Generalized unit commitment by the radar multiplier method

Cèsar Beltran Royo

Memòria presentada per a aspirar al grau de Doctor en Matemàtiques.
Dirigida pel Dr. F. Javier Heredia Cervera, Dept. d'Estadística i Investigació Operativa, Universitat Politècnica de Catalunya.

Barcelona, 3 de maig de 2001.

Agraïments i dedicatòria

El meu agraïment comença per la veneració d'aquells grans científics, la influència dels quals encara s'hi deixa notar. D'entre aquests, vull fer especial esment al filòsof francès René Descartes (1596-1650) i al seu 'Discurs del Mètode', tant ple de saviesa. I, per descomptat, mencionar al matemàtic d'origen italià, Joseph Louis Comte de Lagrange (1736-1813), que entre d'altres coses ens deixà els omnipresents multiplicadors que porten el seu nom. L'altre clàssic del camp de l'optimització a qui voldria retre homenatge és Geoge B. Dantzing, que l'any 1947, amb la invenció del mètode Símplex, va inaugurar una brillant i nova etapa al camp de l'optimització. Gràcies als meus professors he rebut una petita part d'aquest bagatge de coneixements i d'inspiració. D'entre aquests professors vull destacar Narcís Nabona Francisco, qui sempre ha estat al meu costat per ajudar-me i amb el qual tinc l'honor de col·laborar en el grup de recerca liderat per ell mateix.

Però, sens dubte, la persona més implicada i compromesa en la meva tesi ha estat el meu tutor i director, Francisco Javier Heredia Cervera. A més d'iniciar-me i guiar-me en el difícil món de la recerca i de facilitar-me els paquets dissenyats per ell mateix, NOXCB i MAPH, en Javier m'ha dedicat part del seu temps de forma molt generosa i constructiva. Per tot això li estaré sempre molt agraït.

Des de la Universitat Politècnica de Catalunya, concretament des del Departament d'Estadística i Investigació Operativa (EIO), he rebut sempre un gran suport i ajuda, tant per part del personal acadèmic com per part del no acadèmic. Per a tots ells el meu reconeixement. Al mateix temps, les ajudes econòmiques aportades pel CICYT al grup de recerca abans esmentat han estat també un altre suport institucional important.

Afortunada o desafortunadament la realització d'aquesta tesi ha estat perllongada, la qual cosa m'ha permès conèixer molts estudiants i companys al departament d'EIO. A tots ells els vull retre un amistós homenatge. També vull recordar, ací, l'ajuda i suport rebut dels meus companys de treball de l'IES 'La Bastida'.

Seria imperdonable no agrair als membres del tribunal la seva col·laboració, suggeriments i el preuat temps que m'han dedicat. També voldria que el meu agraïment es fes extensiu a totes les persones i institucions que sense voler hagi pogut passar per alt.

És per tot això que des del meu cor vull dedicar aquesta tesi a totes les persones abans esmentades, també a les persones que d'alguna manera han 'sofert' la realització d'aquesta tesi, i finalment als meus pares, germans i dona.

Abstract

This operations research thesis should be situated in the field of power production. The general objective of this work is to efficiently solve the Generalized Unit Commitment (GUC) problem by means of specialized software. The GUC problem generalizes the Unit Commitment (UC) problem by simultaneously solving the associated Optimal Power Flow (OPF) problem. There are many approaches to solve the UC and OPF problems separately, but approaches to solve them jointly, i.e. to solve the GUC problem, are quite scarce. One of these GUC solving approaches is due to professors Batut and Renaud [BR92], whose methodology has been taken as a starting point for the methodology presented herein.

This thesis report is structured as follows. Chapter 1 describes the state of the art of the UC and GUC problems. The formulation of the classical short-term power planning problems related to the GUC problem, namely the economic dispatching problem, the OPF problem, and the UC problem, are reviewed. Special attention is paid to the UC literature and to the traditional methods for solving the UC problem.

In chapter 2 we extend the OPF model developed by professors Heredia and Nabona [HN95] to obtain our GUC model. The variables used and the modeling of the thermal, hydraulic and transmission systems are introduced, as is the objective function.

Chapter 3 deals with the Variable Duplication (VD) method, which is used to decompose the GUC problem as an alternative to the Classical Lagrangian Relaxation (CLR) method. As its name implies, the VD method duplicates some variables so that the GUC problem can be conveniently decomposed into two subproblems of a well known nature: the hydrothermal distribution subproblem (an OPF problem) and the thermal management subproblem (a simplified UC problem). Furthermore, in chapter 3 dual bounds provided by the VD method or by the CLR methods are theoretically compared.

Throughout chapters 4, 5, and 6 our solution methodology, the Radar Multiplier (RM) method, is designed and tested. Three independent matters are studied: first, the auxiliary problem principle method, used by Batut and Renaud to treat the inseparable augmented Lagrangian, is compared with the block coordinate descent method from both theoretical and practical points of view. Second, the Radar Subgradient (RS) method, a new Lagrange multiplier updating method, is proposed and computationally compared with the classical subgradient method. And third, we study the local character of the optimizers computed by the Augmented Lagrangian Relaxation (ALR) method when solving the GUC problem. A heuristic to improve the local ALR optimizers is designed and tested.

Chapter 7 is devoted to our computational implementation of the RM method, the MACH code. First, the design of MACH is reviewed briefly and then its performance is tested by solving real-life large-scale UC and GUC instances.

Solutions computed using our VD formulation of the GUC problem are partially

primal feasible since they do not necessarily fulfill the spinning reserve constraints. In chapter 8 we study how to modify this GUC formulation with the aim of obtaining full primal feasible solutions. A successful test based on a simple UC problem is reported.

The conclusions, contributions of the thesis, and proposed further research can be found in chapter 9.

Regarding software and hardware specifications, we must point out that the MACH code was written using standard FORTRAN 77 and that all the computational results were obtained on a Sun Ultra2 2200 workstation. The data files used for the reported tests can be obtained on request by writing to cesar.beltran@upc.es.

Finally, there is a note on the three levels of this report. The impatient reader, after this introduction, can jump directly to the conclusions (chapter 9). A more curious reader can concentrate on certain chapters since they are sufficiently self-contained. Of course, the reader of the whole report will get the best perspective.

Keywords

Augmented Lagrangian, auxiliary problem principle, block coordinate descent, bundle method, dynamic programming, economic dispatch, generalized unit commitment, Lagrangian relaxation, multiplier method, optimal power flow, radar method, subgradient method, variable duplication.

Notation

Abbreviations

ALR	Augmented Lagrangian Relaxation.
APP	Auxiliary Problem Principle.
BCD	Block Coordinate Descent (method).
CLR	Classical Lagrangian Relaxation.
DPF	Decremental spinning reserve Primal Feasible (optimizer).
DSR	Decremental Spinning Reserve.
ED	Economic Dispatching (problem).
FDP	Forward Dynamic Programming.
FPF	Full Primal Feasible (optimizer).
GUC	Generalized Unit Commitment (problem).
HTE	Hydrothermal Extended (network).
HTTE	Hydrothermal Transmission Extended (network).
IPF	Incremental spinning reserve Primal Feasible (optimizer).
ISR	Incremental Spinning Reserve.
LPF	Load Primal Feasible (optimizer).
LR	Lagrangian Relaxation.
MACH	*Modelo Acoplado de Coordinación Hidrotérmica*, i.e., Coupled Model for Hydrothermal Coordination.
MAPH	*Modelo Acoplado de Planificación Hidrotérmica*, i.e., Coupled Model for Hydrothermal Planning.
NNLSC	Nonlinear Network flow problem with Linear Side Constraints.
NOXCB	Nonlinear network flow with linear side constraints code.
OPF	Optimal Power Flow (problem).
RG	Radar Gradient (method).

RM	Radar Multiplier (method).
RS	Radar Subgradient (method).
SG	Subgradient (method).
SM	Subgradient Multiplier (method).
STHC	Short-Term Hydrothermal Coordination (problem).
UC	Unit Commitment (problem).
VD	Variable Duplication (method).

Symbols

$\nabla f(x_n)$:	gradient of $f(x)$ at x_n.
$\partial f(x_n)$:	subgradient of $f(x)$ at x_n.
A' :	transposed matrix.
α_n :	step length for the subgradient method. for unit t.
b_n :	auxiliary penalty parameter used within the APP method.
β_n :	dual step length for the RS method, i.e. $\lambda_{n+1} = \lambda_n + \beta_n \cdot s_n$.
c_{bt} :	basic thermal cost (included in the management cost $MC(p)$).
c_{lt} :	linear coefficient of the thermal cost $TC(p)$.
c_n :	penalty parameter of the augmented Lagrangian at iteration n.
c_{q_t} :	quadratic coefficient of the thermal cost $TC(p)$.
$C_{htd}(x)$:	cost associated with the hydrothermal distribution subproblem.
$C_m(x)$:	management cost associated with the thermal subproblem.
C_{off_t} :	shut-down cost for unit t.
C_{on_t} :	start-up cost for unit t.
$\mathrm{Co}(X)$:	convex hull of the set X.
$DC(q)$:	distribution cost function.
$D_{_{ED}}$:	economic dispatch domain.

D_{htd} :	hydrothermal distribution domain.
D_m :	management domain.
D_{OPF} :	optimal power flow domain.
d_r^i :	water discharge of reservoir r at interval i.
D_{UC} :	unit commitment domain.
$GC(p,q)$:	global cost function.
g_n :	gradient vector at iteration n.
g_r^i :	hydro-generation of the reservoir r during interval i.
$h_r(d_r^i, v_r^{i-1}, v_r^i, s_r^i)$:	hydro-generation function for reservoir r.
i :	index for intervals.
l :	index for electrical lines.
λ :	vector of Lagrange multipliers for the variable duplication constraint $x - \tilde{x} = 0$.
$L(x,\lambda)$:	classical Lagrangian function.
$L_c(x,\lambda)$:	augmented Lagrangian function.
$MC(p)$:	management cost function.
min_{off_t} :	minimum down time for the unit t.
$min_{on t}$:	minimum up time for the unit t.
n :	index for multiplier updating iterations.
n_g :	number of busses with hydro-generation.
n_i :	number of intervals of the whole period.
n_l :	number of lines in the transmission network.
n_o :	number of basic loops in the transmission network.
n_r :	number of reservoirs.
n_t :	number of thermal units.
$\underline{p}_t$:	lower bound for the power output of unit t.
$\overline{p}_t$:	upper bound for the power output of unit t.
p_t^i :	power generation of unit t during interval i.

$q(\lambda)$:	dual function associated with the classical Lagrangian.
$q_c(\lambda)$:	dual function associated with the augmented Lagrangian.
q_n :	dual function at λ_n.
q_l^i :	power flow through line l during interval i.
r :	index for reservoirs.
${r_D}_t^i$:	decremental spinning reserve of unit t at interval i.
${r_I}_t^i$:	incremental spinning reserve of unit t at interval i.
r_l :	resistance of line l.
s_n :	subgradient vector at iteration n.
SP_n :	supporting plane of the dual function at λ_n.
s_r^i :	water spillage of reservoir r at interval i.
t :	index for thermal units.
$TC(p)$:	thermal cost.
TP_n :	tangent plane of the dual function at λ_n.
$u'v$:	inner product.
v' :	transposed vector.
v_r^i :	water volume of reservoir r at the end of interval i.
$\widetilde{x}$:	copy of variable x within the VD method.
x_l :	reactance of line l.

Contents

Chapter 1

State of the art

The objective of this chapter is to determine the state of the art of the Unit Commitment (UC) problem and its generalizations. First of all, four related problems are presented: the Economic Dispatching (ED) problem, the UC problem, the Optimal Power Flow (OPF) problem and the Generalized Unit Commitment (GUC) problem. Second, we briefly examine the evolution of the different approaches and optimization techniques used to solve the GUC problem. And third, special attention is paid to the works that represent the starting point of this thesis.

1.1 Four related problems

1.1.1 The economic dispatching problem

The Economic Dispatching (ED) problem is one of the main problems in power system management. Given the generators committed to operating during each hour of a utility's planning horizon, the traditional ED problem determines the power output of each generator to meet predicted demand with minimum operating cost. Mathematically this problem can be formulated as

$$\left.\begin{array}{ll} \min & TC(p) \\ \\ s.t. & p \in D_{ED} \end{array}\right\} \tag{1.1}$$

where $TC(p)$ is a linear or quadratic function which stands for thermal cost and depends on the power output vector p. D_{ED} is the ED domain of feasible production policies that must satisfy customer demand, reserve constraints and the technical constraints of the thermal generators. If the generating system includes the hydraulic park, D_{ED} also accounts for the constraints that drive the hydraulic generating system (reservoir maximum and minimum volumes, coupling constraints of cascaded reservoirs, etc.). More details about the ED problem can be found in the book [Gro86].

1.1.2 The optimal power flow problem

The conventional Optimal Power Flow (OPF) problem (Alternate Current (AC) model) aims to minimize the operating cost of thermal resources, subject to satisfying constraints represented by bus active and reactive power balances in terms of voltages and phase angles. A simplified version of the OPF problem can be obtained using a Direct Current (DC) model instead of the AC model. In both cases, bulk power is moved from generation sites to the areas of use by transmission lines, that is, the transmission network. The technical limitations of this network, such as its capacity, must be fulfilled by the OPF solution. We could say that the OPF problem enhances the ED problem by adding the transmission network. Mathematically the OPF problem can be formulated as

$$\left.\begin{array}{ll} \min & TC(p) + DC(q) \\ & \\ s.t. & p \in D_{ED} \\ & q \in D_{OPF} \end{array}\right\} \tag{1.2}$$

where $DC(q)$, usually a linear or quadratic function, stands for distribution cost, which depends on the power flow vector q. D_{OPF} mainly accounts for the tranmission network. The book [Gro86] is a basic reference work for the OPF problem. For a review of selected OPF literature the reader is referred to [MEHA99a] and [MEHA99b].

1.1.3 The unit commitment problem

If the OPF problem enhances the ED problem by adding the transmission network, the Unit Commitment (UC) problem also enhances the ED problem by looking for the best generating schedule. Given a short period of time (one day to one week) divided into subperiods (typically one hour long), a unit schedule indicates those subperiods in which the unit is on or committed and the periods when the unit is off or uncommitted. Therefore, the UC problem aims to determine the lowest cost schedule of generating units within a power system subject to device and operating constraints. Mathematically the UC problem can be formulated as

$$\left.\begin{array}{ll} \min & TC(p) + MC(p) \\ & \\ s.t. & p \in D_{ED} \cap D_{UC} \end{array}\right\} \tag{1.3}$$

where $MC(p)$ is a non-continuous function which stands for management cost, that is, the start-up and shut-down costs, and depends on the power output vector p. D_{UC} represents the set of suitable p vectors which fulfill schedule constraints such as minimum up-time and down-time. Some good examples of literature about the UC problem are [MS83], [ZG88], and [VAIM89]. For a more exhaustive list the reader should consult the UC literature synopsis [SF94].

1.1.4 The generalized unit commitment problem

The OPF and UC problems enhance the ED problem in two different aspects: the transmission network and the schedule of the thermal units, respectively. For practical purposes this splitting of the generation-transmission problem may lead to solutions that are optimal for one of the two problems but not even feasible for the other one and vice-versa. However, the Generalized Unit Commitment (GUC) problem seeks the minimum operating cost under the simultaneously imposed assumptions and constraints of the UC and OPF problems. That is, we wish to schedule generator start-ups, shut-downs, and generation levels in each time period, in order to minimize production and start-up/shut-down costs while satisfying production, line flow constraints, reserve, generator minimum up/down-time and hydraulic system constraints. Mathematically the GUC problem can be formulated as

$$\left.\begin{array}{ll} \min & TC(p) + MC(p) + DC(q) \\ \\ s.t. & p \in D_{ED} \cap D_{UC} \\ & q \in D_{OPF} \end{array}\right\} \qquad (1.4)$$

One of the first authors to use the term GUC problem was Ross Baldick [Bal95]. In the abundant GUC literature we find other synonymous terms, such as 'generation scheduling optimization with transmission constraints' [BR92], 'Short-term Hydrothermal Coordination (STHC) problem' [BH99], and many others. In the research report [SPR+97] the GUC problem is amply analyzed and summarized in terms of modeling and solving techniques.

Other aspects that are being given increasingly close attention within the GUC problem are: voltage constraints [YS97, MS99], ramp rate constraints [Bal95, LB99], fuel and emission limits on thermal units [AaS98, YS97], scheduling of the hydraulic generators [GNLL97, NSS98], etc.

All the works mentioned above model the GUC problem in a deterministic framework. The concept of spinning reserve refers to power margin which can be used within a few minutes to cope with the most usual random disturbances. So far, relatively few works have modeled the GUC problem in a stochastic framework due to its great complexity and computational requirements (see for example [CS98] and [RCCC96]). With the new deregulated power market the stochasticity of the GUC problem is even greater: to the random disturbances of the regulated power market, companies will have to add uncertain load sales and energy prices. Thus, the whole concept of the GUC problem is being accommodated to the new deregulated market [KS96, DSC99].

A selection of UC and GUC literature, mainly from 'IEEE Transactions on power systems', is given below in Table 1.1, which displays the scope of the GUC model used in each work and other important features of the model . More information about the GUC problem can be found at the 'Unit Commitment's Archive' website (http://gear.kku.ac.th/~thana/unitcom.html).

Table 1.1: Selection of literature on UC and GUC problem modeling. *Th= Thermal, Hy= Hydraulic, TN= Transmission Network.*

Author	Reference	Systems			Other features
		Th	*Hy*	*TN*	
Al-agtash	[AaS98]	x	x		Emission constraints.
Alguacil	[AC00]	x	x	x	
Baldick	[Bal95]	x			Fuel and ramp constraints.
Batut	[BR92]	x		x	
Beltran	[BH99]	x	x		Decremental reserve.
Carøe	[CS98]	x	x		Stochastic load.
Ferreira	[Fer94]	x	x		
Greif	[GJL+98]	x			Decremental reserve.
Guan	[GNLL97]	x	x		Hydraulic scheduling.
Jimenez	[JC99]	x	x		
Lai	[LB99]	x			Ramp constraints.
Lauer	[LBSP82]	x			
Li	[LJS93]	x	x		
Li	[LHS+97]	x	x		Hydraulic scheduling.
Li	[LHS+98]	x	x		
Luh	[LZT98]	x	x		
Ma	[MS99]	x		x	Fuel and ramp constraints.
Merlin	[MS83]	x			
Nabona	[NR00]	x	x		
Ni	[NGL99]	x	x		Hydraulic scheduling.
Nilsson	[NS97]		x		Hydraulic scheduling.
Nilsson	[NSS98]		x		Hydraulic scheduling.
Pellegrino	[PRS96]	x	x		
Piria	[PT]	x	x	x	
Renaud	[RCCC96]	x			Stochastic load.
Ruzic	[RR98]	x	x		
Salam	[SNH97]	x	x		Transmission losses.
Salam	[SNH98]	x	x		Transmission losses.
Svoboda	[STLJ97]	x			Ramp constraints.
Tong	[TS90]	x	x		Fuel constraints.
Tseng	[TGS98]	x		x	
Virmani	[VAIM89]	x			
Wang	[WS93]	x	x	x	
Wang	[WS+95]	x		x	Emission constraints.
Yu	[YS97]	x			Emission and ramp constraints.
Zhuang	[ZG88]	x			

1.2 Difficulty of the generalized unit commitment problem

The Generalized Unit Commitment (GUC) problem is twofold. Firstly, we have a nonlinear problem consisting of binary and continuous variables. The binary variables 0-1 indicate whether a generator is active or not. The continuous variables indicate, for example, the electrical power that each generator must produce to fulfill all the requirements (demand, reserve, etc.). Secondly, we have to face large-scale problems that are difficult to solve in real time due to the vast number of variables involved. Therefore the GUC problem falls in the class of large-scale nonlinear mixed integer programming problems.

Due to the great difficulty of the GUC problem, initially the thermal generation system and the transmission system were optimized separately, giving rise to the Unit Commitment (UC) problem and the Optimal Power Flow (OPF) problem respectively. The drawback of this separate approach is that an optimal solution for the UC problem may not be optimal for the OPF problem or, what is worse, it may be OPF infeasible. This would be the case of a power generation schedule which produces a line saturation in the transmission network by exceeding the power limit (see section 1.1 for related bibliography).

Some authors consider a pseudo-joint optimization of the two systems, as for example [WS+95], in which the capacity constraints of the transmission network are added to the UC problem. However, they do not consider distribution losses when deciding on the optimal unit commitment. Neither do they include the hydraulic generating system in their model.

This pseudo-joint optimization of the two systems does not necessarily give an optimal solution to the GUC problem. A real joint optimization of the two systems, which theoretically would give an optimal solution, is first proposed in the remarkable paper [BR92] by means of the decomposition of the GUC problem into two subproblems: the UC and the OPF. These authors simultaneously solve a sequence of UC and OPF subproblems in a coordinated fashion. Eventually, after several iterations, the two sequences of optimal solutions obtained for the two sequences of subproblems converge to a solution, which is simultaneously optimal for both the thermal and the transmission systems. However, they do not consider the hydraulic generating system. Another joint optimization of the two systems can be found in [PT], which does consider the hydraulic generating system. However, the solutions initially obtained in this work are primal infeasible and some heuristic must be used in order to reach primal feasibility.

Because of the problem's size and complexity, and because of the major economic benefits that could result from its improved solution, considerable attention has been devoted to algorithm development. Heuristics such as 'priority lists' [Lee88] have long been used by industry; but in the last three decades, more systematic procedures based on a variety of algorithms have been proposed and tested. These techniques include dynamic programming [HSL+91], branch-and-bound mixed integer programming [HB86], linear and network programming approaches [HFS90], and Benders decomposition methods [AC00], among others. During the last decade meta-heuristic methods have been tested, such as genetic programming [OI98] and simulated annealing [ZG90], along with expert systems and neural networks [TSO91]. Recent improvements in integer programming codes, as for example the CPLEX 6.8 package, has opened up a promising way to improve the solution process of the GUC problem through branch-and-bound mixed integer programming [NR00]. For a broader overview of the optimization methods used to solve the UC and the GUC problem the reader can consult [SPR$^+$97] and [SF94].

Nevertheless, the solution approach that has been most successful, and which is most widely used at present, is Lagrangian relaxation (see Table 1.2). This procedure decomposes the problem by multiplying constraints that couple different generators (such as energy demand and reserve constraints) by Lagrangian multipliers and placing them in the objective function. Given a set of multiplier values, the problem is then separable in the generating units, and a dynamic program of low dimension can be used to obtain a trial schedule for each unit. A process of multiplier adjustment is used to search for feasible near-optimal solutions.

1.3 Specially related papers

Four papers are specially related with this thesis:

(1) [BR92] represents the formulation and methodology starting point. Following Batut and Renaud, we use variable duplication plus the augmented Lagrangian to formulate the Generalized Unit Commitment (GUC) problem. In this thesis, Batut and Renaud's paper is actually improved by: incorporating the hydraulic generating system, by incorporating the transmission network for each period, not only for the first period as in [BR92] (chapter 2), using a more efficient method to separate the augmented Lagrangian (chapter 4) and measuring and improving the quality of the computed (local) optimizer (chapter 6).

(2) [HN95] represents the Optimal Power Flow (OPF) starting point. In order to solve the GUC problem we decompose the GUC problem into a simplified Unit Commitment (UC) subproblem and an OPF subproblem. We adapt the model and software (MAPH package) developed by Heredia and Nabona [HN95] to solve our OPF subproblem. See chapters 2 and 7 for more details.

(3) One of the main contributions of this thesis is the Radar Subgradient (RS) method (chapter 5). The RS method updates the Lagrangian multipliers by computing an accurate step length along the subgradient direction. Lai and Baldick [LB99] use a similar multiplier updating method. Both methods follow the subgradient direction; however, Lai and Baldick's method is based on a second order approximation to the dual function whereas the RS method is based only on a first order approximation to the dual function.

(4) The main disadvantage of the Classical Lagrangian Relaxation (CLR) method

Table 1.2: Selection of literature on the solving techniques of the UC and GUC problems. *C= Classical Lagrangian Relaxation, A= Augmented Lagrangian Relaxation, S= Subgradient method, B= Bundle method, D= Dynamic Programming.*

Author	Reference	Optimization methods					
		C	*A*	*S*	*B*	*D*	*Other methods*
Al-agtash	[AaS98]		x	x		x	
Alguacil	[AC00]						Benders decomposition.
Baldick	[Bal95]		x			x	Variable duplication.
Batut	[BR92]		x			x	Variable duplication.
Beltran	[BH99]		x			x	Radar gradient method.
Caroe	[CS98]	x			x	x	Branch&Bound.
Ferreira	[Fer94]	x					
Greif	[GJL^{+}98]	x					
Guan	[GNLL97]	x		x		x	
Jimenez	[JC99]	x					Cutting plane method.
Lai	[LB99]	x				x	Baldick method.
Lauer	[LBSP82]					x	Branch&Bound.
Li	[LJS93]	x					
Li	[LHS^{+}97]	x				x	
Li	[LHS^{+}98]	x		x			
Luh	[LZT98]	x			x		
Ma	[MS99]						Benders decomposition.
Merlin	[MS83]	x		x		x	
Nabona	[NR00]						Mixed integer programming.
Ni	[NGL99]	x		x		x	
Nilsson	[NS97]	x		x		x	Variable duplication.
Nilsson	[NSS98]	x		x			Variable duplication.
Pellegrino	[PRS96]	x	x		x	x	Variable duplication.
Piria	[PT]	x			x	x	
Renaud	[RCCC96]		x			x	
Ruzic	[RR98]	x					Optimal distance method.
Salam	[SNH97]	x		x		x	Expert system.
Salam	[SNH98]	x		x		x	
Svoboda	[STLJ97]	x		x		x	
Tong	[TS90]	x		x			
Tseng	[TGS98]	x		x			
Virmani	[VAIM89]	x		x		x	
Wang	[WS93]	x					Expert system.
Wang	[WS^{+}95]		x			x	
Yu	[YS97]					x	
Zhuang	[ZG88]	x		x		x	

is the primal infeasibility of the computed solution. Several heuristic methods have been developed to reach primal feasibility. However, the Augmented Lagrangian Relaxation (ALR) method directly provides a primal feasible solution. Pellegrino, Renaud and Socroun [PRS96] solve the GUC problem by means of a two-phase arrangement: in the first phase the CLR method solves the dual of the GUC problem, thus providing a lower cost bound and a primal infeasible solution. In the second phase, starting from the former primal solution, the ALR method yields an optimal primal (feasible) solution. Therefore, with this arrangement no heuristic is needed to obtain primal feasibility. Our Radar Multiplier (RM) method follows the same strategy (CLR plus ALR method). Nevertheless, there is an important difference between the RM method and [PRS96]: we use the same formulation of the GUC problem in the two phases (only an augmentation term is incorporated into the Lagrangian), whereas in [PRS96] two different formulations of the GUC problem are solved (one in each phase). With our approach the optimal Lagrange multipliers of the first phase can be used as a starting point for the second phase; with their approach this is impossible due to the multiplier space used in the first phase is different from the multiplier space used in the second phase.

1.4 Summary

First of all the Economic Dispatching (ED) problem, the UC problem, the Optimal Power Flow (OPF) problem and, finally, the Generalized Unit Commitment (GUC) problem are defined and mathematically stated. During the last four decades, the GUC problem, a large-scale mixed integer non-linear problem, has been solved using many methods which range from heuristic to optimization based methods. Among them, Lagrangian relaxation is the most widespread procedure and therefore, special attention has been paid in this chapter to related literature. Particular emphasis has also been placed on the papers [BR92, HN95], as they represent the actual starting point of this thesis.

Chapter 2

Modeling the generalized unit commitment problem

The problem we are going to deal with is called the *Generalized Unit Commitment* (GUC) problem. The objective of this problem is to optimize electrical production and distribution, considering a short-term planning horizon (from one day to one week). Hydraulic and thermal plants must be coordinated in order to satisfy the customer demand of electricity at the minimum cost and with a reliable service.

The general expression of the GUC problem could be developed in several different ways. The approach adopted in this thesis follows the so called *coupled model.* In this chapter we give a brief description of the coupled model. The reader can find more details in [HN95] and [Her95].

This model takes into account the hydroelectric energy generation system with n_r reservoirs, as well as the thermal system, with n_t thermal units, and, optionally, the transmission network, with n_b busses and n_l transmission lines. The GUC problem can be expressed mathematically as

$$
\text{(GUC)}\begin{cases} \min & GC(p,q) := TC(p) + DC(q) + MC(p) \\[2ex] \text{s.t.} & (d, v, s, p, r_{_I}, g_{_I}, r_{_D}, g_{_D}, q, g) \in D_{htd} \\ & \hfill p \in D_m \end{cases}
$$

where GC stands for global cost, TC for thermal cost, DC for distribution cost, MC for management cost, D_{htd} represents the *hydrothermal distribution* domain, and D_m the *management* domain. The meaning of these two domains will be explained soon in sections 2.2 and 2.3. The decision variables are: d, v, s, p, $r_{_I}$, $g_{_I}$, $r_{_D}$, $g_{_D}$, q, g, and will be described in the next section.

2.1 Variables

The variables of the GUC problem can be divided into *hydro variables*, *thermal variables* and *distribution variables.*

2.1.1 Hydro variables

These are the variables that are related with the hydroelectric system, describing the state of each reservoir r at time interval i: volume at the end of the interval v_r^i, discharges d_r^i and spillages s_r^i ($r = 1, \dots, n_r$, $i = 1, \dots, n_i$). All these variables are non-negative. The volumes and discharges are upper bounded and volumes can also have lower bounds. It is well known [Ros81] that an appropriate representation of the evolution of the state of all the reservoirs in the hydraulic basin throughout the period of study is provided by the so called *replicated network*, a network in which each arc is associated with one hydro variable [HN95].

2.1.2 Thermal variables

In the coupled model of the GUC problem the fundamental magnitudes which describe the state of each thermal unit t at each time interval i are its power output p_t^i and the *incremental* and *decremental spinning reserve* ($\tilde{r}_{It}^{\,i}$, $\tilde{r}_{Dt}^{\,i}$).

Each thermal unit can be in one of two different states: 'on' and 'off'. When thermal unit t is on, it is connected to the transmission network and its power production is delivered to the system. When it is off, it no longer provides electric energy to the system. These two different states are represented in the model by the domain of definition of variable p_t^i. When the thermal unit t is off at interval i, then $p_t^i = 0$, and when it is on, the power output of the unit must be within its operation limits $\underline{p}_t^i \leq p_t^i \leq \bar{p}_t^i$. That is,

$$p_t^i \in \{0\} \cup [\underline{p}_t^i, \bar{p}_t^i]. \tag{2.1}$$

Given that we can assume $\underline{p}_t^i > 0$, we will say the thermal unit is on if $p_t^i > 0$ and off if $p_t^i = 0$.

The *incremental* and *decremental spinning reserve* ($\tilde{r}_{It}^{\,i}$, $\tilde{r}_{Dt}^{\,i}$) are defined as

$$\begin{aligned} \tilde{r}_{It}^{\,i} &:= \min\{\bar{r}_{It}, \bar{p}_t^i - p_t^i\} && (2.2) \\ \tilde{r}_{Dt}^{\,i} &:= \min\{\bar{r}_{Dt}, p_t^i - \underline{p}_t^i\} && (2.3) \end{aligned}$$

for $t = 1, \dots, n_t$ and $i = 1, \dots, n_i$, where $\bar{r}_{It}$ and $\bar{r}_{Dt}$ are, respectively, the maximum allowed value of the incremental and decremental spinning reserve [HN95]. The above relations define the value of the spinning reserve as a non-differentiable function of the thermal output p_t^i. To overcome this difficulty, equations (2.2) and (2.3) are relaxed. Variables $\tilde{r}_{It}^{\,i}$ and $\tilde{r}_{Dt}^{\,i}$ are substituted in the coupled model of GUC (see

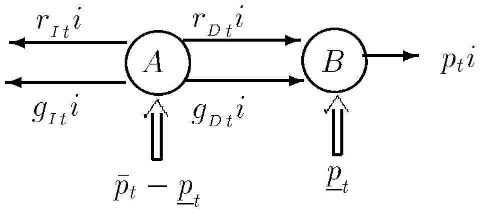

Figure 2.1: Equivalent thermal network

[HN95]) by variables r_{It}^i and r_{Dt}^i defined as $r_{It}^i \leq \tilde{r}_{It}^i$ and $r_{Dt}^i \leq \tilde{r}_{Dt}^i$. Doing so and including the slack variables g_{It}^i and g_{Dt}^i , equation (2.2) can be substituted by

$$\begin{aligned} r_{It}^i + g_{It}^i &= \bar{p}_t^i - p_t^i && (2.4)\\ 0 \leq r_{It}^i &\leq \bar{r}_{It} && (2.5)\\ 0 \leq g_{It}^i, & && (2.6)\end{aligned}$$

and (2.3) by

$$\begin{aligned} r_{Dt}^i + g_{Dt}^i &= p_t^i - \underline{p}_t^i && (2.7)\\ 0 \leq r_{Dt}^i &\leq \bar{r}_{Dt} && (2.8)\\ 0 \leq g_{Dt}^i. & && (2.9)\end{aligned}$$

The slack variables g_{It}^i and g_{Dt}^i are called, respectively, the *incremental* and *decremental gap*. From equation (2.7) we obtain

$$p_t^i = r_{Dt}^i + g_{Dt}^i + \underline{p}_t^i. \qquad (2.10)$$

Substituting this last equation in (2.4) we obtain that equations (2.4) to (2.9) can be replaced by the following set of equations:

$$\begin{aligned} r_{It}^i + g_{It}^i + r_{Dt}^i + g_{Dt}^i &= \bar{p}_t^i - \underline{p}_t^i && (2.11)\\ p_t^i - r_{Dt}^i - g_{Dt}^i &= \underline{p}_t^i && (2.12)\\ 0 \leq r_{It}^i &\leq \bar{r}_{It} && (2.13)\\ 0 \leq g_{It}^i & && (2.14)\\ 0 \leq r_{Dt}^i &\leq \bar{r}_{Dt} && (2.15)\\ 0 \leq g_{Dt}^i. & && (2.16)\end{aligned}$$

This set of equations can be interpreted as the network equations of the graph shown in Figure 2.1. Equation (2.11) corresponds to the balance equation of node A and equation (2.12) is the balance equation of node B. This graph is called the *Equivalent Thermal Network* of thermal unit t at interval i. There is one equivalent thermal network for each thermal unit at each interval.

2.1.3 Distribution variables

If a direct current approach to the transmission network is considered, as it is in our case, this last set of variables includes the power transmission through the transmission line l at each time interval, q_l^i, $l = 1, \ldots, n_l$, and the hydro generation variables

g_j^i, $j = 1, \ldots, n_g$. Variables q_l^i must be within the upper limit imposed by the capacity of the transmission line. The hydro generation variables g_j^i are associated with arcs that feed the n_g hydro generation busses of the transmission network, that is, busses in which the hydro generation produced by the reservoirs are injected into the transmission network. The distribution variables are usually interpreted as the arcs of a directed graph called the *Transmission Network* (see Carvalho et al. [CS87]).

2.2 The hydrothermal distribution domain

In the coupled model the constraints relating the hydro, thermal and distribution variables are expressed through a *network flow model with side constraints.* The general expression of the constraints defining the hydrothermal distribution domain D_{htd} is

$$\begin{aligned}
A_{HTTE} \cdot x &= b_{HTTE} && (2.17)\\
h_G(d, v, g) &= 0 && (2.18)\\
T_{ISR} \cdot \begin{pmatrix} r_I \\ g \end{pmatrix} &\geq b_{ISR} && (2.19)\\
T_{DSR} \cdot \begin{pmatrix} r_D \\ g \end{pmatrix} &\geq b_{DSR} && (2.20)\\
T_{KVL} \cdot q &= 0 && (2.21)\\
\underline{x} \leq x &\leq \bar{x} && (2.22)
\end{aligned}$$

where:

Eq. (2.17) expresses the network constraints associated with the so called *Hydrothermal Transmission Extended (HTTE) network.* The HTTE network integrates the *replicated hydro network*, which accounts for the time and space coupling among the reservoirs of the river basin, the *equivalent thermal network*, which defines the relation between the power output and the spinning reserve level of each thermal unit, and the *transmission network*, which formulates the conservation of the power flow at the busses of the transmission system. More details about the HTTE network can be found in [HN95].

Eq. (2.18) expresses nonlinear side constraints which define the injection of the hydroelectric generation (a nonlinear function of the hydro variables d and v) into the appropriate busses of the transmission network. Let us define $\mathcal{R}_j$ as the set of indices of the reservoirs that feeds its power generation to the j-th hydro generation bus. Then, the nonlinear constraints (2.18) can be expressed as

$$\sum_{r \in \mathcal{R}_j} h_r(d_r^i, v_r^{i-1}, v_r^i, s_r^i) = g_j^i \quad j = 1, \ldots, n_g \, , \; i = 1, \ldots, n_i \qquad (2.23)$$

and therefore, there are $n_g \times n_i$ nonlinear constraints. The hydro generation

$$h_r(d_r^i, v_r^{i-1}, v_r^i, s_r^i)$$

is a nonlinear function which gives the power generated by the hydro station r at interval i, and in the model used in this work has been formulated as a sixth order polynomial function, following the description presented in [HN95].

Eq. (2.19), (2.20). These two sets of linear side constraints impose the satisfaction of the incremental and decremental spinning reserve requirements of the whole system. The summation of the incremental spinning reserve for all the thermal and hydro units must be no less than the required system incremental spinning reserve. The same requirement must be satisfied by the decremental spinning reserve. Then, the linear side constraints expressing these two conditions are

$$\sum_{t=1}^{n_t} {r_{It}}^i + \sum_{r=1}^{n_r} \left(\bar{g}_r^i - g_r^i\right) \geq {R_I}^i \quad i = 1, \ldots, n_i \tag{2.24}$$

$$\sum_{t=1}^{n_t} {r_{Dt}}^i + \sum_{r=1}^{n_r} g_r^i \geq {R_D}^i \quad i = 1, \ldots, n_i \tag{2.25}$$

where $\bar{g}_r^i$ is the maximum value of the r-th hydro generation injection.

Eq. (2.21). This last set of linear side constraints constitutes the formulation of the Kirchoff Voltage (KV) law. These constraints, together with the power flow conservation equations formulated in equations (2.17), represent a direct current approach to the transmission network (see [CS87]). If the (per unit) reactance of a given transmission line l is denoted by x_l, then the voltage fall along this transmission line at interval i will be $x_l q_l^i$. Each transmission network has a number n_o of basic loops which is a characteristic of the network. If we denote by $\mathcal{L}_j$ the set of lines belonging to the j-th basic loop, then the KV law can be expressed through the following set of linear constraints:

$$\sum_{l \in \mathcal{L}_j} x_l q_l^i = 0 \quad j = 1, \ldots, n_o\,,\ i = 1, \ldots, n_i. \tag{2.26}$$

Eq. (2.22). These are the upper and lower bounds to the variables mentioned in section 2.1.

In the GUC problem, a fundamental constraint to be satisfied is that the total load demand must be satisfied at each time interval. This constraint is guaranteed implicitly by the conservation of flow (power) at the load busses of the transmission network. The formulation of the domain D_{htd} as a network flow problem with side constraints allows the use of specialized network optimization codes. Also, the flexibility of this model is such that other relevant system constraints can be easily added, as, for instance, multiple spinning reserve constraints, energy coupling constraints, energy exchange contracts, or security constraints and emission constraints [CNH95].

2.3 The management domain

The management domain D_m takes into account the disconnected domain of variables p_t^i, that is,

$$p_t^i \in \{0\} \cup [\underline{p}_t^i, \bar{p}_t^i], \tag{2.27}$$

whereas in D_{htd}, variables p_t^i are allowed to oscillate between 0 and the upper bound $\bar{p}_t^i$ in order to have a connected domain D_{htd}.

Domain D_m also describes the restrictions to the start-up and shut-down process of the thermal generators. When a thermal generator t has been shut down, then for mechanical reasons it must be off for at least a given time min_{off_t} (*minimum down time*). Equivalently, when a thermal generator t is started up, it must be on for at least a given time $min_{on t}$ (*minimum up time*). Mathematically,

$$\text{if } p_t^{i-1} = 0 \text{ and } p_t^i > 0 \quad \text{then} \quad p_t^j > 0\,(j = i, \ldots, i + min_{on t} - 1), \tag{2.28}$$
$$\text{if } p_t^{i-1} > 0 \text{ and } p_t^i = 0 \quad \text{then} \quad p_t^j = 0\,(j = i, \ldots, i + min_{off_t} - 1). \tag{2.29}$$

Therefore D_m is defined by equations (2.27 - 2.29). These types of constraints, specially the disconnected domain, are difficult to handle with ready to use optimization packages. So far, dynamic programming is the most common method used to deal with them, as we will see in chapter 3.

2.4 Objective function

The aim of the GUC problem is to minimize the global cost $GC(p, q)$ of generating the required electrical power under system constraints. This global cost splits into three subcosts: thermal, distribution and management costs.

2.4.1 Thermal and distribution costs

The expression of the thermal and distribution costs in the coupled model of the GUC problem is

$$TC(p) + DC(q). \tag{2.30}$$

The thermal term $TC(p)$ represents the cost of the fuel consumption of the thermal units, and it is modeled as a quadratic function of the power output of each thermal unit

$$TC(p) := \sum_{i=1}^{n_i} \sum_{t=1}^{n_t} c_{lt} p_t^i + c_{q_t} (p_t^i)^2 \tag{2.31}$$

where c_{q_t} and c_{lt} stand for quadratic and linear coefficients of the thermal cost. Note that the usual constant term c_{b_t} of the thermal cost will be included in the management cost (2.34) for algorithmic purposes. The reason for this is that c_{bt}

must be added if and only if the thermal unit t is on, and it is precisely within the computing of the management cost that we decide the state of each thermal unit.

If the transmission network is considered, a *distribution cost* $DC(q)$ could also be included as an estimation of the cost of the power losses. This loss function is modeled as a quadratic function of the power q transported by the transmission lines. If r_l is the (per unit) resistance of the transmission line l and π^i the cost of one MW lost, then

$$DC(q) := \sum_{i=1}^{n_i} \pi^i \Big(\sum_{l=1}^{n_l} r_l (q_l^i)^2 \Big). \tag{2.32}$$

2.4.2 Management cost

The term $MC(p)$ of the objective function represents the *management cost* of operating the n_t thermal generators. It has been established before that the domain of variables p_t^i is $\{0\} \cup [\underline{p}_t^i, \bar{p}_t^i]$. When $p_t^i = 0$, we say that the thermal unit is 'off', that is, the thermal generator has been turned off, and we consider, as a simplification, that it no longer provides energy to the system. When $p_t^i \in [\underline{p}_t^i, \bar{p}_t^i]$ the thermal generator is turned on and connected again to the transmission network. The act of turning a thermal generator 'on' and 'off' has an associated cost. Although the starting and shutdown costs of unit t could be modeled taking into account the cooling profile of the generator, in the present work these costs have been considered constant: C_{on_t} and C_{off_t} respectively. Hence, the expression of the management cost is

$$MC(p) := \sum_{t=1}^{n_t} \sum_{i=1}^{n_i} MC_t(p_t^0, \dots, p_t^i) \tag{2.33}$$

where the function $MC_t(p_t^0, \dots, p_t^i)$ is defined as

$$MC_t(p_t^0, \dots, p_t^i) := \begin{cases} C_{off_t} & \text{if } p_t^{i-1} > 0 \text{ and } p_t^i = 0 \\ c_{b_t} & \text{if } p_t^{i-1} > 0 \text{ and } p_t^i > 0 \\ C_{on_t} + c_{b_t} & \text{if } p_t^{i-1} = 0 \text{ and } p_t^i > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2.34}$$

p_t^0 are given parameters which determine the state of the thermal units before the first time interval.

2.5 Coupled model of the unit commitment problem

The inclusion of an explicit representation of the transmission network is usually necessary to cope with the constraints imposed by the capacity of the transmission lines. Even if an electric utility owns a very strong transmission network, it is possible that, due to energy trading among neighboring zones, the capacity of the network could be exceeded. This problem is avoided if a model of the network is included explicitly. However, if the electric utility considers that the transmission network is not relevant in the study of the GUC problem, then the coupled model of the GUC problem can easily be reformulated as a Unit Commitment (UC) problem. The new UC problem to be solved is

$$(\mathrm{UC})\begin{cases} \min & GC(p,q) := TC(p) + MC(p) \\ \text{s.t.} & (d,v,s,p,r_{_I},g_{_I},r_{_D},g_{_D}) \in D_{ht} \\ & \hfill p \in D_m \end{cases}$$

where variables q and g have been omitted, in the same way as the term $DC(q)$ in the objective function.

The new domain D_{ht} can easily be reformulated from D_{htd} excluding the transmission network. The new set of constraints defining the domain D_{ht} is

$$\begin{aligned} A_{_{HTE}}x &= b_{_{HTE}} && (2.35)\\ h_L(d,v,p) &= L && (2.36)\\ h_I(r_{_I},d,v) &\geq b_{_{ISR}} && (2.37)\\ h_D(r_{_D},d,v) &\geq b_{_{DSR}} && (2.38)\\ \underline{x} \leq x &\leq \bar{x} && (2.39) \end{aligned}$$

where:

Eq. (2.35) shows the network constraints associated with the so called *Hydrothermal Extended (HTE) network*, similar to the HTTE network but without the equations associated with the transmission network (see [HN95]).

Eq. (2.36) provides the load constraints. This nonlinear side constraint matches the generated power with the required power L^i (*system load*). The vector function $h_L(d,v,p)$ is the total power output of all generators at each interval, that is,

$$h_L^i(d^i,v^{i-1},v^i,s^i,p^i) := \sum_{t=1}^{n_t} p_t^i + \sum_{r=1}^{n_r} h_r(d_r^i,v_r^{i-1},v_r^i,s_r^i) = L^i, \quad i=1,\ldots,n_i. \quad (2.40)$$

Eq. (2.37), (2.38). Incremental and decremental spinning reserve side constraints, which are equivalent to constraints (2.19) and (2.20). Given that in this case the variables g are not considered in the model, the explicit expression of the hydro generation must be included in these constraints, providing a set of $2n_i$ nonlinear side constraints

$$h_I({r_{_I}}^i,d^i,v^{i-1},v^i,s^i) \;:=\; \sum_{t=1}^{n_t} {r_{_{It}}}^i + \sum_{r=1}^{n_r}\left(\bar{h}_r^i - h_r(d_r^i,v_r^{i-1},v_r^i,s_r^i)\right) \geq {R_{_I}}^i \quad i=1,\ldots,n_i$$

$$(2.41)$$

$$h_D({r_{_D}}^i, d^i, v^{i-1}, v^i, s^i) \quad = \quad \sum_{t=1}^{n_t} {r_{_D}}_t^i + \sum_{r=1}^{n_r} h_r(d_r^i, v_r^{i-1}, v_r^i, s_r^i) \geq {R_{_D}}^i \quad i = 1, \ldots, n_i. \tag{2.42}$$

Eq. (2.39). Upper and lower bounds to the variables. The same as in domain D_{htd}.

The management domain D_m is the same as that described in section 2.3. The objective function is

$$GC(p,q) = TC(p) + MC(p)$$

where $TC(p)$ and $MC(p)$ are the functions defined in (2.31) and (2.33) respectively.

The software developed for this thesis implements the above UC model, but other models are also possible. An alternative model is the one already used for the GUC problem (2.17-2.22). However, since the UC problem does not consider the transmission network, the network constraint (2.17) should be modified accordingly and equation (2.21) should be removed.

2.6 Summary

In this chapter we have expounded our model for the Generalized Unit Commitment (GUC) problem. Three main elements form this model. First, the hydrothermal distribution (HTD) domain models the hydrothermal generation system and the distribution network in a connected fashion, that is, the HTD domain is a connected set. Second, the management domain models the management of the thermal generating units (starts, stops, etc.) using a disconnected set. Third, the objective function accounts for a triple cost: the fuel cost, the distribution losses, and the unit management cost.

Chapter 3

Solving the generalized unit commitment problem

The aim of chapter 2 was to model the Generalized Unit Commitment (GUC) problem, whereas the focus now is on how to solve it. In this chapter, by means of the variable duplication method, the GUC problem is decomposed into two subproblems: the thermal subproblem and the hydrothermal distribution subproblem. The thermal subproblem is solved using a dynamic programming algorithm, whereas the hydrothermal distribution subproblem is solved using a specialized nonlinear network flow algorithm. A very small example is given to illustrate the resolution of the thermal subproblem. Furthermore, the variable duplication method is compared from a theoretical point of view with the classical Lagrangian relaxation method.

3.1 The variable duplication method

3.1.1 A simple example of the unit commitment problem

If in the generalized unit commitment problem, presented in chapter 2, we do not consider the distribution network then we have the Unit Commitment (UC) problem. First, we start with the following very simple UC problem:

$$\left.\begin{array}{ll} \min & f(x,y) := 2x^2 + C_{on}(x) + 2y^2 + C_{on}(y) \\ s.t. & x + y = 2 \\ & x \in \{0\} \cup [1,2] \\ & y \in \{0\} \cup [1,2] \end{array}\right\} \tag{3.1}$$

In this example there are two thermal units: x stands for the production of the first thermal unit and y for the production of the second one. We must produce 2 MW of electrical power ($x + y = 2$) at the lowest cost. The production cost increases quadratically and we must also pay an extra starting cost whenever a unit is turned on ($C_{on}(x) = 0$ if $x = 0$, and $C_{on}(x) = C_{on_x} > 0$ if $x > 0$; analogously for $C_{on}(y)$).

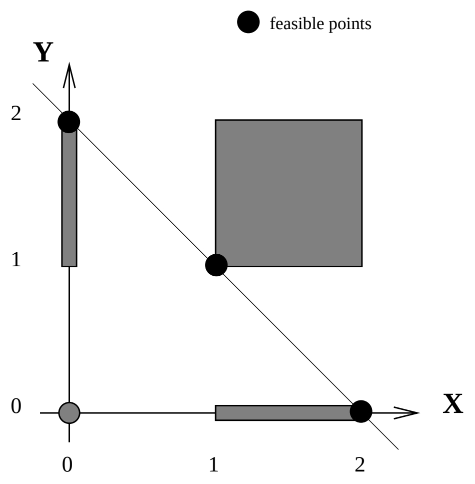

Figure 3.1: Problem (3.1) has only three feasible points.

Most of the literature (see for example [VAL90]) models the binary on/off state of the thermal units by means of binary variables. An equivalent binary variable formulation of problem (3.1) would be

$$\left.\begin{array}{ll} \min & 2x^2 + u_x C_{on_x} + 2y^2 + u_y C_{on_y} \\ s.t. & \quad x + y = 2 \\ & u_x 1 \le x \le u_x 2 \\ & u_y 1 \le y \le u_y 2 \\ & u_x, u_y \in \{0,1\} \end{array}\right\} \tag{3.2}$$

We propose a disconnected feasible set instead, for example $x \in \{0\} \cup [1,2]$, to avoid the binary variables. The x-unit is either switched off and does not produce any power or is switched on and its power production must be in $[1,2]$. The same applies to the y-unit (see Figure 3.1). The two formulations are equivalent, are of combinatorial nature and represent the main difficulty of the UC problem. We have to deal either with an nonlinear mixed integer programming problem or with an equivalent nonlinear problem with a disconnected feasible set. Even if our formulation does not use binary variables, in some situations we will use the term 'number of binary variables', meaning the number of on/off decisions that must be taken in a UC problem.

Therefore we have a nonconvex domain in the simple UC problem above. However, this is not the only nonconvex aspect of the UC problem, since the objective function is nonconvex as well. In order to see that, in the above example let us consider the cost function associated to the x-unit defined in the whole closed interval $[0,2]$ as (note that here $C_{on_x} = 1$)

$$f(x) := \begin{cases} 2x^2 + 1 & \text{if} \quad x \in]0,2], \\ 0 \quad \text{if} & x = 0. \end{cases} \tag{3.3}$$

This function, obviously, is nonconvex since, as it can be appreciated in Figure 3.2, its epigraph is not a convex set. ($f(x)$ is a convex function if and only if $epi(f)$ is a

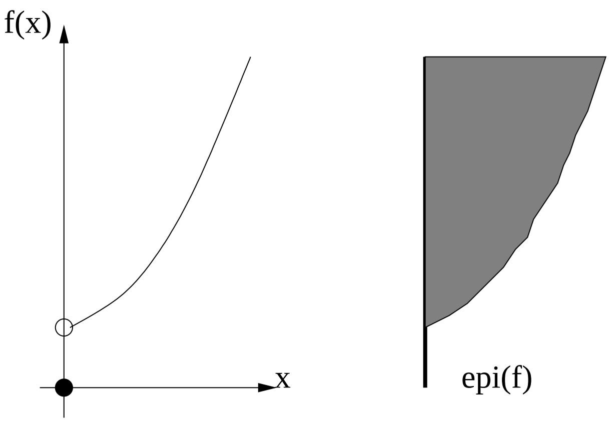

Figure 3.2: x-cost function and its epigraph.

convex set, [BS79], page 85.) Note that in the simple UC problem above $f(x)$ is only used in the feasible set $\{0\} \cup [1,2]$, which does not change the nonconvex character of $f(x)$. Analogously, it can be shown that the total cost function $f(x,y)$ is not convex, either. Thus, we can conclude that the simple UC problem is nonconvex due to both its domain and its objective function. Realistic large-scale UC or GUC problems share this nonconvex nature caused by the on/off state of the thermal units.

With the aim of solving the UC problem, following [BR92], we duplicate each variable (Variable Duplication (VD) method) in order to manage, on the one hand, the coupling constraint $x + y = 2$ (a constraint that prevents the decomposition of the problem) and, on the other hand, the disconnected set $(\{0\} \cup [1,2])^2 \subset R^2$. Obviously problems (3.1) and (3.4) are equivalent regarding the optimal solution set.

$$\left.\begin{array}{ll} \min & x^2 + y^2 + \tilde{x}^2 + C_{on}(\tilde{x}) + \tilde{y}^2 + C_{on}(\tilde{y}) \\ s.t. & x + y = 2 \\ & \tilde{x} \in \{0\} \cup [1,2] \\ & \tilde{y} \in \{0\} \cup [1,2] \\ & x = \tilde{x}, \quad y = \tilde{y} \end{array}\right\} \tag{3.4}$$

The above variable duplication formulation can be recast in a more general form:

$$\left.\begin{array}{ll} \min & f(x,y) + \tilde{f}(\tilde{x},\tilde{y}) \\ s.t. & (x,y) \in \mathcal{D} \\ & (\tilde{x},\tilde{y}) \in \tilde{\mathcal{D}} \\ & (x,y) = (\tilde{x},\tilde{y}) \end{array}\right\} \tag{3.5}$$

where $(x,y) \in R^2$, $\mathcal{D}$ is the connected domain defined by the coupling constraints and $\tilde{\mathcal{D}}$ represents the disconnected domain. This is the primal problem to be solved if we use the VD method.

3.1.2 Variable duplication versus classical Lagrangian relaxation from a theoretical point of view

Let us compare the Classical Lagrangian Relaxation (CLR) method with the Variable Duplication (VD) method as decomposition methods to solve the following problem

$$\left.\begin{array}{ll} \min & f(x) \\ s.t. & Ax \leq b \\ & x \in X \end{array}\right\} \tag{3.6}$$

where A is an $m \times n$ matrix, $X \subset R^n$ is only required to be a compact set (X could be, for example, a set of integer vectors, or a disconnected set, etc.) and $f : R^n \to R$ attains a minimum in X. It is also assumed that problem (3.6) is much more difficult to solve than problem $\min\{f(x) + \mu'(Ax - b) : x \in X\}$, as for example, in the case where $f(x)$ is a separable function, X is a separable domain and $Ax \leq b$ is a coupling constraint.

Traditionally problem (3.6) has been solved by means of the CLR method, which solves the so called dual problem

$$\begin{array}{c} \max \\ \mu \geq 0 \end{array} \left\{ \begin{array}{ll} \min & f(x) + \mu'(Ax - b) \\ s.t. & x \in X. \end{array} \right\} \tag{3.7}$$

In dual theory two functions are defined: the Lagrangian (function) $L(x, \mu) := f(x) + \mu'(Ax - b)$ and the dual function $q(\mu) := \min\{L(x, \mu) : x \in X\}$. With these definitions the dual problem can be recast as $\max\{q(\mu) : \mu \geq 0\}$.

Alternatively the VD method solves a problem which is equivalent to (3.6), that is,

$$\left.\begin{array}{ll} \min & \alpha f(x) + \beta f(\tilde{x}) \\ s.t. & A\tilde{x} \leq b \\ & x \in X \\ & \tilde{x} \in \widetilde{X} \\ & x - \tilde{x} = 0 \end{array}\right\} \tag{3.8}$$

where $\widetilde{X}$ is a compact connected set such that $X \subset \widetilde{X}$ (for example $\widetilde{X} = \mathrm{Co}(X)$), and α and β are positive scalars such that $\alpha + \beta = 1$. Then, the VD method is nothing but the CLR method applied to this equivalent problem when relaxing the equality constraint $x - \tilde{x} = 0$. In this case, the (variable duplication) Lagrangian $L_\alpha(x, \tilde{x}, \lambda) := \alpha f(x) + \beta f(\tilde{x}) + \lambda'(x - \tilde{x})$, the (variable duplication) dual function $q_\alpha(\lambda) := \min\{L_\alpha(x, \tilde{x}, \lambda) : Ax \leq b,\ \ x \in X,\ \ \tilde{x} \in \widetilde{X}\}$, and the associated (variable duplication) dual problem.

$$\begin{array}{c} \max \\ \lambda \in R^n \end{array} \left\{ \left(\begin{array}{ll} \min & \alpha f(x) + \lambda' x \\ s.t. & x \in X. \end{array} \right) + \left(\begin{array}{ll} \min & \beta f(\tilde{x}) - \lambda'\tilde{x} \\ s.t. & A\tilde{x} \leq b \\ & \tilde{x} \in \widetilde{X} \end{array} \right) \right\} \tag{3.9}$$

Note that now λ is free whereas μ must be nonnegative. Very often the dual problem is solved to provide a lower bound to the primal optimum. The important point with the VD method is that its associated dual bounds are, at least, as good as the CLR dual bounds. This result was proved in [GK87] for the particular case of linear integer programming. In the following proposition we extend the same result to the nonlinear function case with linear constraints.

Proposition 3.1 *Let us consider the following primal problem*

$$\left.\begin{array}{ll} \min & f(x) \\ s.t. & Ax \leq b \\ & x \in X \end{array}\right\} \tag{3.10}$$

where $X \subset R^n$ is a compact set, $f : R^n \to R$ (linear or otherwise) attains a minimum in X and A is a full row rank $m \times n$ matrix, and let us define

$$\begin{array}{rl}
\mu^* & (\geq 0) \text{ optimal solution of the CLR dual problem associated with (3.10),} \\
x_{\mu^*} & := \arg\min\{f(x) + \mu^{*\prime}(Ax - b) : x \in X\}, \\
\alpha, \beta & \text{positive scalars such that } \alpha + \beta = 1, \\
\lambda^*_\alpha & \text{optimal solution of the VD dual problem associated with (3.10) for } \alpha, \\
\lambda_0 & := A'\mu^*, \\
x^* & := \arg\min\{\alpha f(x) + \lambda_0{}' x : x \in X\}, \\
\widetilde{X} & \text{compact connected set such that } X \subset \widetilde{X}, \\
\tilde{x}^* & := \arg\min\{\beta f(\tilde{x}) - \lambda_0{}' \tilde{x} : A\tilde{x} \leq b, \tilde{x} \in \widetilde{X}\}.
\end{array}$$

Then:

a) $q_\alpha(\lambda^*_\alpha) \geq q(\mu^*) + \beta[f(\tilde{x}^*) - f(x^*)] - \mu^{*\prime}(A\tilde{x}^* - b)$.

b) For $\alpha = 1$ the associated VD dual bound is at least as good as the one provided by the CLR method, i.e. $q(\mu^*) \leq q_1(\lambda^*_1)$.

Proof.

a) By definition

$$\begin{array}{rcl}
q_\alpha(\lambda^*_\alpha) & \geq & q_\alpha(\lambda_0) = \\
& = & \alpha f(x^*) + \beta f(\tilde{x}^*) + \lambda_0{}' x^* - \lambda_0{}' \tilde{x}^* = \\
& = & \alpha f(x^*) + \beta f(\tilde{x}^*) + \mu^{*\prime} A x^* - \mu^{*\prime} A \tilde{x}^* = \\
& \underset{(\pm\mu^{*\prime}b)}{=} & \alpha f(x^*) + \beta f(\tilde{x}^*) + \mu^{*\prime}(Ax^* - b) - \mu^{*\prime}(A\tilde{x}^* - b) = \\
& \underset{(\pm\beta f(x^*))}{=} & f(x^*) + \mu^{*\prime}(Ax^* - b) + \beta[f(\tilde{x}^*) - f(x^*)] - \mu^{*\prime}(A\tilde{x}^* - b) \geq \\
& \geq & f(x_{\mu^*}) + \mu^{*\prime}(Ax_{\mu^*} - b) + \beta[f(\tilde{x}^*) - f(x^*)] - \mu^{*\prime}(A\tilde{x}^* - b) = \\
& = & q(\mu^*) + \beta[f(\tilde{x}^*) - f(x^*)] - \mu^{*\prime}(A\tilde{x}^* - b).
\end{array}$$

b) If $\alpha = 1$ then $\beta = 0$ and from a) we have that

$$q_1(\lambda_1^*) \geq q(\mu^*) - \mu^{*\prime}(A\tilde{x}^* - b). \tag{3.11}$$

Considering that

$$-\mu^{*\prime}(A\tilde{x}^* - b) \geq 0$$

because $\mu^* \geq 0$, and $A\tilde{x}^* - b \leq 0$, we can say from (3.11) that $q_1(\lambda_1^*) \geq q(\mu^*)$. □

For practical purposes we will prefer $\alpha = \frac{1}{2}$ to $\alpha = 1$, because of the following. If we define $\triangle := f(\tilde{x}^*) - f(x^*)$ then proposition (3.1) states that

$$q_\alpha(\lambda_\alpha^*) \geq q(\mu^*) + \beta \triangle - \mu^{*\prime}(A\tilde{x}^* - b). \tag{3.12}$$

There are three cases depending on the sign of $\triangle$: (1) If $\triangle > 0$ then, from (3.12), the best lower bound for $q_\alpha(\lambda^*)$ is attained at $\beta = 1$. (2) If $\triangle < 0$ then the best $q_\alpha(\lambda^*)$ bound is for $\beta = 0$. (3) The value of β is irrelevant for $\triangle = 0$. Given that, so far, a priori we do not know whether $\triangle \geq 0$ or the other way round, a compromise value for β would be $\frac{1}{2}$, that is, $\alpha = \frac{1}{2}$.

Possibly the greatest drawback with $\alpha = 1$ is the VD subproblem $\min\{\beta f(\tilde{x}) - \lambda'\tilde{x} : A\tilde{x} \leq b,\ \ \tilde{x} \in \widetilde{X}\}$, since the solution to this problem would always be a point at the frontier of the set $\{\tilde{x} \in R^n : A\tilde{x} \leq b,\ \ \tilde{x} \in \widetilde{X}\}$ (as $\beta = 0$, the objective function would be linear). In contrast, the solution to the other VD subproblem $\min\{\alpha f(x) + \lambda' x : x \in X\}$ could be any point of X. Therefore, with $\alpha = 1$, it seems difficult to obtain solutions such that $x^* - \tilde{x}^* \approx 0$. However, with $\alpha = \frac{1}{2}$ we minimize more similar functions in the two VD subproblems ($\frac{1}{2}f(x) + \lambda' x$ and $\frac{1}{2}f(\tilde{x}) - \lambda'\tilde{x}$). In this case it seems more likely that similar solutions can be obtained such that $x^* - \tilde{x}^* \approx 0$. Obviously we have not proved that $\frac{1}{2}$ is the best value for α, but for the above reasons we prefer this approach until new theoretical or practical issues advise other values.

3.1.3 Variable duplication versus classical Lagrangian relaxation in a simple example

In the above proposition we have seen that the Variable Duplication (VD) bound, theoretically, is at least as good as the Classical Lagrangian Relaxation (CLR) bound. This improvement, sometimes, may be remarkable as it is shown in the following example where we solve a problem of type (3.10). Note that the example we use has nothing to do with the Unit Commitment (UC) problem and that the objective function is concave.

The primal problem to solve is

$$\left.\begin{array}{ll} \min & f(x,y) := -2x^2 - 2y^2 \\ s.t. & x + y = 2 \\ & (x,y) \in X := \{(0,0),(1,1),(2,2)\} \end{array}\right\} \tag{3.13}$$

where, of course, $X \subset R^2$ is a compact set, $f : R^2 \to R$ attains a minimum in X, and the equality constraint $x + y = 2$ could have been written using two inequalities. Furthermore, the feasible set of problem (3.13) is $\{(1, 1)\}$ and thus $(x, y)^* = (1, 1)$ and $f((x, y)^*) = -4$.

The CLR method computes a dual bound to the primal problem (3.13) by solving the following dual problem:

$$\begin{array}{c} \max \\ \mu \in R \end{array} \left\{ \begin{array}{ll} \min & -2x^2 - 2y^2 + \mu(x + y - 2) \\ s.t. & (x, y) \in X \end{array} \right\} \tag{3.14}$$

or which is the same, $\max\{q(\mu) : \mu \in R\}$, where the reader can verify that the explicit expression for $q(\mu)$ is given by

$$q(\mu) = \left\{ \begin{array}{lll} 2\mu - 16 & \text{for} & \mu \leq 4, \\ -2\mu & \text{for} & \mu \geq 4. \end{array} \right. \tag{3.15}$$

The maximum of the dual function is attained at $\mu^* = 4$ with $q(\mu^*) = -8$ and, as a consequence, the duality gap is $f((x, y)^*) - q(\mu^*) = -4 - (-8) = 4$.

On the other hand, the VD method (taking arbitrarily $\alpha = \beta = 0.5$ and $\widetilde{X} := \text{Co}(X)$) transforms problem (3.13) into the equivalent problem

$$\left. \begin{array}{ll} \min & f(x, y, \tilde{x}, \tilde{y}) := -x^2 - y^2 - \tilde{x}^2 - \tilde{y}^2 \\ s.t. & \tilde{x} + \tilde{y} = 2 \\ & (x, y) \in X \\ & (\tilde{x}, \tilde{y}) \in \widetilde{X} = \text{Co}(X) \\ & (x, y) = (\tilde{x}, \tilde{y}) \end{array} \right\} \tag{3.16}$$

Then, the VD method computes a dual bound to the primal problem (3.16) by relaxing the constraint $(x, y) = (\tilde{x}, \tilde{y})$ and solving the resulting dual problem

$$\begin{array}{c} \max \\ \lambda \in R^2 \end{array} \left\{ \left(\begin{array}{ll} \min & -\tilde{x}^2 - \tilde{y}^2 + \lambda_1 \tilde{x} + \lambda_2 \tilde{y} \\ s.t. & \tilde{x} + \tilde{y} = 2 \\ & (\tilde{x}, \tilde{y}) \in \text{Co}(X) \end{array} \right) + \left(\begin{array}{ll} \min & -x^2 - y^2 - \lambda_1 x - \lambda_2 y \\ s.t. & (x, y) \in X \end{array} \right) \right\} \tag{3.17}$$

that is, $\max\{q(\lambda) : \lambda \in R^2\}$, where, as in the CLR method, the reader can verify that the explicit expression for $q(\lambda)$ is given by

$$q(\lambda) = \left\{ \begin{array}{lll} -\lambda_1 + -\lambda_2 - 2 & \text{for} & \lambda_1 + \lambda_2 \geq 4 \\ \lambda_1 + \lambda_2 - 10 & \text{for} & \lambda_1 + \lambda_2 \leq 4 \end{array} \right. \tag{3.18}$$

It is not difficult to show that the maximum of this dual function is attained at every point in $\Lambda^* := \{(\lambda_1, \lambda_2) \in R^2 : \lambda_1 + \lambda_2 = 4\}$ with $q^* := q(\lambda^*) = -6 \;\; \forall \lambda^* \in \Lambda^*$ and, thus, the duality gap is $f(x^*) - q^* = -4 - (-6) = 2$.

This particular example confirms the theoretical issue discussed in the previous section, since the VD method has reduced by 50% the duality gap obtained by the CLR method. However, the price we have had to pay is a higher dual dimension.

3.1.4 Unit commitment: variable duplication versus classical Lagrangian relaxation

The Unit Commitment (UC) problem was defined in chapter 1 as

$$\left.\begin{array}{ll} \min & TC(p) + MC(p) \\ s.t. & p \in D_{ED} \\ & p \in D_{UC} \end{array}\right\} \tag{3.19}$$

If we define $f(p) := TC(p) + MC(p)$, A as the demand and reserve matrix defining D_{ED}, that is, $D_{ED} := \{p \in R^n : Ap \leq b\}$ and X the (compact) UC domain, that is, $X := D_{UC}$, then the UC problem fits into the class of problems stated in (3.6). Therefore the UC problem can be solved following the Variable Duplication (VD) method as

$$\begin{array}{c}\max \\ \lambda \in R^n\end{array} \left\{ \left(\begin{array}{ll} \min & \frac{1}{2} \cdot TC(p) + MC(p) + \lambda' p \\ s.t. & p \in D_{UC} \end{array} \right) + \left(\begin{array}{ll} \min & \frac{1}{2} \cdot TC(\tilde{p}) - \lambda' \tilde{p} \\ s.t. & A\tilde{p} \leq b \\ & \tilde{p} \in \widetilde{D}_{UC} \end{array} \right) \right\} \tag{3.20}$$

where $\widetilde{D}_{UC}$ is a connected set that includes D_{UC}.

The main disadvantage of the VD method versus the Classical Lagrangian Relaxation (CLR) method when solving the UC problem is the increase in the number of Lagrange multipliers. If with the CLR method two multipliers per time interval are enough to relax the demand and reserve constraints, in the VD method the number of multipliers per time interval equals the number of thermal units (typically between 30 and 100 for a large-scale UC problem). Nevertheless, we think it is worthwhile using the VD method to solve the UC problem for the following reasons:

a) As was shown in the above section, in comparison with the CLR method, the VD method obtains equal or better dual bounds.

b) Solving the UC problem by the VD method we split the problem into two subproblems of a different nature. On the one hand the subproblem in D_{UC} is one of separable nonlinear mixed integer programming that can be efficiently solved using, for example, dynamic programming. Software previously developed to solve the UC problem by the CLR method can be reused. On the other hand the subproblem in D_{EC} is a typical economic dispatching problem or an optimal power flow (OPF) problem if the considered domain is $D_{ED} \cap D_{OPF}$. Again, software previously developed to solve the OPF problem can be reused. Furthermore, as pointed out in [BR92], the UC and OPF problems can be jointly solved in a coordinated fashion by means of the VD method.

c) The classical UC problem only considers two coupling constraints (demand and reserve) but if other coupling constraints, for example, an emission constraint, were to be added to the model, the CLR Lagrangian would get more and more complicated and difficult to handle, specially if an augmented Lagrangian were to be used. The Lagrangian or augmented Lagrangian used within the VD method is not affected by the addition of new constraints since we only relax the artificial constraint $x - \tilde{x} = 0$.

d) Within the CLR method the dual function associated with problem (3.6) is maximized over a nonnegative set of multipliers in R^m (m is twice the number of time intervals) whereas in the VD method the associated dual function is maximized over the whole R^n (n is the dimension of p in (3.19)). Therefore in the CLR method we perform a constrained maximization of the dual function in contrast with the easier unconstrained maximization within the VD method.

3.2 The variable duplication formulation of the generalized unit commitment problem

Mathematically, the Generalized Unit Commitment (GUC) problem can be formulated as

$$\left.\begin{array}{ll} \min & GC(p,q) := TC(p) + DC(q) + MC(p) \\ s.t. & (p,q,g) \in D_{htd} \\ & p \in D_m \end{array}\right\} \tag{3.21}$$

where:

p :	vector of thermal generation.
q :	vector of the power that circulates through each line of the distribution network.
g :	vector of the hydro variables (volumes, spillages and discharges).
$DC(q)$:	distribution cost.
$MC(p)$:	management cost.
$TC(p)$:	thermal cost.
$GC(p,q)$:	global cost.
$\mathcal{D}_m$:	management domain.
$\mathcal{D}_{htd}$:	hydrothermal-distribution domain.

More details about this formulation of the GUC problem can be found in chapter 2.

We solve this problem using the variable duplication method [BR92], which consists in duplicating the vector of powers p into $\tilde{p}$, adding the artificial equality constraint $p = \tilde{p}$, and relaxing this constraint to obtain the associated dual problem. In chapter 6 we will discuss the advantages (and disadvantages) of the Classical Lagrangian Relaxation (CLR) method versus the Augmented Lagrangian Relaxation (ALR) method within the framework of the VD method. In the current chapter we will solve the GUC problem by means of the VD method (ALR version). Then, in the resolution of the dual problem (see chapter 4), the augmented Lagrangian is split into two functions by means of the Block Coordinate Descent (BCD) method [Ber95]; one of these functions is minimized in the hydrothermal distribution domain $\mathcal{D}_{htd}$ and the other function is minimized in the thermal management domain $\mathcal{D}_m$. The first minimization is called the hydrothermal distribution subproblem and the second one, the thermal subproblem. To be more precise, first, we duplicate the vector p of problem (3.21) into $\tilde{p}$

$$\left.\begin{array}{ll} \min & GC(p,\tilde{p},q) \\ s.t. & (p,q,g) \in \mathcal{D}_{htd} \\ & \tilde{p} \in \mathcal{D}_m \\ & p = \tilde{p} \end{array}\right\} \tag{3.22}$$

and second, we relax the equality constraint $p = \tilde{p}$ to induce the dual problem

$$\max_{\lambda \in R^{n_i \times n_t}} \left\{ \begin{array}{ll} \min & L_c(p,\tilde{p},q,\lambda) \\ s.t. & (p,q,g) \in \mathcal{D}_{htd} \\ & \tilde{p} \in \mathcal{D}_m \end{array} \right\}, \tag{3.23}$$

where n_i is the number of intervals, n_t is the number of thermal units and

$$L_c(p,\tilde{p},q,\lambda) := GC(p,\tilde{p},q) + \lambda'(p-\tilde{p}) + \frac{c}{2}\|p-\tilde{p}\|^2 \tag{3.24}$$

is called the augmented Lagrangian [Ber95, Lue89, GMW95] associated with the function $GC(p,\tilde{p},q)$ and with the constraint $p = \tilde{p}$.

The augmented Lagrangian L_c is not a separable function; thus in order to minimize it in a separable way we first perform some manipulations.

$$\begin{array}{rcl} L_c(p,\tilde{p},q,\lambda) & = & \frac{1}{2}TC(p) + \frac{1}{2}TC(\tilde{p}) + DC(q) + MC(\tilde{p}) \\ & & +\lambda'p - \lambda'\tilde{p} + \frac{c}{2}\|p-\tilde{p}\|^2 = \\ & = & \left(\frac{1}{2}TC(p) + DC(q) + \lambda'p\right) \\ & & + \left(\frac{1}{2}TC(\tilde{p}) + MC(\tilde{p}) - \lambda'\tilde{p}\right) + \frac{c}{2}\|p-\tilde{p}\|^2 \end{array} \tag{3.25}$$

As we can see in (3.25), L_c consists of two almost separable terms in p and $\tilde{p}$ respectively, except for the quadratic term $\|p-\tilde{p}\|^2$. In the framework of the BCD method, at iteration n, we split L_c into two functions: $L_c^n(p,q,\lambda_n)$, to be minimized in the domain $\mathcal{D}_{htd}$; and $\tilde{L}_c^n(\tilde{p},\lambda_n)$, to be minimized in $\mathcal{D}_m$ by fixing one of the vectors p or $\tilde{p}$ each time. More precisely,

$$L_c^n(p,q,\lambda_n) := \frac{1}{2}TC(p) + DC(q) + \lambda_n'p + \frac{c}{2}\|p-\tilde{p}_n\|^2, \tag{3.26}$$

$$\tilde{L}_c^n(\tilde{p},\lambda_n) := \frac{1}{2}TC(\tilde{p}) + MC(\tilde{p}) - \lambda_n'\tilde{p} + \frac{c}{2}\|p_{n+1}-\tilde{p}\|^2, \tag{3.27}$$

where $\tilde{p}_n$ and p_{n+1} are fixed iterates of the decision variables $\tilde{p}$ and p respectively obtained by means of the BCD method (more details can be found in chapter 4). Then, in the Augmented Lagrangian Relaxation (ALR) algorithm that solves the dual problem (3.23), the minimization of the augmented Lagrangian over $\mathcal{D}_{htd}$ and $\mathcal{D}_m$, at each iteration n, is replaced by two subproblems, one in each domain: the hydrothermal subproblem

$$\left.\begin{array}{rl} p_{n+1} := & \arg\min L_c^n(p, q, \lambda_n) \\ & s.t. \;\; (p,q) \in \mathcal{D}_{htd}, \end{array}\right\} \tag{3.28}$$

and the thermal subproblem

$$\left.\begin{array}{rl} \tilde{p}_{n+1} := & \arg\min \tilde{L}_c^n(\tilde{p}, \lambda_n) \\ & s.t. \;\; \tilde{p} \in \mathcal{D}_m. \end{array}\right\} \tag{3.29}$$

3.3 The thermal subproblem

We will now show that the thermal subproblem (3.29) can be decomposed into one sub-subproblem per thermal unit. To do so we define $\tilde{p}_t$ and $\lambda_{n,t}$ as

$$\tilde{p}_t = \begin{pmatrix} \tilde{p}_t^1 \\ \vdots \\ \tilde{p}_t^{n_i} \end{pmatrix} \quad \lambda_{n,t} = \begin{pmatrix} \lambda_{n,t}^1 \\ \vdots \\ \lambda_{n,t}^{n_i} \end{pmatrix},$$

and also we arbitrarily define $\tilde{p}_t^0 = 0$ if initially the thermal unit t was off, and $\tilde{p}_t^0 = 1$ if it was on.

First, we consider $\tilde{L}_{c,t}^n$, the part of $\tilde{L}_c^n$ corresponding to the thermal unit t. That is

$$\begin{array}{rcl} \tilde{L}_{c,t}^n(\tilde{p}_t, \lambda_{n,t}) & := & MC_t(\tilde{p}_t) + \frac{1}{2}TC_t(\tilde{p}_t) - \lambda'_{n,t}\tilde{p}_t + \frac{c}{2}\|p_{n+1,t} - \tilde{p}_t\|^2 = \\ & = & \sum_{i=1}^{n_i} \Big\{ MC_t(\tilde{p}_t^0, \dots, \tilde{p}_t^i) \\ & & + [\frac{1}{2}TC_t(\tilde{p}_t^i) - \lambda_{n,t}^i \tilde{p}_t^i + \frac{c}{2}(p_{n+1,t}^i - \tilde{p}_t^i)^2] \Big\} \end{array} \tag{3.30}$$

with thermal cost

$$TC_t(\tilde{p}_t^i) := c_{q_t}(\tilde{p}_t^i)^2 + c_{l_t}\tilde{p}_t^i \tag{3.31}$$

where c_{q_t} and c_{l_t} stand for quadratic and linear coefficients of the thermal cost. Note that the usual constant term c_{b_t} of the thermal cost will be included in the management cost (3.32) for algorithmic purposes. The reason for this is that c_{b_t} must be added if and only if the thermal unit t is on and it is precisely within the computing of the management cost that we control the state of the thermal unit.

The management cost is defined as

$$MC_t(\tilde{p}_t^0, \dots, \tilde{p}_t^i) \;:=\; \begin{cases} C_{off_t} & \text{if } \tilde{p}_t^i = 0 \text{ and } \tilde{p}_t^{i-1} > 0 \\ C_{on_t} + c_{b_t} & \text{if } \tilde{p}_t^i > 0 \text{ and } \tilde{p}_t^{i-1} = 0 \\ c_{b_t} & \text{if } \tilde{p}_t^i > 0 \text{ and } \tilde{p}_t^{i-1} > 0 \\ 0 & \text{in other cases} \end{cases} \tag{3.32}$$

The reader can find the management domain $\mathcal{D}_m$ fully detailed in chapter 2. Furthermore, c_{b_t} stands for basic thermal cost and represents the constant cost we must pay if and only if the thermal unit is on.

And second, using (3.30), the thermal subproblem (3.29) can be reformulated as

$$\left.\begin{array}{ll} \min & \sum_{t=1}^{n_t} \widetilde{L}^n_{c,t}(\widetilde{p}_t, \lambda_{n,t}) \\ s.t. & \widetilde{p}_t \in \mathcal{D}_{m_t} \quad t = 1, \ldots, n_t. \end{array}\right\} \tag{3.33}$$

In conclusion, it is sufficient to solve a collection of n_t sub-subproblems (one per thermal unit).

$$\left.\begin{array}{ll} \min & \widetilde{L}^n_{c,t}(\widetilde{p}_t, \lambda_{n,t}) \\ s.t. & \widetilde{p}_t \in \mathcal{D}_{m_t} \end{array}\right\} \tag{3.34}$$

For computational purposes in the next sections we will use the term $\widetilde{L}^n_{c,t}(\widetilde{p}_t, \lambda_{n,t})$ rearranged as

$$\widetilde{L}^n_{c,t}(\widetilde{p}_t, \lambda_{n,t}) = \sum_{i=1}^{n_i} [MC_t(\widetilde{p}^0_t, \ldots, \widetilde{p}^i_t) + P^i_t(\widetilde{p}^i_t)] \tag{3.35}$$

where

$$P^i_t(\widetilde{p}^i_t) := \frac{1}{2} TC_t(\widetilde{p}^i_t) - \lambda^i_{n,t} \widetilde{p}^i_t + \frac{c}{2} (p^i_{n+1,t} - \widetilde{p}^i_t)^2 \tag{3.36}$$

is a second degree polynomial.

3.3.1 Dynamic programming and the solution of the thermal subproblem

Dynamic programming has been used to solve the aforesaid thermal subproblem since very early unit commitment (UC) papers and nowadays, as we saw in chapter 1, it is the most popular technique among the methods for the UC problem in the framework of Lagrangian relaxation. Alternatively, one could use the linear mixed integer programming to solve the thermal subproblem at the price of linearizing or approximating the quadratic objective function by a piecewise linear function.

An introduction to dynamic programming can be found in [Bra77]. Dynamic programming is an optimization approach that transforms a complex problem into a sequence of simpler problems; its essential characteristic is the multi-stage nature of the optimization procedure. There are two kinds of dynamic programming. If we start making decisions at the first stage, then move on the second stage to make decisions that depend on the first stage, and so on, we are using the Forward Dynamic Programming (FDP) method. Alternatively, if we start at the last stage and move back to the previous stage and so on, we are applying the Backward Dynamic Programming (BDP) method. The method we adopt to solve the thermal subproblem uses the FDP method, which can be summarized in the following algorithm.

FDP algorithm

* [Method.] Forward dynamic programming.

* [Objective.] To solve the following problem:

$$\left.\begin{array}{rcl} v_n(s_n) & = & \min[v_0(s_0) + f_0(d_0, s_0) + \cdots + f_{n-1}(d_{n-1}, s_{n-1})] \\ s.t. & & s_i = t_{i-1}(d_{i-1}, s_{i-1}) \quad i = 1, \ldots, n \\ & & d_i \in \mathcal{D}_i(s_i) \quad i = 0, \ldots, n-1 \end{array}\right\} \qquad (3.37)$$

* [Parameters, variables and functions.]

i	Stage or each point at which decisions are made.
n	Last stage.
s_i	State or information at stage i that summarizes the knowledge required about the problem in order to make the current decisions.
d_i	Decision at stage i.
$\mathcal{D}_i(s_i)$	Set of permissible decisions for stage i that depends on the current state s_i.
$v_i(s_i)$	Optimal value over the current and previous stages, given that we are in state s_i.
$f_i(d_i, s_i)$	Cost function for state s_i and for decision d_i at the stage i.
t_i	Transition function. Given that we are in state s_i and decide d_i, we then move to state $s_{i+1} := t_i(d_i, s_i)$.

* [Input.] Initial state s_0.

* [Output.] Optimal states $(s_1, \ldots, s_n)$, optimal decisions $(d_0, \ldots, d_{n-1})$ and optimal cost $v_n(s_n)$ for the problem (3.37).

Step 0 [Initialize.] Set $v_0(s_0) = 0$ or equal to the cost of being initially at s_0. Set $i = 1$.

Step 1 [Solve the subproblems i.]

$$\left.\begin{array}{rcl} v_i(s_i) & = & \min[v_{i-1}(s_{i-1}) + f_{i-1}(d_{i-1}, s_{i-1})] \\ s.t. & & s_i = t_{i-1}(d_{i-1}, s_{i-1}) \\ & & d_{i-1} \in \mathcal{D}_{i-1}(s_{i-1}) \end{array}\right\} \qquad (3.38)$$

Step 2 [Update the iteration count.] If $i < n$ set $i = i + 1$ and go back to step 1.

In the following algorithm we adapt the FDP algorithm to solve the thermal subproblem (3.29). Although the general structure of the algorithm remains the same, some minor modifications must be done.

T.FDP Algorithm

* [Method.] Forward dynamic programming.

* [Objective.] To solve the thermal subproblem:

$$\left.\begin{array}{ll} \min & \sum_{t=1}^{n_t} \widetilde{L}^n_{c,t}(\widetilde{p}_t, \lambda_{n,t}) \\ s.t. & \widetilde{p}_t \in \mathcal{D}_{m_t} \quad t = 1, \ldots, n_t. \end{array}\right\} \tag{3.39}$$

* [Parameters, variables and functions.]

i	Interval index. $i = 1, \ldots, n_i$.
s	State index. $s = 1$ means the thermal unit has been turned on for a period of 1 interval. $s = -2$ means the thermal unit has been turned off for a period of 2 intervals. $s = -min_{off}, \ldots, -1, 1, \ldots, min_{on}$.
t	Thermal unit index. $t = 1, \ldots, n_t$.
min_{off_t}	Minimum down-time for the thermal unit t.
$min_{on t}$	Minimum up-time for the thermal unit t.
$C(s,i)$	Best cumulative cost for the couple (s,i).
$\phi(s)$	Predecessor function. $\phi(2) = 1$ means that from state 1 we go to state 2.
$M(s)$	Management cost for the state s.
$T(s,i)$	Optimal thermal cost for the couple (s,i).
$\Phi(s,i)$	Optimal predecessor for the couple (s,i).
$\underline{P}_t$	Lower bound for the power output of the thermal unit t.
$\overline{P}_t$	Upper bound for the power output of the thermal unit t.
$Pow(s,i)$	Optimal power for the couple (s,i).
$C_{on t}$	Start-up cost for the thermal unit t.
C_{off_t}	Shut-down cost for the thermal unit t.

* [Input.] Set for each thermal unit the parameters min_{off_t}, min_{ont}, $\underline{P}_t$, $\overline{P}_t$, C_{ont}, C_{off_t} and the initial state s_{0t} $(t = 1, \ldots, n_t)$.

* [Output.] Given the current value of $\lambda_{n,t}$, for each thermal unit t we get the optimal schedule u_t, the optimal power production $\widetilde{p}_t$ and the optimal objective function $\widetilde{L}^n_{c,t}(\widetilde{p}_t, \lambda_{n,t})$.

Step 0 [Initialize the thermal unit counter.] Set $t = 1$.

Step 1 [Initialize the thermal unit parameters and functions.] For the current thermal unit t initialize

$$
\begin{aligned}
C(s,0) &= \left\{ \begin{array}{lll} 0 & \text{if} & s = s_{0t}, \\ +\infty & \text{if} & s \neq s_{0t}, \end{array} \right\} \\
\phi(s) &= \left\{ \begin{array}{lll} s+1 & \text{if} & s = -min_{off_t}, \ldots, -2, \\ min_{ont} & \text{if} & s = -1, \\ -min_{off_t} & \text{if} & s = 1, \\ s-1 & \text{if} & s = 2, \ldots, min_{ont}, \end{array} \right\} \\
M(s) &= \left\{ \begin{array}{ll} C_{off_t} & \text{if } s = -1, \\ C_{ont} & \text{if } s = 1, \\ 0 & \text{in other cases}, \end{array} \right\} \\
\Phi(s,i) &= \phi(s) \quad i = 1, \ldots, n_i.
\end{aligned}
$$

Step 2 [For each couple (s,i) compute the best level of power according to the interval objective function $MC_t(\tilde{p}_t^i) + P_t^i(\tilde{p}_t^i)$.]

For $i = 1, \ldots, n_i$

$$\left\{ \begin{array}{l} T(s,i) = \left\{ \begin{array}{ll} \min\limits_{x \in [\underline{P}_t, \overline{P}_t]} P_t^i(x) & \text{for } s = 1, \ldots, min_{on_t}, \\ P_t^i(0) & \text{for } s = -min_{off_t}, \ldots, -1. \end{array} \right\} \\ Pow(s,i) = \left\{ \begin{array}{ll} \arg\min\limits_{x \in [\underline{P}_t, \overline{P}_t]} P_t^i(x) & \text{for } s = 1, \ldots, min_{on_t}, \\ 0 & \text{for } s = -min_{off_t}, \ldots, -1. \end{array} \right\} \\ \left\{ \begin{array}{l} \text{For } s = -min_{off_t}, \ldots, -1, 1, \ldots, min_{on_t}, \\ C(s,i) = C(\phi(s), i-1) + M(s) + T(s,i). \end{array} \right\} \end{array} \right\}$$

The states $s = -min_{off}$ and $s = min_{on}$ are the only ones with two possible predecessors. The possible predecessors of $s = -min_{off}$ are $s = -min_{off} + 1$ and $s = -min_{off}$. Take the best and update $C(-min_{off}, i)$ and $\Phi(-min_{off}, i)$. Analogously, the possible predecessors of $s = min_{on}$ are $s = min_{on} - 1$ and $s = min_{on}$. Take the best and update $C(min_{on}, i)$ and $\Phi(min_{on}, i)$ accordingly.

Step 3 [Recover the best decision.] From $C(s,i), Pow(s,i)$ and $\Phi(s,i)$ compute the optimal schedule u_t, the optimal power production $\tilde{p}_t$ and the optimal function value $\tilde{L}_{c,t}^n(\tilde{p}_t, \lambda_{n,t})$.

Step 4 [Update the thermal unit index.] If $t < n_t$ set $t = t + 1$ and go back to step 1.

3.3.2 A small example

Consider the following simplified thermal subproblem provided with only one thermal unit that initially is shut down ($p_0 = 0$) and a period of two hours divided into two intervals. Furthermore, p_i stands for the power to be produced during the interval i. The management cost MC is defined as follows.

$$MC(p_0, \ldots, p_i) := \left\{ \begin{array}{ll} 0 & \text{if } p_{i-1} > 0 \text{ and } p_i = 0. \ (Cost_{off}) \\ 1 & \text{if } p_{i-1} = 0 \text{ and } p_i > 0. \ (Cost_{on}) \\ 0 & \text{in other cases.} \end{array} \right. \tag{3.40}$$

Note that in the objective function there are two constant terms equal to 10 that could have been removed to perform the optimization.

$$\left.\begin{array}{ll} \min & [(p_1)^2 - 4p_1 + 10 + MC_1(p_0,p_1)] + [(p_2)^2 - 6p_2 + 10 + MC_2(p_0,p_1,p_2)] = \\ & = [P_1(p_1) + MC_1(p_0,p_1)] + [P_2(p_2) + MC_2(p_0,p_1,p_2)] \\ & \\ s.t. & p_i \in [2.5, 4] \qquad i = 1,2 \\ & min_{on} = 2 \\ & min_{off} = 2 \\ & s_0 = -2 \end{array}\right\} \tag{3.41}$$

The resolution of this problem through the T.FDP algorithm is as follows:

* [Parameters, variables and functions.]

s	State index.
$C(s,i)$	Best cumulative cost for the couple (s,i).
$\phi(s)$	Predecessor function. $\phi(2) = 1$ means that from state 1 we go to state 2.
$\Phi(s,i)$	Optimal predecessors for the couple (s,i).
$M(s)$	Management cost for a state s.
$T(s,i)$	Optimal thermal cost for the couple (s,i).
$Pow(s,i)$	Optimal power for the couple (s,i).

Step 1 [Initialize the thermal unit parameters and functions.]

s	-2	-1	1	2
$C(s,0)$	0	∞	∞	∞
$\phi(s)$	-1	2	-2	1

Step 2 [For each couple (s,i) we compute the best level of power.]

First, we iterate for $i = 1$

s	-2	-1	1	2
$Pow(s,1)$	0	0	2.5	2.5
$T(s,1)$	0	0	6.3	6.3
$M(s)$	0	0	1	0
$C(\phi(s),0)$	$C(-1,0)=\infty$	$C(2,0)=\infty$	$C(-2,0)=0$	$C(1,0)=\infty$
	$C(-2,0)=0$ (i)			$C(2,0)=\infty$ (ii)
$C(s,1)$ (iii)	10 (iv)	∞	7.3	∞ (v)
$\Phi(s,1)$	-2	2	-2	2

(i) $s=-2$ is the other possible predecessor of $s=-2$.

(ii) $s=2$ is the other possible predecessor of $s=2$.

(iii) $C(s,i) = C(\phi(s), i-1) + M(s) + T(s,i)$.

(iv) The states $s = -min_{off} = -2$ and $s = min_{on} = 2$ are the only ones with two possible predecessors. The possible predecessors of $s = -2$ are $s = -1$ and $s = -2$. Take the best.

$$\begin{aligned} C(-2,1) &= \min\{C(-1,0) + M(-2) + T(-2,1), \\ &\quad C(-2,0) + M(-2) + T(-2,1)\} = \\ &= \min\{\infty + 0 + 10, \quad 0 + 0 + 10\} = 10 \end{aligned}$$

(v) Analogously, the possible predecessors of $s = min_{on} = 2$ are $s = 1$ and $s = 2$. Take the best.

$$\begin{aligned} C(2,1) &= \min\{C(1,0) + M(2) + T(2,1), \quad C(2,0) + M(2) + T(2,1)\} = \\ &= \min\{\infty + 0 + 6.3, \quad \infty + 0 + 6.3\} = \infty \end{aligned}$$

Second, we iterate for $i = 2$

s	-2	-1	1	2
$Pow(s,2)$	0	0	3	3
$T(s,2)$	0	0	1	1
$M(s)$	0	0	1	0
$C(\phi(s),1)$	$C(-1,1) = \infty$	$C(2,1) = \infty$	$C(-2,1) = 10$	$C(1,1) = 7.3$
	$C(-2,1) = 10$ (vi)			$C(2,1) = \infty$ (vi)
$C(s,2)$	20 (vii)	∞	12	**8.3** (vii)
$\Phi(s,2)$	-2	2	-2	1

(vi) Analogously to (i) and (ii).

(vii) Analogously to (iv) and (v).

Step 3 [Recover the best decision.] As the table for $i = 2$ shows, the cheapest final cost is $C(2,2) = 8.3$, which corresponds to $p_2^* = Pow(2,2) = 3$ for $s_2^* = 2$. The predecessor of s_2^* is equal to $\Phi(2,1) = 1$ with a level of power $p_1^* = Pow(1,1) = 2.5$.

In conclusion, the optimal power is $p^* = (2.5, 3)$, with an optimal cost $C^* = 8.3$ and an optimal schedule $u^* = (1,1)$. The above iterations are summarized in Figure 3.3.

3.4 The hydrothermal subproblem

The objective function $L_c^n(p, q, \lambda_n)$ of subproblem (3.28) is a quadratic function of variables p and q (see (3.27)). The constraints defining the hydrothermal distribution domain D_{htd} are those described in chapter 2. Subproblem (3.28) is closely related with a problem called the *Short-term Hydrothermal Scheduling (STHS)* problem, which has been solved in [HN94, HN95]. If we consider the transmission network,

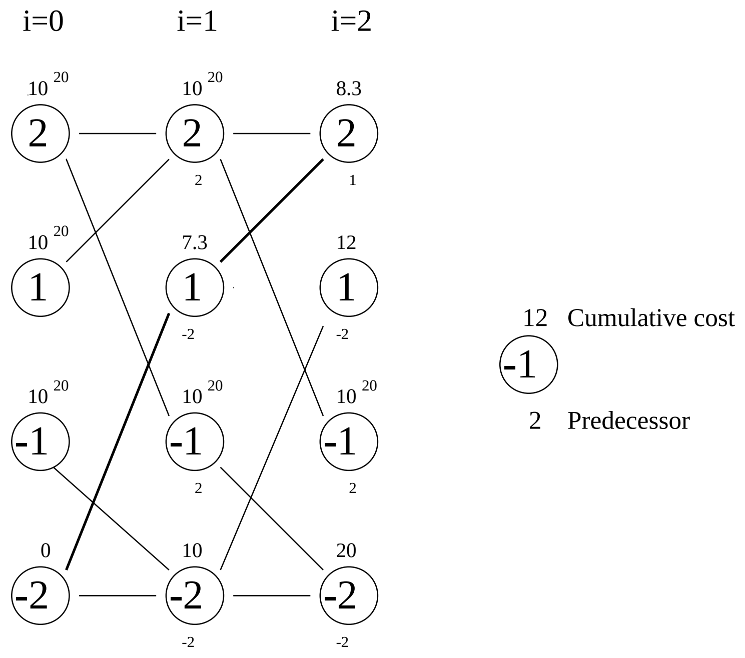

Figure 3.3: Transition graph.

the STHS problem is nothing but the optimal power flow problem, if not, it is the economic dispatching problem. The formulation of the STHS problem is exactly the same as the formulation of subproblem (3.28) except that the STHS objective function does not include the augmented Lagrangian term $\lambda_n' p + \frac{c}{2}\|p - \tilde{p}_n\|^2$, and that in the STHS problem we have lower power bounds different from zero, unlike in subproblem (3.28). However, these differences do not affect the solution method. With or without the transmission network, subproblem (3.28) corresponds to a *Nonlinear Network flow problem with Side Constraints* (NNSC) problem. We consider two alternative ways to solve this problem:

1. It is possible to use any standard nonlinear optimization code, such as MINOS [MS95]. This alternative does not take advantage of the network structure of the subproblem (3.28).

2. As subproblem (3.28) is an NNSC problem, it is possible to take advantage of the network structure using specialized nonlinear network flow codes such as NOXCB [HN92, Her95], PFNRN01 [Mij96], or CPLEX [CPL95].

In this work we have followed the second approach, using the code NOXCB. This is an optimization code specialized in the solution of large-scale *Nonlinear Network flow problems with Linear Side Constraints* (NNLSC), that is, problems which can be formulated as

$$\text{(NNLSC)}\left\{\begin{array}{ll} \min & f(x) \\ s.t. & Ax = b \\ & \underline{c} \le Tx \le \bar{c} \\ & 0 \le x \le \bar{x} \end{array}\right.$$

This problem is equivalent to subproblem (3.28) without the nonlinear side constraints (2.18) and (2.36). The availability of the NOXCB code allows the solution of subproblem (3.28) if the exact value of the hydro generation h_r^i is substituted in the model by its linearization ${h_L}_r^i$ arising from the Taylor's series expansion around a given point, which gives an expression such as

$$h_r^i := h(d_r^i, v_r^{i-1}, v_r^i, s_r^i) \approx {h_L}_r^i := \lambda_{0r}^i + \lambda_{v(i-1)r}^i v_r^{i-1} + \lambda_{v(i)r}^i v_r^i + \lambda_{dr}^i d_r^i \tag{3.42}$$

where λ_{0r}^i is the independent term and $\lambda_{v(i-1)r}^i$, $\lambda_{v(i)r}^i$ and λ_{dr}^i are respectively the linear coefficients of the network variables v_r^{i-1}, v_r^i and d_r^i. The analytical expression of the independent term and of the linear coefficients are easy (though cumbersome) to derive and are given in [HN94]. Expression (3.42) could be used to eliminate the nonlinear expressions $h(d_r^i, v_r^{i-1}, v_r^i, s_r^i)$ from constraints (2.18) and (2.36), so that all the constraints in the model become linear. The successive linearization method reported in [HN95] proceeds, from a given starting point x^0, solving the linearized problem $(\text{NNLSC})^k$ with the hydro generation linearized around the solution x^k, which is taken as the optimal solution of the previous linearized problem $(\text{NNLSC})^{k-1}$. The stopping condition is the reduction of the discrepancy between the exact and linearized hydro generation at each time interval under a given fraction ϵ_L of the forecasted load L^i. The algorithm can be summarized as follows:

1. **Initializations**
   (a) Definition of the network equations of (NNLSC) (constraints (2.17) or (2.35)).
   (b) Selection of the initial solution x^0, $k := 1$.
   (c) Selection of the maximum hydro generation error : ϵ_L.
2. **Major iterations**
   (a) Linearization (3.42) around $x^{k-1} \rightarrow (\text{NNLSC})^k$.
   (b) Optimization of $(\text{NNLSC})^k$ with NOXCB $\rightarrow x^k$.
   (c) If $|\sum_{r=1}^{n_r} \left([{h_L}_r^i]^k - [h_r^i]^k\right)| < \epsilon_L L^i$, $\forall i$, then $x^* := x^k$ STOP.
   (d) $k := k+1$. Go to 2.

This methodology has been implemented (code MAPH3 [Her97]) and applied successfully to solve the STHS problem. The starting point x^0 could be a feasible solution for the network constraints (2.17) or (2.35), a solution with the variables describing the volume of each reservoir at its upper bound, or a solution x stored from a previous execution. In [HN94, HN95, Her95] numerical experiments were done in order to compare the behavior of the NOXCB based methodology with that of the general purpose optimization code MINOS. The results show that the proposed method met the convergence criterion 2c in only two or three linearizations (major iterations), with a discrepancy in the objective function value of no more than 0.091% on average with respect to those provided by MINOS. The execution time of the MAPH3 package was, on average, 9% of the total execution time of MINOS. Based on this results it seems that the linearization of hydro generation with respect to initial and final volume and discharge at each interval produces results of sufficient accuracy and permits the use of specialized network flow codes with linear side constraints, which are much more efficient than general purpose nonlinear optimization codes and prove to be a valid tool for the STHS problem. As subproblem (3.28) is a STHS problem with a modified objective function, it seems reasonable to use the same methodology to solve it.

3.5 Load primal feasibility and full primal feasibility

Lagrangian relaxation is a very common technique used to solve mixed-integer problems. One solves the dual problem to obtain both a lower cost bound and a primal solution. This solution usually does not fulfill the relaxed constraints, i.e. it is primal infeasible. Therefore, solving the Unit Commitment (UC) problem by Lagrangian relaxation leads to solutions which usually are primal infeasible. The ordinary way of gaining primal feasibility, when solving the UC problem, is by means of a heuristic procedure [ZG88, FK00]. The general idea of any of these heuristic procedures is to perturb the current solution in order to gain feasibility, while maintaining optimality as much as possible. Near-optimal solutions are thus obtained.

Alternatively, in order to obtain primal feasibility, one can augment the Lagrangian [BR92] by adding the squared norm of the relaxed constraint. Furthermore, this added term is weighted by a penalty factor which is increased as needed to obtain full primal feasibility. With an augmented Lagrangian, points primal infeasible are thus strongly penalized. Throughout this thesis we follow this approach (Augmented Lagrangian Relaxation (ALR) method) to obtain primal feasible solutions (see chapters 4 and 6). The ALR method, as a method to gain primal feasibility, is a tidy method, not very difficult to implement and can be used for different types of problems apart from the UC problem, since it is indeed a general method. However, in nonconvex problems, the solution thus obtained is a local optimizer which must be assessed (one can use the dual cost bound).

A part from heuristic procedures or the ALR method, there are other possible approaches to gain primal feasibility. [LPS99] solve the traffic equilibrium assignment problem by the subgradient method. Then, they improve primal feasibility by the construction of sequences of weighted averages of primal (infeasible) solutions during the optimization process. Other less common approaches designed to gain primal feasibility are also reviewed in this paper.

In this thesis we consider two levels of primal feasibility: full and partial primal feasibility. In order to define these concepts, let us consider the variable duplicated UC problem (without hydraulic generation for the sake of explanatory simplicity)

$$\min \qquad C_{ther}(p) + C_m(\tilde{p}) \tag{3.43}$$

$$s.t. \qquad \sum_{t=1}^{n_t} p_t^i = L^i, \tag{3.44}$$

$$\sum_{t=1}^{n_t} r_{It}^{\ i} \geq R_I^{\ i}, \tag{3.45}$$

$$\sum_{t=1}^{n_t} r_{Dt}^{\ i} \geq R_D^{\ i}, \tag{3.46}$$

$$i = 1, \ldots, n_i, \tag{3.47}$$

$$(p, r_I, r_D) \in D_{ther}, \quad \widetilde{p} \in D_m, \tag{3.48}$$

$$p - \widetilde{p} = 0, \tag{3.49}$$

where C_{ther}, C_m, D_{ther} and D_m stand for thermal cost, management cost, thermal domain and management domain, respectively.

For $\epsilon > 0$ let D_ϵ be the set

$$\{(p, \widetilde{p}, r_I, r_D) : \quad (p, r_I, r_D) \in D_{ther}, \quad \widetilde{p} \in D_m, \quad \|p - \widetilde{p}\|_\infty < \epsilon\}.$$

We say that a vector $(p, \widetilde{p}, r_I, r_D) \in D_\epsilon$ is Full Primal Feasible (FPF) if fulfills the coupling constraints (3.44–3.46). Regarding the partial feasibility, we say that a vector $(p, \widetilde{p}, r_I, r_D) \in D_\epsilon$ is Load Primal Feasible (LPF), Incremental spinning reserve Primal Feasible (IPF) or Decremental spinning reserve Primal Feasible (DPF) if fulfills constraint (3.44), (3.45) or (3.46) respectively.

After computing an optimal solution $x^* := (p^*, \widetilde{p}^*, r_I{}^*, r_D{}^*)$ to the UC problem, we take the optimal schedule from $\widetilde{p}^*$ since it fulfills all the requirements of the management domain. Unit t is on (off) at interval i if and only if $\widetilde{p}_t^{*i} > 0$ ($\widetilde{p}_t^{*i} = 0$).

Let us see now that at the same time $\widetilde{p}^*$ gives the level of power that approximately solves the associated dispatching problem. If x^* is optimal then $\|p^* - \widetilde{p}^*\|_\infty < \epsilon$, that is,

$$\max_{t=1}^{n_t} \max_{i=1}^{n_i} \mid p_t^{*i} - \widetilde{p}_t^{*i} \mid < \epsilon,$$

i.e.

$$\mid p_t^{*i} - \widetilde{p}_t^{*i} \mid < \epsilon, \text{ for all pair } (t, i).$$

Then

$$p_t^{*i} - \epsilon < \widetilde{p}_t^{*i} < p_t^{*i} + \epsilon, \text{ for all pair } (t, i),$$

which implies

$$\sum_{t=1}^{n_t} p_t^{*i} - n_t\epsilon < \sum_{t=1}^{n_t} \widetilde{p}_t^{*i} < \sum_{t=1}^{n_t} p_t^{*i} + n_t\epsilon, \text{ for all } i,$$

and then considering that p^* satisfies (3.44),

$$L^i - n_t\epsilon < \sum_{t=1}^{n_t} \widetilde{p}_t^{*i} < L^i + n_t\epsilon,$$

i.e.

$$\mid L^i - \sum_{t=1}^{n_t} \widetilde{p}_t^{*i} \mid < n_t\epsilon.$$

Therefore, from a practical point of view, for an $\epsilon > 0$ small enough, we can consider that $\widetilde{p}^*$ also satisfies the load constraint (3.44). For example if $n_t = 100$ and $\epsilon = 10^{-4}$ then $n_t\epsilon = 10^{-2}$.

Let us see now that neither r_I^* fulfills the Incremental Spinning (IS) reserve constraint (3.45) nor r_D^* fulfills the Decremental Spinning (DS) reserve constraint (3.46), that is, $(p^*, \widetilde{p}^*, {r_I}^*, {r_D}^*)$ is a LPF optimizer but not a FPF optimizer.

In chapter 2 we defined the IS reserve as

$$r_{It}^{\,i} := \min\{\bar{r}_{It}, \bar{p}_t^i - p_t^i\}. \tag{3.50}$$

If in an optimal schedule unit t is off at interval i, i.e. $\widetilde{p}_t^{*i} = 0$, then $p_t^{*i} < \epsilon$ (by the stopping criterion $\|p^* - \widetilde{p}^*\|_\infty < \epsilon$). This leads to $\bar{p}_t^i - p_t^{*i} \approx \bar{p}_t^i > 0$ and then

$$r_{It}^{*i} \approx \min\{\bar{r}_{It}, \bar{p}_t^i\} > 0.$$

This contradicts the fact that unit t is off at interval i (only on units can be considered for spinning reserve and the correct value for r_{It}^{*i} would be 0). In consequence, after correcting x^* by setting to 0 all the IS reserve values for off thermal units, it may happen that an optimal solution $(p^*, \widetilde{p}^*, {r_I}^*, {r_D}^*)$ does not fulfill the IS reserve constraint (3.45), that is, the optimal solution is not an IPF solution. Analogously it can be seen that $(p^*, \widetilde{p}^*, {r_I}^*, {r_D}^*)$ neither is an DPF solution. Then our optimal solution is not a FPF optimizer as we wanted to show.

Although our final aim is to obtain FPF solutions, LPF solutions are useful in the context of the stochastic UC problem [TKW00] where one does not consider spinning reserve constraints. Instead, the stochasticity of the problem is dealt with by means of a set of possible scenarios. In chapters 4, 5, 6 and 7 we will compute LPF optimizers, whereas the whole chapter 8 will be devoted to obtain FPF optimizers.

3.6 Summary

We solve the Unit Commitment (UC) problem and the Generalized Unit Commitment (GUC) problem using the Variable Duplication (VD) method instead of the Classical Lagrangian Relaxation (CLR) method. In comparison with the CLR method, the VD method obtains equal or better dual bounds at the price of drastically increasing the number of Lagrange multipliers. The VD method consists of three steps: (1) duplication of the power vector p into $\widetilde{p}$, (2) addition of the artificial equality constraint $p = \widetilde{p}$ and (3) Lagrangian relaxation (augmented or otherwise) of this constraint, which induces a dual problem. In this chapter we have used an augmented Lagrangian relaxation. Then, in the resolution of the dual problem, the augmented Lagrangian splits into two functions by means of the block coordinate descent method; one of these functions is minimized in the hydrothermal distribution domain and the other function is minimized in the thermal management domain. The first minimization is called the hydrothermal distribution subproblem and the second is called the thermal subproblem.

The thermal subproblem is separable giving rise to a sub-subproblem for each thermal unit. The method used to solve the thermal sub-subproblems is forward dynamic programming. The forward dynamic programming algorithm, specialized for the solution of the thermal subproblem, has become the T.FDP algorithm. A very small example illustrates the resolution of the thermal subproblem by the T.FDP algorithm. Finally, the hydrothermal subproblem is solved using a specialized nonlinear network flow code by repeatedly linearizing the nonlinear constraints.

Solutions computed with the described methodology may not be Full Primal Feasible (FPF) since they may fail to fulfill the spinning reserve requirements. The whole chapter 8 will be devoted to obtain FPF solutions.

Chapter 4

Separating the augmented Lagrangian

One of the main drawbacks of the augmented Lagrangian relaxation method is that it introduces a quadratic term, which is not separable. In this chapter we compare empirically and theoretically two methods designed to cope with the non-separability of the Lagrangian function: the Auxiliary Problem Principle method and the Block Coordinate Descent method. We also use the Unit Commitment problem to test both methods.

4.1 The augmented Lagrangian relaxation method

Nowadays the Lagrangian Relaxation (LR) method is the most widespread procedure to solve the Generalized Unit Commitment (GUC) problem. The initial Classical Lagrangian Relaxation (CLR) method was improved in some aspects by the Augmented Lagrangian Relaxation (ALR) method [Ber82, Lue89, Ber95], although advances in the multiplier updating for the CLR method (cutting plane, bundle methods, etc. [HUL96]) have brought the CLR method back into fashion.

The ALR method has two main advantages over the CLR method:

(1) With the ALR method, a solution of the dual problem provides a feasible primal solution in cases where the CLR method yields an infeasible primal solution due to the duality gap (see page 279, volume 1, in [Min83] and [Roc73]). Another related reference is [CZ84] where in section 'Usefulness of augmented Lagrangians' (page 230) we can read 'For nonconvex problems, under essentially the assumption that second-order Kuhn-Tucker conditions hold for the constrained optimization, it has been shown that saddle points of L_c [...] exist for c large enough, but finite. Hence, augmented Lagrangians are a remedy to duality gaps encountered with ordinary Lagrangians for nonconvex problems'. In fact, it can be shown ([HUL96], volume 2, page 184) that 'any integer linear programming problem is equivalent to maximizing the (concave) augmented dual function: an easy problem. The price for this

"miracle" is the (expensive) minimization of the augmented Lagrangian'.

(2) Using the CLR method, the differentiability of the dual function cannot be ensured and therefore a non-differentiable method must be applied in the CLR method. This difficulty can be overcome if an augmented Lagrangian is used, since the dual function q_c is differentiable for an appropriate c (see [Roc73], page 352 in [Ber95] or Theorem 13 in [CZ84]).

At the same time, there are two main advantages of the CLR method over the ALR method: (1) The CLR method gives a separable Lagrangian if the initial cost function is separable as it happens with the GUC problem, however the quadratic term introduced by the augmented Lagrangian is not separable. If we want to solve the GUC problem by decomposition, some method, such as the Auxiliary Problem Principle (APP) method [Coh80] or the Block Coordinate Descent (BCD) method [Ber95], must be used. (2) It is well known that the CLR method, unlike the ALR method, gives a lower bound to the optimal primal cost.

Our starting point is the paper by Batut and Renaud [BR92] and therefore we use variable duplication plus the augmented Lagrangian relaxation method. The aim of the current chapter is to compare the performance of the APP and the BCD methods designed to overcome the inseparability of the augmented Lagrangian. Firstly, a simple example is used to introduce the UC problem, the solution methodologies and the theoretical foundation. Secondly, the same methodologies are applied to solve several large-scale realistic UC instances.

Following [BR92] the Generalized Unit Commitment (GUC) problem can be formulated as ($x \in R^n, \tilde{x} \in R^n$):

$$\left.\begin{array}{ll} \min & f(x) + \tilde{f}(\tilde{x}) \\ s.t. & \quad x \in \mathcal{D}, \quad \tilde{x} \in \tilde{\mathcal{D}} \\ & \quad x - \tilde{x} = 0 \end{array}\right\} \tag{4.1}$$

As we already did in chapter 3, in order to decompose this problem into two subproblems, one in D and the other in $\widetilde{D}$, we relax the coupling constraint $x - \tilde{x} = 0$ using an augmented Lagrangian. That means adding the Lagrangian term $\lambda'(x - \tilde{x})$ and the penalty term $\frac{c}{2}\|x - \tilde{x}\|^2$ to the objective function $f(x) + \tilde{f}(\tilde{x})$. The resulting max-min problem is called the dual problem,

$$\begin{array}{c} \max \\ \lambda \in R^n \end{array} \left\{ \min_{\substack{x \in \mathcal{D} \\ \tilde{x} \in \tilde{\mathcal{D}}}} f(x) + \tilde{f}(\tilde{x}) + \lambda'(x - \tilde{x}) + \tfrac{c}{2}\|x - \tilde{x}\|^2 \right\}. \tag{4.2}$$

As usual, the augmented Lagrangian is defined as

$$L_c(x, \tilde{x}, \lambda) := f(x) + \tilde{f}(\tilde{x}) + \lambda'(x - \tilde{x}) + \frac{c}{2}\|x - \tilde{x}\|^2 \tag{4.3}$$

and the associated dual function is defined as

$$q_c(\lambda) := \left\{ \min_{\substack{x \in \mathcal{D} \\ \tilde{x} \in \tilde{\mathcal{D}}}} L_c(x, \tilde{x}, \lambda) \right\} \tag{4.4}$$

Therefore a compact expression for the dual problem is

$$\begin{array}{c} \max \quad q_c(\lambda) \\ \lambda \in R^n \end{array} \tag{4.5}$$

4.1.1 Compared methods

The focus of this chapter is on the step in which the non-separable augmented Lagrangian is minimized and how we decompose the following problem into smaller subproblems

$$\min_{\substack{x\in\mathcal{D} \\ \tilde{x}\in\tilde{\mathcal{D}}}} f(x) + \tilde{f}(\tilde{x}) + \lambda'(x-\tilde{x}) + \tfrac{c}{2}\|x-\tilde{x}\|^2 \tag{4.6}$$

The method used in [BR92] is the so called Auxiliary Problem Principle (APP) method. Roughly speaking, the APP method linearizes the quadratic term $\frac{c}{2}\|x-\tilde{x}\|^2$ of the augmented Lagrangian at the current iterate $(\mathbf{x_n}, \tilde{\mathbf{x}}_\mathbf{n})$, and adds a quadratic separable term $\frac{b}{2}(\|x-\mathbf{x_n}\|^2 + \|\tilde{x}-\tilde{\mathbf{x}}_\mathbf{n}\|^2)$.

That is

$$\min_{\substack{x\in\mathcal{D} \\ \tilde{x}\in\tilde{\mathcal{D}}}} f(x) + \tilde{f}(\tilde{x}) + \lambda'(x-\tilde{x}) + c(\mathbf{x_n}-\tilde{\mathbf{x}}_\mathbf{n})'(x-\tilde{x}) + \tfrac{b}{2}(\|x-\mathbf{x_n}\|^2 + \|\tilde{x}-\tilde{\mathbf{x}}_\mathbf{n}\|^2). \tag{4.7}$$

The chief utility of the added quadratic term

$$\frac{b}{2}(\|x-\mathbf{x_n}\|^2 + \|\tilde{x}-\tilde{\mathbf{x}}_\mathbf{n}\|^2) = \frac{b}{2}\|(x,\tilde{x})' - (\mathbf{x_n},\tilde{\mathbf{x}}_\mathbf{n})'\|^2,$$

is that it regularizes problem (4.7) in the sense that it makes the (partially linearized) augmented Lagrangian strictly convex ([Ber95] page 475).

Then the minimization of this approximation to L_c breaks into two subproblems ($(\mathbf{x_{n+1}}, \tilde{\mathbf{x}}_\mathbf{n+1})$ being the new iterate):

$$\begin{array}{l} \min \quad f(x) + {\lambda_n}'x + c(\mathbf{x_n}-\tilde{\mathbf{x}}_\mathbf{n})'x + \tfrac{b}{2}\|x-\mathbf{x_n}\|^2 \quad \rightsquigarrow \quad \mathbf{x_{n+1}} \\ x \in \mathcal{D} \end{array} \tag{4.8}$$

$$\begin{array}{l} \min \quad \tilde{f}(\tilde{x}) - {\lambda_n}'\tilde{x} - c(\mathbf{x_n}-\tilde{\mathbf{x}}_\mathbf{n})'\tilde{x} + \tfrac{b}{2}\|\tilde{x}-\tilde{\mathbf{x}}_\mathbf{n}\|^2 \quad \rightsquigarrow \quad \tilde{\mathbf{x}}_\mathbf{n+1} \\ \tilde{x} \in \tilde{\mathcal{D}} \end{array} \tag{4.9}$$

Alternatively, we propose to use the Block Coordinate Descent (BCD) method, also called the nonlinear Gauss-Seidel method. Unlike the APP method, which uses an approximation to L_c, the BCD method directly minimizes L_c. When minimizing in the domain D the variables in the domain $\widetilde{D}$ are frozen to its best known value, say $\tilde{\mathbf{x}}_\mathbf{n}$. Analogously, when minimizing in $\widetilde{D}$ the variables in D are frozen to its best known value, say $\mathbf{x_{n+1}}$, and so on. Then the minimization of L_c decomposes into two subproblems ($(\mathbf{x_{n+1}}, \tilde{\mathbf{x}}_\mathbf{n+1})$ being the new iterate):

$$\begin{array}{l} \min \quad f(x) + {\lambda_n}'x + \tfrac{c}{2}\|x-\tilde{\mathbf{x}}_\mathbf{n}\|^2 \quad \rightsquigarrow \quad \mathbf{x_{n+1}} \\ x \in \mathcal{D} \end{array} \tag{4.10}$$

$$\min_{\tilde{x} \in \tilde{\mathcal{D}}} \ \tilde{f}(\tilde{x}) - \lambda_n{}'\tilde{x} + \tfrac{c}{2}\|\mathbf{x_{n+1}} - \tilde{x}\|^2 \quad \rightsquigarrow \quad \tilde{\mathbf{x}}_{\mathbf{n+1}} \tag{4.11}$$

4.1.2 Compared algorithms

This section uses the well known fact that the gradient of the dual function is equal to the relaxed constraint evaluated at the current iterate, i.e. $\nabla q_{c_n}(\lambda_n) = (x_{n+1} - \tilde{x}_{n+1})$ [Ber95].

The Augmented Lagrangian Relaxation (ALR) method combined with the Auxiliary Problem Principle (APP) method [BR92] is summarized in the following ALR+APP algorithm (the ALR algorithm is also called the multiplier method):

ALR+APP algorithm

Step 0 [Initialize.] $\epsilon > 0, x_1, \tilde{x}_1, \lambda_1, c_1, b_1 \geq 2c_1$ and $n = 1$.

Step 1 [Check the stopping criterion.] If the norm of the gradient of the dual function $\|x_n - \tilde{x}_n\| < \epsilon$ then stop. $(x_n, \tilde{x}_n, \lambda_n)$ is a primal-dual solution.

Step 2 [Compute x_{n+1}.]

$$\min_{x \in \mathcal{D}} \ f(x) + \lambda_n' x + c_n(x_n - \tilde{x}_n)'x + \tfrac{b_n}{2}\|x - x_n\|^2$$

Step 3 [Compute $\tilde{x}_{n+1}$.]

$$\min_{x \in \tilde{\mathcal{D}}} \ \tilde{f}(\tilde{x}) - \lambda_n' \tilde{x} - c_n(x_n - \tilde{x}_n)'\tilde{x} + \tfrac{b_n}{2}\|\tilde{x} - \tilde{x}_n\|^2$$

Step 4 [Dual variable updating.]

$$\lambda_{n+1} = \lambda_n + c_n \cdot (x_{n+1} - \tilde{x}_{n+1})$$

Step 5 [Penalty parameter updating.] If $\|x_{n+1} - \tilde{x}_{n+1}\| > \alpha \cdot \|x_n - \tilde{x}_n\|$ then $c_{n+1} := \beta \cdot c_n$. A suitable choice is $\alpha := 1.10$ and $\beta \in]1, 2]$. Update b_{n+1} such that $b_{n+1} \geq 2c_{n+1}$. Set $n = n + 1$ and go back to step 1.

Analogously, the ALR method combined with the Block Coordinate Descent (BCD) method is summarized in the following ALR+BCD algorithm.

ALR+BCD algorithm

Step 0 [Initialize.] Initialize $\epsilon > 0, x_1, \tilde{x}_1, \lambda_1, c_1$ and $n = 1$.

Step 1 [Check the stopping criterion.] If the norm of the gradient of the dual function $\|x_n - \tilde{x}_n\| < \epsilon$ then stop. $(x_n, \tilde{x}_n, \lambda_n)$ is a primal dual solution.

Step 2 [Compute x_{n+1}.]

$$\min_{x \in \mathcal{D}} \; f(x) + \lambda_n' x + \tfrac{c_n}{2}\|x - \tilde{x}_n\|^2$$

Step 3 [Compute $\tilde{x}_{n+1}$.]

$$\min_{x \in \tilde{\mathcal{D}}} \; \tilde{f}(\tilde{x}) - \lambda_n' \tilde{x} + \tfrac{c_n}{2}\|x_{n+1} - \tilde{x}\|^2$$

Step 4 [Dual variable updating.]

$$\lambda_{n+1} = \lambda_n + c_n \cdot (x_{n+1} - \tilde{x}_{n+1})$$

Step 5 [Penalty parameter updating.] If $\|x_{n+1} - \tilde{x}_{n+1}\| > \alpha \cdot \|x_n - \tilde{x}_n\|$ then $c_{n+1} := \beta \cdot c_n$. A suitable choice is $\alpha := 1.10$ and $\beta \in]1, 2]$. Set $n = n + 1$ and go back to step 1.

Theoretically, in the ALR+BCD algorithm, steps 2 and 3 should be repeated until convergence is reached at a minimizer of $L_c(x, \tilde{x}, \lambda_n)$ within $\mathcal{D} \times \tilde{\mathcal{D}}$. However, given that λ_n is only an approximation to the optimal λ^*, our computational experience shows that CPU time can be saved if this repetition is not performed.

Actually, this inexact minimization of the augmented Lagrangian is a classical variation that can be found, for example, in [Ber95], page 347 or in [GMW95], page 230.

Another important issue is the convergence of the methods above described. A convergence proof of the ALR method, also known in literature as the multiplier method, can be found in [Ber95], page 349. This proof assumes that the augmented Lagrangian is minimized in the whole space R^n. In [Roc73] a convergence proof for the ALR method is also given assuming that the augmented Lagrangian is minimized over a convex set and also assuming convex constraint functions. Unfortunately, in our approach to solve the Generalized Unit Commitment (GUC) problem we neither minimize the augmented Lagrangian in the whole space R^n nor in a convex set, but in a disconnected set. Thus, we use the ALR method to solve the GUC problem, as other authors did [AaS98, BR92, Bal95, WS$^+$95], without having a convergence proof. The same could be said about the BCD method and the APP method. In [Ber95], page 246, we can found a convergence proof for the BCD method, which requires convexity of both the domain set and the objective function. Of course, we still are further of having a convergence proof for the combined ALR+BCD method in the GUC case. [Coh80] gives a convergence proof of the ALR+APP method restricted to the convex case as well.

However, although it is very important and desirable to study and know the convergence of these methods, the fact of having only convergence proofs for the convex case, as G. Cohen says ([Coh80], page 278), 'does not prevent anybody from using the algorithms in practical nonconvex cases, since, if convergence results, the necessary conditions of optimality are generally met in the limit'. From a practical point of view, it is important to remember that the relative duality gap for large-scale instances of the GUC problem is usually small ([Ber95], page 431). Thus, for example, the GUC solutions obtained in chapters 6 and 7, in almost every case, have a relative duality gap lower than 3.5%. Therefore, even if we cannot guarantee convergence of any of the above methods to an optimizer, we can assess the computed solutions by means of its duality gap.

4.1.3 First computational test

In this first test we use an n dimensional version of the simple Unit Commitment (UC) problem presented in chapter 3. Note that in this case we have a simple UC problem with only one interval.

Here n is the number of thermal units, x_i is the output of thermal unit i, Con_i is the starting cost, i.e. the cost of turning the unit i on, d is the demand for electrical power, and l_i and u_i are lower and upper bounds for x_i:

$$\left.\begin{array}{ll} \min & \sum_{i=1}^{n}(2x_i^2 + Con_i(x_i)), \\ s.t. & \sum_{i=1}^{n} x_i = d, \\ & x_i \in \{0\} \cup [l_i, u_i] \quad i = 1, \dots, n, \end{array}\right\} \tag{4.12}$$

where arbitrarily we choose

$$Con_i(x_i) := \begin{cases} 10 + (i-1)\frac{10}{n-1} & \text{if} \quad x_i \in [l_i, u_i], \\ 0 \quad \text{if} \quad x_i = 0. \end{cases} \tag{4.13}$$

Given that the starting cost function Con_i is monotone on i and that the generating cost is the same for all units ($2x_i^2 \quad i = 1, \dots, n$), this problem can exactly be solved (algebraic method). For example, let us solve problem (4.12) for the particular case of 3 thermal units ($n = 3$), demand $d = 6$, lower bound $l_i = 1$ and upper bound $u_i = 6$ ($i = 1, 2, 3$). That is

$$\left.\begin{array}{ll} \min & \sum_{i=1}^{3}(2x_i^2 + Con_i(x_i)), \\ s.t. & x_1 + x_2 + x_3 = 6, \\ & x_i \in \{0\} \cup [1, 6] \quad i = 1, 2, 3, \end{array}\right\} \tag{4.14}$$

with

$$Con_i(x_i) := \begin{cases} k_i := 10 + \frac{10}{3-1}(i-1) & \text{if} \quad x_i \in [1, 6], \\ 0 \quad \text{if} \quad x_i = 0. \end{cases} \tag{4.15}$$

Note that $k_1 = 10$, $k_2 = 15$, and that $k_3 = 20$, i.e. the starting cost is monotone on i as we have already pointed out. If n^* is defined as the optimal number of thermal units, then a priori there are three possibilities: $n^* = 1, 2$ or 3.

If $n^* = 1$, given that unit 1 has the cheapest starting cost ($k_1 = 10$) and the generating cost ($2x_i^2 \ \ i = 1, 2, 3$) is the same for the three units, then the optimal solution would be $x^* = (6, 0, 0)$, with an optimal cost of $2 \cdot 6^2 + 10 = 82$. Analogously, if $n^* = 2$, then it is easy to see that $x^* = (3, 3, 0)$ (units 1 and 2 are the cheapest ones) and, in this case, the optimal cost would be $2 \cdot 3^2 + 10 + 2 \cdot 3^2 + 15 = 61$. Finally, if $n^* = 3$, then it also is easy to see that $x^* = (2, 2, 2)$ and the optimal cost would be $2 \cdot 2^2 + 10 + 2 \cdot 2^2 + 15 + 2 \cdot 2^2 + 20 = 69$. Thus, the optimal solution is $n^* = 2$, $x^* = (3, 3, 0)$ with optimal cost 61. Note that we have computed a *global* optimizer.

The parameters used in this first computational test are: $n = 10, 20, 30, \cdots, 100$; $d = n$; $l_i = 1$ and $u_i = n$. Regarding the tuning parameters for both the ALR+APP and the ALR+BCD algorithms, the initial penalty c_0 has been set equal to 1, 0.1 and 0.05 for each method in three different trials. The reported results correspond to the best trial (lowest local optimum). Table 4.1 displays the best c_0 for each method and each case. The parameter b_n has been set equal to $2c_n$, and the scalars α and β equal to 1.1 and 2 respectively.

This first test is performed in three steps: in the first step, we solve the UC problem exactly for 10 cases ($n = 10, 20, 30, \cdots, 100$) by the above algebraic method in order to know its global optimizer. The optimal number of thermal units and the optimal cost, obtained with the algebraic method, are displayed in Table 4.2 and Table 4.3 respectively (label ALG).

In the second step we solve the same 10 problems using the 'Augmented Lagrangian Relaxation (ALR) method + Block Coordinate Descent (BCD) method', and the 'ALR method + Auxiliary Problem Principle (APP) method' to check the quality of the computed optimizers (are they local or global optimizers?). The relative error $100 * (f_{computed} - f^*)/f^*$ of the computed solution is displayed for the BCD and the APP methods in Table 4.3. For example, in the first case ($n = 10$) the two methods reach the global optimizer, and therefore, the relative error is zero. However, in case $n = 70$ neither method reaches the global optimizer. Table 4.3 shows that these methods obtain either the global optimizer or a high-quality local optimizer (relative error always under 0.5%). On average, the relative error of the computed optima is 0.07% for both the APP method and the BCD method.

In the third step, now that we know that the quality of the solutions is similar for the two methods, we investigate their performance. The CPU time for the two methods can be found in Table 4.4 as well as the time ratio between the APP method versus the BCD method. In most cases the BCD method is the fastest one. Thus, for example, in the first case ($n = 10$), the APP time is 17% greater than the BCD time (see label 'Time Ratio'/APP:BCD). On average, the APP method needs 33% more time than the BCD method.

Table 4.1: Initial penalty parameter c_0.

Thermal units	c_0	
	APP	BCD
10	0.10	0.10
20	0.05	0.05
30	0.10	0.10
40	1.00	0.10
50	1.00	0.10
60	0.10	0.05
70	0.10	0.10
80	0.10	0.05
90	0.10	0.10
100	0.10	0.10

Table 4.2: Optimal number of thermal units.

Thermal units	Optimal number of thermal units		
	ALG	APP	BCD
10	4	4	4
20	8	7	8
30	11	11	11
40	15	16	15
50	19	20	19
60	23	23	22
70	27	26	29
80	30	30	31
90	34	33	33
100	38	38	37
Average	20.9	20.8	20.9

Table 4.3: Quality of the computed optimizers.

Thermal units	Optimum			Relative error (%)	
	ALG	APP	BCD	APP	BCD
10	96.67	96.70	96.67	0.03	0.00
20	194.74	195.38	194.74	0.33	0.00
30	292.60	292.60	292.60	0.00	0.00
40	390.60	390.91	390.60	0.08	0.00
50	488.06	488.84	488.10	0.16	0.01
60	585.92	585.92	586.41	0.00	0.08
70	683.83	684.02	686.82	0.03	0.44
80	781.73	781.73	781.76	0.00	0.00
90	879.50	880.20	880.20	0.08	0.08
100	977.33	977.33	977.85	0.00	0.05
Average	537.10	537.36	537.58	0.07	0.07

Table 4.4: The APP method versus the BCD method.

Thermal units	Time (seconds)		Time ratio
	APP	BCD	APP:BCD
10	2.7	2.3	1.17
20	6.8	7.9	0.86
30	9.6	5.4	1.78
40	18.0	11.1	1.62
50	29.3	19.5	1.50
60	38.0	36.4	1.04
70	49.0	68.0	0.72
80	66.6	89.3	0.75
90	91.3	49.9	1.83
100	149.3	74.5	2.00
Average	46.1	36.4	1.33

4.1.4 Theoretical insight

In the last section the BCD method has shown to be the fastest one from an empirical point of view. In this section we compare the BCD method to the APP method from a theoretical point of view.

Proposition 4.1 *A single iteration of the Auxiliary Problem Principle (APP) method is equivalent to:*

$$\mathbf{x_{n+1}} := \underset{x \in \mathcal{D}}{\operatorname{argmin}}\, f(x) + {\lambda_n}'x + \tfrac{c}{2}\|x - \tilde{\mathbf{x}}_\mathbf{n}\|^2 + \tfrac{b-c}{2}\|x - \mathbf{x_n}\|^2, \tag{4.16}$$

$$\tilde{\mathbf{x}}_\mathbf{n+1} := \underset{\tilde{x} \in \tilde{\mathcal{D}}}{\operatorname{argmin}}\, \tilde{f}(\tilde{x}) - {\lambda_n}'\tilde{x} + \tfrac{c}{2}\|\mathbf{x_n} - \tilde{x}\|^2 + \tfrac{b-c}{2}\|\tilde{\mathbf{x}}_\mathbf{n} - \tilde{x}\|^2, \tag{4.17}$$

where $(\mathbf{x_n}, \tilde{\mathbf{x}}_\mathbf{n})$ *is the input current iterate and* $(\mathbf{x_{n+1}}, \tilde{\mathbf{x}}_\mathbf{n+1})$ *is the output next iterate.*

Proof. Given that

$$c(\mathbf{x_n} - \tilde{\mathbf{x}}_\mathbf{n})'x + \frac{b}{2}\|x - \mathbf{x_n}\|^2 = \tag{4.18}$$

$$c{\mathbf{x_n}}'x - c\tilde{\mathbf{x}}'_\mathbf{n}x + \frac{b}{2}\|x\|^2 + \frac{b}{2}\|\mathbf{x_n}\|^2 - bx'\mathbf{x_n} \pm \frac{c}{2}\|x\|^2 \pm \frac{c}{2}\|\mathbf{x_n}\|^2 \pm \frac{c}{2}\|\tilde{\mathbf{x}}_\mathbf{n}\|^2 = \tag{4.19}$$

$$= \frac{c}{2}\|x\|^2 + \frac{c}{2}\|\tilde{\mathbf{x}}_\mathbf{n}\|^2 - c\tilde{\mathbf{x}}'_\mathbf{n}x + \frac{b-c}{2}\|x\|^2 + \frac{b-c}{2}\|\mathbf{x_n}\|^2 - (b-c){\mathbf{x_n}}'x + \frac{c}{2}\|\mathbf{x_n}\|^2 - \frac{c}{2}\|\tilde{\mathbf{x}}_\mathbf{n}\|^2 = \tag{4.20}$$

$$= \frac{c}{2}\|x - \tilde{\mathbf{x}}_\mathbf{n}\|^2 + \frac{b-c}{2}\|x - \mathbf{x_n}\|^2 + \frac{c}{2}(\|\mathbf{x_n}\|^2 - \|\tilde{\mathbf{x}}_\mathbf{n}\|^2). \tag{4.21}$$

Then the minimization over D

$$\min_{x \in \mathcal{D}}\ f(x) + {\lambda_n}'x + c(\mathbf{x_n} - \tilde{\mathbf{x}}_\mathbf{n})'x + \tfrac{b}{2}\|x - \mathbf{x_n}\|^2, \tag{4.22}$$

can be rewritten as

$$\min_{x \in \mathcal{D}}\ f(x) + {\lambda_n}'x + \tfrac{c}{2}\|x - \tilde{\mathbf{x}}_\mathbf{n}\|^2 + \tfrac{b-c}{2}\|x - \mathbf{x_n}\|^2 + \tfrac{c}{2}(\|\mathbf{x_n}\|^2 - \|\tilde{\mathbf{x}}_\mathbf{n}\|^2). \tag{4.23}$$

Analogously the minimization over $\tilde{D}$

$$\min_{x \in \tilde{\mathcal{D}}}\ \tilde{f}(\tilde{x}) - {\lambda_n}'\tilde{x} - c(\mathbf{x_n} - \tilde{\mathbf{x}}_\mathbf{n})'\tilde{x} + \tfrac{b}{2}\|\tilde{x} - \tilde{\mathbf{x}}_\mathbf{n}\|^2, \tag{4.24}$$

can be rewritten as

$$\min_{x \in \tilde{\mathcal{D}}}\ \tilde{f}(\tilde{x}) - {\lambda_n}'\tilde{x} + \tfrac{c}{2}\|\mathbf{x_n} - \tilde{x}\|^2 + \tfrac{b-c}{2}\|\tilde{x} - \tilde{\mathbf{x}}_\mathbf{n}\|^2 + \tfrac{c}{2}(\|\tilde{\mathbf{x}}_\mathbf{n}\|^2 - \|\mathbf{x_n}\|^2). \tag{4.25}$$

Now, the constant terms $\frac{c}{2}(\|\mathbf{x_n}\|-\|\tilde{\mathbf{x}}_\mathbf{n}\|)$ and $\frac{c}{2}(\|\tilde{\mathbf{x}}_\mathbf{n}\|-\|\mathbf{x_n}\|)$ of the objective functions in (4.23) and (4.25) can be removed without changing the optimal iterate. Therefore a single iteration of the APP method is equivalent to

$$\mathbf{x_{n+1}} := \underset{x \in \mathcal{D}}{\text{argmin}}\, f(x) + \lambda_n' x + \tfrac{c}{2}\|x - \tilde{\mathbf{x}}_\mathbf{n}\|^2 + \tfrac{b-c}{2}\|x - \mathbf{x_n}\|^2, \tag{4.26}$$

$$\tilde{\mathbf{x}}_\mathbf{n+1} := \underset{\tilde{x} \in \tilde{\mathcal{D}}}{\text{argmin}}\, \tilde{f}(\tilde{x}) - \lambda_n'\tilde{x} + \tfrac{c}{2}\|\mathbf{x_n} - \tilde{x}\|^2 + \tfrac{b-c}{2}\|\tilde{\mathbf{x}}_\mathbf{n} - \tilde{x}\|^2 \tag{4.27}$$

as we wanted to prove. □

Corollary 4.1 *For $b = c$ the APP method is nothing but the Jacobi version of the nonlinear Gauss-Seidel or BCD method.*

Proof. This is directly shown by comparing proposition 4.1 with the BCD method below (nonlinear Gauss-Seidel method).

$$\mathbf{x_{n+1}} := \underset{x \in \mathcal{D}}{\text{argmin}}\, f(x) + \lambda_n' x + \tfrac{c}{2}\|x - \tilde{\mathbf{x}}_\mathbf{n}\|^2, \tag{4.28}$$

$$\tilde{\mathbf{x}}_\mathbf{n+1} := \underset{\tilde{x} \in \tilde{\mathcal{D}}}{\text{argmin}}\, \tilde{f}(\tilde{x}) - \lambda_n'\tilde{x} + \tfrac{c}{2}\|\mathbf{x_{n+1}} - \tilde{x}\|^2. \tag{4.29}$$

Note that, for $b = c$, last term in (4.16) and (4.17) vanishes. Therefore, the difference between the two methods is only that in (4.17) the APP method uses the old iterate $\mathbf{x_n}$ (Jacobi method) whereas in (4.29) the BCD method uses the new iterate $\mathbf{x_{n+1}}$ computed in (4.28) (Nonlinear Gauss-Seidel method).□

It is well known in numerical analysis that the nonlinear Gauss-Seidel method is likely to outperform the Jacobi method [Act90, BF93, DB74]. The main reason for that is the nonlinear Gauss-Seidel method immediately incorporates the newly generated information $\mathbf{x_{n+1}}$ within the current iteration (as in (4.11)) whereas the Jacobi method incorporates this new information at the beginning of the next iteration and uses $\mathbf{x_n}$ instead (as in (4.17)). Thus, a first consequence of the precedent corollary is that a faster convergence to the optimum can be expected using the BCD method than using the APP method for the particular case $b = c$.

Let us now study the APP method for the more general case $b > c$. In proposition 4.1, if we consider the minimization of the objective function in $\mathcal{D} \times \tilde{\mathcal{D}}$, the terms $\frac{b-c}{2}\|x - \mathbf{x_n}\|^2$ and $\frac{b-c}{2}\|\tilde{\mathbf{x}}_\mathbf{n} - \tilde{x}\|^2$ in (4.16) and (4.17), respectively, can be joined as

$$\frac{b-c}{2}\|(x, \tilde{x})' - (\mathbf{x_n}, \tilde{\mathbf{x}}_\mathbf{n})'\|^2. \tag{4.30}$$

For $b = c$ the term (4.30) vanishes, but for any $b > c$ it penalizes iterates $(\mathbf{x_{n+1}}, \tilde{\mathbf{x}}_\mathbf{n+1})$ that lie far from $(\mathbf{x_n}, \tilde{\mathbf{x}}_\mathbf{n})$. Therefore, the greater b is the closer $(\mathbf{x_{n+1}}, \tilde{\mathbf{x}}_\mathbf{n+1})$ will be to $(\mathbf{x_n}, \tilde{\mathbf{x}}_\mathbf{n})$. Then, in general, it is likely that for $b > c$ the APP method will take more iterations to converge to a particular optimizer than for $b = c$ since $b > c$ penalizes long steps from the current iterate.

For the reasons given above and considering that in the APP method b must fulfill the condition $b \geq 2c$, then we can expect faster performance of the BCD method in comparison with the APP method.

4.2 Unit commitment: a realistic example

Chapters 1 and 2 were devoted to introducing and modeling the Generalized Unit Commitment (GUC) problem. The next two subsections summarize those chapters for the reader that skipped them.

In subsection 4.1.3 we have just solved the Unit Commitment (UC) problem for one interval. In practical situations one solves the UC problem for several intervals (from 24 to 168 hourly intervals). This problem has been thoroughly studied in the past by many authors [MS83, VAL90]. Nowadays, researchers try to solve enhanced versions of the UC problem, where in addition to the system of thermal units, one simultaneously takes into account other related systems, such as the hydroelectric plants and the transmission network [BR92, JC99, PRS96]. This is the Generalized Unit Commitment (GUC) problem.

The objective of the GUC problem is to optimize electrical production and distribution, considering a short-term planning horizon (from one day to one week). Hydraulic and thermal plants must be coordinated in order to satisfy the customer demand of electricity at the minimum cost and with a reliable service. The model for the GUC problem presented here considers the thermal system, the hydraulic system and the distribution network.

4.2.1 Formulation of the generalized unit commitment problem

The optimization problem here considered is:

$$\left.\begin{array}{ll} \min & f(x) = C_{htd}(x) + C_m(x) \\ s.t. & x \in \mathcal{D}_{htd} \\ & x \in \mathcal{D}_m \end{array}\right\} \qquad (4.31)$$

where $\mathcal{D}_{htd}$ represents the domain defined by the constraints that couple the hydraulic, thermal and distribution systems: load constraints, spinning reserve constraints, etc., $\mathcal{D}_m$ represents the management domain of the thermal units: minimum up and down-times, minimum and maximum output levels, etc., $C_{htd}(x)$ represents the costs associated with $\mathcal{D}_{htd}$ and $C_m(x)$ represents the costs associated with $\mathcal{D}_m$.

4.2.2 Modeling the generalized unit commitment problem

The general expression of the GUC problem (4.31) could be developed in several different ways. Our approach follows the so called *Coupled Model* presented in [HN95]. This model takes into account the hydroelectric generation system together with the thermal system and the transmission network. The variable vector x in problem (4.31) is composed of a series of subvectors, each one representing different magnitudes, that is,

$$x = (d, v, s, p, r_I, r_D, g_I, g_D, q, g)$$

where we have reservoir discharges d, reservoir volumes v, reservoir spillages s, thermal output powers p, incremental spinning reserves r_I, decremental spinning reserves r_D, incremental gaps g_I, decremental gaps g_D, transmission powers q, and hydraulic output powers g. In the coupled model the constraints relating all these variables (domain $\mathcal{D}_{htd}$ of problem (4.31)) are expressed through a network flow model with side constraints:

$$\begin{aligned}
A_{HTTE} \cdot x &= b_{HTTE} && (4.32)\\
h_G(d, v, g) &= 0 && (4.33)\\
T_{ISR} \cdot \begin{pmatrix} r_I \\ g \end{pmatrix} &\geq b_{ISR} && (4.34)\\
T_{DSR} \cdot \begin{pmatrix} r_D \\ g \end{pmatrix} &\geq b_{DSR} && (4.35)\\
T_{KVL} \cdot q &= 0 && (4.36)\\
\underline{x} \leq x &\leq \bar{x} && (4.37)
\end{aligned}$$

where

(4.32) gives the network constraints associated with the so called *Hydrothermal Transmission Extended Network* (HTTEN). The HTTEN integrates the replicated hydro network, which accounts for the time and space coupling between the reservoirs of the river basin, the *thermal equivalent network*, which defines the relation between the power output and the spinning reserve level of each thermal unit, and the transmission network, which formulates the conservation of the power flow at the busses of the transmission system.

(4.33) represents the nonlinear side constraints that define the injection of the hydroelectric generation (a nonlinear function of the variables d, v and g) into the appropriate busses of the transmission network. The solution procedure will be based on a successive linearization of these constraints.

(4.34-4.35) are two sets of linear side constraints that impose the satisfaction of the incremental and decremental spinning reserve requirements of the whole system.

(4.36) is a set of linear side constraints that formulates the Kirchoff Voltage Law. These constraints, together with the power flow conservation equations formulated in (4.32), represent a direct current approach to the transmission network.

(4.37) shows the upper and lower bounds.

The formulation of the domain $\mathcal{D}_{htd}$ as a network flow problem with side constraints allows the use of specialized network optimization codes. Also, the flexibility of this model is such that any other relevant system constraints, for instance, security constraints and emission constraints, can easily be added (see [CNH95]).

The thermal management domain $\mathcal{D}_m$ of problem (4.31) deals with the physical and economic constraints of the thermal units. First, the minimum up and down-times of each thermal unit must be respected, otherwise too many on/off switches will exert wear and tear on the system. Second, for mechanical reasons each unit has a minimum and maximum output level that bounds the production of each unit.

The first term of the objective function of (4.31), $C_{htd}(x)$, represents 50% of the fuel cost of the thermal units, and it is modeled as a quadratic function of the thermal output power p. This term can also include an estimation of the cost of the power losses through a quadratic function of the transmission power q. The second part, $C_m(x)$, includes the remaining 50 % of the fuel cost, the start-up and shut-down costs of the thermal units, and depends only on the power output p.

4.2.3 Second computational test

In this second test, 7 instances of the Unit Commitment (UC) problem are solved. In Table 4.5 we present these instances, and in Table 4.6 we describe their main features, which range from 4 to 1176 binary variables and from very small size (2 intervals, 0 reservoirs and 2 thermal units) up to medium size (168 intervals, 4 reservoirs and 7 thermal units). The UC problems solved here consider the hydraulic and thermal systems.

Unlike in the first test, obviously we cannot know a priori the real optimizer for these large-scale nonlinear mixed integer problems. Therefore, this second test consists of only two steps: Step 1), we solve the same 7 problems using the 'Augmented Lagrangian Relaxation (ALR) method + Block Coordinate Descent (BCD) method' and the 'ALR method + Auxiliary Problem Principle (APP) method' to check the quality of the computed optimizers. Step 2), we compare the efficiency of the ALR+BCD and the ALR+APP methods in terms of CPU time.

The tuning parameters used in the ALR+APP and the ALR+BCD algorithms are: stopping criterion, $\epsilon = 10^{-4}$; initial penalty parameter c_0, as listed in Table 4.7; and the parameters used for updating c_0 are $\alpha = 1.10$ and $\beta = 2$ (only in case 7, $\beta = 1.1$ since with $\beta = 2$ the BCD method did not converge). The ALR+APP algorithm requires the additional parameter b_n, which has been set equal to $2 \cdot c_n$ in all cases.

Results of step 1): Comparing the 'Cost' columns in Table 4.8 we observe that the quality of the computed optimizers is very similar for both algorithms.

Results of step 2): Now that we know the quality of the solutions is very similar for both methods, we investigate their performance. The relative CPU time $Time_{APP}/Time_{BCD}$ is displayed in Table 4.8 (column 'Ratio'). For example, in the

Table 4.5: Solved UC instances.

Case	Hydraulic data file	Thermal data file
1	hz020	ti0200294
2	hp060	ti0400691
3	hp480	thunit40404891
4	reda481	thunit40704896
5	hp480	thunit40704891
6	reda160	thunit40216891
7	reda160	thunit40716891

Table 4.6: Description of the UC instances.

Case	Number intervals	Number reservoirs	Thermal units	Continuous variables	Binary variables
1	2	0	2	16	4
2	6	2	4	138	24
3	48	2	4	1104	192
4	48	4	7	1920	336
5	48	2	7	1680	336
6	168	4	2	3360	336
7	168	4	7	6720	1176

Table 4.7: Initial penalty parameter c_0.

Case	1	2	3	4	5	6	7
c_0	10^{-1}	10^{-3}	10^{-3}	10^{-4}	10^{-3}	10^{-4}	10^{-3}

Table 4.8: Results using the APP or the BCD algorithms.

Case	Iterations		CPU time (seconds)			Cost ($\times 10^6 PTA$)	
	APP	BCD	APP	BCD	Ratio	APP	BCD
1	30	17	12.8	7.2	1.78	0.004	0.004
2	122	139	41.4	50.7	0.82	7.053	7.055
3	66	43	34.7	23.2	1.50	0.991	0.991
4	83	60	65.7	50.4	1.30	6.364	6.348
5	35	43	33.8	27.1	1.25	0.989	0.989
6	62	45	114.3	90.3	1.27	4.439	4.443
7	119	90	734.0	640.0	1.15	2.612	2.600
Average	73.9	62.4	148.1	127.0	1.30	3.207	3.204

first case the APP method takes 78% more time than the BCD method. Although the APP method can perform faster in some instances as in case 2, in most cases the BCD method performs faster. In column 'Ratio' we observe that, on average, the APP method takes 30% more time than the BCD method in this particular test, as expected from a theoretical point of view (see section 4.1.4).

4.3 Summary

Load Primal Feasible (LPF) solutions for the unit commitment problem have been obtained using the Variable Duplication plus Augmented Lagrangian Relaxation (ALR) method. Theoretically and practically, the Block Coordinate Descent Method is shown to be faster than the Auxiliary Problem Principle method to deal with the non-separable augmented Lagrangian.

With the above methodology (ALR method) one obtains either a local or a global LPF optimizer. The quality of the computed solution is unknown and therefore more research is needed. Chapter 6 is devoted to the improvement of the solutions and the measure of their quality.

Chapter 5

Multiplier updating by the radar method

This chapter addresses the issue of Lagrange multiplier updating. First, there is a brief overview of the main updating methods, in which we distinguish two different frameworks: classical and augmented Lagrangian relaxations. Second, a new multiplier updating method, the Radar Gradient (RG) method, is geometrically motivated, algebraically deduced and computationally tested. The RG method is compared with the multiplier method. And third, given that we are updating a set of multipliers in order to maximize a dual function which may not be differentiable, the RG method is extended to the non-differentiable case, the Radar Subgradient (RS) method. The RS method is compared with the subgradient method from a practical point of view.

5.1 Brief overview of Lagrange multiplier updating methods

Lagrange relaxation methods can be classified into two families depending on the perturbation of the cost function: the Classical Lagrangian Relaxation (CLR) family and the Augmented Lagrangian Relaxation (ALR) family. As they depend on the Lagrange relaxation performed, Lagrange multiplier updating methods will likewise fall into one of the two families.

5.1.1 Classical Lagrangian relaxation

Suppose we wish to solve the following primal problem (P):

$$\left.\begin{array}{rl} \min & f(x) \\ s.t. & h(x) = 0 \\ & x \in \mathcal{D} \end{array}\right\} \tag{5.1}$$

where $f(x) : R^n \longmapsto R$ and $h(x) : R^n \longmapsto R^m$, and $\mathcal{D}$ is a compact set.

The (Lagrangian) dual problem (D) of (P) is

$$\begin{array}{c} \max \\ \lambda \in R^m \end{array} \left\{ \begin{array}{c} \min \\ x \in \mathcal{D} \end{array} \; f(x) + \lambda' h(x) \right\}. \tag{5.2}$$

Equivalently, since the Lagrangian function $L(x, \lambda) := f(x) + \lambda' h(x)$, the dual problem can be written as

$$\begin{array}{c} \max \\ \lambda \in R^m \end{array} \left\{ \begin{array}{c} \min \\ x \in \mathcal{D} \end{array} \; L(x, \lambda) \right\}. \tag{5.3}$$

For short, defining the dual function $q(\lambda) := \min \{L(x, \lambda) : x \in \mathcal{D}\}$, the dual problem has the expression

$$\begin{array}{c} \max \\ \lambda \in R^m \end{array} \quad q(\lambda) \tag{5.4}$$

It is well known that the dual function $q(\lambda)$ is a concave function but not necessarily differentiable. If there is no duality gap, that is, if the optimum values of the primal problem (P) and the dual problem (D) are equal, then a solution of the dual problem provides a solution of the primal problem. We solve the dual problem (D) whenever it is easier to solve than the primal problem (P) and there is no duality gap. However, even if there is a duality gap the solution of the dual problem provides a lower bound to the optimal cost that can be useful, for example, in combinatorial optimization. More general details can be found in [Min83, Lue89, Ber95] and a more in-depth theoretical point of view in [HUL96]. The following Classical Lagrangian Relaxation (CLR) algorithm solves the dual problem (D).

Classical Lagrangian relaxation algorithm

- [Method.] Classical Lagrangian relaxation plus resolution of the associated dual problem.
- [Objective.] To solve the primal problem (5.1) or give a lower bound to its optimal primal cost.
- [Input.] λ_0 estimate of the Lagrange multipliers.

* [Output.] (x^*, λ^*) primal-dual solution.

Step 0 [Initialize.] Set $n = 0$.

Step 1 [Compute the dual function for λ_n.]

$$\left.\begin{array}{cc} \min & f(x) + \lambda_n' \cdot h(x) \\ x \in \mathcal{D} & \end{array}\right\} \tag{5.5}$$

If x_n is the calculated solution then $h(x_n)$ is a subgradient of $q(\cdot)$ in λ_n [Ber95].

Step 2 [Check the stopping criterion.] If λ_n is optimal then stop. Without a duality gap, (x_n, λ_n) is a primal-dual solution.

Step 3 [Multiplier updating.] Compute λ_{n+1} (use the subgradient method, cutting plane method, bundle method, etc.)

Step 4 [Update the iteration count.] Set $n = n + 1$ and go back to step 1.

Different versions of the CLR algorithm arise depending on the method used in step 3. There are three main updating methods [JC99] (although we could talk about families of methods since every method has its own variants).

First, in the subgradient method [MS83, NGL99, Ber95, HUL96], $h(x_n)$ being a subgradient of $q(\lambda_n)$, the multipliers are updated as follows:

$$\lambda_{n+1} = \lambda_n + \alpha_n \cdot \frac{h(x_n)}{\|h(x_n)\|}, \tag{5.6}$$

where, if $\lim_{n\to\infty} \alpha_n = 0$ and $\sum_{n=0}^{\infty} \alpha_n = +\infty$, convergence is guaranteed. If the dual function is non-differentiable, the subgradient method progresses slowly to the optimum in an oscillating fashion. In this case, it is very difficult to devise an appropriate stopping criterion and typically it is stopped after a pre-specified number of iterations.

Second, in the cutting plane method [Ber95, HUL96], the new multiplier λ_{n+1} is obtained by solving the following linear problem:

$$\left.\begin{array}{rl} \max & z \\ s.t. & z \leq q(\lambda_k) + h(x_k)'(\lambda - \lambda_k) \\ & \quad k = 1, \ldots, n \\ & \lambda \in \mathcal{C} \end{array}\right\}, \tag{5.7}$$

where $\mathcal{C}$ is a convex and compact set. The cutting plane method obtains a dual optimum by iteratively approximating the dual function with the hyperplanes $z(\lambda) = q(\lambda_k) + h(x_k)' \cdot (\lambda - \lambda_k)$. It does not suffer from oscillations, unlike the subgradient method, but as the number of multiplier updating raises, the computational cost can be high depending on the implementation. The dynamically constrained cutting plane method [JC99] avoids this high computational cost by limiting the number of approximating hyperplanes and by iteratively updating the domain $\mathcal{C}$.

Third, a more sophisticated version of the cutting plane method is the bundle method [HUL96, PRS96, ZLZ99], where the new multiplier λ_{n+1} is obtained by solving the following quadratic problem:

$$\left.\begin{array}{ll} \max & z - \alpha_n \cdot \|\lambda - \Lambda_n\|^2 \\ & \\ s.t. & z \leq q(\lambda_k) + h(x_k)'(\lambda - \lambda_k) \\ & \qquad k = 1, \ldots, n \\ & \lambda \in \mathcal{C} \end{array}\right\}, \tag{5.8}$$

α_n being a penalty parameter and Λ_n representing the 'center of gravity' of the feasible region, which is intended to avoid the typical oscillations of the subgradient method. The tuning of the penalty parameter and other parameters is problem-dependent and hard to achieve.

5.1.2 Augmented Lagrangian relaxation

One of the main drawbacks of the Classical Lagrangian Relaxation (CLR) method is the non-differentiable character of the dual function in many instances. The Augmented Lagrangian Relaxation (ALR) method overcomes the non-differentiability of the dual function by adding a penalty term to the primal function $f(x)$. As a result, the Lagrangian function is augmented by this penalty term. Let us consider again the previous primal problem (P) but with the objective function penalized by the term $\frac{c}{2} \cdot \|h(x)\|^2$. We refer to this problem as the Penalized Primal (PP) problem. It must be noticed that (P) and (PP) are equivalent problems since they have the same optimizers and optimum objective values. Let (PP) be

$$\left.\begin{array}{ll} \min & f(x) + \frac{c}{2} \cdot \|h(x)\|^2 \\ & \\ s.t. & h(x) = 0 \\ & \\ & x \in \mathcal{D} \end{array}\right\}, \tag{5.9}$$

where $f(x) : R^n \longmapsto R$ and $h(x) : R^n \longmapsto R^m$, with $\mathcal{D}$ being a compact set and c a positive penalty parameter.

The (Lagrangian) dual problem (PD) of (PP) is

$$\begin{array}{c} \max \\ \lambda \in R^m \end{array} \left\{ \begin{array}{c} \min \\ x \in \mathcal{D} \end{array} \; f(x) + \lambda' h(x) + \tfrac{c}{2} \cdot \|h(x)\|^2 \right\}. \tag{5.10}$$

Since the augmented Lagrangian function

$$L_c(x, \lambda) := f(x) + \lambda' h(x) + \frac{c}{2} \cdot \|h(x)\|^2,$$

the dual problem can be written as

$$\begin{array}{c} \max \\ \lambda \in R^m \end{array} \left\{ \begin{array}{c} \min \\ x \in \mathcal{D} \end{array} \; L_c(x, \lambda) \right\}, \tag{5.11}$$

or even shorter, defining the dual function $q_c(\lambda) := \min\{L_c(x,\lambda) : x \in \mathcal{D}\}$,

$$\begin{array}{cc} \max & q_c(\lambda) \\ \lambda \in R^m & \end{array} . \tag{5.12}$$

In general $q_c(\lambda)$ is a concave function and differentiable for a large enough c (provided f and h are differentiable as well [Ber95]). Furthermore, the ALR method makes it possible to obtain a saddle-point even in cases where the CLR method presents a duality gap [Min83]. In such cases a solution of the dual problem (PD) provides a solution of the primal problem (PP). Thus, we solve (PD) whenever it is easier to solve than (PP). More details can be found in [Ber82, Min83, Lue89]. The following Augmented Lagrangian Relaxation (ALR) algorithm solves the dual problem (PD).

Augmented Lagrangian relaxation algorithm

* [Method.] Augmented Lagrangian relaxation plus resolution of the associated dual problem.

* [Objective.] To compute a local optimizer of the primal problem (5.1).

* [Input.] λ_0 estimate of the Lagrange multipliers and c_0 initial penalty parameter.

* [Output.] (x^*, λ^*) primal-dual solution.

Step 0 [Initialize.] Set $n = 0$.

Step 1 [Compute the dual function for λ_n.]

$$\left.\begin{array}{cc} \min & f(x) + \lambda_n' \cdot h(x) + \frac{c_n}{2} \cdot \|h(x)\|^2 \\ x \in \mathcal{D} & \end{array}\right\} \tag{5.13}$$

If x_n is the calculated solution, then $\nabla q_c(\lambda_n) = h(x_n)$.

Step 2 [Check the stopping criterion.] If $\nabla q_c(\lambda_n) = 0$ then stop. (x_n, λ_n) is a primal-dual solution.

Step 3 [Multiplier updating.] Compute $\lambda_{n+1} = \lambda_n + c_n \cdot \nabla q_c(\lambda_n)$.

Step 4 [Penalty parameter updating.] Update the penalty parameter $c_{n+1} = \phi(c_n)$. An example for ϕ is $\phi(c) = \beta c$ with $\beta \geq 1$, although there is no a standard function ϕ and some tuning may be needed.

Step 5 [Update the iteration count.] Set $n = n + 1$ and go back to step 1.

The CLR method presented above is also called the multiplier method [Ber82]. The main advantage of the ALR method over the CLR method is that the ALR method makes it possible to obtain a saddle-point of the Lagrangian even in those cases where the CLR method obtains a duality gap. The resolution of (P) by the CLR method often yields an infeasible primal solution x_n due to the duality gap, whereas in the ALR method a solution of the dual problem provides a primal solution [Min83, Roc73]. On top of this, the dual function of the ALR method is differentiable whereas the dual function of the CLR method may not be differentiable, as we said earlier.

The main advantages of the CLR method over the ALR method are: (1) In the ALR method the penalty parameter updating must be tuned and it is problem-dependent, whereas using the CLR method with the last generation of cutting plane multiplier updating, the process is not problem-dependent because the set of cutting planes eventually represents a fine approximation of the dual function. (2) If the primal function $f(x)$ is a separable function, so will the Lagrangian $L(x, \lambda)$, and within the CLR method the Lagrangian $L(x, \lambda)$ can be minimized by decomposition. However, the quadratic term introduced by the augmented Lagrangian is not separable and therefore the augmented Lagrangian $L_c(x, \lambda)$ cannot be minimized by decomposition. In this case, if we wish to solve a problem by decomposition, some method such as the auxiliary problem principle or the block coordinate descent, must be used.

5.2 The radar gradient method

5.2.1 Motivation and deduction

On the one hand, the Classical Lagrangian Relaxation (CLR) method (cutting plane or bundle version), unlike the Augmented Lagrangian Relaxation (ALR) method, takes advantage of the first order information generated each time $L(x, \lambda)$ is minimized. Given that $h(x_n)$ is a subgradient of $q(\lambda_n)$, eventually, the CLR method builds up an accurate first order approximation of the dual function $q(\lambda)$. Then, instead of tuning some parameter, as in the ALR method, the CLR method works upon an accurate knowledge of the dual function. On the other hand, the ALR method, unlike the CLR method, takes advantage of the differentiability of the dual function, that is, the ALR method improves the current iterate using $\nabla q_c(\lambda_n)$, the steepest ascent direction.

Consequently, it seems logical to improve the ALR method by incorporating the first order information about the dual function $q(\lambda)$ exploited by the CLR method, whereas the good qualities of the ALR method remain unchanged. This is the philosophy of the radar gradient method presented below.

The objective of the Radar Gradient (RG) method is to maximize any differentiable and concave function $q(\lambda)$ without constraints, as is the objective of the unconstrained dual problem. This method uses the same information as the cutting plane method but in a different way. The tangent planes obtained in the course of the optimization give a first order approximation of $q(\lambda)$. The cutting plane method directly maximizes the function induced by the successive approximations of $q(\lambda)$, whereas the RG method uses the approximation to $q(\lambda)$ in order to compute the step length along the gradient direction. Another multiplier updating method that shares the same philosophy can be found in [LB99], although the resulting updating

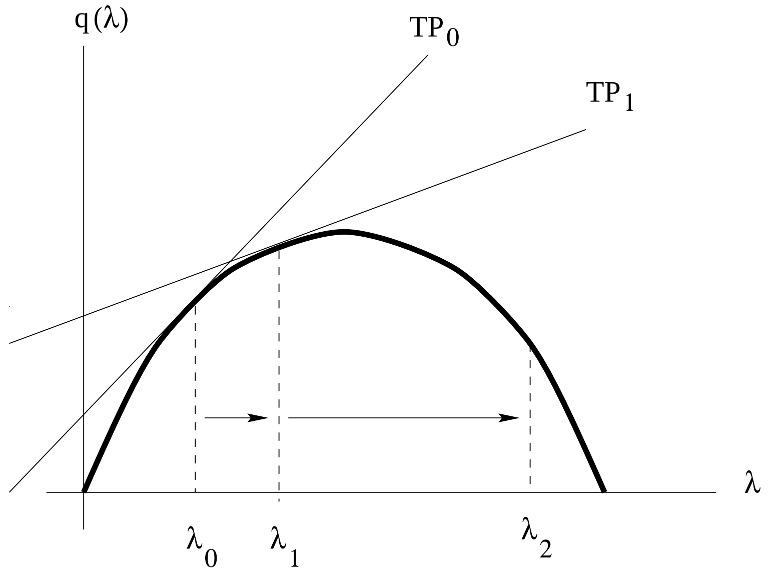

Figure 5.1: Radar gradient method (a).

procedure is different from the one we propose.

The geometrical intuition of the RG method is displayed in Figures 5.1 and 5.2.

The basic idea is to perform a line search along the gradient direction by using the first order approximation of $q(\lambda)$. This line search is defined as follows:

In the first case (Figure 5.1), if no previous stopping tangent planes exist then move λ_n one *step* along the gradient direction. In the second case (Figure 5.2), there is at least one previous stopping tangent plane, then move λ_n up to the first stopping tangent plane along the gradient direction.

A first draft of the RG method is given below.

Radar gradient algorithm (draft)

* [Method.] Radar gradient.

* [Objective.] To solve the following problem, where $q(\lambda)$ is a concave and differentiable function:

$$\max_{\lambda \in R^m} \quad q(\lambda) \tag{5.14}$$

* [Input.] λ_0 initial point.

* [Output.] λ^* optimizer of $q(\lambda)$.

Step 0 [Initialize.] Set $n = 0$.

Step 1 [Compute the gradient vector.] Compute $g_n := \nabla q(\lambda_n)$, the gradient vector. Let TP_n be the first order approximation of $q(\lambda)$ at the point

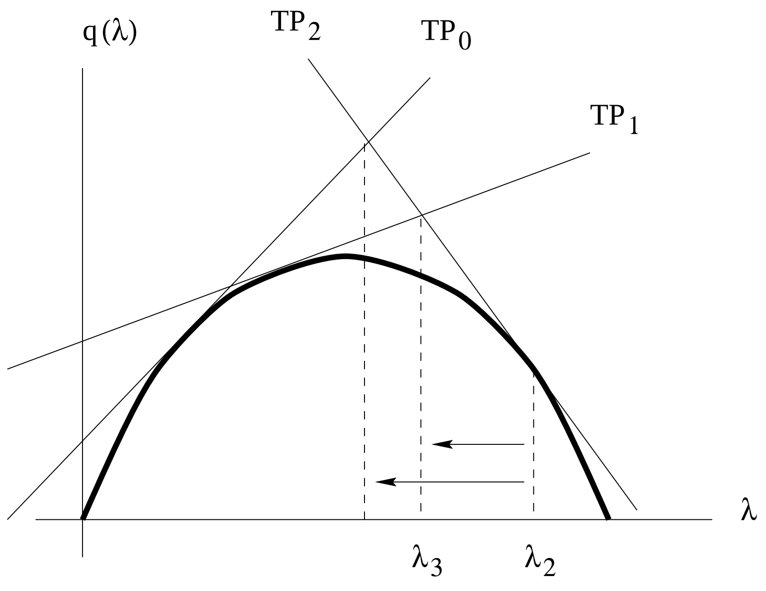

Figure 5.2: Radar gradient method (b).

$(\lambda_n, q(\lambda_n))$, i.e. TP_n is the tangent plane defined by the point $(\lambda_n, q(\lambda_n))$ and the vector g_n. Store the tangent plane TP_n.

Step 2 [Check the stopping criterion.] If $g_n = 0$ then stop. λ_n is the optimizer of $q(\lambda)$.

Step 3 [Compute the step length.] Move upon TP_n in such a way that λ_{n+1} moves on the straight line $\lambda_{n+1} = \lambda_n + \beta \cdot g_n$, with $\beta > 0$. Keep moving up to the first stopping tangent plane TP_k with $k < n$. This means we stop the advance of λ_{n+1} for a value of β, say β_n. If no such stopping plane exists set $\beta_n = \frac{step}{\|g_n\|}$, for a prefixed value of *step*.

Step 4 Compute $\lambda_{n+1} = \lambda_n + \beta_n g_n$. Set $n = n + 1$ and go back to step 1.

Proposition 5.1 *Let us define*

step	*positive scalar,*
n	*Radar Gradient (RG) iteration count,*
λ_n	*current iterate,*
g_n	$:= \nabla q(\lambda_n)$,
q_n	$:= q(\lambda_n)$,
TP_k	$\equiv y_k(\lambda) = q_k + g'_k(\lambda - \lambda_k)$ $\quad (k = 0, \ldots, n)$ *tangent planes,*
$\lambda_{n+1}(\beta)$	$:= \lambda_n + \beta \cdot g_n$ *line defined by the point* λ_n *and the vector* g_n,
$(\lambda_{n,k}, y_{n,k})$	*intersection point of the line* $y_n(\lambda_{n+1}(\beta))$ *with the tangent plane* TP_k,

$\beta_{n,k}$	*step length from* λ_n *to* $\lambda_{n,k}$*, i.e.* $\lambda_{n,k} = \lambda_n + \beta_{n,k} \cdot g_n$*,*
β_n	*step length from* λ_n *to* λ_{n+1} *given by the RG method, i.e.* $\lambda_{n+1} = \lambda_n + \beta_n \cdot g_n$*,*
Ω	*set of the positive steps from* λ_n*, i.e.* $\Omega := \{\beta_{n,k} : \quad \beta_{n,k} > 0 \quad k = 0, \ldots, n-1\}$*,*

and suppose that $q(\lambda)$ *is a concave and differentiable function. Then the step length* β_n *of the previous radar gradient algorithm can be computed as follows. Compute*

$$\beta_{n,k} := \frac{q_k - q_n + (\lambda_n - \lambda_k)' g_k}{(g_n - g_k)' g_n} \qquad k = 0, \ldots, n-1. \tag{5.15}$$

Case a: If there is a stopping tangent plane along the gradient direction g_k*, i.e.* $\Omega \neq \emptyset$*, then set*

$$\beta_n := \min\{\beta_{n,k} : \quad \beta_{n,k} > 0 \quad k = 0, \ldots, n-1\}. \tag{5.16}$$

Case b: If there is no stopping tangent plane along the gradient direction g_k*, i.e.* $\Omega = \emptyset$*, then set*

$$\beta_n := \frac{step}{\|g_k\|}. \tag{5.17}$$

Finally update $\lambda_{n+1} = \lambda_n + \beta_n \cdot g_n$*.*

Proof. Let us compute $\beta_{n,k}$, given by $(\lambda_{n,k}, y_{n,k})$, the intersection point of the tangent planes TP_n and TP_k when λ moves along the line $\lambda_{n+1}(\beta)$. By definition,

$$\left.\begin{array}{rcl} & & \lambda_{n+1}(\beta) := \lambda_n + \beta g_n \\ TP_n & \equiv & y_n = q_n + g_n'(\lambda_{n+1}(\beta) - \lambda_n) \\ TP_k & \equiv & y_k = q_k + g_k'(\lambda_{n+1}(\beta) - \lambda_k) \end{array}\right\} \begin{array}{l} \longrightarrow \quad y_n = q_n + g_n'(\cancel{\lambda_n} + \beta g_n - \cancel{\lambda_n}) \\ \longrightarrow \quad y_k = q_k + g_k'(\lambda_n + \beta g_n - \lambda_k) \end{array} \tag{5.18}$$

At the intersection it happens that $y_n = y_k$, hence

$$\begin{array}{rcl} q_n + g_n'(\beta g_n) & = & q_k + g_k'(\lambda_n + \beta g_n - \lambda_k) \\ q_n + \beta g_n' g_n & = & q_k + g_k'(\lambda_n - \lambda_k) + \beta g_k' g_n \\ \beta (g_n - g_k)' g_n & = & q_k - q_n + (\lambda_n - \lambda_k)' g_k. \end{array} \tag{5.19}$$

And therefore, the intersection point $(\lambda_{n,k}, y_{n,k})$ is defined by the step

$$\beta_{n,k} := \frac{q_k - q_n + (\lambda_n - \lambda_k)' g_k}{(g_n - g_k)' g_n} \qquad k = 0, \ldots, n-1. \tag{5.20}$$

Because the aim of the RG method is to maximize $q(\lambda)$ following the gradient direction, only positive length steps $\beta_{n,k}$ $(k = 0, \dots, n-1)$ improve our objective function. Therefore, given that we wish to stop with the first intersection point among the points $\{(\lambda_{n,k}, y_{n,k}) : k = 0, \dots, n-1\}$, the minimum positive $\beta_{n,k}$ must be chosen. Thus

$$\beta_n := \min\{\beta_{n,k} : \quad \beta_{n,k} > 0 \quad k = 0, \dots, n-1\}. \tag{5.21}$$

Alternatively, it can happen that no such intersection point exists. In this case, we move one prefixed length *step* following the gradient direction. As the maximization proceeds, it is supposed that the gradient g_n will vanish, thus, in order to avoid a lack of progress, the gradient direction must be normalized. This means

$$\lambda_{n+1} = \lambda_n + step \, \frac{g_n}{\|g_n\|} \quad \text{and therefore} \quad \beta_n := \frac{step}{\|g_k\|} \tag{5.22}$$

as we wanted to prove. □
The final version of the algorithm RG incorporates the above proposition.

Radar gradient algorithm

* [Method.] Radar gradient.

* [Objective.] To solve the following problem, where $q(\lambda)$ is a concave and differentiable function:

$$\max_{\lambda \in R^m} \quad q(\lambda) \tag{5.23}$$

* [Input.] λ_0 initial point.

* [Output.] λ^* optimizer of $q(\lambda)$.

Step 0 [Initialize.] Set $n = 0$, and the parameter *step*.

Step 1 [Compute and store the tangent plane TP_n.] Compute the gradient vector g_n and the objective value q_n. Store g_n, q_n and λ_n.

Step 2 [Check the stopping criterion.] If $g_n = 0$ then stop. λ_n is the optimizer of $q(\lambda)$.

Step 3 [Compute the step length β_n]. For $k = 0, \dots, n-1$ compute

$$\beta_{n,k} := \frac{q_k - q_n + (\lambda_n - \lambda_k)' g_k}{(g_n - g_k)' g_n} \tag{5.24}$$

There are two cases depending on

$$\Omega := \{\beta_{n,k} : \quad \beta_{n,k} > 0 \quad k = 0, \dots, n-1\}. \tag{5.25}$$

Case a: If $\Omega \neq \emptyset$, then set

$$\beta_n := \min \Omega \tag{5.26}$$

Case b: If $\Omega = \emptyset$, then set

$$\beta_n := \frac{step}{\|g_k\|}. \tag{5.27}$$

Step 4 [Update the multipliers.] Compute

$$\lambda_{n+1} = \lambda_n + \beta_n \cdot g_n, \tag{5.28}$$

set $n = n + 1$ and go back to step 1.

5.2.2 A one-dimensional example

In the following example $q(\lambda) := -\frac{\lambda^2}{2}$, a concave and differentiable function is maximized over R using the Radar Gradient (RG) method.

$$\begin{array}{c} \max \ -\frac{\lambda^2}{2} \\ \lambda \in R \end{array} \tag{5.29}$$

It is very easy to show that $\lambda^* = 0$ and $q(\lambda^*) = 0$. Before starting with the RG iterations a shorter expression for $\beta_{n,k}$ is computed.

$$g_n := \nabla q(\lambda_n) = -\lambda_n$$

$$\beta_{n,k} := \frac{q_k - q_n + (\lambda_n - \lambda_k)' g_k}{(g_n - g_k)' g_n} = \frac{-\frac{\lambda_k^2}{2} + \frac{\lambda_n^2}{2} + (\lambda_n - \lambda_k) \cdot (-\lambda_k)}{(-\lambda_n + \lambda_k) \cdot (-\lambda_n)} =$$

$$= \frac{\frac{\lambda_n^2}{2} - \frac{\lambda_k^2}{2} - \lambda_n \cdot \lambda_k + \lambda_k^2}{(\lambda_n - \lambda_k)\lambda_n} = \frac{\frac{\lambda_n^2}{2} - \frac{2\lambda_n\lambda_k}{2} + \frac{\lambda_k^2}{2}}{(\lambda_n - \lambda_k)\lambda_n} =$$

$$= \frac{1}{2} \cdot \frac{(\lambda_n - \lambda_k)^2}{(\lambda_n - \lambda_k)\lambda_n} = \frac{1}{2} \cdot \frac{\lambda_n - \lambda_k}{\lambda_n}$$

Thus, the particular expression for $\beta_{n,k}$ within this example is

$$\beta_{n,k} = \frac{1}{2} \cdot \frac{\lambda_n - \lambda_k}{\lambda_n} \qquad k = 0, \ldots, n-1.$$

Input

$$\lambda_0 = -1 \text{ and } step = 3 \quad \text{(arbitrarily chosen)}$$

Iteration 0

$$g_0 = -\lambda_0 = 1$$

$$\Omega := \{\beta_{n,k} : \quad \beta_{n,k} > 0 \quad k = 0, \ldots, n-1\}$$

$$\Omega = \emptyset \;\Rightarrow\; \beta_0 = \frac{step}{\|g_0\|} = \frac{3}{1} = 3$$

$$\lambda_1 = \lambda_0 + \beta_0 \cdot g_0 = -1 + 3 \cdot 1 = 2$$

Iteration 1

$$g_1 = -\lambda_1 = -2$$

$$\beta_{1,0} = \frac{1}{2}\frac{\lambda_1 - \lambda_0}{\lambda_1} = \frac{1}{2}\frac{2-(-1)}{2} = \frac{3}{4}$$

$$\beta_1 = \min\Omega = \min\left\{\frac{3}{4}\right\} = \frac{3}{4}$$

$$\lambda_2 = \lambda_1 + \beta_1 \cdot g_1 = 2 + \frac{3}{4} \cdot (-2) = \frac{1}{2}$$

Iteration 2

$$g_2 = -\lambda_2 = -\frac{1}{2}$$

$$\beta_{2,0} = \frac{1}{2}\frac{\lambda_2 - \lambda_0}{\lambda_2} = \frac{3}{2}$$

$$\beta_{2,1} = \frac{1}{2}\frac{\lambda_2 - \lambda_1}{\lambda_2} = -\frac{3}{2}$$

$$\beta_1 = \min\Omega = \min\left\{\frac{3}{2}\right\} = \frac{3}{2}$$

$$\lambda_3 = \lambda_2 + \beta_2 \cdot g_2 = -\frac{1}{4}$$

The values of λ_n for the first 11 iterations are displayed below:

n	0	1	2	3	4	5	6	7	8	9	10	11
λ_n	-1	2	$\frac{1}{2}$	$\frac{-1}{2^2}$	$\frac{1}{2^3}$	$\frac{-1}{2^4}$	$\frac{1}{2^5}$	$\frac{-1}{2^6}$	$\frac{1}{2^7}$	$\frac{-1}{2^8}$	$\frac{1}{2^9}$	$\frac{-1}{2^{10}}$

Solution

$$\|g_{11}\| = \left|\frac{-1}{2^{10}}\right| = 9.7 \cdot 10^{-4} < 10^{-3} \quad \text{stopping criterion.}$$

$$\lambda_{11} = -\frac{1}{2^{10}} = -9.7 \cdot 10^{-4} \approx 0 \quad \text{computed optimizer,}$$

$$q(\lambda_{11}) = -4.4 \cdot 10^{-7} \approx 0 \quad \text{computed optimal objective.}$$

Table 5.1: Solved UC instances.

Case	Hydraulic data file	Thermal data file
1	hp060	ti0400691
2	hp480	thunit40404891
3	reda481	thunit40704896
4	hp480	thunit40704891
5	reda160	thunit40216891
6	reda160	thunit40716891

Table 5.2: Description of the UC instances.

Case	Number intervals	Number reservoirs	Thermal units	Continuous variables	Binary variables
1	6	2	4	138	24
2	48	2	4	1104	192
3	48	4	7	1920	336
4	48	2	7	1680	336
5	168	4	2	3360	336
6	168	4	7	6720	1176

5.2.3 The radar gradient method versus the multiplier method

The objective of this section is to compare the performance of the multiplier method versus the radar gradient method in terms of CPU time. In chapter 4 we saw the superior performance of the Augmented Lagrangian Relaxation (ALR) method combined with the Block Coordinate Descent (BCD) method (ALR+BCD algorithm). Now we compare two versions of the ALR+BCD algorithm, the initial version that uses the multiplier method to update the Lagrange multipliers and a new version that uses the radar gradient method. We call these versions the ALR+M and the ALR+R methods respectively.

In this test, 6 instances of the Unit Commitment (UC) problem are solved. In Table 5.1 we present these instances, and in Table 5.2 we describe their main features, which range from 24 to 1176 binary variables and from very small size (6 intervals, 2 reservoirs and 4 thermal units) up to medium size (168 intervals, 4 reservoirs and 7 thermal units). The UC problems solved here consider the hydraulic and thermal systems.

The parameters used in both the ALR+M and the ALR+R algorithms are: the stopping criterion $\epsilon = 10^{-4}$, the penalty parameter c_n in Table 5.3, which is maintained

Table 5.3: The penalty parameter c_n.

Case	1	2	3	4	5	6
c_n	20.00	0.01	10.00	10.00	10.00	1 .00

Table 5.4: Results using the ALR+M or the ALR+R algorithm.

Case	Iterations		CPU time (seconds)		Cost ($\times 10^6$ PTA)	
	ALR+M	ALR+R	ALR+M	ALR+R	ALR+M	ALR+R
1	15	15	6.3	7.0	7.269	7.269
2	51	51	27.9	27.5	0.991	0.991
3	2	4	23.3	24.0	9.102	9.102
4	7	39	8.6	31.7	2.447	2.692
5	3	3	42.0	43.7	4.582	4.582
6	5	5	456.0	439.0	6.935	6.934
Average	13.8	19.5	94.0	95.5	5.220	5.260

constant for every iteration, and $step = c_n$.

We observe that the quality of the optimizers is very similar for the two methods ('Cost' column). The number of iterations and the CPU time is similar in the two methods, the ALR+M method being slightly faster. Note that the performance of the two methods is very different in case 4, where they reach different optimezers. Given that in this test the ALR+R method did not outperform the ALR+M method, no more tests were carried out. Although for the augmented Lagrangian case the RG does not seem to improve the multiplier method, in the next section we will see that for the classical Lagrangian case, a non-differentiable version of the RG method proves to be very useful.

5.3 The radar subgradient method

5.3.1 Motivation and deduction

In the above section we applied the radar gradient method to maximize a differentiable dual function. Given that, very often, the dual function is not differentiable it would be interesting to adapt the radar gradient method for the non-differentiable case, giving rise to the so called *radar subgradient method*. Fortunately, we will see in this section that this time the new radar subgradient method drastically outperforms the classical subgradient method from a practical point of view.

The radar gradient method was designed under two main assumptions. First, we do not know an explicit expression for the dual function but we do know a first degree approximation, and second, we can always improve our current iterate by following the gradient direction. With a non-differentiable dual function, we still have a first order approximation of the dual function given by the family of computed subgradients, but the subgradient itself may not be an ascent direction [BS79]. Some strategies have been applied to overcome this problem, for example, computing at each dual iteration the minimum norm subgradient that is shown to be an ascent direction [BS79]. Unfortunately, this is computationally too expensive for practical purposes.

In our approach we follow the direction given by the subgradient since, after all, the subgradient method converges to the optimum and it is still used for practical purposes [LPS99, TGS98, NSS98]. As in the radar gradient method, the first approximation of the dual function can still be used to determine the step length in the subgradient direction. In an early version of the Radar Subgradient (RS) method we applied directly the same formulas as in the differentiable case. In general this early RS method performed faster than the subgradient method but obtained worse dual solutions.

The RS method that we present now improves on that early version of the RS method by being more cautious about the non-differentiability of the dual function. Given that the subgradient direction may lead us to a worse iterate since the subgradient may not be an ascent direction, it is possible that some previous supporting planes of the dual function stop us before reaching the optimum, as shown in Figure 5.3. We call them *premature planes*. To be more precise, in Figure 5.3 let us assume the current dual iterate is λ_n, then, to compute the next iterate λ_{n+1} we should only use the supporting plane SP_b and avoid the premature plane SP_a. Otherwise, future iterates could get stacked under the premature plane SP_a. The key point to improve in the early version of the RS method seems to be, therefore, to detect and avoid these premature planes. If a supporting plane is not premature we call it a *suitable plane*. We shall give a more formal (algebraic) definition of premature and suitable plane later on in this chapter.

A first draft of the RS method is given below.

Radar subgradient algorithm (draft)

* [Method.] Radar subgradient.

* [Objective.] To solve the following problem, where $q(\lambda)$ is a concave function (differentiable or otherwise):

$$\max_{\lambda \in R^m} \quad q(\lambda) \tag{5.30}$$

* [Input.] λ_0 initial point.

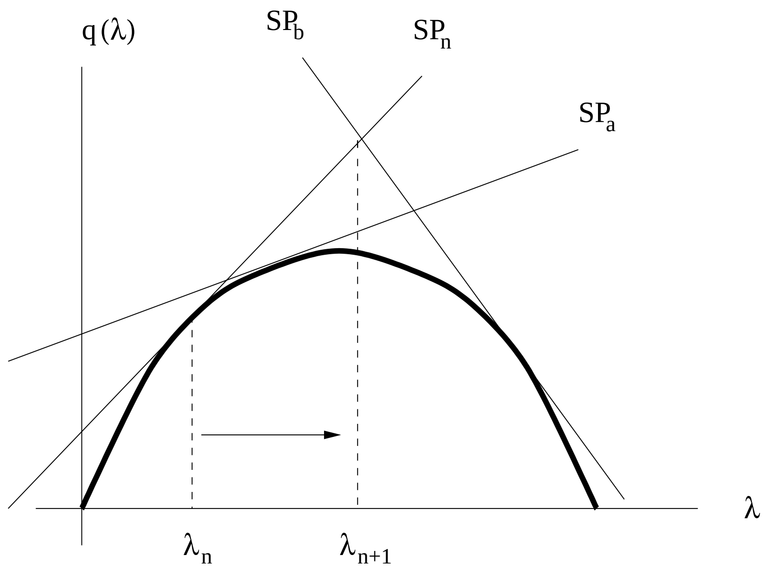

Figure 5.3: Premature and suitable planes.

* [Output.] λ^* optimizer of $q(\lambda)$.

Step 0 [Initialize.] Set $n = 0$.

Step 1 [Compute the dual function for λ_n.] Compute $q(\lambda_n))$ and $s_n \in \partial q(\lambda_n)$, a subgradient vector. Let SP_n be the supporting plane of $q(\lambda)$ at the point $(\lambda_n, q(\lambda_n))$ defined by s_n. Store the supporting plane SP_n.

Step 2 [Check the stopping criterion.] If λ_n does not improve for the last N_{iter} iterations or n reaches a prefixed value then stop.

Step 3 [Compute the step length.] Move upon SP_n in such a way that λ_{n+1} moves along the straight line $\lambda_{n+1} = \lambda_n + \beta \cdot s_n$, with $\beta > 0$. Keep moving up to the first stopping *suitable* plane SP_k with $k < n$. This means we stop the advance of λ_{n+1} for a value of β, say β_n. If no such stopping *suitable* plane exists set $\beta_n = \frac{\alpha_n}{\|s_n\|}$, for a prefixed sequence $\{\alpha_n\}$ that guarantees subgradient convergence.

Step 4 Compute $\lambda_{n+1} = \lambda_n + \beta_n s_n$. Set $n = n + 1$ and go back to step 1.

Proposition 5.2 *Let $q(\lambda)$ be a concave function and let us define*

n	*Radar Subgradient (RS) iteration count,*
λ_n	*current iterate,*
s_n	$\in \partial q(\lambda_n)$,
q_n	$:= q(\lambda_n)$,
SP_k	$\equiv y_k(\lambda) = q_k + s_k'(\lambda - \lambda_k)$ $\quad (k = 0, \dots, n-1)$ *supporting planes,*

$\lambda_{n+1}(\beta)$ $:= \lambda_n + \beta \cdot \frac{s_n}{\|s_n\|}$ *line defined by the point* λ_n *and the vector* s_n,

$y_k(\beta)$ $:= q_k + s_k'(\lambda_{n+1}(\beta) - \lambda_k)$ *line defined on the supporting plane* SP_k *when we move along the line* $\lambda_{n+1}(\beta)$.

Then the slope of any previous supporting plane SP_k $(k = 0, \ldots, n-1)$ *along the line* $\lambda_{n+1}(\beta)$*, i.e. the slope of* $y_k(\beta)$*, is*

$$m_k := \frac{s_k' s_n}{\|s_n\|} \tag{5.31}$$

Proof. Let us express the straight line $y_k(\beta)$ as $b_k + m_k \cdot \beta$, (m_k being its slope).

$$\begin{aligned} y_k(\beta) &:= q_k + s_k'(\lambda_{n+1}(\beta) - \lambda_k) = \\ &= q_k + s_k'(\lambda_n + \beta \cdot \frac{s_n}{\|s_n\|} - \lambda_k) = \\ &= q_k + s_k'(\lambda_n - \lambda_k) + \frac{s_k' s_n}{\|s_n\|} \cdot \beta \end{aligned}$$

whence $b_k =: q_k + s_k'(\lambda_n - \lambda_k)$ and $m_k := \frac{s_k' s_n}{\|s_n\|}$ as we wanted to prove.

Defining $v_n := \frac{s_n}{\|s_n\|}$, an alternative proof is

$$\begin{aligned} m_k &:= \frac{\Delta y_k}{\|\Delta\lambda\|} = \frac{y_k(\lambda_{n+1}) - y_k(\lambda_n)}{\|\lambda_{n+1} - \lambda_n\|} = \\ &= \frac{q_k + s_k'(\lambda_n + \beta \cdot v_n - \lambda_k) - q_k - s_k'(\lambda_n - \lambda_k)}{\|\lambda_n + \beta \cdot v_n - \lambda_n\|} = \\ &= \frac{\beta \cdot s_k' v_n}{\beta \cdot \|v_n\|} = \frac{s_k' s_n}{\|s_n\|} \end{aligned}$$

as we wanted to prove. □

Inspired by Figure 5.3 and the above proposition, we say that the supporting plane SP_k generated by the point q_k and the subgradient s_k $(k < n)$ is a *premature* plane if it has a positive slope along the line $\lambda_{n+1}(\beta)$, that is if $m_k := \frac{s_k' s_n}{\|s_n\|} > 0$. If m_k is not greater than zero, we say that SP_k is a *suitable* plane. Note that for computational purposes it is enough to calculate $s_k' s_n$ to determine whether a supporting plane is premature or suitable.

The final version of the RS method (based on the radar gradient method), which incorporates the algebraic characterization of premature plane, given by the above proposition, is given below.

Radar subgradient algorithm

* [Method.] Radar subgradient.

* [Objective.] To solve the following problem, where $q(\lambda)$ is a concave (differentiable or otherwise) function:

$$\max_{\lambda \in R^m} \quad q(\lambda) \tag{5.32}$$

* [Input.] λ_0 initial point.

* [Output.] λ^* optimizer of $q(\lambda)$.

Step 0 [Initialize.] Set $n = 0$, and ϵ_λ for the stopping criterion.

Step 1 [Compute $q(\lambda_n)$.] Compute $s_n \in \partial q(\lambda_n)$, a subgradient vector and the objective value q_n. Store s_n, q_n and λ_n.

Step 2 [Check the stopping criterion.] If λ_n does not improve for the last N_{iter} iterations or n reaches a prefixed value then stop.

Step 3 [Compute the step length.]

- Compute $s_n' s_n$.
- For $k = 0, \ldots, n-1$
  - Compute $s_k' s_n$.
  - If $s_k' s_n > 0$ reject the premature plane SP_k. Otherwise compute

$$\beta_{n,k} := \frac{q_k - q_n + (\lambda_n - \lambda_k)' s_k}{s_n' s_n - s_k' s_n} \tag{5.33}$$

There are two cases depending on

$$\Omega := \{\beta_{n,k} : \quad \beta_{n,k} > 0 \quad k = 0, \ldots, n-1\}. \tag{5.34}$$

a) If $\Omega \neq \emptyset$, then set $\beta_n := \min \Omega$.

b) If $\Omega = \emptyset$, then set $\beta_n = \frac{\alpha_n}{\|s_n\|}$, for a prefixed sequence $\{\alpha_n\}$ that guarantees subgradient convergence.

Step 4 Compute $\lambda_{n+1} = \lambda_n + \beta_n \cdot s_n$. Set $n = n + 1$ and go back to step 1.

5.3.2 The radar subgradient method versus the subgradient method

The objective of this test is to compare the performance of the Subgradient (SG) method versus the Radar Subgradient (RS) method. The two methods are compared firstly by using a set of small Unit Commitment instances and secondly by using a set of large-scale Unit Commitment instances.

We warn the reader that we have not spent much time on parameter tuning for the two methods. In the case of the RS method this is fine since virtually no tuning is needed. In the case of the SG method, however, better results might have been obtained spending more time on the tuning process. Nevertheless, we still think the test is fair considering we have made the same tuning effort for both methods.

First computational test

In this first test we use an n dimensional version of the simple Unit Commitment (UC) problem presented in chapter 3 with duplicated variables (n is the number of thermal units, x_i and $\tilde{x}_i$ are the output of thermal unit i, Con_i is the starting cost, i.e. the cost of turning a unit on, d is the demand for electrical power, and l_i and u_i are lower and upper bounds for $\tilde{x}_i$):

$$\left.\begin{array}{ll} \min & \sum_{i=1}^{n}(x_i^2+\tilde{x}_i^2+Con_i(\tilde{x}_i)) \\ s.t. & \sum_{i=1}^{n} x_i = d \\ & \tilde{x}_i \in \{0\}\cup[l_i,u_i] \quad i=1,\ldots,n \\ & x_i=\tilde{x}_i \quad i=1,\ldots,n \end{array}\right\} \tag{5.35}$$

where arbitrarily we choose

$$Con_i(\tilde{x}_i) := \left\{\begin{array}{lll} 10+(i-1)\frac{10}{n-1} & \text{if} & \tilde{x}_i\in[l_i,u_i] \\ \\ 0 & \text{if} & \tilde{x}_i=0 \end{array}\right. \tag{5.36}$$

Given that the starting cost function Con_i is monotone on i this problem can easily be solved by algebraic methods.

In this chapter we are interested in solving its dual

$$\max_{\lambda\in R^n}\left\{\begin{array}{ll} \min & \sum_{i=1}^{n}\left(x_i^2+\tilde{x}_i^2+Con_i(\tilde{x}_i)+\lambda_i\cdot(x_i-\tilde{x}_i)\right) \\ s.t. & \sum_{i=1}^{n} x_i = d \\ & \tilde{x}_i\in\{0\}\cup[l_i,u_i] \quad i=1,\ldots,n \end{array}\right\} \tag{5.37}$$

This first test consists of three steps: Step 1), we exactly solve the above UC problem for 10 cases ($n = 10, 20, 30, \ldots, 100$) by algebraic methods in order to know its global optimizer. Step 2), we solve the dual of the same 10 problems using the SG method, and the RS method to check the quality of the computed dual optimizers (dual bounds). Step 3), we compare the efficiency of the SG and the RS methods in terms of the CPU time. The parameters used are: $n = 10, 20, 30, \ldots, 100$; $d = n$; $l_i = 1$ and $u_i = n$.

The tuning parameters are as follows. Regarding the stopping criterion, both the SG and the RS method stop whenever the average variation of λ_n for the last 5 iterations is small enough, i.e. whenever

$$\frac{\sum_{i=0}^{4}\|\lambda_{n-i}-\lambda_{n-i-1}\|_\infty}{5} < \epsilon_\lambda .$$

We use $\epsilon_\lambda = 0.1$ for the SG method, and the RS method requires $\epsilon_\lambda = 0.01$ to obtain dual optima of the same quality. Furthermore, we take $\lambda_0 = 0$, the subgradient step length sequence $\{\alpha_n\}$ for both methods is defined as $\alpha_n = \frac{\alpha_0}{n}$ $(n > 0)$ with $\alpha_0 = 5$ for every case and finally, the maximum number of multiplier updates is 300 for both methods.

Table 5.5: Results using the SG or the RS method.

n	Iterations		CPU time (seconds)			Cost		
	SG	RS	SG	RS	Ratio	ALG	SG	RS
10	141	28	3.0	0.9	3.3	96.7	84.6	84.9
20	154	28	6.0	1.4	4.3	194.7	170.3	169.8
30	179	38	10.7	2.8	3.8	292.6	255.6	256.2
40	189	31	16.8	3.4	4.9	390.6	341.0	340.8
50	178	34	21.8	5.2	4.2	488.1	426.3	427.1
60	190	32	32.2	7.1	4.5	585.9	511.7	512.0
70	198	36	45.3	10.5	4.3	683.8	597.3	597.7
80	198	33	59.6	13.1	4.5	781.7	682.3	682.6
90	198	32	77.9	17.1	4.6	879.5	767.6	767.9
100	201	42	103.2	27.3	3.8	977.3	853.4	854.1
Average	183	33	37.7	8.9	4.2	537.1	469.0	469.3

The optimal primal cost (computed by algebraic methods) is displayed in the column 'ALG' in Table 5.5. The computed optimal dual cost is also displayed for the SG and the RS methods. We observe that the optimal dual costs obtained by the two methods are very similar (on average the SG optimal dual cost is 469.0 and the RS one is 469.3). However, the main noteworthy point in Table 5.5 is the difference in CPU times: on average the SG method needs over four times more than the RS time. As an example of the different convergence behavior between the two methods, Figure 5.4 shows the evolution of the dual function using either the SG method or the RS method for case 10 in Table 5.5. The 141 iterations needed by the SG method are reduced to 28 RS iterations. It is true that the computed optimizers are not primal feasible solutions but this is another question to be dealt with in the next chapter.

Second computational test

In this second test we solve 8 instances of the dual problem associated with the Unit Commitment (UC) problem with duplicated variables (5.38), introduced in chapter 4.

$$\left.\begin{array}{ll} \min & f(x,\tilde{x}) = C_{htd}(x) + C_m(\tilde{x}) \\ s.t. & x \in \mathcal{D}_{ht} \\ & \tilde{x} \in \mathcal{D}_m \\ & x - \tilde{x} = 0 \end{array}\right\} \qquad (5.38)$$

The dual we solve corresponds to the relaxation of the equality constraint $x - \tilde{x} = 0$,

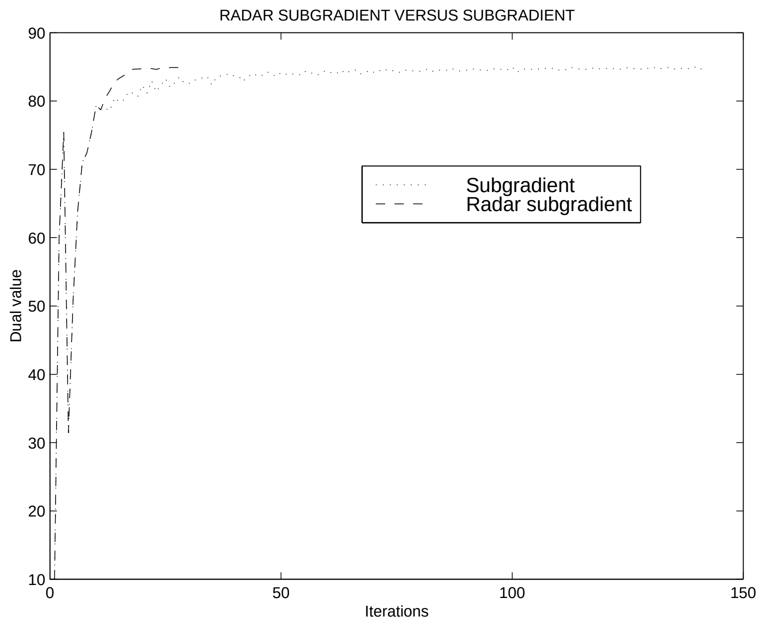

Figure 5.4: Dual function evolution.

that is,

$$\max_{\lambda \in R^n} \left\{ \begin{array}{ll} \min & C_{htd}(x) + C_m(\tilde{x}) + \lambda'(x - \tilde{x}) \\ s.t. & x \in \mathcal{D}_{ht} \\ & \tilde{x} \in \mathcal{D}_m \end{array} \right\} \tag{5.39}$$

In Table 5.6 we present the 8 UC instances, and in Table 5.7 we describe their main features, which range from very small size (2 intervals, 0 reservoirs, 2 thermal units, and 4 binary variables) up to medium size (168 intervals, 4 reservoirs, 11 thermal units, and 1848 binary variables). The UC instances solved here consider the hydraulic and thermal systems.

Unlike in the first test, obviously we cannot know a priori the algebraic optimizer for these large-scale nonlinear combinatorial problems but we can obtain a Load Primal Feasible (LPF) solution that will work as an upper bound to the dual optimum. Then, this second test consists of three steps: Step 1), we compute a LPF solution for the 8 UC instances using the Augmented Lagrangian Relaxation (ALR) method. Step 2), we solve the dual (5.39) for each UC instance using both the SG method and the RS method, to compare the quality of the computed optima. Step 3), we compare the efficiency of the SG and RS methods in terms of CPU time.

The stopping criterion used by both the SG and the RS algorithms is

$$\frac{\sum_{i=0}^{4} \|\lambda_{n-i} - \lambda_{n-i-1}\|_\infty}{5} < \epsilon_\lambda,$$

i.e. both the SG and the RS methods stop whenever the average variation of λ_n for the last 5 iterations is small enough. We use $\epsilon_\lambda = 10^{-4}$ for the SG method, whereas the RS method requires $\epsilon_\lambda = 10^{-5}$ to obtain dual optima of similar quality. The subgradient step length sequence $\{\alpha_n\}$ for both methods is defined as $\alpha_n = \frac{\alpha_0}{n}$ $(n >$

Table 5.6: Solved UC instances.

Case	Hydraulic data file	Thermal data file
1	hz020	ti0200294
2	hp060	ti0400691
3	hp480	thunit40404891
4	reda481	thunit40704896
5	hp480	thunit40704894
6	reda160	thunit40216891
7	reda160	thunit40716891
8	reda160	thunit61116896

Table 5.7: Description of the UC instances.

Case	Number intervals	Number reservoirs	Thermal units	Continuous variables	Binary variables
1	2	0	2	16	4
2	6	2	4	138	24
3	48	2	4	1104	192
4	48	4	7	1920	336
5	48	2	7	1680	336
6	168	4	2	3360	336
7	168	4	7	6720	1176
8	168	4	11	9408	1848

Table 5.8: Tuning parameters.

Case	1	2	3	4	5	6	7	8
α_0	10^0	10^0	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}

Table 5.9: Results using the SG or the RS methods.

Case	Iterations		CPU Time (seconds)			Cost ($\times 10^6$ PTA)		
	SG	RS	SG	RS	Ratio	SG	RS	ALR
1	15	39	6.5	16.0	0.41	0.004	0.004	0.004
2	200	30	60.0	9.8	6.12	5.794	5.810	6.776
3	40	25	25.7	16.3	1.58	0.966	0.961	0.990
4	200	37	169.7	72.4	2.34	6.353	6.324	6.358
5	42	18	31.5	15.7	2.01	1.004	1.004	1.028
6	88	28	150.0	78.6	1.91	4.401	4.330	4.467
7	49	28	632.0	467.0	1.35	2.558	2.530	2.653
8	100	40	25810.0	15479.0	1.67	84.859	84.865	85.896
Average	91.8	30.6	3360.7	2019.4	2.17	13.242	13.229	13.522

0), with α_0 listed in Table 5.8, and the maximum number of multiplier updates is 200 for both methods. Case 8 is an exception since, in order to remain within a reasonable CPU time, we have set $\epsilon_\lambda = 10^{-4}$ for the RS and SG methods, whereas the maximum number multiplier updates has been set equal to 100.

Results of step 1): By augmenting the Lagrangian we have obtained LPF optima (column 'Cost/ALR' in Table 5.9). Therefore from column 'Cost/ALR' we have an upper bound to the dual costs displayed in columns 'Cost/SG' and 'Cost/RS'. Results of step 2): To compare the quality of the dual solutions we compute the Duality Gap $DG := 100 \cdot \frac{f(x^*)-q(\lambda^*)}{q(\lambda^*)}\%$ for each case. The average duality gap is 3.54% for the 8 SG solutions and 3.98% for the 8 RS solutions. Therefore, on average, the quality of the RS solutions is slightly better. Results of step 3): To compare the efficiency of the two methods we compute the ratio $R := \frac{SG\_CPU\_Time}{SR\_CPU\_Time}$ for each case. The average ratio is 2.17. Thus, we can say that in this test, on average, the RS method has obtained similar dual bounds than the SG method in less than half of the time required by the SG method.

5.4 Summary

Lagrangian relaxation methods can be classified into two main families: the Classical Lagrangian Relaxation (CLR) method and the Augmented Lagrangian Relaxation (ALR) method. Within each of these two families different versions arise depending on the Lagrange multiplier updating procedure used. On the one hand, the subgradient, cutting plane and bundle methods are the main multiplier updating methods in the framework of the CLR. On the other hand, within the ALR framework the most important multiplier updating method is the so called multiplier method.

We have introduced a new multiplier updating procedure within the ALR framework, the Radar Gradient (RG) method. The philosophy of this new method is to incorporate the first order information about the dual function already exploited by the CLR method (cutting plane version). This new method uses the same information as the cutting plane method but in a different way, that is, the cutting plane method directly maximizes the successive approximations to the dual function, whereas the RG method uses the approximation to the dual function in order to compute the step length for an ascent direction (such as the gradient). In the tests we carried out, the RG method did not outperform the multiplier method. The non-differentiable version of the RG method is the Radar Subgradient (RS) method. The new RS method clearly outperforms the classical subgradient method from a practical point of view.

Chapter 6

The radar multiplier method

The least announced flaw of the Augmented Lagrangian Relaxation (ALR) method is the local character of the computed solutions when solving non-convex problems, as is also the case with the Unit Commitment (UC) problem. Thus, the first section of the chapter is devoted to analyzing the quality of the ALR solutions and how they can be improved. The second section culminates the efforts made in chapters 4 and 5 to avoid all the drawbacks of Lagrangian relaxation. It turns out that the Classical Lagrangian Relaxation (CLR) method and the ALR method are complementary methods in such a way that the defects of one method are rectified by the other and vice-versa.

6.1 Augmented Lagrangian and local optimizers

Reviewing the Unit Commitment (UC) problem literature one finds that Lagrangian Relaxation is the most widespread procedure to solve this problem. Some authors prefer the Classical Lagrangian Relaxation (CLR) method and others the Augmented Lagrangian Relaxation (ALR) method, as pointed out in chapter 5. When the ALR method is used to solve the UC problem, much effort is devoted to coping with the inseparability of the quadratic term introduced by the augmented Lagrangian (see chapter 4). However, as far as we know, no one has discussed the quality of the computed primal solution. Is it a local or a global optimizer? If local, how good is it? Perhaps the reason for this lack of precision is that it is already difficult enough to compute good feasible primal solutions for the UC problem (a large-scale nonlinear combinatorial problem).

In this section we show that the ALR method may give local optimizers of very poor quality. If we wish to use the ALR method for its virtues, we will have to deal with the local character of its optimizers. In this section we will also show a heuristic method to improve the ALR optimizers and later, in the following section, we will work on finding a lower bound to the global optimum of a UC instance.

6.1.1 Empirical evidence

In this example we use an n dimensional version of the simple Unit Commitment (UC) problem presented in chapter 3 (where n is the number of thermal units, x_i is the output of thermal unit i, Con_i is the starting cost, i.e. the cost of turning a unit on, d is the demand for electrical power, and l_i and u_i are lower and upper bounds for x_i):

$$\left.\begin{array}{ll} \min & \sum_{i=1}^{n}(2x_i^2 + Con_i(x_i)) \\ s.t. & \sum_{i=1}^{n} x_i = d \\ & x_i \in \{0\} \cup [l_i, u_i] \quad i = 1, \dots, n \end{array}\right\} \tag{6.1}$$

where arbitrarily we choose

$$Con_i(x_i) := \begin{cases} 10 + (i-1)\frac{10}{n-1} & \text{if} \quad x_i \in [l_i, u_i] \\ 0 \quad \text{if} \quad x_i = 0 \end{cases} \tag{6.2}$$

Given that the starting cost function Con_i is monotone on i this problem can easily be solved by algebraic methods (see chapter 4).

To study the quality of the optimizers obtained by the Augmented Lagrangian Relaxation (ALR) method, first, as was already done in chapter 5, we solve the UC problem exactly for 10 cases ($n = 10, 20, 30, \dots, 100$) by algebraic methods in order to know its global optimizer, and second, we solve the same 10 problems using the ALR method. The parameters used are: $n = 10, 20, 30, \dots, 100$; $d = n$; $l_i = 1$; $u_i = n$, and $\epsilon = 10^{-4}$ for the ALR stopping criterion.

In Table 6.1 columns 'ALG' and 'ALR' correspond to the exact algebraic method and to the ALR method respectively. For each case we know the optimal number of thermal units started up and the global optimum (column 'ALG'). The main point of this example is that the optimizers obtained by the ALR method are local except for the cases n=10 and n=30 in which a global optimizer is obtained. On average, the relative error ($100 * (Optimum_{ALR} - Optimum_{ALG})/Optimum_{ALG}$) is 1.04%.

6.1.2 Theoretical insight

The local character of the ALR solutions is stated in the following proposition [Lue89].

Proposition 6.1 *Let $f : R^n \to R$ and $g : R^n \to R^m$ be of class C^2. Suppose that the problem $\min\{f(x) : h(x) = 0, \quad x \in R^n\}$ has a local solution in x^* with the associated Lagrange multiplier λ^*. Also suppose that x^*, λ^* satisfy the second order sufficient optimality conditions. Then, there exists c^* such that for every $c \geq c^*$, x^* is a local minimizer of the augmented Lagrangian $L_c(x, \lambda^*)$.*

Table 6.1: The ALR method may obtain local optimizers.

Total number of units	Penalty parameter c_0	Optimal number of units		Optimal cost		Relative error (in %)
		ALG	ALR	ALG	ALR	
10	0.1	4	4	96.67	96.67	0.00
20	0.1	8	7	194.74	195.30	0.29
30	0.1	11	11	292.60	292.60	0.00
40	1.0	15	18	390.60	397.00	1.64
50	1.0	19	22	488.06	494.40	1.30
60	1.0	23	27	585.92	596.20	1.75
70	1.0	27	30	683.83	689.70	0.86
80	1.0	30	36	781.73	795.30	1.74
90	1.0	34	39	879.50	888.60	1.03
100	1.0	38	45	977.33	994.40	1.75
Average		20.9	23.9	537.10	544.02	1.04

6.1.3 An effective heuristic method

Now we present the heuristic method used to improve the quality of the optimizers obtained using the Augmented Lagrangian Relaxation (ALR) method. This heuristic method updates the penalty parameter c_n and is inspired by the procedure given in [Ber95] (page 346). In order to better understand the whole process, the ALR algorithm is displayed below (Block Coordinate Descent (BCD) version studied in chapter 4).

ALR+BCD algorithm

Step 1 [Check the stopping criterion.] If the norm of the gradient of the dual function $\|x_n - \tilde{x}_n\|_\infty < \epsilon$ then stop. $(x_n, \tilde{x}_n, \lambda_n)$ is a primal-dual solution.

Step 2 [Compute x_{n+1}.]

$$\min_{x \in \mathcal{D}} \; f(x) + \lambda_n' x + \tfrac{c_n}{2}\|x - \tilde{x}_n\|^2$$

Step 3 [Compute $\tilde{x}_{n+1}$.]

$$\min_{x \in \tilde{\mathcal{D}}} \; \tilde{f}(\tilde{x}) - \lambda_n' \tilde{x} + \tfrac{c_n}{2}\|x_{n+1} - \tilde{x}\|^2$$

Step 4 [Dual variable updating.]

$$\lambda_{n+1} = \lambda_n + c_n \cdot (x_{n+1} - \tilde{x}_{n+1})$$

Table 6.2: Solved UC instances.

Case	Hydraulic data file	Thermal data file
1	hz020	ti0200294
2	hp060	ti0400691
3	hp480	thunit40404891
4	reda481	thunit40704896
5	hp480	thunit40704894
6	reda160	thunit40216891
7	reda160	thunit40716891
8	reda160	thunit61116896

Step 5 [Penalty parameter updating.]

Step 6 [Counter updating.] Set $n = n + 1$ and go back to step 1.

The idea of the updating heuristic method is to start with a small c_0 and then increase its value *as needed* in order to obtain a differentiable dual function (see [Ber95] for more details). Two main ideas drive the updating heuristic method. First, the actual parameter c_n is augmented if the infeasibility has increased after the last iteration, that is, if $\|x_{n+1} - \tilde{x}_{n+1}\|_\infty > \alpha \cdot \|x_n - \tilde{x}_n\|_\infty$, then $c_{n+1} := \beta \cdot c_n$ (a suitable heuristic choice of parameters is $\alpha \in]1, 1.20]$ and $\beta \in]1, 2]$). Second, we also augment the parameter c_n if the infeasibility has not decreased during the last iteration compared to the previous 5 iterations, that is, if $\|x_{n+1} - \tilde{x}_{n+1}\|_\infty \geq (\sum_{i=0}^{4} \|x_{n-i} - \tilde{x}_{n-i}\|_\infty)/5$, then $c_{n+1} := \beta \cdot c_n$.

Therefore, the penalty parameter updating can be summarized as:

Step 5 [Penalty parameter updating.]

If $\|x_{n+1} - \tilde{x}_{n+1}\|_\infty \geq \alpha \cdot \|x_n - \tilde{x}_n\|_\infty$

or $\|x_{n+1} - \tilde{x}_{n+1}\|_\infty \geq (\sum_{i=0}^{4} \|x_{n-i} - \tilde{x}_{n-i}\|_\infty)/5$

then $c_{n+1} := \beta \cdot c_n$.

A suitable heuristic choice of parameters is $\alpha \in]1, 1.20]$ and $\beta \in]1, 2]$, and $c_0 \in [10^{-3}, 10^{-5}]$.

Let us check this heuristic method by using the same 8 UC instances of the second computational test in section 5.3.2. In Table 6.2 we present again the 8 UC instances, and in Table 6.3 we describe their main features.

The stopping criterion of the ALR+BCD method used for these instances is $\|x_n - \tilde{x}_n\|_\infty < 10^{-4}$. In the penalty parameter updating (step 5) we set $\alpha = 1.10$ and

Table 6.3: Description of the UC instances.

Case	Number intervals	Number reservoirs	Thermal units	Continuous variables	Binary variables
1	2	0	2	16	4
2	6	2	4	138	24
3	48	2	4	1104	192
4	48	4	7	1920	336
5	48	2	7	1680	336
6	168	4	2	3360	336
7	168	4	7	6720	1176
8	168	4	11	9408	1848

$\beta = 2$. The 8 instances have been solved using 3 different values for c_0: 10, 10^{-1}, and 10^{-3} which respectively correspond to 'A', 'B' and 'C' in Table 6.4. In this table, n.c. stands for cases with no convergence. The last row of the table contains the average results of cases 1, 2, 3, 5 and 6.

Three main features can be observed in Table 6.4. First, the computed optimizers are local, as pointed out in subsection 6.1.1. Second, the quality of the computed optimizer increases as the initial penalty parameter c_0 decreases (compare the average cost for subcases A, B, and C). The best optima in this test correspond to $c_0 = 10^{-3}$. Third, in general the CPU time increases as c_0 gets smaller. This slowing down of the method is caused by the fact that the penalty parameter c_n is used as the step length for the multiplier updating within the multiplier method. The smaller c_n is the greater the number of multiplier updates and the CPU time.

As a rule of thumb to tune c_0 one can start by using small values for c_0, say, in the interval $[10^{-2}, 10^{-4}]$. The rationale is that starting with larger values the method could converge to a lower quality optimizer near the starting point, whereas if we initially choose small c_0 values ($c_0 \in [10^{-2}, 10^{-4}]$), c_n will increase after a while and will automatically reach the appropriate level of penalization thanks to the described penalty updating method (step 5 of the ALR+BCD algorithm). Furthermore, we have observed in this test that taking high values for c_0 does not help the the ALR method to converge (note in Table 6.4 that the method failed in three cases for $c_0 = 10$).

6.2 The radar multiplier method

6.2.1 Motivation and deduction

It is clear that the Lagrangian Relaxation (LR) method is a powerful technique to solve 'almost' separable problems such as the Unit Commitment (UC) problem.

Table 6.4: Testing the penalty updating method. Columns A, B, and C correspond to c_0 equal to 10, 10^{-1} and 10^{-3} respectively. n.c. stands for non-convergent. Cases 4, 7 and 8 have not been taken into account to compute the 'Average' row.

Case	CPU Time (seconds)			Optimal cost ($\times 10^6$ PTA)		
	A	B	C	A	B	C
1	2.5	5.3	10.1	0.004	0.004	0.004
2	4.3	13.5	18.0	7.755	7.264	7.059
3	11.5	8.9	20.8	1.693	1.401	0.991
4	n.c.	28.3	39.4	n.c.	8.616	7.965
5	13.2	14.6	30.7	2.349	1.440	1.028
6	43.8	42.7	43.0	5.230	4.501	4.474
7	n.c.	n.c.	553.0	n.c.	n.c.	3.223
8	n.c.	n.c.	600.0	n.c.	n.c.	102.520
Average	15.1	17.0	24.5	3.406	2.922	2.711

Within the LR method two approaches can be followed: the Classical Lagrangian Relaxation (CLR) method and the Augmented Lagrangian Relaxation (ALR) method. Both methods have been described, studied, developed and tested in chapters 4, 5.

The CLR method has two main drawbacks. First, this method induces a dual function which is not necessarily differentiable. The subgradient method was the pioneering method used to treat this dual function. In chapter 5 we developed the Radar Subgradient (RS) method, which notably improves the subgradient method and therefore notably improves the CLR method. Another alternative to the subgradient method is the so called bundle method, but it has yet to be compared with the RS method. The second drawback is that the CLR method usually gives primal infeasible solutions, so solutions must be processed after optimization in order to gain feasibility.

The ALR method also has two main drawbacks. Firstly, we lose the separability of the problem because of the added quadratic term. The whole of chapter 4 is devoted to this question and the Block Coordinate Descent (BCD) method is shown to outperform the Auxiliary Problem Principle (APP) method at coping with the inseparable augmented Lagrangian. Secondly, in this chapter (previous section) we have seen that the ALR method gives local optimizers, and also we have proposed an effective heuristic method that produces good quality local optimizers. Furthermore, within the ALR method, we need some kind of lower bound to the optimal cost in order to control the quality of the computed local optimizers.

Instead of choosing between the CLR and the ALR methods, and having to deal with the inherited drawbacks (primal infeasibility and lack of a lower cost bound) we propose to use both methods complementarily. The CLR method will provide the ALR method with a lower cost bound and the ALR method will process the primal infeasible solution obtained by the CLR method towards feasibility. Thus, we propose to solve the Generalized Unit Commitment (GUC) problem by a two-phase procedure. In the first phase the CLR method (*radar* subgradient version) is applied and produces a primal-dual solution pair $(\widehat{x}, \widehat{\lambda})$. Then, unless $\widehat{x}$ is feasible, the second phase is activated and, the ALR+BCD method (*multiplier* method version) starting from $(\widehat{x}, \widehat{\lambda})$ searches for a new optimal pair (x^*, λ^*) with a primal feasible x^*. From now on this two-phase method will be called the *radar multiplier* (RM) method. An alternative to the RM method could be the Subgradient Multiplier (SM) method if we use the subgradient method in phase 1, or the bundle multiplier method, and so on.

A similar strategy was applied in [PRS96] but in a different way. These authors first use the CLR method (bundle method version) to solve the UC problem and thus they obtain a lower cost bound. Then they use the ALR method (Auxiliary Problem Principle version) to solve an equivalent problem formulation to the initial one. In contrast, we solve the same problem formulation in both phases. Although no empirical comparison has yet been made, a priori, we prefer to use our scheme because the optimal λ^* of the first phase can be used as a starting point for the second phase. With the scheme used in [PRS96] there is no connection between the set of multipliers used in the first phase and the set used in the second phase.

6.2.2 A remark on the duality gap

In the framework of the RM method, we mean by the term 'duality gap' the difference between the local primal optimum computed in the second phase and the dual bound computed in the first phase, that is, $f(x^*) - q(\widehat{\lambda})$, which in general will be greater than zero for a nonconvex problem. Of course, the (*local*) 'duality gap' computed as the difference between the local primal optimum and the local dual bound, both of them computed in the second phase, i.e. $f(x^*) - q_c(\lambda^*)$, must be zero. The reason is that if (x^*, λ^*) is a local saddle point of the augmented Lagrangian $L_c(x, \lambda)$ associated to the problem

$$\min\{f(x) : h(x) = 0, x \in D\},$$

that is,

$$q_c(\lambda^*) = f(x^*) + \lambda^{*'} h(x^*) + \frac{c}{2}\|h(x^*)\|^2,$$

and given that $h(x^*) = 0$, we have

$$q_c(\lambda^*) = f(x^*).$$

To illustrate this matter let us see the following example:

$$\left.\begin{array}{ll} \min & -x^2 \\ s.t. & x \in [-1, 2], \end{array}\right\} \tag{6.3}$$

which is equivalent to

$$\left.\begin{array}{ll} \min & -x^2 \\ s.t. & x^2 - x - 2 \leq 0. \end{array}\right\} \tag{6.4}$$

It is easy to see that using the CLR method, problem (6.4) has only one saddle point $(\widehat{x}, \widehat{\lambda}) = (2, \frac{4}{3})$ with $f(2) = q(\frac{4}{3}) = -4$. However, using the ALR method, with $c = 1$, the same problem has two saddle points: a global saddle point in $(\widehat{x}, \widehat{\lambda}) = (2, \frac{4}{3})$ with $f(2) = q_c(\frac{4}{3}) = -4$, and a local saddle point in $(x^*, \lambda^*) = (-1, \frac{2}{3})$ with $f(-1) = q_c(\frac{2}{3}) = -1$. If we use the ALR method (multiplier method), starting from, $(x_0, \lambda_0) = (\frac{-1}{2}, 1)$, we will converge to the local saddle point (x^*, λ^*). Here we have a local duality gap $f(-1) - q_c(\frac{2}{3}) = 0$, but to assess x^* it is more useful to compare the local optimum $f(-1)$ with the dual bound $q(\frac{4}{3})$. Hence, in the framework of the RM method, by 'duality gap' we mean the difference between the ALR local optimum and the CLR dual bound, in this case $f(-1) - q(\frac{4}{3}) = -1 - (-4) = 3$.

Radar multiplier algorithm (draft)

Let us first give a draft version of the Radar Multiplier method followed by the full version.

* [Method.] Radar multiplier.

* [Objectives.] Two objectives: (1) to obtain a local optimizer of the following problem:

$$\left.\begin{array}{ll} \min & f(x) + \widetilde{f}(\widetilde{x}) \\ s.t. & x \in \mathcal{D}, \quad \widetilde{x} \in \widetilde{\mathcal{D}} \\ & x - \widetilde{x} = 0 \end{array}\right\}, \tag{6.5}$$

  where $\mathcal{D}$ and $\widetilde{\mathcal{D}}$ are compact sets, and (2) to measure the optimizer quality through a dual lower bound to the global optimum of the above problem.

* [Input.] $(x_0, \widetilde{x}_0, \lambda_0)$ initial point.

* [Output.] $(x^*, \widetilde{x}^*)$ local optimizer of $f(x) + \widetilde{f}(\widetilde{x})$, and $\underline{f}^*$ a dual lower bound to the global optimum of problem (6.5).

Phase 1 [Compute a lower cost bound $\underline{f}^*$.]

Using the radar subgradient method solve

$$\max_{\lambda \in R^n} \left\{ \min_{\substack{x \in \mathcal{D} \\ \widetilde{x} \in \widetilde{\mathcal{D}}}} f(x) + \widetilde{f}(\widetilde{x}) + \lambda'(x - \widetilde{x}) \right\}. \tag{6.6}$$

Phase 2 [Compute the local optimizer $(x^*, \widetilde{x}^*)$.]

Using the augmented Lagrangian relaxation method (block coordinate descent version) solve

$$\max_{\lambda \in R^n} \left\{ \min_{\substack{x \in \mathcal{D} \\ \widetilde{x} \in \widetilde{\mathcal{D}}}} f(x) + \widetilde{f}(\widetilde{x}) + \lambda'(x - \widetilde{x}) + \tfrac{c}{2}\|x - \widetilde{x}\|^2 \right\}. \tag{6.7}$$

Radar multiplier algorithm

Phase 1 [Compute a lower cost bound $\underline{f}^*$.]

Step 1.0 [Initialize.] Set $n = 0$, and ϵ_λ for the stopping criterion.

Step 1.1 [Compute the dual function for λ_n.] Compute $s_n \in \partial q(\lambda_n)$, a subgradient vector and the objective dual value $q_n := q(\lambda_n)$ defined as

$$q(\lambda_n) := \min_{\substack{x \in \mathcal{D} \\ \widetilde{x} \in \widetilde{\mathcal{D}}}} f(x) + \widetilde{f}(\widetilde{x}) + \lambda_n'(x - \widetilde{x}). \tag{6.8}$$

Store s_n, q_n and λ_n.

Step 1.2 [Check the stopping criterion.] If λ_n does not improve for the last N_{iter} iterations or n reaches a prefixed value then go to phase 2. Two alternative heuristic criteria have been used for practical purposes:

a) [First stopping criterion.] Stop phase 1 whenever

$$\left| \frac{q_n - (q_{n-1} + q_{n-2} + q_{n-3})/3}{1 + q_n} \right| < \epsilon_\lambda$$

(note that here $N_{iter} = 4$).

b) [Second stopping criterion.] Stop phase 1 whenever

$$\frac{\sum_{i=0}^{4} \|\lambda_{n-i} - \lambda_{n-i-1}\|_\infty}{5} < \epsilon_\lambda$$

(note that here $N_{iter} = 5$).

Step 1.3 [Compute the step length by the radar subgradient method.]

* Compute $s_n' s_n$.
* For $k = 0, \ldots, n-1$
  · Compute $s_k' s_n$.
  · If $s_k' s_n > 0$ reject the premature plane SP_k. Otherwise compute

$$\beta_{n,k} := \frac{q_k - q_n + (\lambda_n - \lambda_k)' s_k}{s_n' s_n - s_k' s_n} \tag{6.9}$$

There are two cases depending on

$$\Omega := \{\beta_{n,k} : \quad \beta_{n,k} > 0 \quad k = 0, \ldots, n-1\}. \tag{6.10}$$

a) If $\Omega \neq \emptyset$, then set $\beta_n := \min \Omega$.

b) If $\Omega = \emptyset$, then set $\beta_n = \frac{\alpha_n}{\|s_n\|}$, for a prefixed sequence $\{\alpha_n\}$ that guarantees subgradient convergence.

Step 1.4 Compute $\lambda_{n+1} = \lambda_n + \beta_n \cdot s_n$. Set $n = n + 1$ and go back to step 1.1.

Phase 2 [Compute the local optimizer $(x^*, \widetilde{x}^*)$.]

Step 2.0 [Initialize.] Set $n = 0$, ϵ for the stopping criterion, and $(x_0, \tilde{x}_0, \lambda_0)$ equal to the optimal point obtained in phase 1. Initialize c_0 to a small value, say, $c_0 \in [10^{-3}, 10^{-5}]$.

Step 2.1 [Check the stopping criterion.] If the norm of the gradient of the dual function $\|x_n - \tilde{x}_n\|_\infty < \epsilon$ then stop. $(x_n, \tilde{x}_n, \lambda_n)$ is a solution.

Step 2.2 [Compute x_{n+1}.]

$$\begin{array}{c} \min \;\; f(x) + \lambda_n' x + \frac{c_n}{2}\|x - \tilde{x}_n\|^2 \\ x \in \mathcal{D} \end{array}$$

Step 2.3 [Compute $\tilde{x}_{n+1}$.]

$$\begin{array}{c} \min \;\; \tilde{f}(\tilde{x}) - \lambda_n' \tilde{x} + \frac{c_n}{2}\|x_{n+1} - \tilde{x}\|^2 \\ x \in \tilde{\mathcal{D}} \end{array}$$

Step 2.4 [Dual variable updating.]

$$\lambda_{n+1} = \lambda_n + c_n \cdot (x_{n+1} - \tilde{x}_{n+1})$$

Step 2.5 [Penalty parameter updating.]

If $\quad \|x_{n+1} - \tilde{x}_{n+1}\|_\infty \geq \alpha \cdot \|x_n - \tilde{x}_n\|_\infty$

or $\quad \|x_{n+1} - \tilde{x}_{n+1}\|_\infty \geq (\sum_{i=0}^{4} \|x_{n-i} - \tilde{x}_{n-i}\|_\infty)/5$

then $\quad c_{n+1} := \beta \cdot c_n$.

A suitable choice of parameters is $\alpha \in]1, 1.20]$ and $\beta \in]1, 2]$.

Step 2.6 Set $n = n + 1$ and go back to step 2.1.

6.2.3 Computational experience

Let us test the performance of the Radar Multiplier (RM) method applied to solving the Unit Commitment (UC) problem. The issues we are aiming to clarify are the quality of the lower cost bounds, the load primal feasibility of the computed primal solutions, and the efficiency of the RM method. We will use the instances of the UC problem introduced at the beginning of this chapter to study these issues.

First test

In this first test we solve again the examples given in section 6.1.1 by means of the RM method and the Subgradient Multiplier (SM) method. The tuning parameters are the initial penalty parameter $c_0 = 10^{-3}$, the first phase stopping criterion is defined as

$$\left| \frac{q_n - (q_{n-1} + q_{n-2} + q_{n-3})/3}{1 + q_n} \right| < 10^{-4},$$

and the second phase stopping criterion with $\epsilon = 10^{-4}$.

Table 6.5: The RM method vs. the SM method (a).

Case	CPU time (seconds)			Optimal cost (PTA)	Cost bound (PTA)	
	SM	RM	Ratio	RM-SM	SM	RM
10	3.1	2.6	1.19	96.6	83.9	84.8
20	5.4	5.7	0.95	194.7	169.2	169.8
30	13.3	7.7	1.73	292.6	253.1	256.3
40	20.1	13.0	1.55	390.6	330.3	340.8
50	13.8	11.7	1.18	488.1	419.6	427.1
60	42.8	39.2	1.09	585.9	498.8	512.0
70	80.3	80.3	1.00	683.8	591.0	597.8
80	110.2	62.2	1.77	781.7	662.7	682.7
90	151.7	81.0	1.87	879.5	760.4	767.7
100	135.7	52.3	2.59	977.3	844.8	852.5
Average	57.64	35.57	1.49	537.098	461.38	469.15

Table 6.6: Initial penalty parameter c_0.

Case	10	20	30	40	50	60	70	80	90	100
	0.10	0.05	0.10	0.10	0.10	0.05	0.10	0.05	0.10	0.10

In Table 6.5 we observe that the CPU time ratio ($SM\ CPU\ time/RM\ CPU\ time$) is, on average, 1.49, that is, for this particular test the SM method takes on average 49% more time than the RM method. Both methods reach the global optimizer for each case (column 'Optimal cost' of the table). We must warn the reader that although for this simple example the RM and the SM methods give a global optimizer, in general with these methods we will obtain a local optimizer. The last column of the table displays the lower cost bounds for the two methods. The bounds given by the RM method are also better than those given by the SM method.

We have just seen that, for this example, the RM method outperforms the SM method. But what about the Augmented Lagrangian Relaxation (ALR) method? One motivation to develop the RM method was to measure the quality of the local optimizers obtained by the ALR method. Let us repeat the previous test in order to compare the ALR and the RM methods. The tuning parameters for the RM method are as before and for the ALR method they are the stopping criterion with $\epsilon = 10^{-4}$, and the initial penalty parameter c_0 displayed in Table 6.6.

In Table 6.7 we observe that the performance of the ALR method and the RM method is very similar. On average the two CPU times are very close (36.4 and 35.6 seconds respectively). The same happens with the average optimal cost (PTA 537.6 and

Table 6.7: The RM method vs. the ALR method (a).

Case	CPU time (seconds)		Optimal cost (PTA)		Cost bound
	ALR	RM	ALR	RM	RM
10	2.3	2.6	96.7	96.7	84.8
20	7.9	5.7	194.7	194.7	169.8
30	5.4	7.7	292.6	292.6	256.3
40	11.1	13.0	390.6	390.6	340.8
50	19.5	11.7	488.1	488.1	427.1
60	36.4	39.2	586.4	585.9	512.0
70	68.0	80.3	686.8	683.8	597.8
80	89.3	62.2	781.8	781.7	682.7
90	49.9	81.0	880.2	879.5	767.7
100	74.5	52.3	977.8	977.3	852.5
Average	36.4	35.6	537.6	537.1	469.2

PTA 537.1 respectively). However, all the RM optimizers are global whereas some of the ALR optimizers, as for example case 70, are local optimizers. For practical purposes, we could say that the two methods have performed equivalently but for one very important detail: the RM method, unlike the ALR method, has also computed a lower bound to the optimal cost which serves as a measure of the optimum quality.

Second test

In this second test we solve again the examples given in section 6.1.3 by means of the RM method and the Subgradient Multiplier (SM) method. The tuning parameters are the initial penalty parameter $c_0 = 10^{-3}$ (except for cases 1 and 8, in which c_0 is 10^{-1} and 10^{-4} respectively), the first phase stopping criterion is defined as

$$\frac{\sum_{i=0}^{4} \|\lambda_{n-i} - \lambda_{n-i-1}\|_\infty}{5} < \epsilon_\lambda.$$

We use $\epsilon_\lambda = 10^{-4}$ for the SM method, whereas the RM method requires $\epsilon_\lambda = 10^{-5}$ to obtain dual optima of similar quality. In the second phase stopping criterion, $\epsilon = 10^{-4}$. The maximum number of multiplier updates is 200 in the two phases. Case 8 is an exception since, in order to remain within a reasonable CPU time, we have set $\epsilon_\lambda = 10^{-4}$ for the RS and SG methods, whereas the maximum number of multiplier updates in phase 1 has been set equal to 100. Finally, in the penalty parameter updating (step 2.5 of the RM algorithm) we have used $\alpha = 1.10$ and $\beta = 1.5$ (but in case 1, $\beta = 2$).

Table 6.8 displays the performance of the RM and SM methods. As in the first test, the RM method performs faster than the SM method; to be exact, on average the SM

Table 6.8: The RM method vs. the SM method (b).

Case	CPU time (seconds)			Optimal cost ($\times 10^6$ PTA)		Cost bound ($\times 10^6$ PTA)	
	RM	SM	Ratio	RM	SM	RM	SM
1	16.1	6.5	0.40	0.004	0.004	0.004	0.004
2	31.3	80.4	2.57	6.776	6.775	5.826	5.794
3	45.0	56.4	1.25	0.990	0.990	0.961	0.966
4	98.3	196.7	2.00	6.358	6.358	6.324	6.353
5	42.3	58.5	1.38	1.028	1.028	1.004	1.004
6	120.0	227.0	1.89	4.467	4.449	4.330	4.402
7	627.0	718.0	1.15	2.653	2.617	2.530	2.558
8	16019.0	27659.0	1.73	85.664	85.744	84.865	84.859
Average	2124.9	3625.3	1.55	13.493	13.496	13.231	13.243

takes 55% more CPU time than the RM method. Relatively to the optimal cost and to the lower cost bound, the results for the two methods are very similar, though on average, the SM cost bounds are slightly better. For these real-life problems, unlike for the first test problems, we do not know the global optimal cost and therefore the lower cost bound becomes crucial to assess the computed optimizer. Thus, the computed lower cost bounds confirm that the two methods obtain high quality optimizers. The main difference between these methods is, then, the superior speed of the RM method.

Third test

The next comparison is between the RM and the ALR methods. For the two methods the initial penalty parameter c_0 is set equal to 10^{-3} (but in case 1, $c_0 = 10^{-1}$). The rest of tuning parameters, for the RM method are as before (second test) and for the ALR method are as in the second phase of the RM method.

In Table 6.9 ('Ratio' column) we observe that, on average, the RM method takes a 322% more time that the ALR method. This dramatic difference would be reduced to 39% of extra time if case 8 were ignored. Let us analyze this case. Reviewing the features of case 8 in Table 6.3 we see that it corresponds to a medium-scale UC instance (1848 binary variables). For this case, the point is that the ALR optimal cost is PTA 89.122.000 and the RM one is PTA 85.896.000. Thus, the explanation for the great CPU time discrepancy is that the ALR method is trapped early on a low quality local optimizer whereas the RM eventually ends at a high quality optimizer and, furthermore, computes a lower cost bound. Case 8 is a good example of the risk one takes using the raw ALR method in combinatorial optimization because this method does not give any measure of the computed optimizer quality. We can say that in this test the ALR method, on average, has obtained worse optimizers to the

Table 6.9: The RM method vs. the ALR method (b).

Case	CPU time (seconds)			Optimal cost ($\times 10^6 PTA$)		Cost bound
	ALR	RM	Ratio	ALR	RM	RM
1	5.6	16.1	2.87	0.004	0.004	0.004
2	28.1	31.3	1.11	7.015	6.776	5.826
3	29.1	45.0	1.55	0.991	0.990	0.961
4	63.1	98.3	1.56	7.134	6.358	6.324
5	34.0	42.3	1.24	1.028	1.028	1.004
6	71.9	120.0	1.67	4.471	4.467	4.330
7	553.0	627.0	1.13	3.223	2.653	2.530
8	699.0	15836.0	22.66	89.122	85.896	84.865
Average	185.5	2102.0	4.22	14.122	13.522	13.231

RM method and a better CPU time, but without any lower cost bound.

Fourth test

So far the RM method has been compared with the SM and ALR methods. The closing test will compare the RM method versus the Radar Subgradient (RS) method. The question we whish to investigate is whether the second phase of the RM method is important or the first phase (RS method) is enough. The parameters for the RM method are as in the second test, as are the parameters for the RS method given that the RS method coincides with the first phase of the RM method. To be more precise, we set the initial penalty parameter $c_0 = 10^{-3}$ (except for cases 1 and 8, in which c_0 is 10^{-1} and 10^{-4}, respectively), the first phase stopping criterion is defined again as

$$\frac{\sum_{i=0}^{4} \|\lambda_{n-i} - \lambda_{n-i-1}\|_\infty}{5} < \epsilon_\lambda$$

with $\epsilon_\lambda = 10^{-5}$ (except for case 8, in which $\epsilon_\lambda = 10^{-4}$), and in the second phase stopping criterion, $\epsilon = 10^{-4}$ in all cases.

In Table 6.10 we can observe that, on average, the RM time is 86% greater than the RS one ('Ratio' column). In these instances x represents the output power vector, that is, the electrical power produced by each thermal unit at each period in MW, and $\tilde{x}$ is its duplication. For example, in case 2 we have that the RS method computes a primal solution whose infeasibility is 230 MW. This means that $\|x - \tilde{x}\|_\infty = 230$ MW, whereas load primal feasibility would be attained for $\|x - \tilde{x}\|_\infty = 0$ MW. Although the RS method has obtained a Load Primal Feasible (LPF) solution in case 1 (infeasibility = 0), the remaining RS solutions are highly infeasible (939 MW on average). In contrast, all the RM solutions here obtained can be considered LPF solutions for practical purposes, since $\|x - \tilde{x}\|_\infty < 10^{-2}$ MW for every case. Given

Table 6.10: The RM method vs. the RS method.

Case	CPU time (seconds)			Infeasibility (MW)	Duality gap (%)
	RS	RM	Ratio	RS	RM
1	16.0	16.1	1.01	0	0.00
2	9.8	31.3	3.19	230	16.31
3	16.3	45.0	2.76	360	3.02
4	72.4	98.3	1.36	1180	0.54
5	15.7	42.3	2.69	470	2.39
6	78.6	120.0	1.53	1390	3.16
7	467.0	627.0	1.34	530	4.86
8	15479.0	16019.0	1.03	3350	0.94
Average	2019.4	2124.9	1.86	939	3.90

that the RS method corresponds to the first phase of the RM method, we conclude that this 86% extra time is spent by the RM method to attain load primal feasibility.

Perhaps the most noteworthy aspect of the UC problem (a combinatorial problem) and Lagrangian relaxation is the fairly good quality of the computed optimizers. In the first test the method obtained the global optimum for every case. For this fourth test, as can be appreciated in Table 6.10, the RM method obtains solutions that should be well suited for practical purposes, with an average duality gap of 3.90%. This is a typical situation in separable problems with coupling constraints, such as the UC problem (see [Ber95]). This author says *'in this case, the duality gap turns out to be relatively small and can often be shown to diminish to zero relative to the optimal primal value as the number of separable terms increases'*.

Nowadays most authors solve the UC problem by Lagrangian relaxation: first, they obtain an infeasible primal solution and second, some heuristic method is applied to reach primal feasibility. One exception is [PRS96], where the Augmented Lagrangian Relaxation (ALR) method is used to reach feasibility after the dual problem is solved. In our work, we have also followed the ALR approach because we think it is superior to any heuristic method for the following reasons: (1) the ALR method reaches a (local) optimizer while a heuristic method may obtain a (local) suboptimal solution, (2) the ALR method does not depend on the UC model used whereas a heuristic method must be designed according to the UC model used, (3) furthermore, the ALR method can be used to reach feasibility in other combinatorial problems whereas a UC heuristic method, if possible to use, should be adapted for the new problem.

6.3 Summary

In non-convex optimization the Augmented Lagrangian Relaxation (ALR) method can yield local optimizers. In section 1 we presented a test of 10 examples where the ALR method reached a global optimizer for 2 cases whereas in the other 8 cases the ALR method stopped at a local optimizer. This situation can be notably improved by tuning the penalty parameter c_n properly. Empirically, the best results are obtained by starting with a small c_0 and gradually increasing it to reach the differentiability of the dual function.

In the second section, the Radar Subgradient (RS) method and the ALR method (also denoted as multiplier method) are combined to yield the Radar Multiplier (RM) method. With this two-phase procedure, firstly the RS method computes a lower cost bound by solving the dual problem and secondly, the ALR method, taking the primal-dual solution of the first phase as the starting point, computes a load primal feasible optimizer. In the run tests we have observed that: (1) If we use the subgradient method in the first phase of the RM method instead of the RS method, the CPU time notably increases while the quality of the solutions and cost bounds remains similar. (2) If we truncate the RM method and only use its second phase, that is, we use the ALR method, the whole process speeds up but as a consequence we may obtain worse optimizers and, what is the most important, we miss the lower cost bound and with it the only measure of the optimizer quality. (3) Alternatively, if we truncate the RM method after the first phase, that is, we use the RS method, we obtain primal infeasible optimizers. All in all, within the run tests, the RM method has shown to be an effective method to obtain load primal feasible optimizers for the UC problem (with a low duality gap) in a reasonable CPU time.

Chapter 7

The MACH code

The previous chapters are about developing the Radar Multiplier (RM) method intended to solve the Generalized Unit Commitment (GUC) problem. In the current chapter, first we show the design of the MACH code, which implements the RM method. Then real-life large-scale UC and GUC instances are solved in order to check MACH's performance. Solutions computed in this chapter are Load Primal Feasible (LPF) but not Full Primal Feasible (FPF) (see section 3.5 for more details). Next chapter is devoted to compute FPF solutions.

7.1 Designing the MACH code

In Spanish MACH stands for *Modelo Acoplado de Coordinación Hidrotérmica*, which means *Coupled Model for Hydrothermal Coordination*, that is, MACH is a code intended to solve the GUC problem using the coupled model (see chapter 2). We could say that MACH's father is MAPH3 [Her97], a code developed to solve the Optimal Power Flow (OPF) problem. What MACH does is to use MAPH3 as a routine to solve the OPF subproblem associated with the GUC problem (see chapter 1), that is, MACH enhances MAPH3 by searching for an LPF optimal unit commitment solution which is also optimal for the OPF problem.

Four optimization points have been studied in this work. First, in chapter 3 we saw that, regarding the theoretical quality of the dual bounds, the variable duplication method is, at least, as good as the classical Lagrange relaxation method. Second, chapter 4 was entirely devoted to comparing two decomposition methods for the augmented Lagrangian function, with the conclusion that the block coordinate descent method outperforms the auxiliary problem principle method. Third, in chapter 5 we proposed the radar subgradient method, a new dual method that drastically reduces the number of subgradient iterations required to solve the dual problem, as well as the total CPU time. And fourth, at the beginning of chapter 6 we proposed a heuristic penalty updating method which improves the quality of the computed local optimizers when the multiplier method is used. All in all, after comparing classical and new methods, the RM method was designed by taking the best options.

The RM method was expounded in chapter 6 without paying any attention to the computational issues. We now show the RM method from a computational point of view or, what is the same, we display the MACH code. Regarding the programming language, the MACH code and all the routines used have been implemented in standard FORTRAN 77.

MACH code

* [Objectives.] To compute a cost lower bound and a high quality local LPF optimizer for the GUC problem.

Phase 1 [Compute a cost lower bound $\underline{f}^*$.]

Step 1.0 [Initialize.] Set $n = 0$, and ϵ_λ for the stopping criterion.

Step 1.1 [Solve the hydrothermal subproblem.] We solve the hydrothermal subproblem described in chapter 3

$$\begin{array}{c} \min\ f(x) + \lambda_n' x \\ x \in \mathcal{D} \end{array}$$

using the MAPH3 code as a routine (note that now the Lagrangian term is $\lambda_n' x$ instead of $\lambda_n' x + \frac{c_n}{2}\|x - \tilde{x}_n\|^2$). The MAPH3 routine is based on the NOXCB nonlinear network flow solver, since the subproblem we have to solve, the hydrothermal subproblem, falls into the class of nonlinear network flow problems with side constraints. More details about MAPH3 and NOXCB can be found in [HN95, Her97].

Step 1.2 [Solve the thermal subproblem.] The thermal subproblem

$$\begin{array}{c} \min\ \tilde{f}(\tilde{x}) - \lambda_n' \tilde{x} \\ x \in \tilde{\mathcal{D}} \end{array}$$

falls into the class of separable nonlinear mixed integer programming problems. It is solved by dynamic programming, details of which are fully explained in chapter 3.

Step 1.3 [Check the stopping criterion.] If λ_n does not improve for the last N_{iter} iterations or n reaches a prefixed value then go to phase 2.

Step 1.4 [Compute the step length β_n by the radar subgradient routine.]

Step 1.5 Compute $\lambda_{n+1} = \lambda_n + \beta_n \cdot s_n$. Set $n = n + 1$ and go back to step 1.1.

Phase 2 [Compute $(x^*, \tilde{x}^*)$ a local LPF optimizer for the GUC problem.]

Step 2.0 [Initialize.] Set $n = 0$, ϵ for the stopping criterion, and $(x_0, \tilde{x}_0, \lambda_0)$ equal to the optimal point obtained in phase 1. Initialize c_0 to a small value, say, $c_0 \in [10^{-3}, 10^{-5}]$.

Step 2.1 [Check the stopping criterion.] If the norm of the gradient of the dual function $\|x_n - \tilde{x}_n\|_\infty < \epsilon$ then stop. $(x_n, \tilde{x}_n)$ is the desired solution.

Step 2.2 [Solve the hydrothermal subproblem.] (MAPH3 routine)

$$\begin{array}{l} \min \;\; f(x) + \lambda_n' x + \frac{c_n}{2}\|x - \tilde{x}_n\|^2 \\ x \in \mathcal{D} \end{array}$$

Step 2.3 [Solve the thermal subproblem.] (Dynamic programming)

$$\begin{array}{l} \min \;\; \tilde{f}(\tilde{x}) - \lambda_n' \tilde{x} + \frac{c_n}{2}\|x_{n+1} - \tilde{x}\|^2 \\ x \in \tilde{\mathcal{D}} \end{array}$$

Note that with steps 2.2 and 2.3 we are minimizing the augmented Lagrangian by the Block Coordinate Descent (BCD) method (chapter 4). A pure BCD procedure will alternate steps 2.2 and 2.3 repeatedly until it reaches the minimum of the augmented Lagrangian for the current λ_n. However, given that this value of λ_n is a mere approximation to λ^*, it is not worth such a great computational effort. Alternatively, we only perform one iteration of the BCD method (steps 2.2 and 2.3) which, from our computational experience, turns out to be more efficient than the pure BCD method.

Step 2.4 [Dual variable updating.]

$$\lambda_{n+1} = \lambda_n + c_n \cdot (x_{n+1} - \tilde{x}_{n+1})$$

Step 2.5 [Penalty parameter updating.]

Step 2.6 Set $n = n + 1$ and go back to step 2.1.

7.2 A real-life large-scale unit commitment instance

Till now, when solving a Unit Commitment (UC) instance, we have paid little attention to the instance itself. Our objective now is to solve a real-life large-scale UC instance, focusing on the data which describe the instance, on the solution process and on the computed solution.

7.2.1 Instance description

The data files for this UC instance are dwhxx481 for the hydraulic system, and dwh17004894 for the thermal system (note that being a UC instance there is no transmission network). The optimization horizon is 48 hours, whereas the generating system is composed of 20 reservoirs and 70 thermal units. In total the number of binary variables is 3360 and the number of continuous ones is 16320. So far, this UC instance is the largest one solved using the MACH package.

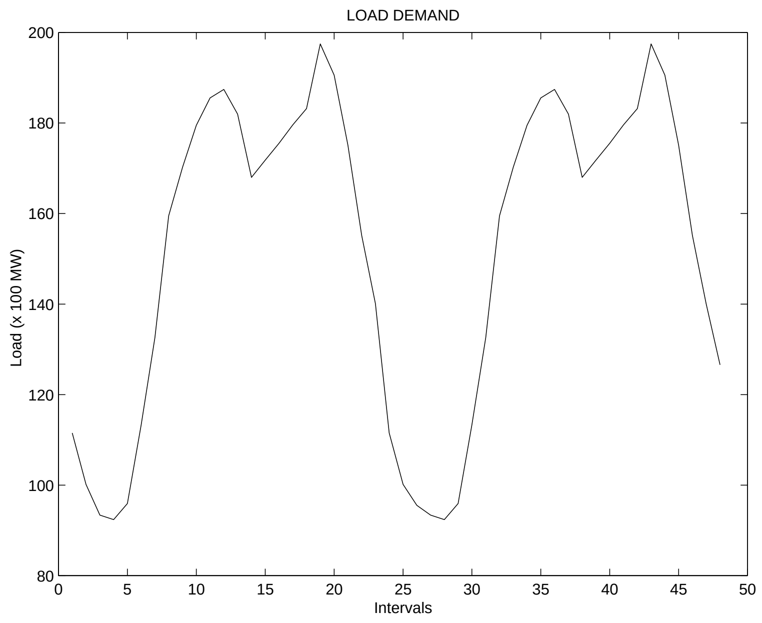

Figure 7.1: Load demand.

The main details of the numerical description of the thermal system are listed in Tables 7.1 and 7.2. That is, for each thermal unit the table defines its working power interval $[P_{min}, P_{max}]$, its thermal cost function $TC(x) = c_b + c_l x + c_q x^2$ for an output power $x \geq P_{min}$, the minimum down-time and the minimum up-time.

Table 7.3 displays the evolution of the load demand over a 48-hour period divided into intervals of one hour each. The incremental spinning reserve is 510 MW in each interval for the whole period and the decremental spinning reserve (DSR) is displayed in the same table. Load demand over the 48-hour period is depicted in Figure 7.1.

The hydraulic system, composed of twenty reservoirs distributed over two hydro-valleys, is depicted in Figure 7.2. Each reservoir has a discharge (D) and a spillage (S) arc connected to the next reservoir with the exception of reservoir R11. The function of this reservoir is to store water to be pumped to reservoir R08 during low demand periods and thus to be reused at peak demand periods. Chapter 2 briefly introduces the hydraulic model and its variables, and full details can be found in [HN95, Her97].

7.2.2 Solution process

Before starting the optimization process some parameters and criteria must be specified. Among them, the first phase stopping criterion

$$\frac{\sum_{i=0}^{4} \|\lambda_{n-i} - \lambda_{n-1-i}\|_\infty}{5} < \epsilon_\lambda \tag{7.1}$$

with $\epsilon_\lambda = 10^{-4}$, the initial penalty parameter $c_0 = 10^{-4}$ and its updating formula $c_{n+1} = 1.1c_n$ (the penalty parameter is updated following the heuristic method de-

Table 7.1: Description of the thermal system. P_{min} = minimum generating power, P_{max} = maximum generating power, thermal cost(x)= $c_b + c_l x + c_q x^2$, MD= minimum down-time, MU= minimum up-time.

Unit	P_{min} (MW)	P_{max} (MW)	c_b (PTA)	c_l (PTA/MW)	c_q (PTA/MW^2)	MD (h)	MU (h)
U01	100	289	290.40	8.44	0.2675D−02	3	6
U02	100	289	286.61	8.44	0.2633D−02	3	6
U03	140	541	421.74	8.54	0.1254D−02	3	6
U04	200	350	749.49	6.17	0.6897D−02	6	24
U05	10^{-10}	990	0.00	4.18	0.0000D+00	6	24
U06	103	214	237.49	7.84	0.8318D−02	6	24
U07	140	542	415.70	8.88	0.7524D−03	3	6
U08	140	541	421.74	8.54	0.1254D−02	3	6
U09	140	377	459.37	8.31	0.1003D−02	3	6
U10	75	155	565.15	2.21	0.3407D−01	6	24
U11	40	140	81.04	9.79	0.0000D+00	3	6
U12	74	148	67.70	11.12	0.0000D+00	2	5
U13	150	350	158.63	9.69	0.0000D+00	2	5
U14	70	141	257.17	9.17	0.0000D+00	2	5
U15	70	141	199.39	9.55	0.0000D+00	2	5
U16	200	330	306.29	9.12	0.0000D+00	2	5
U17	175	350	233.16	9.26	0.0000D+00	2	5
U18	205	350	193.47	9.38	0.0000D+00	2	5
U19	230	350	-37.02	12.92	0.0000D+00	2	5
U20	230	350	-37.02	12.92	0.0000D+00	2	5
U21	210	350	12.38	10.87	0.0000D+00	2	5
U22	185	313	121.26	10.04	0.0000D+00	2	5
U23	100	220	321.66	9.18	0.0000D+00	2	5
U24	230	350	-37.02	12.92	0.0000D+00	2	5
U25	230	350	-37.02	12.92	0.0000D+00	2	5
U26	210	350	12.38	10.87	0.0000D+00	2	5
U27	210	350	12.38	10.87	0.0000D+00	2	5
U28	44	80	83.84	9.61	0.0000D+00	2	5
U29	45	150	142.12	9.28	0.4180D−04	3	6
U30	90	300	256.06	8.00	0.3804D−02	3	6
U31	100	533	433.96	7.83	0.2132D−02	3	6
U32	100	533	392.58	8.38	0.1881D−02	3	6
U33	90	300	212.08	8.68	0.2466D−02	3	6
U34	55	172	125.40	10.37	0.0000D+00	3	6
U35	100	350	307.02	8.75	0.2299D−02	3	6

Table 7.2: Description of the thermal system (cont.).

Unit	P_{min} (MW)	P_{max} (MW)	c_b (PTA)	c_l (PTA/MW)	c_q (PTA/MW^2)	MD (h)	MU (h)
U36	100	350	307.02	8.75	0.2299D–02	3	6
U37	100	350	257.34	9.21	0.2090D–02	3	6
U38	55	172	125.40	10.37	0.0000D+00	3	6
U39	100	350	307.32	8.75	0.2299D–02	3	6
U40	100	350	257.34	9.21	0.2090D–02	3	6
U41	99	160	141.71	10.69	0.0000D+00	2	5
U42	87	160	133.99	9.54	0.0000D+00	2	5
U43	180	360	228.90	9.87	0.0000D+00	2	5
U44	260	543	284.55	9.24	0.0000D+00	2	5
U45	68	313	168.72	8.56	0.2341D–02	3	6
U46	175	350	570.47	5.89	0.5852D–02	6	24
U47	10^{-10}	930	0.00	4.18	0.0000D+00	6	24
U48	10^{-10}	1066	0.00	4.18	0.0000D+00	6	24
U49	176	550	484.20	8.48	0.4180D–03	6	24
U50	180	550	420.22	7.79	0.2006D–02	6	24
U51	61	313	168.72	8.56	0.2341D–02	3	6
U52	10^{-10}	30	0.00	4.18	0.0000D+00	6	24
U53	10^{-10}	930	0.00	4.18	0.0000D+00	6	24
U54	10^{-10}	930	0.00	4.18	0.0000D+00	6	24
U55	10^{-10}	460	0.00	4.18	0.0000D+00	6	24
U56	160	254	27.45	10.51	0.1003D–02	6	24
U57	10^{-10}	1004	0.00	4.18	0.0000D+00	6	24
U58	180	350	192.23	9.57	0.0000D+00	2	5
U59	66	220	240.09	8.23	0.3428D–02	3	6
U60	160	533	627.29	7.11	0.2968D–02	3	6
U61	43	148	125.40	10.37	0.0000D+00	3	6
U62	48	160	125.40	10.37	0.0000D+00	3	6
U63	40	120	107.08	9.40	0.0000D+00	3	6
U64	100	350	502.00	6.76	0.6145D–02	3	6
U65	10^{-10}	160	0.00	4.18	0.0000D+00	6	24
U66	142	270	171.05	9.35	0.0000D+00	2	5
U67	220	350	188.79	9.90	0.0000D+00	2	5
U68	85	154	110.68	9.71	0.0000D+00	2	5
U69	220	350	149.31	9.76	0.0000D+00	2	5
U70	290	550	492.81	11.45	0.0000D+00	2	5

Table 7.3: Hourly load demand.

Hour	Load (MW)	Hour	Load (MW)
1	11154.14	25	10017.18
2	10017.18	26	9554.34
3	9338.50	27	9338.50
4	9237.04	28	9237.04
5	9594.24	29	9594.24
6	11330.46	30	11330.46
7	13260.86	31	13260.86
8	15948.98	32	15948.98
9	17024.38	33	17024.38
10	17950.06	34	17950.06
11	18553.50	35	18553.50
12	18743.12	36	18743.12
13	18196.68	37	18196.68
14	16799.80	38	16799.80
15	17178.66	39	17178.66
16	17553.34	40	17553.34
17	17958.80	41	17958.80
18	18320.18	42	18320.18
19	19747.84	43	19747.84
20	19056.24	44	19056.24
21	17509.26	45	17509.26
22	15517.30	46	15517.30
23	14005.66	47	14005.66
24	11154.14	48	12657.04

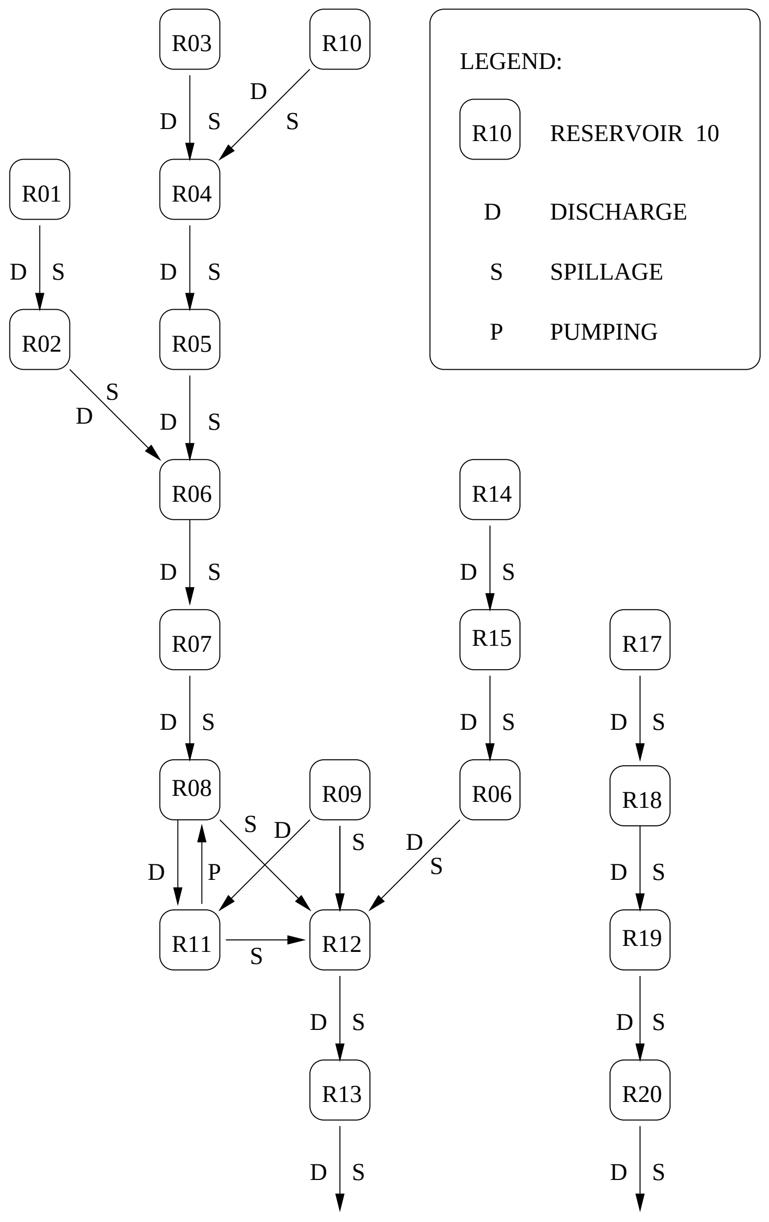

Figure 7.2: Hydraulic system.

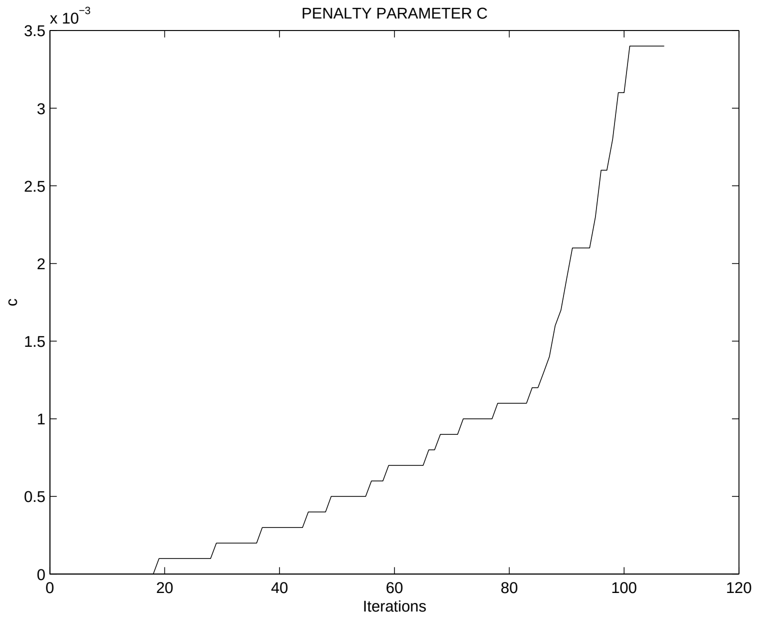

Figure 7.3: Evolution of the penalty parameter c_n.

scribed in chapter 6), the second phase stopping criterion with $\epsilon = 10^{-2}$, and the initial radar step length equal to 10^{-3}.

The evolution of the penalty parameter c_n, the dual function value $q(\lambda_n)$ and the infeasibility $\|x_n - \tilde{x}_n\|_\infty$, during the optimization process, can be studied graphically to obtain a better understanding of the whole process. In Figure 7.3 we can observe: first, that the penalty parameter is 0 for the first 17 iterations (phase 1 of the Radar Multiplier (RM) method), second, that the penalty parameter remains unchanged at some iterations as a consequence of the heuristic updating procedure, and third, the final value of the penalty parameter is under 3.5×10^{-3}, which confirms that we do not need a high penalty term to reach convergence (a well known advantage of the multiplier method versus pure penalty methods [Ber95]).

Figure 7.4 shows the value of the dual function for each iteration. Not much can be said about this function except to draw attention to its rapid stabilization at some value above 10. To gather more details, let us focus first on the dual function for the first phase and then do the same for the second phase. Figure 7.5 is an amplification of Figure 7.4 for the first 17 iterations which corresponds to the radar subgradient method. Analogously, Figure 7.6 is an amplification for the following 89 iterations which corresponds to the multiplier method. The differences between Figures 7.6 and 7.4 are striking (recall that in Figure 7.6 we have only changed the scale of the y-axis but not the function). The explanation for this zigzag behavior is that the multiplier method does not perform any linear search since it uses the current penalty parameter as the step length.

The success of the RM method depends on whether the measure of the infeasibility $\|x_n - \tilde{x}_n\|_\infty$ is small enough at the final point $(x_n, \tilde{x}_n)$. In Figure 7.7 we observe that initially, at iteration 1, this infeasibility is above 1000 MW (10×100MW),

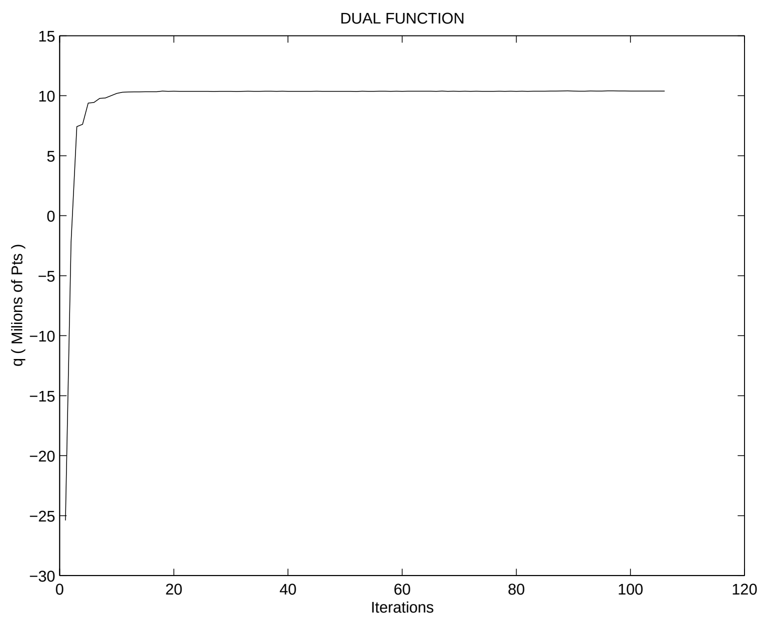

Figure 7.4: Evolution of the dual function.

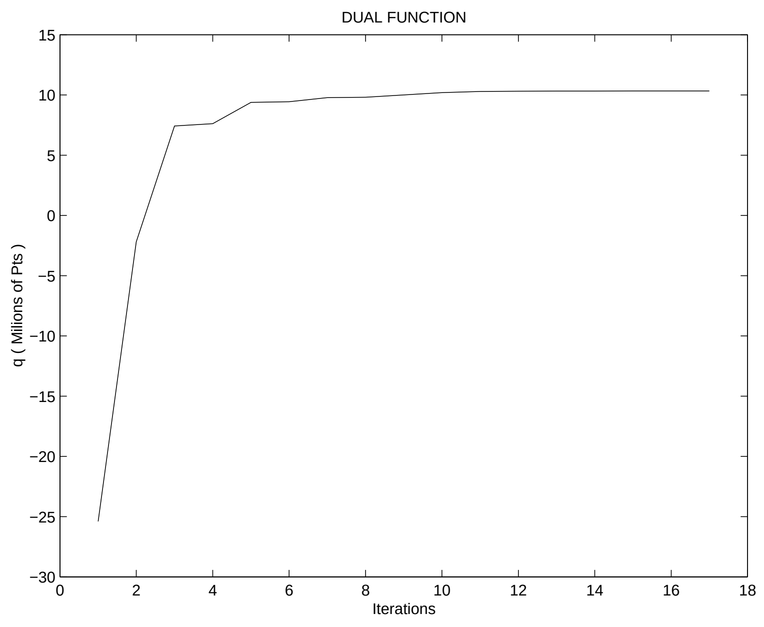

Figure 7.5: Evolution of the dual function (phase 1).

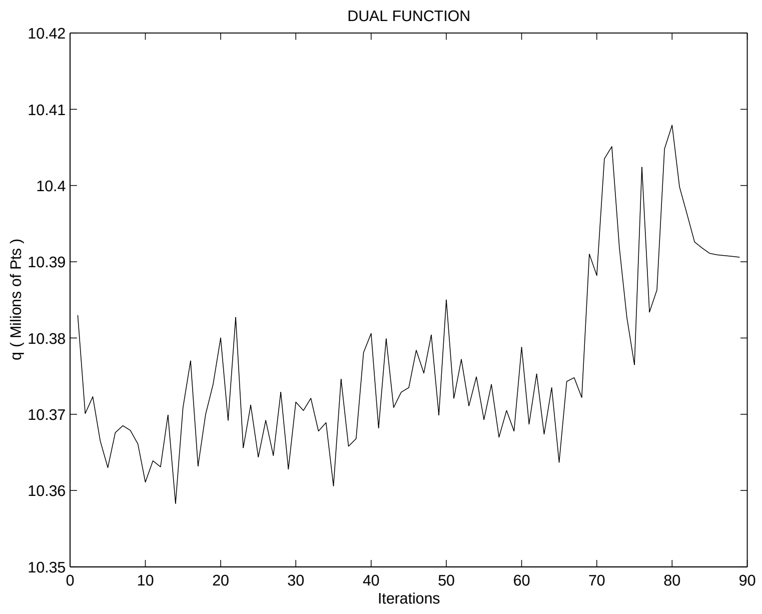

Figure 7.6: Evolution of the dual function (phase 2).

that is, the output power x_1 computed in the hydrothermal subproblem differs from the output power $\widetilde{x}_1$ computed in the thermal subproblem by more than 1000 MW at a component i ($x_1^i - \widetilde{x}_1^i > 1000$ MW). Infeasibility remains high during phase 1 and suddenly drops at around 200 MW at the moment phase 2 begins. From iteration 18 to iteration 106 infeasibility zigzags down till finally it reaches its lowest value ($\|x_{106} - \widetilde{x}_{106}\|_\infty = 0.88MW$). For a more detailed account of the infeasibility evolution during phase 1 and phase 2 the reader is referred to Figures 7.8 and 7.9 respectively.

As was stated in chapter 3, the hydrothermal subproblem is solved with a successively linearized hydro-generation function $h_{L_r}^i$. This method was already successfully tested in [HN95]. Figure 7.10 shows both the linearized and the non-linearized (hydrothermal) generation. The linearized generation corresponds to the optimal generation obtained with MACH, whereas the non-linearized generation corresponds to the exact generation we would obtain if the optimal production policy were to be implemented. Note that the two generations are very close.

7.2.3 Solution description

The most important optimization results are displayed in Table 7.4. There are, in total, 106 iterations, which is the sum of the 17 first phase iterations and the 89 second phase iterations. A total CPU time of 4.68 hours is a very reasonable time considering the size (70 thermal units, 20 reservoirs and 48 intervals) and the difficulty of the problem (a non linear mixed integer problem). As already mentioned above, the infeasibility $\|x^* - \widetilde{x}^*\|_\infty$ is 0.88 MW, which should be good enough for practical purposes. Note that, for numerical reasons, we solve a scaled version of the

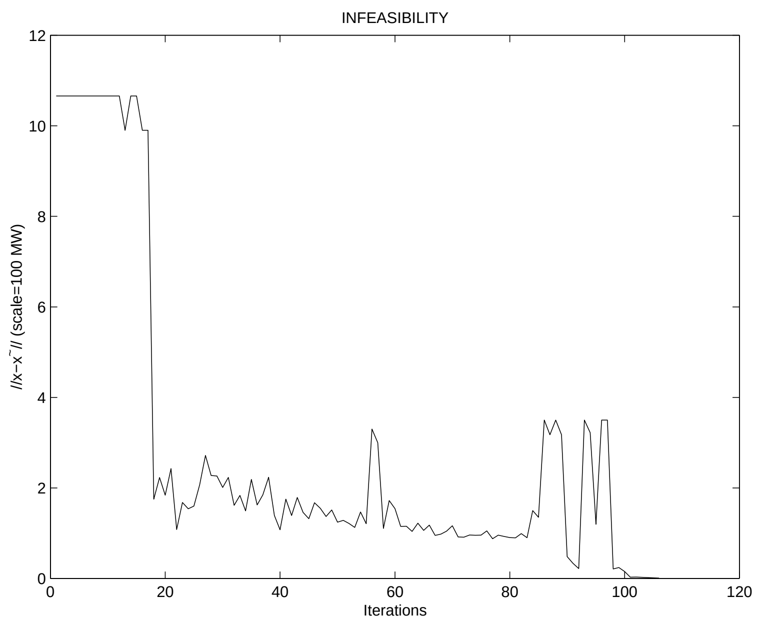

Figure 7.7: Evolution of the infeasibility.

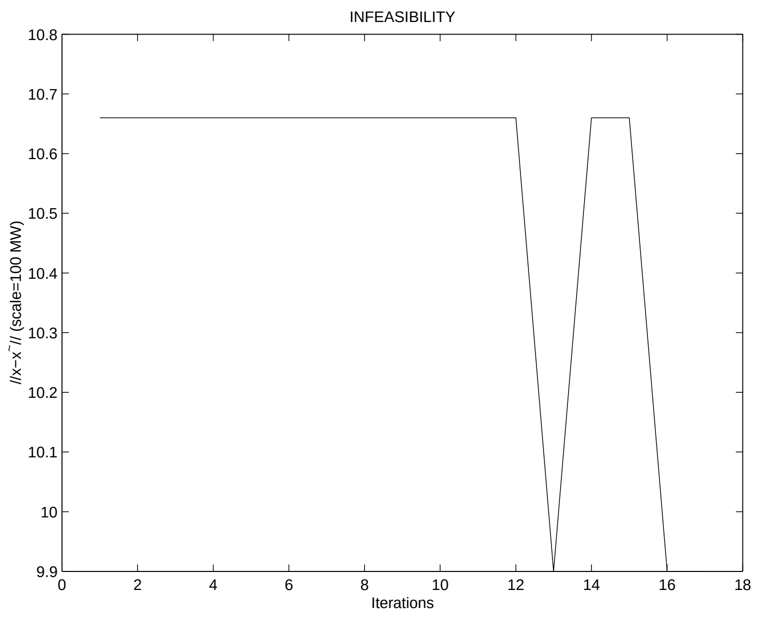

Figure 7.8: Evolution of the infeasibility (phase 1).

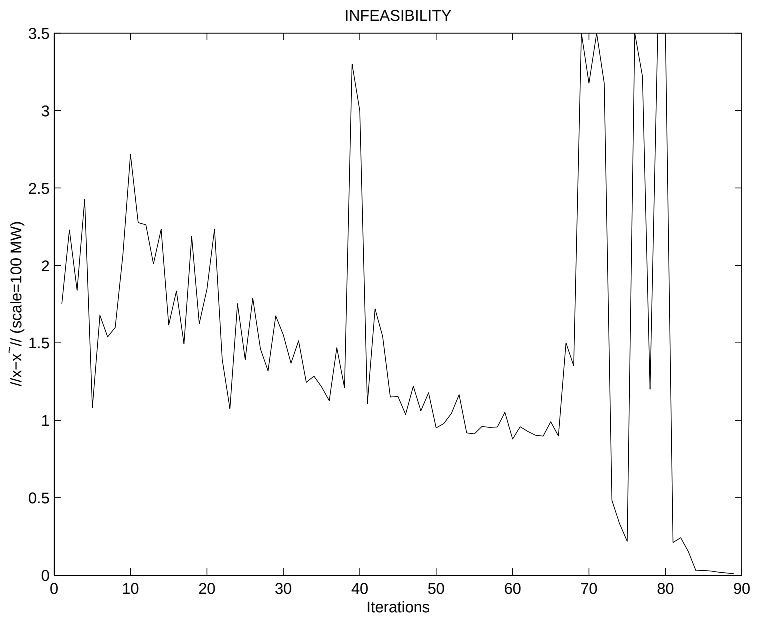

Figure 7.9: Evolution of the infeasibility (phase 2).

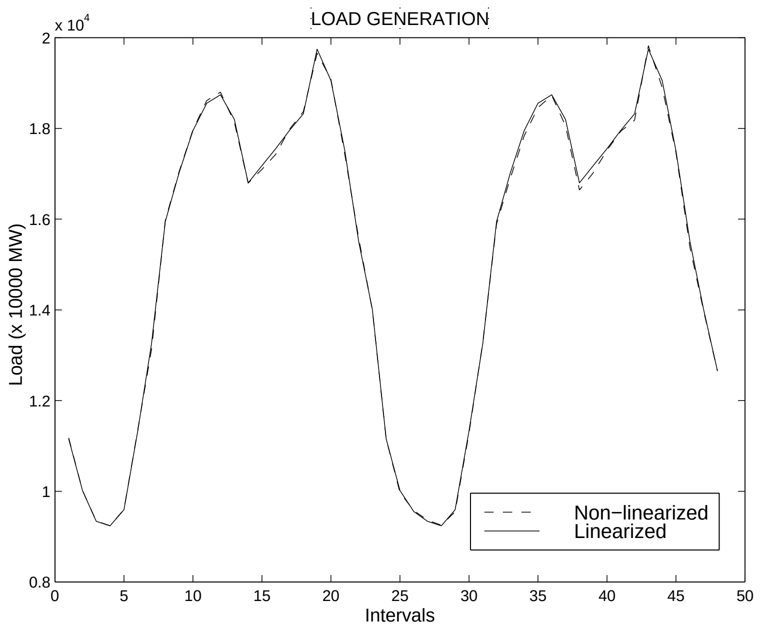

Figure 7.10: The linearized generation is a good approximation to the non-linearized generation.

Table 7.4: Optimization results.

Iterations	Time (hours)	Dual bound	Opt. cost ($\times 10^6 PTA$)	Duality gap (%)	$\|x^* - \tilde{x}^*\|_\infty$ (MW)
106	4.68	10.340	10.391	0.49	0.88

current UC instance, so as the scaled infeasibility of the last iterate is $\|x^* - \tilde{x}^*\|_\infty = 0.0088$, which is lower than the stopping threshold ($\epsilon = 10^{-2}$). However, we report the true value for $\|x^* - \tilde{x}^*\|_\infty$, which is 0.88 MW. Nevertheless, perhaps the best characteristic of the computed local optimal cost (10.391 $\times 10^6$ PTA) is its high quality, since the global optimal cost is bounded from below by the dual bound 10.340 $\times 10^6$ PTA, which is equivalent to a duality gap of only 0.49%.

After the description of the main optimization results, we show the solution graphically. Figure 7.11 represents the optimal UC pattern, which is completely compatible with all the management requirements (minimum up and down-times, initial state specifications, etc.). It is composed of 70 rows (units) of 48 squared intervals each; a white square means the thermal unit is on during that hour (analogously with black squares). Basically three working modes can be observed: first, units that are off for the whole period, second, units that are on for the whole period, and third, units that only work at peak load periods.

Knowing the UC pattern is not enough, the optimal solution must also give the level of power to be produced by each unit. This economic hydrothermal dispatching is represented in Figure 7.12. In accordance with the UC pattern, some units dispatch power for the whole period and more expensive ones only work at peak load periods. Note that the cheapest but limited power, the hydraulic one, is saved for the peak load periods (top layer in Figure 7.12).

7.3 Solving generalized unit commitment instances using MACH

Although in chapter 2 we modeled the Generalized Unit Commitment (GUC) problem and in chapter 3 we developed a solution methodology, so far only instances of the Unit Commitment (UC) problem have been solved (recall that, unlike the GUC problem, the UC problem does not take into account the transmission network). As pointed out in [WS$^+$95], among others, the inclusion of the transmission network within the model is important because UC solutions can be incompatible with the transmission network. More precisely, the UC solution may overload some transmission lines, specially during the peak load period. Therefore, the usefulness of the GUC model is proportional to the saturation of the transmission lines. The other important aspect contributed by the GUC problem is a solution which is an optimal

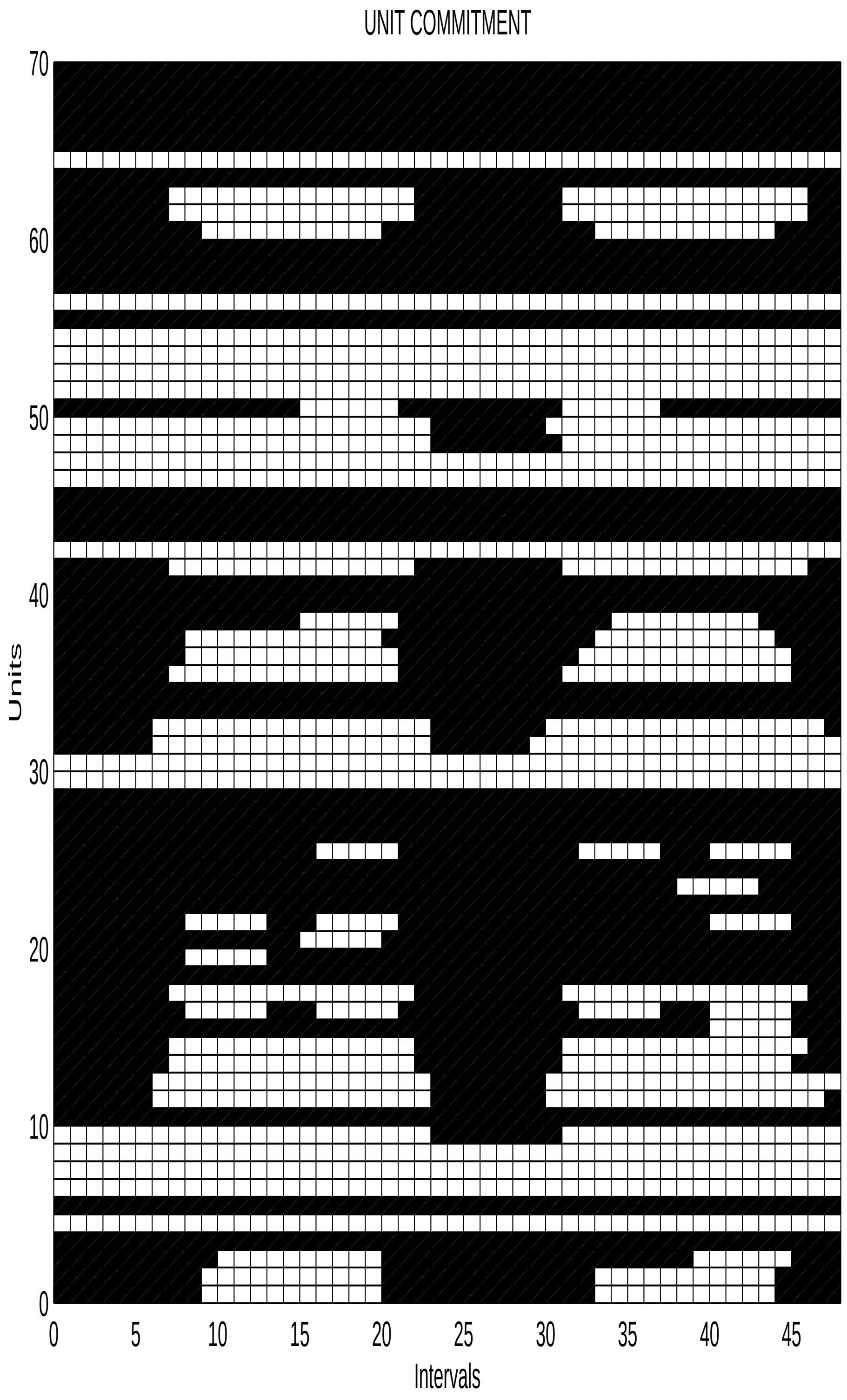

Figure 7.11: Optimal unit commitment pattern.

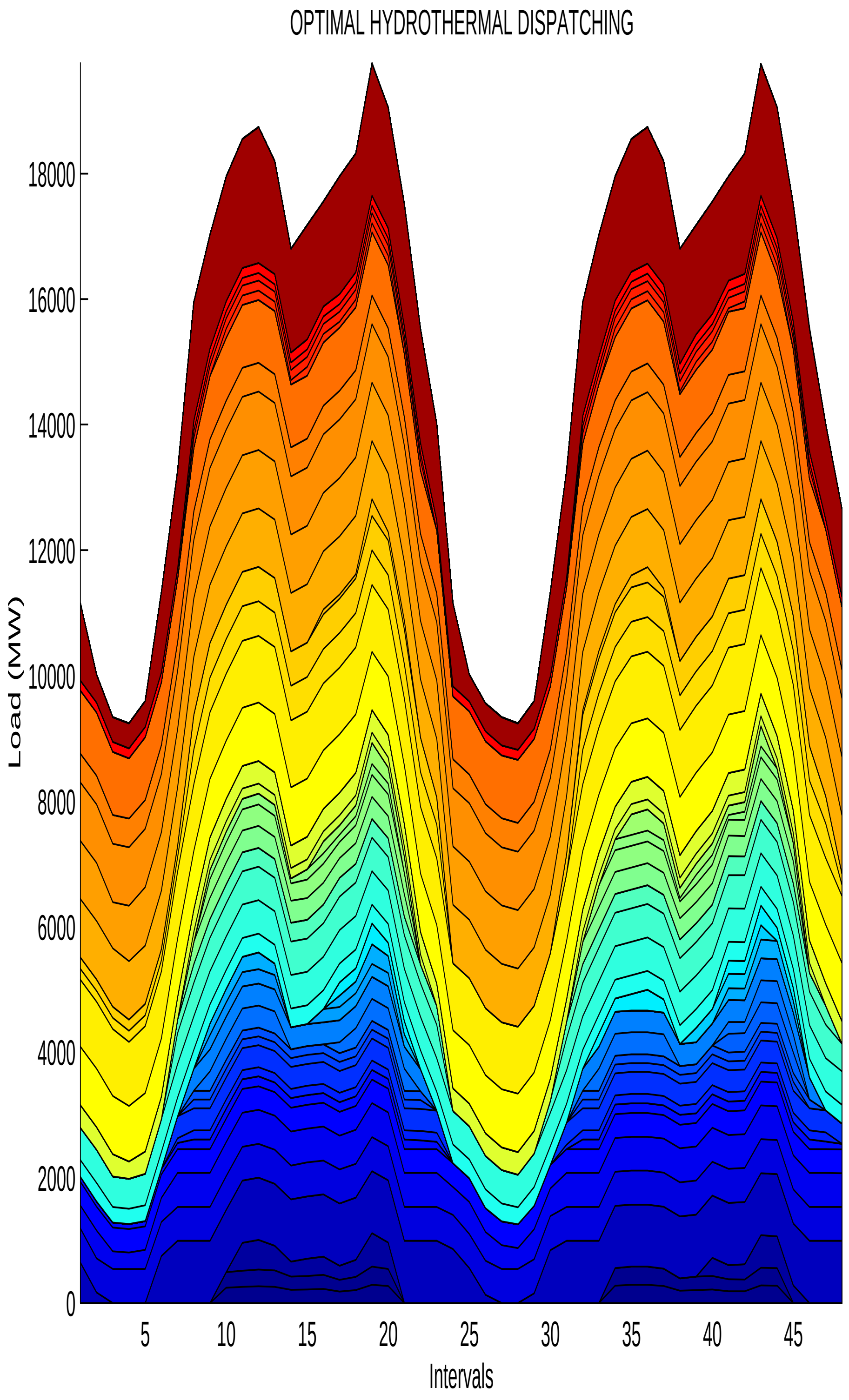

Figure 7.12: Economic dispatching.

Table 7.5: Solved GUC instances.

Case	Hydraulic data file	Thermal data file	Transmission data file
1	hz020	ti0200294	xi030300201
2	hp060	ti0400691	xi060
3	dwhxx241	dwh17002494	dwh1xx241
4	dwhx481	dwh17004894	dwh1x481
5	hz480	dwh02704894	dwh0x482
6	hp480	thunit40404891	xi483
7	reda481	thunit40404896	xibd483
8	hp480	thunit40704894	xibd484

Table 7.6: Description of the GUC instances. Int.= intervals, Res.= reservoirs, Units= thermal units, Cont.= continuous variables, Bin.= binary variables.

Case	Int.	Res.	Units	Buses	Lines	Cont.	Bin.
1	2	0	2	3	3	28	4
2	6	2	4	3	6	192	24
3	24	20	70	6	10	8568	1680
4	48	10	70	6	10	15696	3360
5	48	0	27	6	21	6528	1296
6	48	2	4	3	6	1536	192
7	48	4	4	2	3	1632	192
8	48	2	7	2	3	1920	336

power flow, that is, one of the cheapest load primal feasible flows through the transmission network. In contrast, the optimal solution provided by the UC problem will probably not be optimal when the transmission losses are considered.

In this section we will briefly report the results obtained with MACH when solving a set of eight GUC instances and in the next section we will focus on instance number 4. The data files for the eight GUC instances to be solved are displayed in Table 7.5 and their main features in Table 7.6 (note the large-scale size of the problems 3, 4 and 5).

Regarding the tuning parameters, the first phase stopping criterion is defined as in (7.1) with ϵ_λ displayed in Table 7.7 together with the other tuning parameters.

In Table 7.8 we can see that MACH performed well. As usual, the number of iterations in phase 1 is independent of the dimension of the instance and usually the RM method takes around 30 iterations to compute the dual bound. The number of

Table 7.7: Tuning parameters. c_0 = initial penalty parameter, ϵ_λ = first phase stopping criterion, ϵ = second phase stopping criterion, r_step = initial radar step length, and $c_{k+1} = c_k \cdot c\_var$.

Case	c_0	ϵ_λ	ϵ	r_step	c_var
1	10^{-3}	10^{-5}	10^{-4}	10^{-0}	1.5
2	10^{-3}	10^{-5}	10^{-4}	10^{-0}	1.1
3	10^{-3}	10^{-5}	10^{-2}	10^{-3}	1.1
4	10^{-3}	10^{-5}	10^{-2}	10^{-3}	1.1
5	10^{-3}	10^{-5}	10^{-2}	10^{-3}	1.1
6	10^{-3}	10^{-5}	10^{-4}	10^{-3}	1.1
7	10^{-3}	10^{-5}	10^{-4}	10^{-3}	1.1
8	10^{-3}	10^{-5}	10^{-4}	10^{-3}	1.1

iterations in phase 2 is more variable: from 0 iterations (case1) up to 280 iterations (case 2). Once again the number of iterations in phase 2 seems to be independent of the instance dimension. However, the total CPU time, as usual, strongly depend on the dimension of the instance. As it can be apreciated under label 'Time' all the cases are solved within 2 hours. The RM method takes 6282 seconds to solve case 4, which is the largest GUC instance successfully solved with MACH (15696 continuous variables and 3360 binary variables). Furthermore, Table 7.8 shows that the inclusion of the transmission network does not disrupt the performance of MACH compared to cases without transmission network (see chapter 6).

The duality gap in the small-size cases 2 and 7 is high (10.79% and 7.85% respectively), but this is not cause for concern since it is well known that in the case of separable problems with coupling constraints, as is the case of the GUC problem, (see [Ber95]) *'the duality gap turns out to be relatively small and can often be shown to diminish to zero, relative to the optimal primal value as the number of separable terms increases'*. Note that for the large-scale cases 3, 4 and 5 the duality gap is lower than the average (3.43%).

7.4 A real-life large-scale generalized unit commitment instance

In this section we focus on the Generalized Unit Commitment (GUC) instance labeled as case 4 in the above section. This GUC instance has been derived from the Unit Commitment (UC) instance solved in section 7.2 by also considering the transmission network. Although the UC instance in section 7.2 considered a 20-reservoir hydraulic system, the current GUC instance (case 4) considers only 10 reservoirs because the current version of MACH failed to solve the 20-reservoir version.

Table 7.8: Computational results using MACH.

Case	Iterations		Time (sec.)	Dual bound	Opt. cost ($\times 10^6$ PTA)	Duality gap (%)
	phase 1	phase 2				
1	41	0	19	0.005	0.005	0.00
2	31	280	134	9.038	10.006	10.71
3	32	64	3604	5.522	5.590	1.23
4	32	84	6282	12.183	12.397	1.76
5	32	167	3076	8.713	8.890	2.03
6	17	185	161	1.396	1.404	0.57
7	34	9	163	7.181	7.745	7.85
8	17	77	67	1.193	1.232	3.27
Average	30	108	1688	5.654	5.909	3.43

7.4.1 Instance description

The data files for this GUC instance are dwhx481 for the hydraulic system, dwh17004894 for the thermal system, and dwh1x481 for the transmission network. The description of the thermal system can be found in section 7.2, whereas the hydraulic system corresponds to the first 10 reservoirs of the hydraulic system also used in section 7.2 (R01,..., R10). Load demand throughout the 48-hour period is represented in Figure 7.1. In addition, the topology of the transmission network is depicted in Figure 7.13, and the numerical description of the transmission lines is in Table 7.9. Remember that, as described in chapter 2, the transmission system is modeled as a direct current network. Furthermore, this network is a simplified representation of the real transmission network, since the 70 thermal units and the 10 reservoirs are grouped into only 6 production-consumption nodes (N01, ..., N06).

7.4.2 Solution process

The tuning parameters are those of case 4 in Table 7.7. As in section 7.2, we analyze the evolution of the penalty parameter c_n, the dual function value $q(\lambda_n)$, and the infeasibility $\|x_n - \tilde{x}_n\|_\infty$, during the optimization process, to gain a better understanding of the whole optimization process. In Figure 7.14 note that, as is well known [Ber95], using an augmented Lagrangian the penalty parameter c_n does not need to be increased excessively. For our case $c_n < 0.04$ for all n and hence ill conditioning of the objective function is avoided.

Figure 7.15 shows the value of the dual function for each iteration. The first 32 iterations correspond to the first phase of the Radar Multiplier (RM) method and the following 84 to the RM second phase. We can see that with the RM method

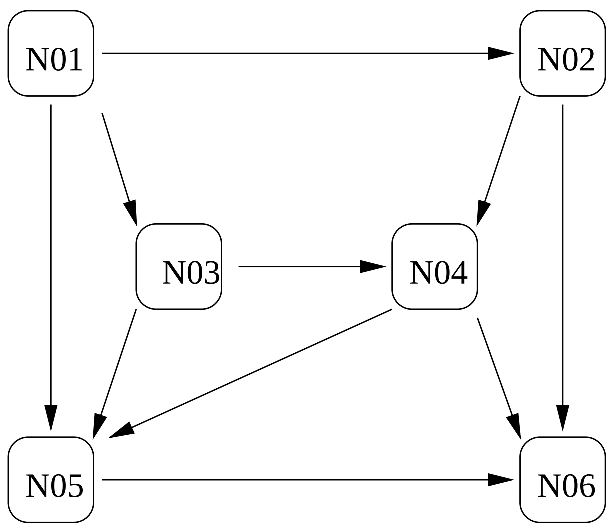

Figure 7.13: Topology of the transmission network.

Table 7.9: Description of the transmission network. V= voltage, L= length, R= resistance, X= reactance, C= capacity.

Line		V	L	R	X	C
		(kV)	(km)	(Ω/km)	(Ω/km)	(A)
from	to					
N01	N02	380	110	0.02037	0.1675	3620.0
N01	N03	380	32	0.01492	0.1033	5160.0
N02	N06	380	128	0.02920	0.2451	2580.0
N03	N04	380	37	0.01735	0.1474	4300.0
N03	N05	380	28	0.01492	0.1033	5160.0
N04	N06	380	24	0.02960	0.1476	2580.0
N05	N06	380	75	0.01915	0.1595	3870.0
N01	N05	380	41	0.02090	0.1472	2580.0
N02	N04	380	106	0.05890	0.3026	1290.0
N04	N05	380	27	0.01440	0.1255	5160.0

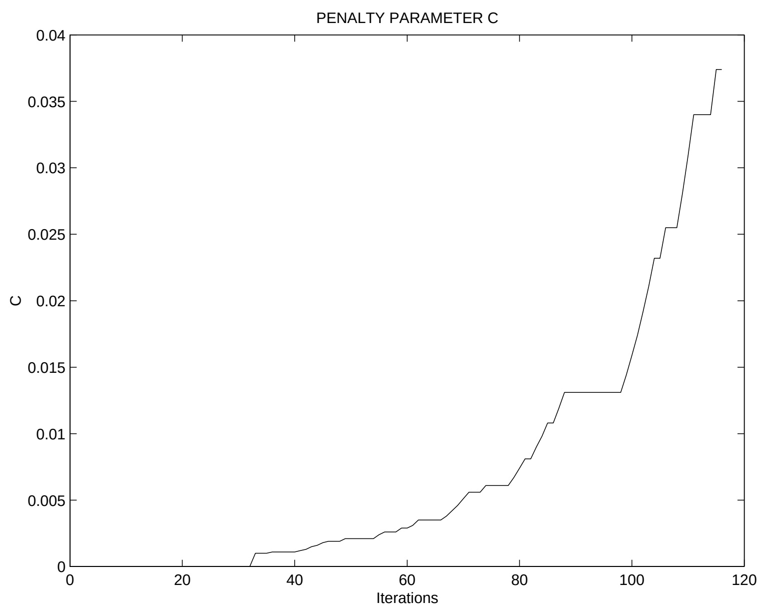

Figure 7.14: Evolution of the penalty parameter c_n.

there is no guaranteed improvement of the dual function at each iteration.

In Figure 7.16 we observe that, as usual, in the first phase of the RM method the primal infeasibility, regarding the relaxed constraint $x_n - \widetilde{x}_n = 0$, remains high. During the second phase, that is, from iteration 33 to iteration 116, the infeasibility measure is very unpredictable till the stopping criterion is fulfilled ($\|x_{116} - \widetilde{x}_{116}\|_\infty < \epsilon$).

The linearized and non-linearized generation, concepts introduced in subsection 7.2.2, are represented in Figure 7.17. Once again, the linearized and non-linearized generations are very close.

7.4.3 Solution description

The most important optimization results are displayed in Table 7.10. With 1.75 hours MACH gives a load primal feasible solution, which should be good enough for practical purposes, since the duality gap is 1.76% and the primal feasibility measure $\|x^* - \widetilde{x}^*\|_\infty = 0.86$ MW. Note that, for numerical reasons, we solve a scaled version of the current GUC instance, so as the scaled infeasibility at the last iterate is $\|x^* - \widetilde{x}^*\|_\infty = 0.0086$, which is lower than the stopping threshold ($\epsilon = 10^{-2}$). However, we report the true value for $\|x^* - \widetilde{x}^*\|_\infty$, which is 0.86 MW.

Once this GUC instance is solved, three facets of the load production policy have been optimized: the unit commitment, the power dispatching and the power flow through the transmission network. It is important to point out that the load primal feasible solution given by MACH is optimal and compatible for the three facets because they are taken into account simultaneously over the whole optimization process. The optimal unit commitment is in Figure 7.18 and the optimal hydrothermal dispatching

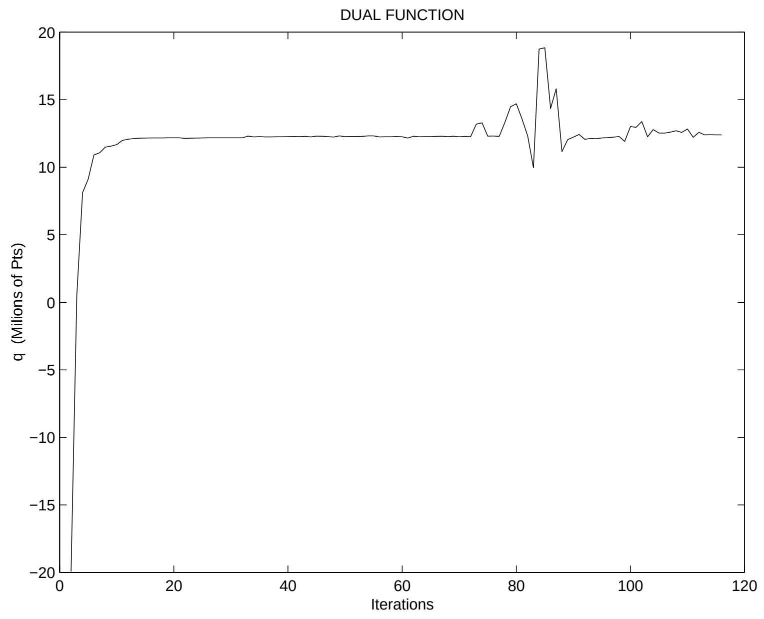

Figure 7.15: Evolution of the dual function.

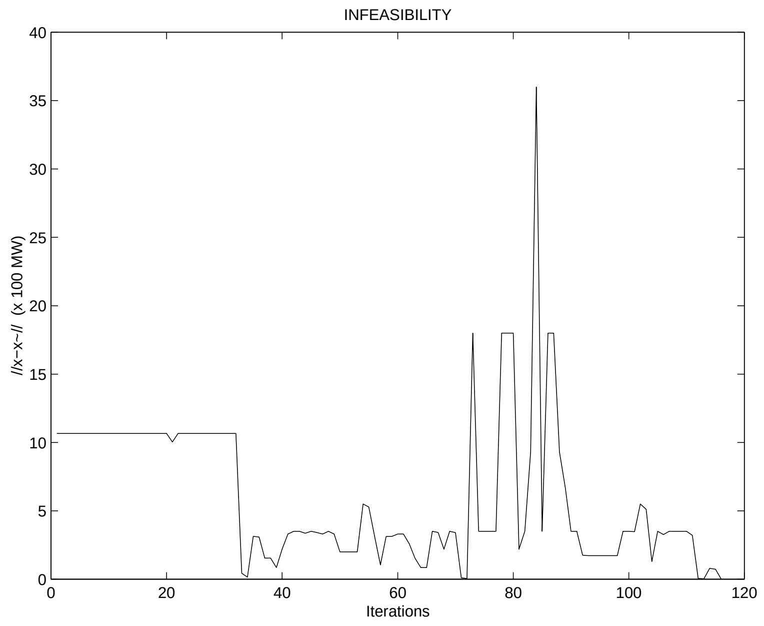

Figure 7.16: Evolution of the infeasibility.

Table 7.10: Optimization results.

Iterations	Time (hours)	Dual bound	Opt. cost ($\times 10^6$ PTA)	Duality gap (%)	$\|x^* - \widetilde{x}^*\|_\infty$ (MW)
116	1.75	12.183	12.397	1.76	0.86

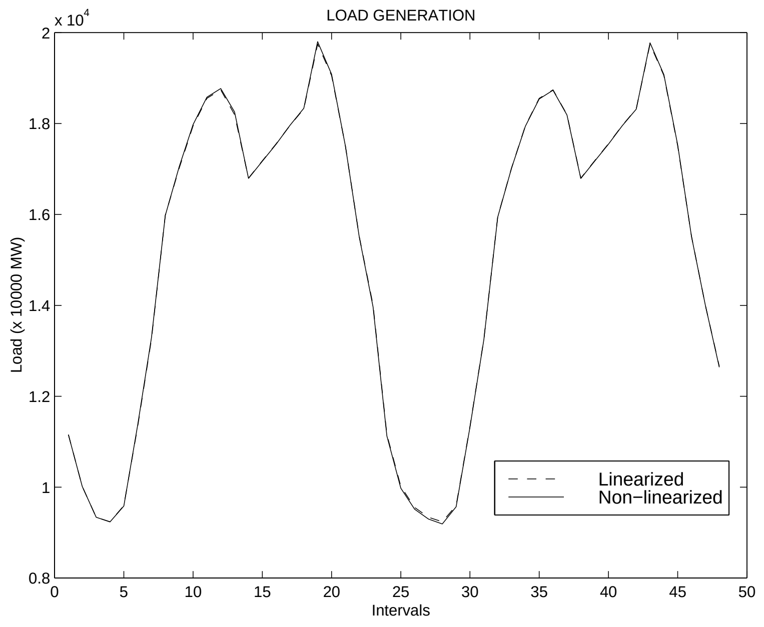

Figure 7.17: Linearized and non-linearized generation are very close.

in Figure 7.19. Regarding the Optimal Power Flow (OPF), none of the lines become saturated and, as an example, the OPF for line N04 $\rightarrow$ N06 is in Figure 7.20. The numerical description of the OPF solution is in the file pub/onl/reports/dwh1x481.sol and can be obtained via anonymous ftp at ftp-eio.upc.es.

7.5 Summary

In previous chapters several strategies and methodologies have been theoretically studied and practically tested to develop the radar multiplier method. Based on this method, the MACH code has been designed, implemented and tested in the current chapter.

Two MACH tests have been carried out. In the first one, MACH has successfully solved a real-life large-scale unit commitment instance composed of 70 thermal units and 20 reservoirs over a 48-hour period. In the second one, a set of 8 Generalized Unit Commitment (GUC) instances, ranging from small to large-scale size, have been satisfactorily solved. In the last section, the largest solved GUC case (70 thermal units, 10 reservoirs, and a 48-hour period) is analyzed in full detail. The inclusion of the transmission network has not disrupted the convergence properties of MACH, except for slightly slowing down the whole process.

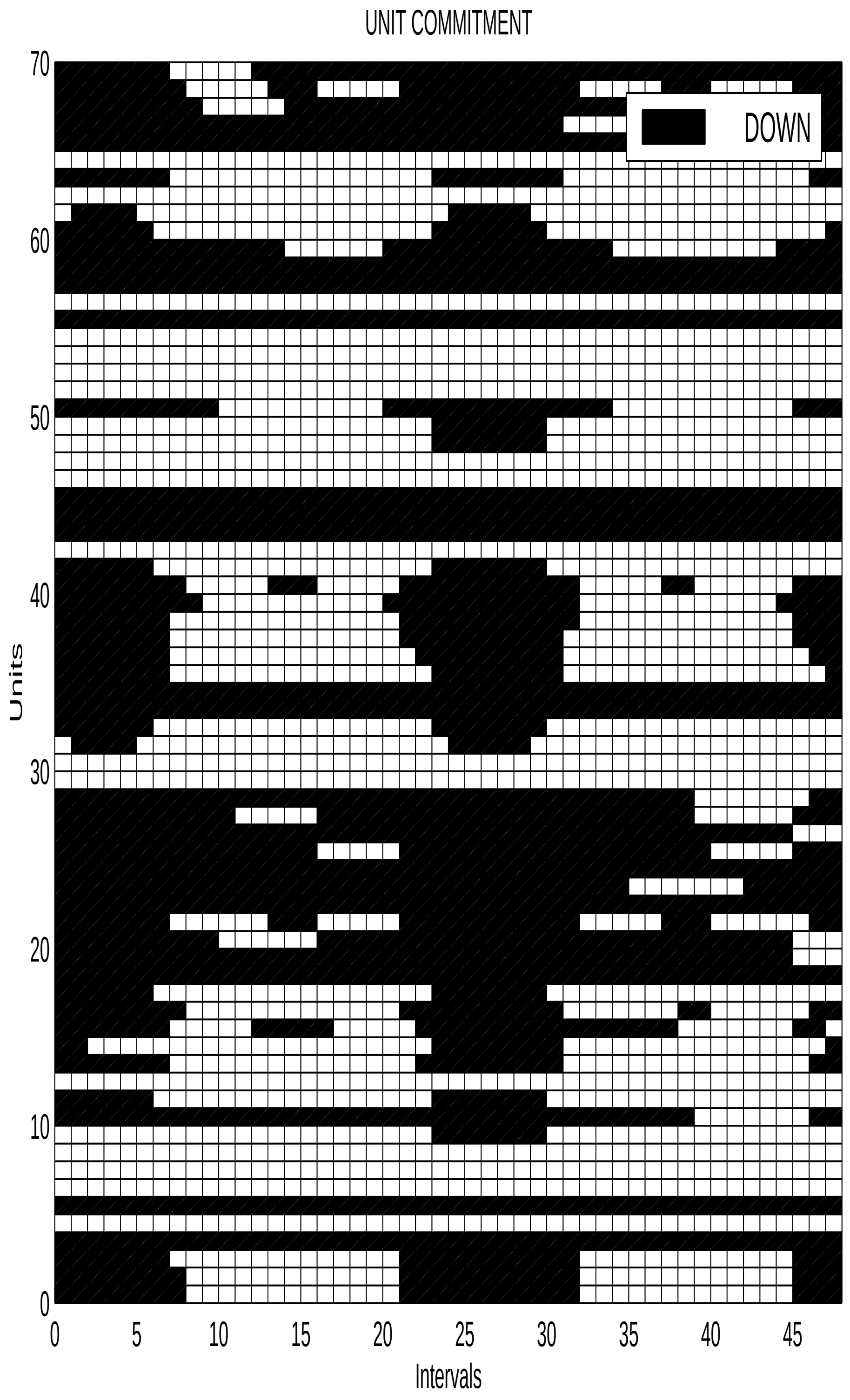

Figure 7.18: Optimal unit commitment pattern.

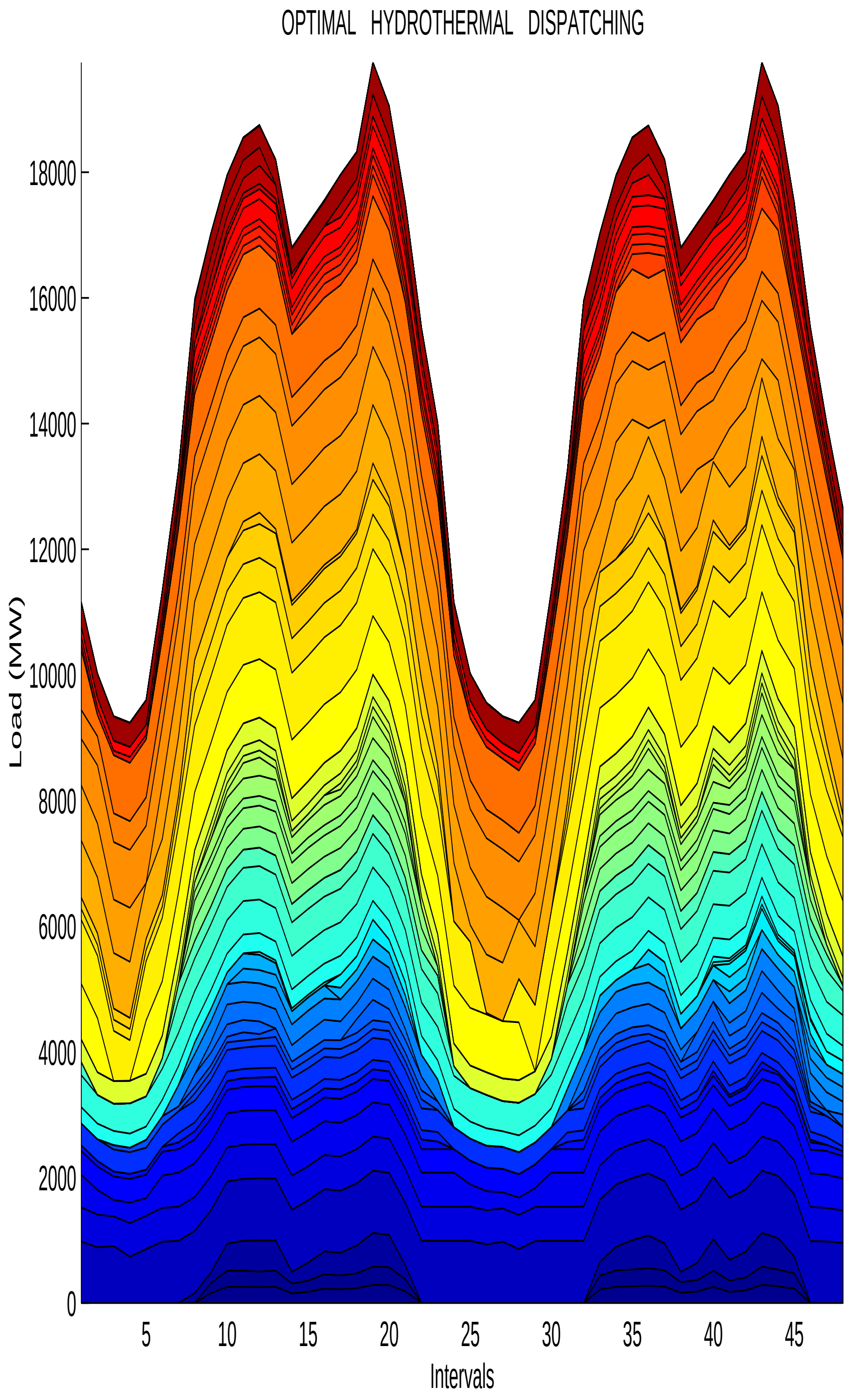

Figure 7.19: Economic dispatching.

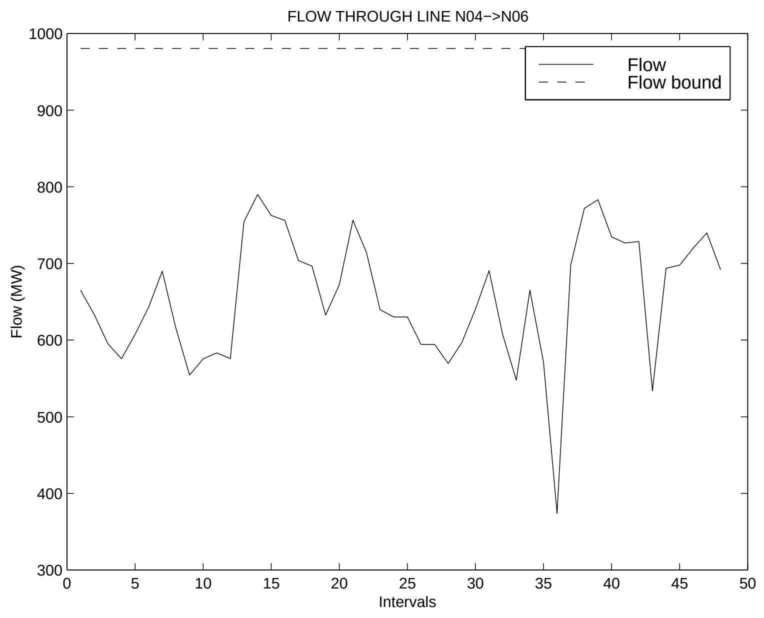

Figure 7.20: Optimal power flow through line N04 $\rightarrow$ N06.

Chapter 8

Primal feasibility

This chapter is based on the following concepts defined in section (3.5): Full Primal Feasible (FPF) solution, Load Primal Feasible (LPF) solution, Incremental spinning reserve Primal Feasible (IPF) solution and Decremental spinning reserve Primal Feasible (DPF) solution. The Generalized Unit Commitment (GUC) solutions obtained so far are LPF but not FPF, that is, they fulfill the load demand constraints but they do not necessarily fulfill the spinning reserve constraints. In this chapter we propose to use the variable duplication method to obtain FPF optimizers. The procedure is tested on a simple version of the Unit Commitment (UC) problem.

8.1 Primal feasibility using variable duplication

In order to obtain FPF solutions by using the variable duplication method, the first step is to change the Incremental Spinning (IS) reserve definition (3.50) into this new IS reserve definition

$$r_{It}^{\,i}(p_t^i) := \begin{cases} \min\{\bar{r}_{It}, \bar{p}_t^i - p_t^i\} \text{ if } p_t^i > 0, \\ \\ 0 \text{ if } p_t^i = 0, \end{cases} \tag{8.1}$$

which also considers the off state of the thermal unit. Analogously, the Decremental Spinning (DS) reserve is redefined as

$$r_{Dt}^{\,i}(p_t^i) := \begin{cases} \min\{\bar{r}_{Dt}, p_t^i - \underline{p}_t^i\} \text{ if } p_t^i > 0, \\ \\ 0 \text{ if } p_t^i = 0. \end{cases} \tag{8.2}$$

The second step is to incorporate

$$r_I(p) := (r_{It}^{\,i}(p_t^i))_{t=1,i=1}^{n_t,n_i} \quad \text{and} \quad r_D(p) := (r_{Dt}^{\,i}(p_t^i))_{t=1,i=1}^{n_t,n_i}$$

into our formulation of the UC problem.

$$\min \qquad C_{ther}(p) + C_m(p) \tag{8.3}$$

$$\begin{aligned} s.t. \quad & \sum_{t=1}^{n_t} p_t^i = L^i, & (8.4)\\ & \sum_{t=1}^{n_t} {r_I}_t^i(p_t^i) \geq {R_I}^i, & (8.5)\\ & \sum_{t=1}^{n_t} {r_D}_t^i(p_t^i) \geq {R_D}^i, & (8.6)\\ & i = 1, \dots, n_i, & \\ & (p, r_I(p), r_D(p)) \in D_{ther}, & (8.7)\\ & (p, r_I(p), r_D(p)) \in D_m, & (8.8) \end{aligned}$$

where we have the thermal cost $C_{ther}(p)$, the management cost $C_m(p)$, the load demand L^i, the IS reserve requirement ${R_I}^i$, the DS reserve requirement ${R_D}^i$, the thermal domain D_{ther} and the management domain D_m (see chapter 2 for more details). Note that the hydrothermal distribution cost function C_{htd} of chapter 2, here is replaced by the thermal cost function C_{ther}, since neither the hydraulic generating system nor the transmission network are considered. These systems could also be handled by the methodology we are introducing, but they are not considered for the sake of explanatory simplicity.

From now on, notation will be simplified by writing $\tilde{r}_I$ and $\tilde{r}_D$ instead of $\tilde{r}_I(\tilde{p})$ and $\tilde{r}_D(\tilde{p})$ respectively (analogously with r_I and r_D). Thermal domain D_{ther} accounts for the lower and upper bounds of vectors p, r_I and r_D. That is

$$D_{ther} := \prod_{i=1}^{n_i} (D_{ther}^i \times D_{ther}^{I^i} \times D_{ther}^{D^i}),$$

where

$$\begin{aligned} D_{ther}^i &:= \{x \in R^{n_t} : x_t \in [0, \bar{p}_t], \quad t = 1, \dots, n_t\},\\ D_{ther}^{I^i} &:= \{x \in R^{n_t} : x_t \in [0, \bar{r}_{It}], \quad t = 1, \dots, n_t\},\\ D_{ther}^{D^i} &:= \{x \in R^{n_t} : x_t \in [0, \bar{r}_{Dt}], \quad t = 1, \dots, n_t\}. \end{aligned}$$

Note that in D_{ther} we have relaxed definitions (8.1–8.2) of ${r_I}_t^i$ and ${r_D}_t^i$ as discontinuous functions. These definitions are taken into account in the management domain D_m.

One way to solve problem (8.3–8.8) is to duplicate vectors p, r_I and r_D into $\tilde{p}$, $\tilde{r}_I$ and $\tilde{r}_D$, respectively, followed by an Augmented Lagrangian Relaxation (ALR).

$$\begin{aligned} \min \quad & C_{ther}(p) + C_m(\tilde{p}) & (8.9)\\ s.t. \quad & \sum_{t=1}^{n_t} p_t^i = L^i, & (8.10)\\ & \sum_{t=1}^{n_t} {r_I}_t^i \geq {R_I}^i, & (8.11)\\ & \sum_{t=1}^{n_t} {r_D}_t^i \geq {R_D}^i, & (8.12)\\ & i = 1, \dots, n_i, & (8.13)\\ & (p, r_I, r_D) \in D_{ther}, \quad (\tilde{p}, \tilde{r}_I, \tilde{r}_D) \in D_m, & (8.14)\\ & p - \tilde{p} = 0, \quad r_I - \tilde{r}_I = 0, \quad r_D - \tilde{r}_D = 0. & (8.15) \end{aligned}$$

Then, the ALR formulation to be solved by the Radar Multiplier (RM) method is

$$\left.\begin{array}{ll}\min & C_{ther}(p) + C_m(\tilde{p}) + \lambda'(p-\tilde{p}) + \frac{c}{2}\|p-\tilde{p}\|^2 \\ & +\lambda_I'(r_I - \tilde{r}_I) + \frac{c_I}{2}\|r_I - \tilde{r}_I\|^2 \\ & +\lambda_D'(r_D - \tilde{r}_D) + \frac{c_D}{2}\|r_D - \tilde{r}_D\|^2 \\ & \\ s.t. & (8.10\text{–}\ 8.14). \end{array}\right\} \qquad (8.16)$$

8.1.1 Problem to be solved in phase 1

We assume that

$$D_m := D_m^1 \times D_m^2 \times \ldots \times D_m^{n_t},$$

$$p_t := (p_t^1, p_t^2, \ldots, p_t^{n_i})',$$

$$p^i := (p_1^i, p_2^i, \ldots, p_{n_t}^i)'$$

and analogously for $\tilde{p}, \lambda, \lambda_I, \lambda_D, \tilde{r}_D$ and $\tilde{r}_I$. Furthermore we suppose

$$C_{ther}(p) = \sum_{i=1}^{n_i} C_{ther}^i(p^i)$$

and

$$C_m(p) = \sum_{t=1}^{n_t} C_{m,t}(p_t).$$

Given that in phase 1 of the RM method the penalty parameters c, c_I and c_D are set equal to 0, then the problem to be solved at each iteration n of phase 1 is

$$\begin{array}{ll}\min & C_{ther}(p) + C_m(\tilde{p}) + \lambda_n'(p-\tilde{p}) + \lambda_{I,n}'(r_I - \tilde{r}_I) + \lambda_{D,n}'(r_D - \tilde{r}_D) \\ s.t. & (8.10\text{–}\ 8.14), \end{array} \qquad (8.17)$$

which decomposes into the following four subproblems (here $(p(n+1), r_I(n+1), r_D(n+1), (\tilde{p}', \tilde{r}_I', \tilde{r}_D')(n+1))$ stands for the new iterate).

$$\left.\begin{array}{rl} p(n+1) := \arg\min & \sum_{i=1}^{n_i}\{C_{ther}^i(p^i) + (\lambda_n^i)'p^i\} \\ s.t. & \sum_{t=1}^{n_t} p_t^i = L^i, \\ & p^i \in D_{ther}^i, \\ & i = 1, \ldots, n_i. \end{array}\right\} \qquad (8.18)$$

$$\left.\begin{array}{rl} r_I(n+1) := \arg\min & \sum_{i=1}^{n_i}(\lambda_{I,n}^i)'{r_I}^i \\ s.t. & \sum_{t=1}^{n_t} {r_{It}}^i \geq {R_I}^i, \\ & {r_I}^i \in D_{ther}^{Ii}, \\ & i = 1, \ldots, n_i. \end{array}\right\} \qquad (8.19)$$

$$\left.\begin{array}{rl} r_D(n+1) := \arg\min & \sum_{i=1}^{n_i} (\lambda^i_{D,n})' r_D{}^i \\ s.t. & \sum_{t=1}^{n_t} r_{Dt}{}^i \geq R_D{}^i, \\ & r_D{}^i \in D^{D^i}_{ther}, \\ & i = 1, \ldots, n_i. \end{array}\right\} \qquad (8.20)$$

$$\left.\begin{array}{rl} (\tilde{p}', \tilde{r}'_I, \tilde{r}'_D)(n+1) := \arg\min & \sum_{t=1}^{n_t} \{C_{m,t}(\tilde{p}_t) - \lambda'_{t,n}\tilde{p}_t \\ & -\lambda'_{I,n_t}\tilde{r}_{It} - \lambda'_{D,n_t}\tilde{r}_{Dt}\} \\ s.t. & (\tilde{p}_t, \tilde{r}_{It}, \tilde{r}_{Dt}) \in D_{m,t}, \\ & t = 1, \ldots, n_t. \end{array}\right\} \qquad (8.21)$$

Subproblem (8.18) can be decomposed into n_i quadratic economic dispatching subsubproblems or, if we consider the transmission system, n_i optimal power flow subsubproblems. These subsubproblems, a part from being linearly constrained, will have nonlinear constraints if the hydraulic system is considered. Analogously, subproblems (8.19) and (8.20) can be decomposed into n_i linear subsubproblems with bounded variables (recall the definition of $D^{I^i}_{ther}$ and $D^{D^i}_{ther}$). Subproblem (8.21) decomposes into n_t subsubproblems that can be solved by means of dynamic programming (see chapter 3).

Regarding the multiplier updating, we can use the Radar Subgradient (RS) method described in chapter 5 considering that we still have an unconstrained dual function to be maximized. Now, the updated formula to compute the step length $\beta_{n,k}$ is

$$\beta_{n,k} := \frac{q_k - q_n + (\hat{\lambda}_n - \hat{\lambda}_k)' s_k}{(s_n - s_k)' s_n}, \quad k = 1, \ldots, n,$$

with

$$\hat{\lambda}'_k := (\lambda'_k, \lambda'_{Ik}, \lambda'_{Dk})$$

and

$$s'_k := (p'_k - \tilde{p}'_k, r_{Ik}' - \tilde{r}'_{Ik}, r_{Dk}' - \tilde{r}'_{Dk}).$$

8.1.2 Problem to be solved in phase 2

The problem to be solved at each iteration n of phase 2 of the Radar Multiplier (RM) method is

$$\left.\begin{array}{rl} \min & C_{ther}(p) + C_m(\tilde{p}) + \lambda'_n(p - \tilde{p}) + \frac{c}{2}\|p - \tilde{p}\|^2 \\ & +\lambda'_{I,n}(r_I - \tilde{r}_I) + \frac{c_I}{2}\|r_I - \tilde{r}_I\|^2 \\ & +\lambda'_{D,n}(r_D - \tilde{r}_D) + \frac{c_D}{2}\|r_D - \tilde{r}_D\|^2 \\ & \\ s.t. & (8.10\text{–} 8.14). \end{array}\right\} \qquad (8.22)$$

which, using the Block Coordinate Descent (BCD) method (see chapter 4), decomposes into four subproblems

$$\left.\begin{aligned} p(n+1) := \arg\min\ & \sum_{i=1}^{n_i} \{C^i_{ther}(p^i) + (\lambda^i_n)' p^i + \tfrac{c}{2}\|p^i - \tilde{p}(n)^i\|^2\} \\ s.t.\ & \sum_{t=1}^{n_t} p^i_t = L^i, \\ & p^i \in D^i_{ther}, \\ & i = 1, \ldots, n_i. \end{aligned}\right\} \tag{8.23}$$

$$\left.\begin{aligned} r_I(n+1) := \arg\min\ & \sum_{i=1}^{n_i} \{(\lambda^i_{I,n})' {r_I}^i + \tfrac{c_I}{2}\|{r_I}^i - \tilde{r}_I(n)^i\|^2\} \\ s.t.\ & \sum_{t=1}^{n_t} {r_{It}}^i \geq {R_I}^i, \\ & {r_I}^i \in D^{Ii}_{ther}, \\ & i = 1, \ldots, n_i. \end{aligned}\right\} \tag{8.24}$$

$$\left.\begin{aligned} r_D(n+1) := \arg\min\ & \sum_{i=1}^{n_i} \{(\lambda^i_{D,n})' {r_D}^i + \tfrac{c_D}{2}\|{r_D}^i - \tilde{r}_D(n)^i\|^2\} \\ s.t.\ & \sum_{t=1}^{n_t} {r_{Dt}}^i \geq {R_D}^i, \\ & {r_D}^i \in D^{Di}_{ther}, \\ & i = 1, \ldots, n_i. \end{aligned}\right\} \tag{8.25}$$

$$\left.\begin{aligned} (\tilde{p}', \tilde{r}'_I, \tilde{r}'_D)(n+1) := \arg\min\ & \sum_{t=1}^{n_t} \{C_{m,t}(\tilde{p}_t) - \lambda'_{t,n}\tilde{p}_t + \tfrac{c}{2}\|p(n+1)_t - \tilde{p}_t\|^2 \\ & -\lambda'_{It,n} r_{It} + \tfrac{c_I}{2}\|r_I(n+1)_t - \tilde{r}_{It}\|^2 \\ & -\lambda'_{Dt,n} r_{Dt} + \tfrac{c_D}{2}\|r_D(n+1)_t - \tilde{r}_{Dt}\|^2 \\ s.t.\ & (\tilde{p}_t, \tilde{r}_{It}, \tilde{r}_{Dt}) \in D^t_m, \quad t = 1, \ldots, n_t. \end{aligned}\right\} \tag{8.26}$$

These four subproblems are similar to those solved in phase 1 except that now subproblems (8.24) and (8.25) have a quadratic objective function whereas in (8.19) and (8.20) the objective function is linear.

In this second phase of the RM method the multiplier updating is done by the multiplier method

$$\lambda_{n+1} = \lambda_n + c \cdot (p(n+1) - \tilde{p}(n+1)),$$

$$\lambda_{I,n+1} = \lambda_{I,n} + c_I \cdot (r_I(n+1) - \tilde{r}_I(n+1)),$$

$$\lambda_{D,n+1} = \lambda_{D,n} + c_D \cdot (r_D(n+1) - \tilde{r}_D(n+1)).$$

8.2 Computational experience

The objective of this section is to test the procedure designed in the above section to obtain Full Primal Feasible (FPF) optimizers. For this test, we enhance the simple Unit Commitment (UC) problem presented in chapter 3 by introducing an Incremental Spinning (IS) reserve constraint (n_t is the number of thermal units, x_i is the output of thermal unit i, r_i is the IS reserve of unit i, Con_i is the starting cost, i.e. the cost of turning a unit on, L is the demand for electrical power, R is the required IS reserve, l_i and u_i are lower and upper bounds for x_i, and $\bar{r}_i$ is the upper bound for r_i).

$$\left.\begin{array}{ll} \min & f(x,r) := \sum_{i=1}^{n_t}(2x_i^2 + Con_i(x_i)) \\ s.t. & \sum_{i=1}^{n_t} x_i = L, \\ & \sum_{i=1}^{n_t} r_i \geq R, \\ & x_i \in \{0\} \cup [l_i, u_i] \quad i = 1,\dots,n, \\ & r_i = \begin{cases} \min\{\bar{r}_i, u_i - x_i\} & \text{if} \quad x_i > 0, \\ 0 \quad \text{if} \quad x_i = 0, \end{cases} \end{array}\right\} \tag{8.27}$$

where arbitrarily we choose

$$Con_i(x_i) := \begin{cases} 10 + (i-1)\frac{10}{n-1} & \text{if} \quad x_i > 0, \\ \\ 0 \quad \text{if} \quad x_i = 0. \end{cases} \tag{8.28}$$

The 3 subproblems to be solved at each iteration n of phase 1 in this case are

$$\left.\begin{array}{rl} x(n+1) := \arg\min & \sum_{i=1}^{n_t} x_i^2 + \lambda_n' x \\ s.t. & \sum_{i=1}^{n_t} x_i = L, \\ & x_i \in [0, u_i] \quad i = 1,\dots,n. \end{array}\right\} \tag{8.29}$$

$$\left.\begin{array}{rl} r(n+1) := \arg\min & \lambda_{I,n}' r \\ s.t. & \sum_{i=1}^{n_t} r_i \geq R, \\ & r_i \in [0, \bar{r}_i] \quad i = 1,\dots,n. \end{array}\right\} \tag{8.30}$$

$$\left.\begin{array}{rl} (\tilde{x}', \tilde{r}')(n+1) := & \arg\min \sum_{i=1}^{n_t}\{\tilde{x}_i^2 + Con_i(\tilde{x}_i) - \lambda_{i,n}\tilde{x}_i - \lambda_{Ii,n}\tilde{r}_i\} \\ & s.t. \quad \tilde{r}_i = \begin{cases} \min\{\bar{r}_i, u_i - \tilde{x}_i\} & \text{if} \quad \tilde{x}_i > 0, \\ 0 \quad \text{if} \quad \tilde{x}_i = 0, \end{cases} \\ & \tilde{x}_i \in \{0\} \cup [l_i, u_i] \quad i = 1,\dots,n. \end{array}\right\} \tag{8.31}$$

On the other hand, at each iteration n of phase 2 we have to solve the 3 following subproblems. Note that the definition of $\tilde{r}_i$ is different in problems (8.31) and (8.34) (see remark 2 in section 8.2.3 for more details).

$$\left.\begin{array}{rl} x(n+1) := \arg\min & \sum_{i=1}^{n_t} x_i^2 + \lambda_n' x + \frac{c_n}{2}\|x - \tilde{x}(n)\|^2 \\ s.t. & \sum_{i=1}^{n_t} x_i = L, \\ & x_i \in [0, u_i] \quad i = 1,\dots,n. \end{array}\right\} \tag{8.32}$$

Table 8.1: Instance description and tuning parameters.

Case	Units	R	$\bar{r}$	α_0	β	β_I
1	20	2	0.240	10	1.5	1.5
2	20	2	0.180	10	1.5	1.5
3	40	4	0.240	10	1.5	1.5
4	40	4	0.172	10	1.5	1.5
5	60	6	0.240	10	1.1	1.5
6	60	6	0.180	10	1.5	2.0
7	80	8	0.240	5	1.5	2.0
8	80	8	0.160	5	1.1	1.2
9	100	10	0.250	5	1.1	1.1
10	100	10	0.180	5	1.1	1.2

$$\left.\begin{array}{rl} r(n+1) := \arg\min & \lambda'_{I,n} r + \frac{c_{In}}{2}\|r - \tilde{r}(n)\|^2, \\ s.t. & \sum_{i=1}^{n_t} r_i \geq R, \\ & r_i \in [0, \bar{r}_i] \quad i = 1, \ldots, n. \end{array}\right\} \tag{8.33}$$

$$\left.\begin{array}{rl} (\tilde{x}', \tilde{r}')(n+1) := & \arg\min \sum_{i=1}^{n_t} \{\tilde{x}_i^2 + Con_i(\tilde{x}_i) \\ & -\lambda_{i,n}\tilde{x}_i + \frac{c_n}{2}(x(n+1)_i - \tilde{x}_i)^2 \\ & -\lambda_{Ii,n}\tilde{r}_i + \frac{c_{In}}{2}(r(n+1)_i - \tilde{r}_i)^2\} \\ & \\ s.t. \quad \tilde{r}_i \in & \left\{\begin{array}{ll} [0, \min\{\bar{r}_i, u_i - \tilde{x}_i\}] & \text{if} \quad \tilde{x}_i > 0, \\ \{0\} & \text{if} \quad \tilde{x}_i = 0, \end{array}\right. \\ & \tilde{x}_i \in \{0\} \cup [l_i, u_i] \quad i = 1, \ldots, n. \end{array}\right\} \tag{8.34}$$

8.2.1 Test results

As we have already seen in chapter 4, given that the starting cost function Con_i is monotone on i and that the generating cost is the same for all units ($2x_i^2 \;\; i = 1, \ldots, n$), this problem can be solved exactly (algebraic method).

In order to test the ability of the Radar Multiplier (RM) method to obtain Full Primal Feasible (FPF) optimizers we use a set of 10 instances of the above simple UC problem with Incremental Spinning (IS) reserve. The parameters that describe the 10 instances are: n, R and $\bar{r}$ displayed in Table 8.1 (note that for a given instance, $\bar{r}$ is the same for all units); L has been set equal to the number of thermal units; $l_i = 1$; $u_i = L$.

In the first phase of the RM method we have used the following stopping condition ($\hat{\lambda}'_n := (\lambda'_n, \lambda'_{In})$)

$$\frac{\sum_{i=0}^{2} \|\hat{\lambda}_{n-i} - \hat{\lambda}_{n-i-1}\|_\infty}{3} < \epsilon_{\hat{\lambda}}$$

with $\epsilon_{\hat{\lambda}} = 10^{-4}$. Furthermore, we take $\hat{\lambda}_0 = 0$ and the subgradient step length sequence $\{\alpha_n\}$ is defined as $\alpha_n = \frac{\alpha_0}{n}$ $(n > 0)$ with α_0 in Table 8.1.

The second phase of the RM method is stopped if $\|x_n - \tilde{x}_n\|_\infty < \epsilon$ and $\|r_n - \tilde{r}_n\|_\infty < \epsilon$ with $\epsilon = 10^{-4}$. The initial penalty parameters are $c_0 = 0.1$ and $c_{I0} = 2.1$ for the 10 cases. In the second phase of the RM method, the updating procedure for c_n depends on α and β (see chapter 6). Analogously c_{In} updating depends on α_I and β_I. We have set $\alpha = \alpha_I = 1.10$, whereas β and β_I are in Table 8.1. More details about the penalty parameters can be found in remark 1 of section (8.2.3).

We will compare the global optimum obtained by using the algebraic method (colum ALG) with the results obtained with the RM method. In Table 8.2 we can see the number of iterations and the CPU time for the RM method (phase 1, phase 2). Note that, on average, the CPU time of phase 2 is over 3 times the CPU time of pase 1. This is due to the fact that in phase 1 subproblem (8.30) is linear whereas in phase 2 subproblem (8.33) is quadratic. Furthermore, subproblem (8.31) decomposes into n simple subsubproblems of dimension 1 given that $\tilde{r}_i$ is a function of $\tilde{x}_i$. However, subproblem (8.34) decomposes into n quadratic subsubproblems of dimension 2 since now $\tilde{r}_i$ is a free variable. Note that the CPU time is high compared to the dimension of the instances. This is due to fact that the instances have been solved using Matlab, which is an interpreter language.

In our opinion, Table 8.3 gives the most important result of this test: for the 10 instances of the simple UC problem with IS reserve the RM method has been able to compute a global FPF optimizer or a high quality local FPF optimizer. The relative error $100(f^*_{RM} - f^*_{ALG})/f^*_{ALG})$ always remains under 1%, on average is 0.17 % and in 6 over 10 cases the RM has obtained a global FPF optimizer. Note that we have obtained full primal feasible solutions since the x infeasibility, $\|x - \tilde{x}\|_\infty$, and the r infeasibility, $\|r - \tilde{r}\|_\infty$, are always equal or less than 10^{-4}. Even more, the r infeasibility is 0 but for case 10.

The optimal number of on units is in Table 8.3 as well. Sometimes the optimal number of on units is the same for the ALG and RM methods but the optimum is different. In this case the two methods have chosen different set of on units.

8.2.2 Focusing on one case

Let us focus on case 4 of the previous test to show the typical behavior of the RM method. Comparing the evolution of the penalty parameters c_n and c_{In} in Figures 8.1 and 8.2, so far, we do not have a clear explanation for the difference between the final value of $c_n \approx 6$ and $c_{In} \approx 1700$. We also observe that, as we already know, the two penalty parameters are 0 for the first 59 iterations (phase 1) and then they are increased as needed during phase 2.

The evolution of the dual function during the optimization process is in Figure 8.3. The value of the dual function becomes stabilized around 344 in phase 1 (iterations 20 to 59), then, along phase 2 (iterations 60 to 115), the RM method obtains higher values of the augmented dual function till reaching the global optimum 444.1. In

Table 8.2: CPU time using the RM method.

Case	Iterations			CPU time (seconds)		
	Phase 1	Phase 2	Total	Phase 1	Phase 2	Total
1	56	75	131	13	38	51
2	69	135	204	16	66	82
3	57	73	130	47	110	157
4	59	56	115	46	82	128
5	67	39	106	144	136	280
6	96	101	197	185	312	497
7	62	33	95	247	221	468
8	97	103	200	383	614	997
9	72	223	295	489	2785	3274
10	71	148	219	450	1844	2294
Average	70.6	98.6	169.2	202	620.8	822.8

Figures 8.4 and 8.5 we have the x and r infeasibility, respectively. In the first phase the infeasibility measure for x, $\|x-\tilde{x}\|_\infty$, basically oscillates around 2.7. In the second phase, by augmenting the Lagrangian, the RM method manages to obtain a LPF solution with $\|x-\tilde{x}\|_\infty < 10^{-4}$. The same could be said about the r infeasibility, only that now $\|r-\tilde{r}\|_\infty \approx 0.172$ for almost the whole phase 1. Being the two infeasibilities below 10^{-4} we can say that we have obtained an FPF optimizer, which is the aim of this chapter.

8.2.3 Tuning and formulation remarks

From our experience, phase 1 of the RM method needs little tuning. However in phase 2 some tuning and formulation aspects have been crucial to attain convergence to Full Primal Feasible (FPF) optimizers.

Remark 1: As usual, the penalty parameter for the output power has been updated using the formula $c_{n+1} = \beta c_n$ with $c_0 = 0.1$ for all the cases in section 8.2.1 and with β displayed in Table 8.1. However, the penalty parameter for the IS reserve has been updated using $c_{_I n+1} = K + \hat{c}_{_I n+1}$ with $\hat{c}_{_I n+1} = \beta_{_I} \hat{c}_{_I n}$ ($\hat{c}_{_I 0} = 0.1$ for all the cases and $\beta_{_I}$ displayed in Table 8.1). With $K = 0$ the RM method sometimes converged to a Load Primal Feasible (LPF) optimizer, which was not an Incremental spinning reserve Primal Feasible (IPF) solution. To understand what happens, we study the quadratic term of the objective function in problem (8.32)

$$\sum_{i=1}^{n_t} \frac{2+c_n}{2} x_i^2 \tag{8.35}$$

Table 8.3: Optimum quality using the RM method.

Case	Optimum				Number of on units		Infeasibility ($\times 10^{-5}$)	
	Dual	RM	ALG	Error (%)	RM	ALG	x	r
1	171.2	197.8	197.8	0.00	9	9	6	0
2	172.4	221.4	221.4	0.00	12	12	9	0
3	343.1	393.1	393.1	0.00	17	17	7	0
4	344.5	444.1	444.1	0.00	24	24	9	0
5	514.7	588.8	588.8	0.00	25	25	6	0
6	517.0	648.0	646.9	0.17	34	34	10	0
7	686.6	787.5	787.5	0.00	34	34	6	0
8	637.5	915.5	911.1	0.48	50	50	7	0
9	794.1	987.8	978.8	0.92	43	40	9	0
10	861.0	1073.9	1072.7	0.11	56	56	0	4
Average	504.2	625.8	624.2	0.17	30.4	30.1	6.9	0.4

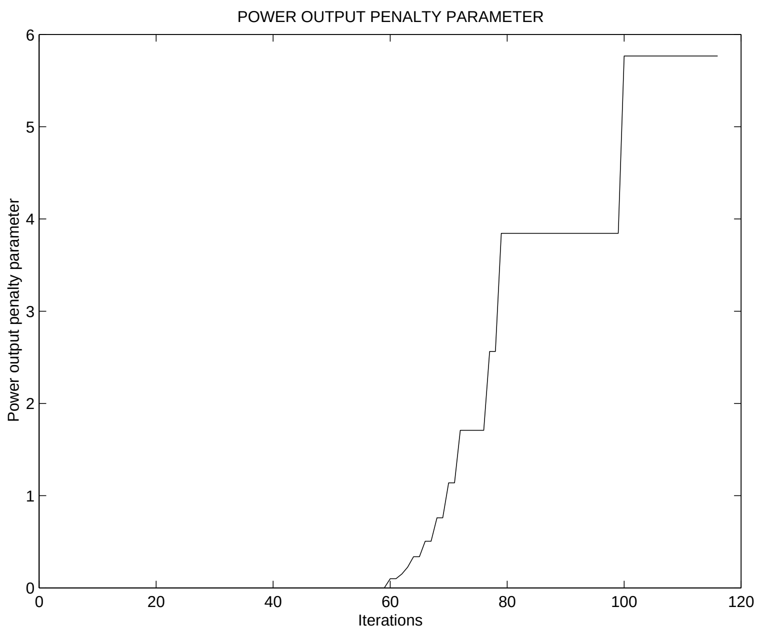

Figure 8.1: Evolution of the penalty parameter c_n.

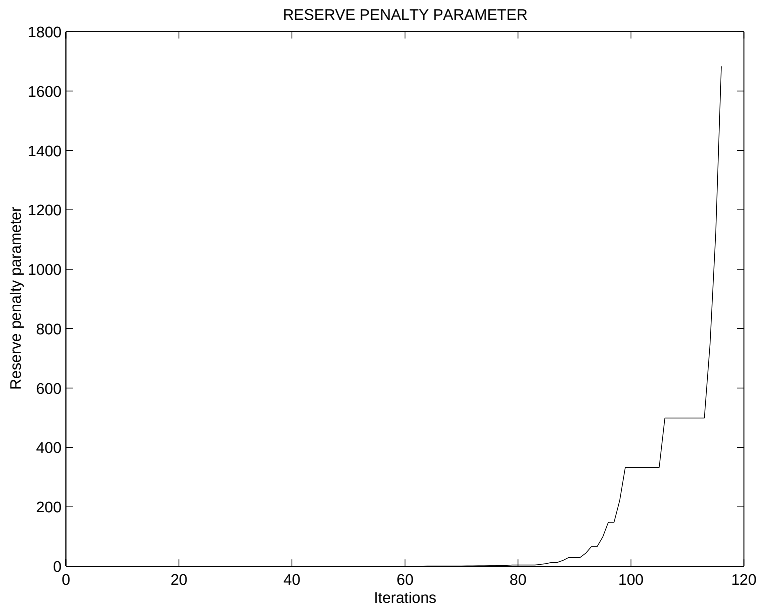

Figure 8.2: Evolution of the penalty parameter c_{In}.

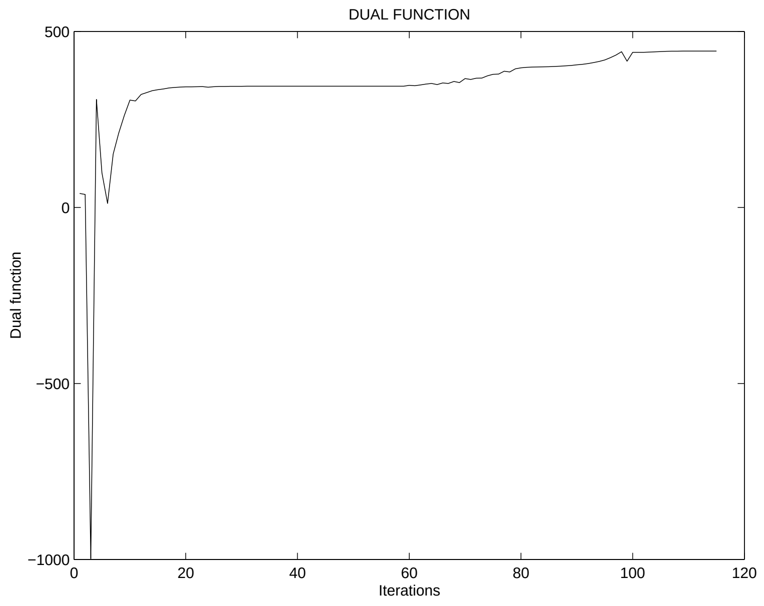

Figure 8.3: Evolution of the dual function.

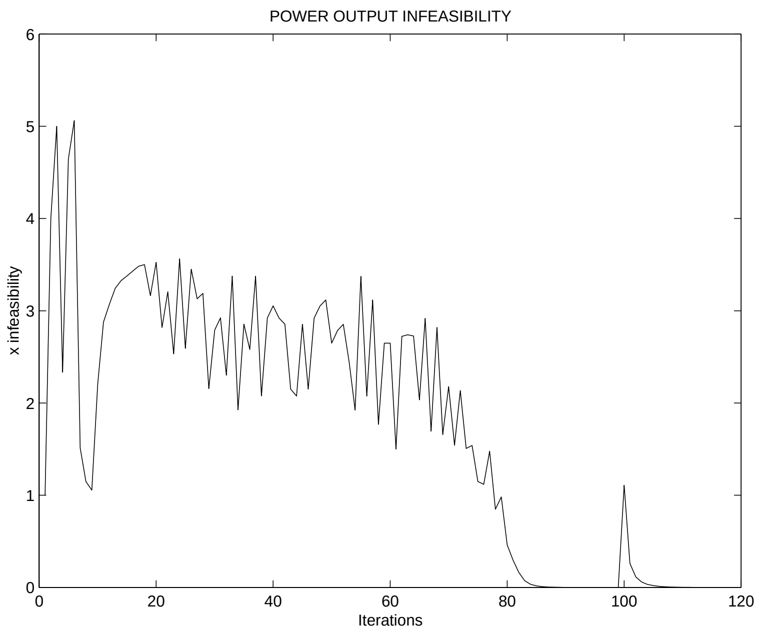

Figure 8.4: Evolution of the power output infeasibility.

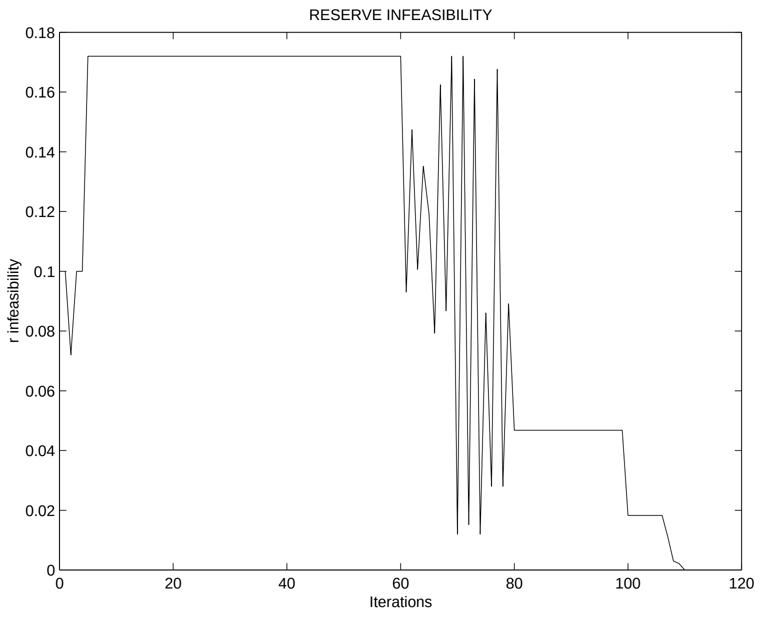

Figure 8.5: Evolution of the reserve infeasibility.

and in problem (8.33)

$$\sum_{i=1}^{n_t} \frac{K + \hat{c}_{In}}{2} r_i^2. \tag{8.36}$$

We observe that for $K = 0$, if we start with small penalty parameters c_n and $\hat{c}_{In}$, the term (8.36) is small relative to the term (8.36). As a result, initially, problem (8.33) has little influence on determining the path followed by the RM sequence of iterates. Thus, the iterates may be trapped in the neighborhood of a point, which is only LPF but not IPF, and eventually converge to it. Setting $K = 2$, we balanced problems (8.32-8.33) and FPF optimizers were obtained (see section 8.2.1).

Remark 2: Another point that has improved the performance of the RM method is the relaxation of the definition of the IS reserve. In subproblem (8.34) the definition of $\tilde{r}_i$

$$\tilde{r}_i = \begin{cases} \min\{\bar{r}_i, u_i - \tilde{x}_i\} & \text{if} \quad \tilde{x}_i > 0, \\ 0 \quad \text{if} \quad \tilde{x}_i = 0, \end{cases} \tag{8.37}$$

has been substituted by the following relaxation:

$$\tilde{r}_i \in \begin{cases} [0, \min\{\bar{r}_i, u_i - \tilde{x}_i\}] & \text{if} \quad \tilde{x}_i > 0, \\ \{0\} \quad \text{if} \quad \tilde{x}_i = 0. \end{cases} \tag{8.38}$$

Let us see that the global optimum of the UC problem (8.27) does not change either with (8.37) or with (8.38) as the $\tilde{r}_i$ definition. Suppose $(\tilde{x}^*, \tilde{r}^*)$ is a global optimizer of the UC problem with the relaxed IS reserve definition (8.38), then, obviously, $f(\tilde{x}^*, \tilde{r}^*) \leq f(\tilde{x}, \tilde{r})$ for each feasible point $(\tilde{x}, \tilde{r})$ of the not relaxed version of the UC problem. At the same time, we set $\tilde{r}_i^+ := \min\{\bar{r}_i, u_i - \tilde{x}_i^*\}$ for the on units $(\tilde{x}_i^* > 0)$ and $\tilde{r}_i^+ := 0$ for the off units $(\tilde{x}_i^* = 0)$. Then, $(\tilde{x}^*, \tilde{r}^+)$ is feasible for the not relaxed version of the UC problem since $(\tilde{x}^*, \tilde{r}^*)$ is a feasible point and $r_i^* \leq \tilde{r}_i^+ \quad (i = 1, \cdots, n_t)$ implies

$$R \leq \sum_{i=1}^{n_t} \tilde{r}_i^* \leq \sum_{i=1}^{n_t} \tilde{r}_i^+.$$

In addition $f(\tilde{x}^*, \tilde{r}^*) = f(\tilde{x}^*, \tilde{r}^+)$ given the definition of $f(x, r)$ in (8.27)

$$f(x, r) := \sum_{i=1}^{n_t} (2x_i^2 + Con_i(x_i)).$$

Thus, $(\tilde{x}^*, \tilde{r}^+)$ is a global optimizer of the not relaxed version of the UC problem. All in all, the global optimum of the UC problem is the scalar $f(\tilde{x}^*, \tilde{r}^*) = f(\tilde{x}^*, \tilde{r}^+)$ whether we take (8.37) or (8.38) to define $\tilde{r}_i$. Therefore the global optimum of the UC problem (8.27) does not depend on the definition of $\tilde{r}_i$ as we wanted to see.

Some of the cases presented in section 8.2.1 did not converge to a FPF solution until this relaxation was incorporated. Furthermore, this relaxation made easier the tuning parameter process. The price to be paid is a more CPU time demanding subproblem (8.34) as we already pointed out.

8.3 Summary

Our formulation of the Generalized Unit Commitment (GUC) problem has been slightly modified with the aim of obtaining Full Primal Feasible (FPF) optimizers. The main difference with the former formulation is that now the incremental and decremental spinning reserves are also duplicated to deal with both on and off states of a thermal unit. With the duplication of the spinning reserves, two more subproblems have to be solved at each iteration of the RM method: one for the Incremental Spinning (IS) reserve and the other for the Decremental Spinning (DS) reserve. The designed procedure has been tested on a simple Unit Commitment (UC) problem with IS reserve. Ten instances have been solved exactly (algebraic method) and by the RM method with a satisfactory result since the relative error of the RM optimizers have always been less than 1%. The price we have to pay to obtain FPF optimizers is a greater CPU time.

Chapter 9

Conclusions, contributions, and further research

With this thesis some aspects of the Generalized Unit commitment (GUC) problem and its solution methodology have been developed, and of course, others aspects have been left aside. First, the conclusions and contributions of the thesis will be enumerated, and then, matters left for further research will be listed.

9.1 Conclusions and contributions of the thesis

The aspects of this thesis are twofold. On the one hand there is the GUC modeling and on the other hand there is its solution methodology.

Regarding the GUC modeling we must point out that, using the Variable Duplication (VD) method, the coupled model developed by professors Heredia and Nabona [HN95] to solve the Optimal Power Flow (OPF) problem has been adapted to solve the GUC problem by, basically, imposing a lower bound to the generated power and redefining the objective function. This means that, if the VD method is to be used, virtually any OPF modeling could be used to model the GUC problem. A major consequence is that existing software developed to solve the OPF problem can be reused quite cheaply to solve the GUC problem. This has naturally been our case; MAPH, the code developed to solve a specific OPF formulation, proved to be a central routine in MACH, the code we developed to solve the GUC problem. This ability to reuse former OPF software is an advantage of the variable duplication method over other methods, as already pointed out in [BR92].

Therefore, because an almost suitable modeling for the GUC problem existed at the beginnings of this work, the emphasis was put mainly on the solution methodology. The starting point for the solution method was the notable work by professors Batut and Renaud [BR92]. In this paper the UC problem is solved using the Augmented Lagrangian Relaxation (ALR) method combined with the Variable Duplication (VD) method. Our contribution has improved Batut and Renaud's approach in four aspects:

a) To deal with the augmented Lagrangian, which is an inseparable function, we use the Block Coordinate Descent (BCD) method instead of the Auxiliary Problem Principle (APP) method. In chapter 3 it has been shown that the BCD method outperforms the APP method both from a practical and from a theoretical point of view.

b) The GUC problem is a highly non-convex problem because of its disconnected domain and non-convex objective function, and because of its constraint functions, which may also be non-convex. Therefore, using an augmented Lagrangian we obtain local optimizers. In this work a heuristic method designed to obtain high quality local optimizers has been proposed and successfully tested. In [BR92] this matter was not considered.

c) Of course, it is not good enough to compute good local optimizers; some measure of the optimizer quality is desirable. A natural measure of this quality is given by the dual optimum and thus, prior to solving a GUC problem by the ALR method, we solve the classical Lagrangian dual problem associated with the GUC problem. Batut and Renaud did not give any measure of the computed optimizer quality.

d) In [BR92] the VD method is used but no comparison with the Classical Lagrangian Relaxation (CLR) method is made. We do compare the two relaxation methods and establish that theoretically the VD method obtains dual bounds that are at least as good as those of the CLR method. In fact, we extend to the nonlinear case the analysis done for the linear case in [GK87].

Possibly our most original contribution is the Radar Subgradient (RS) method, a method designed to maximize an unconstrained non-smooth concave function, as for example the VD dual function. From a practical point of view this method outperforms the classical subgradient method by incorporating the first order information about the dual function given by the set of computed subgradients. However, a differentiable version of this method, the radar gradient method, performed more slowly than the multiplier method.

The final product of this research is the Radar Multiplier (RM) method. The RM method solves the VD modeling of the GUC problem in a two-phase framework. In the first phase, the VD dual problem (classical Lagrangian version) associated with the GUC problem is solved using the RS method. Analogously, in the second phase the VD dual problem (augmented Lagrangian version) associated with the GUC problem is solved using the multiplier method. We would like to stress that this simple idea of using the classical and the augmented Lagrangian relaxations cooperatively in a two-phase framework is very convenient for solving the GUC problem. Approaches that only use the classical Lagrangian relaxation find it difficult to obtain primal feasible solutions. Alternatively, approaches that only use the augmented Lagrangian relaxation lack a necessary quality measure of the computed local optimizer. We have implemented this two-phase approach in the MACH code, whose stability and strength has been shown by solving large-scale UC and GUC instances, some of them arising from real-life data. However, solutions computed using our VD formulation of the GUC problem are partially primal feasible since they do not necessarily fulfill the spinning reserve constraints. In chapter 8 we have shown that a slight modification of this GUC formulation makes possible to obtain Full Primal Feasible (FPF) optimizers with the RM method. This improvement of our methodology has been successfully tested with a set of simple UC problems.

9.2 Further research

Logically, further research can be done into two areas. The first area is concerned with the improvement of the GUC model and software issues. The model used does not take into account ramp constraints for the generated power; it would not be difficult to include them. Also, the direct current approach followed to model the transmission network could be replaced by a better alternating current approach. These and other pending aspects would make our model even more realistic. Nevertheless, the most important aspect to be considered in our next GUC model is the new liberalized power marked. New strategies must be sought to deal with stochastic prices and uncertain demand.

The effectiveness of the MACH package could be improved either by simplifying the hydraulic model or by using an improved Optimal Power Flow (OPF) solver. Effectively, the hydraulic generation function (a sixth degree polynomial) could possible be simplified without harming the accuracy of the model, in order to speed up the cumbersome hydraulic calculations. Regarding software issues, the OPF solver is based on NOXCB, a general purpose nonlinear network solver, which could be replaced by a more specific quadratic programming network solver, given that the underlying OPF problem is quadratic. Furthermore, in MAPH (the current OPF solver) nonlinear constraints are successively linearized. To improve this aspect new versions of MAPH, based on either interior point or projected Lagrangian methods, are being developed in our research group.

As pointed out in the above section, MACH solutions are only partially primal feasible. The VD based methodology, designed to obtain FPF optimizers and tested on a set of simple UC problems in chapter 8, must still be incorporated in the MACH code. Although, so far, results obtained with this methodology are promising, we have not tested it yet to solve large-scale GUC instances.

The second area for further research is concerned with more general optimization aspects:

a) The first of them is the theoretical study of the penalty parameter updating within the Augmented Lagrangian Relaxation method applied to non-convex problems, since currently a heuristic updating method is used.

b) The second research point is the Radar Subgradient (RS) method, which has been developed and tested from a practical point of view but whose convergence properties have not yet been studied. Furthermore, the RS method has been compared with the basic subgradient method as a first step. Hence, the second step should compare the RS method with more sophisticated dual methods such as the bundle family.

We still have to study the extension of the RS method to solve constrained dual problems since we have applied the RS method to the unconstrained case.

c) The third research point is the Variable Duplication (VD) method. We have proved that, the dual bounds obtained with the VD method are equal to or better than those obtained with the Classical Lagrangian Relaxation method, when solving a class of non-convex problems. The question is, then: for which practical problems could solution methodologies based on the CLR method be improved by using the VD method?

d) And finally, the fourth research point is the Radar Multiplier (RM) method. Good results have been obtained when solving the Generalized Unit Commitment (GUC) problem, a nonlinear mixed integer problem. Would it be possible to successfully export this methodology to solve other linear or nonlinear mixed integer problems?

Bibliography

[AaS98] S. Al-agtash and R. Su. Augmented Lagrangian approach to hydrothermal scheduling. *IEEE Transactions on Power Systems*, 13(4):1392–1400, November 1998.

[AC00] N. Alguacil and A. J. Conejo. Multiperiod optimal power flow using benders decomposition. *IEEE Transactions on Power Systems*, 15(1):196–201, February 2000.

[Act90] F. S. Acton. *Numerical Methods that Work.* The Mathematical Association of America, USA, 2n edition, 1990.

[Bal95] R. Baldick. The generalized unit commitment problem. *IEEE Transactions on Power Systems*, 10(1):465–475, February 1995.

[Ber82] Dimitri P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods.* Computer Science and Applied Mathematics. Academic Press, USA, 1982.

[Ber95] Dimitri P. Bertsekas. *Nonlinear Programming.* Athena Scientific, Belmont, Massachusetts, (USA), 1995.

[BF93] R . L. Burden and J. D. Faires. *Numerical Analysis.* PWS Publishing Company, Boston, USA, 5th edition, 1993.

[BH99] C. Beltran and F. J. Heredia. Short-term hydrothermal coordination by augmented Lagrangian relaxation: a new multiplier updating. *Investigacion Operativa*, 8(1,2 and 3):63–76, July–December 1999.

[BR92] J. Batut and A. Renaud. Daily generation scheduling optimization with transmission constraints: A new class of algorithms. *IEEE Transactions on Power Systems*, 7(3):982–987, August 1992.

[Bra77] S. P. Bradley. *Applied Mathematical Programming.* Addison Wesley Pub. Company, 1977.

[BS79] M. S. Bazaraa and C. M. Shetty. *Nonlinear Programming.* John Wiley and Sons, USA, 1979.

[CNH95] A. Chiva, N. Nabona, and F. J. Heredia. Network model of short-term optimal hydrothermal power flow with security constraints. In *Stockholm Power Tech. Proceedings*, volume PS, pages 67–73. IEEE, 1995.

[Coh80] G. Cohen. Auxiliary problem principle and decomposition of optimization problems. *Journal of Optimization Theory and Applications*, 32(3), November 1980.

[CPL95] CPLEX Optimization, Inc., Suite 279 930 Tahoe Blv., Bldg. 802. Incline Village, NV 89451-9436. USA. *Using the CPLEX Callable Library*, 1995.

[CS87] M. F. Carvalho and S. Soares. An efficient hydrothermal scheduling algorithm. *IEEE Transactions on Power Systems*, 2:537–542, 1987.

[CS98] C . C. Carøe and R. Schultz. A two-stage stochastic program for unit commitment under uncertainty in a hydro-thermal power system. Technical report, Universität Leipzing, Mathematisches Institut, 1998.

[CZ84] G. Cohen and D. L. Zhu. *Advances in Large Scale Systems*, volume I. JAI Press Inc., 1984.

[DB74] G Dahlquist and A. Bjorck. *Numerical Methods*. Prentice Hall, USA, 1974.

[DSC99] S. Dekrajangpetc, G. B. Sheblé, and A. J. Conejo. Auction implementation problems using Lagrangian relaxation. *IEEE Transactions on Power Systems*, 14(1):82–87, February 1999.

[Fer94] L. A. F. M. Ferreira. On the convergence of the classic hydro-thermal coordination algorithm. *IEEE Transactions on Power Systems*, 9(2):1002–1008, May 1994.

[FK00] S. Feltenmark and K. C. Kiwiel. Dual applications of proximal bundle methods, including Lagrangian relaxation of nonconvex problems. *SIAM Journal of Optimization*, 10(3):697–721, 2000.

[GJL$^+$98] C. Greif, R. B. Johnson, C. Li, A. J. Svoboda, and K. A. Uemura. Short-term scheduling of electric power systems under minimum load conditions. *IEEE Transactions on Power Systems*, 14(1):280–286, February 1998.

[GK87] M. Guignard and S. Kim. Lagrangean decomposition: a model yielding stronger Lagrangean bounds. *Mathematical Programming*, (39):215–228, 1987.

[GMW95] P. E. Gill, W. Murray, and M. H. Wright. *Practical optimization*. Academic Press Limited, London, 10th edition, 1995.

[GNLL97] X. Guan, E. Ni, R. Li, and P. B. Luh. An optimization-based algorithm for scheduling hydrothermal power systems with cascaded reservoirs and discrete hydro constraints. *IEEE Transactions on Power Systems*, 12(4):1775–1780, November 1997.

[Gro86] C. A. Gross. *Power System Analysis.* John Wiley and Sons Inc., USA, 2nd edition, 1986.

[HB86] H. Habibollahzdeh and J. A. Bubenko. Application of decomposition techniques to short-term operation planning of hydrothermal power systems. *IEEE Transactions on Power Systems*, 1(1):41–47, February 1986.

[Her95] F. J. Heredia. *Optimitazció de Fluxos No Lineals en Xarxes amb Constriccions a Banda. Aplicació a Models Acoblats de Coordinació Hidro-Tèrmica a Curt Termini.* PhD thesis, Dep. d'Estadística i Investigació Operativa. Universitat Politècnica de Catalunya, 1995.

[Her97] F. J. Heredia. Maph3 reference manual. Technical Report DR 97/11, Statistics and Operations Research Dept. Universitat Politècnica de Catalunya, 1997.

[HFS90] H. Habibollahzdeh, D. Frances, and U. Sui. A new generation scheduling program at Ontario hydro. *IEEE Transactions on Power Systems*, 5(1):65–73, February 1990.

[HN92] F. J. Heredia and N. Nabona. Numerical implementation and computational results of nonlinear network optimization with linear side constraints. In *Proceedings of the 15th IFIP Conference on System Modelling and Optimization*, 1992.

[HN94] F. J. Heredia and N. Nabona. Development and computational tests of an un-decoupled optimum short-term hydrothermal scheduling code using network flows. *TOP (Journal of the "Sociedad Española de Estadística e Investigación Operativa")*, 2(1):105–132, 1994.

[HN95] F. J. Heredia and N Nabona. optimum short-term hydrothermal scheduling with spinning reserve through network flows. *IEEE Transactions on Power Systems*, 10(3):1642–1651, august 1995.

[HSL+91] Y. Y. Hsu, C. C. Su, C. C. Liang, C. J. Lin, and C. T. Huang. Dynamic security constrained multi-area unit commitment. *IEEE Transactions on Power Systems*, 6(3):1049–1055, August 1991.

[HUL96] J. B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms*, volume I and II. Springer-Verlag, Berlin, 1996.

[JC99] N. Jiménez and A. Conejo. Short-term hydrothermal coordination by Lagrangian relaxation: Solution of the dual problem. *IEEE Transactions on Power Systems*, 14(1):89–95, February 1999.

[KS96] J. Kumar and G. B. Sheblé. Framework for energy brokerage system with reserve margins and transmission losses. *IEEE Transactions on Power Systems*, 11(4):1763–1769, November 1996.

[LB99] S. Lai and R. Baldick. Unit commitment with ramp multipliers. *IEEE Transactions on Power Systems*, 14(1):58–64, February 1999.

[LBSP82] G. S. Lauer, D. P. Bertsekas, N. R. Sandell, and T. T. Posbergh. Solution of large-scale optimal unit commitment problems. *Transactions on Power Apparatus and Systems*, PAS–101(1):79–86, January 1982.

[Lee88] F. N. Lee. Short-term unit commitment: a new method. *IEEE Transactions on Power Systems*, 3(2):421–428, May 1988.

[LHS$^+$97] C. Li, E. Hsu, A. J. Svoboda, C. Tseng, and R. B. Johnson. Hydro unit commitment in hydro-thermal optimization. *IEEE Transactions on Power Systems*, 12(2):764–769, May 1997.

[LHS$^+$98] C. Li, E. Hsu, A. J. Svoboda, C. Tseng, and R. B. Johnson. A robust unit commitment algorithm for hydro-thermal optimization. *IEEE Transactions on Power Systems*, 13(3):1051–1056, August 1998.

[LJS93] C. Li, P. J. Jap, and D. L. Streiffert. Implementation of network flow programming to the hydrothermal coordination in an energy management system. *IEEE Transactions on Power Systems*, 8(3):1045–1053, August 1993.

[LPS99] T. Larsson, M. Patriksson, and A. Strömberg. Ergodic, primal convergence in dual subgradient schemes for convex programming. *Mathematical Programming*, (86):283–312, 1999.

[Lue89] D. G. Luenberger. *Programación Lineal y no Lineal.* Addison-Wesley Iberoamericana, USA, 1989.

[LZT98] P. B. Luh, D. Zhang, and R. R. Tomastik. An algorithm for solving the dual problem of hydrothermal scheduling. *IEEE Transactions on Power Systems*, 13(2):593–600, May 1998.

[MEHA99a] J. A. Momoh, M. E. El-Hawary, and R. Adapa. A review of selected optimal power flow literature to 1993, Part I: nonlinear and quadratic programming approaches. *IEEE Transactions on Power Systems*, 14(1):96–104, February 1999.

[MEHA99b] J. A. Momoh, M. E. El-Hawary, and R. Adapa. A review of selected optimal power flow literature to 1993, Part II: Newton, linear programming and interior point methods. *IEEE Transactions on Power Systems*, 14(1):105–111, February 1999.

[Mij96] E. Mijangos. *Optimización de flujos no lineales en redes con restricciones laterales mediante técnicas de multiplicadores.* PhD thesis, Universitat Politècnica de Catalunya, 1996.

[Min83] Michel Minoux. *Programation Mathematique*, volume 1. Ed. Dunod, Paris, 1983.

[MS83] A. Merlin and P. Sandrin. A new method for unit commitment at Electricité de France. *IEEE Transactions on Power Systems*, PAS-102(5):1218–1225, May 1983.

[MS95] B. A. Murtagh and M. A. Saunders. Minos 5.4 user's guide. Technical report, Systems Optimization Laboratory. Dept. of Operations Research, Stanford University, 1995.

[MS99] H. Ma and S. M. Shahidehpour. Unit commitment with transmission security and voltage constraints. *IEEE Transactions on Power Systems*, 14(2):757–764, May 1999.

[NGL99] E. Ni, X. Guan, and R. Li. Scheduling hydrothermal power systems with cascaded and head-dependent reservoirs. *IEEE Transactions on Power Systems*, 14(3):1127–1132, August 1999.

[NR00] N. Nabona and F. Rossell. Short-term hydrothermal coordination through mixed-integer linear programming. Submitted to IEEE PES for consideration for publication in 'IEEE Transactions on Power Systems', 2000.

[NS97] O. Nilsson and D. Sjelvgren. Variable splitting applied to modeling of start-up costs in short-term hydro generation scheduling. *IEEE Transactions on Power Systems*, 12(2):770–775, May 1997.

[NSS98] O. Nilsson, L. Söder, and D. Sjelvgren. Integer modeling of spinning reserve requirements in short term scheduling of hydro systems. *IEEE Transactions on Power Systems*, 13(3):959–964, August 1998.

[OI98] S. O. Orero and M. R. Irving. A genetic algorithm modelling framework and solution technique for short term optimal hydrothermal scheduling. *IEEE Transactions on Power Systems*, 13(2):501–518, 1998.

[PRS96] F. Pellegrino, A. Renaud, and T. Socroun. Bundle and augmented Lagrangian methods for short-term unit commitment. In *12th Power System Computation Conference*, volume II, pages 730–739, Dresden (Germany), August 19–23 1996.

[PT] A. Piria and R. F. Tempone. A dual algorithm for the short term power production planning with network constraints. To be published in 'Investigacion Operativa'.

[RCCC96] A. Renaud, P. Carpentier, G. Cohen, and J. C. Culioli. Stochastic optimization of unit commitment: a new decomposition framework. *IEEE Transactions on Power Systems*, 11(2):1067–1073, May 1996.

[Roc73] R. T. Rockafelar. A dual approach to solving nonlinear programming problems by unconstrained optimization. *Mathematical Programming*, (5):354–373, 1973.

[Ros81] R.E. Rosenthal. A nonlinear network flow algorithm for maximization of benefits in a hydroelectric power system. *Operations Research*, 1981.

[RR98] S. Ruzic and N. Rajakovic. optimal distance method for Lagrangian multipliers updating in short-term hydro-thermal coordination. *IEEE Transactions on Power Systems*, 13(4):1392–1400, November 1998.

[SF94] G. B. Sheble and G. N. Fahd. Unit commitment literature synopsis. *IEEE Transactions on Power Systems*, 9(1):128–135, February 1994.

[SNH97] M. S. Salam, S. M. Nor, and A. R. Hamdan. Comprehensive algorithm for hydrothermal coordination. *IEE Proc.-Gener. Transm. Distrib.*, 144(5):482–488, September 1997.

[SNH98] M. S. Salam, K. M. Nor, and A. R. Hamdan. Hydrothermal scheduling based Lagrangian relaxation approach to hydrothermal coordination. *IEEE Transactions on Power Systems*, 13(1):226–235, February 1998.

[SPR$^+$97] P. Sandrin, F. Pellegrino, A. Renaud, et al. Unit commitment. Technical report, CIGRE task force 38-04-01, September 1997.

[STLJ97] A. J. Svoboda, C. Tseng, C. Li, and R. B. Johnson. Short-term resource scheduling with ramp constraints. *IEEE Transactions on Power Systems*, 12(1):77–83, February 1997.

[TGS98] C. Tseng, X. Guan, and A. J. Svoboda. Multiarea unit commitment for large-scale power systems. *IEEE Proceedings on generation, transmission and distribution*, 145(4):415–421, July 1998.

[TKW00] S. Takriti, B. Krasenbrink, and L S.-Y. Wu. Incorporating fuel constraints and electricity spot prices into the stochastic unit commitment problem. *Operations Research*, 48(2):268–280, March–April 2000.

[TS90] S. K. Tong and S. M. Shahidehpour. An innovative approach to generation scheduling in large-scale hydro-thermal power systems with fuel constrained units. *IEEE Transactions on Power Systems*, 5(2):665–673, May 1990.

[TSO91] S. K. Tong, S. M. Shahidehpour, and Z. Ouyang. A heuristic short-term unit commitment. *IEEE Transactions on Power Systems*, 6(3):1210–1216, August 1991.

[VAIM89] S. Virmani, E. C. Adrian, K. Imhof, and S. Mukherjee. Implementation of a Lagrangian relaxation based unit commitment. *IEEE Transactions on Power Systems*, 4(4):1373–1379, October 1989.

[VAL90] S. Virmani, E. C. Adrian, and E. O. Lo. Unit commitment - Lagrangian relaxation. *IEEE Transactions on Power Systems*, pages 37–47, 1990.

[WS93] S. J. Wang and S. M. Shahidehpour. Hydro-thermal scheduling for multiarea hydro-thermal systems with tie line constraints, cascaded reservoirs and uncertain data. *IEEE Transactions on Power Systems*, 8(3):1333–1340, August 1993.

[WS$^+$95] S. J. Wang, S. M. Shahidehpour, et al. Short-term generation scheduling with transmission and environmental constraints using an augmented Lagrangian relaxation. *IEEE Transactions on Power Systems*, 10(3):1294–1301, August 1995.

[YS97] I. Yu and Y. H. Song. Short-term generation scheduling of thermal units with voltage security and environmental constraints. *IEE Proc.-Gener. Transm. Distrib.*, 144(5):469–476, September 97.

[ZG88] F. Zhuang and F. D. Galiana. Towards a more rigorous and practical unit commitment by Lagrangian relaxation. *IEEE Transactions on Power Systems*, 3(2):763–773, May 1988.

[ZG90] F. Zhuang and F. D. Galiana. Unit commitment by simulated annealing. *IEEE Transactions on Power Systems*, 5(1):311–318, February 1990.

[ZLZ99] D. Zhang, P. B. Luh, and Y. Zhang. A bundle method for hydrothermal scheduling. *IEEE Transactions on Power Systems*, 14(4):1355–1361, November 1999.