Solving Nonlinear Single-Unit Commitment Problems with Ramping Constraints

Antonio Frangioni
Dipartimento di Informatica, Università di Pisa
Largo B. Pontecorvo 3, 56127 Pisa – Italy
frangio@di.unipi.it

Claudio Gentile
Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti", C.N.R.
Viale Manzoni 30, 00185 Rome – Italy
gentile@iasi.cnr.it

Abstract

We present a dynamic programming algorithm for solving the Single-Unit Commitment (1UC) problem with ramping constraints and arbitrary convex cost functions. The algorithm is based on a new approach for efficiently solving the single-unit Economic Dispatch (ED) problem with ramping constraints and arbitrary convex cost functions, improving on previously known ones that were limited to piecewise-linear functions. For simple convex functions, such as the quadratic ones typically used in applications, the solution cost of all the involved (ED) problems, comprised that of finding an optimal primal and dual solution, is $O(n^3)$. Coupled with a special visit of the state-space graph in the dynamic programming algorithm, that enables one to solve (1UC) with simple convex functions in $O(n^3)$ overall.

Keywords: *Dynamic Programming, Unit Commitment problem, Ramping Constraints*

1 Introduction

The Single-Unit Commitment problem (1UC) requires to optimally operate one generating thermal unit within a certain discretized time horizon. The cost (or revenue) for generating power varies with each time instant. The generating unit is subject to some technical restrictions, most notably *minimum up- and down-time* constraints, as well as upper and lower bounds over the produced power when the unit is operational.

(1UC) is a mixed-integer nonlinear problem, hence in general nontrivial to solve. It is particularly relevant because it appears as a sub-problem to be repeatedly solved within

Lagrangian schemes for (multi-)Unit Commitment (UC) problems, that require to coordinate the operations of several generating units, within a certain discretized time horizon, to satisfy a given power demand at minimum cost. These Lagrangian schemes are among the most efficient solution techniques for this class of difficult, large-scale mixed-integer nonlinear problems, see e.g. [1], [4], [5], [6], [10], [12] among others, not least because they are easily extended to accommodate contributions from other types of generating units, such as hydro-electric ones. Also, Lagrangian techniques can be relatively easily extended to consider constraints arising from selling the generated power on a free market as in [7].

Within a Lagrangian approach, one (1UC) per each generating unit is repeatedly solved with varying objective function, whence the need for efficient solution methods for this problem. When no *ramping constraints* are imposed, (1UC) can be solved by means of a two-stage process: first the optimal generated power, if the unit is committed, is independently computed for each time period, and then the optimal set of time periods where the unit has to be committed is computed, taking into account the results of the previous phase, by means of a simple dynamic programming procedure. The resulting algorithm has a complexity of $O(n)$, n being the number of time instants in the discretized time horizon, if the start-up cost of the unit is *time invariant*, and $O(n^2)$ if the start-up cost of the unit is *time dependent*, i.e., the cost of committing the unit at a certain time instant depends on how long the unit has been uncommitted.

Unfortunately, this approach fails when ramping constraints need to be considered. Ramping constraints limit the maximum increase or decrease of generated power from one time instant to the next, and reflect the thermal and mechanical inertia that has to be overtaken in order for the unit to increase or decrease its output. These phenomena cannot be disregarded for large units or if the time discretization interval is small (e.g., 15 minutes). The reason of the failure is that the variables representing the power output are no longer independent, once that commitment decisions have been taken; rather, they are linked by the ramping constraints. Hence, it is no longer possible to determine the optimal generated power, if the unit is committed, independently for each time period. Thus, the dynamic programming procedure for the case without ramping constraints cannot be extended to determine the optimal commitment. Discretizing the power variables space one may keep using a standard dynamic programming procedure [3], but the computational burden increases considerably, and the obtained solution is an approximated one.

In [8], an approach is presented for efficiently solving (1UC) with ramping constraints when the cost function is piecewise-linear. The approach is based on the following idea: re-define the state-space of the dynamic programming procedure so that computation of the state costs reduces to a convex (although harder than in the standard case) problem, the *Economic Dispatch with Ramping Constraints* (ED). The efficiency is obtained by using a constructive dynamic programming procedure that solves (ED) with a piecewise-linear cost function, similar to that of [2] and [11]. Thus, two nested dynamic programming procedures are employed in order to obtain an overall efficient approach.

However, in most cases the cost function of the real unit is modeled, in (1UC), with a quadratic function. Closely approximating the quadratic function with a piecewise-linear one may require a large number of pieces, thereby increasing the cost of the overall solution procedure. We propose an efficient algorithm for (ED) with general convex cost functions that solves all the $O(n^2)$ (ED)s required to perform the dynamic programming procedure on the commitment decisions in $O(n^3)$ in the case of quadratic cost functions. The algorithm is simple to implement and works for a very general form of (1UC) with time-varying upper and lower limits over the generated power, as well as time-varying and different limits for ramp-up and ramp-down constraints. Coupled with a special visit of the state-space graph in the dynamic programming algorithm, this enables one to solve (1UC) in $O(n^3)$ overall in the case of quadratic cost functions.

The structure of the paper is as follows. In Section 2 a formulation of (1UC) is briefly presented. In Section 3 the dynamic programming procedure, similar to that of [8], for solving (1UC) is recalled, the corresponding (ED) problems are discussed, and the special visit is described that allows one to solve the problem in $O(n^3)$ once that all the node costs have been computed. Then, in Section 4 the algorithm for solving (ED) is presented and analyzed. Finally, in Section 5 some computational results, obtained in the context of a Lagrangian approach to (UC), are presented for showing the efficiency of the proposed approach, and in Section 6 conclusions are drawn.

2 Formulation

The Single-Unit Commitment problem (1UC) is as follows. A thermal generating unit, burning some type of fuel (oil, gas, coal, ...) is given. The unit is characterized by a maximum and minimum power output, l^t and u^t, respectively, for each time instant (e.g., hour or half-hour) in a set $T = \{1, \ldots, n\}$, covering some time horizon (e.g., a day or a week). If the unit is committed (actively generating power) at time instant t, it is subject to a convex power generating cost function $f^t(p_t)$, where p_t is the amount of power produced. In the following we will only assume that f^t is closed convex and that $f^t(0) = 0$ (any constant term in f^t can be associated with commitment variables, as discussed next). The operation of the unit must satisfy a number of technical constraints, typically the *minimum up- and down-time* ones: whenever the unit is turned on it must remain committed for at least τ^+ consecutive time instants, and, analogously, whenever the unit is turned off it must remain uncommitted for at least τ^- consecutive time instants. It is therefore useful to introduce binary variables x_t indicating (if 1) the commitment of the unit at time instant t. We then define X as the set of schedules respecting minimum up- and down-time constraints; also, for any $x \in X$, we define $c(x)$ as the cost of the schedule: this may comprise fixed generating cost and time-dependent or time-invariant start-up cost. Note that, since costs are usually modified by Lagrangian multipliers, the cost functions $c(x)$ and $f^t(p_t)$ may also take negative values. Other combinatorial constraints and costs could be included as well, as discussed in Section 6, as long as they are consistent with the dynamic programming procedure discussed in the next section.

The last set of technical requirements are the ramping constraints. These require that the maximum increase of generated power from time instant t to the next be limited to $\Delta_+^t > 0$, and, analogously, the maximum decrease of generated power from time instant t to the next be limited to $\Delta_-^t > 0$. Note that this definition can be applied only if the unit is committed in both time periods t and $t+1$. We therefore consider a general form of ramping constraints where an upper bound $\bar{l}^t$, $l^t \leq \bar{l}^t \leq u^t$, is known on the maximum amount of power that can be generated if the unit is turned on in time period t (that is, it was uncommitted in $t-1$)

and, analogously, an upper bound $\bar{u}^t$, $l^t \leq \bar{u}^t \leq u^t$, is known on the maximum amount of power that can be generated if the unit is turned off at the end of time period t (that is, it will be uncommitted in $t + 1$).

A formulation of (1UC) is:

$$\min \; c(x) + \sum_{t \in T} f^t(p_t) \tag{1}$$

$$l^t x_t \leq p_t \leq u^t x_t \qquad t \in T \tag{2}$$

$$p_{t+1} \leq p_t + x_t \Delta_+^t + (1 - x_t)\bar{l}^{t+1} \qquad t = 0, \dots, n-1 \tag{3}$$

$$p_t \leq p_{t+1} + x_{t+1} \Delta_-^t + (1 - x_{t+1})\bar{u}^t \qquad t = 0, \dots, n-1 \tag{4}$$

$$x \in X \tag{5}$$

Constraints (3) are *ramp-up* constraints, i.e., they limit the maximum increase in power attainable at time instant t (assuming that the unit is committed in t). Note that we assume to know the state of the unit at the time instant prior to the beginning of the operation, i.e., its commitment x_0 and the generated power p_0. Also, for the sake of minimum up- and down-time constraints, we assume to know how long the unit has been on (if $x_0 = 1$) or off (if $x_0 = 0$). Constraints (4) are *ramp-down* constraints, i.e., they limit the maximum decrease in power attainable at time instant t. We remark that this formulation is more general than those usually considered (cf. [2, 8, 11]) not only because the cost functions need not be piecewise-linear, but also because we allow different limits Δ_+^t and Δ_-^t for ramp-up and ramp-down constraints, and we allow them to depend on the time instant t.

It is well-known that, if constraints (3) and (4) are not present, (1UC) is easily solvable by a two-stage procedure. First, the unconstrained optimum of each f^t

$$\tilde{p}_t = \operatorname{argmin} \{ f^t(p) \; : \; p \in \mathbb{R} \} \tag{6}$$

(assumed unique for simplicity) is computed, and used to find the optimal power production level if the unit is committed

$$p_t^* = \min \{ \, u^t, \max \{ \, \tilde{p}_t, l^t \, \} \, \} = \operatorname{argmin} \{ \, f^t(p) \; : \; l^t \leq p \leq u^t \, \} \tag{7}$$

by simply projecting $\tilde{p}_t$ over the feasible set $[l^t, u^t]$. The value $z^t = f^t(p_t^*)$ is the contribution of variable p_t to the objective function value if the unit is committed at time instant t ($x_t = 1$), while 0 is the contribution if the unit is uncommitted ($x_t = 0$). Thus, z^t is the cost (or revenue) of committing the unit, to be considered together with fixed costs and start-up costs, effectively eliminating the p_t variables from the problem. The remaining combinatorial problem in the x_t variables alone can be easily solved by dynamic programming. In a dynamic programming approach, the first step is the identification of the *state space* of all the possible solutions and partial solutions. When this space is composed by a finite number of elements, it is possible to represent it as a directed acyclic graph where the nodes are the states and the arcs identifies the transitions among the states; it is then possible to find an optimal solution by computing a shortest path on this graph.

In the simple case where start-up cost are time invariant, that is, the cost of starting up the unit at a certain time instant does not depend on how long the unit has been uncommitted (but it may depend on the specific time instant), the state space of the dynamic programming is made of $2n$ nodes, say $(t, 1)$ and $(t, 0)$ for $t \in T$, representing respectively the unit being committed ($x_t = 1$) or not ($x_t = 0$) at time instant t, plus a source s and a sink d. There are arcs between nodes $(t, 1)$ and $(t + 1, 1)$ for all $t < n$, representing the fact that the unit, that has already passed the τ^+ periods of mandatory commitment, is kept on in time instant t, labeled with the sum of the corresponding z^t and fixed cost (if any). Analogously, there are arcs between nodes $(t, 0)$ and $(t + 1, 0)$ for all $t < n$, representing the fact that the unit, that has already passed the τ^- periods of mandatory uncommitment, is kept off in time instant t, labeled with zero cost. Then, there are arcs for state switches, i.e., arcs from $(t, 1)$ to $(t + \tau^-, 0)$, with zero cost, indicating the shutdown of the unit at time instant $t + 1$ and its remaining uncommitted for the following τ^- periods, and arcs $(t, 0)$ to $(t + \tau^+, 1)$, indicating the start-up of the unit at time instant $t + 1$ and its remaining committed for the following τ^+ periods, with the proper start-up cost plus the generating and fixed costs for all the interval. When $t + \tau^+$ (τ^-) is larger than n, the arcs go to the sink d, and the cost is properly modified. Then, there are arcs from the source s to the nodes compatible with the initial state of the unit. That is, if the unit is initially uncommitted since τ^0 time periods, there is an arc from s to $(\max\{\tau^- - \tau^0, 1\}, 0)$; if $\tau^- > \tau^0$ this indicates that the unit has to remain uncommitted for the first $\tau^- - \tau^0$ time periods. Analogously, if the unit is initially committed since τ^0 time periods, there is an arc from s to $(\max\{\tau^+ - \tau^0, 1\}, 1)$; if $\tau^0 < \tau^+$ this indicates that the unit has to remain committed for the first $\tau^+ - \tau^0$ time periods, with appropriate cost. Finally, there are zero-cost arcs from $(n, 1)$ and $(n, 0)$ to the sink d.

Clearly, every $s - d$ path on this graph represents a feasible solution to (1UC), and the cost of the path is equal to the cost of the solution. Hence, (1UC) is reduced to a shortest path problem on an acyclic graph, that can be solved in linear time on the number of arcs, i.e., in $O(n)$.

If time-dependent start-up costs have to be considered, the graph has to be expanded somewhat, introducing nodes $(t, -k)$ indicating that the unit has remained uncommitted for the last k consecutive time instants, and properly modifying the arcs. The maximum value of k that has to be considered is the number of time instants after which the unit has completely "cooled off", i.e., a restart has the same cost than a cold start; in general, this value may be as large as n, although usually it is smaller. Thus, the size of the graph grows from $O(n)$ to $O(n^2)$ in the worst case, and the complexity of the procedure increases accordingly.

This procedure, however, fails if constraints (3) and (4) are present. In fact, the p_t variables are no longer independent, once that the x_t variables are fixed, since they are linked together by the ramping constraints; hence, it is no longer possible to determine the optimal generated power p_t^*, and the corresponding contribution z^t of variable p_t to the objective function value, independently for each time period.

3 The dynamic programming procedure

In order to solve (1UC) with constraints (3) and (4), a different dynamic programming procedure can be used. The state space of the dynamic programming comprises, in principle, all pairs (h, k) for $h, k \in T$ and $k \geq h$, plus a source s and a sink d. The meaning of each state (h, k) is: the unit is turned on at time instant h (i.e., it was uncommitted at time instant $h-1$), and it will be turned off again at time instant k (i.e., it will be uncommitted at time instant $k+1$). Clearly, all states such that $k < h + \tau^+ - 1$ correspond to infeasible operations and need not to be constructed.

In the state-space graph G, there is an arc between node (h, k) and node (r, q) if $r \geq k + \tau^- + 1$, i.e., it is feasible to turn on the unit at time instant r given that it has been turned off at time instant k. Each of these arcs are labeled with the start-up cost of the unit at time instant r; note that time-dependent start-up costs are easily handled within this framework. There are also arcs from the source s to all nodes (h, k) compatible with the initial state of the unit. That is, if the unit is uncommitted since τ^0 time periods there is an arc from s to each node (h, k) such that $h + \tau^0 - 1 \geq \tau^-$; these arcs are labeled with the corresponding start-up cost. If instead the unit is committed since τ^0 time periods, there is an arc from s to each node $(1, k)$ such that $k + \tau^0 \geq \tau^+$, labeled with zero cost. Finally, there is a zero-cost arc from each node to the sink d.

Clearly, every $s-d$ path on this graph represents a feasible solution to (1UC). By now, the cost of the path only represent the contribution of start-up costs to the objective function. Obvioulsy, fixed generating costs (if any) can also be easily included: we can associate with each node (h, k) the sum of all fixed costs of all periods from h to k (extremes included) as cost of the node, since the unit will be committed in that interval.

Furthermore, for each node (h, k), the optimal contribution of the variable generating costs, that depend on the p_t variables, can be computed in polynomial time by solving the following *Economic Dispatch with Ramping Constraints* problem for the interval $[h, k]$:

$$\min \sum_{t=h}^{k} f^t(p_t) \tag{8}$$

$$l^t \leq p_t \leq u^t \qquad h \leq t \leq k \tag{9}$$

$$p_h \leq \bar{l}^h \tag{10}$$

$$p_{t+1} \leq p_t + \Delta_+^t \qquad t = h, \ldots, k-1 \tag{11}$$

$$p_t \leq p_{t+1} + \Delta_-^t \qquad t = h, \ldots, k-1 \tag{12}$$

$$p_k \leq \bar{u}^k \; . \tag{13}$$

We will denote problem (8) – (13) as (ED_{hk}). Since all the relevant binary variables are fixed, this is an optimization problem with convex objective function and linear constraints. Hence, its optimal objective function value $z^*_{hk} = z(ED_{hk})$ can be computed in polynomial time. By summing z^*_{hk} to the weight of each node (h, k), the cost of each $s-d$ path on the graph is that of the feasible solution it represents. Hence, once again (1UC) is reduced to a shortest path problem on an acyclic graph with $O(n^2)$ nodes and $O(n^4)$ arcs. Thus, the problem can be solved in $O(n^4)$ once that all the data has been computed.

However, the complexity of the visit can be reduced by exploiting some structural properties of the state-space graph G. Consider the set of nodes (h, k) in G partitioned into levels $V_k = \{(h, k) \; : \; 1 \leq h \leq k\}$ for $k \geq 1$ (level V_0 contains only the starting node s). From the definition of G, it immediately follows that:

- all nodes in V_k have the same set of adjacent nodes;

- the cost of the arc between (h, k) and (r, q) only depends on k and r.

Therefore, it is possible to visit G in ascending order of level k, avoiding to explicitly explore the forward star of all but one node for each level.

More in detail, the procedure works as follows. For each $k = 1, \ldots, n$ we keep a list $S_k \subseteq V_k$ of the reached nodes $(h, k) \in V_k$ with the label d_{hk} corresponding to the length of the shortest path found so far. S_0 contains s with label 0. For $k = 0, 1, \ldots, n$ we repeat the following steps:

- evaluate $z^*_{hk} = (ED_{hk})$ for all nodes in S_k (for $k = 0$ the result is zero, if $S_k = \emptyset$ skip to next value of k);
- find the node (h, k) in S_k with smallest value of $z^*_{hk} + d_{hk}$;
- visit all the adjacent nodes (r, q) of (h, k) computing the new value as the sum of $z^*_{hk} + d_{hk}$ and the cost of the arc between (h, k) and (r, q); if the node (r, q) is visited for the first time it is inserted in S_q, otherwise its label is updated if the new value is smaller than its old value.

Clearly, the chosen order is a valid one, and the visit terminates having determined a shortest $s - d$ path. In principle, all the $O(n^2)$ nodes of G are visited, and therefore the computation of all the corresponding z^*_{hk} values is required. However, for each k we only consider the node (h, k) associated with the shortest path from s, so that we need only check its $O((n-k)^2)$ outgoing arcs. Therefore, the complexity of the visit is reduced to $O(n^3)$ plus the cost of solving the $O(n^2)$ convex problems (ED_{hk}), with up to n variables.

Despite the relatively low size of the problem, this may turn out to be a heavy task, especially considering that several (1UC) problems are typically solved at each one of the many iterations of Lagrangian approaches to more complex (UC) problems (cf., e.g., [1, 4, 5, 6, 7, 10, 12]). Hence, solving (ED_{hk}) efficiently—or, more to the point, solving *all* the $O(n^2)$ of them efficiently—is crucial. In the next section we will develop an efficient dynamic programming algorithm for the solution of (sequences of) (ED_{hk}).

The approach is inspired by that of [8], where a similar state-space graph is defined. However, that paper was limited to (1UC) problems with piecewise-linear cost functions, and strongly used the (piecewise) linearity of the objective function in order to solve the Economic Dispatch problems; the idea was, using linear duality, to trace how the breakpoints and the slopes of the cost function change when moving from (ED_{hk}) to $(ED_{h(k+1)})$. Moreover, in that approach ramp-up and ramp-down constraints were not allowed to depend on the time instant, and initial and final upper bounds $\bar{l}^t$ and $\bar{u}^t$ are not handled.

4 Solving the economic dispatch problem

We will devise an algorithm for efficiently solving sequences of (ED_{hk}) problems for $k = h, \ldots, n$. In order to solve $(ED_{h(k+1)})$ by exploiting the solution of (ED_{hk}), it is necessary to introduce the parametric problem $(ED_{hk}(\bar{p}))$, i.e., the restriction to (ED_{hk}) corresponding to fixing the last variable p_k to the fixed value $\bar{p}$ (equivalently, imposing the extra constraint $p_k = \bar{p}$). We then study the properties of the optimal objective function value of $(ED_{hk}(\bar{p}))$

as a function of the parameter $\bar{p}$. To simplify matters somewhat, however, it is convenient to give a slightly different definition of the function under examination:

$$z_{hk}(\bar{p}) = \begin{cases} \min\{\ f^h(p_h)\ :\ (9)\ ,\ (10)\ ,\ p_h = \bar{p}\ \} & \text{if } h = k, \\ \min\{\ \sum_{t=h}^{k} f^t(p_t)\ :\ (9)\ ,\ (10)\ ,\ (11)\ ,\ (12)\ ,\ p_k = \bar{p}\ \} & \text{otherwise.} \end{cases}$$

That is, we allow $\bar{p}$ to assume any value in the interval $[l^k, u^k]$, even those values such that fixing $p_k = \bar{p}$ in formulation (8) – (13) would result in an infeasible problem due to the stricter upper bound imposed by constraint (13). This is done since we will use z_{hk} to compute $z_{h(k+1)}$; in the latter problem, constraint (13) corresponds to variable p_{k+1}, and therefore it is no longer binding for p_k.

We first state some general properties of the function:

Proposition 1 *The function z_{hk} is convex. Moreover, it has a piecewise nature, that is, it is finite-valued only in $v+1$ intervals $[m_0, m_1]$, $[m_1, m_2]$ $\dots [m_v, m_{v+1}]$, with $l^k \leq m_0$, $m_{v+1} \leq u^k$ and $v \leq 2(k-h)$, in which*

$$z_{hk}(\bar{p}) = z^i(\bar{p}) \qquad \textit{if } \bar{p} \in [m_i, m_{i+1}] \textit{ for } i = 0, 1, \dots, v$$

where each function z^i is the sum of at most $k-h+1$ functions f^t for $t \in \{h, h+1, \dots, k\}$ (and therefore it is convex).

Convexity of z_{hk} is a consequence of well-known general properties that need not be discussed here beyond noting that z_{hk} is the *value function* ([9]) of the convex program $(ED_{hk}(\bar{p}_k))$ with respect to the right-hand side $\bar{p}_k$ of its constraint $p_k = \bar{p}_k$. Its piecewise nature, and the more specific properties will be demonstrated next by outlining the steps for *efficiently constructing* the piecewise representation of z_{hk}.

We will proceed by induction to prove that the claimed properties are true; equivalently, we will (efficiently) construct piecewise representations of the functions z_{hh}, $z_{h(h+1)}$, $\dots z_{hk}$, in this order. At each step we will exploit the previously computed representation to construct that of the next problem. During the process, for each step k we will also (efficiently) compute and exploit

$$p^*_{hk} \ = \ \text{argmin}\{\ z_{hk}(p)\ :\ p \in [l^k, u^k]\ \}\ ,$$

that is, the k-th (last) component of the optimal solution of (ED_{hk}) where constraint (13) is relaxed.

At the basis of the induction process, the case $k = h$ is straightforward, since

$$z_{hh}(\bar{p}) = f^h(\bar{p})$$

for all $\bar{p} \in [l^h, \bar{l}^h]$. Hence, there are $v+1 = 1$ intervals and $v+1 = 1$ functions with the required properties (i.e., $0 = v \leq 2(h-h) = 0$). In this case, p^*_{hk} is just p^*_h as computed with formulae (6) and (7), but with upperbound equal to $\bar{l}^h$. When $h = 1$ and the unit was already committed, the set in which $z_{11}(\bar{p})$ is defined is slightly different: constraint (10) is not present, while p_0 and the ramping constraints must be considered, therefore $\bar{p} \in [\max\{l^1, p_0 - \Delta^1_-\}, \min\{u^1, p_0 + \Delta^1_+\}]$.

Now, we assume the claim proved for some value of k—and the corresponding set of intervals and functions to have already been explicitly computed—and proceed in proving that Proposition 1 holds for $k+1$, too. We will denote $\bar{m}_i$ the extremes of the intervals for $z_{h(k+1)}$, $\bar{z}^i$ the corresponding functions and $\bar{v}+1$ their number.

Consider any fixed value $\bar{p} \in [l^{k+1}, u^{k+1}]$. Constraints (11) and (12), written for $p_{k+1} = \bar{p}$ and p_k, result in

$$\bar{p} - \Delta_+^k \leq p_k \leq \bar{p} + \Delta_-^k \ .$$

Since z_{hk} is infinite-valued for p_k outside $[m_0, m_{v+1}]$, one has to set $\bar{m}_0 = \max\{l^{k+1}, m_0 - \Delta_-^k\}$ and $\bar{m}_{\bar{v}+1} = \min\{u^{k+1}, m_{v+1} + \Delta_+^k\}$; in fact, $z_{h(k+1)}$ is clearly infinite-valued outside this interval, and finite-valued inside it. Note that if $\bar{m}_0 > \bar{m}_{\bar{v}+1}$ infeasibility of the (1UC) problem has been detected.

Now, consider how the optimal solution to $(ED_{h(k+1)}(\bar{p}))$ can be computed, exploiting the (already computed) piecewise representation of z_{hk}. The problem can clearly be rewritten as

$$z_{h(k+1)}(\bar{p}) = f^{k+1}(\bar{p}) + \min\{\ z_{hk}(p_k) \ : \ m_0 \leq p_k \leq m_{v+1} \ , \ \bar{p} - \Delta_+^k \leq p_k \leq \bar{p} + \Delta_-^k \ \}.$$

In other words, the optimal solution to $(ED_{h(k+1)}(\bar{p}))$—at least, its k-th component—is just the constrained minimum of z_{hk} in the *intersection* of the intervals $[\bar{p} - \Delta_+^k, \bar{p} + \Delta_-^k]$ and $[m_0, m_{v+1}]$; we will denote that minimum as $p_k^*(\bar{p})$. In order to trace the function $z_{h(k+1)}(\bar{p})$, it is only necessary to understand how $p_k^*(\bar{p})$ behaves as $\bar{p}$ changes. This is, however, very simple to picture.

Consider the constrained minimum p_{hk}^* of z_{hk} over $[m_0, m_{v+1}]$ (or, equivalently, $[l^k, u^k]$), that we assume to have already computed: since z_{hk} is convex, $p_k^*(\bar{p})$ is just its projection over the feasible interval

$$[\bar{p} - \Delta_+^k, \bar{p} + \Delta_-^k] \tag{14}$$

(cf. (7)), that is,

$$p_k^*(\bar{p}) = \min\ \{\ \bar{p} + \Delta_-^k, \max\ \{\ p_{hk}^*, \bar{p} - \Delta_+^k \}\ \}\ . \tag{15}$$

Note that we are assuming p_{hk}^* to be unique; however, the following arguments can be easily extended to cases where z_{hk} has a (known) non-pointed interval as set of optimal solutions.

We can pictorially describe the process as follows, with the help of Figure 1. Basically, as $\bar{p}$ varies from $\bar{m}_0$ to $\bar{m}_{\bar{v}+1}$, three different cases can arise:

(a) When $\bar{p}$ is on the leftmost part of the interval $[\bar{m}_0, \bar{m}_{\bar{v}+1}]$ where $z_{h(k+1)}$ is finite-valued, e.g., $\bar{p} = \bar{m}_0$, p_{hk}^* is "on the right" of the feasible interval (14) (i.e., $p_{hk}^* > \bar{p} + \Delta_-^k$), $p_k^*(\bar{p}) = \bar{p} + \Delta_-^k$, that is, $p_k^*(\bar{p})$ is a linear function of $\bar{p}$.

(b) As $\bar{p}$ increases, eventually p_{hk}^* falls inside the feasible interval (14): then, $p_k^*(\bar{p})$ is equal to p_{hk}^*, remaining fixed until $\bar{p}$ becomes too large.

(c) Finally, $\bar{p}$ becomes larger than $p_{hk}^* + \Delta_+^k$, that is, p_{hk}^* does not fall anymore inside of the feasible interval (14), this time remaining "on the left"; then $p_k^*(\bar{p}) = \bar{p} - \Delta_+^k$, so, again, $p_k^*(\bar{p})$ increases linearly as $\bar{p}$ does.

Of course, not all three cases (a), (b) and (c) need necessarily happen. For instance, p_{hk}^* may already belong to (14) for $\bar{p} = \bar{m}_0$, or it may never leave it even if $\bar{p} = \bar{m}_{\bar{v}+1}$, and so on. However, the above three cases cover all that can possibly happen.

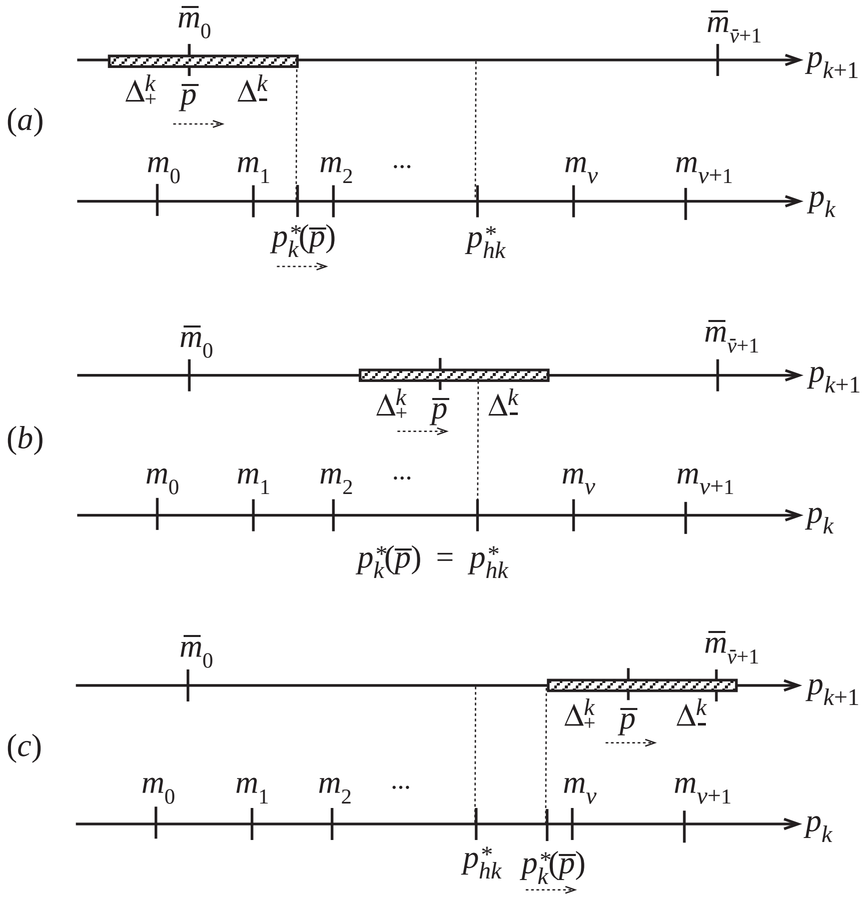

Figure 1: Evolution of $p_k^*(\bar{p})$ as $\bar{p}$ varies

It is now easy to see how, given the explicit description of z_{hk} in terms of the $v+1$ sub-intervals of $[m_0, m_{v+1}]$ and the associated functions z^i, we can efficiently construct a piecewise representation of $z_{h(k+1)}$ with $\bar{v}+1$ intervals where $\bar{v} \leq v+2$.

0) Set $\bar{p} = \bar{m}_0$, $\bar{v} = 0$, and let $0 \leq q \leq v$ be the index of the interval to which $p_k^*(\bar{p}) = \bar{p} + \Delta_-^k$ belongs (if it is a breakpoint, choose the interval on the right). Set $\bar{u} = \min\{u^{k+1}, m_{v+1} + \Delta_+^k\}$.

1) If case (a) is not verified goto step 2), otherwise set $\bar{z}^{\bar{v}}(p) = f^{k+1}(p) + z^q(p + \Delta_-^k)$. Compute the maximum value of $\bar{p}$ such that $p_k^*(\bar{p})$ remains in the q-th interval, p_{hk}^* remains outside the feasible interval and $\bar{p}$ remains feasible, that is, $\bar{p} = \min\{m_{q+1} - \Delta_-^k, p_{hk}^* - \Delta_-^k, \bar{u}\}$. Set $\bar{v} = \bar{v}+1$, $\bar{m}_{\bar{v}} = \bar{p}$, if $\bar{p} \neq p_{hk}^* - \Delta_-^k$ then $q = q+1$, and repeat step 1).

2) If case (b) is not verified goto step 3), otherwise set $\bar{z}^{\bar{v}}(p) = f^{k+1}(p) + z^q(p_{hk}^*)$. Compute the maximum value of $\bar{p}$ such that p_{hk}^* remains inside the feasible interval and $\bar{p}$ remains feasible, that is, $\bar{p} = \min\{p_{hk}^* + \Delta_+^k, \bar{u}\}$. Set $\bar{v} = \bar{v}+1$, $\bar{m}_{\bar{v}} = \bar{p}$ and goto step 3).

3) If $\bar{p} = \bar{u}$ then terminate, otherwise set $\bar{z}^{\bar{v}}(p) = f^{k+1}(p) + z^q(p - \Delta_+^k)$. Compute the

maximum value of $\bar{p}$ such that $p^*_k(\bar{p}) = \bar{p} - \Delta^k_+$ remains in the q-th interval and $\bar{p}$ remains feasible, that is, $\bar{p} = \min\{m_{q+1} + \Delta^k_+, \bar{u}\}$. Set $\bar{v} = \bar{v} + 1$, $\bar{m}_{\bar{v}} = \bar{p}$, $q = q + 1$ and repeat step 3).

Clearly, the total number of intervals for $z_{h(k+1)}$ is at most equal to that for z_{hk} plus the two ones corresponding to p^*_{hk} "entering" and "leaving" the feasible set, that is, the former interval q in step 2) is replaced by at most 3 new intervals. Note that the intervals with right extreme less than or equal to $p^*_{hk} - \Delta^k_-$, if any, correspond to intervals for z_{hk} "shifted left" by Δ^k_-, while the intervals with left extreme greater than or equal to $p^*_{hk} + \Delta^k_+$, if any, correspond to intervals for z_{hk} "shifted right" by Δ^k_+. The final number of intervals may well be strictly less than $v + 3$. Also, since each z^i is composed of the sum of at most $k - h + 1$ original functions f^t, each $\bar{z}^j$ is composed of the sum of at most $k - h + 2 = (k + 1) - h + 1$ original functions f^t. This completes the proof of Proposition 1.

During the above process, it is very easy to compute not only $p^*_{h(k+1)}$, but also the optimal solution of

$$\min\{\ z_{h(k+1)}(p)\ :\ p \in [l^{k+1}, \bar{u}^{k+1}]\ \}\ ,$$

that is, the last component of the optimal solution of $(ED_{h(k+1)})$, where constraint (13) is imposed. Thus, the above procedure can be used to solve (ED_{hk}) problems.

The complexity of the procedure depends on the actual form of the functions f^t; if the functions are quadratic, as common in practical applications, each step of the procedure is $O(1)$. Therefore, assuming that $(ED_{h(k-1)})$ has already been solved (with the same method), the complexity to solve (ED_{hk}) is $O(k - h)$, and, consequently, the complexity to solve all the problems (ED_{hh}), $(ED_{h(h+1)})$, $\ldots$, (ED_{hk}) is $O((k - h)^2)$. Hence, solving all the $O(n^2)$ (ED) problems in the dynamic programming procedure of the previous paragraph has $O(n^3)$ complexity. All in all, combining the special visit of the state-space graph G with the above efficient procedure for solving (ED_{hk}), we can solve (1UC), for the quadratic case, in $O(n^3)$. The same complexity bound holds for any other class of convex functions closed under the sum operation and such that an $O(1)$ closed-form formula exists for computing the unconstrained minima, such as, among others, polynomial functions of degree at most five. The approach is however likely to prove efficient even for other classes of functions, since univariate unconstrained optimization approaches can be used to compute the required unconstrained minima. This is likely to be much more efficient than the corresponding multivariate constrained optimization approaches required to solve (ED_{hk}) as a whole.

Although the optimal objective function value of each (ED_{hk}) problem is all that is needed for solving (1UC), the optimal solutions are then required to reconstruct a full optimal solution—both in the commitment variables and in the power variables—of the problem. More specifically, the optimal solutions of all the (ED_{hk}) problems corresponding to all nodes in the optimal path are needed. However, those solutions are easily found with a "backward pass", using the available information constructed in the "forward pass". In fact, consider a given problem (ED_{hk}). As discussed above, the optimal value of the last variable p_k, say $\bar{p}_k$, is available when the problem is solved. Then, it is immediate to compute the optimal value $\bar{p}_{k-1}$ of the previous variable p_{k-1} (if $k > h$) by just computing the projection of the available constrained minimum $p^*_{h(k-1)}$ (of $z_{h(k-1)}$, over $[l^{k-1}, u^{k-1}]$) onto $[\bar{p}_k - \Delta^{k-1}_+, \bar{p}_k + \Delta^{k-1}_-]$. Iterating this procedure, the whole solution of (ED_{hk}) can clearly be found in $O(k-h)$. Since only the optimal solution of the relevant (ED_{hk}) problems—those corresponding to nodes in the optimal path—is required, and the total number of time instants in which the unit is committed, in those nodes, is at most n, the optimal solution to (1UC), in terms of the power variables, can be found in $O(n)$.

It is easy to see that the dual optimal solution to each (ED_{hk}) can also be constructed, during the "backward pass", together with the optimal primal solution. In fact, each $\bar{p}_t$ for $h \leq t \leq k$ is the constrained minimum of z_{ht} subject to (9) – (13), that ultimately define a nonempty interval in the real line. Thus, if $\bar{p}_t$ lies in the strict interior of the interval, i.e., none of the constraints is active, then all the corresponding optimal dual variables are zero. Assume instead that exactly one constraint, say (11), is active in $\bar{p}_t$ and that z_{ht} is differentiable in $\bar{p}_t$ (the argument can be easily extended to the case of multiple active constraints). The Karush-Kuhn-Tucker conditions require that

$$z'_{ht}(\bar{p}_t) = \pi$$

where π is the optimal dual variable of (11). Similar formulae can be easily derived for all other constraints. Hence optimal dual information is readily available at the only cost of computing the derivative of z_{ht}. In the quadratic case, where this is $O(1)$, the total cost of retrieving the dual optimal solution to (ED_{hk}) is $O(k-h)$. Clearly, the above technique can be extended to a nondifferentiable z_{ht}; only, left and right derivatives are to be computed.

Somewhat surprisingly, it does not appear that the procedure can be significantly streamlined or simplified if further assumptions are made on the data. For instance, in many practical applications one has $l^t = l$, $u^t = u$, $\Delta^t_- = \Delta^t_+ = \Delta < u - l$ (for if $\Delta \geq u - l$ then ramping constraints are redundant), $\bar{l}^t = \bar{u}^t = l + \Delta$ for all $t \in T$, and $f^t(p) = ap^2 + b_t p$ (that is, only the linear part of the quadratic objective function depends on the time instant). Yet, it does not appear that the worst-case complexity results can be improved even if all the above assumptions are made.

However, it is possible to improve the performances of the method in practice by avoiding to (forming and) visiting all the state-space graph G of the dynamic programming procedure. This can be done by observing that every arc and node in G represents a certain number of (consecutive) time instants in T, and each $s - d$ path in G ultimately represents exactly n time instants. Thus, adding to the cost of each arc and node a quantity proportional to the number of time instants it covers, say M times the number of time instants where M is the same for all arcs and nodes, the cost of every $s - d$ path increases by Mn, and therefore the optimal solution does not change. Actually, one may even define a different value M_t for each $t \in T$ and add it to each node/arc that contains t. This allows us to make the cost of every arc and node in G nonnegative by simply choosing M large enough. In typical applications we do not even need to compute the actual cost of every arc and node for being able to compute a suitable value for M; in fact, only the costs of the nodes can be negative, hence, computing $z^*_t = \min\{f^t(p) \ : \ p \in [l^t, u^t]\}$ (cf. (7)) and setting $M = -\min\{\min\{\ z^*_t \ : \ t \in T\ \}, 0\}$ one ensures that all the resulting node (and arc) costs are nonnegative.

Then, knowing the objective function value of one—hopefully, good—solution, that is, the cost of one $s - d$ path, it may be possible to early terminate the visit of some part of the graph, avoiding to generate some of its nodes and the corresponding arcs. In fact, having all arc and node costs made nonnegative, the cost of any partial $s - d$ path cannot be smaller than the cost of any $s-d$ path containing it. Thus, if a partial path is found whose cost is larger than that of the best known solution, the visit of the graph from its last node can be interrupted. In a Lagrangian setting, a reasonable choice for the initial incumbent $s-d$ path could be the optimal solution of the (1UC) problem corresponding to the same unit at the previous Lagrangian iteration. Of course, as soon as a better $s-d$ path is found during the visit, the value of the incumbent can be updated.

5 Computational results

In order to test the actual efficiency of the proposed approach, we implemented it and compared it with the CPLEX 8.0 general-purpose Mixed-Integer Quadratic Programming solver. We remark that our code could surely be improved, for instance by the preprocessing techniques described at the end of the previous paragraph; however, even this somewhat preliminary implementation already obtained satisfactory results.

In order to test the approach, we considered three (UC) problems, each with the same 100 thermal units but with different values of n: 24, 96, and 168. We solved the problems with the Lagrangian approach of [6], that, in all three cases, performed 23 iterations to reach the optimal solution of the Lagrangian dual with respect to demand constraints and spinning-reserve constraints. We recorded the dual prices of these constraints at four particular iterations, that can be considered a representative sample: 1, 12, 16 and 23, that is, the first one, the last one and two intermediate iterations. This gave us a set of $100 \times 3 \times 4 = 1200$ (1UC) problems, each one corresponding to a specific unit, a given time horizon and one of the iterations of the Lagrangian approach; we solved all these instances both with our implementation of the proposed approach and with CPLEX 8.0 on a Pentium IV 2.5 GHz processor with 1.5Gb of RAM under Debian Linux 3.0. Our code was compiled with g++ version 3.0.4 and optimization option -O3; CPLEX was given a maximum time limit of 300 seconds to solve each instance.

The results of this computational experience are summarized in Table 1. Each row is associated with the 100 instances of a particular Lagrangian iteration and time horizon. In the first two columns we report the average solution time and the relative standard deviation obtained with the dynamic programming approach. In the following two columns we report the same information for the CPLEX 8.0 MIQP solver; since the latter was not always able to solve the instance within its time limit, in the last two columns we also report the average gap at termination (zero when not shown) and the number of instances that could not be solved by CPLEX at optimality.

The dynamic programming algorithm solves all instances, on average, much faster than CPLEX (almost three orders of magnitude faster on the largest instances), with a negligible standard deviation. For the largest instances, CPLEX was not able to solve a significant fraction of the instances to optimality within the time limit; the average gap of 1% on these instances means that for some unsolved instance the gap was greater than 8%. Also, CPLEX shows a very high standard deviation, meaning that while some instances were solved relatively fast (mainly due to a good preprocessing phase), others took very long. All in all, these results show that while implementing a Lagrangian approach to ramp-constrained (UC) by solving the (1UC) subproblems with CPLEX is hardly feasible, the proposed dynamic

instance		DP		CPLEX			
n	iter.	time	st.dev.	time	st.dev.	gap%	fail
24	1	.001	3e-3	0.05	0.05		0
	12	.002	4e-3	0.08	0.05		0
	16	.002	4e-3	0.08	0.05		0
	23	.002	4e-3	0.08	0.05		0
96	1	0.04	2e-3	10.74	41.99		1
	12	0.04	3e-3	17.57	50.93	0.06	2
	16	0.04	2e-3	32.64	76.87	0.02	6
	23	0.04	3e-3	32.21	76.12	0.03	6
168	1	0.20	6e-3	47.73	103.68	1.09	13
	12	0.20	6e-3	117.94	142.61	1.20	35
	16	0.20	5e-3	117.49	142.11	0.50	35
	23	0.20	6e-3	117.46	141.87	1.23	35

Table 1: Computational results of the DP algorithm versus the CPLEX MIQP solver

programming algorithm can solve the subproblems efficiently enough.

6 Conclusions

We have proposed an efficient dynamic programming algorithm for solving (1UC) with ramping constraints and general convex cost functions. The algorithm requires to solve $O(n^2)$ convex programs, with up to n variables each, in order to compute the data for the dynamic programming procedure; the main contribution of the paper is precisely the proposal of a new efficient algorithm for solving these problems. The resulting algorithm is simple to implement and works for a very general form of (1UC) with time-varying upper and lower limits over the generated power, as well as time-varying and different limits for ramp-up and ramp-down constraints. Coupled with a special visit of the state-space graph in the dynamic programming algorithm, this enables one to solve (1UC) in $O(n^3)$ overall for suitable cost functions, such as quadratic ones. Computational results showed that even a preliminary implementation of the proposed approach is indeed efficient and it clearly outperforms the MIQP solver of CPLEX 8.0 for solving (1UC).

It is worth noting that the proposed approach can be extended to more general versions of (1UC) as well:

- **Data dependent on the history of the unit.** It is easy to see that the approach immediately extends, with almost no change, to problems where the data of (ED_{hk})—(coefficient of the) cost functions, coefficients of the ramping constraints, maximum and minimum production levels—depend not only on t, but on h as well, that is, on how long the unit has been committed. This may be useful, e.g., to exploit better data fitting for the coefficients of the cost functions, in order to more accurately reflect the true operational cost of the unit. Note that a "monolithic" Integer NonLinear Programming model implementing this feature would be significantly larger than (1)

– (5), and therefore significantly more difficult to solve by standard means, while our approach handles this generalization almost for free.

- **Different discretization intervals for commitment and power variables.** In some cases, one may want to use different—typically, finer—discretization intervals for power variables than for commitment decisions. This may be due either to specific regulations of the operating context, or to the need to better reflect the operating characteristics of the unit. It is easy to see that our approach can be easily extended to handle this case as well; the total complexity becomes $O(m^2n)$, where $m(\geq n)$ is the number of power variables.

Acknowledgements

We are grateful to Fabrizio Lacalandra for useful discussions on these topics.

References

[1] L. Bacaud, C. Lemaréchal, A. Renaud, and C. Sagastizábal. Bundle Methods in Stochastic Optimal Power Management: A Disaggregated Approach Using Preconditioners. *Computational Optimization and Applications*, 20:227–244, 2001.

[2] C.H. Bannister and R.J. Kaye. A rapid method for optimization of linear systems with storage. *Operations Research*, 39(2):220–232, 1991.

[3] T.E. Bechert and H.G. Kwatny. On the optimal dynamic dispatch of real power. *IEEE Transactions on Power Apparatus and Systems*, PAS-91(1):889–898, 1972.

[4] A. Belloni, A. Diniz, M.E. Maceira, and C. Sagastizábal. Bundle relaxation and primal recovery in unit commitment problems. The brazilian case. *Annals of Operations Research*, 120:21–44, 2003.

[5] A. Borghetti, A. Frangioni, F. Lacalandra, A. Lodi, S. Martello, C.A. Nucci, and A. Trebbi. Lagrangian Relaxation and Tabu Search Approaches for the Unit Commitment Problem. In Saraiva, J.T. and Matos, M.A., editors, *Proceedings IEEE 2001 Power Tech Porto Conference*, 2001. Paper n. PSO5-397.

[6] A. Borghetti, A. Frangioni, F. Lacalandra, and C.A. Nucci. Lagrangian Heuristics Based on Disaggregated Bundle Methods for Hydrothermal Unit Commitment. *IEEE Transactions on Power Systems*, 18:313–323, 2003.

[7] A. Borghetti, A. Frangioni, F. Lacalandra, C.A. Nucci, and P. Pelacchi. Using of a cost-based Unit Commitment algorithm to assist bidding strategy decisions. In Borghetti, A., Nucci, C.A., and Paolone, M., editors, *Proceedings IEEE 2003 Power Tech Bologna Conference*, 2003. Paper n. 547.

[8] W. Fan, X. Guan, and Q. Zhai. A new method for unit commitment with ramping constraints. *Electric Power Systems Research*, 62:215–224, 2002.

[9] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms* II—*Advanced Theory and Bundle Methods*, volume 306 of *Grundlehren Math. Wiss.* Springer-Verlag, New York, 1993.

[10] M. Madrigal and V.H. Quintana. An Interior-Point/Cutting-Plane Method to Solve Unit Commitment Problems. In *Proceedings IEEE-PES Power Industry Computer Applications Conference*, pages 179–185, 1999.

[11] D.L. Travers and R.J. Kaye. Dynamic dispatch by constructive dynamic programming. *IEEE Transactions on Power Systems*, 13:72–78, 1998.

[12] F. Zhuang and F.D. Galiana. Towards a more rigorous and practical unit commitment by Lagrangian relaxation. *IEEE Transactions on Power Systems*, 3:763–773, 1988.