Q$  do for all  $q \in (Q \cap [p - \Delta^+, p + \Delta^-]) - D$  do 9: Remove elements from end of D that 10: have value > max{ $H_t^{M_{up}}(q), H_t^{M_{up}-1}(q)$ } 11: Add q to the end of D 12: end for 13: Remove  $q \in D - (Q \cap [p - \Delta^+, p + \Delta^-])$  from D 14:  $q \leftarrow$  the front of D 15:  $H_t^{\tau}(p) = \max\{H_t^{M_{\rm up}}(q), H_t^{M_{\rm up}-1}(q)\} + f_t(p)$ 16: end for 17: 18: end for 19: Backtrack to get the solution

Creating all  $H_t^{\tau}$  has a time complexity of  $O(n \cdot |Q|)$ . The overall time complexity of *RRH* therefore becomes  $O(n \cdot |Q|)$  which is an improvement to complexity of  $O(n \cdot |Q|^2)$  from Guan et al. [3].

$f_t$ is convex piece-wise linear								
Author/ Name	Time	Time Detailed	Restrictions					
Fan et al. [4]	$O(n^3)$		$\Delta^+ = \Delta^-, SU = SD$					
Guan et al. [3]	O(n)	$O(n \cdot  Q ^2)$	$\Delta^+ = \Delta^-, SU = SD$					
RRH (this work)	O(n)	$O(n \cdot  Q )$						
		TABLE II						
OVERVIEW 1UC ALGORITHMS								

# IX. COMPUTATIONAL RESULTS

To test the efficiency we have implemented five algorithms for linear and quadratic generation cost and tested it on generator data from instances in the UC literature. We gathered the generator data of power systems from the following sources:

- A110, 110 generator instance (Orero and Irving [6]).
- TAI38, 38 generator instance (Huang et al. [7]).
- GA10, 10 generator instance (Kazarlis et al. [8]).
- KOR140, 140 generator instance (Park et al. [9]).

• RCUC200, 200 generator instance (Frangioni et al. [10]). For all instances we solved the problem with 10 different time series of electricity prices for every generator for 10 different horizons  $\in \{100, 200, \ldots, 1000\}$ . For every time step,  $f_t$  is constructed from the generation cost of the generator and Lagrangian multipliers. In total we ran 49,900 different 1UC problems and compared the following algorithms:

- RRF+, here we solve the recurrence relation c by constructing  $F_t^{\tau}$  and remove irrelevant functions as described in section VI.
- *RRH*, here we solve the recurrence relation c' by constructing H<sup>τ</sup><sub>t</sub> with values in Q efficiently by the method described in section VIII.
- Guan\*, here we solve the recurrence relation c' by constructing  $H_t^{\tau}$  with values in Q but without the proposed time complexity improvements. Here the algorithm is equivalent to the algorithm of Guan et al. [3] if the formulation was complete and extended for non-equal ramping limits (hence the star).
- *RRF*, here we solve recurrence relation c by constructing  $F_t^{\tau}$  without removing irrelevant functions. This algorithm is, therefore, comparable to Frangioni and Gentile [2] with the graph reduction mentioned in [3].
- *Gurobi*, at last we implemented the problem as a MI(Q)P in Gurobi with a standard 3-bin formulation [11].

All algorithms where written in C# and run on an i7-8700K 3.70 GHz processor running on Windows 10.

#### A. Results

		$f_t$ is linear				$f_t$ is quadratic			
		RH	nan*	RF+	RF	iqan	RF+	RF	urobi
Instance	n	R	5	R	R	5	R	R	G
GA10	100	0.2	0.3	0.8	2.2	17.1	0.6	2.5	22.6
TAI38		1	37.2	0.4	1.7	16.1	0.4	2.1	100
A110		0.1	0.2	0.4	1.6	13.1	0.3	2.1	17.4
KOR140		0.2	0.8	0.4	1.5	12.4	0.3	2	141.4
RCUC200		2.3	101.5	0.5	1.9	20.9	0.4	2.3	105.4
GA10	500	0.7	1.6	3.5	55.1	162.9	2.7	60.3	453.4
TAI38		4.3	191.9	2.4	44.1	44.8	1.7	52.6	853.2
A110		0.5	0.7	1.5	38.8	56.8	1.2	51.2	317.4
KOR140		1.1	3.8	2.4	38.4	55.2	1.7	49.7	2436.5
RCUC200		11	497.4	2.5	49.5	141.2	1.9	57.3	3410.9
GA10	1000	1.4	3.5	6.7	213.4	395.5	5.3	237.7	1355.8
TAI38		8.7	380.7	4.7	167.9	112.8	3.4	208.4	3838.3
A110		0.8	1.3	2.9	148.5	128.2	2.4	198.3	977.2
KOR140		2.2	7.8	4.8	152	122.2	3.4	198.9	3398.6
RCUC200		21.8	991.1	4.9	192	391.2	3.8	225.7	4980
TABLE III									

Average computation time to solve 1UC of 10 time series of electricity prices for all generators in 5 instances for 3 horizons (100, 500, and 1000 time steps), both for the piece-wise linear and quadratic cost function (in milliseconds).

	j	$f_t$ is linear	$f_t$ is quadratic					
Instance	$\max  Q $	$\max m$	$\max k$	$\max m$	$\max k$			
GA10	15	6	5	9	5			
TAI38	284	5	2	7	2			
A110	12	6	5	9	5			
KOR140	32	6	4	10	4			
RCUC200	251	7	4	8	4			
TABLE IV								

PER INSTANCE, MAXIMUM VALUES OF m, k and |Q| all generators in all time steps of 10 horizons with 10 time series of electricity prices.



Fig. 4. The growth of the average computation time in milliseconds when the amount of time steps increases when the generation cost function is linear.

The results are given in Table III, Table IV, Figure 4 and Figure 5. Table III contains the average running time in milliseconds of the 1UC problems. Table IV contains for each instance the maximum amount of points in Q, the maximum number of intervals and the maximum number of relevant functions for all experiments. Figure 4 shows the growth in computation time when the time horizon is increased and Figure 5 shows the ratio of problems solved for different performance ratios.

For all problem instances RRF + and RRH outperform the other algorithms in terms of computation time. From Figure 5 you can see that in 40% of the cases RRH and in 60% of the cases RRF + has the lowest computation time.

When the generation cost is linear the geometric average speed up of RRH compared to RRF+,  $Guan^*$ , RRF and Gurobi is 1.3, 7.3, 17.6 and 45.8. The geometric average speed up of RRF+ compared to  $Guan^*$ , RRF and Gurobi is 5.7, 13.6 and 35.7. When the generation cost is quadratic the geometric average speed up of RRF+ compared to  $Guan^*$ , RRF and Gurobi is 22.0 and 387.9. Table III shows that even for large time steps both algorithms solve 1UC in a few milliseconds.

The fact that *Gurobi* performs as one of the worst is also not surprising since it is a general solver that tries to compete



Fig. 5. Graph that shows for increasing performance ratios the ratio of problems that are solved within a factor, the performance ratio, of the best performing algorithm for each instance.

with algorithms specifically designed for 1UC.

RRF + outperforms RRF as it is a direct improvement of RRF with a little computational overhead to identify redundant functions. In theory (for now) the worst case analysis of RRF + is worse than RRF but in practice it reduces the amount of functions needed from O(n) to some small amount. For these experiments we considered 49,900 different 1UC problems and k was at most 5 and m at most 10 (Table IV). These values where the same across all time horizons and did not increase when we increased n. We can also see the direct result of the linear vs quadratic growth in Figure 4.

The algorithm RRH is a direct improvement to  $Guan^*$ . For some instances where the set of optimal points Q is small (Table IV) there is little to no difference (A110,GA10) but for other instances where Q is much larger this difference becomes significant (see TAI38 and RCUC200 in Table III).

#### X. CONCLUSION

We introduced a recurrence relation that solves 1UC. In order to solve this recurrence relation we created multiple functions  $F_t^{\tau}$  that for each  $p_t \in [\underline{P}, \overline{P}]$  return the cost of the optimal schedule that is on at time t for  $\tau$  time steps and produces  $p_t$ . By creating these functions inductively we were able to identify irrelevant functions and remove them. Resulting in a time complexity of  $O(nmk^2)$  where k is the maximum number of relevant functions. We showed that for the instances studied k was at most 5 and m at most 10 and did not increase when we increased the amount of time steps. In practice this results in a computation time that grows linear in the amount of time steps which improves the previous cubic growth of the algorithm of Frangioni and Gentile.

We introduced a different recurrence relation  $H_t^{\tau}$  that requires fewer functions to be stored. These functions are harder to create than  $F_t^{\tau}$ . However we showed that in the special case where generation cost is piece-wise linear we only need to keep track of a finite number of points Q to represent these functions. The method of representing cost only at a finite number of points is similar to the DP algorithm of Guan et al. However our method works for non-equal ramping limits and we showed how to efficiently compute the value at Q, resulting in an improved algorithm in terms of generality and time complexity of  $O(n \cdot |Q|)$ .

We performed computational experiments with generator data from multiple power systems. The results show that our algorithm, RRF+, that identifies and removes irrelevant functions  $F_t^{\tau}$  and our algorithm, RRH, of representing  $H_t^{\tau}$  as a finite set points outperforms other methods with piece-wise linear and quadratic generation cost.

Both methods increase the efficiency and can solve the single-unit commitment problem for large time horizons in a few milliseconds. This could lead to significant improvements for solving large scale unit commitment problems with Lagrangian relaxation or related methods.

#### ACKNOWLEDGEMENT

This work is part of the research programme "Energie: Systeem Integratie en Big Data" with project number 647.003.005, which is financed by the Dutch Research Council (NWO).

#### REFERENCES

- Wim van Ackooij, I. Danti Lopez, and Antonio Frangioni, Fabrizio Lacalandra and Milad Tahanan, *Large-scale unit commitment under uncertainty: an updated literature survey* Annals of Operations Research 271.1, pp. 11–85, 2018.
- [2] Antonio Frangioni, and Claudio Gentile, Solving nonlinear single-unit commitment problems with ramping constraints Operations Research 54.4, pp. 767–775, 2006.
- [3] Yongpei Guan, Kai Pan and Kezhuo Zhou, Polynomial time algorithms and extended formulations for unit commitment problems IISE transactions 50.8, pp. 735–751, 2018.
- [4] Wei Fan, Xiaohong Guan, and Qiaozhu Zhai, A new method for unit commitment with ramping constraints Electric Power Systems Research 62.3, pp. 215–224, 2002.
- [5] Qiaozhu Zhai, Xiaohong Guan, and Feng Gao. Optimization based production planning with hybrid dynamics and constraints IEEE Transactions on Automatic Control 55.12, pp. 2778–2792, 2010.
- [6] SO Orero and MR Irving, Large scale unit commitment using a hybrid genetic algorithm International Journal of Electrical Power & Energy Systems 19.1, pp. 45–55, 1997.
- [7] Kun-Yuan Huang, Hong-Tzer Yang, and Ching-Lien Huang, A new ther- mal unit commitment approach using constraint logic programming IEEE Proceedings of the 20th International Conference on Power Industry Computer Applications, pp. 176–185, 1997.
- [8] Spyros A Kazarlis, AG Bakirtzis, and Vassilios Petridis, A genetic algorithm solution to the unit commitment problem IEEE transactions on power systems 11.1, pp. 83–92, 1996.
- [9] Jong-Bae Park et al. An improved particle swarm optimization for nonconvex economic dispatch problems IEEE Transactions on Power Systems 25.1, pp. 156–166, 2010.
- [10] Antonio Frangioni, Claudio Gentile, and Fabrizio Lacalandra. *Tighter ap proximated MILP formulations for unit commitment problems* IEEE Transactions on Power Systems 24.1, pp. 105–113, 2009.
- [11] Ostrowski, J., Anjos, M. F., Vannelli, A. *Tight mixed integer linear programming formulations for the unit commitment problem* IEEE Transactions on Power Systems 27(1), pp. 39–46, 2009.
- [12] Li, Zhigang, Wenchuan Wu, Boming Zhang, Bin Wang, and Hongbin Sun. Dynamic economic dispatch with spinning reserve constraints considering wind power integration. IEEE Power & Energy Society General Meeting, pp. 1-5. IEEE, 2013.

# XI. Appendix, proofs about B and Q

# **Proposition XI.1.** If $f_t$ is a piece-wise linear function then the break points of $F_t^{\tau}$ are only in B

**Proof.** For  $\tau = 1$  its trivial. For  $\tau > 1$  assume  $F_{t-1}^{\tau-1}$  only has breakpoints in B. Recall that  $\min_{p_{t-1} \in [p_t - \Delta^+, p_t + \Delta^-]} F_{t-1}^{\tau-1}(p_{t-1})$  is constructed from  $F_{t-1}^{\tau-1}$  by shifting the intervals of  $F_{t-1}^{\tau-1}$  and introducing one new interval in the case where  $f_t$  is a piece-wise linear cost function. The new breakpoints are those already in B but shifted down by  $\Delta^-$  or up by  $\Delta^+$  and if they exceed  $\underline{P}$  or  $\overline{P}$  they become  $\underline{P}$  or  $\overline{P}$ . In the first case those new points are in B by definition 2) and 3), in the later case they are in B by 1). The addition of  $f_t$  can add breakpoints at  $B^{f_t}$ .  $\Box$ 

**Proposition XI.2.** If  $F_t^{\tau}$  is a piece-wise linear function then a minimal point in interval [p, p'] of  $F_t^{\tau}$  is in  $B \cup \{p, p'\}$ 

*Proof.* Suppose the optimal point  $p_t^*$  lies in [p, p']. Then from Proposition XI.1 we know  $p_t^* \in B$ . Suppose  $p_t^* \notin [p, p']$  since  $F_t^{\tau}$  is convex the minimal point in [p, p'] is as close as possible to  $p_t^*$  bounded by the interval, this is either p or p'.  $\Box$ 

**Proposition XI.3.** If  $F_t^{\tau}$  is a piece-wise linear function then then the minimum value at the end of a on-period has a production value in  $B \cup \{SD\}$ .

*Proof.* If the generator is last on at t then it can only produce  $p_t \in [\underline{P}, SD]$ . Combined with Proposition XI.2 this is minimal in  $B \cup \{SD\}$ .

The minimum value at the end of a *on-period* has a production value in  $B \cup \{SD\}$ . These points however do not only depend on points in  $B \cup \{SD\}$ . Optimal points can also depend on non-optimal points. We can iteratively capture these additional points by the set Q. That is a set of all the points from which  $F_t^{\tau}$  is minimal plus all the points that are needed to compute the minimal points.

The recursive rules of Q come from the following observation: suppose we want to know the value of  $F_t^{\tau}(p_t)$  for some point  $p_t \in \{B \cup SD\}$ . The value of this point is constructed from the minimum in  $[p_t - \Delta^+, p_t + \Delta^-]$ . From Proposition XI.2 we know this minimum is in  $Q \cup \{p_t - \Delta^+, p_t + \Delta^-\}$ . The set Q therefore needs to contain all the points that are on the path found by backtracking from optimal points.

From (28) its easy to see the three propositions also hold for  $H_t^{\tau}$ . Moreover Q are the only points we need to store to represent  $H_t^{\tau}$ .

