

THÈSE

Pour obtenir le grade de

DOCTEUR

Spécialité : **Génie Électrique**

Arrêté ministériel :

Présentée par

Ghaith Warkozek

Thèse dirigée par **M Frédéric WURTZ**
et codirigée par **M Stéphane PLOIX**

préparée au sein **Laboratoire de Génie Électrique de Grenoble (G2eLab)**
et de **Électronique, Électrotechnique, Automatique et Traitement du Signal**

Génération automatique de problèmes d'optimisations pour la conception et la gestion des réseaux électriques des bâtiments intelligents multi-sources multi-charges

Thèse soutenue publiquement le **07/09/2011**,
devant le jury composé de :

Mme, Mireille Jacomino

Professeur à l'université de Grenoble, Présidente

M, Hubert Piquet

Professeur à l'université de Toulouse, Rapporteur

M, Pascal Berruet

Maître de conférence à l'université de Bretagne-Sud, Rapporteur

M, Jean Jacques Roux

Professeur à INSA de Lyon, Examinateur

M, Frédéric Wurtz

Directeur de recherche à l'INP Grenoble (G2eLab), Directeur de thèse

M, Stéphane Ploix

Professeur à l'INP Grenoble (GSCOP), Co-Directeur de thèse



Remerciements

Quatre ans ont déjà coulé depuis mon premier jour à Grenoble. Quatre ans de nouveaux rencontres, de naissance des riches amitiés et bien sûr de travail. Quatre ans vont tamponner ma vie pour jamais.

C'est difficile de résumer une approche scientifique, mais c'est encore plus difficile de résumer un sentiment de reconnaissance et de gratitude que on a pour certaines personnes.

Cela est mon cas pour Frédéric Wurtz et Stéphane Ploix. Je vous remercie de fond de mon coeur pour l'ouverture d'esprit que vous m'avez appris, je vous remercie pour le support scientifique et moral que vous m'avez donné.

Je tiens à remercier Frédéric Wurtz, pour la compétence scientifique qu'il a mis à ma disposition, pour son rigueur grâce au qu'elle j'ai pu année ce travail. Les mots échappent de moi pour remercier Stéphane Ploix. Le travail avec lui, est un mélange parfait entre profession et amitié Il y a toujours un lieu entre la discussion scientifique pour un bref pause géopolitique.

J'ai eu de la chance de travailler au sein de G2elab et GSCOP. Je tiens à remercier tous les personnels dans ces laboratoires pour tout ce qu'ont offert pour moi depuis mon arrivé pour le Master. Notamment M. Seddik Bacha celui qui restera pour jamais pour moi un grand frère.

Je ne me permets pas d'oublier de remercier la famille Perret, Robert et Françoise, vous êtes ma famille ici en France, merci pour le temps que vous m'avez consacré afin de m'aider et de me supporter.

Que je peux dire pour mes collèges et mes amis, Octavian, Diana, Cristian, Adrian, Anca, Petre, Iulian , Monica et Antoanetta. Les sorties, les pause café... , mes amis je vous dis 'Nu te voi uita'.

Puisque je suis venu en France avec une bourse de gouvernement syrien, la moindre chose à faire et de noter mon reconnaissance pour ce pays magnifique, j'espère que dans le futur le monde changera son pré avis et voir la réalité de la Syrie. Merci aussi pour le CROUS qui a organisé administrativement mon séjour, et aussi pour les belles rencontres avec d'étudiants de quatre points de monde, surtout aux enfants de 'mère Noelle' (Latéfa, Marina et Sinan).

Enfin, ma source de vie, mon prof pour jamais, maman et papa.. je sais que ce travail est loin d'être à approprié de votre support, mais permettez moi de le dédier pour vous et pour mes deux grand pères.

TABLE DES MATIÈRES

REMERCIEMENTS	3
INTRODUCTION GÉNÉRALE	3
I Problématiques et définitions	5
1 PROBLÉMATIQUE ÉNERGÉTIQUE DANS LE BÂTIMENT	7
1.1 INTRODUCTION	8
1.2 DÉFINITION DU BÂTIMENT INTELLIGENT (BI)	10
1.3 PROBLÉMATIQUES	11
1.3.1 Niveau méthodologique : L'efficacité énergétique et la modélisation du BI	12
1.3.2 Un nouvel outil : la génération dynamique de problèmes	14
1.4 LA PROBLÉMATIQUE DE DIMENSIONNEMENT OPTIMAL	15
1.4.1 Vers le dimensionnement optimal	15
1.4.2 Vers les études de sensibilité pour les sources énergétiques lors du dimensionnement des équipements de bâtiment	16
1.5 LA PROBLÉMATIQUE DE GESTION OPTIMALE DE FLUX ÉNERGÉTIQUES	19
1.5.1 Des approches existantes pour optimiser la gestion	19
1.5.2 Du dimensionnement vers la gestion	19
1.6 POSITIONNEMENT DE LA THÈSE DANS LE CONTEXTE DU BÂTIMENT INTELLIGENT	20
1.7 CONCLUSION	22
II Méthodes et concepts fondamentaux pour résoudre les problèmes d'optimisation dans le Bâtiment	25
2 GÉNÉRATION AUTOMATIQUE DE PROBLÈMES D'OPTIMISATION	27
2.1 INTRODUCTION	28
2.2 NOTION DE TYPES PRÉDÉFINIS DE SOUS-PROBLÈMES D'OPTIMISATION	29
2.3 NOTIONS DE PORTS ET DE CONTRAINTES GLOBALES	29
2.4 NOTION DE CONCEPTEUR GRAPHIQUE	31
2.5 CONCLUSION	33
3 PROBLÉMATIQUE DE MULTIPLES ENVIRONNEMENTS D'OPTIMISATION	35
3.1 INTRODUCTION	36
3.2 ALGORITHMES D'OPTIMISATION CONSIDÉRÉS	36

3.3 ENVIRONNEMENTS D'OPTIMISATION CIBLES	36
3.3.1 Fonctionnement de CPLEX	37
3.3.2 Fonctionnement de CADES	39
3.4 CONCLUSION	42
4 PROJECTION VERS DIFFÉRENTS ENVIRONNEMENTS D'OPTIMISATION	43
4.1 PROJECTION AUTOMATIQUE VERS DES ENVIRONNEMENTS DE RÉSOLUTION MULTIPLES : LES CONCEPTS ISSUS DE L'INGÉNIERIE DIRIGÉE PAR LES MODÈLES	44
4.2 DÉFINITION DE L'INGÉNIERIE DIRIGÉE PAR LES MODÈLES IDM	44
4.3 ADAPTATION DES CONCEPTS ISSUS DE L'IDM AU CONTEXTE DU BÂTIMENT	47
4.4 PROJECTION DES PROBLÈMES COMME UNE PROCÉDURE DE TRANSFORMATION DES MODÈLES	51
4.4.1 Principes	51
4.4.2 Transformation utilisée dans le contexte du bâtiment	53
4.5 GRANDES LIGNES DE L'APPROCHE DE GÉNÉRATION 'MULTI SOLVEUR' DE PROBLÈMES D'OPTIMISATION APPLIQUÉS AU CONTEXTE DU BÂTIMENT INTELLIGENT	53
4.6 CONCLUSION	56
5 MISE EN OEUVRE LOGICIELLE ET TECHNIQUES DE PROJECTION	57
5.1 INTRODUCTION : UN FORMALISME POUR DÉCRIRE LES MÉTAMODÈLES . .	58
5.2 LE MÉTAMODÈLE DU PIM OP-XML (NIVEAU M2)	58
5.2.1 Type Paramètre	60
5.2.2 Type Variable	61
5.2.3 Type Contrainte	63
5.2.4 Type Fonction objectif	66
5.2.5 Paquetage Java pour la génération de problèmes OP-XML	67
5.3 LES PROJECTEURS DES PSM (PIM OP-XML VERS DES PSM OP-CADES ET DES PSM OP-CPLEX)	68
5.3.1 La méthode	68
5.3.2 L'implémentation	71
5.3.2-1 Partie CADES : Projecteurs PIM OP-XML vers PSM OP-CADES	71
5.3.2-2 Partie CPLEX : Projecteur PIM OP-XML vers PSM OP-CPLEX	75
5.4 EXEMPLE D'IMPLÉMENTATION DE L'APPROCHE GÉNÉRATION AUTOMATIQUE MULTI SOLVEUR POUR UN UTILISATEUR D'APPLICATION	76
5.5 CONCLUSION	81
6 CONCLUSIONS PARTIE II	83

III	Implémentation de l'approche de génération automatique multi solveur pour résoudre les problèmes d'optimisation du bâtiment	85
7	PROBLÈMES DE DIMENSIONNEMENT OPTIMAL DES SOURCES ÉLECTRIQUES DANS LE BÂTIMENT	87
7.1	PRINCIPE DE DIMENSIONNEMENT DE SYSTÈMES BÂTIMENT	89
7.2	MÉTHODES DE DIMENSIONNEMENT DES SOURCES ÉLECTRIQUES	90
7.2.1	Contexte global	90
7.2.2	Méthode énergétique/économique utilisée	90
7.3	DIFFICULTÉS LIÉES À LA RÉSOLUTION DE PROBLÈMES DE DIMENSIONNEMENT DE SOURCES	91
7.3.1	Complexité des données nécessaires	91
7.3.2	Nécessité d'outils de résolution appropriés	94
7.4	CAPITALISATION DU SAVOIR FAIRE POUR LA PROBLÉMATIQUE DE DIMENSIONNEMENT DES SOURCES	97
7.4.1	Besoin d'outil métier	97
7.4.2	Les métiers et les acteurs concernés par les problématiques de dimensionnement	98
7.5	MISE EN OEUVRE POUR UN DIMENSIONNEMENT ÉNERGÉTIQUE-ÉCONOMIQUE DE SOURCES	101
7.5.1	Partie développement logiciel (pour le <i>développeur de logiciels</i>)	102
7.5.2	Partie modélisation (pour le <i>concepteur de types de modèles</i>)	102
7.5.2-1	Ajout des charges électriques et les données météorologique au PIM OP-XML	102
7.5.2-2	Ajout du modèle du fournisseur d'énergie électrique au PIM OP-XML	105
7.5.2-3	Ajout du modèle de panneaux solaires au PIM OP-XML	108
7.5.2-4	Ajout du stockage électrique au PIM OP-XML	112
7.5.3	Partie composition de bâtiments (pour le <i>concepteur de systèmes bâtiment</i>)	116
7.6	ÉTUDE PARAMÉTRIQUE DU PROBLÈME POUR LE DIMENSIONNEMENT DES SOURCES	119
7.6.1	Impact des données météorologiques	119
7.6.2	Impact des données relevant des hypothèses de type marché <i>HSEM</i>	123
7.6.2-1	Variation du coût des systèmes PV	123
7.6.2-2	Variation du coût du système de stockage	124
7.6.3	Impact de l'hypothèse sur le type d'usagers	125
7.6.3-1	Exemple d'hypothèses spécifiques aux usagers avec double tarif	125
7.6.3-2	Exemple d'hypothèses d'usagers spécifiques avec simple tarif	126
7.7	CONCLUSION	127
8	PROBLÈMES DE GESTION OPTIMALE DES SOURCES ÉLECTRIQUES DANS LE BÂTIMENT	129
8.1	DU DIMENSIONNEMENT À LA GESTION	130

8.2	MÉTHODES, ALGORITHMES ET Outils POUR LA GESTION GLOBALE DES SYSTÈMES BÂTIMENT	130
8.3	RÉSOLUTION DES PROBLÈMES DE GESTION OPTIMALE DE SYSTÈMES BÂTIMENT	132
8.3.1	Différents types de charges électriques	133
8.3.2	Différentes configurations de sources électriques	134
8.3.3	Différents environnements de résolution	134
8.4	MISE EN OEUVRE D'UN OUTIL MÉTIER POUR LA GESTION DES SOURCES	135
8.4.1	Partie Modélisation pour le <i>concepteur de types</i>	135
8.4.1.1	Ajout du type fournisseur d'énergie	135
8.5	OUTIL POUR LE DIMENSIONNEMENT ET LA GESTION DES SYSTÈMES BÂTIMENT	140
8.6	CONCLUSION	143
9	CARACTÉRISATION ET DÉTECTION DES PROBLÈMES DE GESTION À MULTIPLES SOLUTIONS	145
9.1	INTRODUCTION	147
9.2	MISE EN ÉVIDENCE D'UN EFFET INDÉSIRABLE GRÂCE À LA PROJECTION MULTI-SOLVEUR : L'EFFET W	147
9.2.1	Problème d'optimisation à projeter	147
9.2.2	Résultats obtenus par projection multi-solveur : solutions équivalentes et mise en évidence de l'effet W	149
9.2.3	Cas de Multiple solutions équivalentes : pas d'effet W	149
9.2.4	Cas de Multiples solutions équivalentes : apparition de l'effet W	155
9.3	RECHERCHE DES CONDITIONS POUR LESQUELLES SE PRODUIT L'EFFET W : PROBLÈMES D'OPTIMISATION À SOLUTIONS MULTIPLES	157
9.3.1	Définition du problème à multiples solutions équivalentes	157
9.3.2	Première analyse des conditions conduisant à de multiples solutions équivalentes	159
9.4	PROPOSITION DE MÉTHODES ET D'ALGORITHMES POUR DÉTECTOR LES PROBLÈMES D'OPTIMISATION À MULTIPLES SOLUTIONS ÉQUIVALENTE	159
9.4.1	Une approche par une optimisation mono-paramétrique	160
9.4.1.1	Implémentation de l'approche pour le problème de gestion .	162
9.4.1.2	Limite de l'approche	164
9.4.2	Une approche par une optimisation multi-paramétrique (algorithme D_{max})	164
9.4.2.1	Algorithme D_{max} en formulation linéaire mixte (<i>distance linéaire</i>)	165
9.4.2.2	Résultat d'implémentation d'algorithme D_{max} en formulation linéaire mixte	167
9.4.2.3	Algorithme D_{max} en formulation non linéaire (<i>distance quadratique</i>)	170
9.4.2.4	Résultat de l'implémentation de l'algorithme D_{max} en formulation non linéaire	171
9.4.2.5	Limites de l'algorithme D_{max} en formulation non linéaire	174

9.5 PERSPECTIVE POUR APPRÉHENDER LES PROBLÈMES D'OPTIMISATION À SOLUTIONS MULTIPLES POUR ACCROÎTRE LA ROBUSTESSE DE LA STRATÉGIE DE GESTION	174
9.5.1 Proposition pour découvrir les solutions équivalentes	175
9.5.2 Proposition pour utiliser les solutions équivalentes	177
9.6 CONCLUSION	178
10 CONCLUSIONS PARTIE III	179
IV Conclusions et Perspectives	181
11 CONCLUSIONS ET PERSPECTIVES	183
V Bibliographie	185
BIBLIOGRAPHIE	194
VI Annexes	3
12 ENVIRONNEMENTS D'OPTIMISATION ET ALGORITHMES UTILISÉS	5
12.1 EXEMPLE D'UN MODÈLE EN SML ET SES SPÉCIFICATIONS EN XML	5
12.2 ALGORITHMES D'OPTIMISATION UTILISÉS	5
12.2.1 L'algorithme du simplexe	5
12.2.3 LES ÉTAPES D'ALGORITHME DU SIMPLEX	6
12.2.4 L'ALGORITHME DE PROGRAMMATION SÉQUENTIELLE QUADRATIQUE (SQP)	7
12.2.5 LE PRINCIPE DE FONCTIONNEMENT DE L'ALGORITHME SQP	7
13 MÉTAMODÈLES ET GÉNÉRATEURS DE CODES M0	9
13.1 LES MÉTA MODÈLES D'ENVIRONNEMENTS D'OPTIMISATION (PM OP-*) ET LES GÉNÉRATEURS DE CODES M0	9
13.1.1 Le métamodèle pour CADES (PM OP-CADES)	9
13.1.2 Le métamodèle pour CPLEX (PM OP-CPLEX)	10
13.1.3 Instancier les métamodèles PM OP-CADES et PM OP-CPLEX	11
13.1.4 Générateurs de codes M0	11
13.1.4.1 Générateur de codes en CADES (OP-CADES)	11
13.1.4.2 Générateur de codes en CPLEX (OP-CPLEX)	13
14 AJOUTER LES MODÈLES DE PANNEAUX PHOTOVOLTAIQUES ET DE BATTERIES AU PIM OP-XML	15
14.1 AJOUTER LE PIM OP-XML POUR LES SOURCES ÉLECTRIQUES (PARTIE POUR LE <i>concepteur de type de modèles</i>)	15
14.1.1 PIM OP-XML de panneau photovoltaïque	15
14.1.2 PIM OP-XML de la batterie	16
15 OUTIL POUR LA GESTION DE BÂTIMENT (G-HOMETECH)	19

15.1 G-HOME TECH : UNE IMPLÉMENTATION DE SYSTÈME DE GESTION ÉNERGÉTIQUE POUR UN HABITAT	19
15.1.1 Introduction	19
15.1.2 Génération automatique de problèmes avec G-homeTech	19
15.1.3 Méthodes de gestion appliquées	20
15.1.4 Architecture du système	21
15.1.5 Procédure d'optimisation	22
15.1.6 Validation en temps réel	25
16 MÉTHODE ET MODÈLE ÉCONOMIQUE POUR LE DIMENSIONNEMENT	27
16.1 LES CRITÈRES ÉCONOMIQUES POUR LE CHOIX D'INVESTISSEMENT	27
17 TENTATIVE DE MISE EN OEUVRE D'UNE APPROCHE STRUCTURELLE POUR DÉTECTER LES PROBLÈMES D'OPTIMISATION À MULTIPLES SOLUTIONS ÉQUIVALENTES : PRINCIPAUX RÉSULTATS ET LIMITES MISES EN ÉVIDENCE	31
17.1 LA DÉCOMPOSITION DULMAGE-MENDELSON	31
17.1.1 Pourquoi l'approche par Dulmage-Mendelsohn (DM)	31
17.1.2 Description de découplage Dulmage-Mendelsohn (DM)	31
17.1.3 Application du découplage Dulmage-Mendelsohn pour l'analyse de systèmes linéaires	34
17.1.3-1 Exemple de découplage Dulmage-Mendelsohn (DM) pour un problème d'optimisation linéaire	34
17.1.3-2 Implémentation de découplage DM pour le problème de la gestion	36
17.1.4 Limite de l'approche par décomposition Dulmage-Mendelsohn pour la détection de l'effet W	38
17.1.4-1 Le pseudo-inverse de la matrice de coefficient	38
17.1.4-2 Implémentation de la pseudo-inverse	39
17.2 INTERPRÉTATION DU RÉSULTAT DE DÉCOUPLAGE DM EN MATLAB	42
17.3 RÉSULTAT DE DÉCOUPLAGE DM POUR PROBLÈME DE GESTION PAR MATLAB	44
18 TRANSFORMATION DE VALEUR ABSOLUE D'UNE EXPRESSION AVEC L'AIDE DES VARIABLES BINAIRES	47
18.1 OBJECT MANAGEMENT GROUP OMG	48

TABLE DES FIGURES

1.1	Consommation d'énergie primaire par énergie en France 2009 (MEDD 2010)	8
1.2	Consommation d'énergie électrique par secteur en France 2009 (MEDD 2010)	9
1.3	Évolution d'émission de CO ₂ en France depuis 1990 et la répartition des émission de CO ₂ liées à la combustion de l'énergie (ADEME 2007)	10
1.4	Les huit thématiques de recherche autour du bâtiment à énergie positive (Visier et al. 2009)	11
1.5	La croissance de la capacité de production d'énergie solaire installé en France	12
1.6	Les quatre systèmes composant le Bâtiment Intelligent BI	12
1.7	Objectifs pour les bâtiments existants	13
1.8	Les facteurs influençant la conception et la gestion des BI	14
1.9	Le principe de conception de systèmes bâtiments multi-sources	16
1.10	Données météorologiques typiques sur 12 jours	18
1.11	Deux fonctionnements équivalents de la batterie dans un système multi-sources	21
1.12	Synthèse du manuscrit de la thèse	23
2.1	Processus de résolution usuelle	28
2.2	Principe de composition automatique des problèmes d'optimisation	29
2.3	Conception de sous problèmes selon l'approche proposée	30
2.4	Notion de port d'un modèle de composant, Par exemple, dans le problème 2.2, les variables i_1 et i_2 et U sont des ports des modèles de R_1 et R_2	31
2.5	Deux types de concepteurs graphiques, à gauche un concepteur pour générer les modèles, et à droite, un concepteur pour implémenter les modèles existants	32
2.6	Démarche de génération automatique proposée	33
3.1	ILOG CPLEX Concert Technology pour les développeurs Java	37
3.2	Codes pour générer le problème d'optimisation comme un objet de ILOCT en Java	38
3.3	Appel de CPLEX en interactif pour la résolution de <i>demo.lp</i>	38
3.4	Modèle exploitable du problème 3.1 pour CPLEX	39
3.5	Démarche d'optimisation avec CADES : (a) décrire le modèle analytiquement 'boîte blanche' (b) générer le composant icar 'boîte noire' (c) appeler un algorithme d'optimisation et fournir la réponse	39
3.6	Une copie d'écran du générateur CADES pour le modèle analytique de <i>demo.sml</i>	40
3.7	Une copie d'écran de la calculette CADES	40
3.8	Une copie d'écran de l'optimiseur CADES, ajout d'une spécification (contrainte c_2) pour la variable x_2	41

3.9 Une copie d'écran de l'optimiseur CADES après la résolution	41
3.10 Aperçu de fonctionnement de CADES, modèle analytique du problème 3.1 en sml et spécification en xml	42
4.1 Les domaines d'application de l'ingénierie dirigée par les modèles (Miller & Mukerji n.d.)	45
4.2 Quatre niveaux d'abstraction proposés par l'ingénierie dirigée par les modèles	46
4.3 Transformation en Y de PIM en PSM selon le concept de l'IDM	47
4.4 Plates-formes nécessaires pour la génération 'multi solveur'	48
4.5 Modèles nécessaires pour la génération 'multi solveur' et leurs niveaux d'abstraction en IDM	49
4.6 Modèles du problème d'optimisation pour l'application bâtiment et niveaux d'abstraction	50
4.7 Un exemple des règles de transformation des modèles	52
4.8 Transformation de PIM vers PSM sous forme d'Y	53
4.9 Étapes de génération 'multi solveur' de problèmes d'optimisation avec les niveaux de modélisation et les acteurs concernés	54
4.10 Génération de problèmes d'optimisation dans l'application bâtiment pour un utilisateur d'application	55
4.11 Principe de métamodélisation adopté	56
5.1 Partie du méta modèle de PIM OP-XML correspondant aux déclarations des quatre éléments principaux d'un problème d'optimisation	59
5.2 Partie du méta modèle de PIM OP-XML de OP-XML avec les paramètres a, b, c et d	60
5.3 Partie du méta modèle de PIM OP-XML, pour définir un type appelé <i>VariableType</i> permettant de contrôler la restriction sur les types de variables à utiliser	61
5.4 Partie du méta modèle de PIM OP-XML, pour définir un type appelé <i>ImplementationInConstraint</i> permettant de décrire les informations liées à l'implémentation des variables dans les contraintes	61
5.5 Partie du méta modèle de PIM OP-XML et OP-XML illustré avec la variable x_1 du problème 3.1	62
5.6 Partie du méta modèle de PIM OP-XML, relative aux descriptions de sous élément <i>type</i>	63
5.7 Partie du méta modèle de PIM OP-XML, relative aux descriptions de sous élément <i>expression</i>	64
5.8 Partie du méta modèle de PIM OP-XML qui décrit l'élément de type <i>Constraint</i> et une partie de OP-XML correspondante à la description de c_1 du problème 3.1	65
5.9 Partie du méta modèle de PIM OP-XML et le OP-XML correspondant pour le problème 3.1	66
5.10 Diagramme des classes pour générer le modèle OP-XML	67
5.11 Étapes de la transformation des modèles	68

5.12 Chaque projecteur fait appel aux règles de transformation afin de générer les modèles cibles pour chaque environnement d'optimisation	69
5.13 Règle de transformation d'une variable de PIM OP-XML vers une variable PSM OP-CPLEX	70
5.14 Règle de transformation d'une variable de PIM OP-XML vers une variable PSM OP-CADES	70
5.15 Le <i>parser</i> lit l'OP-XML puis met les informations à la disposition des règles de transformation	71
5.16 Les codes et l'OP-XML généré pour l'exemple 5.1	72
5.17 Spécification de la contrainte 5.1 en XML pour CADES	73
5.18 La formulation 5.2 en sml générée automatiquement par le projeteur CADES et les spécifications en xml	74
5.19 Règle de transformation dans le projecteurs CPLEX pour ajouter une variable sur une contrainte avec une expression arithmétique	75
5.20 La contrainte 5.1 générée automatiquement pour CPLEX	76
5.21 Exemple en code Java pour construire le PIM OP-XML	77
5.22 Modèle OP-XML du problème d'optimisation 3.1	79
5.23 Modèles du problème 3.1 générés automatiquement	80
5.24 Capture écran du modèle .lp généré automatiquement 3.1	80
 6.1 Localisation dans le manuscrit de thèse	84
 7.1 Le principe de dimensionnement de systèmes bâtiments multi-sources utilisé dans la thèse	91
7.2 Valeur nette actualisée (NPV) pour un investissement sur 20 ans pour plusieurs prix de surplus	92
7.3 Partie du problème de dimensionnement de sources à écrire manuellement avec l'approche habituelle, et avec la syntaxe pour CADES : à gauche, les contraintes en SML et à droite, les spécifications en XML	95
7.4 Partie du problème de dimensionnement de sources à écrire manuellement avec l'approche habituelle, et avec la syntaxe AML pour CPLEX	96
7.5 Le besoin d'outil métier et les savoirs faire nécessaires	97
7.6 Le concepteur de bâtiments utilise l'application logicielle avec son savoir faire pour concevoir le bâtiment	99
7.7 Les trois acteurs principaux du dimensionnement des systèmes bâtiments multi-source	99
7.8 Processus de dimensionnement des sources et les rôles des acteurs avec l'approche de génération automatique multi solveur	100
7.9 Équilibre énergétique et économique	101
7.10 Le <i>développeur logiciel</i> développe les interfaces de type de modèles pour que le <i>concepteur de types de modèles</i> les implémente avec ses modèles spécifiques	102
7.11 Le <i>développeur logiciel</i> développe un éditeur de système bâtiment qui permet de grouper les concepteurs graphiques développés par <i>concepteur de types de modèles</i> pour les types de modèles	103
7.12 La charge totale et la partie décalable avec une distribution initiale	103

7.13 La charge totale avant et après la nouvelle distribution	104
7.14 Le concepteur graphique pour ajouter la charge électrique et les données météorologiques. Il est développé par le <i>concepteur de types de modèles</i> et à utiliser par le <i>concepteur de systèmes bâtiments</i>	105
7.15 Partie de classe Java du modèle de charge électrique qui doit être fait par le <i>concepteur de types de modèles</i> . Les contraintes locales s'ajoutent automatiquement dans l'OP-XML complet, les ports énergétiques sont groupés dans une seule contrainte globale par le SDK	106
7.16 Partie de l'OP-XML du problème complet généré automatiquement : le noeud économique dans lequel il y a la variable de coût initial du système avant l'ajout des composants de type fournisseur d'énergie (à gauche) et après l'ajout (à droite)	107
7.17 L'interface graphique à utiliser par le <i>concepteur de systèmes bâtiments</i> pour ajouter le fournisseur d'électricité	108
7.18 Aperçu de la variation du prix des modules solaires depuis 2001	109
7.19 Deux compteurs énergétiques pour calculer la revente et l'achat	110
7.20 Le concepteur graphique développé par le <i>concepteur de types de modèles</i> est à utiliser par le <i>concepteur de bâtiment</i> pour instancier le PV	112
7.21 Le concepteur graphique développé par le <i>concepteur de types de modèles</i> est à utiliser par le <i>concepteur de bâtiment</i> pour instancier la batterie	115
7.22 Partie de l'OP-XML dont la contrainte <i>systemInitiaCost</i> s'évalue automatiquement selon la composition faite par le <i>concepteur de systèmes bâtiments</i>	116
7.23 Partie de l'OP-XML : l'évaluation automatique de la contrainte <i>systemInitiaCost</i> après l'ajout de fournisseur et de PV	117
7.24 Partie de l'OP-XML : l'évaluation automatique de la contrainte <i>systemInitiaCost</i> après l'ajout de fournisseur et de PV et la batterie	118
7.25 Interface graphique de saisie des données météorologiques	119
7.26 Impact du choix des données météorologiques sur le dimensionnement et la rentabilité économique du bâtiment	120
7.27 Données utilisées dans l'étude paramétrique	122
7.28 Impact du prix du surplus et de celui des modules PV sur la valeur nette actualisée	123
7.29 Impact du prix de la batterie et du prix de rachat du surplus sur la valeur nette actualisée	124
7.30 Impact de la surface disponible pour installer les modules sur la valeur nette actualisée	126
 8.1 Charges vues comme des agents qui négocient leurs besoins énergétiques avec le système de pilotage	133
8.2 Exemple de répartition optimale des sources électriques sur une journée . .	134
8.3 Principe de génération du PIM OP-XML pour un équipement	136
8.4 Partie de classe Java qui représente le PIM OP-XML du fournisseur d'énergie (pour x_1) à écrire par le <i>concepteur de types de modèles</i>	136
8.5 Interface graphique permettant d'ajouter le PIM OP-XML du type fournisseur d'énergie	137

8.6	Partie de l'OP-XML illustrant x_1 pour l'instant $t = 0h$	138
8.7	Parties de l'OP-CPLEX résultantes de la projection automatique de l'OP-XML complet (x_1 pour l'instant $t = 0h$)	138
8.8	Évolution de l'OP-XML lors de la composition du système bâtiment, au moment où le <i>concepteur des gestionnaires énergétiques</i> ajoute un composant via l'interface graphique, la contrainte du bilan énergétique dans l'OP-XML sera modifiée automatiquement par le générateur de problèmes	139
8.9	Interface graphique pour l'outil de gestion et de dimensionnement des sources et charges	140
8.10	Interface graphique pour l'acquisition de données météorologiques	141
8.11	Aperçu de l'interface graphique pour analyser les résultats obtenus par CPLEX	142
9.1	Système bâtiment étudié	148
9.2	Données de problème 9.1	149
9.3	Utilisation de la batterie par CPLEX pour alimenter la charge électrique .	150
9.4	Illustration de gain et de la perte en euros sur les 24h trouvés par CPLEX .	150
9.5	Résolution de problème 9.1 avec CPLEX	151
9.6	Comparaison entre la résolution de 9.1 avec CADES et CPLEX	153
9.7	Comparaison entre le gain et la perte en euros sur les 24h trouvés par CPLEX et CADES	154
9.8	Gestion des sources trouvée par CPLEX pour le problème 9.1	155
9.9	Gestion des sources pour le problème 9.1 : en bleu CPLEX et en vert CADES	156
9.10	Problème à multiples solutions équivalentes : plusieurs valeurs des variables donnent la même valeur à la fonction objectif	158
9.11	Problème à multiple solutions équivalentes : résolution d'exemple 9.4 . .	158
9.12	Algorithme pour détecter l'existence de solutions équivalentes par une optimisation mono-paramétrique	161
9.13	Algorithme pour détecter un cas de solution multiple avec la marge associée	166
9.14	Valeur absolue de la différence (d_{max}) trouvée entre la solution S^1 et celle S^2 pour la gestion de la batterie $x_2(t)$ sur 24h	168
9.15	Comparaison entre deux gestions trouvées avec CPLEX en implémentant D_{max}	169
9.16	Algorithme pour détecter un cas de multiples solutions avec la marge maximale associée en formulation non linéaire	170
9.17	Comparaison entre deux gestions trouvées avec CADES en implémentant D_{max}	172
9.18	Marge d'énergie entre deux solutions équivalentes trouvées par CADES sur 24h	173
9.19	L'Initialisation de D_{max} peut affecter la distance maximale trouvée entre deux solutions équivalentes	174
9.20	Ensemble de solutions équivalentes	176
9.21	Diamètre de stabilité de <i>Stotskov</i>	177
9.22	Diamètre de stabilité de <i>Stotskov</i> pour le problème posé	177
10.1	Localisation dans le manuscrit de la thèse	180

12.1 Le modèle analytique de problème 3.1 en sml et ses spécifications en xml	5
12.2 Représentation schématique de la méthode du SIMPLEX	6
13.1 Approche classique pour utiliser CADES et l'amélioration menée grâce à la transformation de modèles	9
13.2 Méta modèle de PSM d'un problème de type LP en CADES (PM OP-CADES)	10
13.3 Méta modèle de PSM du LP en CPLEX (PM OP-CPLEX)	11
13.4 Étapes de transformation de modèles par programmation	12
13.5 Grandes lignes pour générer le modèle employable du LP pour CADES	12
13.6 Grandes lignes pour générer l'objet IloCplex du LP	13
14.1 Concepteur graphique développé par le <i>concepteur de type de modèles</i> pour que le <i>concepteur de gestionnaire du système bâtiment</i> puisse instancier le PIM OP-XML des panneaux photovoltaïques puis l'ajouter sur l'OP-XML complet du problème	16
14.2 Concepteur graphique développé par le <i>concepteur de type de modèles</i> pour que le <i>concepteur de gestionnaire de système bâtiment</i> puisse instancier le PIM OP-XML Batterie puis l'ajouter sur l'OP-XML complet du problème	17
15.1 Architecture de G-homeTech	20
15.2 Une représentation schématique pour le mécanisme anticipatif de résolution	22
15.3 Le principe d'optimisation multi couche	24
15.4 Exemple de résultats de validation de commande - contrôle de chauffage électrique en temps réel	25
15.5 Architecture matérielle G-HomeTech et simulateur temps réel RT-LAB	26
16.1 Valeur présente nette NPV pour un investissement sur 20 ans pour plusieurs prix de surplus	28
17.1 Un graphe biparti et un couplage	32
17.2 Décomposition DM d'un graphe biparti en trois composants canoniques	33
17.3 Décomposition DM sous MATLAB	35
17.4 Décomposition DM pour l'exemple 17.3	35
17.5 Codes MATLAB pour décomposition DM du problème 17.5	37
17.6 Décomposition DM après la transformation de l'exemple 17.9	40
17.7 Décomposition DM après la transformation de l'exemple 17.11	41
17.8 Limite de l'approche structurelle pour détecter l'effet W	41
17.9 Résultats de la décomposition DM sous MATLAB	43
17.10 La décomposition DM	43
17.11 Les indices d'équations p ordonnées selon la priorité de résolution par le découplage DM	44
17.12 Les indices de variables q ordonnées selon la priorité de résolution par le découplage DM	45
18.1 Les entreprises internationales participantes en OMG (Soley 2010)	49

Introduction générale

Avec l'ouverture du marché énergétique et l'arrivée des nouvelles technologies, le réseau électrique subit une modification non seulement dans son architecture mais aussi par rapport au rôle de ses gestionnaires. Le bâtiment est un exemple de cette transformation. Classiquement le bâtiment était un noeud de consommation or, sa capacité à produire de l'énergie électrique croît et cela oblige les concepteurs et les gestionnaires de réseaux électriques et de bâtiments à modifier leur vision pour intégrer cette nouvelle fonctionnalité. Le bâtiment devient de plus en plus un système complexe où les flux énergétiques doivent être gérés en fonction des usages : on parle de bâtiments intelligents. Il s'ensuit une complexité croissante pour les concepteurs, qui doivent s'intéresser autant au bâtiment lui-même (plusieurs sources électriques et multiplication des charges) qu'à ses équipements, sa gestion énergétique mais aussi aux interactions avec l'environnement extérieur (flux d'informations exogènes sur le marché d'énergie, prix d'achat et de revente, subventions à l'auto-consommation, etc...). Afin de bien gérer cette complexité, des outils de conception capables d'intégrer toutes les dimensions d'un système bâtiment compris au sein d'un environnement doivent être conçus.

Il est désormais nécessaire de coupler la phase de conception avec celle de gestion énergétique du bâtiment. Les travaux de cette thèse visent à proposer une démarche méthodologique permettant de formuler automatiquement les problèmes d'optimisation exploitables autant en conception qu'en exploitation du système bâtiment. La démarche est basée sur les concepts issus de l'ingénierie dirigée par les modèles. Notre terrain d'application est la conception et la gestion optimale des systèmes énergétiques des bâtiments multi-sources et multi-charges.

Dans la première partie, nous définissons le terme de bâtiment intelligent multi-sources et multi-charges¹. Nous caractérisons alors les problématiques de conception et de gestion d'énergie dans ces bâtiments. Nous montrons alors l'intérêt d'adapter les outils issus de l'ingénierie dirigée par les modèles dans ce contexte. Ce chapitre se termine par une présentation schématique de la thèse.

Dans la deuxième partie, nous abordons les méthodes et les concepts sur lesquels s'appuie notre démarche méthodologique. Dans ce cadre, nous commençons par introduire certains notions utilisées pour générer les problèmes d'optimisation progressivement, puis nous montrerons rapidement les environnements de résolution utilisés, puis un état de l'art sur l'IDM² est illustré. Nous présentons alors comment adapter les concepts de l'IDM pour générer les problèmes de conception et la gestion de bâtiments. Une mise en oeuvre logicielle et les techniques de projection sont illustrés.

La troisième partie présente l'implémentation réelles des concepts précédents pour la problématique de dimensionnement et la gestion de sources électriques dans le bâtiment multi source. Nous montrons comment les principes présentés dans deuxième partie permettent de résoudre des problèmes difficiles à résoudre habituellement. Le paradigme IDM permet de projeter le même problème vers des solveurs différents. Certains résultats montrent qu'il n'y a pas toujours une solution unique au problème de gestion d'énergie, mais un ensemble de solutions équivalentes. Une démarche structurelle et numérique est présentée pour détecter l'existence de tels espaces de solutions afin que le concepteur ou le gestionnaire puisse améliorer la gestion d'énergie dans le bâtiment.

1. intelligent est entendu au sens de géré énergétiquement

2. ingénierie dirigée par les modèles

Nous conclurons par un bilan sur l'ensemble des travaux présentés et présenterons des perspectives à ce travail.

Première partie

Problématiques et définitions

Chapitre 1

Problématique énergétique dans le Bâtiment

Nous savons bien que ce que nous faisons n'est que une goutte dans l'océan. Mais si cette goutte n'était pas dans l'océan, elle manquerait

mère Teresa

SOMMAIRE

1.1	INTRODUCTION	8
1.2	DÉFINITION DU BÂTIMENT INTELLIGENT (BI)	10
1.3	PROBLÉMATIQUES	11
1.3.1	Niveau méthodologique : L'efficacité énergétique et la modélisation du BI	12
1.3.2	Un nouvel outil : la génération dynamique de problèmes	14
1.4	LA PROBLÉMATIQUE DE DIMENSIONNEMENT OPTIMAL	15
1.4.1	Vers le dimensionnement optimal	15
1.4.2	Vers les études de sensibilité pour les sources énergétiques lors du dimensionnement des équipements de bâtiment	16
1.5	LA PROBLÉMATIQUE DE GESTION OPTIMALE DE FLUX ÉNERGÉTIQUES	19
1.5.1	Des approches existantes pour optimiser la gestion	19
1.5.2	Du dimensionnement vers la gestion	19
1.6	POSITIONNEMENT DE LA THÈSE DANS LE CONTEXTE DU BÂTIMENT INTELLIGENT	20
1.7	CONCLUSION	22

Résumé

Dans ce chapitre, nous montrons l'importance et le potentiel des bâtiments des points de vue énergétique et économique. Nous présenterons un état de l'art sur le concept de bâtiment intelligent (BI). Nous exposons les problématiques de conception et de gestion optimale de sources électriques et de charges en vue d'améliorer l'efficacité énergétique. Des travaux précédents ont conduit à des applications logicielles pour étudier ces problématiques en formalisant des problèmes d'optimisation dans un environnement de résolution donné et pour une configuration de bâtiment unique. Or, un système bâtiment peut avoir différentes configurations dépendant des sources, des charges mais aussi des occupants et de l'environnement du système bâtiment. Lorsqu'il faut appréhender de nouvelles configurations et changer d'environnement d'optimisation, il faut passer par une étape coûteuse de réécriture du problème d'optimisation. Nous présentons les raisons pour lesquelles les approches avec environnement de résolution unique ne sont plus adaptées aux problématiques du bâtiment intelligent.

1.1 Introduction

Depuis son apparition jusqu'à nos jours, l'homme a cherché à accroître son confort. Cela n'a pas épargné le bâtiment. Avec l'avancée des technologies, ces exigences se sont accrues et se sont accompagnées d'un accroissement régulier des consommations énergétiques sans nécessairement maîtriser l'impact environnemental. En France, en 2009, la consommation d'énergie primaire pour la production d'énergie électrique représente 111 [Mtep] sur un total de 259 [Mtep]¹, ce qui représente 42,8% (MEDD 2010) (figure 1.1).

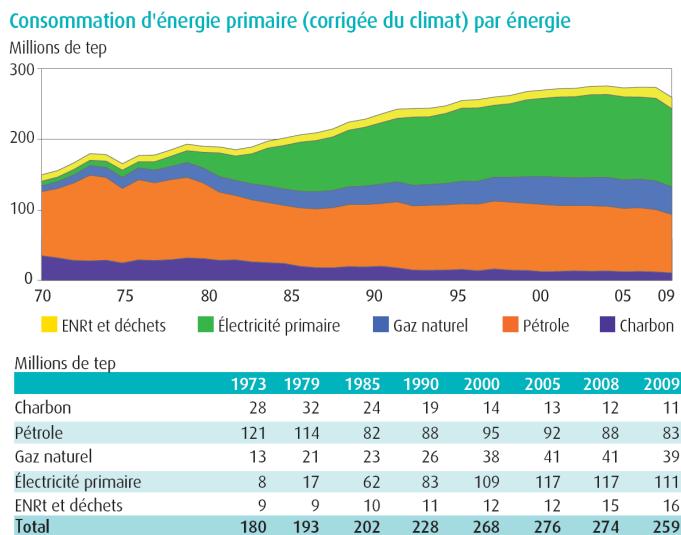


FIGURE 1.1 – Consommation d'énergie primaire par énergie en France 2009 (MEDD 2010)

Le secteur résidentiel-tertiaire consomme 289 TWh sur un totale de 441 TWh ce qui représente 65,5% de la production d'énergie électrique totale en 2009 (l'Industrie, l'agriculture et le transport représentent le reste de consommation, voir figure 1.2).

En parallèle, on constate que l'émission de gaz à effet de serre se stabilise depuis 1990 (figure 1.3). La France est en position de faire mieux que respecter ses engagements pris dans le cadre du protocole de Kyoto². Ceci a encouragé les chercheurs à envisager le bâtiment comme une piste importante de gain en efficacité énergétique.

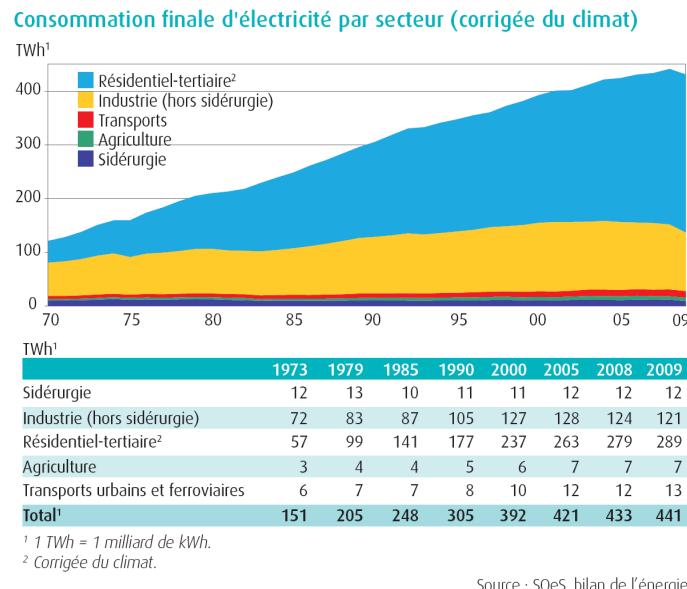
Les statistiques économiques, sociologiques et énergétiques établissent par ailleurs que la consommation électrique du secteur résidentiel et tertiaire représente environ 37% de la consommation électrique mondiale (IEA 2009). Cela s'explique par le fait que la majorité de notre temps est passée dans les bâtiments (soit au travail, soit à la maison). Le bâtiment présente ainsi des enjeux dans les domaines suivants :

- Des enjeux énergétiques et écologiques.

La consommation d'énergie électrique due au secteur bâtiment et son impact au niveau du climat sont des points sensibles. Ils font d'ailleurs l'objet de beaucoup de recherches ayant comme objectif d'améliorer l'efficacité énergétique du bâtiment. Cet objectif se décline aujourd'hui en la mise au point de bâtiments à énergie positive, qui

1. la *tep* tonne d'équivalent pétrole est une unité internationale pour l'énergie. Elle vaut selon les conventions 42 GJ. Le Mtep est un mégatonne d'équivalent pétrole.

2. stabilisation des émissions sur la période 2008-2012 par rapport à 1990.



Source : SoeS, bilan de l'énergie

FIGURE 1.2 – Consommation d'énergie électrique par secteur en France 2009 (MEDD 2010)

marient nouvelles technologies et équipements électro-mécaniques. La définition de bâtiment à énergie positive reste néanmoins floue et de multiples exceptions existent. Pour appréhender ce concept, nous proposons de revenir sur les huit objectifs présentés dans ([Visier et al. 2009](#)) (voir figure 1.4) qui conduisent à définir les bâtiments à énergie positive comme des bâtiments à faible consommation produisant plus d'énergie électrique qu'ils n'en consomment tout au long de leur cycle de vie.

- Des enjeux économiques.

C'est une sorte de conséquence de l'aspect énergétique car, en augmentant l'efficacité, nous réduisons la dépense (sur le long terme, car il y a aussi un investissement initial important qu'il faut prendre en compte). Or, les dépenses économiques dépendent de la politique énergétique des pays. En Allemagne par exemple, l'état encourage les particuliers à installer des panneaux photovoltaïques chez eux en subventionnant l'énergie autoconsommée.

En France, à partir du 1er janvier 2010, de nouveaux tarifs d'achat d'énergie solaire ont été appliqués. Le prix d'achat de l'énergie solaire produite chez les particuliers est autour de 0.58 euro/kWh ([Presse 2010](#)) (pour des panneaux photovoltaïques intégrés dans le bâtiment³). Selon les statistiques économiques sur l'investissement photovoltaïque en France (voir figure 1.5), la filière photovoltaïque est en passe de devenir un secteur stratégique dans le paysage énergétique français (la capacité de production installée est passée de 105 MW en 2008 à 250 MW en 2009 ([Prunak & Térouanne 2010](#))). Le calcul économique et de la profitabilité sont détaillés dans l'annexe 16.1.

Les courbes à droite sur la figure 1.5 présentent le TRI (taux moyen de retour sur investissement des projets intégrant les nouveaux tarifs de rachat). Dans le secteur résidentiel le taux n'est pas très élevé, car le coût d'un kW installé en France (6

3. cette valeur peut subir de changement, c'est un choix politique

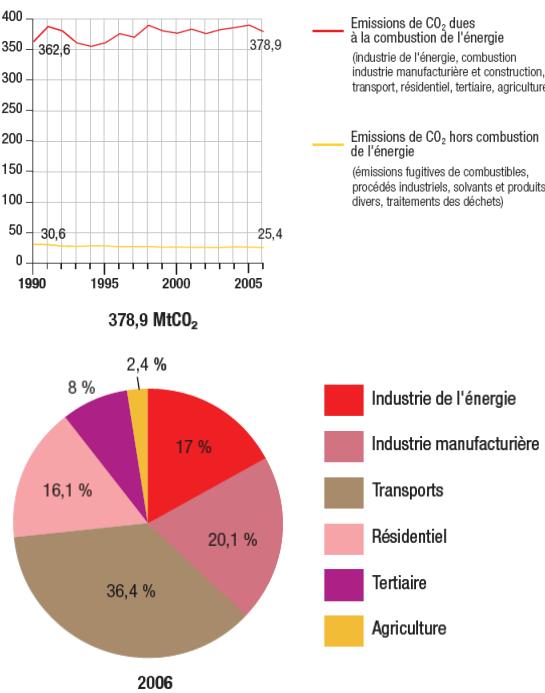


FIGURE 1.3 – Évolution d'émission de CO₂ en France depuis 1990 et la répartition des émissions de CO₂ liées à la combustion de l'énergie ([ADEME 2007](#))

euro/kW) est élevé (en Allemagne il vaut 3 euro/kW) ([TENERRDIS 2010](#)).

- L'enjeu architectural : Le bâtiment présente une surface potentielle importante pour installer des panneaux solaires. Il y a actuellement des projets qui ont profité de cette surface avec l'aide de l'architecte en ajoutant des PV sur les toitures et les façades.

1.2 Définition du Bâtiment intelligent (BI)

Une première application de *bâtiment intelligent* est apparue aux Etats-Unis dans les années 1970 sous le nom de *Building Energy management System* ou BEMS ([Nikolaou et al. 2004](#)). Il s'agit d'un centre d'agrégation et de traitement de données (*dump outstation*) pour les bâtiments. L'idée de bâtiment intelligent s'est consolidée dans les années 1980, renforcée par les développements des nouvelles technologies et de l'informatique.

Bien qu'il soit difficile de proposer une définition exacte de *l'intelligence*, nous pouvons dire que c'est la capacité à comprendre les choses et les faits c'est-à-dire de percevoir, puis d'agir. Pratiquement elle représente le pouvoir de s'adapter à des situations différentes, d'apprendre et de communiquer avec l'environnement.

Dans les travaux de ([Abras 2008](#)), nous trouvons une définition de Bâtiment Intelligent BI que nous adoptons dans cette thèse : "Ce bâtiment est capable de percevoir, de raisonner et d'agir sur son environnement afin de fournir des services à ses occupants comme la sécurité, la gestion de l'énergie, ou la surveillance pour des personnes dépendantes".

Le bâtiment intelligent BI devrait intégrer les quatre systèmes suivants ([Nikolaou et al. 2004](#)) (figure 1.6) :

1. Système d'automatisation du bâtiment (*Building automation System*) :

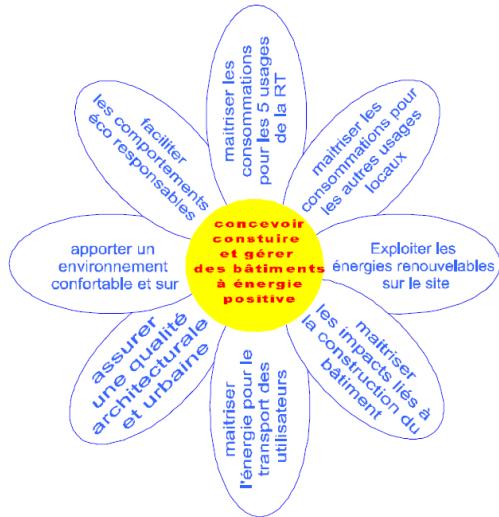


FIGURE 1.4 – Les huit thématiques de recherche autour du bâtiment à énergie positive (Visier et al. 2009)

Il fournit les services aux résidents, mais au niveau de l'ensemble du bâtiment : le système de ventilation et d'air conditionné (HVAC : *Heating Ventilation and Air Conditioning*), les systèmes d'éclairages et les systèmes de sécurité par exemple.

2. Systèmes d'automatisation de pièce (*Room automation System*) :

Il suit le même principe que le système précédent mais rapporté au niveau de chaque pièce.

3. Systèmes de contrôle/commande (*Computer Aided Facility Management System*) :

C'est le centre de gestion globale du bâtiment.

4. Système de communication (*Telecommunications System*) :

C'est l'infrastructure de communication entre les quatre systèmes et l'extérieur. Par exemple, les communications entre les capteurs de présence, capteurs thermiques avec les services HVAC, l'éclairage et le système de commande.

En 2006, il y a en France 30,7 millions de logements, dont 25,8 millions de résidences principales, 3 millions de résidences secondaires et 1,9 millions de logements vacants (Despretz 2009). La majorité d'entre eux ne comporte pas de systèmes comme ceux qui sont évoqués précédemment.

Nous considérons, dans cette thèse, que l'intelligence d'un bâtiment est due à la gestion d'énergie. Elle résulte d'abord d'une optimisation du dimensionnement des équipements du bâtiment, mais aussi, d'une optimisation de leurs fonctionnement.

Nous nous intéressons en particulier à la génération automatique des problèmes de dimensionnement et de gestion de bâtiments.

1.3 Problématiques

Dans cette section, nous introduisons les nouvelles problématiques liées au *bâtiment intelligent*.

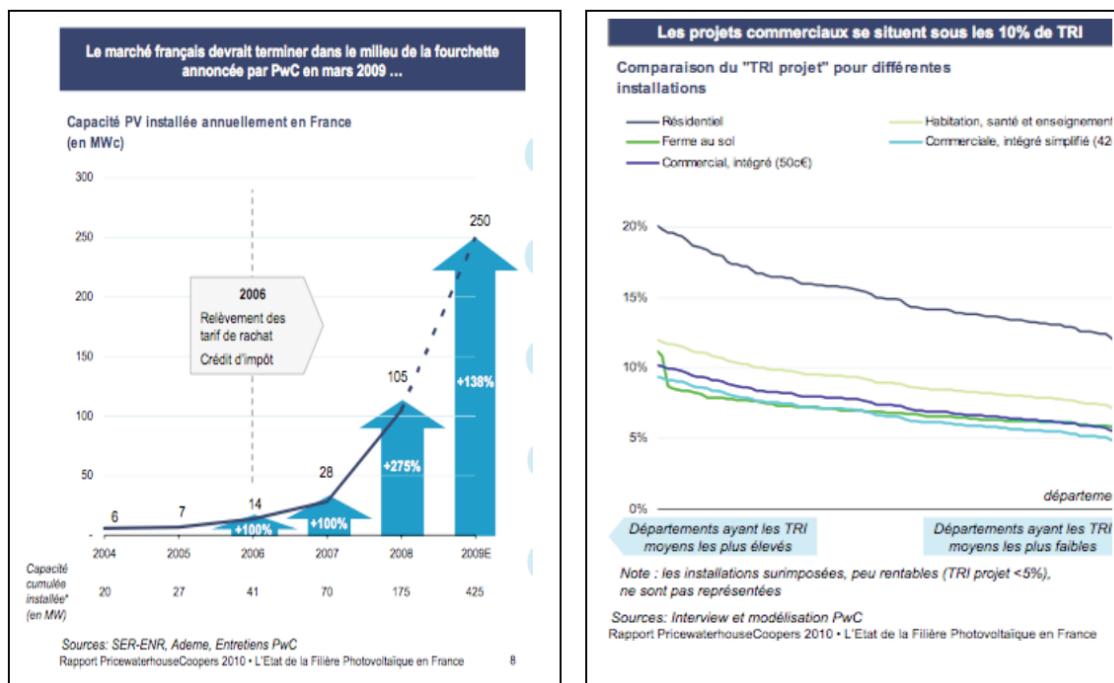


FIGURE 1.5 – La croissance de la capacité de production d'énergie solaire installé en France

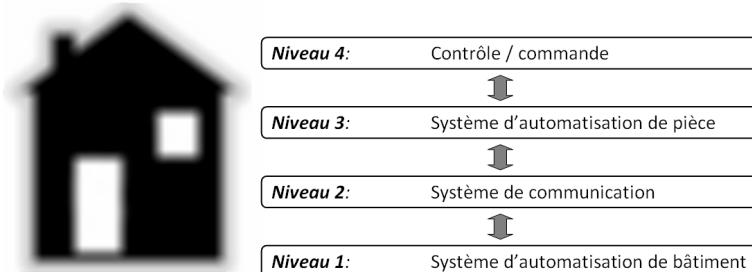


FIGURE 1.6 – Les quatre systèmes composant le Bâtiment Intelligent BI

1.3.1 Niveau méthodologique : L'efficacité énergétique et la modélisation du BI

Le Grenelle de l'environnement, qui s'est déroulé en 2008, a mis en évidence trois axes pour lutter contre le changement climatique conformément à la directive européenne dite des "3X20". Les trois axes de progrès décrits sont :

1. Réduction d'au moins 20% des émissions de gaz à effet de serre.
2. Amélioration de 20% de l'efficacité énergétique.
3. 20% d'énergie renouvelable dans la production énergétique.

Le Grenelle 1 contient 50 articles, mais ce qui concerne le secteur du bâtiment se trouve dans les articles 4 et 5. L'article 4 renforce la réglementation thermique dans le bâtiment neuf, alors que l'article 5 se concentre sur les bâtiments existants, et prend comme objectif de diminuer au moins de 28% la consommation énergétique dans le bâtiment à l'horizon

de 2020 (voir figure 1.7, (Despretz 2009)). D'un point de vue technique, cela induit un usage plus étendu de la modélisation car elle est un moyen indispensable pour concevoir des bâtiments performants et mettre au point des systèmes de gestion énergétique.

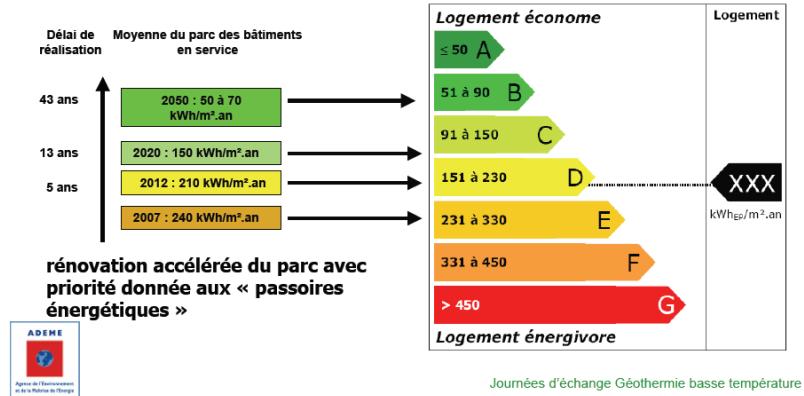


FIGURE 1.7 – Objectifs pour les bâtiments existants

L'efficacité énergétique du bâtiment va nécessiter de s'intéresser aux technologies suivantes :

1. climatisation et chauffage très performants
2. éclairage naturel avec exploitation de la lumière de jour par des équipements appropriés
3. appareils de basse consommation (utiliser des appareils électroménagers à haute performance énergétique).
4. isolation thermique renforcée
5. systèmes de contrôle/commande.

Les nouvelles problématiques énergétiques se déclinent en deux directions principales :

1. Comment formuler la problématique de gestion énergétique pour s'adapter aux attentes de confort des occupants tout en minimisant l'énergie requise ?
2. Comment améliorer le rendement des équipements du bâtiment ? L'efficacité énergétique n'est pas limitée à l'enveloppe du bâtiment mais s'étend à tous ses équipements.

La problématique 1 se décline suivant deux aspects :

1. variété des domaines d'application. Il faudra être capable de marier des outils issus de différents métiers, qu'il s'agisse de thermique, d'électromécanique, d'électricité ou de contrôle/commande.
2. diversité et dynamicité des éléments impliqués et des interactions. Chaque système bâtiment est unique et évolue au regard de sa durée de vie. Cette diversité et cette dynamicité des éléments du système influencent largement la conception et la gestion des bâtiments. La figure 1.8 illustre les quatre facteurs majeurs influençant le système bâtiment.

Les besoins et comportements des occupants sont très divers, même au niveau des membres d'un même foyer. Les usagers influencent fortement les problématiques de

gestion des sources et des charges. La modélisation du comportement des occupants est un élément crucial de la conception de bâtiment (Hawarah et al. 2010).

Les variables météorologiques induisent chez les occupants certains comportements comme, par exemple, de démarrer le chauffage ou la climatisation. Ce facteur doit donc être pris en compte par les systèmes de gestion énergétique.

Les installations photovoltaïques (PV) augmentent en France (voir figure 1.5) certains pays permettent aussi l'installation de systèmes de stockage d'énergie conjointement au PV. Les bâtiments producteurs d'énergie rendent les problématiques de gestion plus complexes.

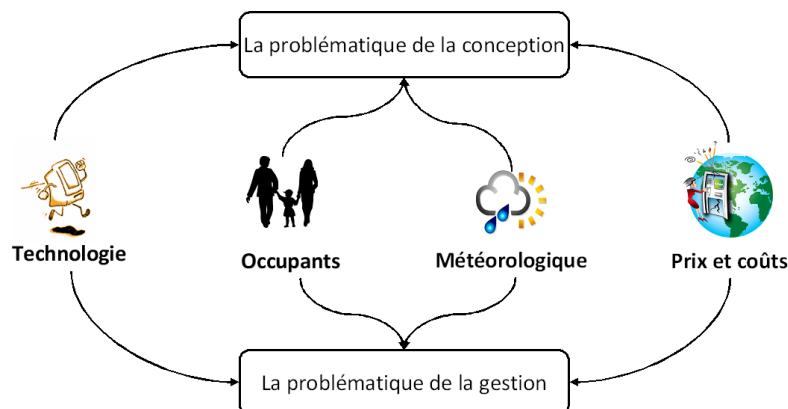


FIGURE 1.8 – Les facteurs influençant la conception et la gestion des BI

Face à ces deux axes, la problématique devient difficile à étudier avec des approches de génération existantes (nous allons en parler dans la suite), c'est-à-dire formuler la problématique avec un langage et l'associer avec un solveur en particulier ; En plus, ces approches limitent la générnicité.

Pratiquement, nous ne pouvons pas classifier les quatre éléments de la figure 1.8 selon leurs impacts sur le dimensionnement et la gestion de sources du bâtiment. Cela demande une étude statistique hors de l'objectif de cette thèse. Par contre, nous pouvons attirer l'attention sur certains aspects présentés dans les paragraphes suivants.

1.3.2 Un nouvel outil : la génération dynamique de problèmes

Il existe de nombreuses configurations de sources et de charges pour un système bâtiment. Un bâtiment peut être connecté ou isolé du réseau de distribution électrique, il peut disposer d'un système de stockage électrique ou non (Muselli et al. 2000), etc.... Ces charges peuvent changer ou être déplacées (Ha et al. 2008). Il faut donc mettre au point des outils de conception capables de s'adapter facilement à la variété des configurations mais aussi à leur évolution dynamique.

1.4 La problématique de dimensionnement optimal

1.4.1 Vers le dimensionnement optimal

Il s'agit de trouver les paramètres optimaux pour le dimensionnement des sources face aux charges électriques. Chaque source électrique est caractérisée par un ensemble de paramètres, parmi lesquels il y a ceux qui définissent la capacité productive. La phase de dimensionnement consiste à trouver ces paramètres de dimensionnement à partir de certaines caractéristiques connues et des modèles analytiques des sources.

Le travail de [Mondol et al. \(2006\)](#) propose une étude détaillée sur les facteurs influençant le dimensionnement de l'onduleur et des panneaux solaires pour un système multi sources, en faisant varier l'angle d'inclinaison, le site géographique et la surface disponible.

Le contexte économique a un impact sur le dimensionnement. [Bakos et al. \(2003\)](#) illustre que la rentabilité d'un système PV dépend de la stratégie appliquée par le fournisseur d'énergie et le prix de rachat d'énergie produite avec des sources locales. Cela impose des contraintes économiques sur le dimensionnement de système PV à installer. Dans ([Smiley & Jones 2000](#)) et ([Nafeh 2009](#)) le nombre optimal de panneaux solaires est recherché en minimisant le coût du cycle de vie pour un système multi sources isolé (PV, batterie sans raccordement au réseau de distribution). En ([Smiley & Jones 2000](#)) un simple programme est développé pour trouver le nombre optimal de panneaux solaires. Par contre, le programme ne permet pas de déterminer la taille de la batterie ni la puissance du générateur diesel. En ([Nafeh 2009](#)) un cas d'étude précis a été pris en compte pour trouver le dimensionnement des PV.

Un autre facteur de l'aspect économique dans les systèmes raccordés au réseau de distribution, est le coût d'abonnement avec le fournisseur d'énergie. Dans ([Angioletti & Despretz 2004](#)) nous trouvons que l'abonnement avec le fournisseur d'énergie se fait sous une tranche de consommation (6, 9, 12 kW), et le coût d'abonnement est différent pour chaque tranche.

Dans ([Notton et al. 2009](#)) le dimensionnement de systèmes PV s'est fait selon des contraintes techniques de l'onduleur et l'angle d'inclinaison des PV. Des différents technologies de fabrication des PV sont comparées pour voir l'impact sur le dimensionnement.

Les potentiels des techniques de l'intelligence artificielle comme un outil pour la conception dans le dimensionnement d'un système PV sont présentées dans ([Mellit 2007](#)). En particulier, dans les régions isolées où les données météorologiques sont rarement disponibles.

Au sein du laboratoire G2Elab, des chercheurs travaillent autour de la problématique de dimensionnement des sources dans le BI. Par exemple, [Pham et al. \(2009\)](#) développent une démarche pour dimensionner des systèmes multi-sources. Les auteurs proposent de concevoir des systèmes bâtiments multi-sources en prenant en compte des variations de facteurs exogènes, comme les prix de l'énergie, la subvention pour les panneaux, les courbes d'ensoleillement..etc. Le problème a été formulé sous la forme d'un problème de programmation linéaire en nombres entiers (PLNE). Le solveur utilisé était ILOG CPLEX ([ILOG 2010](#)). La figure [7.1](#) présente l'objectif de cette étude.

Ce travail avait pour but d'étudier les problématiques de dimensionnement sans travailler sur l'aspect génération de problème. En effet, chaque nouvelle configuration requiert une nouvelle formulation de problème et une nouvelle résolution.

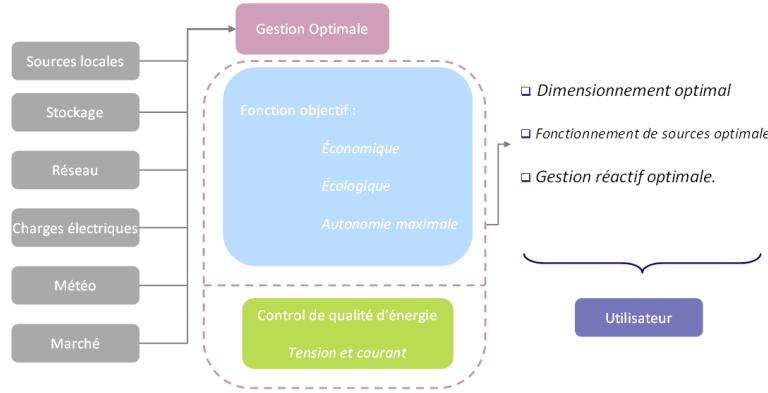


FIGURE 1.9 – Le principe de conception de systèmes bâtiments multi-sources

1.4.2 Vers les études de sensibilité pour les sources énergétiques lors du dimensionnement des équipements de bâtiment

Les questions qui préoccupent le concepteur de système multi-sources sont : quels types, quelles dimensions et combien de sources doit-il y avoir dans un bâtiment intelligent ? Quel est le coût d'investissement optimal pour un retour sur investissement donné ?

Les réponses à ces questions sont liées aux quatre facteurs montrés dans la figure 1.8. Il faut donc faire des études de sensibilité dans lesquelles les variables de dimensionnement seront tracées par rapport aux variations des facteurs exogènes et internes. Pour faciliter ce genre d'étude, nous allons proposer de classifier les facteurs impliqués selon trois catégories d'hypothèses [Warkozek, Ploix, Jacomino & Wurtz \(2010\)](#)⁴ :

1. les hypothèses spécifiques aux utilisateurs (HSU) : dans cette catégorie, nous introduisons les paramètres et les hypothèses associées qui ont un lien direct avec les occupants et leur environnement personnel perçu comme par exemple l'ensoleillement, la température ressentie, etc. Nous considérons aussi dans cette catégorie d'hypothèses, le crédit et les subventions reçues pour acheter les équipements du système bâtiment car ils dépendent du choix de l'utilisateur (par exemple acheter les panneaux avec un crédit ou comptant) alors que leurs valeurs dépendent du marché.
2. les hypothèses spécifiques aux sources électriques (HSS) : nous introduisons dans cette catégorie les paramètres et les hypothèses liés directement aux sources électriques, comme par exemple les données du cahier des charges (rendement de l'onduleur, puissance crête du panneau solaire, etc.), la surface disponible pour monter les panneaux solaires, . . .
3. les hypothèses spécifiques au marché énergétique (HSEM) : nous introduisons dans cette catégorie les paramètres qui ont un aspect économique et financier, comme les prix de l'énergie électrique et de l'énergie solaire, le taux de subvention, le taux d'inflation, le niveau d'abonnement avec le réseau électrique, . . .

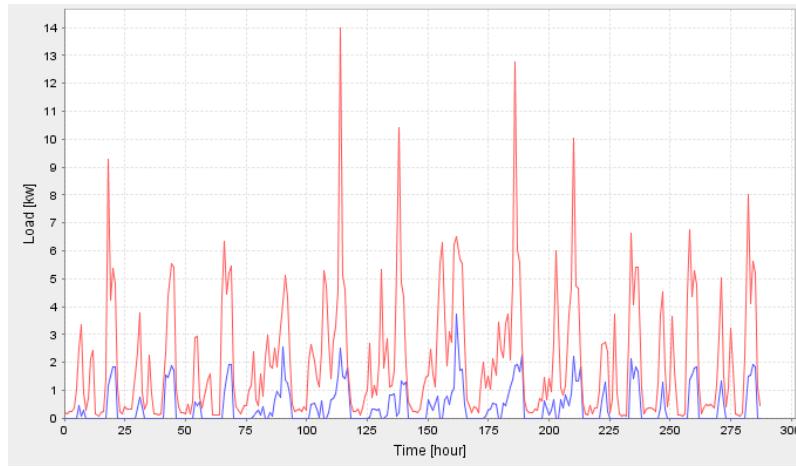
En guise d'illustration, considérons le problème suivant. Un occupant dont les courbes de consommation globale et les courbes météorologiques sont données par la figure 1.10

4. Cette classification ne veut dire pas que les hypothèses sont parfaitement indépendantes.

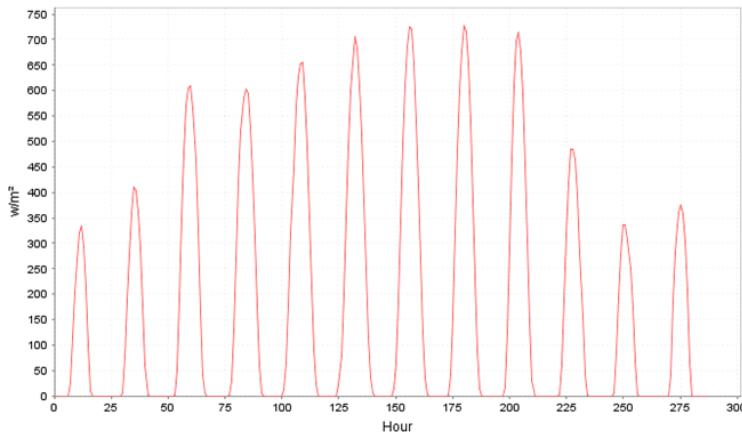
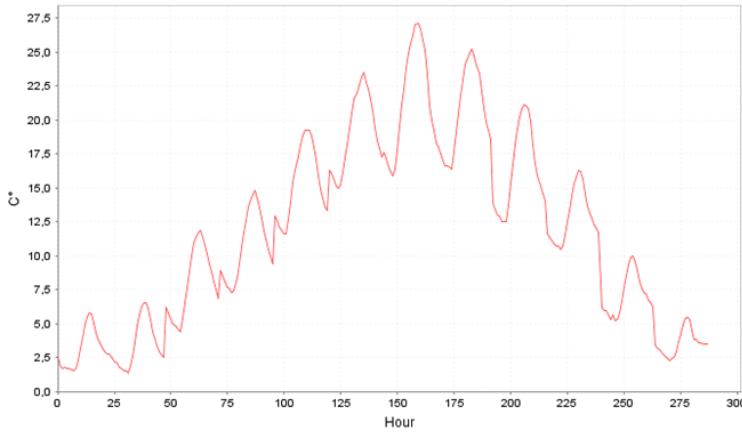
veut connaître la profitabilité d'un système raccordé au réseau électrique et composé de panneaux solaires avec une batterie, la surface où il peut installer les panneaux étant de $75m^2$ maximum.

Si le concepteur/gestionnaire de bâtiment veut connaître l'impact d'une variation de la subvention de 0% jusqu'à 30% (*HSU_0.0,HSU_15,HSU_30*) par rapport au dimensionnement d'un tel système, ou même tester une autre topologie de réseau de bâtiment (en ajoutant ou supprimant une source) avec les approches existantes, il aurait besoin de reformuler un problème d'optimisation, puis de le résoudre avec un solveur approprié ; mais au regard de la taille de ce problème (nombre de variables et de contraintes), seuls certains solveurs en sont capables. Ainsi, en adoptant une approche classique, il aurait fallu récrire le problème pour chaque cas d'étude dans des formalismes compréhensibles par chacun des solveurs testés, jusqu'à ce que le solveur approprié ait été trouvé.

Dans le chapitre 7, nous montrerons l'avantage de cette catégorisation, ainsi que le gain résultant de l'approche proposée dans cette thèse.



(a) Estimation de la consommation globale (en bleu la partie de charge qu'on peut la décaler, en rouge la charge totale)



(b) Mesures d'ensoleillement (en bas) et de température (en haut)

FIGURE 1.10 – Données météorologiques typiques sur 12 jours

1.5 La problématique de gestion optimale de flux énergétiques

1.5.1 Des approches existantes pour optimiser la gestion

Dans la littérature nous trouvons des travaux où les techniques de l'intelligence artificielle (IA) sont appliquées pour la gestion des sources dans le bâtiment. ([Penya 2003](#)) propose une approche basée sur les algorithmes génétiques pour la gestion d'énergie, le problème de gestion est formulé comme un problème multi-objectif avec une gestion de charges électriques. Les systèmes multi agents sont aussi un domaine où l'intelligence artificielle est utilisée pour la gestion. ([Boman et al. 1998](#)) illustre qu'il est possible d'atteindre une économie sur la consommation en utilisant un tel système.

Il y a des travaux sur les techniques de contrôle/commande pour les sources et les charges dans le bâtiment. Par exemple, ([Henze & Dodier 2003](#)) présente une proposition de commande adaptative d'un système PV isolé avec stockage. Dans ce travail les auteurs utilisent la commande prédictive pour anticiper une solution pour le système de chauffage et de climatisation dans le bâtiment. Alors que ([Ha et al. 2006](#)) propose un système de gestion avec commande anticipative et réactive pour la gestion de bâtiment.

([Long 2007](#)) propose une approche de pilotage de plusieurs niveaux. Dans un premier temps, un algorithme anticipatif a été appliqué pour mieux gérer les sources et les charges. Si les données changent au cours du temps, un algorithme réactif adaptera la commande suite à ce changement pour garantir un bon compromis entre le niveau de confort pour l'usager et le coût économique. Les techniques de contrôle/commande ont été appliquées et validées à la gestion d'énergie d'un bâtiment photovoltaïque et à la prévention de blackouts par un délestage intelligent.

([Abras 2008](#)) s'appuie sur des outils issus de l'intelligence distribuée (système multi-agents) pour mieux piloter les sources et les charges sans centraliser la commande. Il s'agit de trouver une solution au pilotage informatique du système énergétique, soit à distance, soit localement. Chaque agent associé à plusieurs équipements (service) communique avec les autres pour négocier ses besoins énergétiques ainsi que sa capacité à fournir de l'énergie aux autres. Contrairement à l'approche centralisée, une partie de la résolution du problème énergétique se fait au niveau de l'agent lui-même. Chaque agent produit indépendamment ses plans locaux, puis il envoie son plan à un agent facilitant la résolution (agent solving) afin d'établir un plan global de consommation et de production. Selon cette étude, le délestage « intelligent » réalisé par ces agents permet de garantir un bon niveau de satisfaction auprès des utilisateurs en s'appuyant sur les flexibilités des services.

Les deux approches précédentes (centralisée, distribuée) ont un point commun : les deux formulent des problèmes d'optimisation qui évoluent au cours du temps.

1.5.2 Du dimensionnement vers la gestion

Pour améliorer l'efficacité énergétique des bâtiments, le dimensionnement des équipements du bâtiment doit être complété par une conception efficace des lois de commande du système de gestion énergétique, dont le rôle est d'adapter le fonctionnement des équipements aux différentes situations rencontrées en pratique durant la vie d'un système bâtiment ([Webster & Sebastien 2006](#)) et ([Merieux & Pleynet 1992](#)).

Nous avons favorisé cette approche dans les travaux implémentés dans cette thèse. Cela veut dire que le même problème d'optimisation formulé pour trouver le dimensionnement contient aussi une partie pour trouver la gestion correspondante à ce dimensionnement. En conséquence, au niveau de modélisation et de formulation de problèmes dans cette thèse, on distingue une différence entre la génération de problèmes de dimensionnement et la génération de problème de gestion du système. Cela tient en deux points principaux suivants.

Pour le dimensionnement optimal, on joue sur les paramètres qui ont un impact sur les points de fonctionnement de système. Par ailleurs, le dimensionnement est un problème couvrant en même temps une gestion à long terme, à l'échelle de quelques années, alors que la gestion optimale est un problème de court terme, à l'échelle de quelques minutes voire de quelques heures. Cela induit une différence au niveau de l'horizon temporel (donc au niveau de la modélisation). Par exemple, comme nous allons voir dans le chapitre 7, en dimensionnement, nous pouvons modéliser l'énergie achetée au fournisseur comme une variable d'optimisation de taille n ; plus n est grand et plus le dimensionnement est fiable ($n = 24h, 48h, 8760h$.etc). Alors que pour le cas de la gestion, la même variable n vaudra souvent 24h (gestion anticipative $j-1$) ou une autre unité de temps dans le cas de gestion réactive (voir l'annexe 15.1).

La problématique de gestion étant à plus court terme, il est nécessaire de calculer des solutions rapidement. Les optimisations à effectuer sont donc soumises à des contraintes de temps de calcul plus sévères, cela pourrait limiter l'implémentation avec certains solveurs qui doivent pouvoir être sollicités régulièrement.

Face à toutes ces problématiques, la génération des problèmes de dimensionnement et de gestion des sources électriques doit être générique et dynamique.

1.6 Positionnement de la thèse dans le contexte du bâtiment intelligent

Dans ce contexte, les travaux de cette thèse visent d'abord à proposer aux concepteurs/gestionnaires de bâtiment des outils permettant de formuler dynamiquement des problèmes d'optimisation, que ce soit pour du dimensionnement ou pour de la gestion. Il s'agit notamment de proposer une solution logicielle pour projeter facilement les problèmes générés vers différents environnements de résolution.

Nous proposons une approche basée sur un paradigme venu du monde du génie logiciel : il s'agit de l'ingénierie dirigée par les modèles (IDM) (*Model Driven Engineering* MDE) et de la transformation de modèles.

Le but est de mettre au point un outil qui va permettre de générer automatiquement le problème posé, à partir d'une description abstraite et standardisée, sous un format transportable directement dans un environnement d'optimisation.

Aujourd'hui, pour utiliser ces environnements, l'utilisateur (de ces environnements) doit généralement reformuler son problème dans l'environnement d'optimisation retenu, consacrant ainsi du temps pour se familiariser avec les différentes syntaxes, le déverminage des codes et le traitement de résultats. Ceci est source de perte de temps pour de nombreux utilisateurs. Il serait préférable de profiter du temps disponible pour développer la problé-

matique et pour analyser les résultats. La génération automatique ne signifie pas que tout se fait sans l'intervention de l'utilisateur de l'outil mais que certaines tâches sont factorisées une bonne fois pour toutes : celles qui relèvent de la projection dans un environnement de résolution particulier.

Pour valider l'implémentation concrète de l'approche basée sur l'IDM, la résolution du problème d'optimisation sera effectuée dans deux environnements d'optimisation. Au G2Elab, les chercheurs ont développé un environnement d'optimisation multi-physique appelé CADES (Delinchant et al. 2007). CADES a montré une performance importante pour le dimensionnement dans le domaine des machines électriques, actionneurs et capteurs électromagnétiques. Nous souhaitons augmenter la capacité de CADES en intégrant la problématique énergétique du BI, en développant un outil métier qui va permettre de projeter les problèmes énergétiques sur CADES. Le deuxième environnement est CPLEX (ILOG 2010), la performance de CPLEX pour les problèmes de type PLNE⁵ est reconnue. Les présentations de ces deux environnements se trouvent dans le **chapitre 3**.

Nous avons réussi à appliquer les paradigmes issus de l'IDM à cette problématique pour résoudre le problème de dimensionnement des sources et d'utilisation dans différents solveurs (**chapitre 7**) puis pour résoudre le problème de génération dynamique des problèmes de gestion (**chapitre 8**).

Nous avons constaté en comparant deux solutions de gestion dans deux solveurs différents que des ensembles de solutions théoriquement équivalentes peuvent exister et que cela pose des problèmes pratiques réels (**chapitre 9**). La figure 1.11 montre un exemple de deux stratégies équivalentes de fonctionnement de la batterie. Les deux solutions ont le même impact économique à la fin de la journée (il s'agit de minimiser le facteur énergétique sur 24h). Pratiquement, une solution est meilleure que l'autre du point de vue de la batterie car elle comporte des variations de charge/décharge de la batterie moins rapides (en rouge).

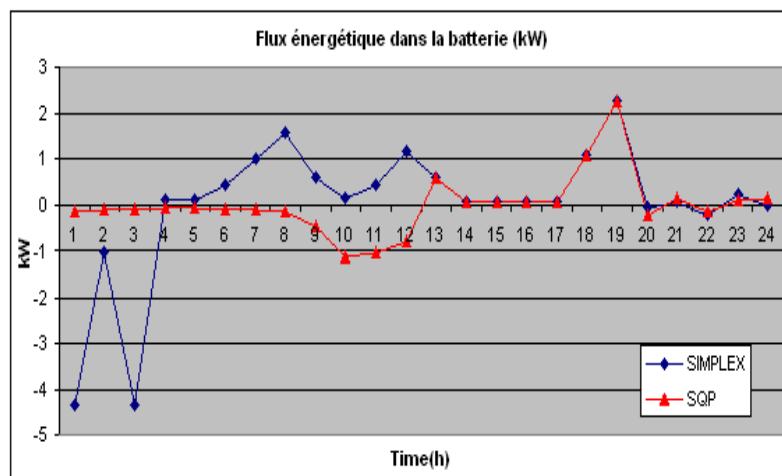


FIGURE 1.11 – Deux fonctionnements équivalents de la batterie dans un système multi-sources

Il s'agit donc d'ajouter un niveau d'abstraction supplémentaire pour formaliser le pro-

5. Programmation Linéaire en Nombres Entiers

blème d'optimisation. Avec cette approche, nous espérons être capable de :

1. générer dynamiquement les problèmes énergétiques du bâtiment intelligent sous forme de problème d'optimisation.
2. résoudre le problème énergétique du bâtiment intelligent dans des environnements d'optimisation différents.

1.7 Conclusion

Dans ce chapitre nous avons illustré les nouvelles problématiques introduisant par le concept de bâtiment intelligent tant au niveau énergétique qu'économique. Nous avons présenté la définition adoptée dans cette thèse de bâtiment intelligent.

Pour atteindre les objectifs du Grenelle de l'environnement, ce bâtiment aura plusieurs sources d'énergie et un plus grand niveau d'intelligence pour la gestion énergétique. Nous avons présenté l'état de l'art sur le dimensionnement et la gestion des sources électriques reliées au bâtiment.

Chaque bâtiment est un système unique et en évolution permanente. Nous avons regroupé les facteurs d'influence sous quatre catégories différentes : les habitudes des résidents, le marché, la météo.etc. Nous avons présenté quelques approches qui ont étudié la gestion d'énergie dans le bâtiment sans envisager la génération automatique de problèmes. Cela répond mal aux nouvelles problématiques évoquées.

Dans cette thèse, nous allons présenter une approche générique pour la construction automatique de problèmes d'optimisation, concernant aussi bien le dimensionnement que la gestion optimisée des systèmes bâtiments. Cette approche est fondée sur le concept de l'ingénierie dirigée par les modèles, qui propose des patrons de résolution afin d'accroître la générericité de formulation et de permettre la projection vers différents environnements d'optimisation.

Une synthèse des chapitres du manuscrit est présentée sur la figure 1.12.

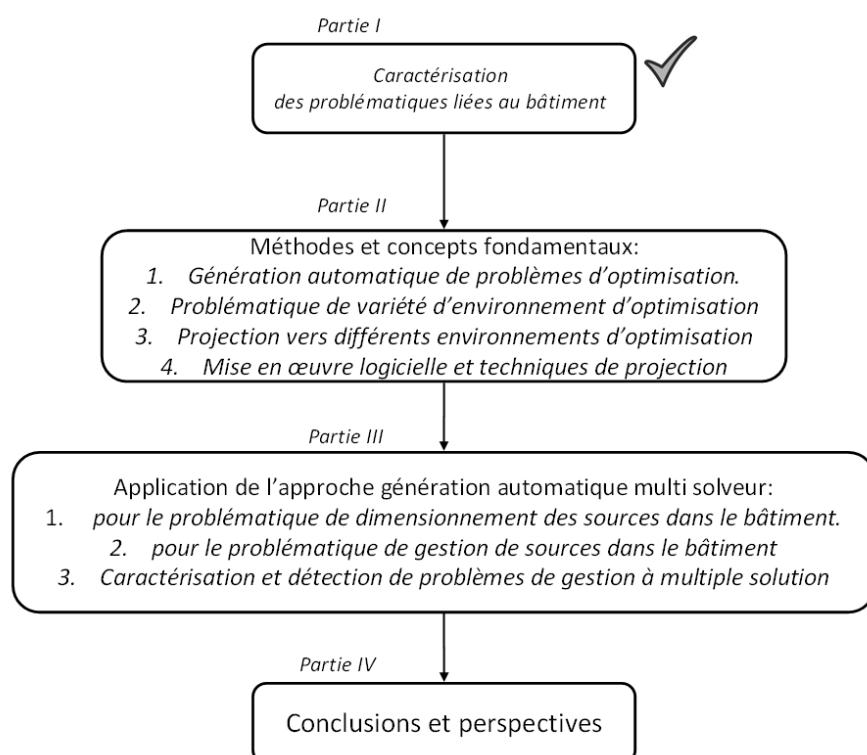


FIGURE 1.12 – Synthèse du manuscrit de la thèse

Deuxième partie

Méthodes et concepts fondamentaux pour résoudre les problèmes d'optimisation dans le Bâtiment

Chapitre 2

Génération automatique de problèmes d'optimisation

Il suffit de regarder une chose avec attention pour qu'elle devienne intéressante.

Eugenio D'Ors. Espagne

SOMMAIRE

2.1	INTRODUCTION	28
2.2	NOTION DE TYPES PRÉDÉFINIS DE SOUS-PROBLÈMES D'OPTIMISATION	29
2.3	NOTIONS DE PORTS ET DE CONTRAINTES GLOBALES	29
2.4	NOTION DE CONCEPTEUR GRAPHIQUE	31
2.5	CONCLUSION	31

Résumé

Les bâtiments sont caractérisés par une grande diversité tant au niveau de l'environnement, des équipements que des occupants. La génération manuelle de problèmes d'optimisation est une tâche consommatrice de temps qu'il est préférable d'éviter, surtout si le concepteur de bâtiments veut comparer plusieurs scénarios d'usage différents. Dans ce chapitre, une proposition de génération automatique de problèmes d'optimisation est présentée. Nous présentons certaines notions qui nous ont aidé à implémenter ce générateur automatique.

2.1 Introduction

Prenons l'exemple d'un circuit électrique composé d'une résistance et d'une source de tension. Nous voulons maximiser le courant électrique (voir problème 2.1). La démarche classique pour générer un problème d'optimisation correspondant consiste à écrire les contraintes sur le courant, sur la résistance et sur la source de tension en une seule étape, puis de faire appel à un solveur pour résoudre le problème (cf. figure 2.1).

$$\begin{aligned}
 & \text{Max} && i \\
 & C_1 : U = R_1 \times i \\
 & C_2 : 0 \leq R_1 \leq 5 \\
 & C_3 : 0 \leq U \leq 4
 \end{aligned} \tag{2.1}$$

Malgré le simplicité du problème d'optimisation, le changement de topologie du circuit obtenu en ajoutant une autre résistance, induit une reformulation du problème : il faut ajouter de nouvelles contraintes et modifier les anciennes (voir problème 2.2).

$$\begin{aligned}
 & \text{Max} && i \\
 & C_1 : U = R_1 \times i_1 \\
 & C_2 : 0 \leq R_1 \leq 5 \\
 & C_3 : 0 \leq U \leq 4 \\
 & C_4 : i = i_1 + i_2 \\
 & C_5 : 0 \leq R_2 \leq 3 \\
 & C_6 : U = R_2 \times i_2
 \end{aligned} \tag{2.2}$$

Si le travail d'adaptation n'est pas considérable pour ce problème, il n'en va pas de même pour l'application bâtiment où chaque source peut contenir des dizaines de contraintes. Le travail de reformulation du problème devient vite considérable. Comment minimiser le travail d'adaptation induit par chaque nouvelle configuration envisagée, par chaque nouveau contexte ?

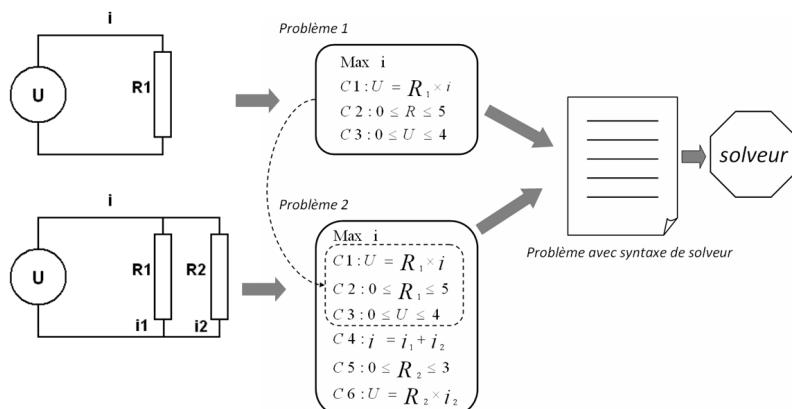


FIGURE 2.1 – Processus de résolution usuelle

2.2 Notion de types prédéfinis de sous-problèmes d'optimisation

La solution passe par la génération de sous-problèmes d'optimisation où chaque équipement est représenté par un ensemble de contraintes contribuant au problème global. Ces contraintes sont propres à un composant. Elles le décrivent de façon autonome comme C_1 et C_2 dans le problème 2.2. Nous allons les appeler contraintes locales.

Chaque composant apportera donc ses contraintes locales au problème d'optimisation globale. Cela signifie que le problème d'optimisation commence par une sorte de problème vierge où chaque équipement inscrit ses contraintes. Le problème d'optimisation est alors finalisé en ajoutant les contraintes globales, généralement ayant trait aux équilibres énergétiques ou économiques comme nous allons le voir dans la partie 3.

Le concepteur de problème d'optimisation décrit les constituants d'un système via une interface graphique (appelée concepteur) pour que le problème d'optimisation complet soit généré automatiquement à la fin. La figure 2.2 illustre le principe de génération automatique proposé.

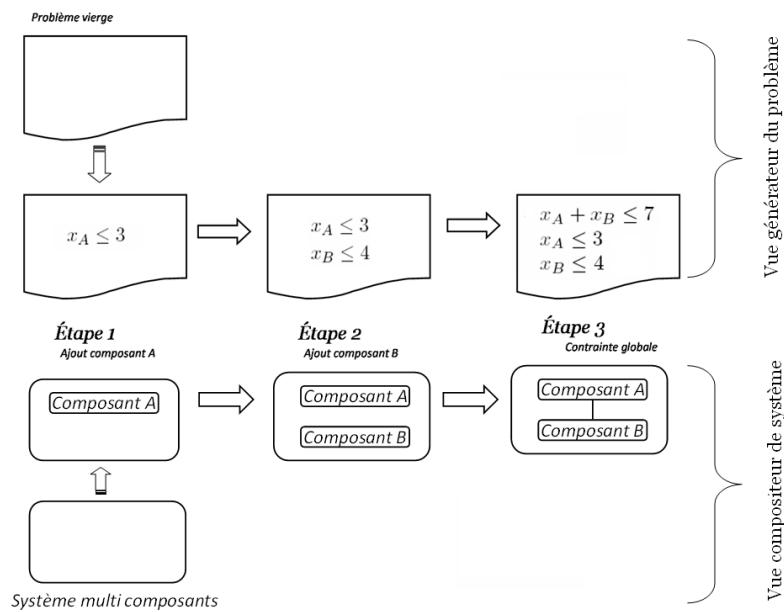


FIGURE 2.2 – Principe de composition automatique des problèmes d'optimisation

Pour implémenter cette approche dans l'application bâtiment, il faut faire les modèles des composants du bâtiment. Dans ces modèles, on trouve les contraintes décrivant le fonctionnement de ces composants (ou équipements physiques) sous forme de sous problèmes d'optimisation. Nous allons présenter ces modèles dans les chapitres 7 et 8.

2.3 Notions de ports et de contraintes globales

Pour compléter le problème d'optimisation et connecter les sous-problèmes, il faut ajouter des contraintes globales. Ces contraintes contiennent des variables communes avec celles de sous-problèmes. Ces variables sont appelées des ports de modèles de composants grâce

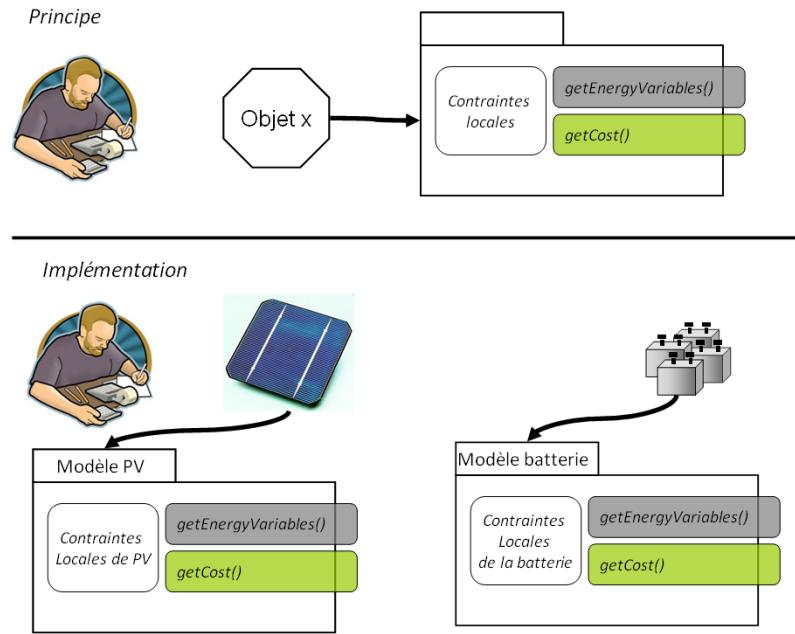


FIGURE 2.3 – Conception de sous problèmes selon l'approche proposée

auxquelles la génération peut être automatisée (les rectangles vertes et grise dans la figure 2.3). Ainsi, pour générer le problème complet, on peut utiliser un générateur automatique de problèmes qui récupère ces variables dans les modèles de composants, et les ajoute aux contraintes globales (cf. figure 2.4).

Dans le cas de l'application bâtiment, nous proposons la classification suivante :

1. ports énergétiques : Ce sont les variables qui participent au bilan énergétique du système complet du bâtiment. Elles représentent les flux énergétiques alloués à un composant. Par exemple, l'énergie de décharge d'une batterie.
2. ports économiques : Ce sont les variables qui participent au bilan économique comme par exemple le coût initial d'une installation PV.

Quelque soit le problème d'optimisation issu du bâtiment, nous prédéfinissons ces deux bilans comme des contraintes globales et on les ajoute à la formulation du problème¹.

Les variables d'optimisation de chaque composant sont i_1 et i_2 . Supposons qu'il faille maximiser la somme des courants i . Les contraintes locales du composant R_1 sont C_1 , C_2 et C_3 , les ports de R_1 sont i_1 et U . Pour le composant R_2 , nous avons les contraintes locales C_5 et C_6 , les ports de ce composant sont i_2 et U . Le bilan énergétique dans ce cas est la contrainte C_4 .

Première configuration : un seul composant R_1

Suivant la démarche proposée, au moment où l'utilisateur ajoute le composant R_1 au circuit, le composant R_1 ajoute ses contraintes locales au problème d'optimisation. Ensuite, le générateur automatique va ajouter le port i_1 dans le bilan énergétique prédéfini C_4 . La génération le problème 2.1 est dans ce cas terminée.

1. Nous préparons deux type de connecteurs dans la formulation de problème dans lesquels le générateur ajoutera les variables correspondantes.

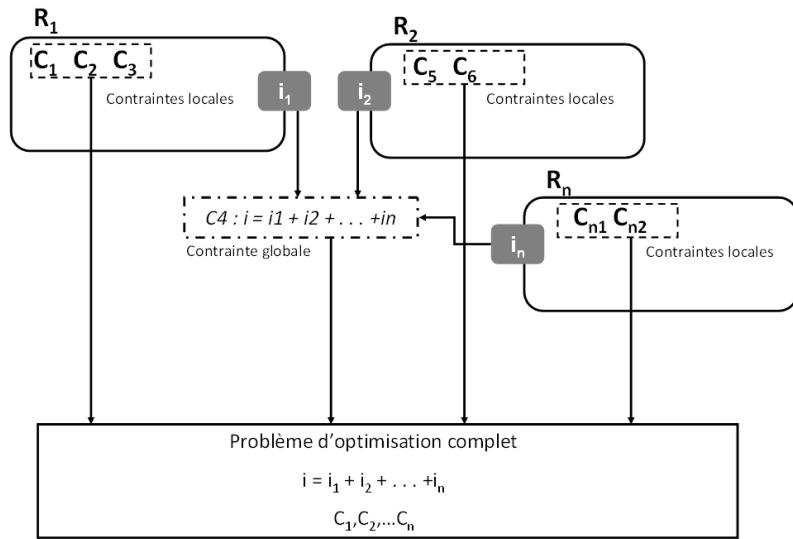


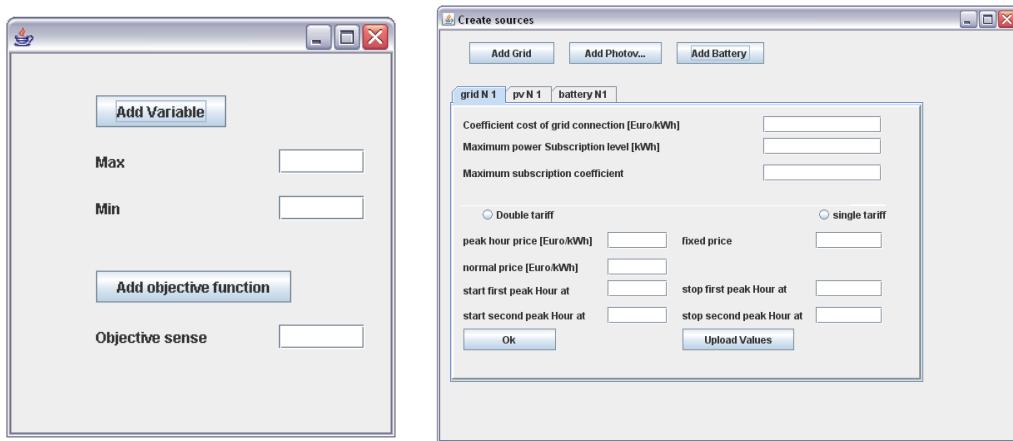
FIGURE 2.4 – Notion de port d'un modèle de composant, Par exemple, dans le problème 2.2, les variables i_1 et i_2 et U sont des ports des modèles de R_1 et R_2 .

Deuxième configuration : deux composants R_1, R_2

Si l'utilisateur veut changer la configuration, il suffit qu'il ajoute au circuit le composant R_2 (par exemple via une interface graphique) pour que le générateur reformule le problème : les contraintes locales de R_2 s'ajoutent automatiquement, puis le générateur cherche les ports de R_2 et les ajoute sur le bilan C_4 .

2.4 Notion de concepteur graphique

La quantité et le type d'informations données par l'utilisateur peuvent varier notamment en fonction de son savoir faire sur la formulation de problèmes. Un concepteur graphique est une interface graphique permettant de décrire un composant (un sous problème d'optimisation) dans le système. Il est possible de mettre au point plusieurs concepteurs graphiques pour générer un même problème d'optimisation. Dans ce cas, le générateur peut être appelé générateur 'multi concepteur'. Par exemple, la figure 2.5 montre deux exemples de concepteurs graphiques. A gauche, il y a un concepteur de modèles pour générer les modèles, et à droite, un concepteur de système bâtiment pour implémenter les modèles existants.



(a) Une interface graphique pour un concepteur de problème

(b) Une interface graphique pour un utilisateur d'application conçu

FIGURE 2.5 – Deux types de concepteurs graphiques, à gauche un concepteur pour générer les modèles, et à droite, un concepteur pour implémenter les modèles existants

2.5 Conclusion

Pour automatiser la génération de problèmes de conception de systèmes bâtiments (dimensionnement ou gestion des sources), nous avons proposé que chaque composant ou équipement physique du bâtiment soit modélisé comme un sous problème d'un problème de conception globale. Dans ce cas, la personne qui modélise l'équipement écrit les contraintes locales des composants et la déclaration de ports économiques et énergétiques. Ces ports sont des variables des composants qui participent aux contraintes globales comme le bilan énergétique et le bilan économique. En conséquence, le générateur de problème (un automate) construit le problème global par assemblage progressif des sous-problèmes ; puis pour compléter ce problème il va chercher les ports des modèles puis il les ajoute dans les contraintes globales (cf. figure 2.6).

Selon le type de concepteur de systèmes de bâtiment, la première étape de génération automatique d'un problème diffère.

1. Pour le développeur d'application, il s'agit de construire les types de modèles de composants, mais sans instancier les paramètres, susceptibles d'apparaître dans le système à étudier, comme par exemple des types associés aux équipements électroménagers blancs ou bruns, aux zones thermiques dans un appartement avec leur système de chauffage, aux installations photovoltaïques, etc... .
2. Pour l'utilisateur d'application, il s'agit d'utiliser les types de modèles nécessaires en fixant les valeurs des paramètres puis en générant le problème d'optimisation globale du système en profitant de l'approche par construction progressive proposée.

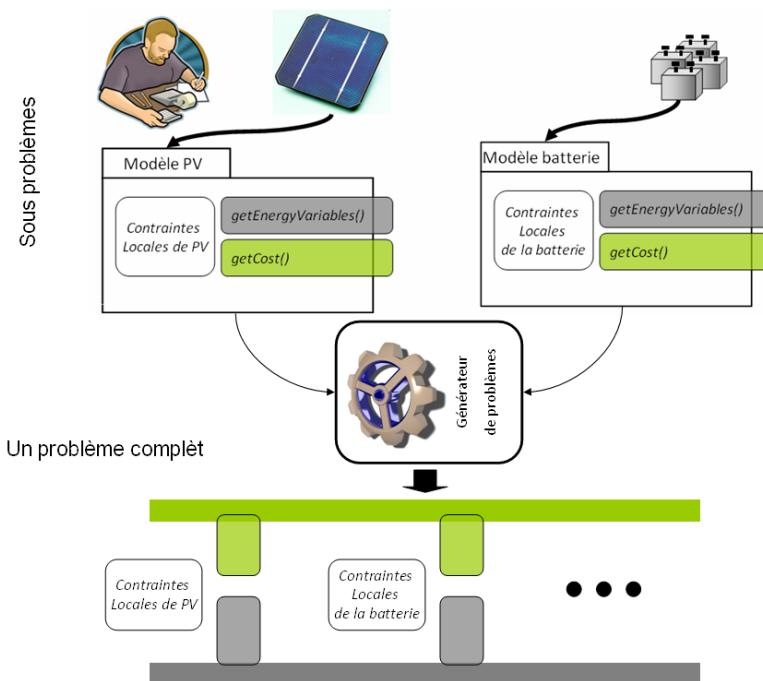


FIGURE 2.6 – Démarche de génération automatique proposée

Nous allons montrer dans les chapitres 7 et 8 comment nous avons implémenté cette approche pour le dimensionnement et la gestion de sources électriques dans le bâtiment.

Chapitre 3

Problématique de multiples environnements d'optimisation

*Le commencement de toutes les sciences, c'est
l'étonnement de ce que les choses sont ce qu'elles sont*

Aristote

SOMMAIRE

3.1	INTRODUCTION	34
3.2	ALGORITHMES D'OPTIMISATION CONSIDÉRÉS	34
3.3	ENVIRONNEMENTS D'OPTIMISATION CIBLES	34
3.3.1	Fonctionnement de CPLEX	35
3.3.2	Fonctionnement de CADES	37
3.4	CONCLUSION	40

Résumé

Les problèmes d'optimisation générés précédemment peuvent être formulés comme des problèmes linéaires ou non linéaires. Cela induit le besoin d'environnements d'optimisations adaptés. Certains environnements d'optimisation ont plus de capacité à résoudre les problèmes pour les systèmes bâtiment que d'autres. Dans ce chapitre, nous présentons deux algorithmes d'optimisation et deux environnements d'optimisation qui implémentent ces algorithmes. Dans cette thèse, nous allons nous en servir pour résoudre les problèmes de dimensionnement et de gestion de bâtiments.

3.1 Introduction

En fonction des problématiques énergétiques rencontrées dans le bâtiment, des environnements d'optimisation reposant sur des algorithmes de programmation linéaire mixte PLNE (ou Mixed Integer Linear Programming) (Philip et al. 1981) ou de programmation non linéaire PNL, de type programmation séquentielle quadratique (ou Sequential Quadratic Programming) (Powell 1985), peuvent être utilisés. En conséquence, les problèmes générés par l'approche proposée dans le chapitre précédent peuvent être résolus dans deux environnements d'optimisation différents. Nous examinerons en particulier les environnements d'optimisation CPLEX (PLNE) et CADES (SQP).

3.2 Algorithmes d'optimisation considérés

La PLNE est retenue comme une méthode pour formuler le problème de conception et de gestion de l'énergie dans les applications bâtiment parce qu'elle permet de résoudre des problèmes de très grande dimension, mais requiert parfois un travail de linéarisation (Long 2007).

CPLEX (ILOG 2010) est un outil de type PLNE très utilisé par la communauté scientifique pour sa puissance de calcul. Il combine différents algorithmes dont celui du simplexe et une procédure de séparation évaluation (Branch&Bound) (Chinneck 2010). Le SIMPLEX est un algorithme de résolution qui travaille sur la frontière de la région de faisabilité (voir la figure 12.2), raison pour laquelle cet algorithme est classé dans la catégorie des *Active set method*. Il cherche la solution optimale à partir d'une base de solutions qui active certaines contraintes (nous allons détailler cela dans la suite).

CADES (Delinchant et al. 2007) résout les problèmes non-linéaires à variables continues. Un algorithme de programmation quadratique successive (SQP) (Powell 1985) est utilisé. Nous avons choisi cet environnement car nous voulons être capables de résoudre des problèmes continus non linéaires.

Nous présentons les détails de ces deux algorithmes dans l'annexe 12.2.

3.3 Environnements d'optimisation cibles

Admettons qu'on ait le problème d'optimisation suivant :

$$\begin{aligned}
 \text{Min } z = & 4x_1 + 5x_2 \\
 c1 : & 2x_1 + x_2 \leq a \\
 c2 : & x_2 \leq b \\
 c3 : & x_1 + 2x_2 \leq c \\
 c4 : & 2x_1 + 3x_2 = d \\
 & x_1 \leq 10
 \end{aligned} \tag{3.1}$$

Sachant que :

– $a = 8$, $b = 3$, $c = 7$ et $d = 5$.

Nous allons présenter le fonctionnement de CADES et CPLEX à partir de cet exemple démonstratif.

3.3.1 Fonctionnement de CPLEX

CPLEX est un logiciel développé par ILOG¹ ([ILOG 2010](#)), permettant de résoudre des problèmes linéaires de type PLNE et quadratique. Récemment, un nouveau support a été ajouté pour résoudre des problèmes avec des contraintes convexes quadratiques. L'utilisateur peut choisir entre une formulation en langage mathématique textuel (Algebraic Mathematical Language comme OPL ou AMPL) ou utiliser l'interface de programmation (*Application Programming Interface*) pour appeler CPLEX via Java ou C++.

L'interface de programmation est un ensemble de fonctions, classes et procédures qui sont mises à disposition par une bibliothèque logicielle. Elle permet l'interaction des programmes les uns avec les autres.

Concrètement, nous pouvons travailler avec CPLEX de trois manières différentes :

1. via l'optimiseur interactif (*ILOG CPLEX Interactive Optimizer*). Il s'agit d'un exécutable qui peut lire le problème d'optimisation, soit en mode interactif, soit en lisant un fichier texte contenant un code AML au format standardisé. L'optimiseur interactif appelle alors le solveur pour lire ce fichier et résoudre le problème en écrivant la solution dans un fichier résultat.
2. via *ILOG Concert Technology*. Il s'agit d'un package logiciel qui offre une API avec des librairies pour faciliter l'intégration de CPLEX dans un programme en C++, Java ou C# (Pour Windows ce package contient : ilocplex.lib, concert.lib, et cplex.jar et ILOG.CPLEX.dll).
3. via *ILOG CPLEX Callable Library*. Il s'agit d'une librairie en C pour que l'utilisateur puisse intégrer l'optimiseur dans toutes les applications écrites en C ou dans les langages qui peuvent appeler des fonctions en C. La librairie est fournie sous la forme d'un fichier *Dynamic-Link Library*.

Nous allons travailler avec *ILOG Concert Technology* que nous nommerons *ILOCT*. L'interaction entre *ILOCT* et le programme de l'utilisateur est résumé dans la figure 3.1.

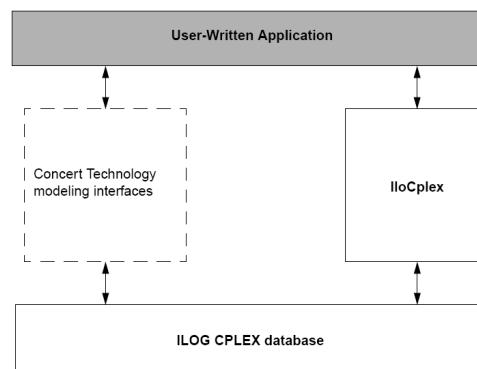


FIGURE 3.1 – ILOG CPLEX Concert Technology pour les développeurs Java

Pour résoudre un problème d'optimisation, il faut un moyen pour modéliser les variables, les contraintes et la fonction objectif d'un problème d'optimisation comme des

1. ILOG est une entreprise française, éditeur de logiciels de gestion. L'entreprise a été rachetée par IBM en 2009

objets de IloCplex. Pour cela, il faut utiliser l'interface de *ILOCT* qui est implémentée dans IloCplex, car ceci donne l'accès aux objets ILOG CPLEX situés dans CPLEX. En d'autres termes, les éléments d'un problème d'optimisation restent dans CPLEX pour que le solveur puisse les utiliser mais leur accès est géré par l'interface *ILOCT*.

Pour résoudre le problème 3.1 avec *ILOCT*, il faut créer une classe Java pour construire l'objet IloCplex. Cet objet conduit au fichier *demo.lp* (voir figure 3.4). La figure 3.2 montre les lignes de codes qui permettent la génération de 3.1 au format IloCplex pour l'exporter sous un format AML *demo.lp* avant de résoudre le problème. La figure 3.3 montre les traces de résolution par appel de CPLEX en ligne de commande.

```
// créer un object IloCplex
IloCplex iloCplexObject = new IloCplex();
// Ajouter les variables
IloNumVar x1 = iloCplexObject.numVar(0, 10, "x1");
IloNumVar x2 = iloCplexObject.numVar(0, 3, "x2");
// Ajouter les contraintes
IloNumExpr expr_C1 = iloCplexObject.sum(iloCplexObject.prod(2, x1),
                                         iloCplexObject.prod(2, x2));
IloNumExpr expr_C3 = iloCplexObject.sum(iloCplexObject.prod(1, x1),
                                         iloCplexObject.prod(2, x2));
IloNumExpr expr_C4 = iloCplexObject.sum(iloCplexObject.prod(2, x1),
                                         iloCplexObject.prod(3, x2));

// créer l'expression de la fonction objectif
IloNumExpr expr_Objective = iloCplexObject.sum(iloCplexObject.prod(4, x1),
                                                iloCplexObject.prod(5, x2));

// Ajouter la fonction objectif
IloObjective objective = iloCplexObject.addMaximize(expr_Objective, "Z");

// mettre les valeurs numérique pour les contraintes
iloCplexObject.addLe(expr_C1, 8);
iloCplexObject.addLe(expr_C3, 7);
iloCplexObject.addEq(expr_C4, 5);

// générer le problème en fichier *.lp
iloCplexObject.exportModel(this.profileName+" .lp");

// appeler le solver et résoudre le problème
iloCplexObject.solve();
```

FIGURE 3.2 – Codes pour générer le problème d'optimisation comme un objet de ILOCT en Java

```
C:\ILOG\CPLEX101\bin\x86_win32\cplex.exe
Welcome to CPLEX Interactive Optimizer 10.1.1
with Simplex Mixed Integer & Barrier Optimizers
Copyright © ILOG S.A. 2001
CPLEX is a registered trademark of ILOG

Type 'help' for a list of available commands.
Type 'help' followed by a command name for more
information on commands.

CPLEX> read demo.lp
Problem 'demo.lp' read.
Model read in 0.00 sec.
CPLEX> optimize
Tried aggregator 1 time.
No LP presolve or aggregator reductions.
Presolve time = -0.00 sec.

Iteration log :
Iteration: 1 Dual objective      =      20.000000
Dual simplex - Optimal: Objective = 1.999999999e+001
Solution time = 0.00 sec. Iterations = 2 <0>
CPLEX>
```

FIGURE 3.3 – Appel de CPLEX en interactif pour la résolution de *demo.lp*

```

\Problem name: demo.lp

Maximize
  obj: 4 x1 + 5 x2
Subject To
  IloC0: 2 x1 + 2 x2 <= 8
  IloC1: x1 + 2 x2 <= 7
  IloC2: 2 x1 + 3 x2 = 5
Bounds
  0 <= x1 <= 10
  0 <= x2 <= 3
End

```

FIGURE 3.4 – Modèle exploitable du problème 3.1 pour CPLEX

3.3.2 Fonctionnement de CADES

CADES est un environnement d'optimisation multi-phérique développé au sein de G2Elab ([Delinchant et al. 2007](#)). Il est composé de trois modules ; chacun est dédié à une tâche précise. L'esprit de CADES est la modélisation systémique avec une causalité explicite et toujours définie (quel que soit l'objet à modéliser, CADES va l'interpréter comme un modèle orienté avec ses entrées et ses sorties, ce qui n'était pas le cas de CPLEX). CADES est aussi capable d'implémenter des méta-heuristiques d'optimisation pour la résolution, comme des algorithmes génétiques ou stochastiques.

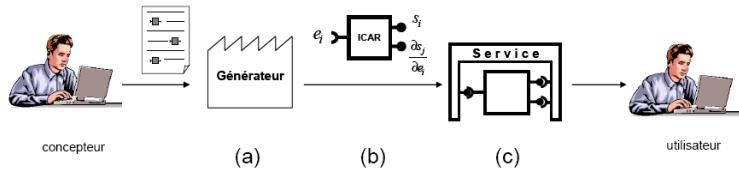


FIGURE 3.5 – Démarche d'optimisation avec CADES : (a) décrire le modèle analytique 'boîte blanche' (b) générer le composant icar 'boîte noire' (c) appeler un algorithme d'optimisation et fournir la réponse

Pour résoudre le problème 3.1 avec CADES, les trois modules à utiliser sont :

- *CADES Generator*. Ce module interprète le modèle analytique décrit dans un langage AML nommé SML². La figure 12.1(a) montre comment on écrit le modèle analytique du problème 3.1. Le Génératuer génère alors un composant de calcul. Ce composant doit respecter une norme de composant logiciel nommée ICAR³ ([Fischer 2004](#)). Une copie d'écran de ce module est illustrée dans la figure 3.6.
- *CADES calculator*. Est un module de calcul qui permet d'étudier le comportement du système modélisé par le composant ICAR. *CADES calculator* détermine un ordonnancement de résolution optimale et en déduit les variables de décision qu'il prend comme entrées (voir ([Allain 2003](#))) (pour l'exemple 3.1, x_1 et x_2 sont les entrées,

2. pour *System Modeling Language*

3. pour *Interface Component Architecture*

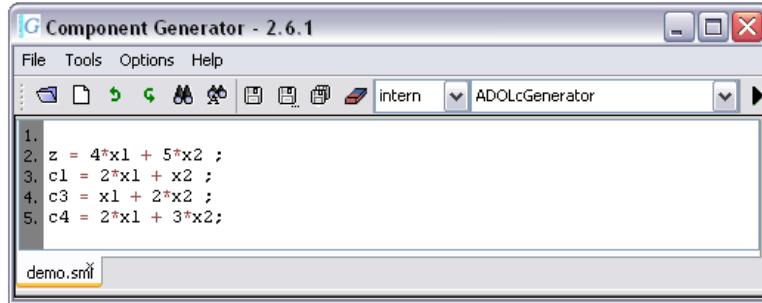


FIGURE 3.6 – Une copie d'écran du générateur CADES pour le modèle analytique de *demo.sml*

z , c_1 , c_3 et c_4 sont les sorties). Grâce à l'ordonancement causal, *CADES calculator* permet de valider le modèle. Lors de la résolution, le gradient de toutes les sorties en fonction de toutes les entrées est généré et *CADES calculator* permet de calculer et d'afficher le Jacobien correspondant. Ce Jacobien est utilisé par l'algorithme SQP embarqué (une copie d'écran de *CADES calculator* est montrée dans la figure 3.7).

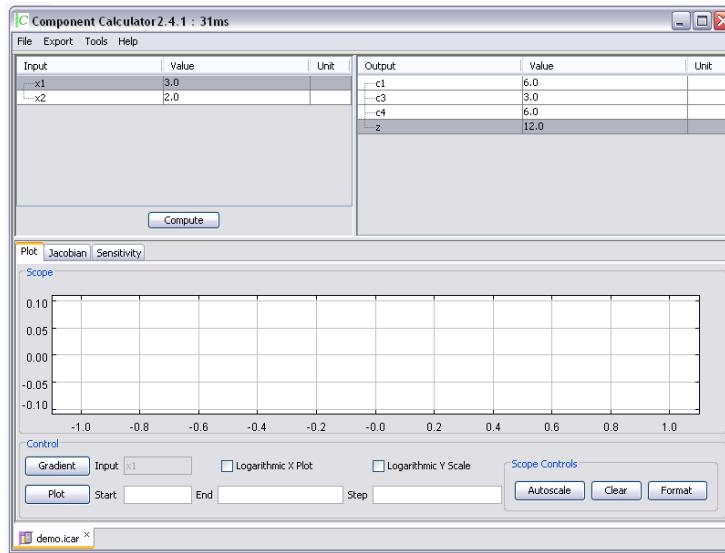


FIGURE 3.7 – Une copie d'écran de la calculette CADES

- *CADES optimizer*. Ce module implémente un algorithme d'optimisation pour résoudre le composant *.icar généré précédemment. Il faut préalablement définir le domaine de variation des variables directement sur l'interface graphique (figure 3.8), cela va permettre de générer un fichier de spécification au format *xml* (voir la figure 12.1(b) de l'annexe 12.1). On déclare finalement la variable à minimiser ou à maximiser via la même interface graphique (la fonction objectif représentée par la variable z ici). La solution s'affiche dans le module *CADES optimizer* avec la possibilité de tracer les valeurs d'optimisation obtenues à chaque itération (voir figure 3.9).

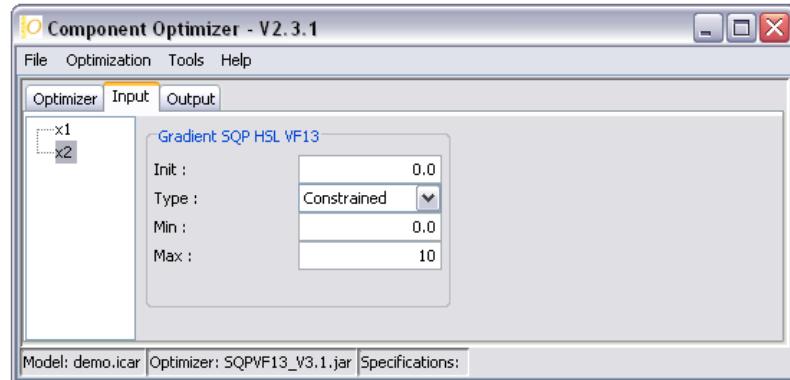


FIGURE 3.8 – Une copie d'écran de l'optimiseur CADES, ajout d'une spécification (contrainte c_2) pour la variable x_2

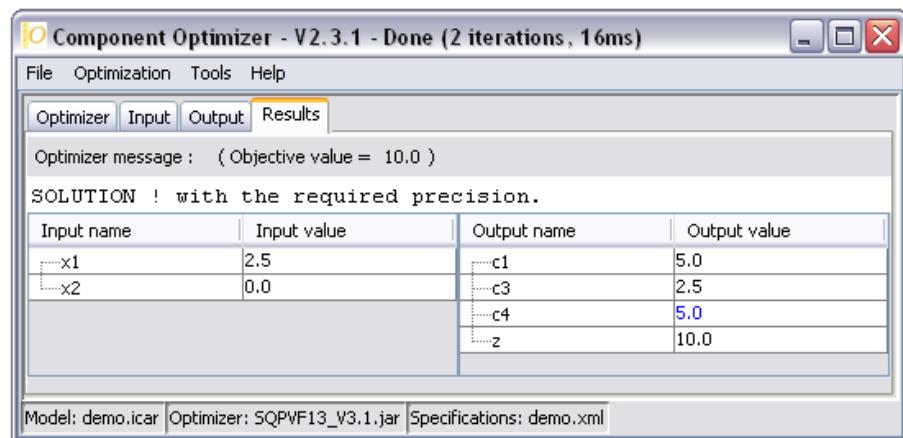


FIGURE 3.9 – Une copie d'écran de l'optimiseur CADES après la résolution

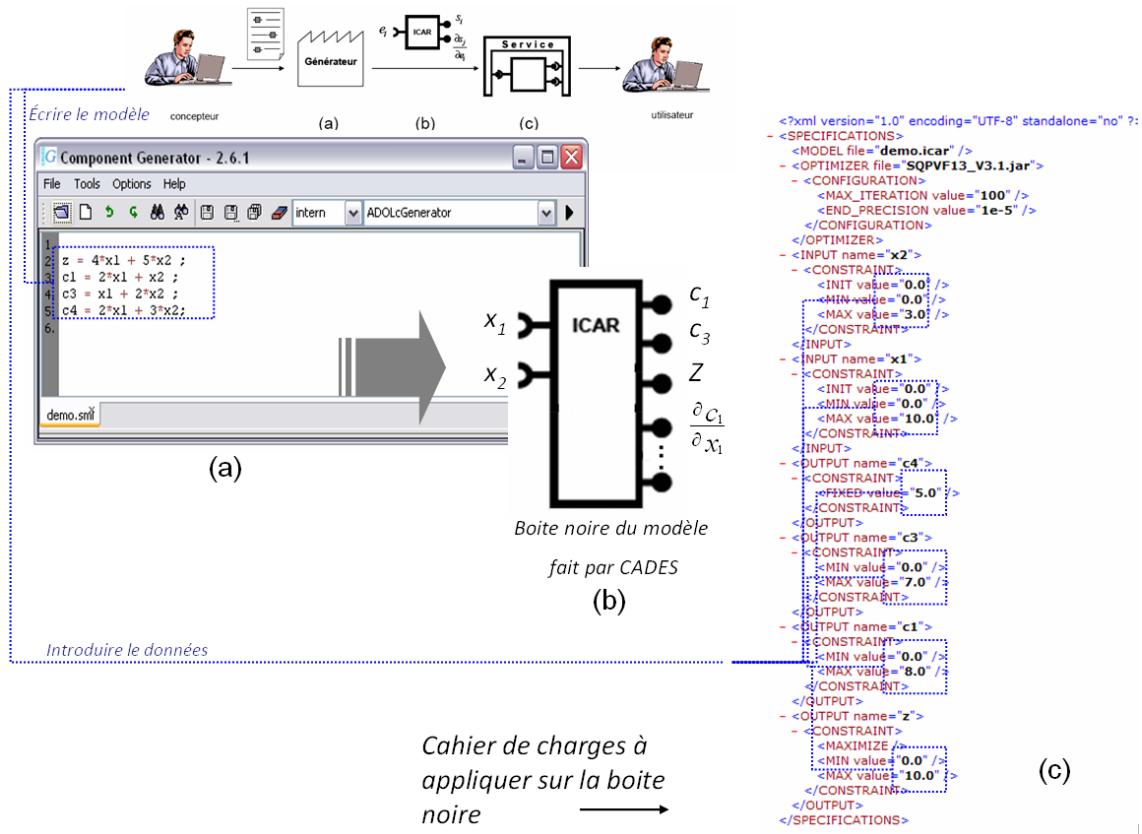


FIGURE 3.10 – Aperçu de fonctionnement de CADES, modèle analytique du problème 3.1 en sml et spécification en xml

Selon la figure 3.10, nous constatons que :

1. Dans le modèle décrit en sml, on ne voit pas la contrainte c_2 car en CADES, les contraintes qui sont composées d'une seule variable à l'instar de c_2 , sont ajoutées directement dans le fichier de spécifications.
2. Dans le fichier de spécifications (cf figure 3.10 partie C), les paramètres du problème 3.1 $a = 8$, $b = 3$ et $c = 7$ ont été remplacés par leurs valeurs numériques directement. En effet, la notion de variable dans le fichier de spécifications n'existe pas. L'utilisateur doit saisir directement les valeurs numériques.

Les caractéristiques des modèles résultant d'une projection vers CADES correspondent aux paramètres apparaissant dans le modèle analytique *.sml et dans les spécifications sur les variables qui sont dans le fichier *.xml.

3.4 Conclusion

Les problèmes d'optimisation liés au bâtiment peuvent être formulés avec plusieurs algorithmes puis résolus dans différents environnements d'optimisation. Dans ce chapitre, les environnements CPLEX et CADES ont été pris en compte comme deux environnements potentiels de résolution. Le fait d'avoir une possibilité de multiples environnements d'optimisation induit une autre problématique : il s'agit d'automatiser la génération de problèmes liés au bâtiment pour chaque environnement. Ceci est l'objet du chapitre suivant.

Chapitre 4

Projection vers différents environnements d'optimisation

Le métier évolue moins vite que la technique

SOMMAIRE

4.1	PROJECTION AUTOMATIQUE VERS DES ENVIRONNEMENTS DE RÉSOLUTION MULTIPLES : LES CONCEPTS ISSUS DE L'INGÉNIERIE DIRIGÉE PAR LES MODÈLES	42
4.2	DÉFINITION DE L'INGÉNIERIE DIRIGÉE PAR LES MODÈLES	42
4.3	ADAPTATION DES CONCEPTS ISSUS DE L'IDM AU CONTEXTE DU BÂTIMENT	45
4.4	PROJECTION DES PROBLÈMES COMME UNE PROCÉDURE DE TRANSFORMATION DES MODÈLES	48
4.4.1	Principes	48
4.4.2	Transformation utilisée dans le contexte du bâtiment	49
4.5	GRANDES LIGNES DE L'APPROCHE DE GÉNÉRATION 'MULTI SOLVEUR' DE PROBLÈMES D'OPTIMISATION APPLIQUÉS AU CONTEXTE DU BÂTIMENT INTELLIGENT	50
4.6	CONCLUSION	53

Résumé

Comme les problèmes d'optimisation peuvent être résolus dans différents environnements (CPLEX et CADES), l'automatisation de la génération de problèmes dans des formats spécifiques à chaque environnement est utile. Or, profiter de l'approche de génération précédente pour générer un seul format indépendant de tout environnement, puis automatiser sa transformation vers des formats spécifiques aux différents environnements offre de nouvelles perspectives à la résolution de problèmes. Dans ce chapitre, une solution pour automatiser la transformation des problèmes générés automatiquement avec l'approche du chapitre 2 est illustrée. La solution vient de concepts issus du domaine du génie logiciel, il s'agit de l'ingénierie dirigée par les modèles (IDM).

4.1 Projection automatique vers des environnements de résolution multiples : Les concepts issus de l'ingénierie dirigée par les modèles

Reprendons l'exemple du circuit électrique dont on rencontre une analogie dans la modélisation thermique équivalente d'une zone. Si on ajoute une résistance non linéaire ([Cours 2003](#)) en parallèle avec la résistance existante, cela va introduire de nouvelles contraintes non-linéaires. Dans le cas où un solveur linéaire a été utilisé jusque là pour bénéficier de leur grandes performances, il ne peut plus l'être et il faudrait reformuler le problème pour un solveur non linéaire.

Admettons que l'on ait deux solveurs : un solveur S_1 pour les problèmes linéaires et un solveur S_2 pour certains problèmes non-linéaires. Nous pouvons développer des projecteurs qui permettent de transformer le problème global décrit génériquement vers des syntaxes compréhensibles soit par S_1 , soit par S_2 . S'il s'avère que S_2 ne convient pas, la même opération peut-être effectuée vers un solveur S_n . Mais comment pouvons nous éviter de reformuler le même problème d'optimisation pour chaque solveur ?

L'application du concept de métamodélisation issu de l>IDM permet de construire des outils de résolution capables de s'adapter à de nombreuses problématiques ([OMG 2010](#)). Elle fournit les outils d'abstraction nécessaires à la résolution de problèmes dans différents environnements d'optimisation.

Les systèmes informatiques évoluent fréquemment : de nouvelles plateformes apparaissent, d'autres disparaissent, certaines font évoluer la syntaxe des codes interprétés. Pour pouvoir suivre ces évolutions, les entreprises développant des logiciels ont adopté une approche fondée sur la séparation des préoccupations. Il s'agit de séparer la conception d'un objet (modèle) de son implémentation. Quelle que soit la plateforme cible, la conception d'objets constituant une partie du savoir-faire n'est pas remise en cause ([Bézivin 2003](#)) et ([Atkinson1 et al. 2004](#)).

Revenons à la problématique énergétique dans le bâtiment. L'implémentation des concepts issus de l'ingénierie dirigée par les modèles offrirait l'avantage de faciliter la réutilisation des modèles en séparant clairement les tâches de conception des types de modèles, des tâches de projection vers différents environnements de résolution. Notre étude est un travail d'adaptation au bâtiment des concepts issus de l'ingénierie dirigée par les modèles qui s'appliquent à d'autres domaines (voir figure 4.1).

Dans les paragraphes suivants, nous présentons certains principes de l>IDM utilisés dans cette thèse.

4.2 Définition de l'ingénierie dirigée par les modèles IDM

L'ingénierie dirigée par les modèles est un ensemble de concepts et outils pour construire, manipuler et transformer les modèles. Les principaux concepts de l>IDM ont été introduits en 1999 par l'*Object Management Group* ([OMG 2010](#))¹ pour réduire les coûts de production des logiciels en favorisant la réutilisation des modèles et en augmentant leur flexibilité

1. OMG est une association à but non-lucratif dont l'objectif est d'établir des standards permettant de résoudre les problèmes d'interopérabilité des systèmes d'information (voir figure 18.1)

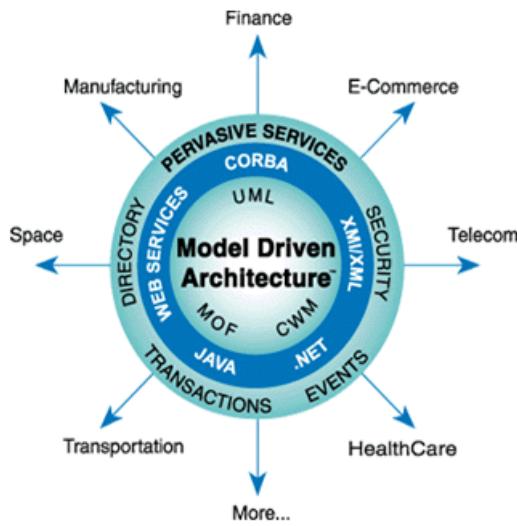


FIGURE 4.1 – Les domaines d’application de l’ingénierie dirigée par les modèles (Miller & Mukerji n.d.)

face aux nouvelles technologies. Depuis 2002 jusqu’à 2007, 50% des outils de programmation orienté objet (*OO modeling*) ont implémentés certains concepts de l’IDM (Harmon 2010).

La solution logicielle proposée dans cette thèse est de s’appuyer sur l’architecture générale de méta modélisation avec l’IDM qui est constituée de quatre niveaux de modélisation (ce que l’on appelle souvent le schéma à quatre niveaux). Ces quatre niveaux sont (Caron 2007) :

1. Le niveau M0. C’est le niveau le plus concrète. Il s’agit des objets exécutables dans une plate forme précise. Dans notre cas, il contient des codes qui seront utilisable directement dans un environnement d’optimisation.
2. Le niveau M1. Ce niveau représente ainsi le modèle de code à générer pour un environnement précis.
3. Le niveau M2. Ce niveau contient les méta modèles des modèles du niveau M1.
4. Le niveau M3. Ce niveau est le plus abstrait, il contient les méta méta modèles : une entité s’appelant MOF (Model Object Facility) (OMG 2010).

Dans la figure 4.2, l’architecture de l’IDM en quatre niveaux est représentée. Le niveau d’abstraction descend de M3 à M0. Pour cela, dans certains travaux comme par exemple (Caron 2007) et (Bézivin & Grebé 2001) les trois niveaux M3, M2 et M1 sont groupés dans une catégorisation appelée *Model World*, alors que le niveau M0 est dans la catégorisation *Real World*. Le passage entre les différents niveaux se fait par transformation (nous allons le détailler dans la suite).

En (Lamotte et al. 2008), les auteurs présente une implémentation industrielle de la génération de code pour le contrôle/commande d’un train électrique avec les techniques de l’IDM. Ce travail a montré comment les techniques du génie modèle pourrait être utilisé pour générer de contrôle / commande du code à partir d’une description de haut niveau du système. Cette génération fournit non des avantages négligeables. Le design est effectué

sur un modèle de haut niveau du système. En utilisant ce modèle assure la cohérence entre l'analyse effectuée sur le modèle et sa mise en œuvre, depuis la même entrée est prévue pour les deux activités. Ensuite, il permet d'appliquer facilement des modifications, soit sur le système physique, ou sur le principe de fonctionnement des composants.

Le passage entre les quatre niveaux est souvent bijectif (le passage est représenté par des flèches bidirectionnelles), sauf dans le cas où M₀ est l'ensemble des codes s'exécutant sur une plate-forme car, dans ce cas, nous ne pouvons pas déduire le modèle en amont (en M₁).

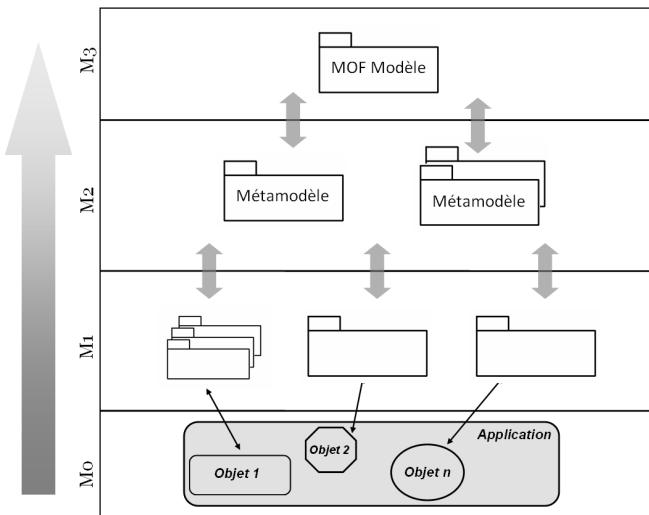


FIGURE 4.2 – Quatre niveaux d'abstraction proposés par l'ingénierie dirigée par les modèles

Comme l'IDM vise à séparer les modèles reposant sur une logique métier, des modèles liés à l'implémentation logicielle (Atkinson et al. 2004) afin de garder la pérennité de la logique de métier malgré le changement d'environnement (ULMER et al. 2010). Il faut donc concevoir d'une part un modèle indépendant de la plate forme (PIM 'Platform Independent Model'). Nous mettrons dans ce modèle le savoir faire métier. D'autre part des modèles dépendants de la plate forme (PSM 'Plateform specific model') dans lesquels les spécification d'implémentation de PIM dans une plate forme précise sont définies.

Pour générer les codes exécutables dans une plate forme, le concept de l'IDM permet de transformer le PIM en PSM qui de son côté sert comme une base pour générer les codes finaux. Pour cela il faut concevoir le PIM comme une grammaire pour construire le PSM, cela en prenant en compte aussi la description de la plate forme (son métamodèle).

Cet transformation est souvent mentionnée comme la transformation en Y ce que la figure 4.3 l'illustre.

Dans l'état de l'art, il y a travaux avec lesquels la séparation entre le métier et le savoir est concrétisé. En (Lamotte et al. 2007) l'ingénierie des modèles a été utilisé avec succès pour décrire une architecture des systèmes de fabrication configurable et d'analyser sa tolérance. Il fournit des outils pour automatiser la manipulation de modèle et d'assurer la cohérence entre les nombreux modèles.

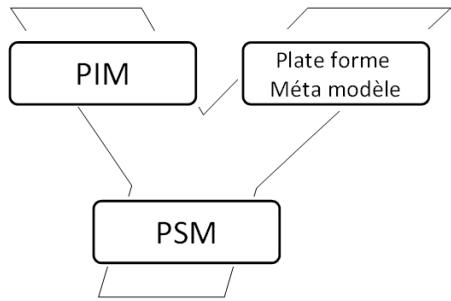


FIGURE 4.3 – Transformation en Y de PIM en PSM selon le concept de l'IDM

Il existe plusieurs outils intégrant des concepts issus de l'IDM comme *Rational Software Modeler* et *Softeam MDA Modeler*. Les travaux de développement menés dans le cadre de cette thèse ont été réalisés sur la plate-forme *Eclipse* ([Foundation 2010](#)), développée originellement en Java, mais qui inclut désormais de puissants outils de métamodélisation et de projection (grâce aux extensions Eclipse Modeling Framework, Graphical Modeling Framework, Papyrus, Omondo...).

4.3 Adaptation des concepts issus de l'IDM au contexte du bâtiment

En fonction de la définition de 'Modèle'² proposée en ([Bézivin & Grebé 2001](#)), de la définition de 'métamodèle'³ donnée par ([OMG-MOF 2010](#)) (version 14), de la définition de plate forme⁴ pour générer les problèmes d'optimisation dans deux environnements différents (cf. figure 4.4) en conformité avec la démarche pour proposée par l'IDM, nous avons besoin de :

1. construire un métamodèle de PIM de problèmes d'optimisation (cela est au niveau M2).
2. concrétiser ce métamodèle en modèle PIM (cela est au niveau M1).
3. choisir un environnement d'optimisation.
4. construire un métamodèle de cet environnement (dans le niveau M2).
5. définir les règles de transformation (dans le niveau M2).
6. transformer le PIM de problème d'optimisation en modèle spécifique PSM aux environnements d'optimisation choisi. (dans le niveau M1).
7. raffiner le PSM pour obtenir des codes utilisables directement dans l'environnement spécifique. Ces codes sont dans le niveau (M0).

Plus concrètement :

-
2. *A model is a simplification of a system built with an intended goal in mind. The model should be able to answer questions in place of the actual system*
 3. *A metamodel is a model that defines the language for expressing a model*
 4. *une plate-forme est une infrastructure d'exécution*

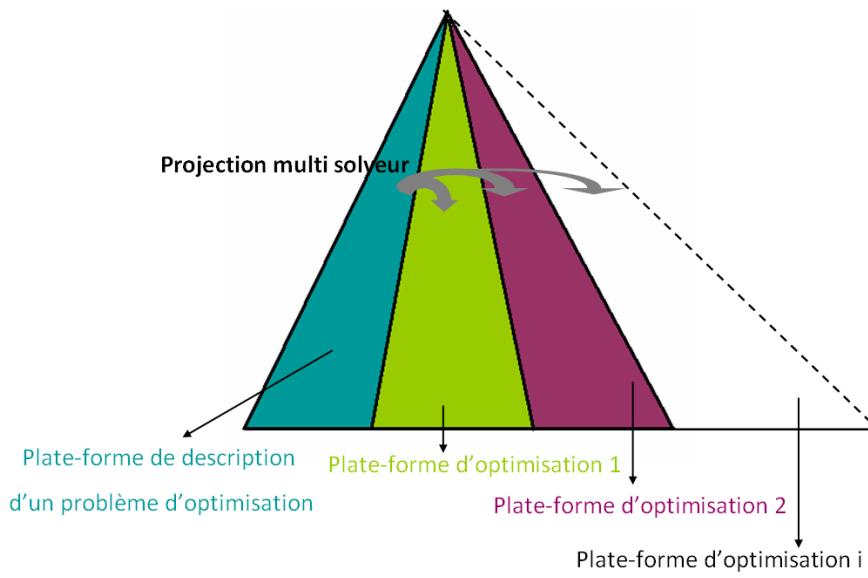


FIGURE 4.4 – Plates-formes nécessaires pour la génération 'multi solveur'

PIM de problèmes d'optimisation - pour le besoin d'un formalisme neutre des problèmes d'optimisation, il faut un modèle pour décrire les entités de problèmes sans aucun lien avec les environnements d'optimisation. Dans l'architecture d'IDM, ce modèle doit être dans le niveau M1. Nous proposons le langage XML comme modèle de description. On appelle ce modèle PIM OP-XML (niveau M1) : PIM parce qu'il est neutre au langage de résolution (AMPL⁵ ou SML). OP veut dire *optimization problem* et *XML* car il est décrit en XML.

Métamodèle de PIM de problèmes d'optimisation - le PIM OP-XML a besoin qu'on définisse pour lui des règles de construction, donc son métamodèle. Pour cela un schéma XML ([Charles & Prescod 1998](#)) est proposé comme métamodèle. On l'appelle le métamodèle de PIM OP-XML, il doit être dans le niveau M2 (des détails sont donnés dans le chapitre 5).

Linstanciation du métamodèle de PIM OP-XML donne le PIM OP-XML, puis la spécification avec un cas précis d'un problème d'optimisation donne lOP-XML. Ce dernier est un document XML représentant un problème d'optimisation donné (avec des valeurs numériques des paramètres, de contraintes) : il est au niveau M0 dans l'architecture d'IDM.

Métamodèle d'environnement d'optimisation - Pour le besoin d'une définition des environnements d'optimisation utilisés. Il faut créer un métamodèle de problème d'optimisation pour chaque environnement, car chacun a son propre langage pour décrire les problèmes d'optimisation. Ces métamodèles sont appelés PM OP-CADES et PM OP-CPLEX pour CADES et CPLEX respectivement. Ils décrivent la structure de problèmes d'optimisation et aussi les entités selon le langage de CADES (SML) et de CPLEX (AMPL).

Ces PM sont au niveau M2 de l'architecture d'IDM. Or, pour approcher du monde

5. AMPL est une langue de type AML : A modeling Language for Mathematical Programming

réel, il faut des modèles de niveau M1. Pour cela, on propose le PSM OP-CPLEX et le PSM OP-CADES.

PSM d'environnement d'optimisation - Ce sont des modèles dépendants des plateformes (PSM). Ils représentent des modèles génériques de problèmes d'optimisation dans les plateformes de résolution CADES et CPLEX. Ainsi, il y a deux PSM qui sont :

1. PSM OP-CPLEX. Il s'agit d'un modèle générique de problèmes d'optimisation en java pour la plateforme CPLEX.
2. PSM OP-CADES. Il s'agit d'un modèle générique de problèmes d'optimisation en java pour la plateforme CADES.

L'arrivée au niveau M0 se fait par générateur de codes des modèles utilisables dans les environnements d'optimisation. Ces modèles sont les problèmes d'optimisation prêts à être utilisés dans CPLEX et CADES. Ainsi on aura trois modèles :

1. *OP-XML.xml* : le problème d'optimisation instancié de PIM OP-XML pour une application bâtiment précise.
2. *OP.lp* : le problème d'optimisation généré automatiquement avec le langage AML pour solveur CPLEX.
3. Pour CADES on aura :
 - *OP.sml* : le problème d'optimisation généré automatiquement pour le solveur CADES en langage SML.
 - *Op.xml* : la spécification d'un problème d'optimisation pour le solveur CADES (voir paragraphe 3.3.2).

La figure 4.5 illustre les modèles précédents et les niveaux d'abstraction correspondants. Suite à cette adaptation, nous proposons le tableau de la figure 4.6 qui illustre la présentation des entités des problèmes d'optimisation dans le bâtiment intelligent ainsi que les niveaux d'abstraction correspondant aux concepts de l'IDM.

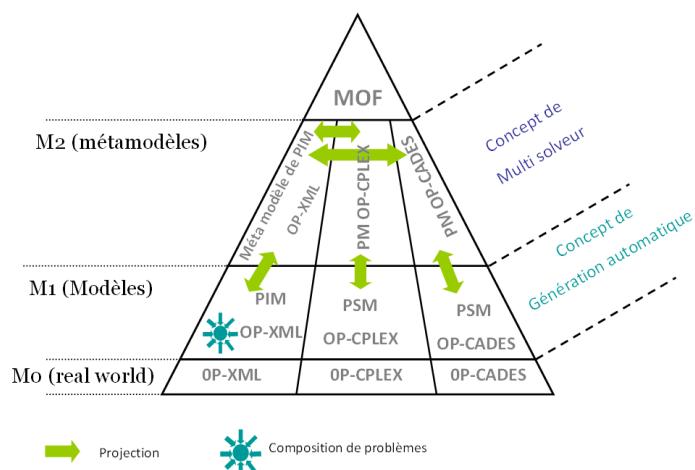
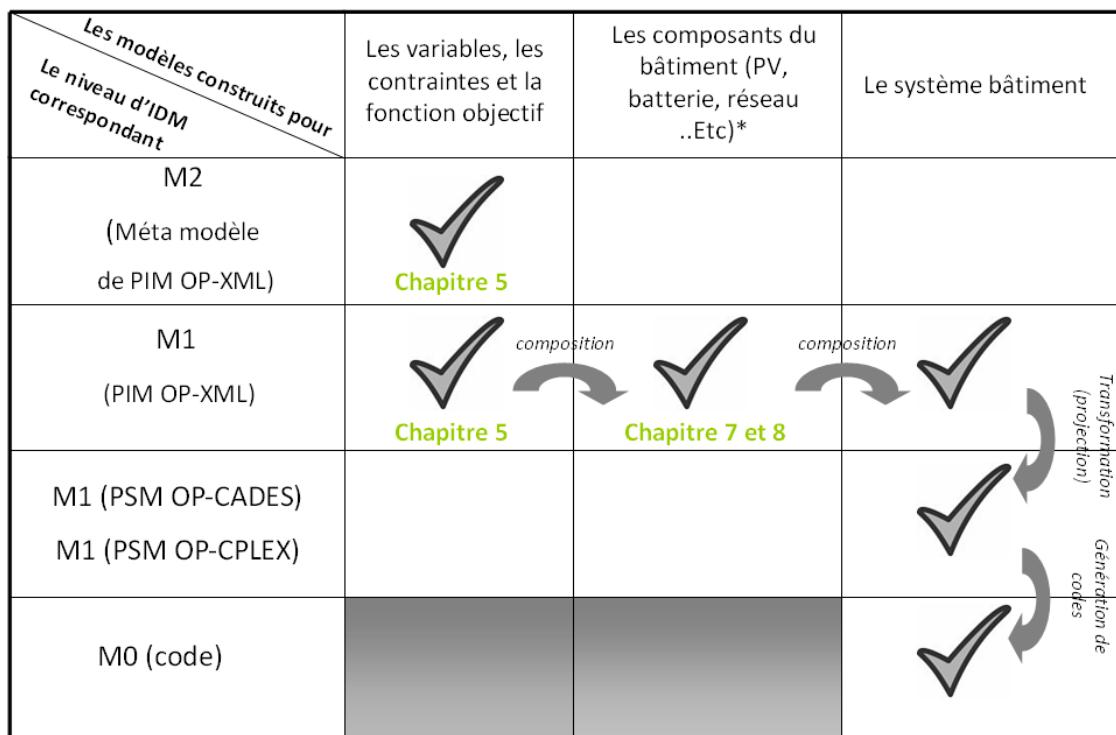


FIGURE 4.5 – Modèles nécessaires pour la génération 'multi solveur' et leurs niveaux d'abstraction en IDM



* Modèles pour la gestion et pour la dimensionnement.

FIGURE 4.6 – Modèles du problème d'optimisation pour l'application bâtiment et niveaux d'abstraction

4.4 Projection des problèmes comme une procédure de transformation des modèles

L'adaptation de concepts issus de l'IDM conduit à la problématique de transformation de modèles afin de projeter les problèmes d'optimisation d'un format neutre vers des formats dépendant de plateformes.

4.4.1 Principes

La mise en oeuvre des concepts issus de l'IDM requiert d'une part, une modélisation à chacun des niveaux d'abstraction et d'autre part, la formalisation des relations entre les niveaux de modélisation. Des outils de transformation de modèles peuvent être rencontrés entre chacun des niveaux. Dans (Czarnecki & Helsen 2003), on trouve une classification fine des approches de transformation. Globalement, les deux types de transformation de modèles sont :

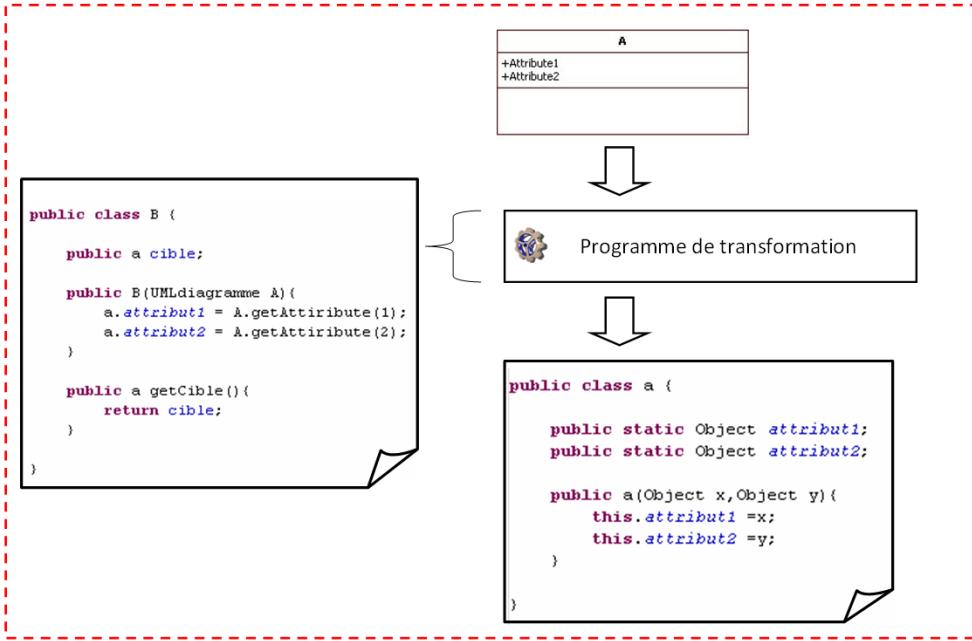
1. la transformation de PIM vers PIM.
2. la transformation de PIM vers PSM ou PSM vers PSM. On ajoute des informations aux modèles PIM pour qu'ils soient utilisables sur une plate-forme d'exécution ou transformés en modèle d'une plate-forme vers une autre. Par contre, avoir des codes à partir de PSM est un génération de codes plutôt que transformation.

La transformation de modèles se fait avec des règles de correspondances. Selon la forme de ces règles, nous trouvons trois approches de transformation (Blanc 2005) :

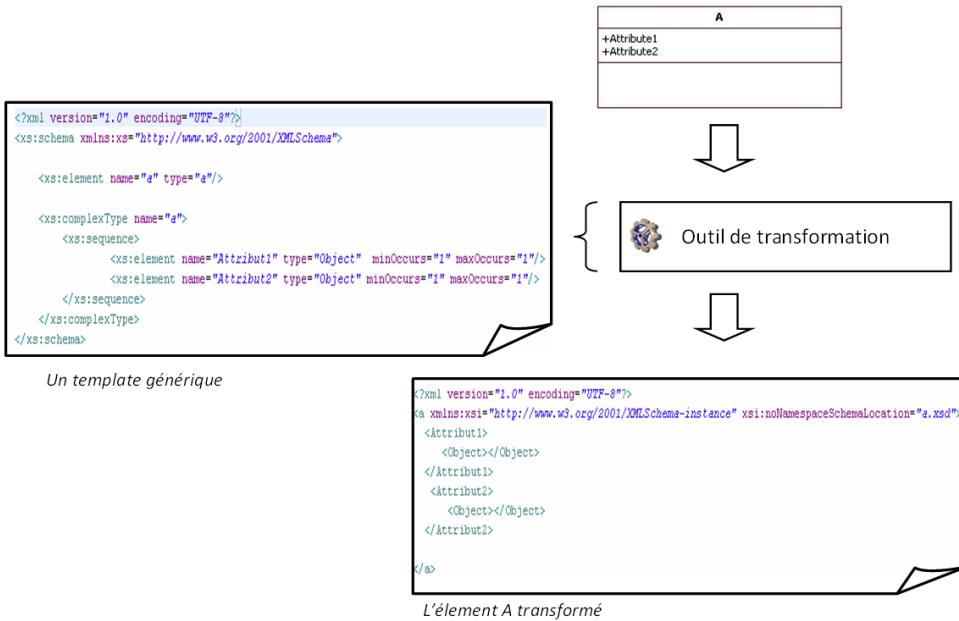
1. l'approche par programmation. On utilise la programmation pour manipuler les modèles. Cette approche est la plus utilisée. Par exemple, pour transformer une classe A décrite en UML vers une classe B décrite en Java, il y aura une classe Java $\tau_{A,B}$ dans laquelle on trouve les méthodes pour faire la transformation correspondante qui s'appuient sur les métamodèles UML (diagramme des classes) et Java (voir figure 4.7(a)).
2. l'approche par gabarit (*template*). Dans cette approche, on définit les canevas des modèles cibles (transformés) en y déclarant des paramètres. On remplace ces paramètres par des informations écrites dans le modèle source. L'approche par gabarit est très utilisée dans les développements de site Web avec des technologies comme PHP⁶. Par exemple, pour transformer l'objet A précédent en un document XML constituant une classe C , le schéma XML de ce document doit être défini. Ce schéma décrit dans le langage XSD constitue le métamodèle de C . Le document C sera généré à partir des informations dans l'objet A et des métamodèles de A et C grâce à une transformation $\tau_{A,C}$ (voir figure 4.7(b)).
3. l'approche par modélisation. Comme dans le paradigme de l'IDM 'tout est modèle', même les règles de transformation sont méta modélisées (Wimmer et al. 2011)⁷.

6. Hypertext Preprocessor. Il s'agit d'un langage de script libre principalement utilisé pour produire des pages Web dynamiques. L'idée est de définir les canevas des pages Web devant être affichées en y déclarant des paramètres

7. Nous n'avons pas entré dans les détails de cette approche. Plus d'information son accessible via le lien suivant d'http://www.emn.fr/z-info/atlanmod/index.php/Main_Page



(a) transformation par programmation



(b) transformation par gabarit

FIGURE 4.7 – Un exemple des règles de transformation des modèles

4.4.2 Transformation utilisée dans le contexte du bâtiment

Le but est de transformer le PIM OP-XML (niveau M1) composé par un utilisateur d'application vers un PSM OP-CPLEX et un PSM OP-CADES (niveau M1) puis de générer les codes M0 et les donner à cet utilisateur (éventuellement avec la solution de problème). L'approche de transformation utilisée procède par programmation.

La représentation communément admise des transformations est une représentation en forme d'Y (voir figure 4.8 (Bézivin 2003)).

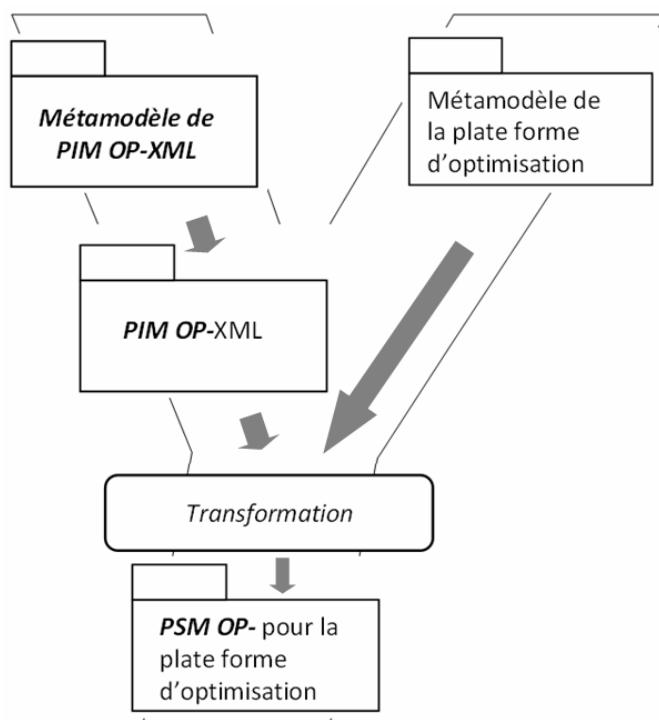


FIGURE 4.8 – Transformation de PIM vers PSM sous forme d'Y

La mise en oeuvre logicielle et les techniques de projection sont décrites dans le chapitre 5. Pour faciliter la lecture, nous avons mis les métamodèles (PM) dans l'annexe 13.1.

4.5 Grandes lignes de l'approche de génération 'multi solveur' de problèmes d'optimisation appliqués au contexte du bâtiment intelligent

Nous faisons la différence entre deux acteurs liés à la génération multi solveur de problèmes d'optimisation qui sont :

1. le développeur d'applications qui a une connaissance informatique pour développer les modèles et les projecteurs.
2. l'utilisateur d'applications qui se sert de logiciels développés par l'acteur précédent pour résoudre ses problèmes dans différents environnements.

En conséquence, l'approche suggérée pour la construction de générateur automatique de problèmes d'optimisation pour un développeur d'applications est constituée de trois étapes (ces étapes ont été réalisées dans cette thèse, nous les montrons dans la suite de la partie II) :

1. Métamodéliser le problème d'optimisation dans une syntaxe neutre (méta modèle de PIM OP-XML). Il faut construire les éléments du métamodèle de *variable*, de *contrainte* et de *fonction objectif*. Chaque élément contient une description sous forme de labels, de coefficients, de paramètres,...
2. Métamodéliser les langages des solveurs disponibles (les PM OP-CADES et PM OP-CPLEX).
3. Construire les projecteurs (règles de transformation) des PIM OP-XML vers les métamodèles associés aux solveurs.
4. Valider les transformations au niveau des PSM : projeter un modèle PIM OP-XML vers un langage compris par un solveur.

Pour les utilisateurs d'applications d'optimisation pour le bâtiment, les étapes sont (Nous montrerons ces étapes dans la partie III) :

1. Utiliser un concepteur graphique adapté pour générer le PIM OP-XML.
2. Utiliser un projecteur mis au point par un développeur d'applications vers un modèle utilisable par un solveur.
3. Récupérer les modèles utilisables (récupérer les résultats du solveur dans le modèle OP-XML est possible avec CPLEX, ce qui n'est pas encore le cas pour CADES).

La figure 4.9 résume les tâches des deux types d'utilisateur avec les niveaux de modélisation liés.

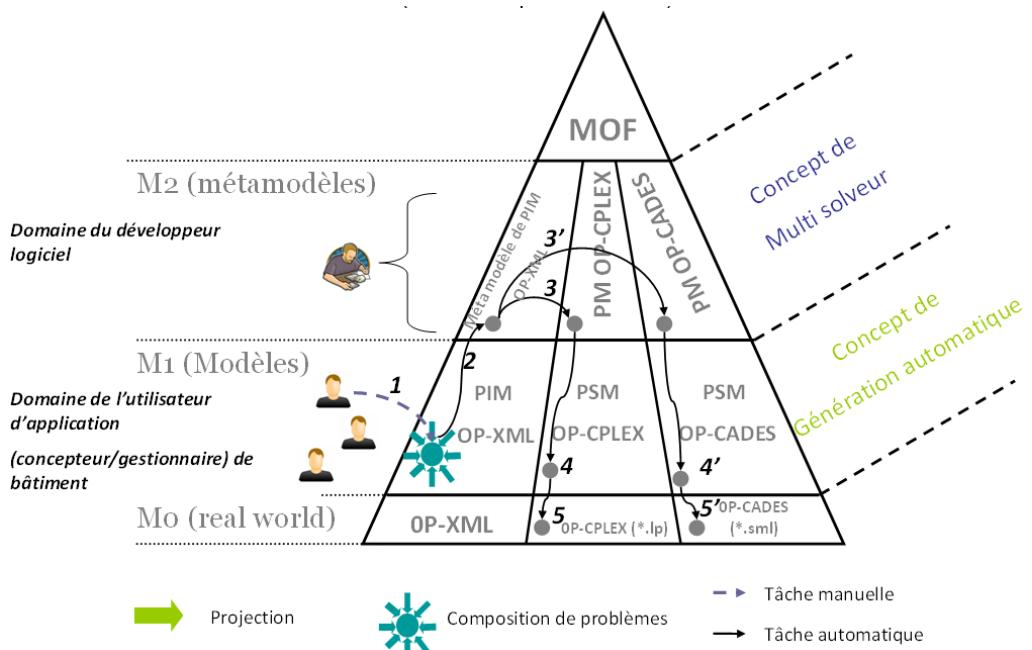


FIGURE 4.9 – Étapes de génération 'multi solveur' de problèmes d'optimisation avec les niveaux de modélisation et les acteurs concernés

L'utilisateur d'application peut composer son application bâtiment intelligent comme la figure 4.10 le présente grâce à une interface dédiée, le PIM OP-XML sera généré automatiquement en conformité avec le besoin 'multi solveur' puis instancié (OP-XML) avec des valeurs numériques d'utilisateur.

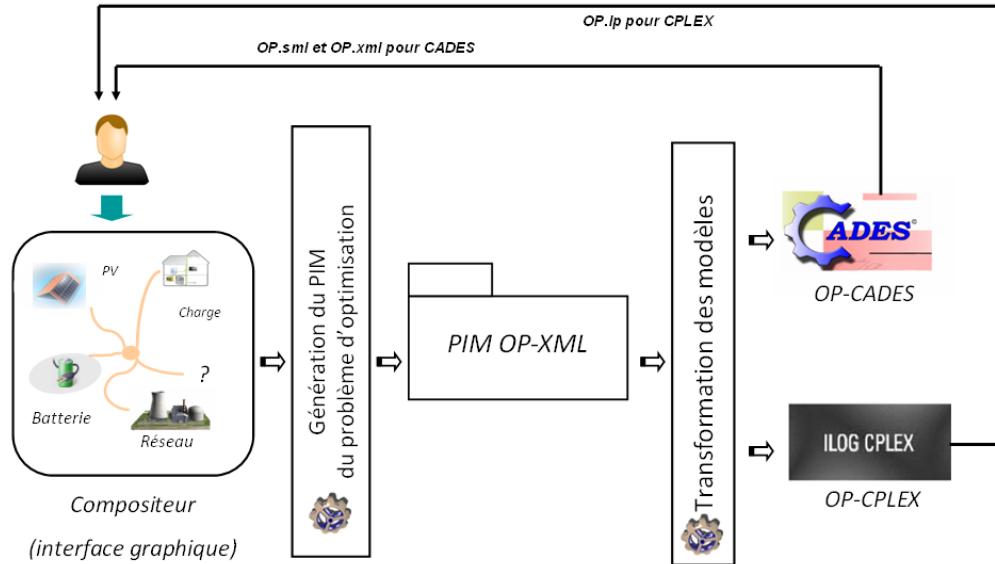


FIGURE 4.10 – Génération de problèmes d'optimisation dans l'application bâtiment pour un utilisateur d'application

4.6 Conclusion

La composition automatique présentée dans le chapitre 2 permet de générer automatiquement un format neutre du problème d'optimisation (PIM OP-XML). En complément, ce format généré peut être projeté vers différents solveurs en s'inspirant de concepts issus de l'ingénierie dirigée par les modèles (IDM). Cela se réalise au prix de la construction de deux métamodèles : un par solveur et un méta modèle de PIM OP-XML. Ainsi des projecteurs pourront être développés : c'est ce que nous appelons l'approche multi solveurs (figure 2.2). Le méta modèle de PIM OP-XML et ces projecteurs sont détaillés dans le chapitre suivant. Les métamodèles PM OP-CADES et PM OP-CPLEX sont illustrés dans l'annexe 13.1.

La figure 4.11 illustre la dépendance entre tous les modèles (de M_2 et de M_1).

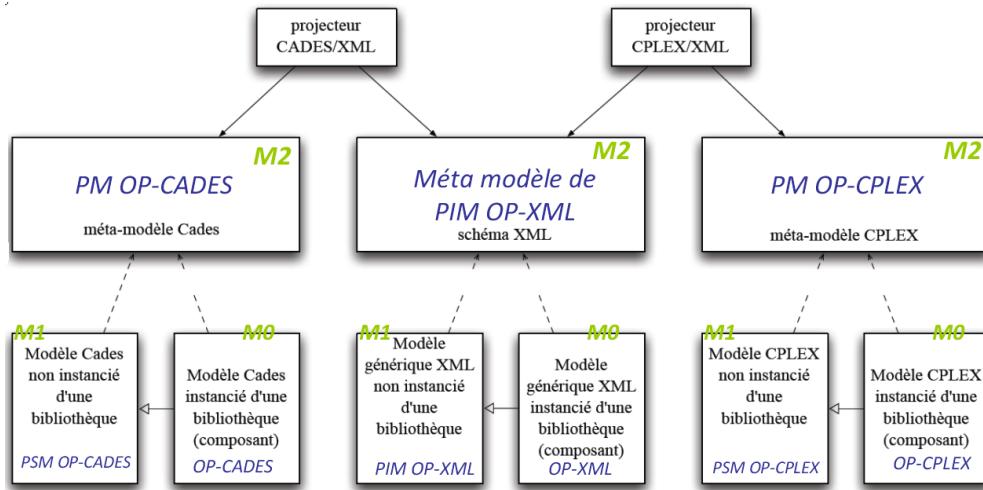


FIGURE 4.11 – Principe de métamodélisation adopté

Chapitre 5

Mise en oeuvre logicielle et techniques de projection

Une idée pour laquelle tu n'as pas souffert ne t'appartient pas

Mihai Ralea. Roumanie

SOMMAIRE

5.1	INTRODUCTION : UN FORMALISME POUR DÉCRIRE LES MÉTAMODÈLES	56
5.2	LE MÉTAMODÈLE DU OP-XML (PIM OP-XML)	56
5.2.1	Type Paramètre	58
5.2.2	Type Variable	59
5.2.3	Type Contrainte	61
5.2.4	Type Fonction objectif	63
5.2.5	Paquetage Java pour la génération de problèmes OP-XML	64
5.3	LES PROJECTEURS DES PSM (PSM OP-XML VERS DES PSM OP-CADES ET DES PSM OP-CPLEX)	65
5.3.1	La méthode	65
5.3.2	L'implémentation	67
5.4	EXEMPLE D'IMPLEMENTATION DE L'APPROCHE GÉNÉRATION AUTOMATIQUE MULTI SOLVEUR POUR UN UTILISATEUR D'APPLICATION	72
5.5	CONCLUSION	76

Résumé

Dans ce chapitre, nous présentons une implémentation logicielle du formalisme neutre de problèmes d'optimisation linéaires (méta modèle de PIM OP-XML). Il s'agit d'un schéma XML (XSD) dans lequel figurent les descriptions structurelles des composants d'un problème d'optimisation linéaire. Nous illustrons les projecteurs de PIM OP-XML vers CPLEX et CADES. Un exemple démonstratif d'utilisation d'un générateur automatique multi solveur est présenté.

5.1 Introduction : Un formalisme pour décrire les métamodèles

Revenons à l'exemple du circuit électrique. Supposons que R varie entre $0,1\Omega$ et $0,8\Omega$ et I entre $0,2A$, $0,9A$. La variable R peut être métamodélisée par :

- le nom de la variable : R
- la valeur maximale qu'elle peut atteindre (R_{max}).
- la valeur minimale qu'elle peut atteindre (R_{min}).

Ce métamodèle décrit R abstraitemen. Il comprend la structure nécessaire pour modéliser R , mais il ne présume rien de l'intégration dans l'environnement d'optimisation. L'ensemble des attributs (nom, valeur maximale, valeur minimale) est la représentation de la variable R dans le problème d'optimisation OP-XML.

La difficulté est de garantir qu'après projection, le même problème d'optimisation sera résolu dans chaque environnement de résolution cible. Pour réussir ces projections, le PIM OP-XML doit être unique et avoir les caractéristiques suivantes :

1. Il doit être écrit dans un langage descriptif précis afin d'éviter toute ambiguïté.
2. Il doit être facile à faire évoluer. Revenons à l'exemple précédent. Le PIM OP-XML doit être constitué d'un ensemble de variables, d'un ensemble de contraintes égalité ou inégalité et d'une fonction objectif à minimiser ou maximiser. Avec l'ajout de nouvelles variables, ou contraintes, la structure va évoluer mais les trois types d'éléments fondamentaux resteront les mêmes.

XML est un langage de modélisation qui répond aux deux exigences précédentes ([Charles & Prescod 1998](#)). Il permet de décrire un objet textuellement et d'assigner des propriétés, des attributs, ou même d'ajouter des descendants avec des nouveaux attributs.

Il existe des travaux ([Fourer & Lopes 2004](#)) dans lesquels un programme linéaire est décrit en langage XML (même l'appel à l'algorithme de résolution est fait dans le document XML). Utiliser XML permet de relier plusieurs langages de modélisation AML dédiés à la résolution de problèmes PLNE ([Fourer et al. 2003](#)) car XML offre des facilités de transformation de données avec le standard XSLT.

Généralement les règles qui définissent la structure d'un document XML sont contenues dans un schéma appelé le schéma XSD (*XML Schema Definition*) qui spécifie et permet de valider la structure d'un document XML (des compléments sur les schémas XSD et la validation apparaissent dans ([Charles & Prescod 1998](#))). Pour cette raison, nous allons adopter le schéma XSD comme un moyen pour construire le métamodèle de PIM OP-XML.

Dans la suite de ce chapitre nous allons illustrer le métamodèle de PIM OP-XML (M2) et le document XML résultant (M0).

5.2 Le métamodèle du PIM OP-XML (niveau M2)

Quel que soit le type de problème d'optimisation (PL, PLNE, PNL,...), il comporte quatre ensembles. La nature et la taille de ces ensembles changent le type du problème. Dans notre approche, nous restons sur des problèmes d'optimisation mono-objectif linéaires. Du point de vue global, tous ces problèmes d'optimisation contiennent les ensembles suivants :

- des variables,

- des contraintes (égalité ou inégalité),
- des paramètres,
- une fonction objectif.

Pour les problèmes de type PLNE, il existe des variables réelles et entières. Les relations entre les variables sont linéaires tout comme la fonction objectif. Nous allons profiter de cette régularité pour construire une architecture générique décrivant ce type de problèmes.

L’élément racine du schéma XML doit légitimement être l’élément *Problem*. Comme nous le constatons dans la figure 5.1, on déclare que le *Problem* est composé des éléments principaux suivants :

1. *parameterSection*. Il contient la description des éléments de type paramètre.
2. *variableSection*. Il contient la description des éléments de type variable.
3. *constraintSection*. Il contient la description des éléments de type contrainte.
4. *objectiveFunctionSection*. Il contient la description de fonction objectif.

Pour chaque élément, on déclare le type à l’aide d’un attribut *type* et l’occurrence des éléments dans l’élément racine *Problem* au moyen des attributs *minOccurs* et *maxOccurs* (si on ne déclare pas explicitement ces valeurs, cela veut dire qu’elles valent 1).

```

<xss:element name="problem" type="Problem"/>

<xss:complexType name="Problem">
    <xss:sequence>
        <xss:element name="parameterSection" type="ParameterSection"/>
        <xss:element name="variableSection" type="VariableSection"/>
        <xss:element name="constraintSection" type="ConstraintSection"/>
        <xss:element name="objectiveFunctionSection" type="ObjectiveSection"/>
    </xss:sequence>
</xss:complexType>

<xss:complexType name="ParameterSection">
    <xss:sequence>
        <xss:element name="parameter" type="Parameter" minOccurs="0" maxOccurs="unbounded" />
    </xss:sequence>
</xss:complexType>

<xss:complexType name="VariableSection">
    <xss:sequence>
        <xss:element name="variable" type="Variable" maxOccurs="unbounded"/>
    </xss:sequence>
</xss:complexType>

<xss:complexType name="ConstraintSection">
    <xss:sequence>
        <xss:element name="constraint" type="Constraint" maxOccurs="unbounded" />
    </xss:sequence>
</xss:complexType>

<xss:complexType name="ObjectiveSection">
    <xss:sequence>
        <xss:element name="objective" type="Objective" maxOccurs="unbounded" />
    </xss:sequence>
</xss:complexType>
```

FIGURE 5.1 – Partie du méta modèle de PIM OP-XML correspondant aux déclarations des quatre éléments principaux d’un problème d’optimisation

Comme nous implémentons de nouveaux types dans le schéma, il faut les déclarer. Les déclarations de chaque type sont présentées par les éléments suivants.

5.2.1 Type Paramètre

Pour donner un accès plus générique à certaines données numériques dans les contraintes, nous avons introduit l'élément *Parameter*, son métamodèle est dans la figure 5.2(a).

1. name : le nom de paramètre.
2. value : valeur numérique.

Utilisons par exemple, ce métamodèle pour le problème 3.1. Dans ce cas, le OP-XML du problème complet aura une partie liée aux paramètres a , b , c et d comme la figure 5.2(b) le montre¹.

```
<xs:complexType name="Parameter">
  <xs:sequence>
    <xs:element name="name" type="xs:ID"/>
    <xs:element name="value" type="xs:double"/>
  </xs:sequence>
</xs:complexType>
```

(a) Partie du méta modèle de PIM OP-XML, pour description d'éléments de type *Parameter*

```
<parameterSection>
  <parameter>
    <name>a</name>
    <value>8</value>
  </parameter>
  <parameter>
    <name>b</name>
    <value>3</value>
  </parameter>
  <parameter>
    <name>c</name>
    <value>7</value>
  </parameter>
  <parameter>
    <name>d</name>
    <value>5</value>
  </parameter>
</parameterSection>
```

(b) Partie de l'OP-XML de problème 3.1, décrivant les paramètres a , b , c et d

FIGURE 5.2 – Partie du méta modèle de PIM OP-XMLet de OP-XML avec les paramètres a , b , c et d

1. les types *ID* et *double* sont de types prédéfinis dans les standards de w3c Charles & Prescod (1998). L'*ID* est pour référencer cette élément dans le document XML, le *double* signifie que la contenu de cette élément est une valeur numérique réelle

5.2.2 Type Variable

Les informations essentielles dont nous avons besoin sont dans les sous-éléments. Nous décrivons donc la variable textuellement via les sous-éléments suivants (figure 5.5(a)) :

1. name : le nom de la variable.
2. min : la valeur minimum.
3. max : la valeur maximum
4. Init : la valeur initiale.
5. type : variable réelle ou entière. Pour cela, on distingue trois types qui sont : *CONTINUOUS* pour les variables réelles, *INTEGER* pour les variables entières et *BINARY* pour les variables binaires (voir figure 5.3).

```
<xs:simpleType name="VariableType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="CONTINUOUS"/>
        <xs:enumeration value="INTEGER"/>
        <xs:enumeration value="BINARY"/>
    </xs:restriction>
</xs:simpleType>
```

FIGURE 5.3 – Partie du méta modèle de PIM OP-XML, pour définir un type appelé *VariableType* permettant de contrôler la restriction sur les types de variables à utiliser

6. implementationInConstraint : Dans chaque éléments de ce type, on trouve les noms des contraintes qui contiennent cette variable, et les coefficients de cette variable. Pour cela, nous avons défini un nouveau type dans lequel on met ces informations (voir figure 5.4).

```
<xs:complexType name="ImplementationInConstraint">
    <xs:sequence>
        <xs:element name="constraintName" type="xs:IDREF"/>
        <xs:choice>
            <xs:element name="parameter" type="xs:IDREF"/>
            <xs:element name="coefficient" type="xs:double"/>
        </xs:choice>
    </xs:sequence>
</xs:complexType>
```

FIGURE 5.4 – Partie du méta modèle de PIM OP-XML, pour définir un type appelé *ImplementationInConstraint* permettant de décrire les informations liées à l'implémentation des variables dans les contraintes

Par exemple, l'OP-XML pour le problème 3.1 contient une partie liée à la variable x_2 comme la figure 5.5(b) le montre.

```

<xs:complexType name="Variable">
  <xs:sequence>
    <xs:element name="name" type="xs:ID"/>
    <xs:element name="min" type="xs:double"/>
    <xs:element name="max" type="xs:double"/>
    <xs:element name="init" type="xs:double" minOccurs="0"/>
    <xs:element name="type" type="VariableType"/>
    <xs:element name="implementationInConstraint" type="ImplementationInConstraint"
      minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

```

- (a) Partie du méta modèle de PIM OP-XML, relative à la description d'un élément de type *Variable* réutilisant les types *VariableType* (voir figure 5.3) et *ImplementationInConstraint* (voir figure 5.4)

```

<variable>
  <name>x1</name>
  <min>0.0</min>
  <max>10</max>
  <init>0.0</init>
  <type>CONTINUOUS</type>
  <implementationInConstraint>
    <constraintName>c1</constraintName>
    <coefficient>2</coefficient>
  </implementationInConstraint>
    <implementationInConstraint>
      <constraintName>c3</constraintName>
      <coefficient>1</coefficient>
    </implementationInConstraint>
      <implementationInConstraint>
        <constraintName>c4</constraintName>
        <coefficient>2</coefficient>
      </implementationInConstraint>
        <implementationInConstraint>
          <constraintName>objectiveFunction</constraintName>
          <coefficient>4</coefficient>
        </implementationInConstraint>
      </implementationInConstraint>
    </implementationInConstraint>
  </implementationInConstraint>
</variable>

```

- (b) Partie de l'OP-XML de problème 3.1, décrivant la variable x_1

FIGURE 5.5 – Partie du méta modèle de PIM OP-XML et OP-XML illustré avec la variable x_1 du problème 3.1

5.2.3 Type Contrainte

La partie du méta modèle de PIM OP-XML liée aux descriptions de contraintes est constituée de :

1. name : le nom de la contrainte.
2. type : inférieure et égale ou supérieure et égale.

```
<xss:simpleType name="Type">
    <xss:restriction base="xss:string">
        <xss:enumeration value="LEConstraint" />
        <xss:enumeration value="GEConstraint" />
        <xss:enumeration value="EQConstraint" />
    </xss:restriction>
</xss:simpleType>
```

FIGURE 5.6 – Partie du méta modèle de PIM OP-XML, relative aux descriptions de sous élément *type*

3. expression : expression de la contrainte. Pour les problèmes linéaires ce champ ne sert à rien, car nous avons déjà indiqué au niveau des variables quelles sont les contraintes dans lesquelles elles interviennent avec le coefficient linéaire associé (cf. figure 5.4). Par contre nous pouvons utiliser ce champ pour le cas de problèmes non linéaires, parce que dans ce champ, nous mettrons l'expression non linéaire de cette contrainte. Il faudrait dans ce cas préciser dans quel langage est décrite l'expression (SML, AMPL, autre...), les variables qui sont implémentés dans cette expression et leurs coefficients. Pour cela, nous avons défini un nouveau type pour ce sous élément (*expression*) afin de grouper ces informations (voir figure 5.7²).
4. bound : le second membre de la contrainte.

A titre d'exemple, l'OP-XML du problème 3.1 contient la déclaration de la contrainte C_1 que nous mettons à titre d'illustration sur la figure 5.8(b).

2. le type *IDREF* est un type prédéfini dans le langage XML, il permet d'utiliser une élément défini quelque part dans le document comme une valeur d'un attribut pour une autre élément

```

<xs:complexType name="Expression">
    <xs:sequence>
        <xs:element name="formalism" type="Formalism"/>
        <xs:element name="formula" type="xs:string" maxOccurs="unbounded"/>
        <xs:element name="variableNames" type="VariablesInExpression" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:simpleType name="Formalism">
    <xs:restriction base="xs:string">
        <xs:enumeration value="SML"/>
        <xs:enumeration value="OPL"/>
    </xs:restriction>
</xs:simpleType>

<xs:complexType name="VariablesInExpression">
    <xs:sequence>
        <xs:element name="variableName" type="xs:IDREF"/>
        <xs:element name="coefficient" type="Value"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="Value">
    <xs:choice>
        <xs:element name="parameter" type="xs:IDREF"/>
        <xs:element name="coefficient" type="xs:double"/>
    </xs:choice>
</xs:complexType>

```

FIGURE 5.7 – Partie du métamodèle de PIM OP-XML, relative aux descriptions de sous élément expression

```

<xs:complexType name="Constraint">
  <xs:sequence>
    <xs:element name="name" type="xs:ID"/>
    <xs:element name="type" type="Type"/>
    <xs:element name="expression" type="Expression" minOccurs="0"/>
    <xs:element name="bound" type="Parameter"/>
  </xs:sequence>
</xs:complexType>

```

(a) Partie du métamodèle de PIM OP-XML, description d'élément de type *Constraint*

```

<constraint>
  <name>c1</name>
  <type>LEConstraint</type>
  <expression>
    <formalism>SML</formalism>
    <formula>2x1 +x2</formula>
    <variableNames>
      <variableName>x1</variableName>
      <coefficient>
        <coefficient>2</coefficient>
      </coefficient>
    </variableNames>
    <variableNames>
      <variableName>x2</variableName>
      <coefficient>
        <coefficient>1</coefficient>
      </coefficient>
    </variableNames>
  </expression>
  <bound>
    <parameter>a</parameter>
  </bound>
</constraint>

```

(b) Partie de l'OP-XML de problème 3.1 qui contient C_1 FIGURE 5.8 – Partie du métamodèle de PIM OP-XML qui décrit l'élément de type *Constraint* et une partie de OP-XML correspondante à la description de c_1 du problème 3.1

5.2.4 Type Fonction objectif

Le PIM OP-XML contient une description d'une fonction objectif. Il a deux caractéristiques à respecter qui sont montrées dans la figure 5.9 :

1. `variableName` : le nom de la variable qui représente la fonction objectif.
2. `optimizationType` : la direction d'optimisation. Pour cela nous avons défini un type illustré dans la figure 5.9(a).

```

<xs:complexType name="Objective">
  <xs:sequence>
    <xs:element name="variableName" type="xs:IDREF"/>
    <xs:element name="optimizationType" type="OptimizationType"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="OptimizationType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="MAXIMIZE"/>
    <xs:enumeration value="MINIMIZE"/>
  </xs:restriction>
</xs:simpleType>
```

(a) Partie du méta modèle de PIM OP-XML, description d'élément de type fonction objectif

```

<objectiveFunctionSection>
  <objective>
    <variableName>z</variableName>
    <optimizationType>MINIMIZE</optimizationType>
  </objective>
</objectiveFunctionSection>
```

(b) Partie de OP-XML correspondant pour le problème 3.1

FIGURE 5.9 – Partie du méta modèle de PIM OP-XML et le OP-XML correspondant pour le problème 3.1

5.2.5 Paquetage Java pour la génération de problèmes OP-XML

Pour générer le modèle OP-XML, nous avons développé un paquetage informatique codé en Java. Ce package, nommé PIMgenerator, est présenté dans la figure 5.10.

Le PIMgenerator est composé de quatre parties qui sont :

1. *pim_OPXML* : c'est un package composé d'objets java pour représenter le métamodèle PIM OP-XML.
2. *ProblemBuilder* : c'est une classe pour représenter le modèle du problème d'optimisation comme un objet Java. Elle contient des méthodes pour ajouter des variables, des contraintes et la fonction objectif.
3. *PIMdocumentGenerator* : c'est une classe qui génère le document XML à partir de l'objet *ProblemBuilder*.
4. *XMLHandler* : c'est une boîte à outils pour transformer l'objet *ProblemBuilder* en document XML.

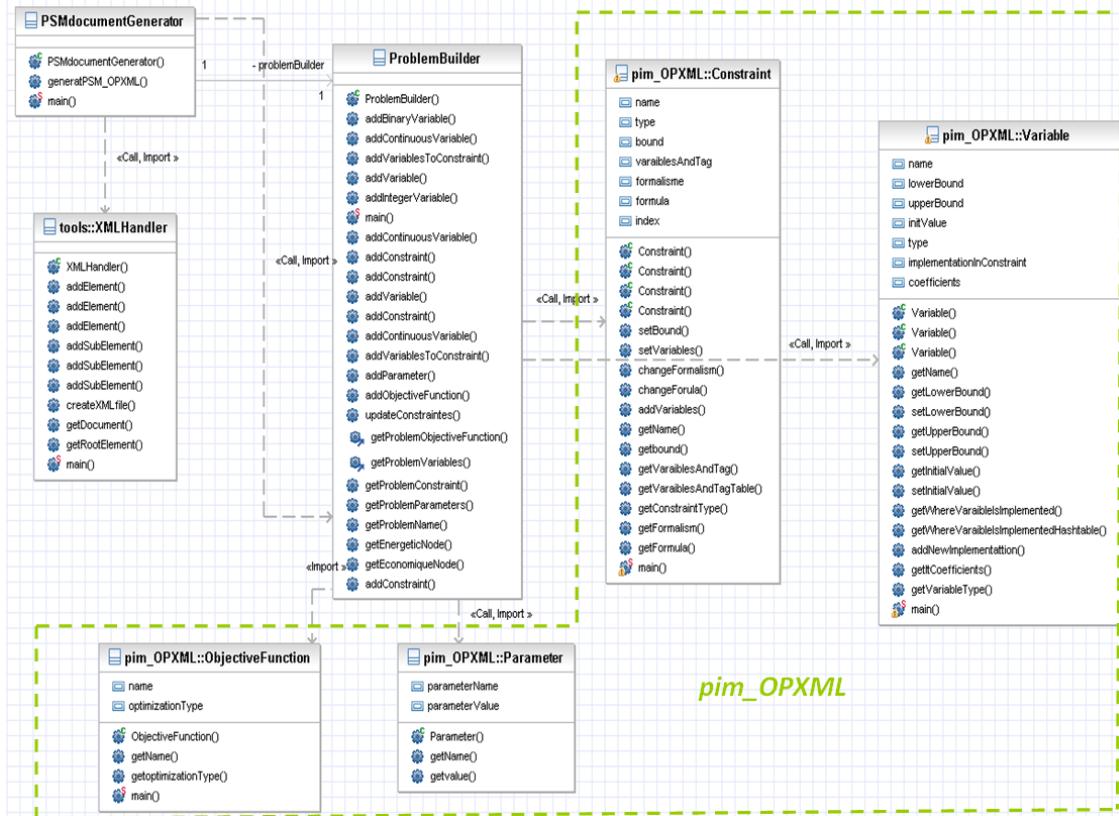


FIGURE 5.10 – Diagramme des classes pour générer le modèle OP-XML

Les flèches représentent le référencement d'une classe par une autre : les objets issus de la classe de départ de la flèche comportent une référence vers des objets de la classe d'arrivée.

5.3 Les Projecteurs des PSM (PIM OP-XML vers des PSM OP-CADES et des PSM OP-CPLEX)

5.3.1 La méthode

La transformation de modèle est le point principal du paradigme de l'ingénierie dirigé par les modèles. Elle représente le cœur de notre travail de développement. Quel que soit le concepteur de logiciel et pour que son outil soit 'multi solveur', nous lui proposons d'implémenter le paquetage que nous avons développé.

Avec la transformation, les problèmes éventuels de solveurs non adaptés (nous pouvons utiliser plusieurs solveurs sans obligation de refaire l'écriture) peuvent être surpassés. De surcroît, la projection sur deux solveurs différents nous a permis de découvrir un phénomène très intéressant pour la gestion de sources, en comparant les résultats des solveurs pour le même problème d'optimisation. Les détails de ce phénomène sont présentés dans le chapitre 9.

Afin de mener la transformation et de générer les modèles employables, nous avons défini certaines règles de transformation sur le PIM construit précédemment. Pour chaque environnement, nous disposons d'un projecteur dans lequel on fait appel à ces règles de transformation. Il y a donc un projecteur pour CPLEX et un autre pour CADES (voir les figures 5.11).

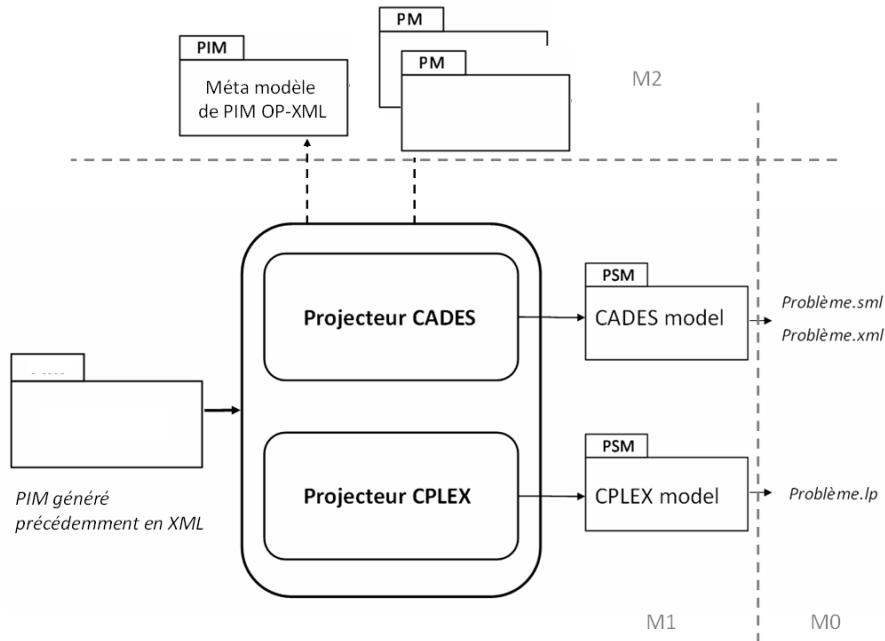


FIGURE 5.11 – Étapes de la transformation des modèles

Les projecteurs traduisent les éléments de PIM OP-XML complet du problème en contraintes et variables dans les environnements cibles. Les projecteurs vont générer les PSM OP-CPLEX et PSM OP-CADES de niveau M1. Puis à partir de ces derniers, ils génèrent les codes qui sont utilisables directement par le gestionnaire du bâtiment niveau M0 (voir figure 5.12).

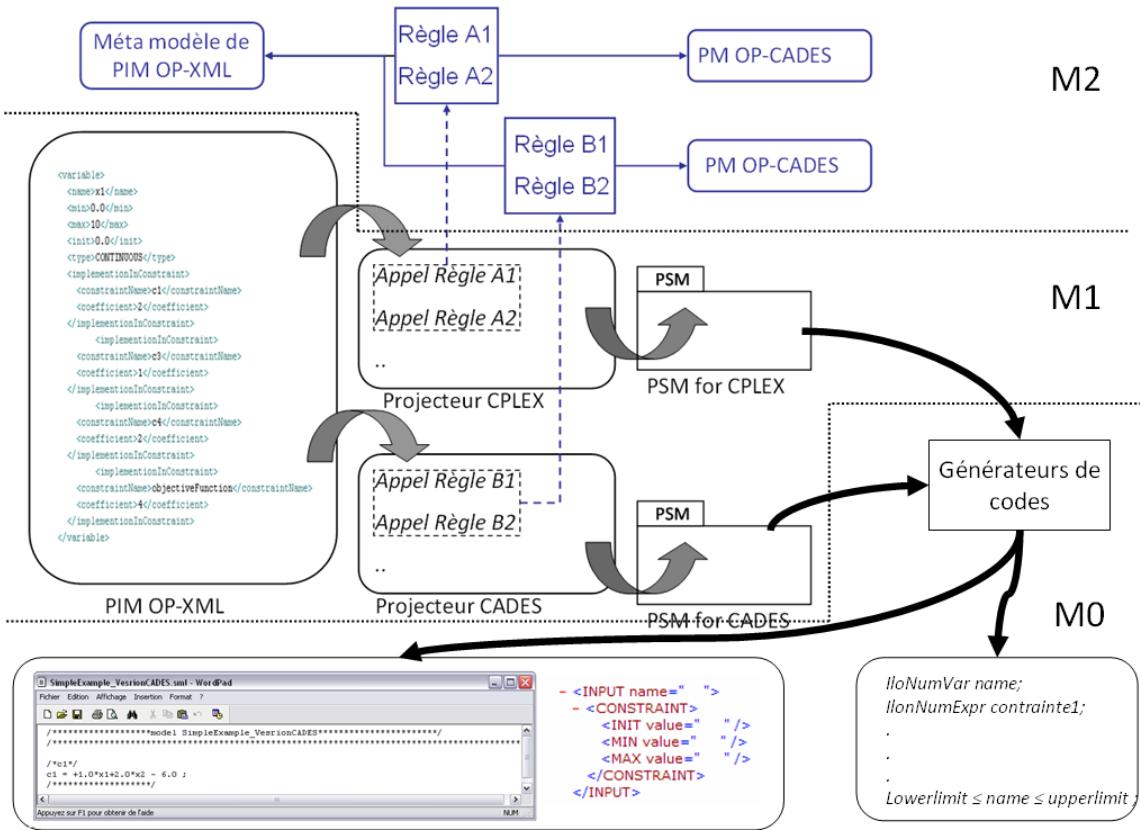


FIGURE 5.12 – Chaque projecteur fait appel aux règles de transformation afin de générer les modèles cibles pour chaque environnement d'optimisation

Un élément de PIM OP-XML de type Variable peut être transformé et puis ajouté dans le PSM OP-CPLEX sous *ILOOPT3* en utilisant les données dans le PIM OP-XML qui sont : son nom, sa valeur minimale et sa valeur maximale, en prenant en compte le métamodèle d'une variable en CPLEX :

IloNumVar Variable = Cplexmodel.addVariable(VarName,lowerLimit,UpperLimit)

La règle de transformation sera donc un code. Selon le type de cet élément, s'il s'agit d'un élément de type Variable, la méthode précédente est appelée pour générer une variable dans le modèle CPLEX (voir la figure 5.13).

Le même élément de PIM OP-XML peut être transformé en une variable en CADES. En s'appuyant sur le métamodèle de CADES (PM OP-CADES). Il y a une différence entre CPLEX et CADES au niveau de la déclaration d'une variable : en CADES la déclaration de la variable n'est pas séparée de son implémentation comme en CPLEX. En CADES, on écrit l'implémentation en sml puis on met les spécifications de cette variable dans le fichier de spécification (sa valeur minimale, maximale et initiale) (voir figure 5.14).

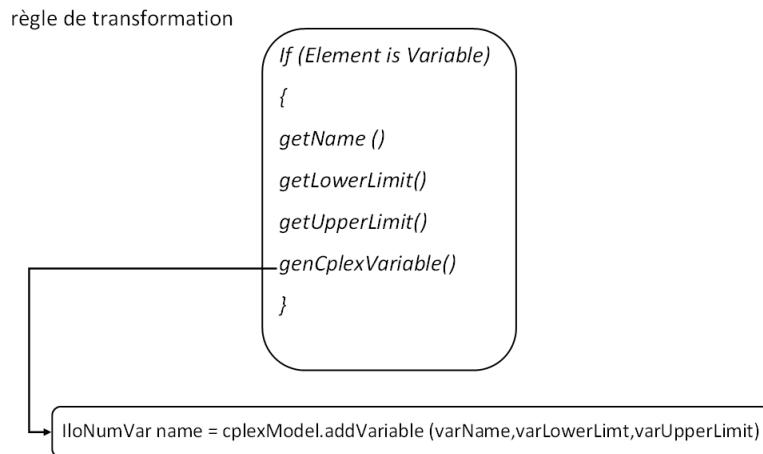


FIGURE 5.13 – Règle de transformation d'une variable de PIM OP-XML vers une variable PSM OP-CPLEX

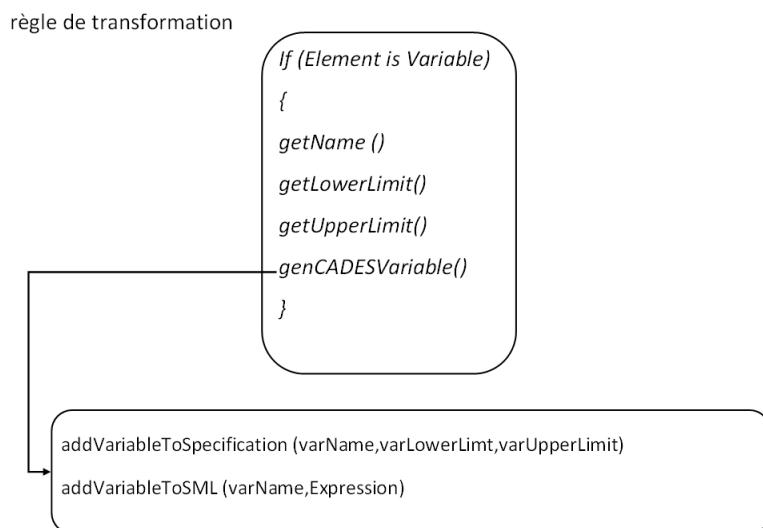


FIGURE 5.14 – Règle de transformation d'une variable de PIM OP-XML vers une variable PSM OP-CADES

5.3.2 L'implémentation

Chaque projecteur contient une classe (techniquement on l'appelle *parser*) pour lire le PIM OP-XML. Le *parser* donc recueille les informations dans le PIM OP-XML puis il les met à la disposition des règles de transformation (voir la figure 5.15). En fait, le *parser* est une classe Java pour lire les documents XML.

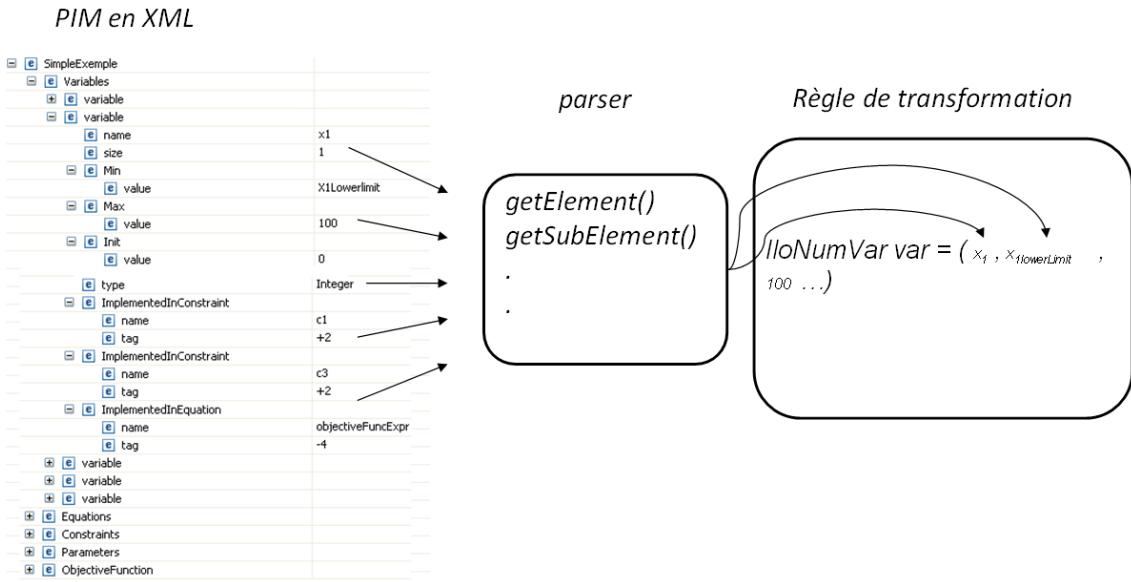


FIGURE 5.15 – Le *parser* lit l'OP-XML puis met les informations à la disposition des règles de transformation

5.3.2-1 Partie CADES : Projecteurs PIM OP-XML vers PSM OP-CADES

La règle de transformation a besoin de connaître la structure des éléments cibles. Cette structure est l'ensemble des métamodèles des variables et des contraintes de l'environnement d'optimisation. Pour l'environnement CADES, nous avons construit notre propre métamodèle pour l'utilisateur en Java (PM OP-CADES). Il s'agit d'un package dans lequel on donne au concepteur logiciel les méthodes pour générer des variables (scalaires ou vectorielles), des contraintes et des fonctions externes dans le problème d'optimisation (voir la figure 13.2). Nous générerons le modèle dépendant de plate-forme pour CADES (PSM OP-CADES) comme instance de ce métamodèle.

La spécificité du projecteur CADES est qu'il faut à la fois générer les variables (dans un fichier **.sml*) et leurs spécifications (dans autre fichier **.xml*). Le fichier **.sml* pour être utilisé dans le module *CADES generator* et le fichier **.xml* pour être utilisé ensuite dans le module *CADES optimizer*.

Pour expliquer notre démarche, nous allons prendre un exemple démonstratif : Supposons qu'on ait la formulation suivante :

$$\begin{aligned}
 c_1 : \quad & x_1 + 2x_2 \leq 6 \\
 & 0 \leq x_1 \leq 10 \\
 & 0 \leq x_2 \leq 10
 \end{aligned} \tag{5.1}$$

```
Variable x1 = pbBuilder.addVariable("x1", 0,10,0, "CONTINUOUS",
        new String[]{"c1"}, new String[]{"+1"});
Variable x2 = pbBuilder.addVariable("x2",0,10,0, "CONTINUOUS",
        new String[]{"c1"}, new String[]{"+2"});

Constraint c1 = pbBuilder.addConstraint("c1",6, "LEConstraint");
```

(a) Les codes pour générer l'OP-XML

constraintExemple		
parameterSection		
variableSection		
variable		
name	x1	
min	0.0	
max	10.0	
init	0.0	
type	CONTINUOUS	
implementationInConstraint		
constraintName	c1	
coefficient	1.0	
variable		
name	x2	
min	0.0	
max	10.0	
init	0.0	
type	CONTINUOUS	
implementationInConstraint		
constraintName	c1	
coefficient	2.0	
constraintSection		
constraint		
name	c1	
type	LEConstraint	
bound	6.0	
objectiveFunctionSection		

(b) L'OP-XML

FIGURE 5.16 – Les codes et l'OP-XML générés pour l'exemple 5.1

Les codes pour générer l'OP-XML sont simples et ils sont illustrés figure 5.16.

Le *parser* récupère les informations de la variable x_1 comme : $x_{1lowerLimit}, x_{1upperLimit}$ et $ImplementedIn = c_1$ et $ImplementedInCoef = +1$. Il récupère les informations sur x_2 de la même façon. Ensuite le projecteur CADES passe par tous les éléments du PSM en commençant par l'élément c_1 . Le projecteur CADES initialise au début les contraintes du modèle (il y en a une seule dans cet exemple), cela veut dire qu'il prépare un formulaire de type *Constraint* avec le même nom c_1 . Ensuite, il cherche dans toutes les variables celles qui sont implémentées dans cette contrainte, puis il les mémorise temporairement.

La règle de transformation est : l'élément x_1 est de type *Variable* qui est implémentée dans c_1 ; il faut donc d'abord l'ajouter sur le formulaire de c_1 avec son coefficient et en même temps ajouter la spécification de x_1 dans le fichier de spécifications en XML (voir la figure 5.17). Le projecteur CADES applique la même règle pour x_2 .

```

<?xml version="1.0" encoding="ISO-8859-1" ?>
- <SPECIFICATIONS>
  - <OPTIMIZER file="SQPVF13_V3.0-Beta2.jar">
    - <CONFIGURATION>
      <MAX_ITERATION value="100" />
      <END_PRECISION value="1e-5" />
    </CONFIGURATION>
  </OPTIMIZER>
  <MODEL file="SimpleExample_VersionCADES.icar" />
  - <INPUT name="x1">
    - <CONSTRAINT>
      <INIT value="0.0" />
      <MAX value="10.0" />
      <MIN value="0.0" />
    </CONSTRAINT>
  </INPUT>
  - <INPUT name="x2">
    - <CONSTRAINT>
      <INIT value="0.0" />
      <MAX value="10.0" />
      <MIN value="0.0" />
    </CONSTRAINT>
  </INPUT>
  - <OUTPUT name="c1">
    - <CONSTRAINT>
      <INIT value="0.0" />
      <MAX value="0.0" />
      <MIN value="-10.0" />
    </CONSTRAINT>
  </OUTPUT>
</SPECIFICATIONS>
```

FIGURE 5.17 – Spécification de la contrainte 5.1 en XML pour CADES

Pour écrire cette contrainte en sml, il faut modifier sa représentation car nous ne pouvons pas écrire la formulation explicite de 5.1). La contrainte c_1 doit être comme :

$$c_1 = x_1 + 2x_2 - 6 \quad (5.2)$$

Le problème d'optimisation peut avoir plusieurs contraintes ; il faut donc automatiser la re-formulation. Pour cela le projeteur prend en charge cette tâche, va générer une variable supplémentaire sous le nom c_1 (cela apparaît dans le sml et dans la spécification en xml), puis écrit automatiquement en sml la formulation 5.2 comme illustré en figure 5.18. En même temps, dans le fichier de spécifications, on met le seuil maximal de c_1 à une valeur nulle et le seuil minimal à une représentation de l'infini négatif⁴ (voir la figure 5.18).

L'utilisateur peut se servir directement du *problème.sml* pour générer le *problème.icar* avec le module *CADES Generator*, puis utiliser le *problème.xml* et lancer l'optimisation avec le module *CADES Optimizer* (comme montré dans le paragraphe 3.3.2).

4. c'est une valeur à définir par l'utilisateur d'application car mathématiquement cette valeur peut changer la domaine de solution

```

*****model SimpleExample_VersionCADES*****
/*
*c1*/
c1 = +1.0*x1+2.0*x2 - 6.0 ;
*/

```

(a) La formulation en sml

```

<?xml version="1.0" encoding="ISO-8859-1" ?>
- <SPECIFICATIONS>
  - <OPTIMIZER file="SQPVF13_V3.0-Beta2.jar">
    - <CONFIGURATION>
      <MAX_ITERATION value="100" />
      <END_PRECISION value="1e-5" />
    </CONFIGURATION>
  </OPTIMIZER>
  <MODEL file="SimpleExample_VersionCADES.icar" />
  - <INPUT name="x1">
    - <CONSTRAINT>
      <INIT value="0.0" />
      <MAX value="10.0" />
      <MIN value="0.0" />
    </CONSTRAINT>
  </INPUT>
  - <INPUT name="x2">
    - <CONSTRAINT>
      <INIT value="0.0" />
      <MAX value="10.0" />
      <MIN value="0.0" />
    </CONSTRAINT>
  </INPUT>
  - <OUTPUT name="c1">
    - <CONSTRAINT>
      <INIT value="0.0" />
      <MAX value="0.0" />
      <MIN value="-10.0" />
    </CONSTRAINT>
  </OUTPUT>
</SPECIFICATIONS>

```

(b) les spécifications en xml

FIGURE 5.18 – La formulation 5.2 en sml générée automatiquement par le projeteur CADES et les spécifications en xml

5.3.2-2 Partie CPLEX : Projecteur PIM OP-XML vers PSM OP-CPLEX

L'environnement CPLEX ([ILOG 2010](#)) donne aux développeurs en java un ensemble des packages java dans lequel on trouve toutes les méthodes pour manipuler un objet en CPLEX. Nous nous sommes appuyés sur cette interface pour développer le métamodèle CPLEX (PM OP-CPLEX). Ce métamodèle est illustré par la figure [13.3](#).

Les règles de transformation dans le projecteur CPLEX vont implémenter ce métamodèle pour transformer un élément de type *Variable* en variable, et un élément de type *Constraint* en syntaxe CPLEX (IloNumVar et IloNumExpr respectivement).

Grâce à l'interface de ILOG, les règles de transformation que nous avons développées sont moins compliquées que celles pour CADES. Par exemple, la règle de transformation de la contrainte de [5.1](#) est : c_1 est une contrainte d'inégalité, il faut donc générer un objet de type *IloNumExpr* avec le nom c_1 , puis donner les limites de cette contrainte avec les méthodes *.addLe()* et *.addGe()* qui sont fournies par ILOG. Il faut ensuite générer deux *IloNumVar* avec la classe java *IloNumVar()* sous les noms x_1, x_2 respectivement, puis ajouter leurs limites en utilisant les méthodes *setMax()* et *setMin()* (aussi fournis par ILOG). Enfin, il faut ajouter ces deux variables dans l'expression de c_1 selon leurs coefficients en utilisant les méthodes *sum()*, *diff()*, *prod()*. Un exemple de la règle pour ajouter une variable sur une contrainte est illustré figure [5.19](#), Dans cette règle, nous implementons les méthodes existant dans la bibliothèque de CPLEX.

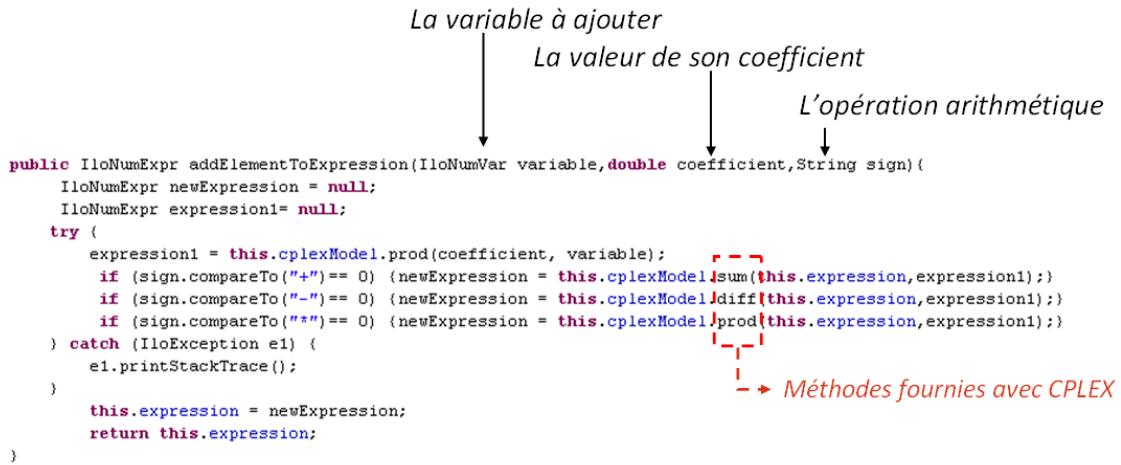


FIGURE 5.19 – Règle de transformation dans le projecteur CPLEX pour ajouter une variable sur une contrainte avec une expression arithmétique

Enfin, le résultat de la transformation est les modèles utilisables directement avec CPLEX *SimpleExample.lp* (figure [5.20](#)).

```
\Problem name: SimpleExample.lp
Minimize
  obj:
Subject To
  c1: 2 x2 + x1 <= 6
Bounds
  0 <= x2 <= 10
  0 <= x1 <= 10
End
```

Appuyez sur F1 pour obtenir de l'aide

FIGURE 5.20 – La contrainte 5.1 générée automatiquement pour CPLEX

5.4 Exemple d’implémentation de l’approche génération automatique multi solveur pour un utilisateur d’application

Nous allons illustrer comment générer un OP-XML à partir de PIM OP-XML avec le paquetage Java PSMgenerator.

Supposons que l’on ait le problème suivant à résoudre 3.1 :

$$\begin{aligned} \text{Min } FO \quad z = & 4x_1 + 5x_2 \\ c1 : & 2x_1 + x_2 \leq a \\ c2 : & 1 \leq x_2 \leq b \\ c3 : & 1 \leq x_1 \leq 10 \end{aligned} \tag{5.3}$$

sachant que :

– $a = 8$, $b = 3$.

Globalement, les étapes pour générer le OP-XML sont (cf. figure 5.21) :

1. Ajouter les paramètres du problème (a et b).
2. Déclarer les variables (x_1 , x_2 et z).
3. construire les contraintes d’inégalité (c_1 , c_2 et c_3).
4. construire une contrainte d’égalité qui représente FO .
5. Déclarer que la fonction objectif est z avec un objectif de le maximiser.

Les étapes en plus de détails pour générer le problème sont :

1. Construire le modèle OP-XML : avec le package Java (illustré dans le paragraphe 5.2.5), il faut construire le modèle de la figure 5.22. Cela recouvre les étapes suivantes :
 - (a) Ajouter les paramètres avec leurs valeurs numériques. Par exemple pour introduire le paramètre a :
 - i. name : on met a .
 - ii. value : on met $+8$.
 - (b) Ajouter les contraintes avec leurs types et les valeurs de second membre.
- Par exemple, pour c_1 , on va déclarer les spécifications suivantes :
- i. Name : c_1 .

```

/*
*****
Parameter a = pbBuilder.addParameter("a", 8);
Parameter b = pbBuilder.addParameter("b", 3);
Parameter x1lowerLimit = pbBuilder.addParameter("x1lowerLimit", 1);
Parameter x1upperLimit = pbBuilder.addParameter("x1upperLimit", 10);
Parameter x1initValue = pbBuilder.addParameter("x1initValue", 1);
Parameter x1coeffinC1 = pbBuilder.addParameter("x1coeffinInC1", 2);
Parameter x1coeffinC3 = pbBuilder.addParameter("x1coeffinInFO", 4);

Constraint c1 = pbBuilder.addConstraint("c1", a, "LEConstraint", "AMPL", "");
Constraint c2 = pbBuilder.addConstraint("c2", b, "LEConstraint");
Constraint fo = pbBuilder.addConstraint("FO", 0, "EQConstraint");

Variable x1 = pbBuilder.addVariable("x1", x1lowerLimit, x1upperLimit, x1initValue, "CONTINUOUS",
    new String[]{c1.getName(), fo.getName(),
    new Parameter[]{x1coeffinC1,x1coeffinC3}});

Variable x2 = pbBuilder.addContinuousVariable("x2", 1, 10, 0,new String[]{c1.getName(),
    c2.getName(),
    fo.getName()}, new String[]{"1","1","5"});

Variable z = pbBuilder.addContinuousVariable("z", Double.MIN_VALUE, Double.MAX_VALUE, 0,
    new String[]{fo.getName()}, new String[]{"-1"});

ObjectiveFunction obj = pbBuilder.addObjectiveFunction("z", "MINIMIZE");

pbBuilder.updateConstraints();

String metamodelName = "SimpleExample";
String foldername= "PIM_OPXML";
PSMdocumentGenerator metamodelGenerator = new PSMdocumentGenerator(metamodelName, foldername, pbBuilder);
metamodelGenerator.generatePSM_OPXML();
*****

```

FIGURE 5.21 – Exemple en code Java pour construire le PIM OP-XML

- ii. bound : c'est la valeur de second membre ou le paramètre correspondant, car nous avons déclaré dans le schéma (PIM OP-XML) le type de donnée de cet élément est : soit une valeur, soit une référence à un paramètre. Dans ce cas on met *a*.
 - iii. Type : on déclare que c'est une contrainte de type inférieure ou égale (*LE-Constraint*)
 - iv. formalisme : on choisit le langage de formalisme (par exemple AMPL).
 - v. formula : ce champ sera rempli par le générateur de problèmes avec l'expression mathématique de cette contrainte, l'utilisateur d'application le laisse vide pendant la phase de construction de OP-XML.
- (c) Ajouter les variables x_1 , x_2 avec leurs spécifications de type : valeurs minimale, maximale et initiale, puis indiquer les noms des contraintes où apparaissent ces variables.
- Par exemple, pour ajouter x_1 sur le problème OP-XML, il faut passer par les étapes suivantes (les autres variables ont été ajouté à l'instar de x_1) :
- i. nom : x_1 .
 - ii. Min : le paramètre correspondant à la valeur minimale ($x1lowerLimit$).
 - iii. Max : le paramètre correspondant à la valeur maximale ($x1upperLimit$).
 - iv. Init = le paramètre correspondant à la valeur initiale ($x1initValue$).
 - v. type : on met CONTINUOUS.

- vi. `implementationInConstraint` : x_1 on met les noms des contraintes où elle est implémentée c_1 , FO . Puis on met les coefficients de la variable dans ces contraintes, dans ce cas sont des paramètres $x1coeffinC1$ et $x1coeffinFO$.
- (d) Ajouter la fonction objectif et le sens de l'optimisation (maximiser ou minimiser) : on déclare que la variable z est à minimiser ce qui correspond dans ce cas à la fonction objectif.
2. choisir la plateforme d'optimisation cible (par exemple CADES, sinon sera CPLEX).
 3. Lancer la transformation : l'utilisateur aura directement les modèles utilisables dans CADES. On aboutira par exemple à des fichiers `exemple.sml` et `exemple-spec.xml`. Ces modèles sont générés automatiquement ultérieurement.

Après projection, l'utilisateur CADES obtient un problème d'optimisation prêt à l'emploi (voir figure 5.23). L'utilisateur avec le module *CADES Generator* génère un fichier `exemple.icar`. Il peut alors démarrer l'optimisation via le module *CADES Optimizer*.

Si l'utilisateur voudrait résoudre le même problème avec CPLEX, il suffit d'utiliser l'autre projecteur qui va projeter le même modèle OP-XML vers CPLEX. L'utilisateur obtiendra alors un fichier `exemple.lp` prêt à être utilisé. La figure 5.24 montre une copie d'écran du problème sous CPLEX.

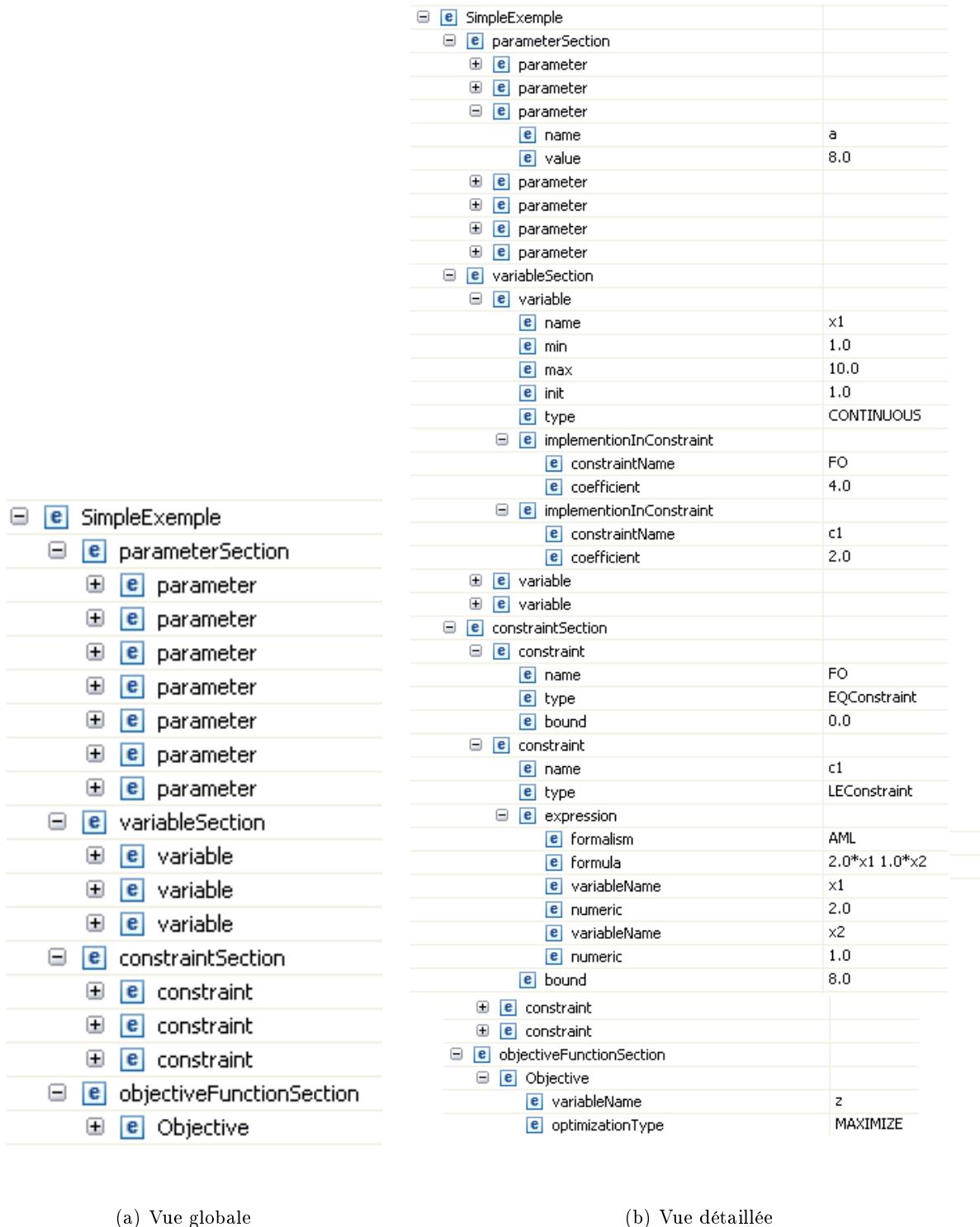


FIGURE 5.22 – Modèle OP-XML du problème d'optimisation 3.1

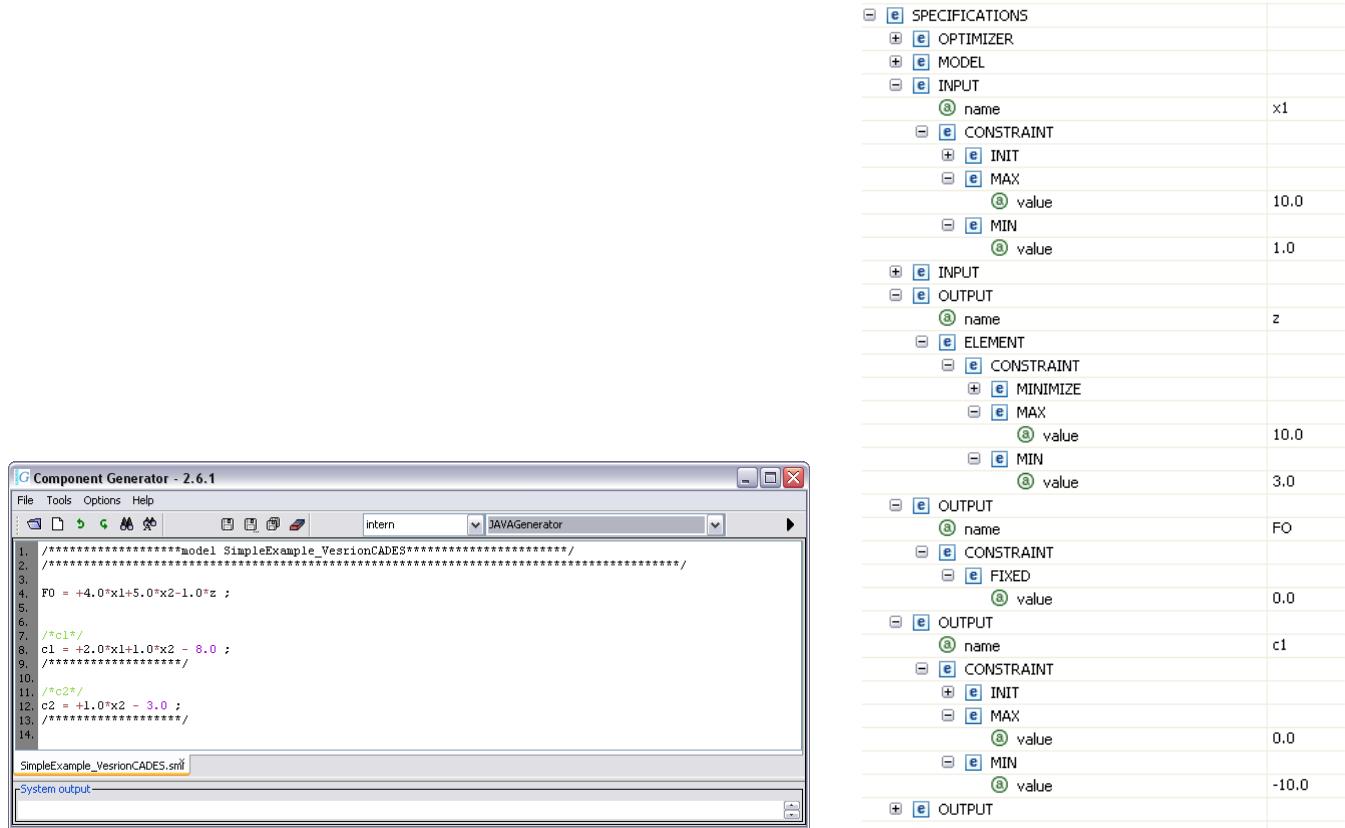


FIGURE 5.23 – Modèles du problème 3.1 générés automatiquement

```

\Problem name: JNICPLEX.lp

Minimize
  obj: z
Subject To
  FO: - z + 5 x2 + 4 x1 = 0
  c1: x2 + 2 x1 <= 8
  c2: x2 <= 3
Bounds
  3 <= z <= 10
  1 <= x2 <= 10
  1 <= x1 <= 10
End

```

Appuyez sur F1 pour obtenir de l'aide

FIGURE 5.24 – Capture écran du modèle .lp généré automatiquement 3.1

5.5 Conclusion

Avec la proposition structurelle de méta modèle de PIM OP-XML, nous pouvons décrire abstrairement un problème d'optimisation linéaire avec des nombres entiers (PLNE) sans être lié à un environnement de résolution particulier. Le modèle PIM OP-XML résultant est :

1. simple car les informations sont décrites sous forme textuelle. Elles décrivent chaque élément d'un problème d'optimisation.
2. bien structuré selon le type d'élément du problème d'optimisation ;
3. évolutif car l'ajout ou la suppression d'un élément est naturel.

Les projecteurs vont transformer les modèles OP-XML instanciers de PIM OP-XML en des problèmes utilisables directement dans les environnements cibles (CPLEX et CADES).

Chapitre 6

Conclusions partie II

Au début de cette partie, nous avons présenté des notions qui nous ont aidé à implémenter une approche pour la génération automatique de problèmes d'optimisation. Chaque composant du bâtiment doit être modélisé comme un sous problème avec ses contraintes locales. Les modèles doivent déclarer des ports énergétiques et économiques qui sont des variables qui participent aux contraintes globales du problème final. En effet, quand un concepteur décrit un problème et ajoute un composant, le sous problème correspondant est ajouté au problème complet, puis le générateur de problème va intégrer les ports dans les contraintes globales.

Vu qu'il existe une grande diversité d'environnements de résolution, nous avons proposé de profiter d'approche de génération automatique de problèmes et de projection pour produire un seul formalisme indépendant de tout environnement (PIM OP-XML), puis de transformer ce formalisme vers ou dans les environnements souhaités. Nous avons appelé cette approche 'génération automatique multi solveur'. Nous avons pris en compte deux environnements cibles CPLEX et CADES. Nous avons illustré leur fonctionnement et les exigences demandées sur les modèles sous-jacents.

L'ingénierie dirigée par les modèles conceptualise des solutions pour implémenter l'approche multi solveur, via les techniques de transformation de modèles vers ou dans différents environnements de résolution.

Un modèle de problème linéaire avec variables entières (PLNE) indépendant de toute plate-forme de résolution (PIM OP-XML) est présenté. Nous avons réussi grâce aux caractéristiques avantageuses de XSD à l'implémenter.

Nous avons construit un méta modèle pour chaque environnement (CADES méta modèle et CPLEX méta modèle). En nous appuyant sur ces méta modèles nous pourrons obtenir des modèles spécifiques du PLNE pour les composants dans chaque environnement (PSM). Deux projecteurs vont générer des modèles utilisables du problème en CADES et en CPLEX.

D'un côté, l'intérêt d'utiliser les concepts issus de l'IDM avec l'approche de génération automatique tient dans les points suivants :

1. Dans la problématique du bâtiment, si nous avons besoin d'utiliser un autre solveur, il suffit d'ajouter des règles au projecteur pour qu'il soit adapté aux spécifications du nouveau solveur, par contre, la plate-forme de description en XML (cf. figure 4.4) reste inchangée.
2. Si l'architecture du bâtiment est changée, par exemple, en ajoutant une nouvelle source ou charge électrique, et si nous n'avons pas dans la bibliothèque de composant un modèle de cette source, il suffit d'introduire son modèle dans la bibliothèque en respectant le PIM et sans modifier la description de la plate-forme car la transformation avec l'IDM garantit la génération de modèles ensuite en M1. La transformation peut

être utilisée car elle est assez générique et indépendante de l'architecture physique du bâtiment.

D'un autre côté, la mise en oeuvre des concepts précédents demande un savoir faire au niveau du développement logiciel. Cela induit une limite à l'implémentation de notre approche.

L'implémentation de tous ces concepts et méthodes pour les problèmes de gestion et de dimensionnement des sources est développé dans les chapitres suivants. Le positionnement au sein du manuscrit est représenté figure 6.1 :

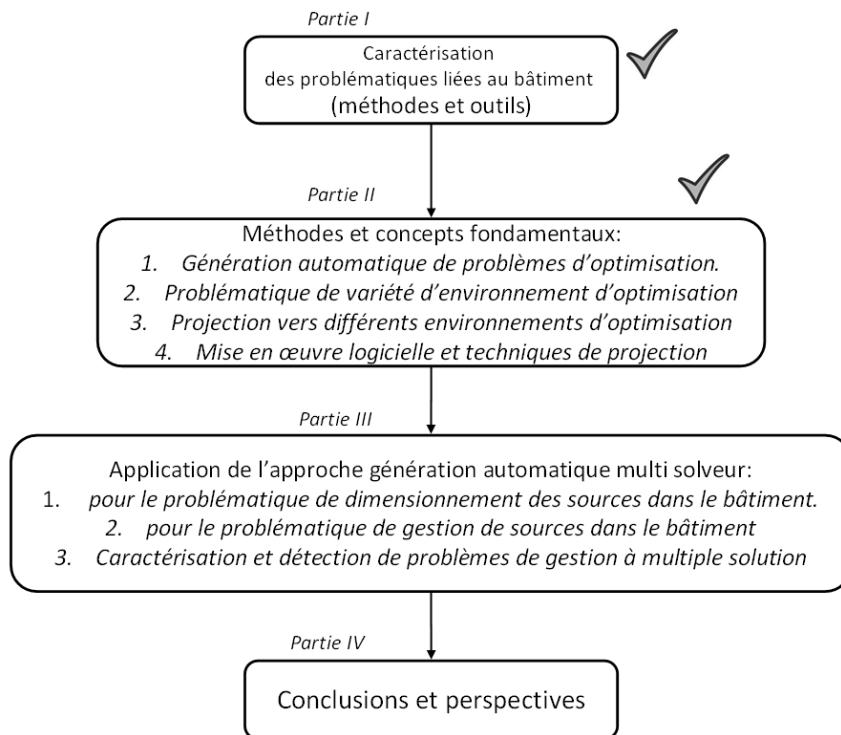


FIGURE 6.1 – Localisation dans le manuscrit de thèse

Troisième partie

Implémentation de l'approche de
génération automatique multi solveur
pour résoudre les problèmes
d'optimisation du bâtiment

Chapitre 7

Problèmes de dimensionnement optimal des sources électriques dans le bâtiment

ne fais jamais un geste que tu regretteras ultérieurement
prophète Mohammed

SOMMAIRE

7.1	PRINCIPE DE DIMENSIONNEMENT DE SYSTÈMES BÂTIMENT	82
7.2	MÉTHODES DE DIMENSIONNEMENT DES SOURCES ÉLECTRIQUES	83
7.2.1	Contexte global	83
7.2.2	Méthode énergétique/économique utilisée	83
7.3	DIFFICULTÉS LIÉES À LA RÉSOLUTION DE PROBLÈMES DE DIMENSIONNEMENT DE SOURCES	84
7.3.1	Complexité des données nécessaires	84
7.3.2	Nécessité d'outils de résolution appropriés	87
7.4	CAPITALISATION DU SAVOIR FAIRE POUR LA PROBLÉMATIQUE DE DIMENSIONNEMENT DES SOURCES	89
7.4.1	Besoin d'outil métier	89
7.4.2	Les métiers et les acteurs concernés par les problématiques de dimensionnement	90
7.5	MISE EN OEUVRE POUR UN DIMENSIONNEMENT ÉNERGÉTIQUE-ÉCONOMIQUE DE SOURCES	93
7.5.1	Partie développement logiciel (pour le <i>développeur de logiciels</i>)	94
7.5.2	Partie modélisation (pour le <i>concepteur de types de modèles</i>)	94
7.5.3	Partie composition de bâtiments (pour le <i>concepteur de systèmes bâtiment</i>)	106
7.6	ÉTUDE PARAMÉTRIQUE DU PROBLÈME POUR LE DIMENSIONNEMENT DES SOURCES	109
7.6.1	L'impact des données météorologiques	109
7.6.2	Impact des données relevant des hypothèses de type marché <i>HSEM</i>	113
7.6.3	Impact de l'hypothèse sur le type d'usagers	115
7.7	CONCLUSION	117

Résumé

Dans ce chapitre, nous illustrons l'intérêt de la génération automatique de problèmes d'optimisation pour la résolution du dimensionnement des sources dans le bâtiment. Le principe est de générer automatiquement les problèmes de dimensionnement en considérant trois acteurs : un développeur logiciel, un concepteur de types de modèles et un concepteur de systèmes de bâtiments. Plusieurs architectures multi-sources seront étudiées afin de montrer l'avantage de la génération automatique de problèmes d'optimisation. Enfin, une étude paramétrique avec plusieurs scénarios sera présentée pour

valider la démarche en analysant les problèmes générés et la sensibilité des variables de dimensionnement vis-à-vis des variations des données du problème.

7.1 Principe de dimensionnement de systèmes bâtiment

La définition du mot dimensionnement ([Larousse 2010](#)) est donnée par :

Déterminer les dimensions ou les caractéristiques fonctionnelles qu'il convient de donner à un élément pour qu'il joue convenablement le rôle qui lui revient.

Le dimensionnement des sources électriques dans le domaine du génie électrique peut se comprendre dans deux sens :

1. Un dimensionnement physique (géométrique) des composants électriques, comme par exemple trouver le diamètre du rotor ou la largeur de dent d'une machine synchrone réduisant les pertes Joules. Les chercheurs dans ce domaine utilisent des méthodes numériques ou analytiques pour optimiser le dimensionnement comme dans ([BOMMÉ 2009](#)). Cela se fait au niveau du composant.
2. Un dimensionnement fonctionnel des sources électriques. Cela se fait au niveau d'un système composé de plusieurs sources. Le but est de trouver les caractéristiques fonctionnelles ou la capacité productive d'énergie d'une source. Dans cette thèse, nous implémentons cette définition en tenant compte l'aspect système multi sources. Nous allons présenter certaines méthodes de dimensionnement par la suite.

Dans un système composé d'une seule source électrique, les charges doivent être alimentées par l'unique source électrique. Le bilan est donc simple ($P_{chargeTotale} = P_{source}$). Dans le cas de systèmes multi-sources, l'énergie totale résulte de la participation de plusieurs sources. En conséquence, le dimensionnement doit déterminer les capacités de production de chaque source en tenant compte les contraintes de dimension mais aussi la conservation des flux énergétiques dans le système. Il s'agit ainsi de trouver les variables de dimensionnement.

Parfois, les variables de dimensionnement sont calculées directement à partir d'un modèle mathématique donné. Or, ce n'est pas toujours possible : il faut généralement résoudre un problème inverse pour en déduire les variables de dimensionnement. Dans notre démarche, le problème de dimensionnement sera formulé comme un problème d'optimisation (qui résout un problème inverse).

Par exemple, le dimensionnement d'une source de type fournisseur d'énergie électrique consiste à déterminer le type d'abonnement à contractualiser avec le fournisseur d'énergie (en France, P_{Max} , type de tarif). Ceci pourra être formulé par une contrainte de type :

$$0 \leq P_{reseau} \leq P_{Max} \quad (7.1)$$

Pour les panneaux solaires, le concepteur de systèmes bâtiments cherche à trouver le nombre optimal de panneaux sans dépasser certaines dimensions, comme la surface disponible, et en préservant un amortissement financier intéressant. Il y a plusieurs facteurs qui peuvent influencer le nombre de panneaux à installer comme le rayonnement solaire journalier, le prix du panneau, le prix de rachat de l'énergie solaire, etc... Il faut donc formuler le problème de dimensionnement comme un ensemble de contraintes pour trouver un nombre de panneaux tout en prenant en compte tous ces facteurs. Notons qu'un certain nombre de ces contraintes restent spécifiques à la résolution du problème de dimensionnement et qu'elles n'apparaissent pas dans les problématiques de gestion énergétique.

Parfois les panneaux photovoltaïques sont accompagnés par un système de stockage. Le besoin du concepteur de systèmes bâtiments consiste à trouver la capacité optimale de

la batterie de stockage pour un fonctionnement optimal. La capacité dépend de certaines contraintes comme la décharge maximale autorisée, la vitesse de charge et de décharge, le prix de kWh, etc...

Tous ces aspects seront étudiés dans la suite de ce chapitre.

7.2 Méthodes de dimensionnement des sources électriques

7.2.1 Contexte global

La conception des sources électriques n'est pas un problème nouveau. Par exemple, ([Bakos et al. 2003](#)) montre que la rentabilité de systèmes PV dépend de la stratégie appliquée par le fournisseur d'énergie et du prix de rachat de l'énergie produite avec des sources locales. ([Smiley & Jones 2000](#)) et ([Nafeh 2009](#)) s'intéressent au nombre optimal de panneaux solaires pour un cas de système isolé (PV, batterie sans accordement au réseau de distribution). Un programme de calcul est donné dans ([Smiley & Jones 2000](#)) pour le nombre optimal de panneaux solaires ; le programme ne permet pas de déterminer la taille de la batterie ni la puissance du générateur diesel utilisé. Néanmoins, dans ([Nafeh 2009](#)), un cas d'étude précis a été pris en compte pour trouver le dimensionnement de l'installation PV.

([Notton et al. 2009](#)) propose une méthode pour dimensionner un système PV connecté au réseau en prenant en compte les différentes technologies de fabrication de modules PV. Différentes situations géographiques sont comparées. Dans ([Singh 2009](#)) les chercheurs présentent une nouvelle méthode pour calculer le coût de la génération d'énergie solaire à long terme en prenant en compte les variations de facteurs économiques (comme le taux d'intérêt, le crédit, etc...) mais ils n'ont pas étudié les conséquences de cette variation sur la gestion d'énergie de fournisseur et de PV.

Les facteurs influant le dimensionnement de l'onduleur et des panneaux solaires pour un système multi sources sont étudiés dans le travail de [Mondol et al. \(2006\)](#), ceci en faisant varier l'angle d'inclinaison, le site géographique et la surface disponible.

Il faut prendre en compte le niveau contractuel d'énergie avec le fournisseur, dans ([Angioletti & Despretz 2004](#)) nous trouvons que l'abonnement avec le fournisseur d'énergie se fait sous une tranche de consommation (6, 9, 12 kW), et le coût d'abonnement est différent pour chaque tranche.

Nous remarquons qu'il y a plusieurs manières et éléments pour faire le dimensionnement. Dans ce travail, nous implémentons la démarche de génération automatique multi solveur en prenant en compte la méthode de dimensionnement suivante.

7.2.2 Méthode énergétique/économique utilisée

La conception d'un système bâtiment comporte généralement une dimension énergétique et une dimension économique qui s'exprime souvent en terme de retour sur l'investissement ([Percebois 1989](#)). Pour formuler un compromis énergétique-économique, les problèmes de dimensionnement doivent souvent formuler des hypothèses de gestion à long terme (de manière souvent plus sommaire que pour un problème de gestion en temps-réel).

Nous avons prolongé les travaux de ([Pham et al. 2009](#)) ; dans ces travaux, les chercheurs proposent une méthode énergétique-économique pour le dimensionnement des sources. Le

principe est de formuler un problème avec lequel on cherche les variables de dimensionnement qui rendent le système le plus rentable économiquement (la durée nécessaire pour amortir les investissements initiaux est la plus courte possible) et qui garantissent une fonctionnalité optimale des sources (en évitant les cas de sous ou sur dimensionnement des sources électriques). La fonction objectif sera de maximiser la valeur nette présente¹ sur la période de l'étude².

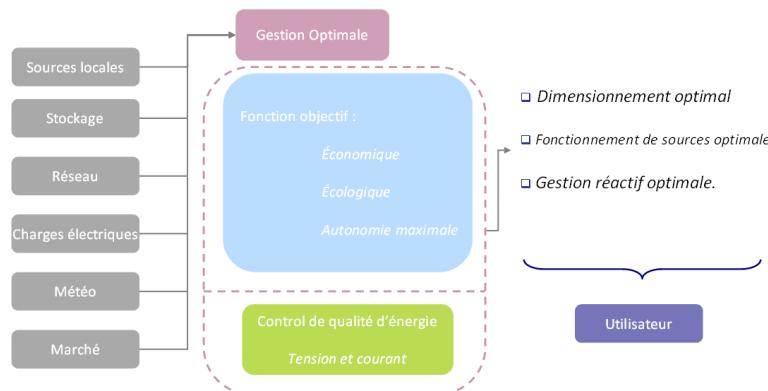


FIGURE 7.1 – Le principe de dimensionnement de systèmes bâtiments multi-sources utilisé dans la thèse

Dans l'annexe 16.1, nous présentons des méthodes avec lesquelles nous calculons des critères économiques qui déterminent si l'investissement est rentable ou pas.

7.3 Difficultés liées à la résolution de problèmes de dimensionnement de sources

7.3.1 Complexité des données nécessaires

La problématique de dimensionnement de systèmes bâtiment est très complexe étant donné le nombre de paramètres d'entrée (voir figure 7.1). En effet, le dimensionnement est une procédure influencée par plusieurs éléments comme la capacité de financement des investisseurs ou l'architecture d'un bâtiment.

La figure 7.2 montre comment la variation de prix de rachat du surplus peut affecter la rentabilité d'un système bâtiment. Dans cette figure, on présente l'évolution de la valeur nette actualisée (NPV) sur la vie de l'installation. Cette valeur donne une indication sur la viabilité financière de l'installation PV. L'axe horizontal représente la vie de l'installation en année, l'axe vertical représente la valeur nette présente en euro. Chaque courbe est la valeur nette présente pour un prix de surplus (p_s) différent. Or le prix de l'énergie du fournisseur (EDF dans cet exemple) est figé à $p_g = 0.1018 \text{ €/kWh}$. La courbe en bleu représente le cas le plus favorable car on amortit les coûts initiaux plus vite que dans les autres choix (en sept ans) : le gain en fin de vie de l'installation est plus important. Par

1. cf. annexe 16.1

2. normalement 20 ans

contre, la courbe rouge présente un cas où l'investissement n'est pas rentable (un de critères économiques est l'indice de profitabilité PI cf. annexe 16.1 formulation numéro 16.4. Si $PI > 1$, l'investissement est rentable et le taux de rentabilité est ($PI-1$), par contre si $PI < 1$, l'investissement n'est pas rentable).

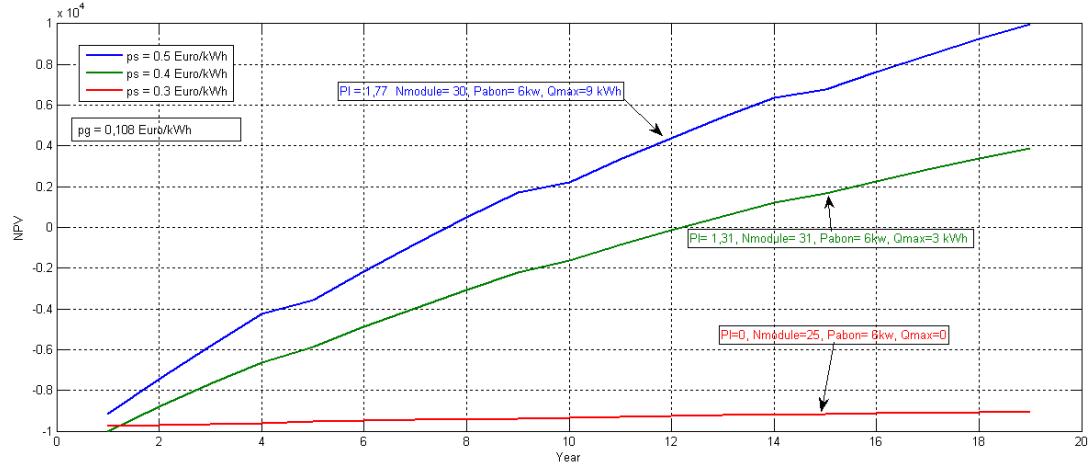


FIGURE 7.2 – Valeur nette actualisée (NPV) pour un investissement sur 20 ans pour plusieurs prix de surplus

Nous proposons le tableau 7.1 qui présente la classification que l'on propose pour les données de chaque composant selon les trois types d'hypothèses présentées dans le chapitre 1 et dans (Warkozek, Ploix, Jacomino & Wurtz 2010).

source électrique	Hypothèses spécifiques à utilisateur (HSU)	Hypothèses spécifiques aux sources (HSS)	Hypothèses spécifiques au marché de l'énergie (HSEM)
Réseau	niveau d'abonnement coefficient d'abonnement maximal tarif double ou simple		coefficient de linéarisation prix de l'énergie
Photovoltaïque	surface disponible taux de surface utilisable	coefficient de puissance crête rendement de l'onduleur PV/Onduleur taux de dimensionnement	le prix d'un module euro/kW _c le prix de l'onduleur euro/kW _c taux de subvention le prix de surplus
Batterie		taux de charge et de décharge taux maximum admissible pour la charge/décharge capacité maximum admissible taux de charge initiale rendement de la batterie	prix de kWh

TABLE 7.1 – La classification des données du problème selon les trois types d'hypothèses relatives aux sources

Les données relevant des hypothèses de type *HSU* comprennent les courbes de charges et les données météorologiques (l'ensoleillement et la température). Nous ne les avons pas ajoutées au tableau 7.1 car elles n'ont pas de lien avec les sources. Connaître le bon historique pour ces données est une question délicate car exploiter un historique très grand va induire des calculs qui peuvent dépasser la capacité du solveur. Cela nous amène à la deuxième problématique pour résoudre les problèmes de dimensionnement correspondants.

7.3.2 Nécessité d'outils de résolution appropriés

Avec l'approche habituelle de génération de problèmes de dimensionnement des sources dans le bâtiment (mono solveur), certaines difficultés apparaissent lors de la génération de problèmes : la difficulté à tester différentes configurations de bâtiments afin de les comparer et puis à adopter la configuration la plus rentable, et la capacité de solveur choisi pour résoudre le problème.

Admettons qu'un concepteur (de système bâtiment) veuille dimensionner un système bâtiment composé d'un réseau électrique et d'un ensemble des panneaux photovoltaïques, suivant une approche habituelle (étapes de l'encart 7.3.2).

Le concepteur choisit par exemple d'utiliser l'environnement d'optimisation CADES.

Il commence par modéliser les composants du bâtiment selon son expérience, puis il écrit les contraintes représentatives du réseau et celles des panneaux solaires dans un fichier de description de type *.sml, par exemple *eneco.sml* (voir figure 7.3). Ensuite, grâce à *CADES Generator*, il génère le composant logiciel *eneco.icar* qu'il va utiliser dans le module *CADES Optimizer*. Le concepteur donne alors les valeurs numériques pour les paramètres comme spécifications dans le module *CADES optimizer* et les domaines de variation des variables un par un (voir figure 7.3). Enfin, il appelle le solveur et résout le problème.

Si le concepteur a choisi l'environnement CPLEX, il doit écrire les contraintes dans une fichier *.lp comme illustré dans la figure 7.4, puis appeler le solveur CPLEX en mode interactive (cf. paragraphe 3.3.1.)

Afin d'évaluer la viabilité économique d'une autre architecture (par exemple sans le PV), le concepteur est obligé de réécrire la plupart des contraintes décrites en SML ainsi que les spécifications en XML (voir figure 7.3). Cela consomme du temps et demande un effort important.

```

EnergyBalanceEquation_11 = +1.0*batInputEnergy_BNO_11+1.0*gridEnergyVariable_GNO_11-1.0*modifiedload_11-1.0*pvSurplus_PVNO_11 ;
EnergyBalanceEquation_12 = +1.0*batInputEnergy_BNO_12+1.0*gridEnergyVariable_GNO_12-1.0*modifiedload_12-1.0*pvSurplus_PVNO_12 ;
EnergyBalanceEquation_13 = +1.0*batInputEnergy_BNO_13+1.0*gridEnergyVariable_GNO_13-1.0*modifiedload_13-1.0*pvSurplus_PVNO_13 ;
EnergyBalanceEquation_14 = +1.0*batInputEnergy_BNO_14+1.0*gridEnergyVariable_GNO_14-1.0*modifiedload_14-1.0*pvSurplus_PVNO_14 ;
EnergyBalanceEquation_15 = +1.0*batInputEnergy_BNO_15+1.0*gridEnergyVariable_GNO_15-1.0*modifiedload_15-1.0*pvSurplus_PVNO_15 ;
EnergyBalanceEquation_16 = +1.0*batInputEnergy_BNO_16+1.0*gridEnergyVariable_GNO_16-1.0*modifiedload_16-1.0*pvSurplus_PVNO_16 ;
SOCFinalAndInitialValue_BNO = +1.0*batteryStateOfCharge_BNO_00-1.0*batteryStateOfCharge_BNO_23 ;
SOCInitialization_BNO = +1.0*batteryStateOfCharge_BNO_00 ;
SoldEnergyEquation_PVNO_00 = +0.3*pvSurplus_PVNO_00-1.0*soldEnergy_PVNO_00 ;
SoldEnergyEquation_PVNO_01 = +0.3*pvSurplus_PVNO_01-1.0*soldEnergy_PVNO_01 ;
SoldEnergyEquation_PVNO_02 = +0.3*pvSurplus_PVNO_02-1.0*soldEnergy_PVNO_02 ;
SoldEnergyEquation_PVNO_03 = +0.3*pvSurplus_PVNO_03-1.0*soldEnergy_PVNO_03 ;

- <INPUT name="gridEnergyVariable_GNO_02">
- <CONSTRAINT>
  <INIT value="0.0" />
  <MAX value="6.0" />
  <MIN value="0.0" />
</CONSTRAINT>
- <INPUT name="gridEnergyVariable_GNO_03">
- <CONSTRAINT>
  <INIT value="0.0" />
  <MAX value="6.0" />
  <MIN value="0.0" />
</CONSTRAINT>
- <INPUT name="gridEnergyVariable_GNO_04">
- <CONSTRAINT>
  <INIT value="0.0" />
  <MAX value="6.0" />
  <MIN value="0.0" />
</CONSTRAINT>
- <INPUT name="gridEnergyVariable_GNO_05">
- <CONSTRAINT>
  <INIT value="0.0" />
  <MAX value="6.0" />
  <MIN value="0.0" />
</CONSTRAINT>
- <INPUT name="gridEnergyVariable_GNO_06">
- <CONSTRAINT>
  <INIT value="0.0" />
  <MAX value="6.0" />
  <MIN value="0.0" />
</CONSTRAINT>
- <INPUT name="gridEnergyVariable_GNO_07">
- <CONSTRAINT>
  <INIT value="0.0" />
  <MAX value="6.0" />
  <MIN value="0.0" />
</CONSTRAINT>
- <OUTPUT name="EnergyBalanceEquation_17">
- <CONSTRAINT>
  <FIXED value="0.0" />
</CONSTRAINT>
</OUTPUT>

```

FIGURE 7.3 – Partie du problème de dimensionnement de sources à écrire manuellement avec l'approche habituelle, et avec la syntaxe pour CADES : à gauche, les contraintes en SML et à droite, les spécifications en XML

```

gridPowerUBconstraint_GNO_17: gridEnergyVariable_GNO_17
- 3 gridSubscriptionCoeff_GNO <= 0
gridPowerUBconstraint_GNO_18: gridEnergyVariable_GNO_18
- 3 gridSubscriptionCoeff_GNO <= 0
gridEnergyVariable_GNO_19: gridEnergyVariable_GNO_19
- 3 gridSubscriptionCoeff_GNO <= 0
gridEnergyVariable_GNO_20: gridEnergyVariable_GNO_20
- 3 gridSubscriptionCoeff_GNO <= 0
gridEnergyVariable_GNO_21: gridEnergyVariable_GNO_21
- 3 gridSubscriptionCoeff_GNO <= 0
gridEnergyVariable_GNO_22: gridEnergyVariable_GNO_22
- 3 gridSubscriptionCoeff_GNO <= 0
gridEnergyVariable_GNO_23: gridEnergyVariable_GNO_23
- 3 gridSubscriptionCoeff_GNO <= 0
gridSubscriptionCoeffUBconstraint_GNO: gridSubscriptionCoeff_GNO <= 3
gridSubscriptionReductionEquation_GNO: gridSubscriptionReduction_GNO
+ gridFixedCost_GNO = 126
netPresentValueEquation:
- systemInitialCost + CashFlow_01
+ CashFlow_00 + CashFlow_02 + CashFlow_03
+ CashFlow_04 + CashFlow_05 + CashFlow_06
+ CashFlow_07 + CashFlow_08 + CashFlow_09
+ CashFlow_10 + CashFlow_11 + CashFlow_12
+ CashFlow_13 + CashFlow_14 + CashFlow_15
+ CashFlow_16 + CashFlow_17 + CashFlow_18
+ CashFlow_19 - NetPresentValue = 0
panelsProductionEquation_PVN_0_00: - panelsProduction_PVNO_00 = 0
panelsProductionEquation_PVN_0_01: - panelsProduction_PVNO_01 = 0
panelsProductionEquation_PVN_0_02: - panelsProduction_PVNO_02 = 0
panelsProductionEquation_PVN_0_03: - panelsProduction_PVNO_03 = 0
panelsProductionEquation_PVN_0_04: - panelsProduction_PVNO_04 = 0
panelsProductionEquation_PVN_0_05: - panelsProduction_PVNO_05 = 0
panelsProductionEquation_PVN_0_06: - panelsProduction_PVNO_06
+ 0.000122237091842272 optimalModulesNumber_PVNO
= 0
panelsProductionEquation_PVN_0_07: - panelsProduction_PVNO_07
+ 0.00359202044364653 optimalModulesNumber_PVNO
= 0
panelsProductionEquation_PVN_0_08: - panelsProduction_PVNO_08
+ 0.0115914664779995 optimalModulesNumber_PVNO
= 0
panelsProductionEquation_PVN_0_09: - panelsProduction_PVNO_09
+ 0.0196710932494581 optimalModulesNumber_PVNO
= 0
panelsProductionEquation_PVN_0_10: - panelsProduction_PVNO_10
+ 0.0281236941590352 optimalModulesNumber_PVNO
= 0

```

FIGURE 7.4 – Partie du problème de dimensionnement de sources à écrire manuellement avec l'approche habituelle, et avec la syntaxe AML pour CPLEX

7.4 Capitalisation du savoir faire pour la problématique de dimensionnement des sources

7.4.1 Besoin d'outil métier

Au sein de G2Elab³, les travaux de Do (2010) et de Dupeloux (2006) ont alimenté le développement d'outils métiers pour le dimensionnement des machines électriques et d'éléments électroniques de puissance, comme par exemple l'outil Reluctool (Do 2010). Le travail illustré dans cette partie s'inscrit dans la même axe de développement d'outils métiers mais pour le système bâtiment.

L'outil métier qu'on vise doit répondre aux problématiques précédentes. La formulation de problèmes d'optimisation (qui représente le problème de dimensionnement) doit être facile, pour que l'utilisateur d'outil puisse composer le système bâtiment aisément. De plus, la formulation ne doit pas être limitée à un seul environnement de résolution.

Pour cela, nous avons besoin de mettre en évidence trois types d'acteurs concernés par la problématique de dimensionnement. Ces trois acteurs ont différents types de savoir faire, et ils interviennent dans des temporalités différentes. Dans la figure 7.5, on met en évidence deux savoirs faire nécessaires qui sont :

1. le savoir faire de modélisation des sources et des charges (savoir faire métier),
2. la savoir faire de retranscription des modèles dans un code exécutable par un solveur (savoir faire informatique).

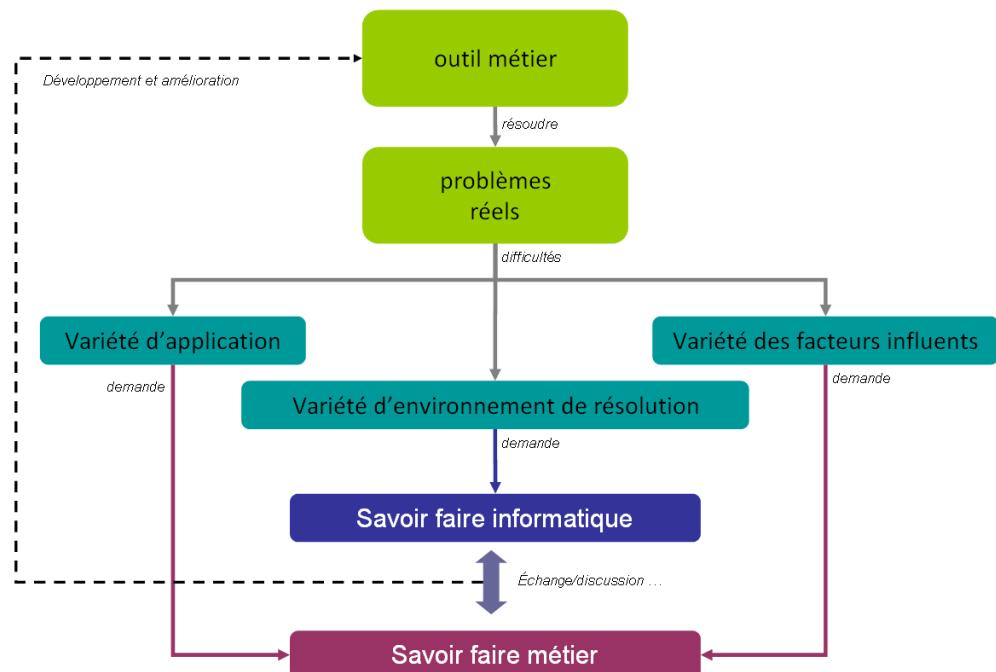


FIGURE 7.5 – Le besoin d'outil métier et les savoirs faire nécessaires

Dans la suite de ce chapitre, nous présentons comment nous pouvons implémenter l'approche de génération automatique multi solveur en prenant en compte les savoirs faire

3. en partenariat avec CEDRAT

précédents. Nous commencerons par spécifier les trois acteurs majeurs, aux rôles distincts. Nous illustrons ensuite les rôles de ces acteurs pour l'approche proposée.

7.4.2 Les métiers et les acteurs concernés par les problématiques de dimensionnement

Dans le chapitre précédent, nous avons fait la différence entre l'utilisateur et le développeur d'application. Dans ce chapitre nous affinons d'avantages les acteurs en identifiant trois avec des métiers différents. Ces trois acteurs concernés par le dimensionnement des sources dans le bâtiment sont (voir figure 7.7) :

1. le *développeur logiciel* : il est celui qui a un savoir faire informatique. Il développe :
 - la structure du générateur de problèmes et les projecteurs vers différents environnements de résolution. Concrètement, il développe le package Java illustré précédemment dans la figure 5.10 et les projecteurs présentés dans le paragraphe 5.3.
 - un éditeur de type : ceci est un moyen pour que l'acteur suivant (le *concepteur de type de modèles*) puisse développer des types (modèles de sources) en conformité avec le générateur et les projecteurs précédents. Parfois cet éditeur s'appelle un SDK (Software Developpement Kit).
 - un éditeur de systèmes pour le concepteur de systèmes bâtiments qui permet de paramétrier les types de modèles et de les assembler pour construire le système bâtiment.
2. le *concepteur de type de modèles* : il est celui qui a le savoir-faire en modélisation, il utilise l'éditeur de type (ou le SDK) construit par le *développeur de logiciel* pour construire un type avec les modèles analytiques (sans instancier les paramètres) de composants de bâtiments, les bons ports et le bon concepteur graphique⁴.

Par exemple, il modélise les contraintes sur l'état de charge de la batterie (formulation numéro 7.2) :

$$Q_{min} \leq SOC_t \leq Q_{max} \quad (7.2)$$

en appelant :

- Q_{max} : est la capacité maximale de la batterie.
- Q_{min} : est la capacité minimale de la batterie.

Puis il prépare un concepteur graphique celui que la figure 7.6 présente.

3. le *concepteur de systèmes bâtiments* : il est celui qui a la connaissance du système bâtiment et qui connaît mieux que les autres acteurs les technologies du domaine. Il utilise l'application logicielle, et notamment les concepteurs graphiques, développés par le *modélisateur* pour spécifier les paramètres et assembler le système bâtiment, il utilise aussi les projecteurs développés par le *développeur logiciel* pour faire des analyses de sensibilité dans différents solveurs.

Dans ce contexte, les travaux présentés dans ce chapitre proposent une solution pour séparer et faciliter les tâches des différents acteurs, pour mieux organiser l'échange de savoirs-faire entre les acteurs métiers qui peuvent ainsi se spécialiser. Une fois qu'un acteur

4. c'est une interface graphique

7.4. Capitalisation du savoir faire pour la problématique de dimensionnement des sources



FIGURE 7.6 – Le concepteur de bâtiments utilise l’application logicielle avec son savoir faire pour concevoir le bâtiment

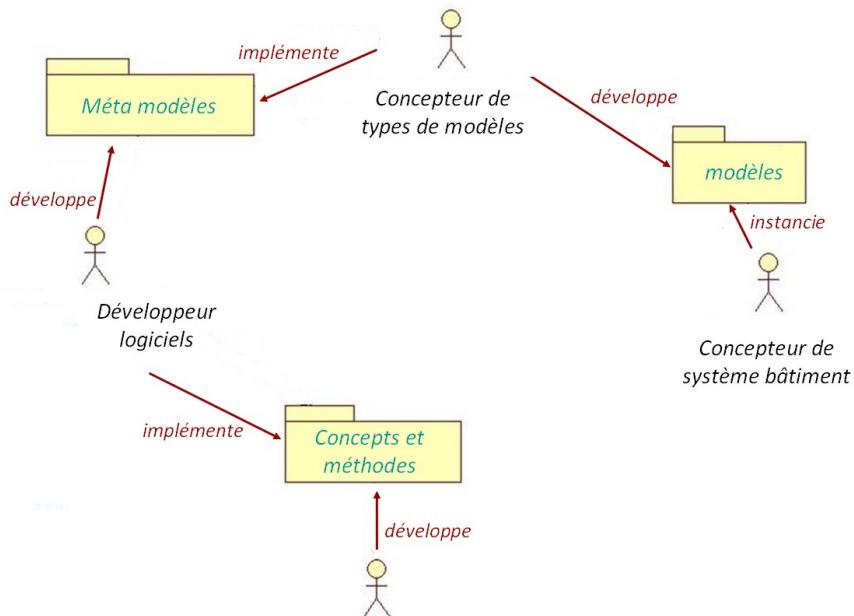
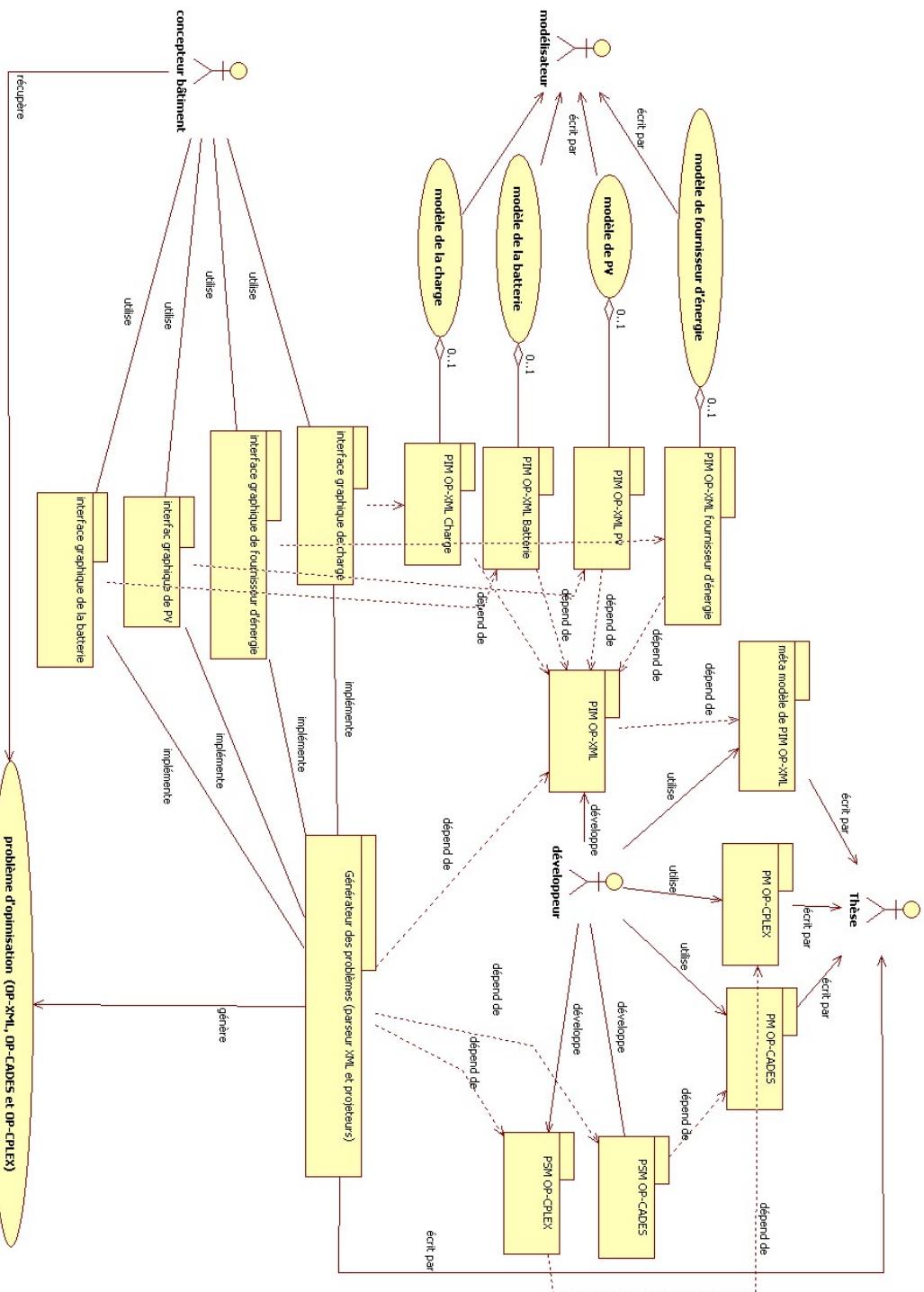


FIGURE 7.7 – Les trois acteurs principaux du dimensionnement des systèmes bâtiments multi-source

fait les tâches précédentes, son rôle est terminé. Le *développeur logiciel* utilise notre plate-forme pour générer des projecteurs exploitant des modèles de niveaux M2 (méta modèle de PIM OP-XML, PM OP-CADES et PM OP-CPLEX) qui permettent de transcrire des modèles vers différents environnements de résolution. Ensuite, le *concepteur de type de modèles* peut créer des types de modèles de niveau M1 et les concepteurs graphiques associés. Enfin, le *concepteur de systèmes bâtiments* instancie ces types pour obtenir des modèles de niveau M0 (voir figure 7.8) qui peuvent être projetés vers différents environnements de résolution. Dans la suite du chapitre, à chaque fois que nous présentons une partie de la solution, nous rappellerons l’acteur concerné par cette partie de solution.

FIGURE 7.8 – Processus de dimensionnement des sources et les rôles des acteurs avec l'approche de génération automatique multi solveur



7.5 Mise en oeuvre pour un dimensionnement énergétique-économique de sources

Les notions de ports et de contraintes globales illustrées dans les paragraphes 2.3 sont utilisées pour générer automatiquement le OP-XML complet du problème de dimensionnement. La figure 7.9 illustre que chaque source dans le bâtiment a deux types de ports : port énergétique (représenté par un triangle vert) et port économique (représenté par un triangle jaune). Ces ports permettent au générateur de problèmes d'ajouter les participations des composants du système bâtiment dans deux contraintes globales OP-XML qui sont : le bilan énergétique et le bilan économique (qui prend en compte les contraintes de calcul de flux de trésorerie, de l'investissement initial, et de valeur nette actualisée).

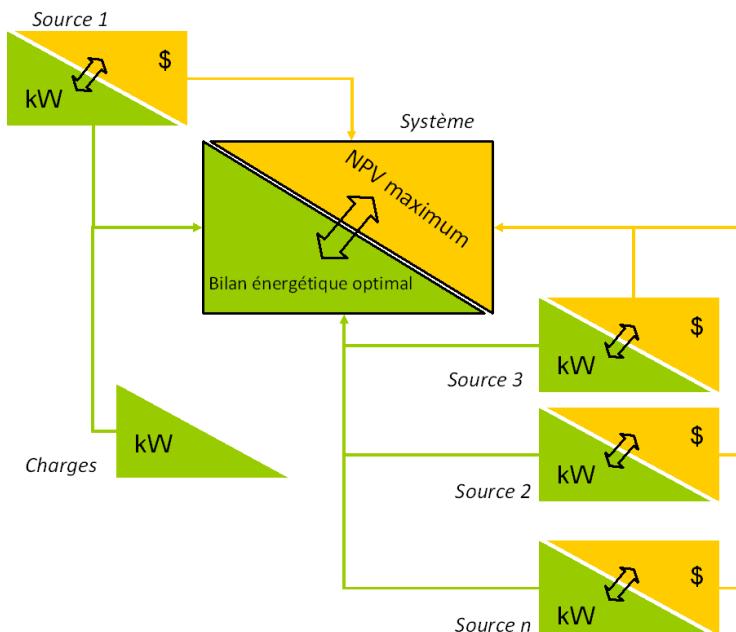


FIGURE 7.9 – Équilibre énergétique et économique

La notion de ports est implémentée dans le SDK (*Software Development Kit*) que nous avons développé (étant un acteur *développeur logiciel*) comme une méthode java qui doit être implémentée par le *concepteur de type de modèles* dans chaque classe qui représente un type (un modèle de sources). Pour les ports énergétiques nous avons appelé cette méthode `getEnergyVariables()`. Elle renvoie les variables de ce type utilisées dans le bilan énergétique global du système bâtiment. Pour les ports économiques, nous avons utilisé une méthode `getInitialCost()` qui renvoie une variable représentant le coût initial de ce type.

Pour les contraintes locales de chaque type de modèles, le SDK permet au *concepteur de type de modèles* d'ajouter ces contraintes directement au OP-XML. Cela revient au concept illustré dans le paragraphe 2.2 où chaque composant de bâtiment peut ajouter sa contribution au problème complet.

7.5.1 Partie développement logiciel (pour le *développeur de logiciels*)

Il s'agit de développer les classes Java (ou le SDK) qui permettent de générer le problème OP-XML complet et les OP-CADES, OP-CPLEX. Nous avons présenté ceci dans le paragraphe 5.2.5. Cet acteur développe aussi les projecteurs vers CPLEX et CADES (voir paragraphe 5.3).

Il développe aussi l'éditeur de types. Nous avons défini trois interfaces Java pour les trois types utilisés dans ce travail : une pour le type fournisseur d'énergie, une pour le type PV et une pour le type batterie électrique (voir figure 7.10) avec deux interfaces pour les données météorologiques et la charge électrique. Le *concepteur de types de modèles* va implémenter ces trois interfaces pour ensuite générer les types.



FIGURE 7.10 – Le *développeur logiciel* développe les interfaces de type de modèles pour que le *concepteur de types de modèles* les implémente avec ses modèles spécifiques

Il développe un éditeur de systèmes bâtiments, il s'agit d'un ensemble de paquetages dans lesquels les concepteurs graphiques (interfaces) développés par le *concepteur de types de modèles* peuvent être groupés dans une seule interface, comme illustré la figure 7.11.

Une fois que toutes ces tâches sont terminées, cet acteur n'intervient que s'il faut ajouter un nouveau type qui n'existe pas dans les interfaces de types de modèles.

7.5.2 Partie modélisation (pour le *concepteur de types de modèles*)

7.5.2-1 Ajoût des charges électriques et les données météorologique au PIM OP-XML

Bien que le but de l'approche énergétique-économique soit de dimensionner les sources électriques, nous nous intéressons aussi à la représentation simplifiée de l'effet du pilotage des charges car cela a un impact sur le dimensionnement optimal des sources. Dans (Long 2007), nous trouvons trois types de charge. Nous allons prendre en compte un seul type. Il

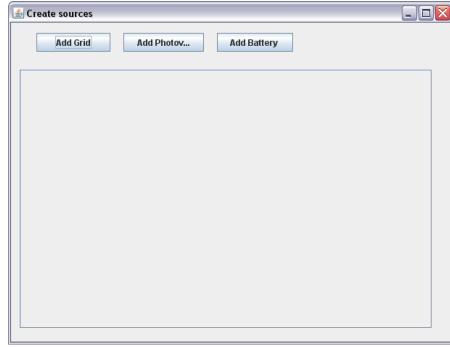


FIGURE 7.11 – Le *développeur logiciel* développe un éditeur de système bâtiment qui permet de grouper les concepteurs graphiques développés par *concepteur de types de modèles* pour les types de modèles

s’agit d’une charge décalable (cela veut dire que nous pouvons décaler le démarrage de la charge de l’instant prévu ‘souhaité’ par l’utilisateur vers celui trouvé par l’optimisation). La charge électrique est donc représentée par deux types de données estimées sur la période de l’étude⁵). La première partie représente la charge totale sur 24h, la seconde représente les charges décalables sur 24h (voir la figure 7.12 et 7.13).

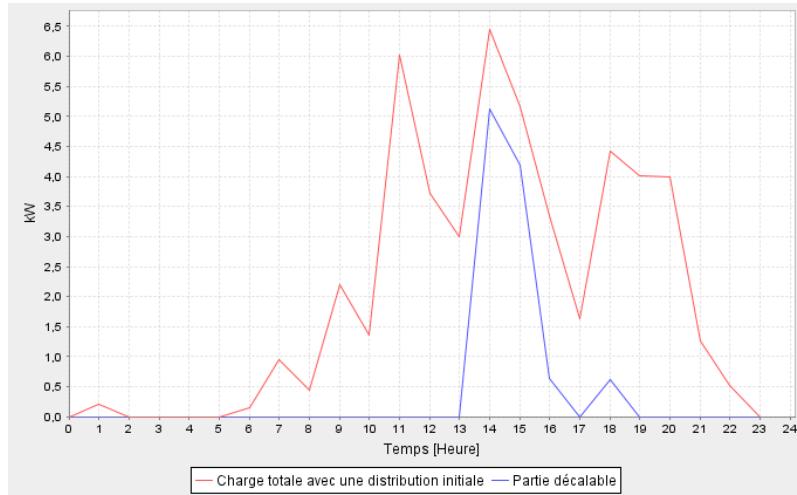


FIGURE 7.12 – La charge totale et la partie décalable avec une distribution initiale

Nous calculons l’énergie totale et l’énergie décalable sur 24h comme :

$$\begin{aligned} a_{13} &= \sum_{t=0}^{t=StudyPeriod} a_{15}(t) \\ a_{14} &= \sum_{t=0}^{t=StudyPeriod} a_{16}(t) \end{aligned} \quad (7.3)$$

sachant que :

- a_{13} est l’énergie totale de la charge sur la période d’étude en kWh.
- a_{14} est l’énergie de la partie décalable sur la période d’étude en kWh.
- a_{15} est la répartition initiale de la charge totale à l’instant t en kWh.

5. Des méthodes d’estimation et de gestion des charges se trouvent dans (Boëda 2009) et dans (Bartles & Fiebig 1996)

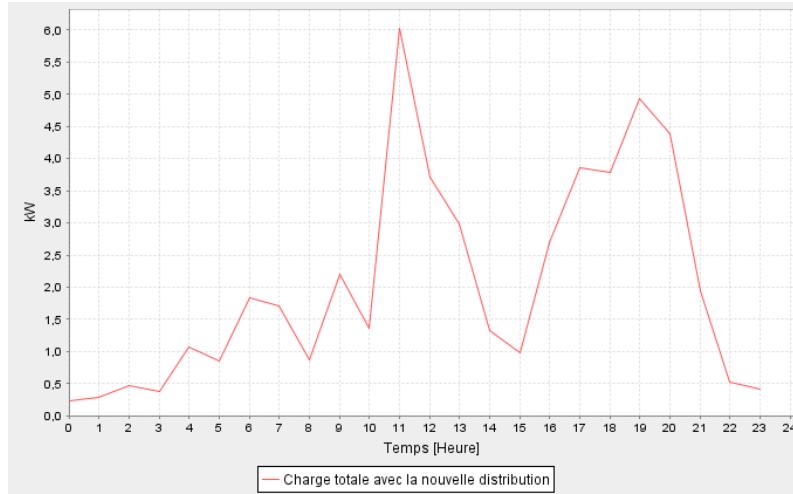


FIGURE 7.13 – La charge totale avant et après la nouvelle distribution

- a_{16} est la répartition initiale de la charge décalable à l'instant t en kWh.
- Le modèle utilisé est identique à 7.4 :

$$\begin{aligned}
 CC1 \quad & x_{14}(t) + x_{13}(t) = x_{12}(t) \\
 CC2 \quad & \sum_{t=0}^{t=StudyPeriod} x_{13}(t) = a_{14} \\
 CC3 \quad & \sum_{t=0}^{t=StudyPeriod} x_{12}(t) = a_{13} \\
 CC4 \quad & x_{13}(t) \leq a_{14}(t) \\
 CC5 \quad & x_{14}(t) \leq \infty
 \end{aligned} \tag{7.4}$$

sachant que :

- x_{12} est l'énergie de la charge à l'instant t pendant un pas de temps kWh.
- x_{13} est l'énergie de la partie décalable de la charge à l'instant t en kWh.
- x_{14} est l'énergie de la partie non décalable de la charge à l'instant t en kWh.
- CC1,CC2,... sont les noms des contraintes.

Le *concepteur de types de modèles* implémente l'interface de type charge (appelé *AbstractLoad* dans la figure 7.10) préparée auparavant par le *développeur logiciel* pour générer un type de modèle de charges électriques dans l'outil métier. Ce type contient les contraintes en 7.4, les a_{15} et a_{16} sont les paramètres à instancier par le *concepteur de système bâtiment*. En choisissant via l'interface graphique (voir figure 7.14) un ou plusieurs mois type. Le *concepteur de systèmes bâtiments* pourra instancier les courbes de charges correspondants et aussi les données météorologiques (ensoleillement et température) à partir de valeurs données par le *concepteur de types de modèles* dans une fichier *.txt.

Concrètement, il utilise le SDK pour écrire une classe java qui représente un sous problème dans le OP-XML complet (Ce sous-problème est défini par les contraintes de charge électrique tel que illustré dans le modèle 7.4). Pour cette raison, il utilise le PIMgenerator (du chapitre précédent). Il écrit une classe Java en prenant en compte que la variable x_{12} est un port énergétique pour le modèle de charge, alors que les contraintes CC1,CC2... CC5 sont des contraintes locales. Le SDK va intégrer le port de charge dans l'équation de bilan énergétique et le sous problème lié à la charge dans le OP-XML complet (voir figure 7.15).

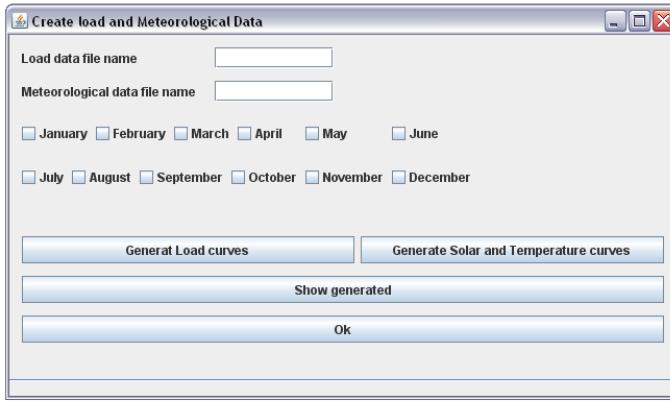


FIGURE 7.14 – Le concepteur graphique pour ajouter la charge électrique et les données météorologiques. Il est développé par le *concepteur de types de modèles* et à utiliser par le *concepteur de systèmes bâtiments*

7.5.2-2 Ajout du modèle du fournisseur d'énergie électrique au PIM OP-XML

Il y a deux types de dépenses liées au réseau de distribution. D'abord le coût de l'abonnement du fournisseur d'énergie qui représente un coût fixe dans le temps, puis le coût de consommation qui varie selon le temps et le prix de l'énergie. L'acteur *concepteur de types de modèles* formule cette dépense comme illustré dans la formule 7.5.

$$C_{grid} = C_{abonnement} + C_{consommation} \quad (7.5)$$

$$C_{abonnement} = P_{abon} \times T_{abon} \quad (7.6)$$

sachant que :

- P_{abon} est le niveau d'abonnement en kW.
- T_{abon} est le tarif d'abonnement avec le fournisseur en euro/kW.

$$C_{consommation} = \sum_{t=0}^{t=periode} consommation(t) \times p_g(t) \quad (7.7)$$

Les systèmes multi-sources offrent des degrés de liberté qui permettent d'effectuer des économies sur la consommation. Ces sources correspondent à des moyens de production locaux comme des modules PV, ou à des moyens de stockage comme une batterie⁶ (voir équation 7.8).

$$R_{localProduc} = \sum_{t=0}^{t=periode} P_{local}(t) \times p_g \quad (7.8)$$

sachant que P_{local} est l'énergie produite par les sources locales.

6. Dans le cas où il n'y a que le réseau, le bâtiment achète toute sa consommation du fournisseur.

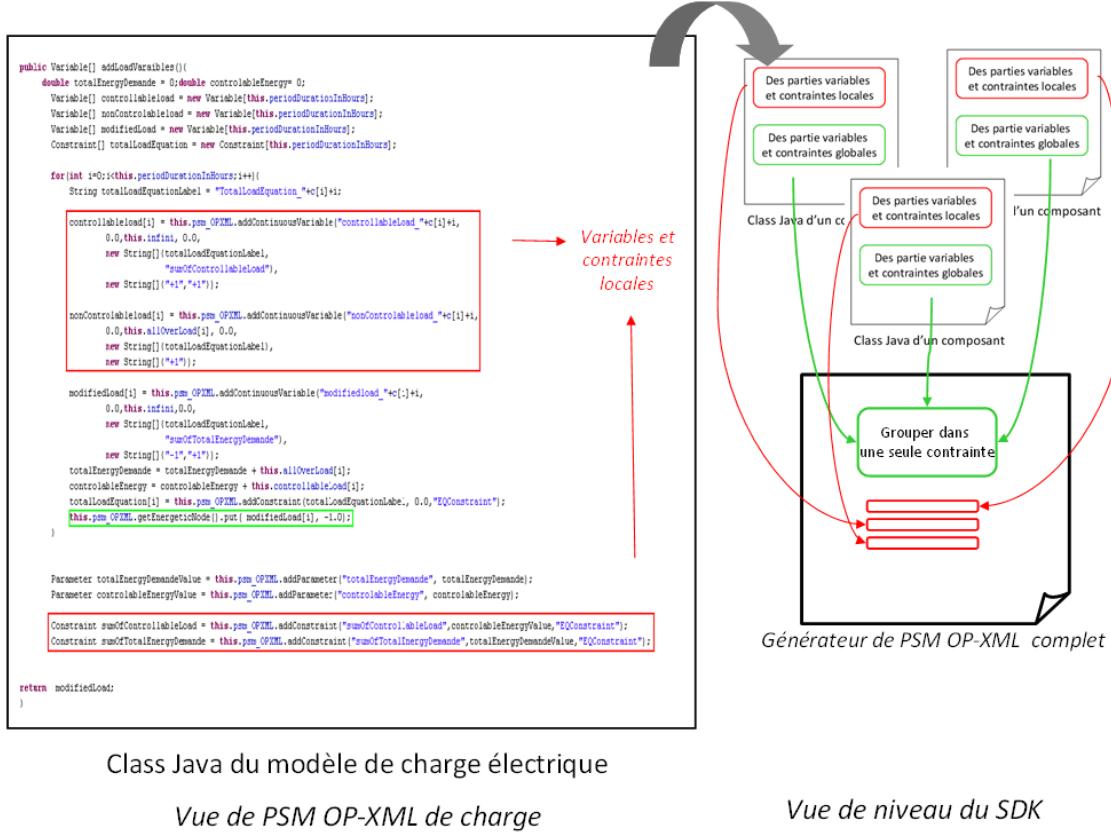


FIGURE 7.15 – Partie de classe Java du modèle de charge électrique qui doit être fait par le *concepteur de types de modèles*. Les contraintes locales s’ajoutent automatiquement dans l’OP-XML complet, les ports énergétiques sont groupés dans une seule contrainte globale par le SDK

Le modèle énergétique du fournisseur est un ensemble de contraintes qui limitent la consommation d’énergie dans le cadre de bornes fixées par le *concepteur de systèmes bâtiments*. Ce modèle est illustré dans (7.9) :

$$\forall t \in 0..Studyperiod$$

$$\begin{aligned}
CG1 \quad & a_1 \times x_2 \leq a_2 \\
CG2 \quad & x_1(t) \leq a_2 \\
CG3 \quad & x_3(t) = a_3(t) \times x_1(t) \\
CG4 \quad & x_1(t) = (1 - x_7(t)) \times Sup(x_1) \\
CG5 \quad & x_1(t) \geq 0 \\
CG6 \quad & Sup(x_1) \leq a_2
\end{aligned} \tag{7.9}$$

sachant que :

- a_1 : le coefficient d’abonnement est une variable entière (exemple : 1,2,3) qui représente les trois niveaux d’abonnement avec le réseau électrique en France (3,6,9 kVA)⁷.
- a_2 : le niveau maximum d’abonnement défini par le concepteur de bâtiment.

7. pour le type résidentiel et tertiaire

- a_3 : le prix de l'énergie du fournisseur (p_g).
- x_1 : l'énergie du fournisseur consommée en kWh.
- x_2 : le niveau d'abonnement (variable de dimensionnement).
- x_3 : le coût de la consommation de l'énergie du fournisseur ($consommation(t)$).
- x_7 : une variable de commande pour gérer la revente d'énergie au fournisseur ; comme nous allons le montrer dans la suite, nous n'autorisons pas l'achat et la revente en même temps. Donc elle est de type binaire (0-1), égale à 1 quand on revend de l'énergie vers le fournisseur, et égale à 0 sinon. .
- $Sup(x_1)$: la limite supérieure de x_1 . La contrainte $CG6$ permet à trouver un nouveau niveau d'abonnement (s'il est réalisable) avec le fournisseur inférieur de ceci donné par le concepteur de bâtiment a_2 . Nous pouvons contrôler l'achat (éventuellement, comme nous allons voir dans la suite, la revente) avec la contrainte $CG4$:
 - si $x_7=0 \Rightarrow x_1 \leq Sup(x_1) \Rightarrow x_1 \neq 0$ (cas d'achat).
 - si $x_7=1 \Rightarrow x_1 = 0$ (cas de revente).

Par analogie avec les composants du système bâtiment type charge électrique, le *concepteur de types de modèles* utilise l'interface de type (*AbstractGrid* dans la figures 7.10) pour définir ce type de source électrique. Il écrit, donc une classe qui représente le sous problème modélisé en 7.9 (cela correspond à 'PIM OP-XML Fournisseur d'énergie' dans la figure 7.8) en respectant la structure de méta modèle de PIM OP-XML. Dans cette classe il déclare les ports de ce modèle. Le port énergétique dans ce modèle est la variable x_1 et le port économique est la variable C_{grid} de l'équation 7.5. En effet, le générateur de problème pourra ajouter les contraintes de ce composant dans le OP-XML complet si le *concepteur de système bâtiment* l'ajoute pendant la phase de composition de système. Par exemple la figure 7.16 illustre une partie de OP-XML complet liée au bilan économique, dans ce noeud on trouve le coût initial du système complet comme une addition des coûts de chaque composant.

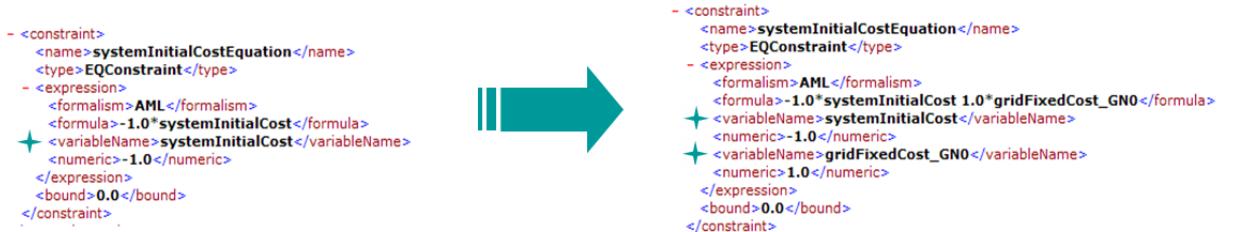


FIGURE 7.16 – Partie de l'OP-XML du problème complet généré automatiquement : le noeud économique dans lequel il y a la variable de coût initial du système avant l'ajout des composants de type fournisseur d'énergie (à gauche) et après l'ajout (à droite)

L'acteur *concepteur de types de modèles* prépare le concepteur graphique de ce type de modèles qui permet au *concepteur de système bâtiment* de paramétriser le modèle (en donnant des valeurs numériques pour les a_1 , a_2 et a_3) et puis de l'ajouter au réseau de système. La figure 8.5 illustre un exemple de concepteur graphique pour le type fournisseur d'énergie électrique.

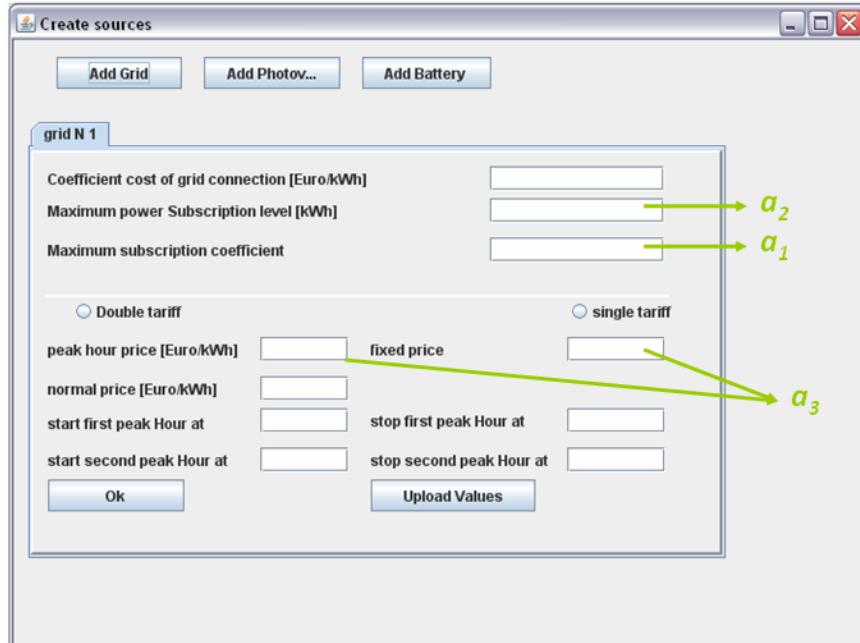


FIGURE 7.17 – L’interface graphique à utiliser par le *concepteur de systèmes bâtiments* pour ajouter le fournisseur d’électricité

7.5.2-3 Ajoût du modèle de panneaux solaires au PIM OP-XML

Depuis dix ans, le prix des modules solaires ne cesse de diminuer comme on le voit sur la figure 7.18 de ([Solar Energy Research and Consultancy 2010](#)). Même avec un prix autour de 4 euros pour le Wc, l’investissement dans les panneaux solaires représente 60% de l’investissement global ([Okunski 2008](#)). Choisir un compromis énergétique-économique consiste à trouver le nombre de panneaux solaires optimal qui minimise le coût du système PV et maximise la valeur nette actualisée.

Le *modélisateur* écrit l’équation 7.10 qui présente le coût initial du système PV (ceci est une contrainte locale) :

$$C_{pvs} = C_{pv} + C_{ond} \quad (7.10)$$

Sachant que :

$$C_{pv} = N_{pv} \times p_m \quad (7.11)$$

$$C_{ond} = P_{ond,crete} \times p_{ond} \quad (7.12)$$

avec :

- N_{pv} : le nombre de modules PV.
- p_m : le prix d’un module PV en euro/kw_c.
- $P_{ond,crete}$: la puissance crête de l’onduleur.
- p_{ond} : le prix de l’onduleur [euro/kw_c].
- C_{ond} : le coût de l’onduleur [euro].

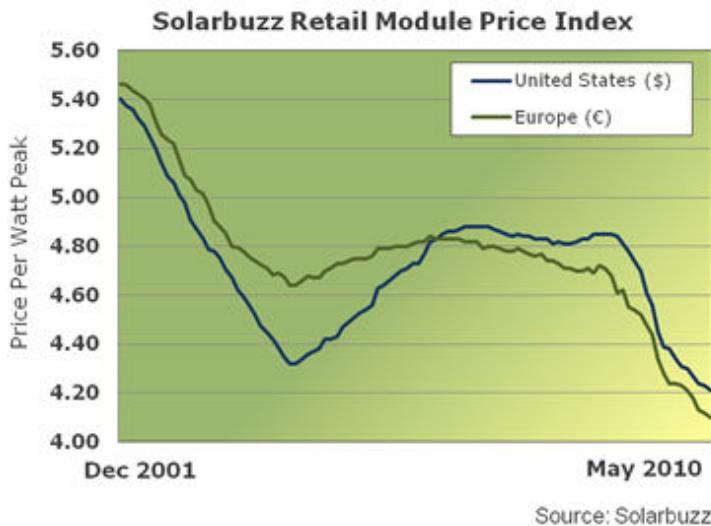


FIGURE 7.18 – Aperçu de la variation du prix des modules solaires depuis 2001

La puissance crête de l'onduleur dépend de celle des modules PV. En effet, nous pouvons calculer la puissance crête des modules PV à partir du nombre de modules et du type d'intégration dans le bâtiment comme présenté dans l'équation 7.13 (Zirngibl 2008). Il s'agit donc d'une autre contrainte locale. La puissance crête de l'onduleur est calculée comme un ratio (r_s) de la puissance crête de l'ensemble de PV. La valeur de r_s dépend de la technologie de modules PV et du type d'onduleur⁸ (Notton et al. 2009), ce ratio est illustré par l'équation (7.14) :

$$P_{pv,crte} = N_{pv} \times S_{pv} \times C_{pcrete} \quad (7.13)$$

sachant que :

- S_{pv} : est la surface du module PV.
- C_{pcrete} : est le coefficient de puissance crête. La valeur de ce coefficient dépend de type d'intégration de PV (en fonction de l'orientation)⁹.

$$r_s = \frac{P_{pv,crte}}{P_{ond,crte}} \quad (7.14)$$

où $P_{pv,crete}$, est la puissance crête du PV.

r_s peut être considéré comme un paramètre d'entrée qui permet de calculer le dimensionnement de l'onduleur ainsi que le coût du système PV. Le *concepteur de types de modèles* peut instancier l'interface de type PV donnée par l'acteur *développeur logiciel* (appelé *AbstractPhotovoltaicModel* dans la figure 7.10) et définir plusieurs types de modèles de PV selon la technologie de fabrication (polycristallin, nanocristallin etc...) donc des r_s différents, puis il donne ces choix de types de PV au *concepteur de systèmes bâtiments*.

8. pour des modules PV polycristallin, nanocristallin et CIS (Copper Indium Selenium ou *thin film*), les valeurs de r_s sont entre 0.68-1.14, alors qu'elles sont moins élevées pour les modules PV amorphe, typiquement entre 0.67-1.04 selon le type d'onduleur

9. Dans (Juquois 2002) des valeurs en guise d'exemple de cas PV connecté au réseau sont données (il vaut 65% en France et métropolitaine et 80% en DOM/TOM).

Dans cette mise en oeuvre, nous avons défini un seul type de PV en instanciant l'interface *AbstractPhotovoltaicModel* de la figure 7.10 avec une valeur de $r_s = 0.9$.

Un système multi-source est conçu pour revendre le surplus d'énergie afin d'amortir les capitaux initiaux. Selon la politique énergétique du pays, le surplus pourrait être la production solaire brute ou l'énergie de la décharge de la batterie (en France, il est interdit aujourd'hui de revendre l'énergie stockée dans la batterie). Dans la recette, nous aurons deux parties, une qui vient de la revente de l'énergie $R_{surplus}$ (contrainte locale cf. paragraphe 2.2) et l'autre de la consommation de l'énergie produite par des sources locales $R_{localProduc}$ (une contrainte globale calculée précédemment avec 7.8) :

$$R = R_{surplus} + R_{localProduc} \quad (7.15)$$

Supposons que les sources soient contrôlables suivant des décisions de type ON/OFF sur le démarrage des sources¹⁰. Lorsque la variable associée à la commande est à ON. La recette due au surplus $R_{surplus}$ pour un prix de rachat p_s est calculée par :

$$R_{surplus} = P_{surplus} \times p_s \quad (7.16)$$

Dans certains pays comme l'Allemagne, les particuliers peuvent se faire rembourser pour l'énergie produite et consommée localement (autoconsommation). Alors la recette économisée grâce aux sources locales¹¹ est donnée par l'équation 7.8, P_{local} peut être donc calculé par :

$$P_{local} = (P_{pv} - P_{surplus} + P_{dchg}) \quad (7.17)$$

Pratiquement, cette équation demande le montage de deux compteurs d'énergie sur la sortie du PV : un vers le fournisseur, l'autre vers le tableau central du bâtiment (figure 7.19). P_{dchg} représente l'énergie de la décharge de la batterie (voir dans le paragraphe suivant).

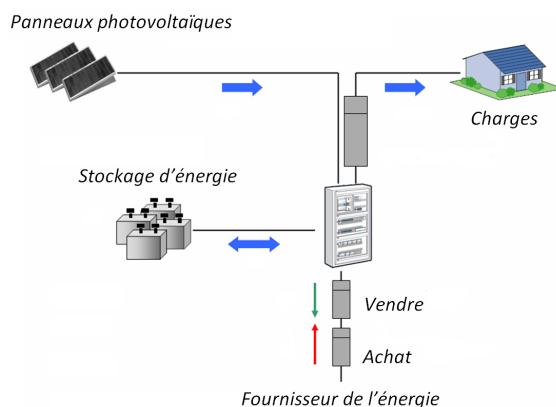


FIGURE 7.19 – Deux compteurs énergétiques pour calculer la revente et l'achat

Le modèle énergétique du PV à faire par le *modélisateur* est :

10. Ce sont les commandes des disjoncteurs qui ouvrent ou ferment les interrupteurs des sources

11. en prenant en compte la décharge de la batterie P_{dchg}

$$\begin{aligned}
CPV1 : \quad & x_4(t) = a_4 \times x_8(t) \\
CPV2 : \quad & x_5 = \sum_{t=0}^{t=studyPeriod} x_6(t) \\
CPV3 : \quad & x_6(t) = x_{10}(t) - x_8(t) \\
CPV4 : \quad & x_8(t) \leq (1 - x_7(t)) \times (a_5 \times a_6 \times a_7 \times a_8 / S_{pv}) \\
CPV5 : \quad & x_{10}(t) = a_5 \times a_6(t) \times x_9 \\
CPV6 : \quad & x_9 \leq a_7 \times a_8 / S_{pv} \\
CPV7 : \quad & x_9 \geq 0 \\
CPV8 : \quad & x_8 \geq 0
\end{aligned} \tag{7.18}$$

avec :

- a_4 : le prix du surplus en euro/kWh (p_s).
- a_5 : le rendement de l'onduleur.
- a_6 : la production d'un seul module PV en kW. Ce paramètre doit être calculé en fonction de l'ensoleillement, et du type de panneaux solaires présents dans le système ¹² et de la surface disponible pour installer les panneaux.
- a_7 : la surface totale disponible pour installer les panneaux.
- a_8 : pourcentage qui représente la surface utile en prenant en compte l'effet d'ombrage (une arbre, un autre bâtiment) (Notton et al. 2010).
- S_{pv} : la surface d'un module PV.
- x_4 : la recette de la vente du surplus ($R_{surplus}$) à l'instant (t).
- x_5 : l'énergie produite par le PV et consommée localement dans la période.
- x_6 : l'énergie produite par le PV sur un pas de temps (t) et consommée localement.
- x_7 : la variable de commande de l'achat du surplus voir 8.4.1-1 à l'instant (t).
- x_8 : le surplus en kW ($P_{surplus}$) à l'instant (t).
- x_9 : le nombre de panneaux solaires.
- x_{10} : la production totale de PV (P_{pv}) à l'instant (t).

Le contrôle de revente du surplus au fournisseur se fait par la contrainte CPV4 :

Si $x_7=0 \implies x_8 = 0$ (cas d'achat $x_1 \neq 0$).

si $x_7=1 \implies x_8 \neq 0$ (cas de revente).

Le *concepteur de types de modèles* écrit une classe java dans laquelle on trouve les contraintes 7.18 du module PV (PIM OP-XML PV dans la figure 7.8). Il définit dans cette classe les ports pour le module PV. Les ports économiques sont le coût d'un module 7.10 sur l'investissement initial du système et le gain économique possible à obtenir grâce à la revente du surplus 7.15. Le port énergétique est la production des panneaux et le surplus possible. En conséquence, cela permettra au générateurs de problème l'intégration du PIM du PV afin de compléter le modèle OP-XML du problème.

Le *concepteur de types de modèles* prépare un concepteur graphique pour dimensionnement d'installation photovoltaïque comme le montre la figure 7.20. Le *concepteur de système bâtiment* peut paramétriser le modèle en donnant des valeurs numériques pour a_4 , a_5 , a_6 , a_7 , a_8 , le prix d'onduleur p_{ond} et le prix de module p_m ¹³

12. dans notre étude, ce paramètre est déduit du type de module PV : Module Solar-Fabrik Série SF 150/2A

13. le *coefficient of peak power* et le *total surface* et *useful surface rate* sont des données pour calculer la puissance crête de panneaux selon le standard CEN EN 15316-4-6 (Zirngibl 2008) (cf. équation 7.13). Le *concepteur de bâtiment* peut aussi choisir un type de PV parmi plusieurs en choisissant le numéro de série *Serial number*. S'il y a une subvention donné au prix de module, le *concepteur de bâtiment* ajoute sa

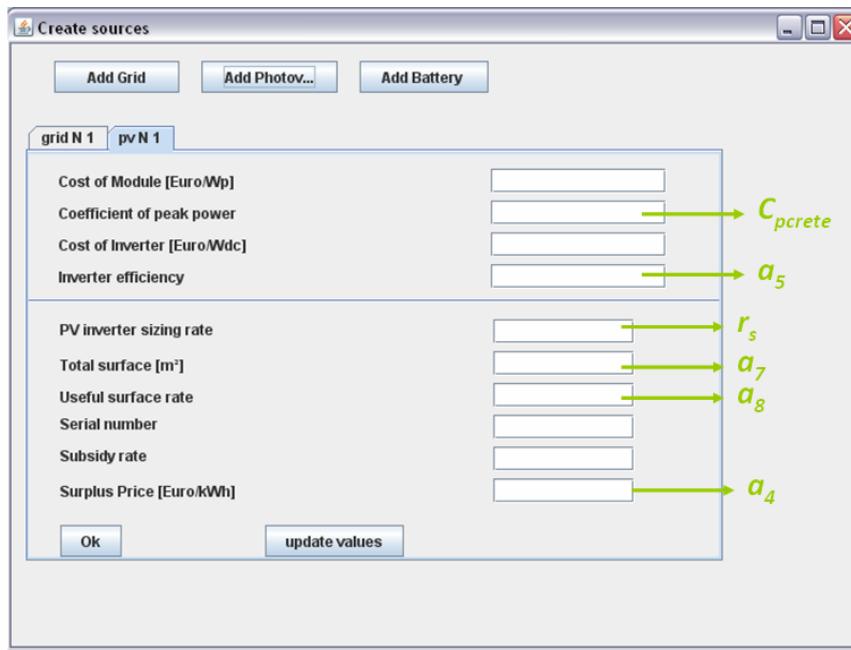


FIGURE 7.20 – Le concepteur graphique développé par le *concepteur de types de modèles* est à utiliser par le *concepteur de bâtiment* pour instancier le PV

7.5.2-4 Ajoût du stockage électrique au PIM OP-XML

La batterie a un rôle particulier dans les systèmes multi-sources. Elle est un composant coûteux qui permet d'ajuster le bilan énergétique. Elle donne un degré de liberté au pilotage pour planifier les sources et les charges. Souvent le stockage d'énergie est associé aux systèmes PV isolés. Néanmoins, dans notre application, le stockage sert à d'autres objectifs : maximiser le gain économique quand le calcul ne favorise pas la consommation d'énergie du réseau de distribution et encourage la revente de toute la production PV.

Pratiquement, la batterie subit différents cycles de charge et de décharge, ce qui provoque une réduction de son efficacité (ou vieillissement) avec le temps. Pendant la phase de conception du système énergétique du bâtiment, il faut prendre en compte son coût de remplacement. Il y a plusieurs moyens de modéliser le vieillissement de la batterie, comme le calcul des Ampères-heures qui la traverse pendant sa vie, ou le calcul de nombre de cycles de charge et de décharge ([Sieglinde & Stephen 1995](#)). Pour simplifier le modèle de la batterie, on a supposé qu'elle doit être remplacée tous les 5 ans (environ 1500 cycles) ([Kiehne 2003](#)). Le vieillissement, le coût initial et le coût du remplacement ont été calculés en conséquence.

Le coût de la batterie est une contrainte locale, mais il participe au calcul de la contrainte globale de coût initial du système, il représente, donc un port économique de ce modèle. Le générateur l'ajoute à la formule de calcul des dépenses annuelles du système automatiquement. Or, avec une approche de génération habituelle (mono solveur présentée dans le paragraphe [7.3.2](#)), cela impose un important travail de reformulation de problème d'optimisation.

valeur dans le champ *subsidy rate*.

Le concepteur de types de modèles peut formuler le coût de la batterie comme :

$$C_{bat} = C_{bini} + C_{remplacement} \quad (7.19)$$

avec :

- C_{bini} : le coût initial de la batterie.
- $C_{remplacement}$: le coût de remplacement de la batterie.

$$C_{bini} = Q_{max} \times p_{bat} \quad (7.20)$$

et :

- Q_{max} : la capacité de la batterie [kWh].
- p_{bat} : le prix du kWh.

Le modèle énergétique de la batterie est illustré dans la formulation qui suit :

$$\begin{aligned} Sup(x_{15}) &= a_{12} \times a_{13} \\ Sup(x_{14}) &= a_{11} \times a_{13} \\ x_{15}(t) &\leq (1 - x_{11}(t)) \times Sup(x_{15}) \\ x_{14}(t) &\leq (1 - x_{11}(t)) \times Sup(x_{14}) \\ x_{15}(t) + x_{13}(t-1) - x_{12} &\leq 0 \\ x_{13}(t-1) - x_{14}(t) - a_{10} \times x_{12} &\geq 0 \\ x_{13}(t) - x_{12} &\leq 0 \\ x_{13}(t) - a_{10} \times x_{12} &\geq 0 \\ x_{15}(t) - x_{14}(t) + x_{13}(t-1) - x_{13}(t) &= 0 \\ x_{12} &\leq a_{13} \end{aligned} \quad (7.21)$$

avec :

- a_9 : le pourcentage définissant définir la valeur initiale de l'état de la charge de la batterie.
- a_{10} : le pourcentage définissant la valeur minimale admissible de la charge de la batterie.
- a_{11} : la vitesse de décharge de la batterie.
- a_{12} : la vitesse de la charge de la batterie.
- a_{13} : la charge (capacité) maximale admissible par la batterie en kWh (Q_{max}).
- x_{11} : la commande de charge/décharge de la batterie : c'est une variable binaire.
- x_{12} : la charge optimale de la batterie en kWh.
- x_{13} : l'état de la charge de la batterie (*State Of Charge*).
- x_{14} : l'énergie de la décharge de la batterie en kWh.
- x_{15} : l'énergie de la charge de la batterie en kWh.
- $Sup(x_{15})$: la borne supérieure de x_{15} .
- $Sup(x_{14})$: la borne supérieure de x_{14} .

Dans ce tableau, nous présentons des explications de chaque contrainte dans le modèle de la batterie.

Ce modèle initialise et impose la périodicité des solutions avec les valeurs de référence du concepteur. L'initialisation oblige le solveur à trouver une solution qui conserve l'état de charge de la batterie à la valeur de charge initiale à la fin du temps d'étude. Cela se fait

$Sup(x_{15}) = a_{12} \times a_{13}$	limite maximale sur l'énergie de charge de la batterie
$Sup(x_{14}) = a_{11} \times a_{13}$	limite maximale sur l'énergie de décharge de la batterie
$x_{15}(t) \leq (1 - x_{11}(t)) \times Sup(x_{15})$	éviter la charge et décharge au même moment (t)
$x_{14}(t) \leq (1 - x_{11}(t)) \times Sup(x_{14})$	éviter la charge et décharge au même moment (t)
$x_{15}(t) + x_{13}(t - 1) - x_{12} \leq 0$	limite pour ne pas dépasser la capacité en chargeant la batterie
$x_{13}(t - 1) - x_{14}(t) - a_{10} \times x_{12} \geq 0$	limite pour ne pas décharger la batterie en dessous de valeur minimale
$x_{13}(t) - x_{12} \leq 0$	l'état de charge doit être toujours inférieur de la capacité
$x_{13}(t) - a_9 \times x_{12} \geq 0$	pour éviter la décharge profonde
$x_{15}(t) - x_{14}(t) + x_{13}(t - 1) - x_{13}(t) = 0$	le bilan énergétique au niveau de la batterie
$x_{12} \leq a_{13}$	limite sur la capacité optimale

TABLE 7.2 – Explication des contraintes du modèle de la batterie

via ces deux contraintes :

$$x_{13}(0) = a_8 \times x_{12} \quad (7.22)$$

$$x_{13}(Studyperiod) = x_{13}(0) \quad (7.23)$$

Le *concepteur de types de modèles* écrit une classe en implémentant l'interface de type *AbstractBatteryModel* dans la figure 7.10. Cette classe (PIM OP-XML Batterie dans la figure 7.8) écrite en utilisant le SDK pour respecter le méta modèle de PIM OP-XML représente le sous-problème de composant batterie. Il écrit les contraintes locales 14.2 directement dans cette classe et spécifie les ports énergétiques (variables x_{14} , x_{15}) et les ports économiques C_{bat} , le générateur de problème complet ajoute ce coût automatiquement dans le bilan économique au OP-XML complet.

Le concepteur graphique pour le stockage développé par le *concepteur de types de modèles* est illustré dans la figure 7.21. Le *concepteur de système bâtiment* peut instancier ce modèle avec des données numériques pour a_8 , a_9 , a_{10} , a_{11} et le prix de kWh p_{bat} .

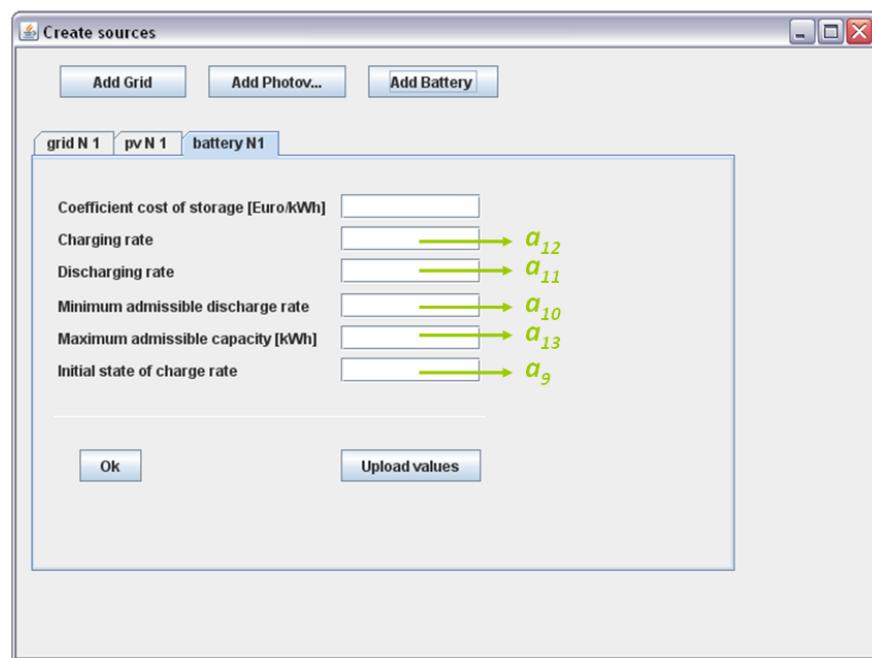


FIGURE 7.21 – Le concepteur graphique développé par le *concepteur de types de modèles* est à utiliser par le *concepteur de bâtiment* pour instancier la batterie

7.5.3 Partie composition de bâtiments (pour le *concepteur de systèmes bâtiment*)

Les types de modèles de sources sont prêts à être instanciés par le *concepteur de systèmes bâtiment* via les concepteurs graphiques, cela passe par la saisie des valeurs des paramètres.

Pour tester plusieurs scénarios et plusieurs topologies de configuration de réseau électrique de système bâtiment, le *concepteur de systèmes bâtiments* utilise directement et uniquement les concepteurs graphiques. Le SDK dans le paragraphe 7.5.1 (qui contient le générateur de problèmes et les projeteurs) s'en charge pour adapter la génération de OP-XML complet selon la composition choisie par cet acteur (voir les figures 7.22, 7.23 et 7.24), puis il génère les modèles de M0 (OP-CADES et OP-CPLEX).

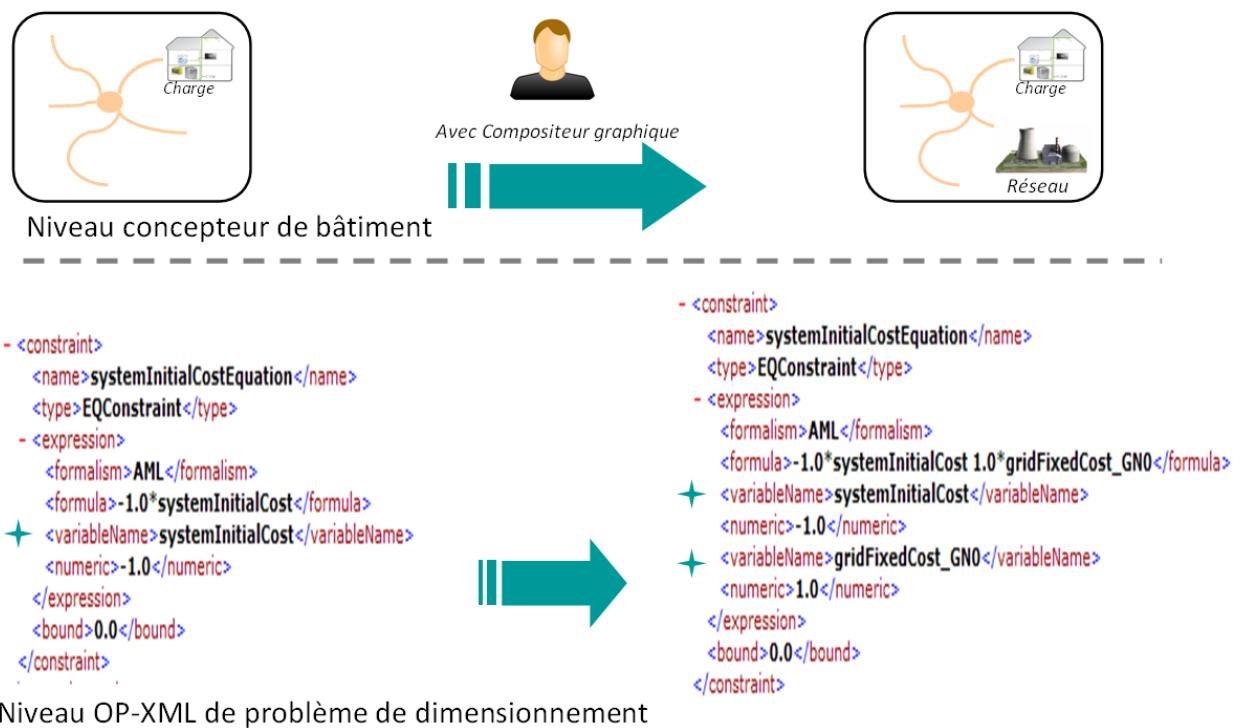


FIGURE 7.22 – Partie de l’OP-XML dont la contrainte *systemInitiaCost* s’évalue automatiquement selon la composition faite par le *concepteur de systèmes bâtiments*

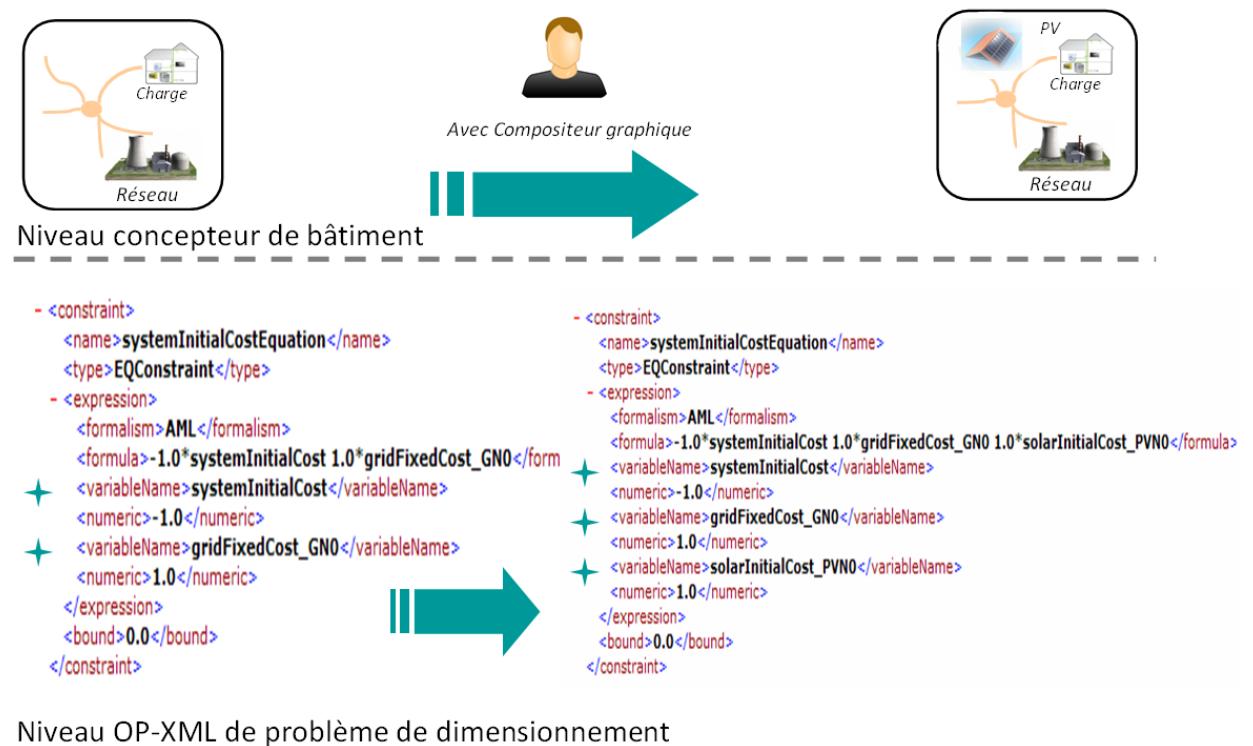


FIGURE 7.23 – Partie de l'OP-XML : l'évaluation automatique de la contrainte `systemInitialCost` après l'ajout de fournisseur et de PV

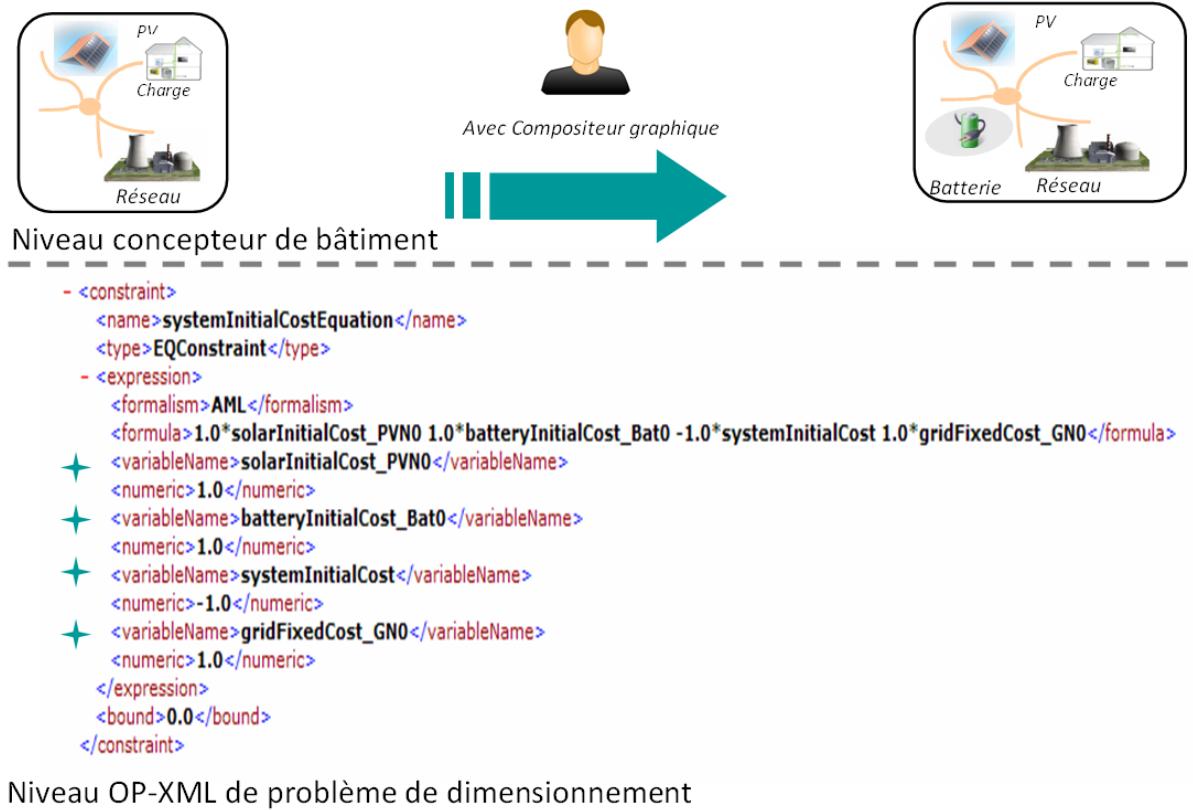


FIGURE 7.24 – Partie de l’OP-XML : l’évaluation automatique de la contrainte *systemInitialCost* après l’ajout de fournisseur et de PV et la batterie

7.6 Étude paramétrique du problème pour le dimensionnement des sources

Le but de cette partie est de montrer comment le *concepteur de systèmes bâtiments* peut utiliser les outils et les processus décrits précédemment pour aller plus loin que la simple conception (de système bâtiment) : il peut en effet réaliser facilement des études de sensibilité vis-à-vis de différents types de données et de voir leur impact sur les variables de dimensionnement mais aussi sur la rentabilité économique de l'investissement. Nous avons choisi des scénarios simples, dont les solutions sont triviales pour que nous puissions détecter les dysfonctionnements possibles.

7.6.1 Impact des données météorologiques

Objectif : tester l'impact de l'historique de données météorologiques utilisées pour la conception.

Démarche : comparer le résultat de conception d'un système bâtiment en faisant varier l'historique des données météorologiques et en gardant les mêmes hypothèses spécifiques aux sources et au marché d'énergie.

Le *concepteur de système bâtiment* peut choisir la formulation du problème sur une journée type (journée moyenne représentative), ou travailler avec des historiques de données de plusieurs jours types (jour d'été, d'hiver etc). Avec la génération automatique, le choix de l'historique des données est facile. Le concepteur de bâtiment choisit la durée d'étude via l'interface présentée dans la figure 8.9 et sélectionne les journées types souhaitées (voir figure 7.25). Le générateur de problèmes prend en charge l'intégration de ces données chaque fois que le concepteur teste un autre scénario (alors que le génération habituel de problème de dimensionnement doit modifier le fichier d'optimisation pour chaque cas).

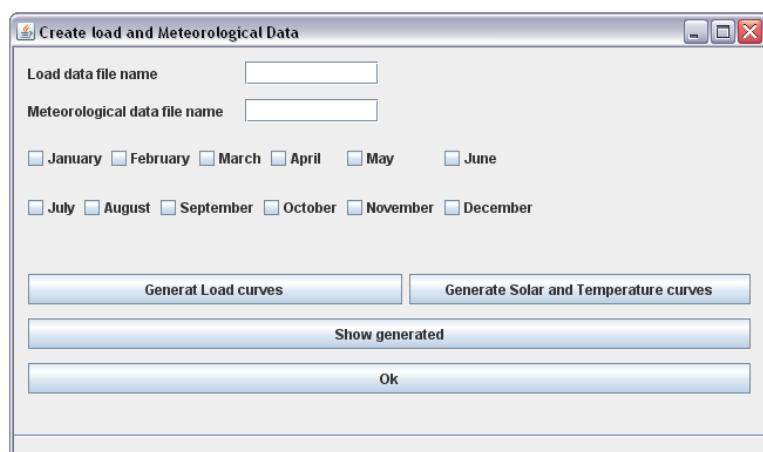
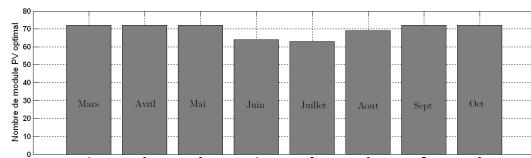


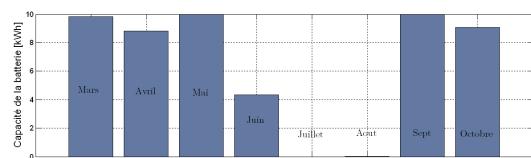
FIGURE 7.25 – Interface graphique de saisie des données météorologiques

Est ce qu'un mauvais choix de formulation peut engendrer une erreur de dimensionnement importante ? Un sous dimensionnement ou sur dimensionnement ?

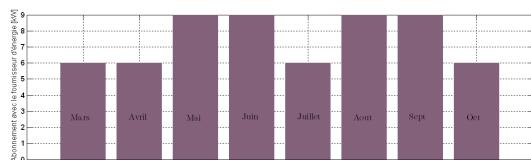
Pour répondre à cette question, nous avons effectué une étude dans laquelle nous avons retenu une formulation pour une journée type (24h), puis nous avons pris les données météorologiques de 12 journées types (288h). Nous avons alors comparé la sensibilité des résultats aux variables de dimensionnement et à l'indice de profitabilité (PI) (cf. figure 7.26). Pour les jours qui représentent les mois de Janvier, Novembre et Décembre, le problème d'optimisation est infaisable (Nous avons fait une analyse : le solveur CPLEX a trouvé des solutions infaisables). Il apparaît que, si un *concepteur de système bâtiment* ne prend en compte que les données d'une journée type, il est vraisemblable que la meilleure solution trouvée soit mal adaptée au problème posé. Par exemple, si uniquement les données d'hiver ont été utilisées, le nombre des panneaux solaires sera supérieur au besoin réel, car le rayonnement solaire n'est pas assez fort ; cela introduira un sur dimensionnement, donc une perte financière est à escompter à cause du coût initial important. Alors que dans le cas où le *concepteur de système bâtiment* ne prend en compte que les données d'été, le nombre de panneaux ne sera pas suffisant pour alimenter la charge prévue pour les mois d'hiver. Cela peut laisser à penser que la gestion des sources ne sera pas optimale.



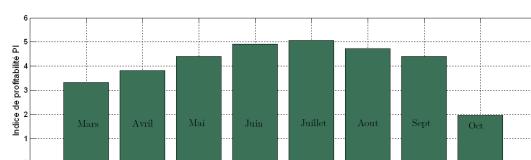
(a) Variation du nombre de panneaux solaires



(b) Variation de la capacité maximale de la batterie



(c) Variation de l'abonnement réseau



(d) Variation de l'indice de profitabilité du système (cf. paragraphe 16.1)

FIGURE 7.26 – Impact du choix des données météorologiques sur le dimensionnement et la rentabilité économique du bâtiment

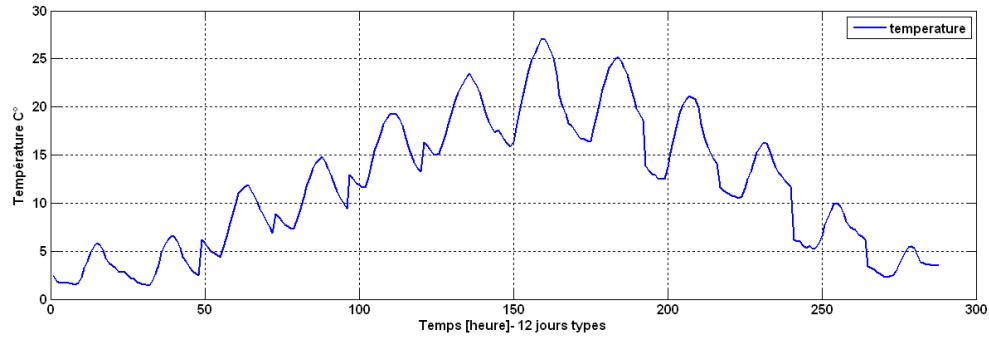
L'augmentation de la taille de l'historique des données accroît la justesse des résultats de dimensionnement mais accroît aussi la taille du problème d'optimisation. Or, il est important de souligner que la formulation du problème est limitée par la capacité de résolution de solveur utilisé. Cela rend la résolution délicate dans certain cas. Par exemple, avec CADES qui est habituellement utilisé pour résoudre des problèmes non linéaires en électromagnétisme, la taille de ces problèmes est de quelques dizaines de contraintes, alors que dans le cas du bâtiment, nous avons environ 200 voire 300 contraintes linéaires, cela pour une formulation sur 24h. En conséquence, nous sommes limités par certaines formulations (24h) avec CADES. Ceci démontre un des intérêts de l'approche de projection multi-solveur car les problèmes de conception ont été projetés et résolus par CPLEX.

Conclusion :

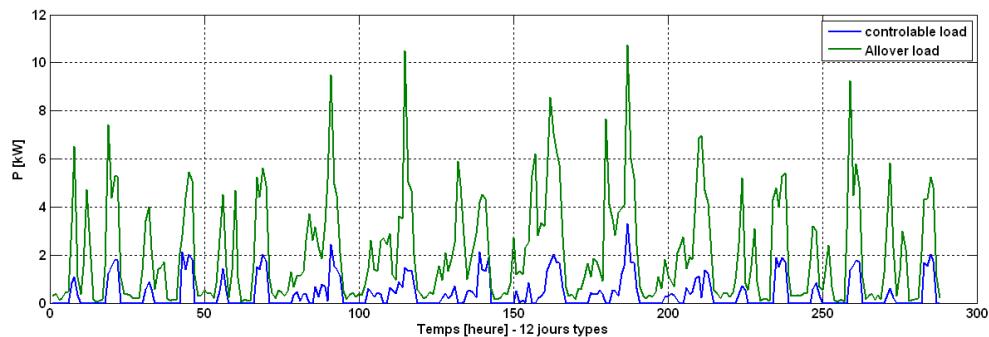
Pour améliorer la formulation sur 24h (un jour), nous allons formuler le problème de conception sur 12 journées types (chaque journée est représentative d'un mois). Cela signifie que la formulation couvre 288h effectives par an (notre outil peut aller jusqu'à 365 jours, mais dans ce cas il faut saisir les données par une autre interface graphique représentée dans la figure 8.10(a)). La figure 7.27 illustre les données météorologiques et les courbes de charges utilisées pour achever cette étude paramétrique.

Dans les études suivantes, nous avons figé les valeurs numériques pour certaines contraintes physiques de composants du bâtiment. Ces contraintes sont :

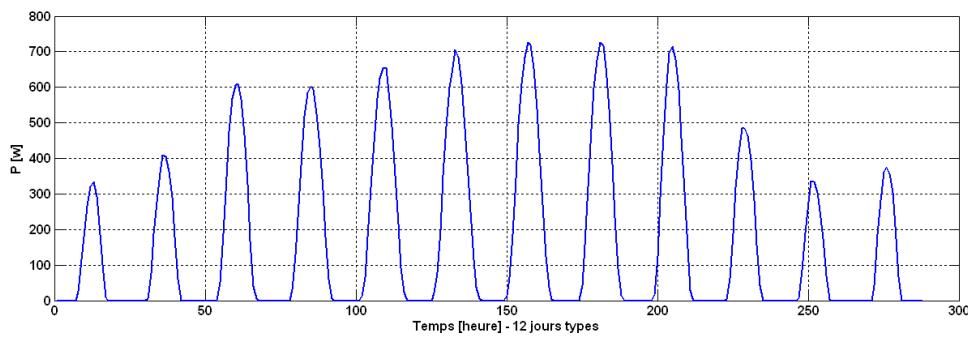
1. Le niveau d'énergie contractuel maximal avec le fournisseur d'énergie est à 9 kW.
2. La capacité maximale de la batterie est de 20 kWh.
3. La production d'un seul panneau solaire est limité par l'ensoleillement de la figure 7.27(c).



(a) Courbe de la température (pour calculer la production de panneaux solaires)



(b) Courbe de charge



(c) Courbe d'ensoleillement utilisée (pour calculer la production de panneaux solaires)

FIGURE 7.27 – Données utilisées dans l'étude paramétrique

7.6.2 Impact des données relevant des hypothèses de type marché HSEM

Nous avons besoin de confirmer la justesse du dimensionnement proposé sur des hypothèses valables sur le long terme pour le dimensionnement. Cela nous oblige à tester plusieurs scénarios du marché de l'énergie afin d'étudier la sensibilité des éléments liés aux *HSEM* sur le retour sur investissement par exemple.

7.6.2-1 Variation du coût des systèmes PV

Objectif : évaluer l'impact de variations du prix des modules et du tarif de rachat du surplus de production PV sur les variables de dimensionnement et sur l'indice de profitabilité dès la phase de la conception.

Démarche : effectuer différents essais avec des prix de rachat du surplus de 0.4 et de 0.5 euro/kWh. Pour chaque prix de rachat du surplus, nous avons aussi fait varier le prix des modules de 4 jusqu'à 5 euros/ W_c . Un abonnement avec le fournisseur d'électricité en double tarif (heures pleines/creuses) est utilisé pour le prix d'achat de l'énergie du réseau. Les valeurs ont été prises sur le site Internet EDF ([EDF 2010](#)) (le prix de l'heure pleine est de 0.1154 euro/kWh et celui de l'heure creuse est de 0.0734 euro/kWh). Le prix de la batterie est fixé à 125 €/kWh.

Le *concepteur de système bâtiment* doit alors modifier les données du réseau et des panneaux solaires avant de générer et de projeter le problème vers un solveur.

Dans la figure 7.28, on voit les deux ensembles de prix de rachat du surplus (ps). Pour un prix de 0.5 euro/kWh, on amortit l'investissement initial plus vite que pour les autres prix. Par contre, l'impact du prix des modules est plus visible sur l'indice de profitabilité que sur la valeur nette actualisée. L'impact sur les variables de dimensionnement est illustré par le tableau 7.3. L'impact sur le retour d'investissement (ou la valeur nette actualisée) est présenté dans la figure 7.28.

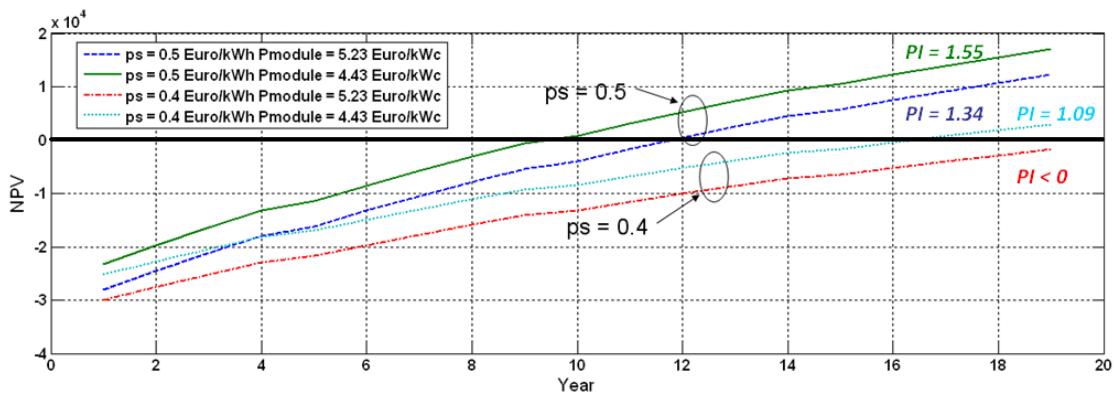


FIGURE 7.28 – Impact du prix du surplus et de celui des modules PV sur la valeur nette actualisée

Remarque : Comme illustré dans le tableau 7.3, le prix des modules PV n'affecte pas le nombre de modules et l'abonnement au réseau. La solution favorise l'occupation de toute la surface par des modules PV dans tous les cas, puis il ajuste l'impact de la variation du prix par la variation de la capacité de la batterie. Si le prix du surplus est important,

Pmodule [€/ W_c]	ps [€/kWh]	N_{module}	P_{abon} [kW]	Q_{max} [kWh]	PI
4.23	0.5	72	6	12	1.55
5.23	0.5	72	6	13	1.34
4.23	0.4	72	6	10	1.09
5.23	0.4	72	6	10	<0.0

TABLE 7.3 – Impact du prix du surplus et de celui des modules PV sur le dimensionnement du système

le solveur favorise une batterie de grosse capacité pour assurer la consommation locale, et vend la majorité de la production solaire au réseau. En conséquence, on voit dans ce cas que l'indice de profitabilité (PI) est plus important que le cas de $ps = 0.4$. Bien évidemment, le PI décroît un peu avec l'augmentation du prix des modules, car on ajoute des dépenses aux capitaux initiaux.

7.6.2-2 Variation du coût du système de stockage

Objectif : Savoir comment le solveur va réagir pour différents prix de stockage.

Démarche : Nous avons fait varier le coût du système de stockage avec le prix du surplus. Le prix de l'énergie du fournisseur est toujours sous l'hypothèse précédente heure pleine/heure creuse. Le prix de modules PV est fixé à 4.23 €/ W_c .

Dans les essais précédents, la capacité du système de stockage était limitée par une capacité initiale donnée pour un certain prix du kWh. Faire augmenter le prix du stockage permet de limiter via l'économie, la capacité de stockage mise à disposition du besoin local. Le prix du kWh est appelé BatteryPrice dans la figure 7.29.

Le *concepteur de systèmes bâtiments* intègre les nouvelles données pour la batterie et pour les panneaux solaires via les concepteurs graphiques. Le SDK génère et projette alors le problème vers CPLEX. La figure 7.29 illustre l'impact économique (représenté par la valeur nette actualisée). Le tableau 7.4 présente l'impact sur les variables de dimensionnement.

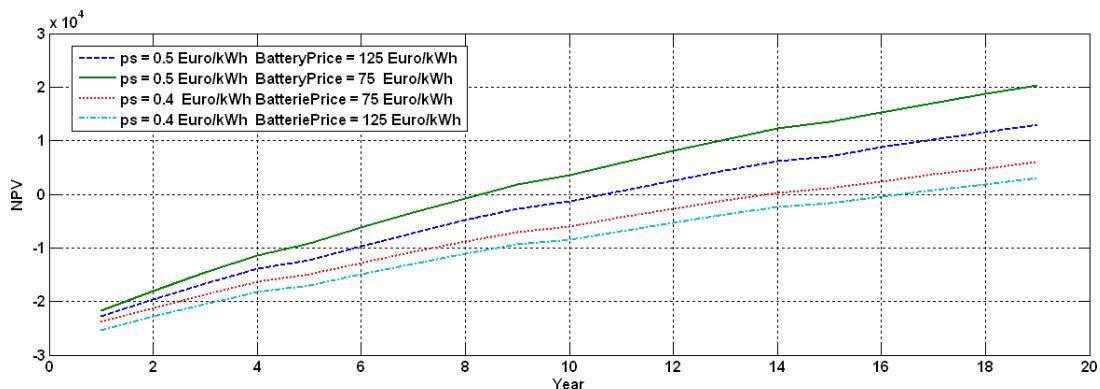


FIGURE 7.29 – Impact du prix de la batterie et du prix de rachat du surplus sur la valeur nette actualisée

On voit sur la figure 7.29 que l'impact sur l'année d'amortissement pour le même prix

Prix de la batterie [€/kWh]	ps [€/kWh]	N_{module}	P_{abon} [kW]	Q_{max} [kWh]	PI
75	0.5	72	6	18	1.68
125	0.5	72	6	11	1.44
75	0.4	72	6	15	1.2
125	0.4	72	6	10	1.09

TABLE 7.4 – Impact du prix du surplus et de celui de la batterie sur le dimensionnement du système

de surplus et différents prix de batterie n'est pas significatif. L'effet sur le temps nécessaire pour amortir les capitaux initiaux ne sera donc pas très important.

Remarque : Le solveur favorise l'installation de modules PV sur toute la surface de toit disponible, puis ajuste l'impact de la variation du prix sur la capacité de la batterie (qui a un faible impact économique, comme illustré dans la figure 7.29). Pour un même prix de surplus, on voit que l'augmentation du coût de la batterie diminue la capacité optimale obtenue (ce qui est trivial) ; le nombre de modules PV et l'abonnement avec le fournisseur ne sont pas varié dans cet exemple.

7.6.3 Impact de l'hypothèse sur le type d'usagers

Nous arrivons maintenant au troisième type d'éléments influençant la conception : les hypothèses spécifiques aux utilisateurs (HSU). Ce type d'hypothèse a l'impact le plus important sur la conception. Différents usagers subissent les mêmes conditions de marché et les mêmes données météorologiques, mais chacun a ses conditions. Nous montrons l'impact de différentes hypothèses comme la surface disponible pour installer les panneaux et le type d'abonnement sur le dimensionnement.

7.6.3-1 Exemple d'hypothèses spécifiques aux usagers avec double tarif

Objectif : Voir l'impact de la surface disponible pour le PV sur le dimensionnement pour un abonnement avec le fournisseur à double tarif.

Démarche : Le prix de rachat du surplus vaut 0.4 euro/kWh. Le tarif est pour l'heure pleine, 0.1154 euro/kWh et pour l'heure creuse, 0.0734 euro/kWh. Nous avons fait varier la surface disponible pour installer les panneaux de $40m^2$ à $70m^2$. Les prix des modules et de la batterie sont fixés à 4.23 euro/ W_c et 125 euro/kWh respectivement.

Remarque : L'ajout de panneaux solaires n'a pas un impact sur la taille de la batterie nécessaire (environ 14 kWh dans tous les cas). L'abonnement avec le fournisseur est resté inchangé (tableau 7.5). Si on augmente le nombre des panneaux, on augmente l'investissement initial mais on gagne plus pendant l'exploitation de système. Cela permet d'amortir l'investissement plus rapidement (figure 7.30).

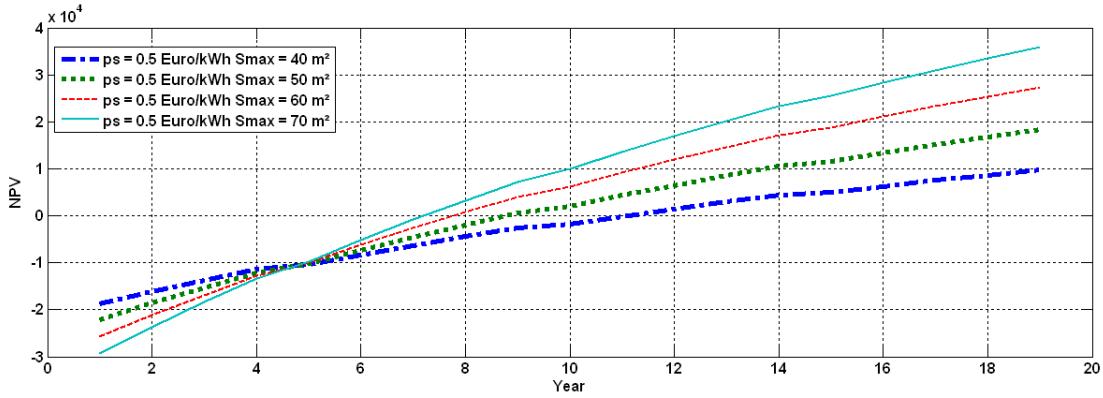


FIGURE 7.30 – Impact de la surface disponible pour installer les modules sur la valeur nette actualisée

surface [m ²]	ps [€/kWh]	N _{module}	P _{abon} [kW]	Q _{max} [kWh]	PI
40	0.5	58	6	13	1.4
50	0.5	72	6	13	1.6
60	0.5	87	6	13	1.76
70	0.5	101	6	14	1.87

TABLE 7.5 – Impact de la surface disponible pour installer le PV sur le dimensionnement du système

7.6.3-2 Exemple d'hypothèses d'usagers spécifiques avec simple tarif

Objectif : Voir l'impact de la surface disponible pour installer le PV sur le dimensionnement pour un abonnement avec le fournisseur à simple tarif.

Démarche : Nous avons repris le scénario précédent mais avec un abonnement réseau en simple tarif. Le prix de l'énergie achetée au réseau est fixe pour toute la journée (0.1081 euro/kwh).

surface [m ²]	ps [€/kWh]	N _{module}		P _{abon} [kW]		Q _{max} [kWh]		PI	
		DT ¹⁴	ST ¹⁵	DT	ST	DT	ST	DT	ST
50	0.5	72	72	6	6	13	11	1.6	1.44
60	0.5	87	87	6	6	13	12	1.76	1.62
70	0.5	101	101	6	6	14	12	1.87	1.75

TABLE 7.6 – Impact de la surface disponible pour installer le PV sur le dimensionnement du système, pour un abonnement avec double tarif (DT) et un abonnement avec simple tarif (ST)

Remarque : En comparant les résultats (voir tableau 7.6) avec ceux du double tarif, on voit que le double tarif est plus avantageux. La capacité optimale de la batterie est plus grande que celle du simple tarif. Le prix de l'énergie du fournisseur étant variable sur la journée, une grosse batterie est avantageuse pour profiter de la période où l'énergie

du réseau est moins chère. On charge alors la batterie au maximum, puis on l'utilise pour la consommation locale durant la période où le prix d'achat est élevé. Par conséquent, la profitabilité est plus importante que le simple tarif.

7.7 Conclusion

Le fait qu'il y ait plusieurs prototypes de bâtiments et différents acteurs, conduit à une grande variété de problèmes de dimensionnement requérant différents environnements d'optimisation, ce qui n'est pas toujours facile à anticiper. En conséquence, la génération de problèmes de dimensionnement pour un solveur précis présente un risque qui peut coûter cher lorsqu'il faut migrer vers un nouveau type de solveur. Le besoin de moyens automatiques pour générer les problèmes d'optimisation pour différents environnements d'optimisation est plus que nécessaire pour une application de type bâtiment intelligent multi-source.

Nous avons montré dans ce chapitre une approche concrétisée comme un outil métier pour générer automatiquement les problèmes de dimensionnement de bâtiments et les projeter vers différents solveurs. Nous avons défini trois types de métiers concernés par les problèmes de dimensionnement : le métier de développeur logiciel, le métier de modélisateur de type de modèles et le métier de conception de systèmes bâtiments. Chaque métier est représenté par un acteur jouant un rôle. Cette définition des rôles permet d'être plus efficace face aux difficultés de dimensionnement car chacun de ces acteurs peut se spécialiser et intervenir dans des temporalités différentes.

Un problème de dimensionnement de sources dans lequel les aspects énergétiques et économiques sont couplés est présenté. Nous essayons avec la formulation de problème de trouver le dimensionnement optimal des sources électriques tout en optimisant la profitabilité économique du système global.

Nous avons présenté une analyse de sensibilité paramétrique sur le dimensionnement pour montrer l'intérêt de l'approche pour le *concepteur de systèmes bâtiments*. Nous avons réussi grâce à l'outil métier développé, à vérifier certaines solutions évidentes, à quantifier précisément cela dans le cadre des hypothèses et à dimensionner avec une stratégie de pilotage optimale .

Chapitre 8

Problèmes de gestion optimale des sources électriques dans le bâtiment

Faites quelque chose et, si ça ne réussit pas, essayez autre chose

F.Roosevelt

SOMMAIRE

8.1	DU DIMENSIONNEMENT À LA GESTION	121
8.2	MÉTHODES, ALGORITHMES ET OUTILS POUR LA GESTION GLOBALE DES SYSTÈMES BÂTIMENT	121
8.3	RÉSOLUTION DES PROBLÈMES DE GESTION OPTIMALE DE SYSTÈMES BÂTIMENT	123
8.3.1	Différents types de charges électriques	124
8.3.2	Différentes configurations de sources électriques	124
8.3.3	Différents environnements de résolution	125
8.4	MISE EN OEUVRE D'UN OUTIL MÉTIER POUR LA GESTION DES SOURCES	126
8.4.1	Partie Modélisation pour le <i>concepteur de types</i>	126
8.5	OUTIL POUR LE DIMENSIONNEMENT ET LA GESTION DES SYSTÈMES BÂTIMENT	131
8.6	CONCLUSION	133

Résumé

Dans ce chapitre, nous continuons l'implémentation de la génération automatique de problèmes d'optimisation et de la projection vers différents solveurs pour les problèmes de gestion des sources d'énergie électrique dans les bâtiments. Comme pour le dimensionnement, nous commençons par introduire les différentes problématiques liées à la formulation des problèmes de gestion comme, par exemple, les différents types de charges électriques, les différentes configurations possibles des sources raccordées au bâtiment mais aussi les différentes méthodes avec lesquelles nous étudions le problème de gestion du bâtiment. Parmi ces méthodes, nous nous sommes intéressés aux approches qui procèdent par optimisation. Pour faciliter l'étude de la problématique de gestion, à l'instar de la problématique de dimensionnement, nous proposons de spécialiser les rôles des acteurs en trois rôles principaux : le développeur logiciel, le concepteur de type de modèles et le concepteur de gestionnaire de systèmes bâtiments. Nous examinerons comment ces acteurs peuvent travailler ensemble. De là, nous présentons deux outils de gestion de flux énergétiques pour les systèmes bâtiment. Le premier a été complètement développé dans le cadre de cette thèse afin de valider notre démarche. Le second est le fruit du projet ANR Multisol auquel nous avons contribué et qui applique les principes de génération automatique de problèmes de gestion et la projection vers différents solveurs.

8.1 Du dimensionnement à la gestion

La gestion d'énergie est importante pour améliorer l'efficacité énergétique des systèmes bâtiment mais aussi pour améliorer la maîtrise de la demande d'électricité (MDE) par les gestionnaires de réseaux. Selon Brambley et al. (2005), la réduction globale annuelle de la consommation suite à la mise en place de stratégies de pilotage efficaces est estimée à 15% d'économie.

Contrairement au dimensionnement qui est résolu en un moment précis, les problèmes de gestion sont résolus quotidiennement car les stratégies de gestion sont directement liées aux activités des occupants d'un système bâtiment. Le système de gestion énergétique qui pilote globalement les sources et les charges doit satisfaire aux contraintes de fonctionnement qui correspondent autant aux besoins des usagers (par exemple, les besoins de confort thermique, le besoin de conservation des aliments, etc...) qu'aux contraintes liées au bon fonctionnement des équipements. Des facteurs exogènes comme la température et l'ensoleillement doivent aussi être considérés dans les problématique de gestion énergétique car ils affectent directement certains services comme par exemple le chauffage qui représente, aujourd'hui, autour de 60% de la consommation électrique dans l'habitat (Quenard & Marcoux 2009)).

Un système de pilotage global du bâtiment doit s'adapter aux changements de configuration, comme par exemple l'ajout ou le retrait de charges ou de sources. Cela implique la capacité des systèmes de gestion à générer dynamiquement les bons problèmes de gestion de flux énergétiques quelles que soient les modifications de la configuration du système bâtiment. La résolution des problèmes générés ne doit pas être limitée à un seul environnement d'optimisation car, comme nous l'avons vu avec la problématique de dimensionnement, il y a des cas où certains solveurs ne sont pas capables de résoudre les problèmes ou, plus pragmatiquement sont assujettis à des coûts de licence alors que d'autres sont open-source. Les outils de projection vers différents solveurs présentent là encore un intérêt.

8.2 Méthodes, algorithmes et outils pour la gestion globale des systèmes bâtiment

Nous pouvons distinguer deux catégories d'études pour la gestion des systèmes bâtiments.

Etudes liées directement à la conception de gestionnaire énergétique

Dans ce cas, on essaye d'anticiper le comportement de tous les équipements :

- Par une étude procédant par modélisation et simulation, il est possible d'établir plusieurs scénarios représentatifs de la réalité.
- Par optimisation de la gestion des équipements du système bâtiment préalablement dimensionné, il est possible de calculer des plans de fonctionnement anticipé optimaux.

Fabrizio (2008) illustre les techniques et les méthodes utilisées pour simuler les composants de bâtiment (séries temporelles, modèles stochastiques). Dans (Paris et al. 2010), une approche d'optimisation linéaire utilisant l'algorithme du Simplexe est implantée pour un bâtiment avec plusieurs sources électriques. Dans (Warkozek et al. 2009), une

approche d'optimisation avec Simplexe et SQP est présentée. Des outils issus de l'intelligence artificielle sont aussi mis en oeuvre pour optimiser l'intermittence de la production des ressources. Dans ([Penya 2003](#)) et ([Sopian et al. 2007](#)), des algorithmes génétiques sont utilisés. Des réseaux de neurones sont aussi proposés dans ([Warkozek \(2007\)](#)) et un système multi-agents a été proposé dans ([Boman et al. 1998](#)).

Des travaux couplant les approches d'optimisation et une validation en simulation sont apparus récemment ; ([Missaoui et al. 2010](#)) et ([Paris et al. 2010](#)) valident la commande optimale des sources avec une simulation temps réel des équipements de systèmes bâtiment (sources et charges).

La gestion par optimisation peut être conçue durant la phase d'exploitation d'un système bâtiment sur la base des vrais équipements ou avant, avec des simulations des équipements.

Ce pilotage global des sources et des charges électriques peut être effectué par anticipation en s'appuyant sur différentes prédictions, et de manière réactive pour faire face aux aléas en tenant compte des différentes contraintes de confort et de fonctionnement des équipements d'un système bâtiment. G-HomeTech ([Ghometech 2009](#)) propose une solution logicielle pour réaliser cette gestion globale. ([Missaoui et al. 2010](#)) propose de réaliser un couplage entre la gestion par optimisation et la simulation du système bâtiment (voir l'annexe [15.1.6](#)).

La gestion anticipative de G-HomeTech, par exemple, procède par optimisation. Les plans anticipatifs (J-1) sont calculés en trois étapes :

1. établissement d'un plan prévisionnel du fonctionnement des charges conformément aux demandes des occupants,
2. récupération des prévisions météorologiques et des prévisions des prix de l'énergie lorsqu'ils sont variables,
3. formulation du problème d'optimisation correspondant puis calcul du plan optimal de fonctionnement pour les charges et les sources.

Le plan obtenu au jour (J-1) n'est pas forcément celui qui sera réalisé au jour (J), car il peut y avoir des variations météorologiques ou des variations d'usage. Ainsi, l'anticipation doit être accompagnée par un système réactif pour adapter les commandes aux changements imprévus. On voit dans ce cas que le problème de gestion peut être reformulé plusieurs fois. Cela augmente la nécessité d'un système de génération automatique de problèmes de gestion.

Etudes d'évaluation et de diagnostic de systèmes bâtiment

Ces études visent à mieux comprendre l'influence des facteurs intérieurs et extérieurs au système bâtiment comme dans ([Chenailler et al. 2010](#)) et dans ([Chesné et al. 2010](#)). Cela permet d'améliorer la conception du système bâtiment équipé, éventuellement, de gestionnaires énergétiques.

Outils d'analyse et de simulation de systèmes bâtiment

Dans ([Fabrizio 2008](#)), nous trouvons une liste d'outils commerciaux pour la modélisation et le dimensionnement de systèmes multi-sources. Les outils sont importants autant avant qu'après la construction d'un système bâtiment. Dans ([Guavarch et al. 2010](#)), on trouve

un progiciel d'aide à la conception pour la phase avant la réalisation et pour la phase d'esquisse du système bâtiment.

Comme évoqué dans le chapitre précédent, nous visons dans cette thèse à proposer des outils pour supporter une approche collaborative entre des acteurs spécialisés liés à la conception de systèmes bâtiment, équipés ou pas de gestionnaires énergétiques pilotant les sources électriques. Dans ce chapitre, nous prolongeons cette démarche en exploitant la programmation linéaire mixte pour gérer les sources et les charges du système bâtiment. Dans l'annexe 15.1, nous présentons l'outil G-HomeTech, dont nous sommes l'un des inventeurs recensés dans la déclaration d'invention ([Ghometech 2009](#)), qui vise à gérer globalement les sources et les charges des systèmes bâtiment. Il est le résultat du projet ANR MULTISOL. Dans G-HomeTech, l'approche de génération automatique des problèmes d'optimisation et la transformation des modèles ont été appliquées.

8.3 Résolution des problèmes de gestion optimale de systèmes bâtiment

La gestion d'énergie dans les systèmes bâtiment peut être conçue globalement ou être décomposée en deux parties : la gestion des sources et celle de charges. Dans ([Levermore 2003](#)) et ([Brambley et al. 2005](#)), les systèmes de gestion d'énergie, nommés *Gestion Technique de Bâtiments*, ou (*Building Energy Management Systems*), se concentrent sur les techniques de communication et sur la gestion des charges comme l'éclairage, la climatisation et le chauffage. L'existence de plusieurs sources et l'arrivée probable du véhicule électrique connecté au système bâtiment accroît la complexité de la gestion des sources de production locale.

Nous rappelons les rôles de trois acteurs majeurs pour la résolution des problèmes de gestion :

Le *concepteur de gestionnaire de systèmes bâtiment* a la charge d'adapter les outils de gestion énergétique à un système bâtiment particulier. Son rôle est de paramétriser des types de modèles développés par le *concepteur de types de modèles*, de les intégrer dans un gestionnaire énergétique dédié à un cas particulier et de valider le gestionnaire ainsi obtenu. Dans le cas d'une gestion optimisée, le gestionnaire intègre un générateur de problèmes, des projecteurs vers des environnements d'optimisation et au moins un solveur. Cet acteur dispose donc d'un éditeur qui lui permet de paramétriser les modèles via les interfaces graphiques conçues par le *concepteur de types de modèles* et de connecter les différents éléments du système bâtiment via les ports des modèles.

Le *concepteur de types de modèles* s'appuie sur une plateforme de développement conçue par le *développeur logiciel* pour construire des types de modèles qui seront paramétrés par les *concepteur de gestionnaire de systèmes bâtiment* au moyen des concepteurs graphiques ou IHM conçues aussi par le *concepteur de types de modèles*.

Le *développeur logiciel* construit le *framework* logiciel qui permet d'insérer des types de modèles, de gérer des problèmes d'optimisation et de collecter les résultats. Cet acteur développe les projecteurs vers les solveurs jugés intéressants. Il a aussi la charge de construire un générateur d'éditeurs de modèles pour faciliter la tâche au *concepteur de types de modèles*.

Chronologiquement, le *développeur logiciel* intervient le premier, suivi du *concepteur de types de modèles*. Le *concepteur de gestionnaire de systèmes bâtiment* intervient ensuite. Les acteurs interviennent donc dans des temporalités différentes.

Intéressons-nous en particulier au *concepteur de types de modèles* qui a la charge de modéliser les différents types d'équipements que l'on rencontre dans un système bâtiment.

8.3.1 Différents types de charges électriques

Dans ([Fabrizio 2008](#)), les charges sont classées selon les services qu'elles fournissent : charges pour le chauffage, charges pour le refroidissement et les autres charges électriques. Nous trouvons une taxonomie des types de charges selon les possibilités de contrôle dans ([Long 2007](#)). La classification n'est pas exclusive : une charge peut appartenir à plusieurs catégories. Nous avons retenu quatre catégories :

1. décalable (comme une machine à laver),
2. interruptible (comme un ventilateur et certains appareils électroménagers),
3. modulable (comme le chauffage),
4. accumulable (comme les appareils de stockage).

Il existe de nombreuses charges électriques mais toutes peuvent être classées dans les catégories précédentes. Le *développeur logiciel* doit donc s'appuyer sur ces catégories pour concevoir les interfaces qui permettront au *concepteur de types de modèles* de développer les types rencontrés.

Parfois les modèles sont mal connus, par exemple parce que les fabricants d'électroménagers ne les communiquent pas. Dans ce cas, l'approche de génération automatique de problèmes proposée doit être remplacée par une autre comme par exemple l'approche à base de système multi-agent proposée dans ([Abras 2008](#)). Le modèle est encapsulé dans un agent, sans être communiqué au gestionnaire énergétique : l'agent intervient alors dans la résolution du problème via des processus de négociation (voir la figure 8.1).

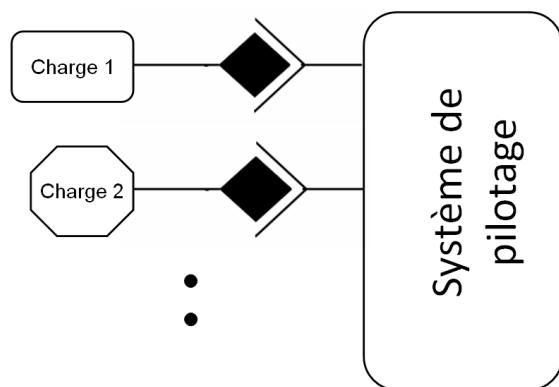


FIGURE 8.1 – Charges vues comme des agents qui négocient leurs besoins énergétiques avec le système de pilotage

La configuration d'un système bâtiment évolue en permanence car l'usager peut utiliser de nombreuses charges durant 24h. En conséquence, la formulation du problème doit être

générée automatiquement et le plus dynamiquement possible. Cela peut être résolu par le générateur de problèmes proposé dans cette thèse ([Missouai et al. 2010](#)).

8.3.2 Différentes configurations de sources électriques

La gestion des sources électriques est un processus dans lequel les règles de contrôle/commande des sources sont établies. La figure 8.2 montre un exemple de planification des sources sur une journée d'un système composé du fournisseur d'énergie électrique, de panneaux solaires et d'une batterie. Le *concepteur de types de modèles* peut ajouter les types de modèles correspondant à ces sources dans la bibliothèque de types et profiter du générateur automatique de OP-XML¹.

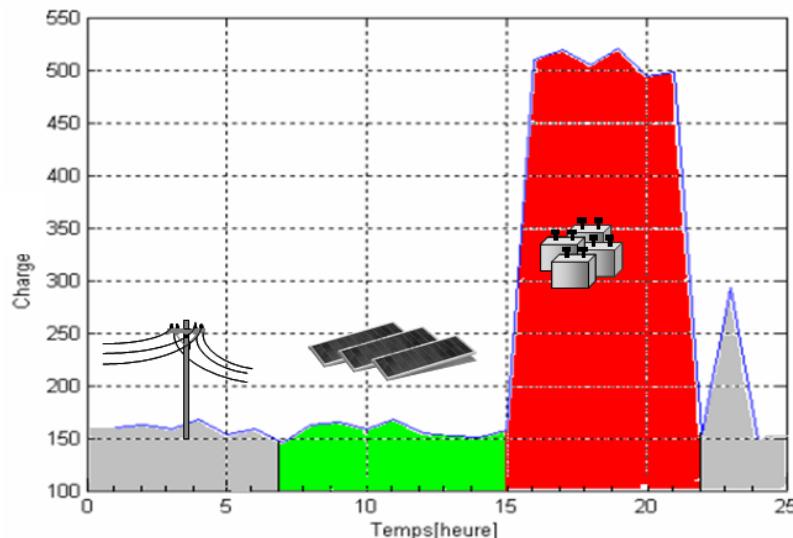


FIGURE 8.2 – Exemple de répartition optimale des sources électriques sur une journée

Le *concepteur de gestionnaires de bâtiment* peut formuler le problème de gestion des sources en profitant des concepteurs graphiques préparés par le *concepteur de types* (comme illustré dans le chapitre précédent). L'adaptation aux différentes configurations sera résolue grâce au générateur automatique de problèmes d'optimisation (cf. figure 8.8).

8.3.3 Différents environnements de résolution

Comme dans la problématique de dimensionnement, il est intéressant de bénéficier de plusieurs environnements de résolution. Les solveurs basés sur des algorithmes de programmation linéaire avec nombres entiers (PLNE) sont plus efficaces que les algorithmes de programmation séquentielle quadratique (SQP) ([Ghomtech 2009](#)) pour les problèmes de grande taille (beaucoup de variables et de contraintes) lorsque les problèmes peuvent être rendus linéaires. Il existe d'ailleurs plusieurs solveurs qui peuvent être utilisés par G-homeTech : GLPK, un solveur libre et CPLEX, un solveur nécessitant une licence et

1. Il peut aussi exister d'autres sources comme un groupe électrogène ou des éoliennes. Il serait souhaitable qu'à l'instar des charges, il existe des catégories de sources, mais il n'y a pas beaucoup de différences entre les éoliennes et le PV : la production d'énergie est dans les deux cas dite fatale (elle est subi sans moyen de la contrôle)

généralement plus puissant que GLPK. Les algorithmes de programmation séquentielle quadratique peuvent eux gérer les problèmes non linéaires. Le temps de résolution avec un solveur basé sur SQP est généralement plus important que celui nécessaire à des algorithmes de type SIMPLEX et des procédures de séparation évaluation (résolution instantanée), ou Branch&Bound en anglais. Cela illustre le fait qu'un unique solveur ne peut convenir à toutes les situations. Le *concepteur de gestionnaires de systèmes bâtiment* peut comparer les résultats obtenus avec différents solveurs grâce aux projecteurs.

8.4 Mise en oeuvre d'un outil métier pour la gestion des sources

La méthode de développement illustrée et utilisée pour générer les problèmes de dimensionnement a été adaptée aux problèmes de gestion de sources. Les mêmes principes illustrées dans le chapitre précédent pour concevoir une plateforme de développement conçue par le *développeur logiciel* seront mis en oeuvre. Le *concepteur de types* construit les types de modèles paramétrés des équipements du système bâtiment sous forme de PIM OP-XML. Il conçoit alors les concepteurs graphiques nécessaires pour que le *concepteur de gestionnaires énergétiques* puisse instancier les paramètres des types et récupère les problèmes de niveau M0 (GLPK, CPLEX ou CADES).

8.4.1 Partie Modélisation pour le *concepteur de types*

L'ajout de types de modèles de charges électriques correspond à la procédure présentée dans le chapitre précédent. Nous allons prendre comme exemple le cas de l'ajout du type fournisseur d'énergie, du type panneau photovoltaïque, quant au type batterie, il est détaillé dans l'annexe 14.1.

8.4.1-1 Ajoût du type fournisseur d'énergie

Le *concepteur de type* écrit le modèle analytique paramétré du fournisseur d'énergie pour la gestion comme illustré avec la formulation 8.1 :

$$\forall t \in 0..Studyperiod$$

$$\begin{aligned} CR_1 : & 0 \leq x_1(t) \leq a_1 \\ CR_2 : & x_2(t) = a_2(t) \times x_1(t) \\ CR_3 : & x_1(t) = (1 - x_6(t)) \times a_1 \end{aligned} \quad (8.1)$$

avec :

- a_1 : la puissance d'abonnement maximum souscrite définie par le gestionnaire. Cette valeur était une variable d'optimisation durant la phase de dimensionnement du système bâtiment,
- $a_2(t)$: le prix d'un kWh d'énergie en provenance du réseau [€/kWh], qui peut varier à chaque instant t ,
- x_1 : l'énergie par tranche horaire achetée au réseau en kWh qui varie en fonction de l'heure,
- x_2 : le coût de l'énergie du réseau [€],

- x_6 : une variable de commande pour gérer la revente d'énergie au réseau. Comme nous allons le montrer dans la suite, nous n'autorisons pas l'achat et la revente d'énergie en même temps. Cette variable est de type binaire (0 ou 1), valant 1 lorsqu'on revend de l'énergie au fournisseur d'électricité, et 0 lorsqu'on achète de l'énergie..

Le contrôle sur l'achat et, éventuellement, sur la revente est contraint par l'astuce suivante (sachant que $x_1 \geq 0$) :

- si $x_6=0 \implies x_1 = a_1$ (cas d'achat).
 si $x_6=1 \implies x_1 = 0$ (cas de revente).

Le *concepteur de types de modèles* encapsule alors le type de modèle en s'appuyant sur le méta modèle de PIM OP-XML. Il peut utiliser le paquetage Java PIMgenerator² (figure 8.3), que nous avons développé. La variable d'énergie achetée au fournisseur d'électricité (x_1) est un port énergétique de ce type de modèle, qui sera ajouté automatiquement dans le bilan énergétique construit par le *concepteur de gestionnaires énergétiques*. Le *concepteur de types* déclare cette variable à l'instar des codes Java illustrés dans la figure 8.4.

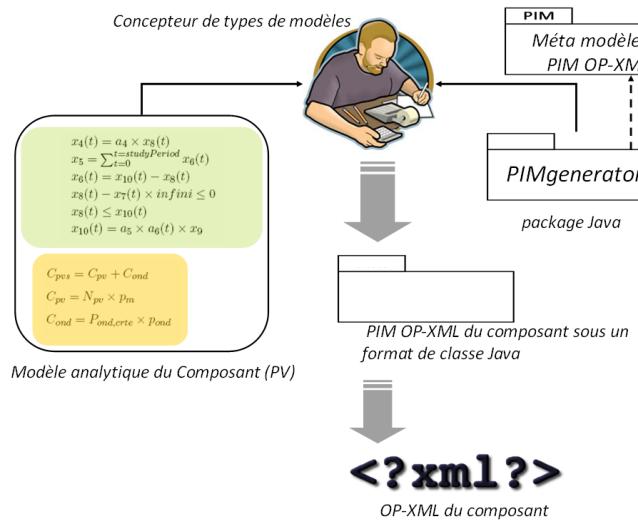


FIGURE 8.3 – Principe de génération du PIM OP-XML pour un équipement

```

Nom Représentatif
de X1 ↓
gridEnergyVariable[i]=this.pim_OPXML.addContinuousVariable("gridEnergyVariable"+_GN"+gridID+"_"+c[i]+i,0.0,
  this.infini,0.0,
  new String[]{boughtEnergyEquation[i].getName(),} Les endroits d'implémentation (CR2 , CR1)
  gridPowerUpperLimit[i].getName()),)
  new String[]{"+"+getEnergyPrices()[i],"+1"}); } ses coefficients dans (CR2 , CR1)
this.pim_OPXML.getEnergeticNode().put(gridEnergyVariable[i],+1.0) Ajouter au nœud énergétique
  
```

FIGURE 8.4 – Partie de classe Java qui représente le PIM OP-XML du fournisseur d'énergie (pour x_1) à écrire par le *concepteur de types de modèles*

L'interface graphique dédiée au *concepteur de gestionnaires énergétiques de système bâtiment* permet d'ajouter le PIM OP-XML associé au type *fournisseur d'énergie*. Cette

2. figure 5.10, paragraphe 5.2.5

interface apparaît dans la figure 8.5. Le *concepteur de gestionnaires énergétiques* instancie les paramètres spécifiques au cas d'étude, comme par exemple, les valeurs maximales d'énergie contractuelles, le prix de l'énergie du fournisseur, etc...

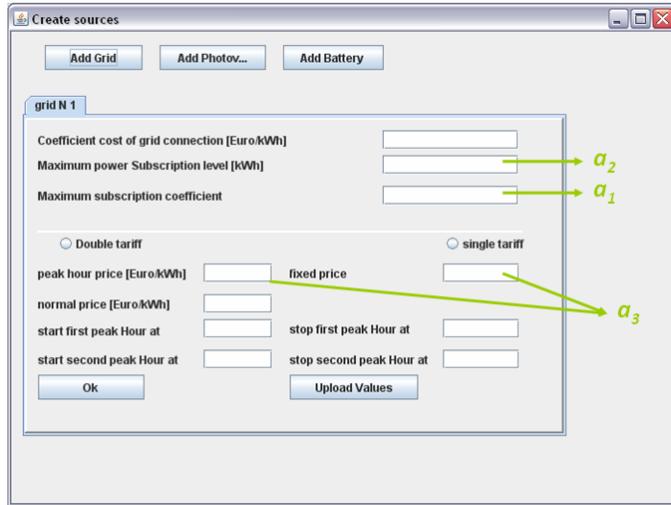


FIGURE 8.5 – Interface graphique permettant d'ajouter le PIM OP-XML du type fournisseur d'énergie

Une partie d'OP-XML du type fournisseur générée est illustrée dans la figure 8.6. Dans cette partie, nous avons x_1 pour un instant ($t = 0$).

La figure 8.7 présente une partie du problème de gestion, telle qu'elle est projetée automatiquement pour CPLEX à partir du l'OP-XML de la figure 8.6.

```

<!-- <constraintName>TotalLoadEquation_23</constraintName>
    <coefficient>1.0</coefficient>
  </implementationInConstraint>
</variable>
- <variable>
  <name>gridEnergyVariable_GNO_00</name>
  <min>0.0</min>
  <max>1000000.0</max>
  <init>0.0</init>
  <type>CONTINUOUS</type>
- <implementationInConstraint>
  <constraintName>boughtEnergyEquation_GNO_00</constraintName>
  <coefficient>0.12</coefficient>
</implementationInConstraint>
- <implementationInConstraint>
  <constraintName>EnergyBalanceEquation_00</constraintName>
  <coefficient>1.0</coefficient>
</implementationInConstraint>
- <implementationInConstraint>
  <constraintName>gridControlConstraint_GNO_00</constraintName>
  <coefficient>1.0</coefficient>
</implementationInConstraint>
- <implementationInConstraint>
  <constraintName>gridPowerUBconstraint_GNO_00</constraintName>
  <coefficient>1.0</coefficient>
</implementationInConstraint>
</variable>
- <variable>
  <name>gridEnergyVariable_GNO_01</name>
  <min>0.0</min>
  <max>1000000.0</max>

```

The diagram illustrates the mapping of specific XML constraints to constraints CR2, CR3, and CR1. The constraints are grouped by brace on the right side:

- CR2**: Contains the constraint `boughtEnergyEquation_GNO_00`.
- Participation au nœud énergétique**: Contains the constraint `EnergyBalanceEquation_00`.
- CR3**: Contains the constraint `gridControlConstraint_GNO_00`.
- CR1**: Contains the constraint `gridPowerUBconstraint_GNO_00`.

FIGURE 8.6 – Partie de l'OP-XML illustrant x_1 pour l'instant $t = 0h$

```

<!-- batteryControlVariable_BNO_75 <= 1
0 <= batteryControlVariable_BNO_76 <= 1
0 <= batteryControlVariable_BNO_77 <= 1
0 <= batteryControlVariable_BNO_78 <= 1
0 <= batteryControlVariable_BNO_79 <= 1
0 <= batteryControlVariable_BNO_80 <= 1
0 <= batteryControlVariable_BNO_81 <= 1
0 <= batteryControlVariable_BNO_82 <= 1
0 <= batteryControlVariable_BNO_83 <= 1
0 <= batteryControlVariable_BNO_84 <= 1
0 <= batteryControlVariable_BNO_85 <= 1
0 <= batteryControlVariable_BNO_86 <= 1
0 <= batteryControlVariable_BNO_87 <= 1
0 <= batteryControlVariable_BNO_88 <= 1
0 <= batteryControlVariable_BNO_89 <= 1
0 <= batteryControlVariable_BNO_90 <= 1
0 <= batteryControlVariable_BNO_91 <= 1
0 <= batteryControlVariable_BNO_92 <= 1
0 <= batteryControlVariable_BNO_93 <= 1
0 <= batteryControlVariable_BNO_94 <= 1
0 <= batteryControlVariable_BNO_95 <= 1
0 <= batteryControlVariable_BNO_96 <= 1
0 <= batteryControlVariable_BNO_97 <= 1
0 <= batteryControlVariable_BNO_98 <= 1
0 <= batteryControlVariable_BNO_99 <= 1
0 <= batOutputEnergy_BNO_00 <= 100000
0 <= gridPowerVariable_GNO_00 <= 9
0 <= alpha_PVNO_00 <= 1
0 <= gridPowerVariable_GNO_01 <= 9
0 <= alpha_PVNO_01 <= 1
0 <= gridPowerVariable_GNO_02 <= 9
0 <= alpha_PVNO_02 <= 1
0 <= gridPowerVariable_GNO_03 <= 9
0 <= alpha_PVNO_03 <= 1
0 <= gridPowerVariable_GNO_04 <= 9
0 <= alpha_PVNO_04 <= 1
0 <= gridPowerVariable_GNO_05 <= 9
0 <= alpha_PVNO_05 <= 1
0 <= gridPowerVariable_GNO_06 <= 9
0 <= alpha_PVNO_06 <= 1
0 <= gridPowerVariable_GNO_07 <= 9
0 <= alpha_PVNO_07 <= 1
0 <= gridPowerVariable_GNO_08 <= 9
0 <= alpha_PVNO_08 <= 1

```

```

batteryOutputUBConstLabel_BNO_87: - 0.3 maximumCapacityInKiloWattHour_BNO
+ batOutputEnergy_BNO_87 <= 0
batteryOutputUBConstLabel_BNO_88: - 0.3 maximumCapacityInKiloWattHour_BNO
+ batOutputEnergy_BNO_88 <= 0
batteryOutputUBConstLabel_BNO_89: - 0.3 maximumCapacityInKiloWattHour_BNO
+ batOutputEnergy_BNO_89 <= 0
batteryOutputUBConstLabel_BNO_90: - 0.3 maximumCapacityInKiloWattHour_BNO
+ batOutputEnergy_BNO_90 <= 0
batteryOutputUBConstLabel_BNO_91: - 0.3 maximumCapacityInKiloWattHour_BNO
+ batOutputEnergy_BNO_91 <= 0
batteryOutputUBConstLabel_BNO_92: - 0.3 maximumCapacityInKiloWattHour_BNO
+ batOutputEnergy_BNO_92 <= 0
batteryOutputUBConstLabel_BNO_93: - 0.3 maximumCapacityInKiloWattHour_BNO
+ batOutputEnergy_BNO_93 <= 0
batteryOutputUBConstLabel_BNO_94: - 0.3 maximumCapacityInKiloWattHour_BNO
+ batOutputEnergy_BNO_94 <= 0
batteryOutputUBConstLabel_BNO_95: - 0.3 maximumCapacityInKiloWattHour_BNO
+ batOutputEnergy_BNO_95 <= 0
batteryOutputUBConstLabel_BNO_96: - 0.3 maximumCapacityInKiloWattHour_BNO
+ batOutputEnergy_BNO_96 <= 0
batteryOutputUBConstLabel_BNO_97: - 0.3 maximumCapacityInKiloWattHour_BNO
+ batOutputEnergy_BNO_97 <= 0
batteryOutputUBConstLabel_BNO_98: - 0.3 maximumCapacityInKiloWattHour_BNO
+ batOutputEnergy_BNO_98 <= 0
batteryOutputUBConstLabel_BNO_99: - 0.3 maximumCapacityInKiloWattHour_BNO
+ batOutputEnergy_BNO_99 <= 0
gridControlConstraint_GNO_00: gridPowerVariable_GNO_00
+ 100000 alpha_PVNO_00 <= 100000
gridControlConstraint_GNO_01: gridPowerVariable_GNO_01
+ 100000 alpha_PVNO_01 <= 100000
gridControlConstraint_GNO_02: gridPowerVariable_GNO_02
+ 100000 alpha_PVNO_02 <= 100000
gridControlConstraint_GNO_03: gridPowerVariable_GNO_03
+ 100000 alpha_PVNO_03 <= 100000
gridControlConstraint_GNO_04: gridPowerVariable_GNO_04
+ 100000 alpha_PVNO_04 <= 100000
gridControlConstraint_GNO_05: gridPowerVariable_GNO_05
+ 100000 alpha_PVNO_05 <= 100000

```

```

_EnergyBalanceEquation_00: - 30 boughtEnergy_GNO_00 - _ElectricBill = 0
- batInputEnergy_BNO_00
+ batOutputEnergy_BNO_00
+ gridPowerVariable_GNO_00 - pvSurplus_PVNO_00
- modifiedload_00 = 0
_EnergyBalanceEquation_01: - batInputEnergy_BNO_01
+ batOutputEnergy_BNO_01
+ gridPowerVariable_GNO_01 - pvSurplus_PVNO_01
- modifiedload_01 = 0

```

Participation de composant fournisseurs d'énergie dans le nœud énergétique

FIGURE 8.7 – Parties de l'OP-CPLEX résultantes de la projection automatique de l'OP-XML complet (x_1 pour l'instant $t = 0h$)

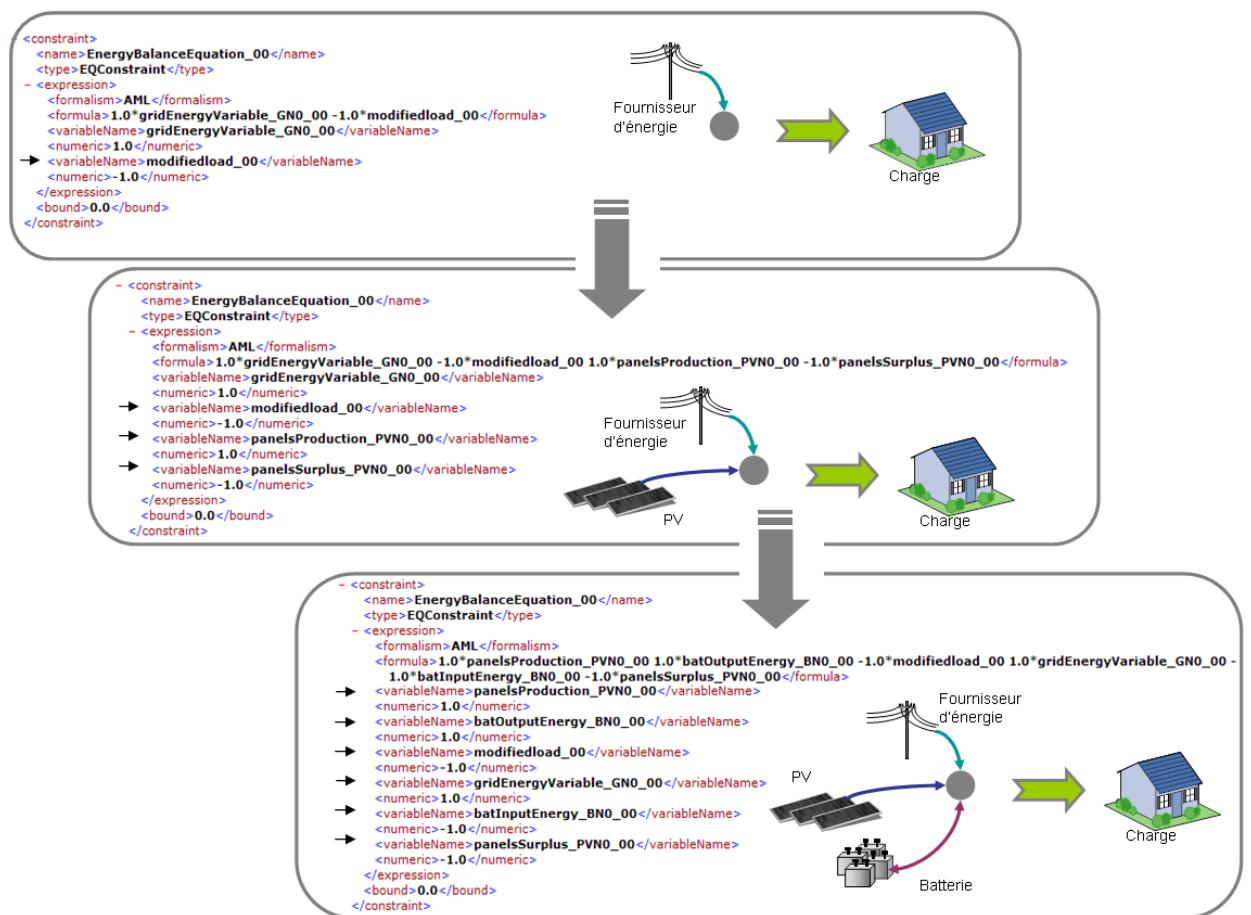


FIGURE 8.8 – Évolution de l'OP-XML lors de la composition du système bâtiment, au moment où le *concepteur des gestionnaires énergétiques* ajoute un composant via l'interface graphique, la contrainte du bilan énergétique dans l'OP-XML sera modifiée automatiquement par le générateur de problèmes

8.5 Outil pour le dimensionnement et la gestion des systèmes bâtiment

Nous avons concrétisé l'approche de génération automatique et de transformation de modèles par un outil développé en Java. Avec ce programme, nous pouvons générer automatiquement des problèmes de gestion pour l'application système bâtiment. Le programme est divisé en deux parties en conformité avec l'approche de génération de problèmes présentée dans cette thèse. La première partie est manuelle. Elle est assurée par le *concepteur de gestionnaires énergétiques* ou le *concepteur de systèmes bâtiment* pour l'aspect dimensionnement. Il s'agit de composer le système bâtiment en assemblant les sources et les charges de la configuration. Dans G-homeTech, la détection des équipements peut être faite automatiquement, car il est possible de lancer une procédure de découverte de la configuration d'un système bâtiment. La deuxième partie est automatique. Elle consiste à générer l'OP-XML complet puis de le projeter vers différents environnements d'optimisation.

L'intégration de nouveaux types de modèles peut être faite sans modifier les projecteurs. Il suffit qu'un acteur de type *concepteur de types* intègre ces nouveaux types dans la bibliothèque. La projection vers d'autres environnements de résolution demande l'intervention d'un *développeur logiciel* qui va ajouter les règles de transformation pour projeter vers cet environnement. Ceci n'affecte pas la bibliothèque de types qui est décrite sous la forme générique OP-XML.

Nous avons développé des concepteurs graphiques pour différents équipements du système bâtiment afin de permettre au *concepteur de gestionnaires énergétiques* de saisir les données numériques des paramètres propres au système étudié (les a_i dans les modèles analytiques), mais aussi pour spécifier l'environnement cible de la transformation (CPLEX ou CADES ici, ou GLPK et CPLEX pour G-homeTech).

Les *concepteurs de gestionnaires énergétiques* spécifient via cette interface l'objectif recherché en phase d'exploitation (figure 8.9 : *Sizing* ou *Management*).

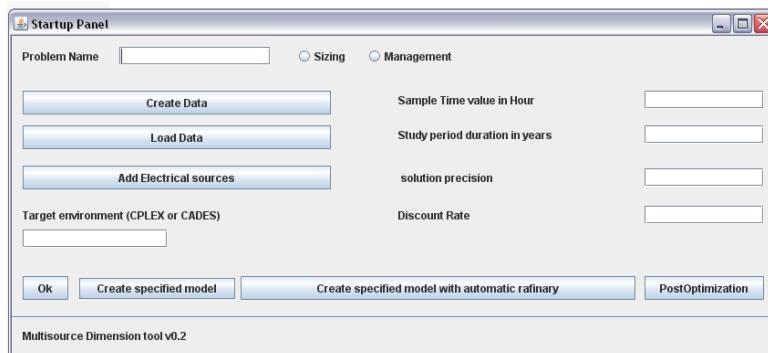
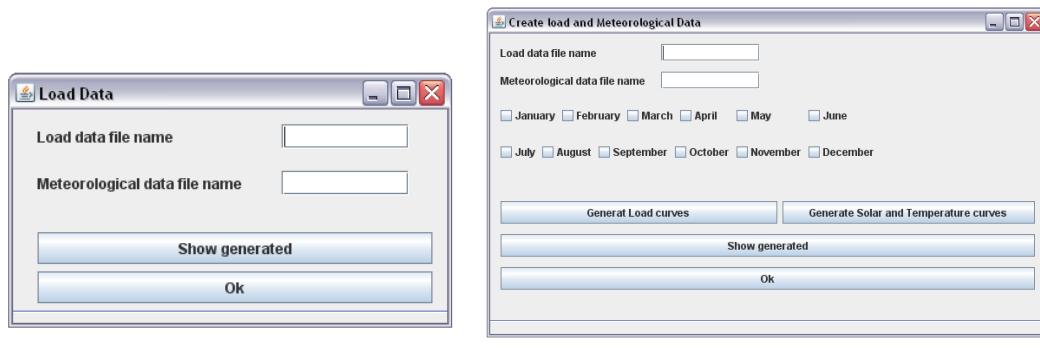


FIGURE 8.9 – Interface graphique pour l'outil de gestion et de dimensionnement des sources et charges

Les *concepteurs de gestionnaires énergétiques* peuvent ajouter des données météorologiques ainsi que des données de consommation typique des charges (voir la figure 8.10).

Les *concepteurs de gestionnaires énergétiques* peuvent éditer le système bâtiment grâce



(a) Téléchargement des données existantes

(b) Génération des données à partir de celles de la base données de l'outil

FIGURE 8.10 – Interface graphique pour l’acquisition de données météorologiques

à un éditeur graphique. Ils peuvent ajouter ou supprimer une source électrique du système comme celles présentées dans les figures 14.1, 8.5 et 14.2. Une fois terminée la composition du système, le concepteur peut choisir l’environnement d’optimisation cible de transformation (en spécifiant soit CPLEX, soit CADES).

Nous avons développé une autre interface graphique pour afficher les résultats d’optimisation obtenus par CPLEX (CADES fourni déjà un module pour afficher les résultats d’optimisation *CADES post Optimizer*). Notre prototype d’interface est présenté dans la figure 8.11.

Grâce à la projection vers ces deux environnements, nous avons trouvé un phénomène intéressant pour la gestion des sources, l’analyse de ce phénomène est effectuée dans le chapitre suivant.

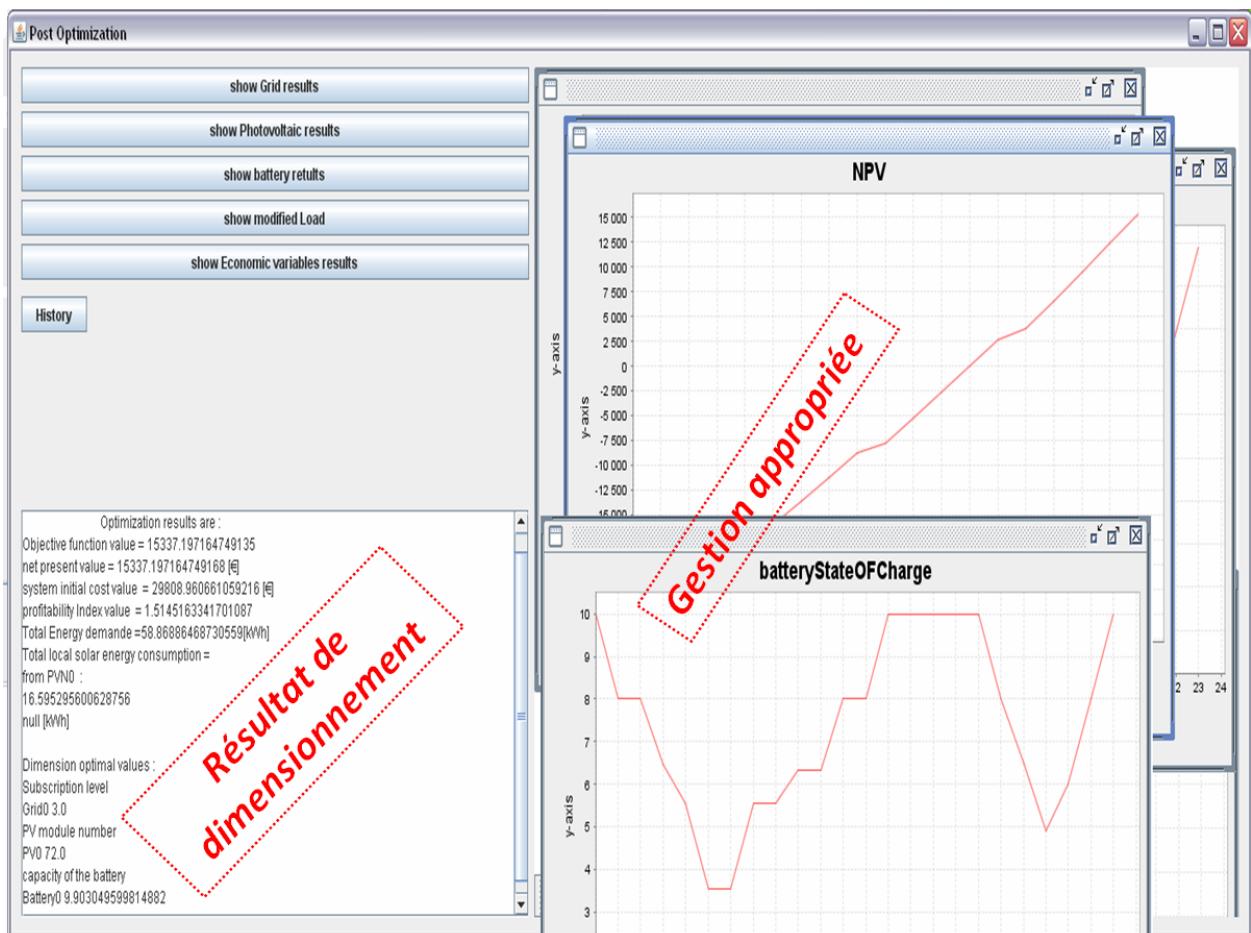


FIGURE 8.11 – Aperçu de l'interface graphique pour analyser les résultats obtenus par CPLEX

8.6 Conclusion

Comme dans la problématique de dimensionnement de systèmes bâtiments, la résolution de problèmes de gestion révèle un besoin en terme de génération automatique problème, plus marqué encore que pour l'aspect dimensionnement et un besoin de projection vers différents environnements de résolution.

Nous avons proposé dans ce chapitre une approche basée sur une collaboration entre trois métiers spécialisés pour concevoir des systèmes de gestion énergétique. Les mêmes principes que ceux énoncés pour le développement d'une plateforme dédiée au dimensionnement, composée d'un générateur de problèmes, de projecteurs et de générateurs d'éditeurs, ont été adoptés pour résoudre la problématique de gestion : c'est le *développeur logiciel* qui porte cette charge. Le *concepteur de types* peut alors créer des interfaces ou concepteurs graphiques pour chaque type de modèles. Le *concepteur de gestionnaires énergétiques* peut alors utiliser ces interfaces pour paramétriser les types, puis récupérer automatiquement les modèles de niveau M0, pour le solveur qu'il aura choisi.

Cette approche rend le travail du *concepteur de gestionnaires énergétiques* plus structuré. En effet, il est possible d'appréhender des problèmes de très grande complexité (plusieurs milliers de variables) en permettant une spécification des métiers.

La projection vers CPLEX et CADES a montré un avantage de CPLEX au niveau de la rapidité de résolution pour le même problème.

Parmi les perspectives figure la nécessité de développer des éditeurs de création de nouveaux type de modèles qui pourront enrichir la bibliothèque de types de modèles, comme par exemple un type éolienne ou un type voiture électrique.

L'application de cette approche peut être étendue à un niveau plus complexe que le bâtiment : par exemple à la gestion d'un groupe de bâtiments.

Dans le chapitre suivant nous allons résoudre un problème de gestion dans deux environnements de résolution différents et montrer que les solutions obtenues ne coïncident pas nécessairement.

Chapitre 9

Caractérisation et détection des problèmes de gestion à multiples solutions

La façon dont on trouve n'est pas celle dont on prouve

Albert Einstein

SOMMAIRE

9.1	INTRODUCTION	136
9.2	MISE EN ÉVIDENCE D'UN EFFET INDÉSIRABLE GRÂCE À LA PROJECTION MULTI-SOLVEUR : L'EFFET W	136
9.2.1	Problème d'optimisation à projeter	136
9.2.2	Résultats obtenus par projection multi-solveur : solutions équivalentes et mise en évidence de l'effet W	138
9.2.3	Cas de Multiple solutions équivalentes : pas d'effet W	138
9.2.4	Cas de Multiples solutions équivalentes : apparition de l'effet W	144
9.3	RECHERCHE DES CONDITIONS POUR LESQUELLES SE PRODUIT L'EFFET W : PROBLÈMES D'OPTIMISATION À SOLUTIONS MULTIPLES	146
9.3.1	Définition du problème à multiples solutions équivalentes	146
9.3.2	Première analyse des conditions conduisant à de multiples solutions équivalentes	147
9.4	PROPOSITION DE MÉTHODES ET D'ALGORITHMES POUR DÉTECTOR LES PROBLÈMES D'OPTIMISATION À MULTIPLES SOLUTIONS ÉQUIVALENTES . .	148
9.4.1	Une approche par une optimisation mono-paramétrique	149
9.4.2	Une approche par une optimisation multi-paramétrique (algorithme D_{max})	153
9.5	PERSPECTIVE POUR APPRÉHENDER LES PROBLÈMES D'OPTIMISATION À SOLUTIONS MULTIPLES POUR ACCROÎTRE LA ROBUSTESSE DE LA STRATÉGIE DE GESTION	163
9.5.1	Proposition pour découvrir les solutions équivalentes	164
9.5.2	Proposition pour utiliser les solutions équivalentes	166
9.6	CONCLUSION	167

Résumé

Dans ce chapitre, nous présentons un aspect lié à la formulation des problèmes d'optimisation. La projection de problèmes multi-solveur met en évidence que le même problème résolu par différents solveurs peut conduire à différentes solutions équivalentes, au sens du critère choisi, au lieu d'une solution unique commune à tous les environnements. Bien que ces solutions aient la même valeur de fonction objectif, elles correspondent à différentes stratégies de commande. Pratiquement, nous utilisons un seul solveur pour résoudre le problème de gestion. En conséquence, il est préférable de détecter le cas où ces solutions sont possibles sans passer par la projection multi

solveur. Nous allons étudier ces problèmes d'optimisation pour deux raisons ; d'une part, il y a un risque pour que certaines solutions induisent des variations brutales dans la stratégie de commande engendrant ainsi une usure prématuée des composants sensibles du système comme les batteries ; d'autre part, nous proposons de profiter de l'existence de solutions équivalentes pour améliorer la robustesse du système de commande. Des algorithmes permettant de détecter et de prévenir les gestionnaires de systèmes bâtiments sur l'existence de telles solutions sont présentés dans ce chapitre.

9.1 Introduction

Dans les chapitres précédents, nous avons illustré comment il est possible de générer automatiquement des problèmes de dimensionnement et de gestion d'équipements pour différents environnements d'optimisation. Nous allons voir dans ce chapitre que cette projection multi solver nous a permis de mettre en évidence un phénomène important pour la gestion d'énergie.

Nous avons pris un exemple de problème de gestion d'un système bâtiment présenté en 9.1, puis nous avons implémenté les techniques de projections présentées précédemment pour le résoudre en utilisant les solveurs CPLEX et CADES. En comparant les résultats d'optimisation de ces deux solveurs, nous avons mis en évidence ils peuvent trouver deux solutions distinctes bien qu'elles donnent la même valeur de la fonction objectif. Pratiquement, cela signifie qu'il y a deux stratégies de commande différentes.

Du point de vue de la physique du système, nous avons constaté que commuter d'une solution à l'autre, au fur et à mesure des recalculs de plans anticipatifs, peut accélérer la fatigue de certains éléments. Il est intéressant de détecter les situations où un ensemble de solutions équivalentes existent. Ce chapitre vise à proposer des algorithmes pour résoudre ce problème.

Bien que ce phénomène ait été mis en évidence par la projection vers différents solveurs, les algorithmes recherchés doivent détecter le phénomène sans avoir recours à projections multiples.

Dans ce chapitre, nous caractérisons et définissons les problèmes d'optimisation qui conduisent à des ensembles de solutions équivalentes. Deux algorithmes de détection de ces situations par optimisation sont proposés. Le premier pour les environnements de résolution de problèmes non linéaires comme CADES, l'autre pour les environnements linéaires comme CPLEX. Pour finir, nous proposons une méthode permettant d'exploiter l'existence de solutions équivalentes pour accroître la robustesse de la gestion énergétique des systèmes bâtiments.

9.2 Mise en évidence d'un effet indésirable grâce à la projection multi-solveur : l'effet W

9.2.1 Problème d'optimisation à projeter

Étudions un problème de gestion des flux énergétiques dans un système bâtiment composé d'un réseau de distribution, de panneaux solaires, d'un système de stockage et d'une charge électrique. Le système peut revendre ou acheter l'énergie du réseau. Il est illustré sur la figure 9.1.

Le problème de gestion est un problème linéaire dont la formulation est donnée par :

$$\forall t \in [0..23],$$

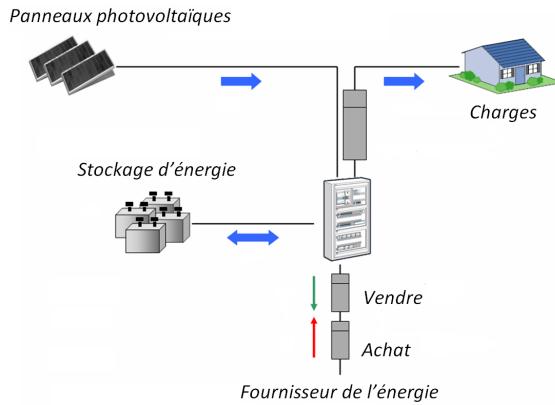


FIGURE 9.1 – Système bâtiment étudié

$$\begin{aligned}
 MaxZ : & \sum_{t=23}^{t=0} (x_3(t) \times p_s(t) - x_1(t) \times p_g(t)) \\
 & x_1(t) + x_2(t) - x_3(t) = P_{load}(t) - P_{pv}(t) \\
 & x_4(t) - x_4(t-1) - x_2(t) = 0 \\
 & 0 \leq x_1(t) \leq P_{abon} \\
 & 0 \leq x_3(t) \leq P_{pv}(t) \\
 & -Q_{max} \times a_2 \leq x_2(t) \leq Q_{max} \times a_3 \\
 & Q_{max} \times a_1 \leq x_4(t) \leq Q_{max}
 \end{aligned} \tag{9.1}$$

avec :

- a_1 : pourcentage de la capacité (charge) maximale de la batterie en dessous duquel il ne faut pas descendre
- a_2 : proportion avec laquelle on limite l'énergie qu'on peut introduire dans la batterie à chaque pas de temps, cette limite est calculée comme une proportion de la capacité maximale de la batterie.
- a_3 : proportion avec laquelle on limite l'énergie qu'on peut retirer de la batterie à chaque pas de temps, cette limite est calculée comme une proportion de la capacité maximale de la batterie.
- Q_{max} : capacité de la batterie en kWh
- x_1 : énergie achetée au fournisseur d'énergie en kWh
- x_2 : énergie de charge et de décharge de la batterie en kWh.
- x_3 : surplus en kWh revendu au fournisseur d'énergie
- x_4 : état de la charge de la batterie (*State Of Charge*)
- P_{pv} : production des panneaux solaires en kWh
- P_{load} : charge électrique en kWh
- P_{abon} : limite autorisée par l'abonnement avec le fournisseur d'énergie
- p_g : prix de l'énergie acheté au fournisseur d'énergie en euro/kWh
- p_s : prix de revente du surplus en euro/kWh.

Nous avons représenté les données pour le problème 9.1 dans la figure 9.2. Il s'agit des courbes d'ensoleillement et de température pour 24h, de la courbe de la charge électrique et des prix d'achat et de revente de l'énergie.

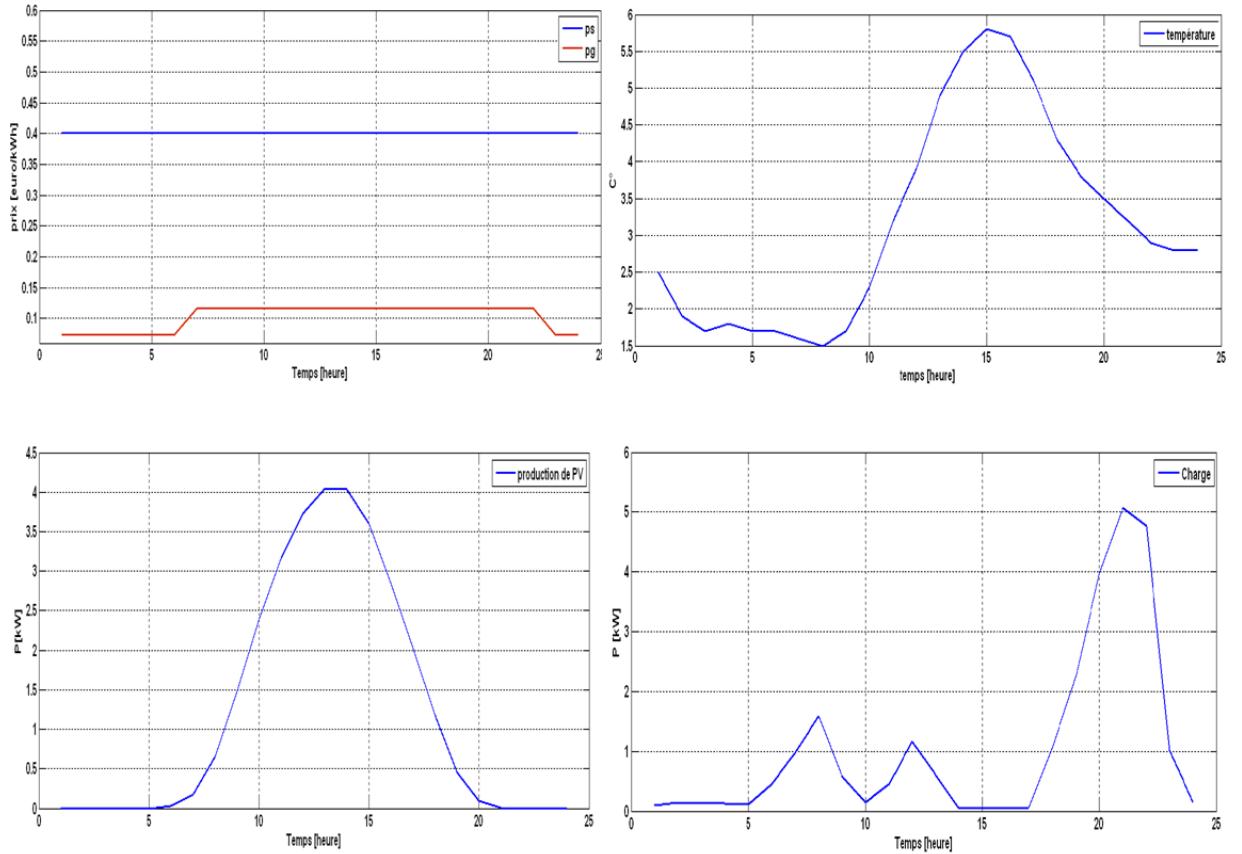


FIGURE 9.2 – Données de problème 9.1

9.2.2 Résultats obtenus par projection multi-solveur : solutions équivalentes et mise en évidence de l'effet W

9.2.3 Cas de Multiple solutions équivalentes : pas d'effet W

Nous avons projeté ce problème 9.1 vers CPLEX. Comme le prix d'achat de l'électricité est bas au début de la journée, la solution fournie par CPLEX favorise la charge de la batterie par l'énergie du réseau (entre 2h00 et 8h00, cf. figure 9.3). Cela induit une perte légère comme la figure 9.4 le montre. La batterie est ensuite utilisée pour alimenter la charge électrique du système bâtiment car le prix d'achat est plus important dans cette période entre 6h00 et 18h00. En même temps, toute l'énergie produite par les panneaux solaires est revendue. Cela permet de gagner d'argent (cf. figure 9.4). En effet, le système permet de gagner 9.78 euros durant la journée considérée. Cette valeur doit être relativisée car le coût d'investissement n'est pas pris en compte dans ce cas.

La solution obtenu en CPLEX est illustrée dans la figure 9.5.

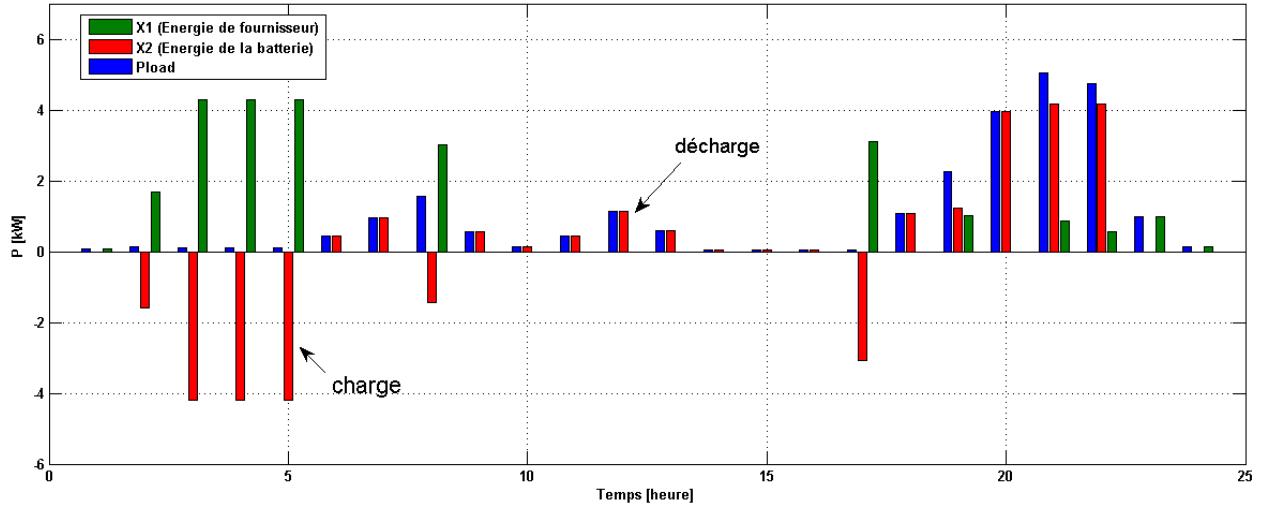


FIGURE 9.3 – Utilisation de la batterie par CPLEX pour alimenter la charge électrique

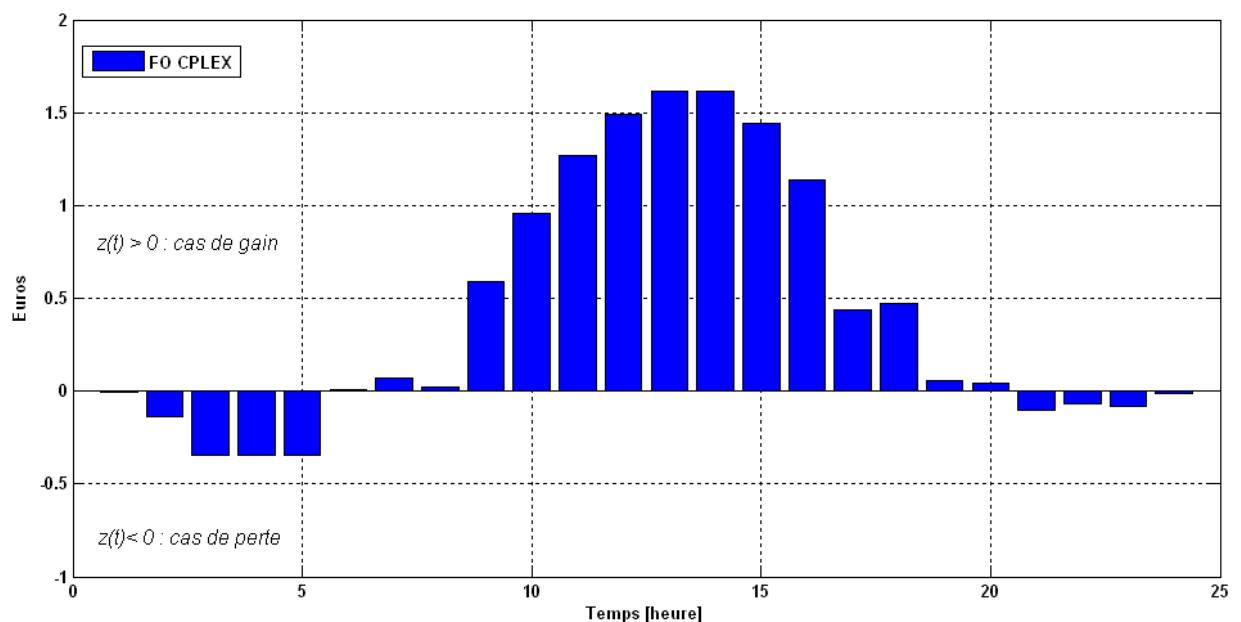
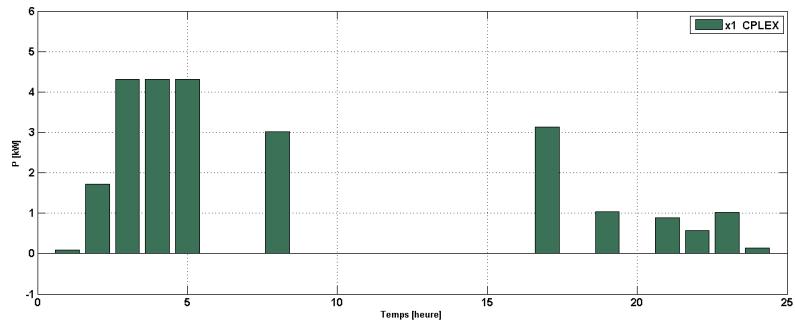
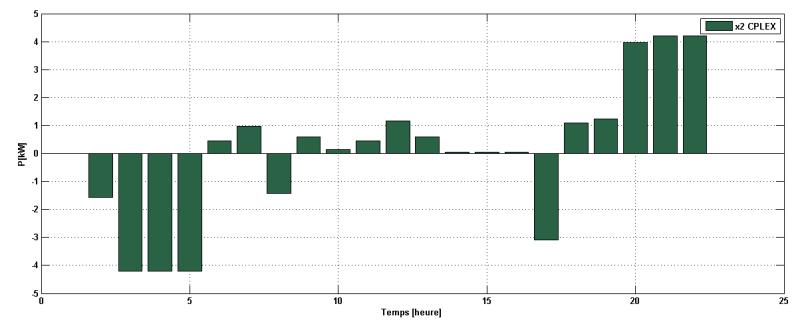


FIGURE 9.4 – Illustration de gain et de la perte en euros sur les 24h trouvés par CPLEX

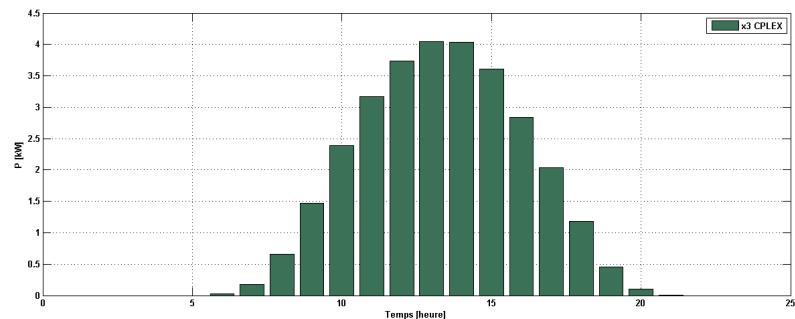
9.2. Mise en évidence d'un effet indésirable grâce à la projection multi-solveur : l'effet **M1**



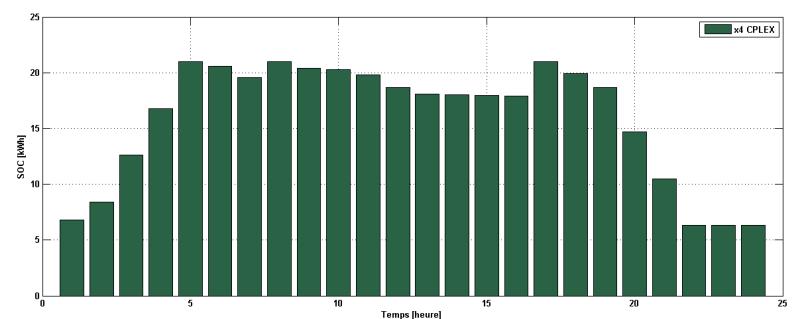
(a) Énergie du fournisseur



(b) Energie de la batterie



(c) Surplus



(d) État de la charge de la batterie

FIGURE 9.5 – Résolution de problème 9.1 avec CPLEX

Le même problème a été projeté vers CADES. En comparant le résultat obtenu avec celui issu de CPLEX, nous voyons que la solution de CADES est différente de la solution CPLEX. Il y a des solutions équivalentes (même valeur de fonction objectif, mais stratégies de pilotage différentes) voir figure 9.6.

9.2. Mise en évidence d'un effet indésirable grâce à la projection multi-solveur : l'effet W3

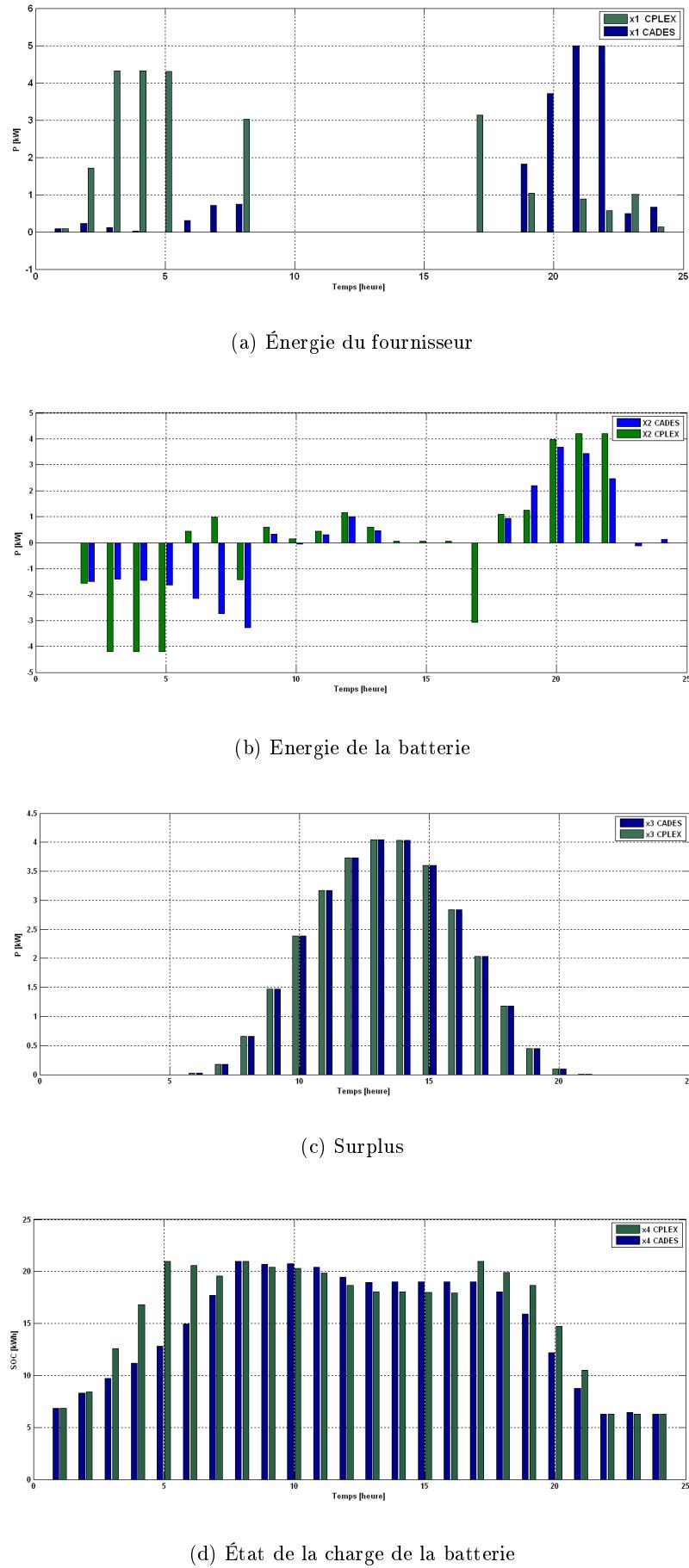


FIGURE 9.6 – Comparaison entre la résolution de 9.1 avec CADES et CPLEX

Fonction objectif	CADES	CPLEX
Z [€]	9.78	9.78

TABLE 9.1 – Ensemble de solutions équivalentes donnant la même valeur à la fonction objectif

La solution CADES consiste à charger la batterie doucement et sans faire de cycles rapides de charge et décharge. Cela est préférable en pratique car la durée de vie de la batterie dépend du nombre de cycles de charge et de décharge qu'elle subit Kiehne (2003). Néanmoins, La figure 9.7 illustre que ces deux solutions donnent des valeurs instantanées différentes de la fonction objectif $z(t)$, mais la somme sur les 24h est identique. En conséquence, ces deux solutions, d'un point de vue mathématique, ont la même valeur de la fonction objectif Z : elles sont donc équivalentes (voir tableau 9.1).

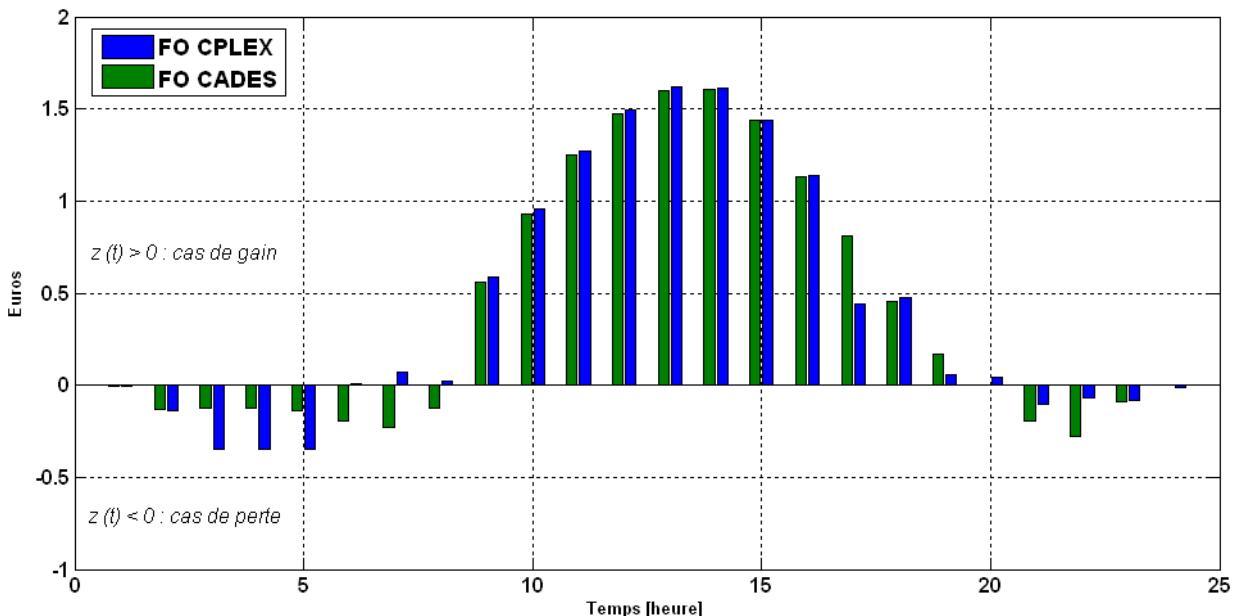


FIGURE 9.7 – Comparaison entre le gain et la perte en euros sur les 24h trouvés par CPLEX et CADES

On constate que des solutions égales d'un point de vue de la valeur de la fonction objectif peuvent se traduire par des stratégies de commande différentes.

9.2.4 Cas de Multiples solutions équivalentes : apparition de l'effet W

Pour augmenter l'autonomie du système au lendemain ($J+1$), la charge de la batterie au début et à la fin d'un jour (J) doit être identique. Nous avons modifié le problème 9.1 en ajoutant une nouvelle contrainte sur l'état de la charge de la batterie x_4 . Il s'agit de figer la valeur de l'état de charge à la fin de la journée à la valeur initiale (au début de la journée, cf. équation 9.2). Dans le pire scénario, la journée peut commencer avec une batterie faiblement chargée (par exemple : 30% de sa capacité). Avec ce scénario, nous avons obtenu la solution de la figure 9.8.

$$x_4(\text{Studyperiod}) = x_4(0) \quad (9.2)$$

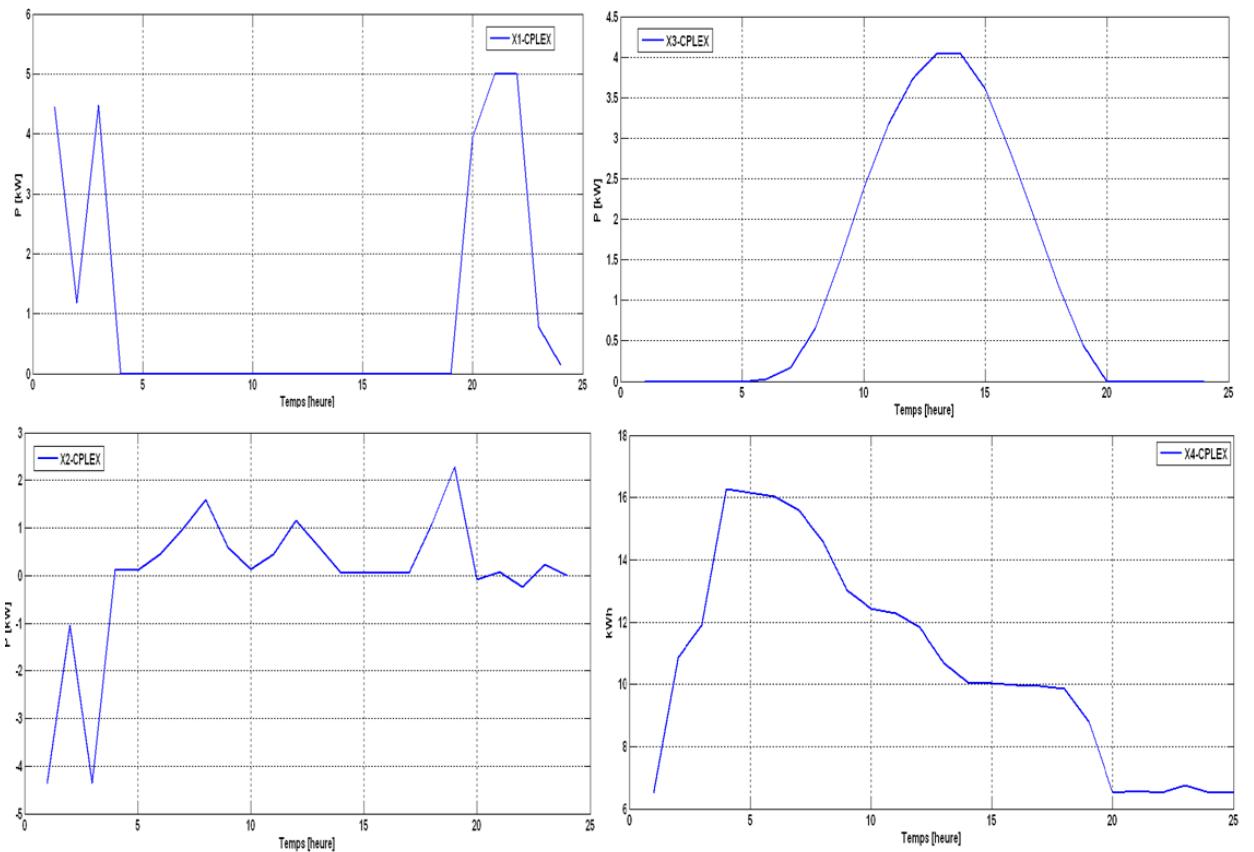


FIGURE 9.8 – Gestion des sources trouvée par CPLEX pour le problème 9.1

Le même problème a été projeté vers CADES. En comparant le résultat obtenu avec celui issu de CPLEX, nous voyons que la solution de CPLEX subit une oscillation en forme de la lettre W au niveau de l'énergie de la batterie et de l'énergie échangée avec le réseau, tout en atteignant la même valeur d'état de charge de la batterie à la fin de la journée. Cette variation n'apparaissait pas dans la solution CADES (voir figure 9.9).

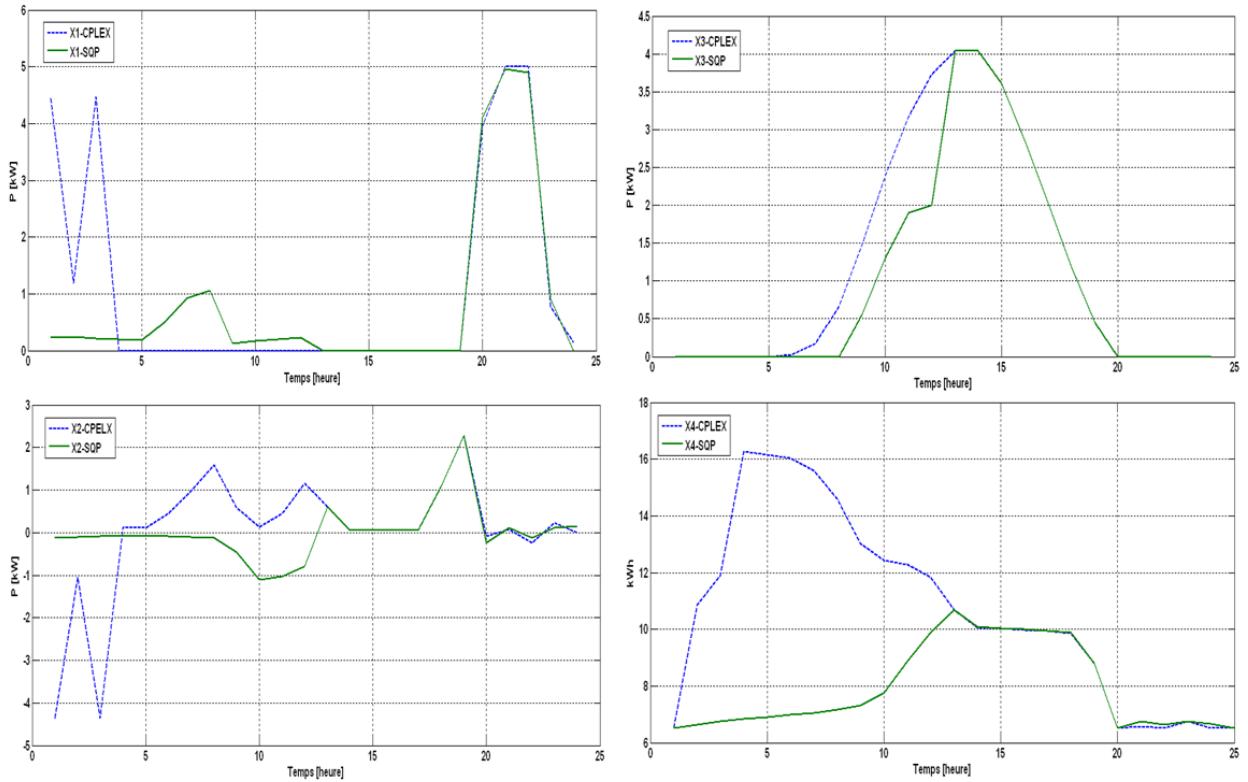


FIGURE 9.9 – Gestion des sources pour le problème 9.1 : en bleu CPLEX et en vert CADES

Constat :

1. Les problèmes de gestion (comme par exemple 9.1) peuvent avoir un ensemble de solutions qui conduisent à une même valeur de la fonction objectif.
2. Avoir un ensemble de solutions comporte le risque que certaines solutions usent pré-maturément certains composants dans le système multi sources.
3. La variation en forme de W peut être due à la formulation du problème et à la nature de l'algorithme du Simplexe (cf. fonctionnement 12.3) car Simplexe cherche la solution optimale parmi les points qui sont aux extrémités d'une région réalisable. En conséquence, une commutation d'un point solution vers un autre peut engendrer une variation brutale de commande (W). Alors que CADES, avec le solveur de type SQP (qui est un algorithme d'optimisation locale), favorise la recherche de solutions proches et pas nécessairement aux extrêmes de l'espace réalisable. C'est pour ça que la gestion donnée par CADES est plus douce que celle obtenue avec CPLEX.
4. La découverte de ces solutions ne doit pas forcément être mise en évidence par une projection multi solveur.

9.3 Recherche des conditions pour lesquelles se produit l'effet W : problèmes d'optimisation à solutions multiples

Généralement nous formulons un problème d'optimisation en définissant le domaine des variables, les contraintes et un critère à minimiser ou à maximiser afin de trouver l'optimum parmi toutes les solutions faisables. Comme nous l'avons évoqué précédemment, la résolution de ce problème peut conduire à un ensemble de solutions équivalentes et non à une unique solution.

On va faire le postulat que l'existence de solutions multiples au problème d'optimisation indique qu'il y a un risque d'avoir un effet W. C'est pourquoi, dans notre recherche, nous allons essayer de détecter l'existence de solutions équivalentes comme un indicateur de risque d'effet W.

9.3.1 Définition du problème à multiples solutions équivalentes

Nous pouvons décrire un problème d'optimisation admettant plusieurs solutions équivalentes avec la formulation suivante ([Warkozek, Ploix, Wurtz & Jacomino 2010b](#)) :

X^* est solution de

$$\begin{aligned} \text{Min } & f(X) \\ \text{avec } & AX \leq B \\ \text{et } & \exists S = \{X^*\} \text{ tel que } \text{card}(S) > 1 \end{aligned} \tag{9.3}$$

Exemple trivial

Supposons que l'on ait le problème suivant :

$$\begin{aligned} \text{Min } & z = x_1 + x_2 \\ C_1 : & x_1 + x_2 \geq 4 \\ C_2 : & x_1 \leq 3 \\ C_3 : & x_2 \leq 4 \\ & x_1, x_2 \geq 0 \end{aligned} \tag{9.4}$$

Le problème ainsi formulé admet un ensemble de solutions équivalentes qui sont :

$$\begin{aligned} \forall x_1 \in [0, 3], \quad & \forall x_2 \in [1, 4], \\ & x_1 + x_2 = 4 \end{aligned} \tag{9.5}$$

La figure [9.10](#) illustre ce cas où tous les points entre A et B minimisent la fonction objectif (z). Nous supposons qu'il y ait un risque d'effet W. Une projection de ce problème vers CPLEX et CADES a confirmé qu'il y a des multiples solutions équivalentes. la figure [9.11](#) montre ces deux solutions.

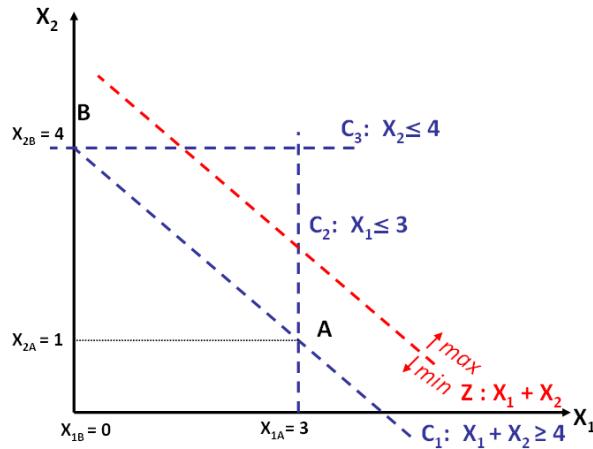


FIGURE 9.10 – Problème à multiples solutions équivalentes : plusieurs valeurs des variables donnent la même valeur à la fonction objectif

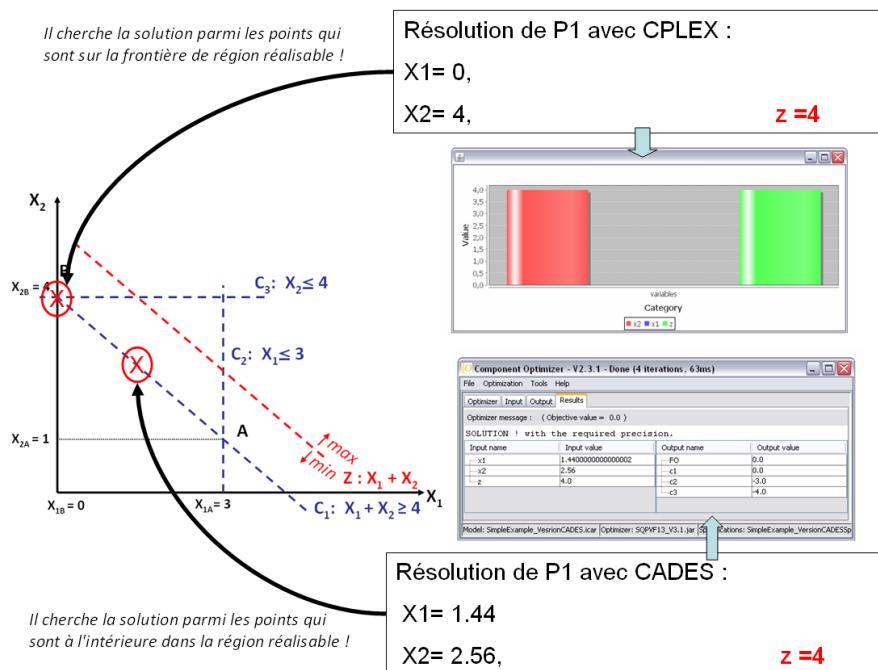


FIGURE 9.11 – Problème à multiple solutions équivalentes : résolution d'exemple 9.4

9.3.2 Première analyse des conditions conduisant à de multiples solutions équivalentes

Pour détecter les ensembles de solutions multiples, on peut passer par une analyse structurelle du problème à résoudre. L’objectif de cette analyse est d’analyser avant même la résolution s’il y a une solution ou plusieurs solutions optimales. Cette analyse structurelle va permettre de répondre aux trois questions suivantes :

1. **Est-ce que le problème est juste-déterminé ou sur-déterminé ?**

Le rapport entre le nombre de contraintes et le nombre de variables est important pour la résolution d’un problème d’optimisation. Souvent le problème d’optimisation a un nombre de contraintes N_c plus important que le nombre de variables N_v . Si le nombre de contraintes est égal ou supérieur au nombre de variables, le problème est juste-déterminé ou sur-déterminé : il n’admet pas nécessairement de solutions et, en tout état de cause, il ne peut pas conduire à de multiples solutions équivalentes.

2. **Est-ce que le problème est sous déterminé ?**

Le problème sous déterminé est un problème où $N_c < N_v$. Le problème est donc sous-contraint et les chances d’avoir des solutions multiples sont importantes.

3. **Est-ce qu’il existe une contrainte parallèle à la fonction objectif ?**

Lorsqu’un problème est sous-déterminé, ce qui est généralement le cas en optimisation, il faut d’autres outils pour anticiper l’existence de solutions multiples. Nous avons trouvé deux causes qui induisent l’existence de solutions multiples équivalentes. La première apparaît quand la fonction objectif est parallèle à une direction des variables de décision : il s’agit des contraintes liées à la positivité des variables de décision. L’autre apparaît quand la fonction objectif est parallèle à une contrainte.

9.4 Proposition de méthodes et d’algorithmes pour détecter les problèmes d’optimisation à multiples solutions équivalentes

Nous avons effectué deux types d’étude pour la détection de possibles solutions multiples. La première est structurelle : il s’agit d’implémenter la décomposition de Dulmage Mendelsohn ([Dulmage & Mendelsohn 1959](#)) (DM) sur la structure du problème d’optimisation où des variables virtuelles positives ont été introduites pour ramener les contraintes inégalités à des contraintes égalités (reformulation nécessaire pour appliquer la décomposition DM). Le but est de profiter de l’algorithme de décomposition DM¹ pour trouver si les variables sont toutes dans la partie sur-déterminée ou juste-déterminée, ceci signifie que, s’il existe une solution, elle est unique et qu’il n’y a pas de risque de solutions multiples pouvant amener à un effet W.

1. La décomposition permet notamment de trouver les parties sur-déterminée, juste-déterminée et sous-déterminée

S'il existe une partie sous-déterminée, la seule analyse structurelle n'est pas suffisante pour savoir si le problème admet des multiples solutions équivalentes ([Warkozek, Ploix, Wurtz & Jacobino 2010a](#)). Pour aller plus loin dans l'analyse, il faut prendre la nature des contraintes et de la fonction objectif. De plus amples discussions apparaissent dans l'annexe [17.1](#).

Nous proposons de compléter l'analyse structurelle par une approche plus spécifique qui s'appuie sur l'optimisation. Il s'agit de trouver par optimisation si le problème admet de multiples solutions. Nous allons montrer ces détails dans la suite.

9.4.1 Une approche par une optimisation mono-paramétrique

L'analyse que nous proposons dans ce paragraphe est faite après la résolution de problèmes d'optimisation. Comme nous proposons au gestionnaire la projection vers au moins deux environnements, ceci peut nous permettre de le prévenir que la solution qu'il obtient par un solveur est unique ou pas.

Examinons la sensibilité de la fonction objectif vis à vis de la variation des variables de décision. Nous essayons de trouver un voisinage d'une solution optimale qui ne change pas la valeur de la fonction objectif. Comme nous l'avons évoqué précédemment, si la fonction objectif est parallèle à une direction d'une de ses variables de décision ², il y a un risque d'avoir plusieurs solutions équivalentes. Dans ce cas, le gradient de la fonction objectif est nul :

$$\nabla \vec{FO} = \frac{\delta FO}{\delta x_i} dx_i = 0 \quad (9.6)$$

Cela signifie que x_i pourrait varier sans affecter la valeur de la fonction objectif.

Nous pouvons examiner la pente de la fonction objectif numériquement avec ce qu'on appelle une optimisation mono-paramétrique (OMP) sur une direction comme dans ([Warkozek et al. 2009](#)). Nous choisissons la direction d'une variable, puis nous nous positionnons sur une solution optimale et nous faisons un pas dans une région faisable qui satisfait (9.6). Nous procédons alors à une nouvelle optimisation pour le point ainsi trouvé et ainsi de suite. Si les valeurs de la fonction objectif obtenue par chaque optimisation correspondent à la valeur originale, l'existence de solutions équivalentes multiples est démontrée. L'algorithme correspondant à cette approche est illustré dans la figure [9.12](#).

Le choix de variations suivant les variables peut être fait judicieusement en choisissant en priorité les variables appartenant à des blocs sous-déterminés de la décomposition DM, car cela économise le nombre d'optimisations. Ainsi nous proposons de commencer par la direction la moins contrainte, autrement dit, par la variable la moins présente dans les contraintes et pour laquelle on peut supposer que les chances de trouver des solutions multiples sont importantes.

Exemple

Illustrons cela pour un exemple de problème d'optimisation. Nous indiquons les différentes étapes de l'algorithme proposé. Supposons que l'on ait le problème d'optimisation suivant :

2. correspondant aux contraintes de positivité des variables de décision

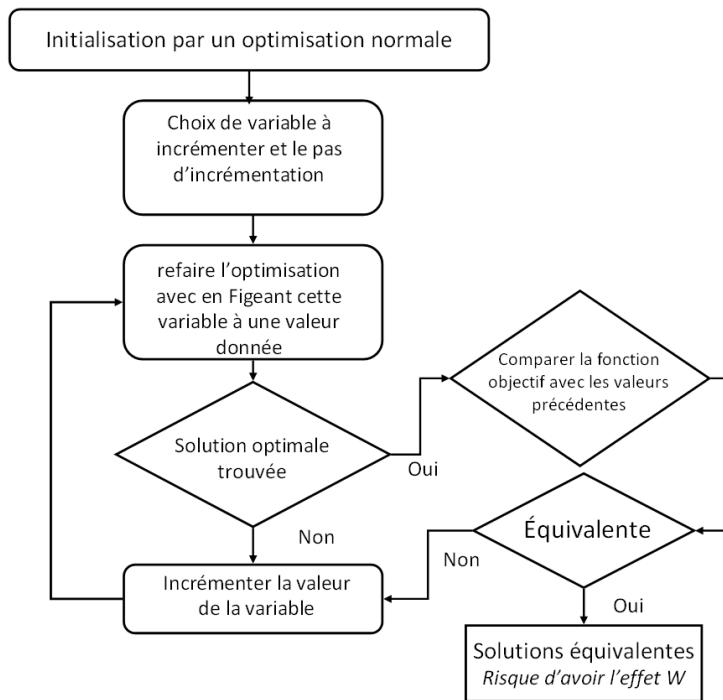


FIGURE 9.12 – Algorithme pour détecter l'existence de solutions équivalentes par une optimisation mono-paramétrique

$$\begin{aligned}
 & \text{Min } x_1 + x_2 \\
 & c1 : x_1 + 3x_2 + x_3 \leq 12 \\
 & c2 : x_1 + x_2 \leq 8 \\
 & c3 : x_1 \leq 4 \\
 & c4 : x_2 \leq 3 \\
 & c5 : x_3 \leq 6
 \end{aligned} \tag{9.7}$$

Pour appliquer la décomposition *Dulmage-Mendelsohn* il faut réécrire le système sous forme de contraintes égalités (cf. annexe 17.1.3), nous aurons ainsi :

$$\begin{aligned}
 & \text{Min } x_1 + x_2 \\
 & c1 : x_1 + 3x_2 + x_3 + \epsilon_1 = 12 \\
 & c2 : x_1 + x_2 + \epsilon_2 = 8 \\
 & c3 : x_1 + \epsilon_3 = 4 \\
 & c4 : x_2 + \epsilon_4 = 3 \\
 & c5 : x_3 + \epsilon_5 = 6
 \end{aligned} \tag{9.8}$$

avec $x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, \epsilon_1 \geq 0, \epsilon_2 \geq 0, \epsilon_3 \geq 0, \epsilon_4 \geq 0$ et $\epsilon_5 \geq 0$.

La matrice d'incidence s'écrit :

	x_1	x_2	x_3	ϵ_1	ϵ_2	ϵ_3	ϵ_4	ϵ_5	
c_1	1	1	1	1	0	0	0	0	
c_2	1	1	0	0	1	0	0	0	
c_3	1	0	0	0	0	1	0	0	
c_4	0	1	0	0	0	0	1	0	
c_5	0	0	1	0	0	0	0	1	

(9.9)

Il s'agit donc d'un système sous-déterminé (5 contraintes égalités pour 8 variables). La décomposition de Delmage-Mendelsohn conduit à (voir annexe 17.1.3) :

	ϵ_1	ϵ_2	ϵ_5	x_1	x_2	x_3	ϵ_3	ϵ_4	
c_1	1	0	0	1	1	1	0	0	
c_2	0	1	0	1	1	0	0	0	
c_5	0	0	1	0	0	1	0	0	
c_3	0	0	0	1	0	0	1	0	
c_4	0	0	0	0	1	0	0	1	

(9.10)

Il existe un bloc juste-déterminé : $(\epsilon_1, \epsilon_2, \epsilon_5, x_1, x_2) \times (c_1, c_2, c_5, c_3, c_4)$. Selon cette décomposition, il s'agit de l'ensemble des variables qui sont déterminées complètement par les contraintes du problème (couplage complet)³. Dans ce cadre les variables x_3 , ϵ_3 et ϵ_4 sont des variables non-déterminées. Comme seule la variable x_3 nous intéresse, nous en déduisons qu'il faut commencer par étudier la direction de x_3 car les valeurs de x_1 et de x_2 sont entièrement déterminées par les 5 contraintes.

Pour commencer l'optimisation, il faut choisir la bonne valeur du pas de variation de x_3 , qui ne doit pas violer la contrainte $x_3 \leq 6$ ni provoquer d'erreur d'arrondi numérique. Nous avons trouvé le cas de solutions multiples avec un pas valant 2 (ce qui veut dire que x_3 a pris les valeurs 1, 3 et 5) en vérifiant la valeur de la fonction objectif obtenue pour chaque cas⁴.

Le résultat du test de voisinage indique qu'il y a plusieurs solutions équivalentes. Un ensemble des solutions équivalentes (S^1 , S^2 et S^3) est présenté dans le tableau 9.2. En faisant apparaître les contraintes actives (voir l'encart), nous pouvons remarquer qu'il y a toujours N_c contraintes actives $\leq N_v$, ce qui est une condition nécessaire mais n'est pas suffisante pour avoir des solutions multiples.

Définition d'une contrainte active (Philip et al. 1981) : Soit $g : R^n \rightarrow R$; une contrainte inégalité $g(x) \leq a$ est dite active pour x^* si : $g(x^*) = a$, et inactive pour x^* si $g(x^*) < a$ où x^* est une solution optimale.

9.4.1-1 Implémentation de l'approche pour le problème de gestion

Nous avons implémenté l'approche Optimisation mono paramétrique (OMP) avec CADES⁵ pour le problème 9.11.

3. Il s'agit de trouver le bloc triangulaire inférieur (btf) : une sous matrice diagonale du tableau 9.10. Si l'intersection d'une colonne avec un ligne est un élément de diagonale et égale à 1, la variable correspondant à cette colonne est déterminée par la contrainte qui correspond à cette ligne.

4. Le choix de bonne valeur de pas pose une difficulté à utiliser cette approche (OMP)

5. parce que CADES contient l'outil approprié nommé *CADES Calculette*

	x_1	x_2	x_3	FO	contraintes actives
S^1	2	3	1	5	c1 et c4 sont actives
S^2	3	2	3	5	c1 actives
S^3	4	1	5	5	c1 et c3 sont actives

TABLE 9.2 – Un ensemble de solutions équivalentes

$\forall t \in [0..23]$

$$\begin{aligned}
 Min \quad & (x_1(t) \times p_g(t) - x_3(t) \times p_s(t)) \\
 & x_1(t) + x_2(t) - x_3(t) = P_{load}(t) - P_{pv}(t) \\
 & x_4(t) - x_4(t-1) - x_2(t) = 0 \\
 & 0 \leq x_1(t) \leq P_{abon} \\
 & 0 \leq x_3(t) \leq P_{pv}(t) \\
 & -Q_{max} \times a_2 \leq x_2(t) \\
 & x_2(t) \leq Q_{max} \times a_3 \\
 & Q_{max} \times a_1 \leq x_4(t) \\
 & x_4(t) \leq Q_{max}
 \end{aligned} \tag{9.11}$$

Comme la décomposition DM pour le problème 9.1, il faut qu'on transforme les contraintes en équations comme dans l'exemple précédent ; pour cela nous allons ajouter sept variables d'écart (e_i ; $i=1..7$). la formulation pour un instant (t) de 9.12 est donc :

$$\begin{aligned}
 c1 : \quad & x_1(t) + x_2(t) - x_3(t) = P_{load}(t) - P_{pv}(t) \\
 c2 : \quad & x_4(t) - x_4(t-1) - x_2(t) = 0 \\
 c3 : \quad & x_1(t) + \epsilon_1 - P_{abon} = 0 \\
 c4 : \quad & x_2(t) + \epsilon_2 - Q_{max} \times a_3 = 0 \\
 c5 : \quad & x_2(t) - \epsilon_3 + Q_{max} \times a_2 = 0 \\
 c6 : \quad & x_3(t) + \epsilon_4 - P_{pv}(t) = 0 \\
 c7 : \quad & x_3(t) - \epsilon_5 = 0 \\
 c8 : \quad & x_4(t) + \epsilon_6 - Q_{max} = 0 \\
 c9 : \quad & x_4(t) - \epsilon_7 - Q_{max} \times a_1 = 0
 \end{aligned} \tag{9.12}$$

La décomposition Dulmage-Mendelsohn pour un instant (t) donne une seule partie sous-déterminée (le tableau 9.13) :

	ϵ_4	ϵ_6	x_1	x_2	x_3	x_4	ϵ_1	ϵ_2	ϵ_3	ϵ_5	ϵ_7
c_1	0	0	1	1	1	0	0	0	0	0	0
c_2	0	0	0	1	0	1	0	0	0	0	0
c_6	1	0	0	0	1	0	0	0	0	0	0
c_8	0	1	0	0	0	1	0	0	0	0	0
c_3	0	0	1	0	0	0	1	0	0	0	0
c_4	0	0	0	1	0	0	0	1	0	0	0
c_5	0	0	0	1	0	0	0	0	1	0	0
c_7	0	0	0	0	1	0	0	0	0	1	0
c_9	0	0	0	0	0	1	0	0	0	0	1

A cause que la décomposition DM donne un seul bloc sous-déterminé, nous ne pouvons donc pas appliquer l'approche précédente pour déterminer avec quelle variable nous devons

commencer l'optimisation mono paramétrique OMP (Aucune variable d'optimisation, dans l'ensemble $\{x_1, x_2, x_3, x_4\}$ n'est en effet dans un bloc juste-déterminé et peut donc à ce titre être éliminé). Or, le choix de commencer l'OMP avec x_1 est fait car elle est moins contrainte que les autres variables (Selon le tableau 9.13, x_2 se présente dans 4 contraintes, alors que x_3 et x_4 sont présentes dans 3 contraintes). Normalement, il faudrait tester le voisinage de $x_1(t)$ pour tous les instants (t), mais pour simplifier la présentation, nous n'avons pris que l'instant $t = 10h$. Nous avons fait des essais pour $x_1(10) = 0$, $x_1(10) = 1.66$ et $x_1(10) = 3$. Le tableau 9.3 montre que, pour toutes ces valeurs de $x_1(10)$ dans le domaine de faisabilité ($0 \leq x_1(10) \leq P_{abon}$), la fonction objectif ne change pas. Ainsi tous ces points représentent un ensemble de solutions équivalentes (les valeurs négatives représentent un gain selon la formulation de la fonction objectif).

$x_1(10)$ [kW]	0	1.66	3
FO [€]	-9. 70372	-9. 70372	-9. 70372

TABLE 9.3 – Ensemble de solutions équivalentes pour l'instant $t = 10h$

9.4.1-2 Limite de l'approche

Pour les problèmes d'optimisation de faible complexité avec un nombre de variables inférieur ou égal à 10, cette approche est bien adaptée. Cependant, pour le problème de gestion des flux d'énergie où le nombre des variables est très important (car les variables définies pour une période donnée doivent être multipliées par le nombre de périodes à considérer, typiquement 24), l'approche s'avère vite gourmande en temps de calcul. De plus, pour tester un voisinage avec l'optimisation mono paramétrique, il faut trouver une bonne valeur de pas. Laisser le choix de cette valeur à l'utilisateur de l'approche OMP est délicat : soit cette valeur du pas amène un point en dehors du domaine faisable, soit provoque une erreur numérique pour le solveur (par exemple un pas de valeur 0.99999).

Par ailleurs, avec l'approche décrite précédemment, il n'est pas possible de détecter les cas de parallélisme entre certaines contraintes et la fonction objectif. Dans ce cas, les points figurant en limite⁶ de cette contrainte sont des solutions équivalentes.

Une approche permettant de surmonter les problèmes évoqués précédemment est présentée dans le paragraphe suivant.

9.4.2 Une approche par une optimisation multi-paramétrique (algorithme D_{max})

Considérons deux des trois acteurs intervenant sur le bâtiment : le *concepteur de type de modèles*, qui décrit un type de modèles pour la gestion, et le *concepteur de gestionnaire du système bâtiment*. Nous proposons dans ce paragraphe de prendre le modèle initial du *concepteur de type P_1* , puis de le reformuler afin de générer un autre problème d'optimisation P_2 dans lequel on recherche l'existence de multiples solutions équivalentes. On peut alors informer le *concepteur de gestionnaire* que le modèle utilisé avec les données numériques induit l'existence d'un effet W possible.

6. le cas où l'inégalité devient égalité

Pour le nouveau problème P_2 , on cherche la *distance* maximale entre la solution S^2 de cette nouvelle formulation et la solution optimale S^1 obtenue initialement avec P_1 tout en conservant l'ancienne fonction objectif à la valeur trouvée en S^1 .

Avec l'approche précédente (OMP), la *distance* entre le point optimal et son voisinage recherché devait être spécifiée (en fixant la valeur du pas), ce qui n'est plus le cas ici.

Pratiquement, pour le problème de gestion, on va rechercher la marge maximale de fonctionnement des sources qui donne le même gain économique en fin de journée. Les paragraphes suivantes présentent cet algorithme qu'on a appelé algorithme D_{max} .

Afin de détecter l'existence de multiples solutions pour les environnements de résolution linéaires et aussi non linéaires, nous proposons deux formulations de l'algorithme D_{max} .

9.4.2-1 Algorithme D_{max} en formulation linéaire mixte (*distance linéaire*)

Admettons que la solution optimale S^1 du problème initial P_1 (cf. problème 9.1) est $x_i^*, \forall i \in \{1, 2, 3, 4\}$. Nous cherchons à trouver la solution équivalente S^2 de P_2 qui donne la distance maximale entre S^1 et S^2 sans affecter la valeur de la fonction objectif (cf. figure 9.13). En conséquence, l'existence de cette distance $dx_i(t)$ indique l'existence de plusieurs solutions équivalentes. Puisque nous ne pouvons pas savoir si la valeur de $x_i(t)$ est supérieure ou inférieure de $x_i^*(t)$, nous proposons de calculer la distance comme la valeur absolue de : $x_i(t) - x_i^*(t)$.

Algorithme :

1. On définit $dx_i(t)$ dans P_2 qui est $x_i(t) - x_i^*(t)$.
 - (a) $x_i^*(t)$ est la valeur optimale trouvée avec P_1 .
 - (b) $x_i(t)$ est la nouvelle variable d'optimisation à trouver pour P_2 .
2. On applique les contraintes de P_1 sur les nouvelles variables $x_i(t)$ de P_2 .
3. On fige la variable qui représente la fonction objectif à la valeur trouvée en P_1 .

Pour chercher la taille maximale de la région des solutions équivalentes, nous calculons la *distance* maximale D_{max} telle que la formulation 9.14 le montre :

$$\text{Max}_{dx_i; i \in \{0, \dots, n\}} D_{max} \quad \text{avec } D_{max} = \sum_{t=0}^{23} \sum_{i=0}^n |dx_i(t)| \quad (9.14)$$

Si D_{max} est nul cela signifie qu'il n'existe pas d'ensemble de solutions équivalentes : il n'y a pas de risque d'effet W.

Pour qu'on puisse implémenter l'algorithme dans un environnement linéaire, il faut transformer $|dx_j(t)|$. Cela est fait grâce à la transformation détaillée dans l'annexe 18. Brièvement, la transformation aide à remplacer la valeur absolue d'une expression par des variables binaires comme indiqué dans l'encart 18.

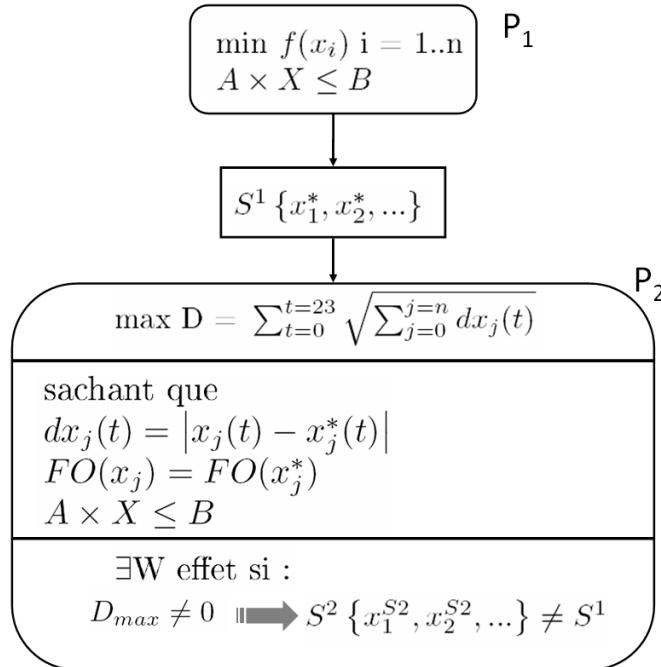


FIGURE 9.13 – Algorithme pour détecter un cas de solution multiple avec la marge associée

Soit l'expression à transformer :

$$E = \sum_{i=0}^n |dx_i| \quad (9.15)$$

avec $dx_i \in [-M_i, M_i]$

Comme illustré dans l'annexe 18, cette expression peut être remplacée par la formulation suivante :

$$\begin{aligned} E &= \sum_{i=0}^n (z_i - dx_i) & (9.16) \\ dx_i &\geq (\delta_i - 1)M_i \\ dx_i &< \delta_i M_i \\ z_i &\leq 2\delta_i M_i \\ z_i &\geq -2\delta_i M_i \\ z_i &\leq 2dx_i + 4M_i(1 - \delta_i) \\ z_i &\geq 2dx_i - 4M_i(1 - \delta_i) \end{aligned}$$

La formulation du problème 9.1 avec l'algorithme D_{max} comme un PLNE est la suivante :

$\forall t \in [0..23]$

$$\begin{aligned}
 & x_1(t) + x_2(t) - x_3(t) = P_{load}(t) - P_{pv}(t) \\
 & x_2(t) + x_4(t) - x_4(t-1) = 0 \\
 & dx_1(t) = x_1(t) - x_1^*(t) \\
 & dx_2(t) = x_2(t) - x_2^*(t) \\
 & dx_3(t) = x_3(t) - x_3^*(t) \\
 & dx_4(t) = x_4(t) - x_4^*(t) \\
 & 0 \leq x_1(t) \leq P_{abon} \\
 & 0 \leq x_3(t) \leq P_{pv}(t) \\
 & -Q_{max} \times a_2 \leq x_2(t) \leq Q_{max} \times a_3 \\
 & Q_{max} \times a_1 \leq x_4(t) \leq Q_{max} \\
 & \sum_{t=0}^{t=23} ((x_1(t) + dx_1(t)) \times p_g(t) - (x_3(t) + dx_3(t)) \times p_s(t)) = a_4 \\
 & \forall i = 1, 2, 3 \\
 & dx_i \geq (\delta_i - 1)M_i \\
 & dx_i < \delta_i M_i \\
 & z_i \leq 2\delta_i M_i \\
 & z_i \geq -2\delta_i M_i \\
 & z_i \leq 2dx_i + 4M_i(1 - \delta_i) \\
 & z_i \geq 2dx_i - 4M_i(1 - \delta_i)
 \end{aligned}$$

Sachant que a_4 est la valeur de la fonction objectif du problème initial P_1 défini en 9.1.
La fonction objectif de P_2 est :

$$Max_{dx_j; j \in \{0, \dots, 3\}} D_{max} \quad \text{avec } D_{max} = \sum_{t=0}^{23} \sum_{i=0}^n (z_i(t) - dx_i(t)) \quad (9.17)$$

9.4.2-2 Résultat d'implémentation d'algorithme D_{max} en formulation linéaire mixte

Nous avons utilisé cet algorithme pour le problème 9.1. Le résultat confirme l'existence de solutions multiples. Nous avons tracé les deux solutions équivalentes (S^1 et S^2) dans la figure 9.15.

La gestion équivalente S^2 trouvée grâce à P_2 donne une nouvelle stratégie de pilotage pour la batterie avec une nouvelle gestion d'achat d'énergie au fournisseur. Alors que la revente de surplus n'a pas été modifiée par rapport à S^1 . Nous avons tracé la marge d'énergie disponible entre les deux solutions ($x_2(t)$ en S^1 et $x_2(t)$ en S^2). Le résultat a montré qu'il y a environ 4 kW de différence à certains moments de la journée (entre 02h et 06h, voir figure 9.14)⁷. Cette marge peut être utilisée en perspective de ce travail pour améliorer la robustesse du système de gestion contre les aléas de fonctionnement des sources notamment l'achat au fournisseur d'énergie.

7. Les courbes ont été tracées sous forme de fleur. Les abscisses des figures précédentes correspondent aux diamètres de la 'fleur' et représentent le pas de temps en heure (de 0h jusqu'à 23h), alors que les ordonnées précédentes correspondent aux cercles intermédiaires ; chacun représente un niveau de marge d'énergie.

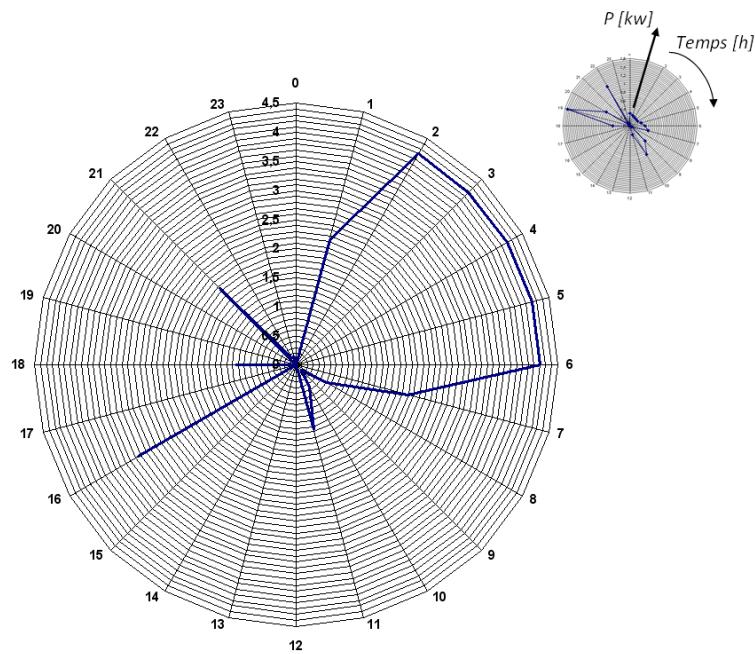
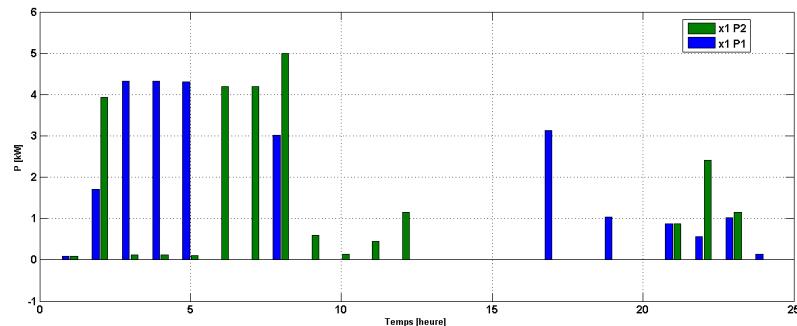
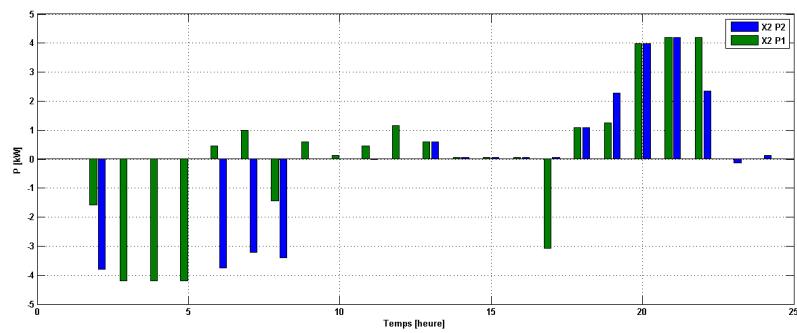


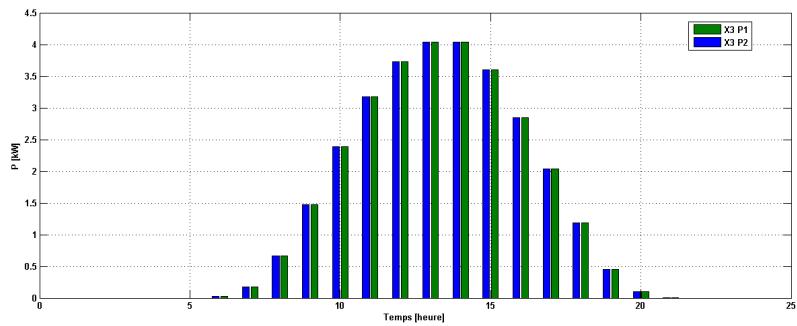
FIGURE 9.14 – Valeur absolue de la différence (d_{max}) trouvée entre la solution S^1 et celle S^2 pour la gestion de la batterie $x_2(t)$ sur 24h



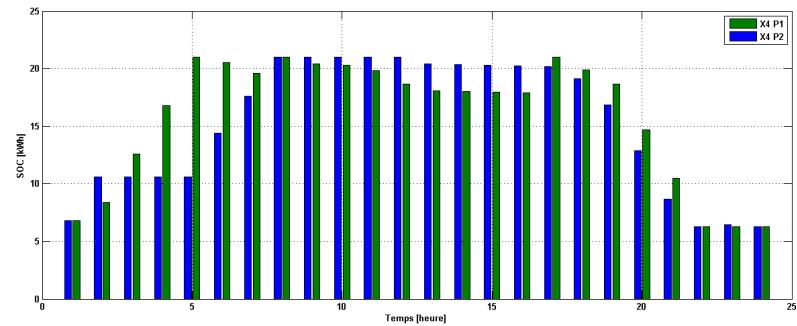
(a) Énergie du fournisseur



(b) Énergie de la batterie



(c) Surplus



(d) État de la charge de la batterie

FIGURE 9.15 – Comparaison entre deux gestions trouvées avec CPLEX en implémentant D_{max}

9.4.2-3 Algorithme D_{max} en formulation non linéaire (*distance quadratique*)

Pour les environnements non linéaires, on a remplacé la valeur absolue par la norme au deuxième ordre de $dx_i(t)$ (voir figure 9.16) telle que la formule suivante illustre :

$$dx_i(t) = (x_i(t) - x_i^*(t))^2 \quad (9.18)$$

Dans ce cas, le problème d'optimisation à résoudre pour trouver la *distance maximale* est : $\forall t \in [0..23]$

$$\begin{aligned} x_1(t) + x_2(t) - x_3(t) &= P_{load}(t) - P_{pv}(t) \\ x_2(t) + x_4(t) - x_4(t-1) &= 0 \\ dx_1(t) &= (x_1(t) - x_1^*(t))^2 \\ dx_2(t) &= (x_2(t) - x_2^*(t))^2 \\ dx_3(t) &= (x_3(t) - x_3^*(t))^2 \\ dx_4(t) &= (x_4(t) - x_4^*(t))^2 \\ 0 \leq x_1(t) &\leq P_{abon} \\ 0 \leq x_3(t) &\leq P_{pv}(t) \\ -Q_{max} \times a_2 &\leq x_2(t) \leq Q_{max} \times a_3 \\ Q_{max} \times a_1 &\leq x_4(t) \leq Q_{max} \\ \sum_{t=0}^{t=23} ((x_1^*(t) + dx_1(t)) \times p_g(t) - (x_8^*(t) + dx_8(t)) \times p_s(t)) &= a_4 \end{aligned} \quad (9.19)$$

La *distance* dans toutes les directions aura la forme suivante :

$$Max_{dx_j; j \in \{0, \dots, n\}} D_{max} \quad \text{avec } D_{max} = \sum_{t=0}^{23} \sqrt{\sum_{j=0}^n dx_j(t)} \quad (9.20)$$

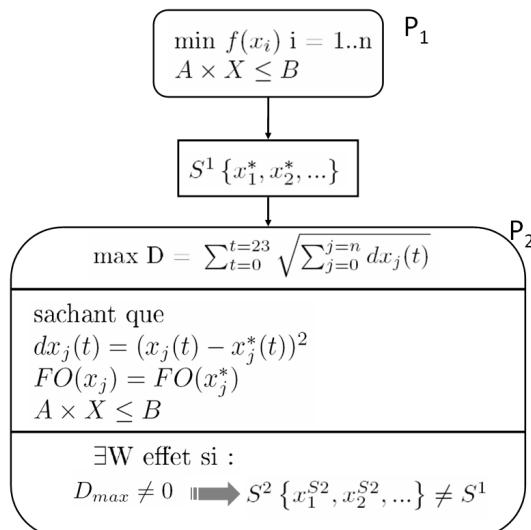
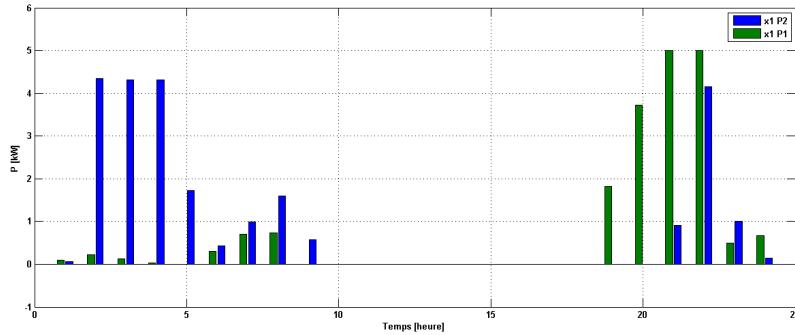


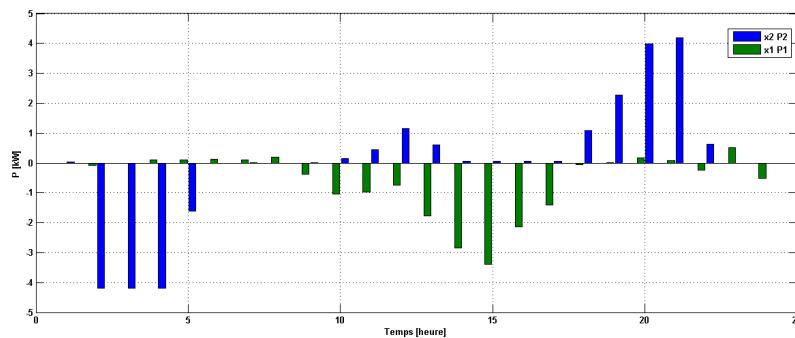
FIGURE 9.16 – Algorithme pour détecter un cas de multiples solutions avec la marge maximale associée en formulation non linéaire

9.4.2-4 Résultat de l’implémentation de l’algorithme D_{max} en formulation non linéaire

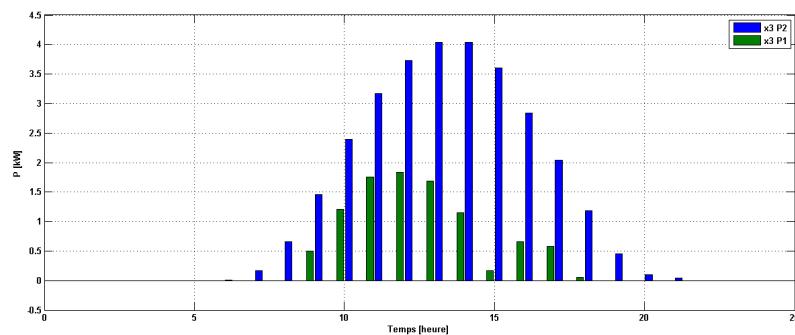
Nous avons utilisé l’environnement CADES pour implémenter cet algorithme. La figure 9.17 montre les deux solutions équivalentes : S^1 trouvée pour le problème P_1 et S^2 trouvée pour P_2 avec l’algorithme D_{max} . La figure 9.18(a) montre qu’à 19h, la marge d’énergie du réseau est de $\approx 1,58$ kW.



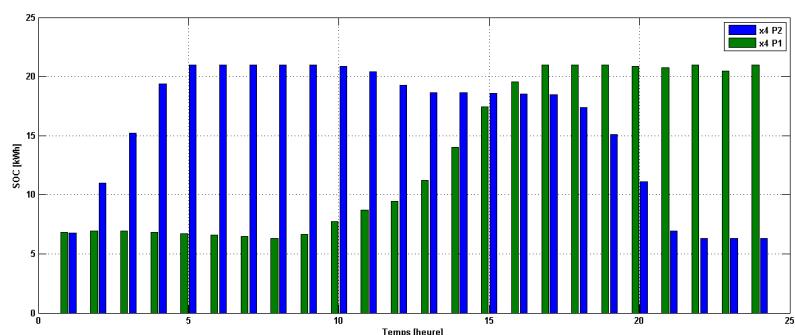
(a) Énergie du réseau



(b) Énergie de la batterie

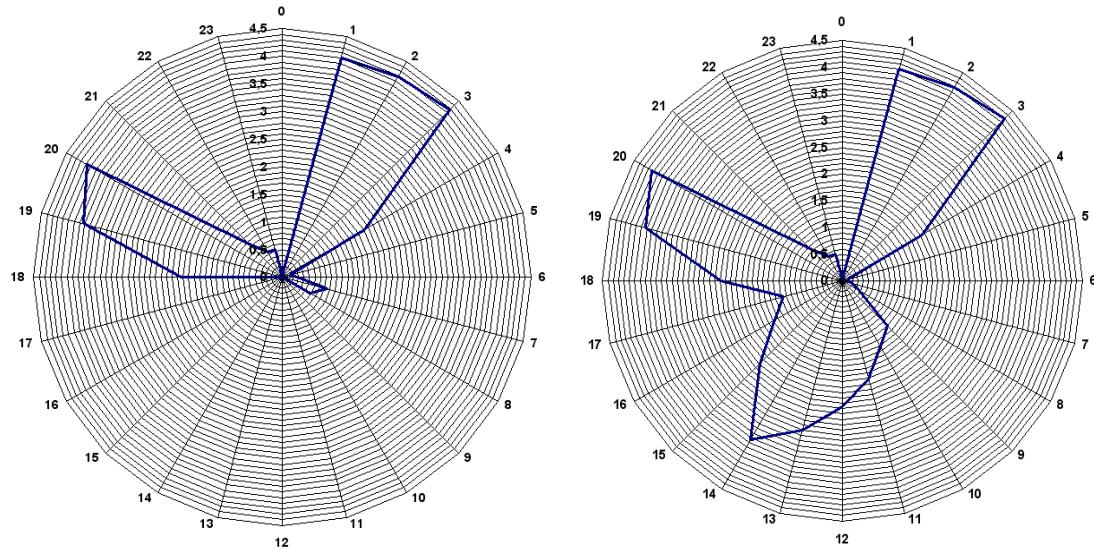


(c) Surplus



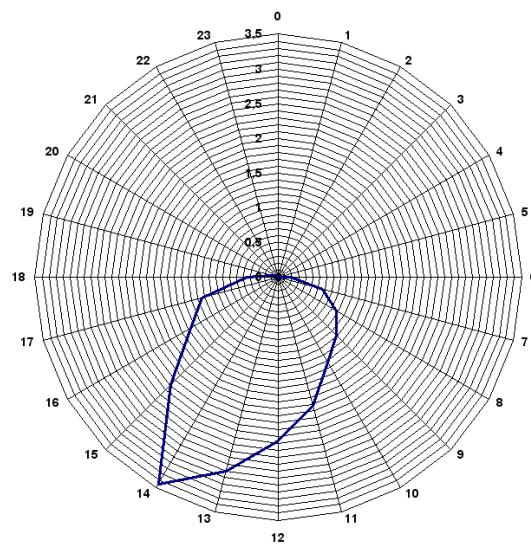
(d) État de la charge de la batterie

FIGURE 9.17 – Comparaison entre deux gestions trouvées avec CADES en implémentant D_{max}



(a) Marge d'énergie du réseau

(b) Marge d'énergie de la batterie



(c) Marge d'énergie sur le surplus

FIGURE 9.18 – Marge d'énergie entre deux solutions équivalentes trouvées par CADES sur 24h

9.4.2-5 Limites de l'algorithme D_{max} en formulation non linéaire

Initialiser l'algorithme de D_{max} avec une solution différente peut affecter sa convergence vers la solution la plus éloignée :

Il est possible que le solveur CADES ait convergé vers un optimum local S^1 , cela peut restreindre la possibilité de trouver toutes les solutions équivalentes. Par exemple, la solution S^2 la plus éloignée de S^1 trouvée avec l'algorithme CADES est différente que celle trouvée avec CPLEX (figure 9.19).

Nous avons initialisé D_{max} par deux solutions équivalentes, la première est trouvée par CPLEX, la deuxième par CADES, puis nous avons comparé la solution S^2 trouvé par CADES pour les deux cas. Le résultat de la comparaison (figure 9.19) montre que CADES n'a pas convergé vers la même solution. En conséquence, cette sensibilité à l'initialisation est une limite à corriger en perspective pour cet algorithme.

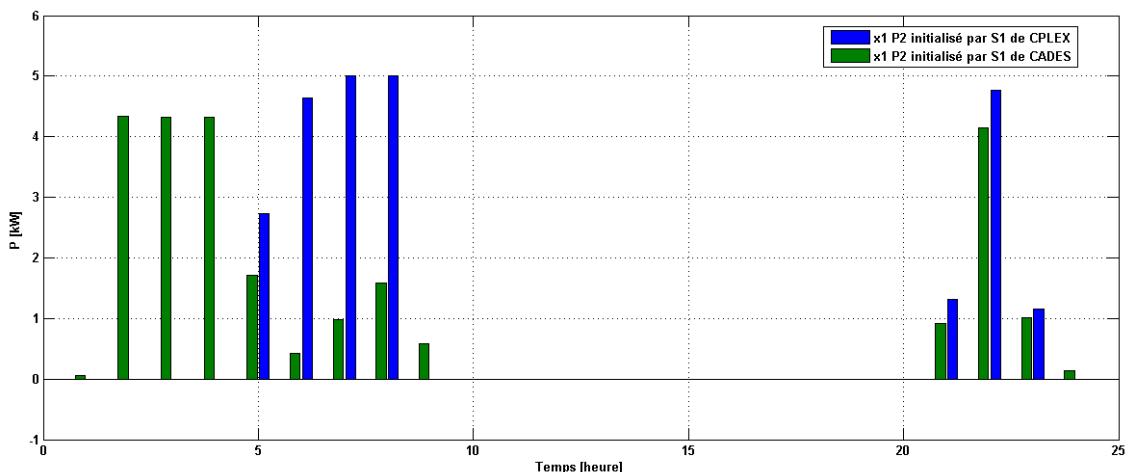


FIGURE 9.19 – L'Initialisation de D_{max} peut affecter la distance maximale trouvée entre deux solutions équivalentes

9.5 Perspective pour appréhender les problèmes d'optimisation à solutions multiples pour accroître la robustesse de la stratégie de gestion

Souvent, l'incertitude sur les données issues des prévisions induit des incertitudes sur le problème de gestion optimale, car le plan anticipatif trouvé par la gestion dépend de ces prévisions (courbe de charge ou prévisions météorologiques). Dans (LE et al. 2011), les chercheurs proposent une approche adaptée à plusieurs profils de consommation pour prendre en compte les incertitudes dans l'anticipation de problèmes de planification. Ainsi une distribution d'énergie plus robuste peut être déterminée.

En complément de l'algorithme D_{max} , des méthodes d'évaluation des risques et une approche robuste comme celle qui est décrite dans LE et al. (2011) peut être utilisée. Puisque nous ne pouvons pas, actuellement, déterminer la solution équivalente la plus robuste parmi toutes les solutions équivalentes, l'approche proposée dans (LE et al. 2011)

pourra amener une réponse pour choisir la solution à appliquer.

Le paragraphe suivant propose une méthode pour trouver certaines solutions équivalentes.

9.5.1 Proposition pour découvrir les solutions équivalentes

Nous présentons une manière de calculer des solutions équivalentes à partir de la marge obtenue avec l'algorithme D_{max} ([Warkozek, Ploix, Wurtz & Jacomino 2010b](#)). La formulation qui donne cet ensemble de solutions est la suivante [9.21](#) :

$$[X] = [X^*] + K \times \Delta X \quad (9.21)$$

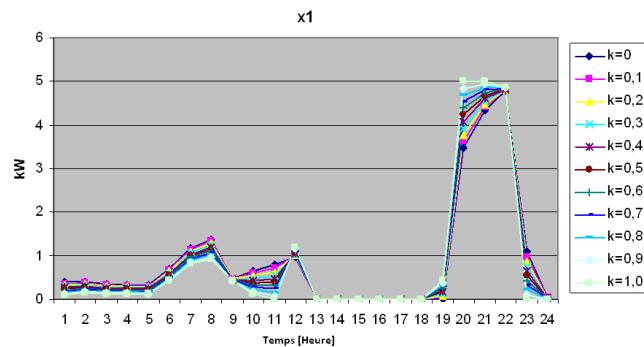
avec :

- X : une matrice de $(n \times m, 1)$ ⁸ solutions intermédiaires où n est le nombre de variables et m est la période d'étude.
- ΔX : une matrice de $(n \times m, 1)$ résulte de $([X^{D_{max}}] - [X^*])$.
- X^* : une matrice de $(n \times m, 1)$ de solution obtenue avec la résolution du problème initial P1.
- $X^{D_{max}}$: une matrice de $(n \times m, 1)$ de solution obtenue avec l'algorithme D_{max} .
- K : une matrice diagonale $(n \times m, n \times m)$ dans laquelle les éléments de diagonales ont une valeur k comprise entre $[0,1]$. En faisant varier la valeur de k , on obtient un ensemble de solutions équivalentes.

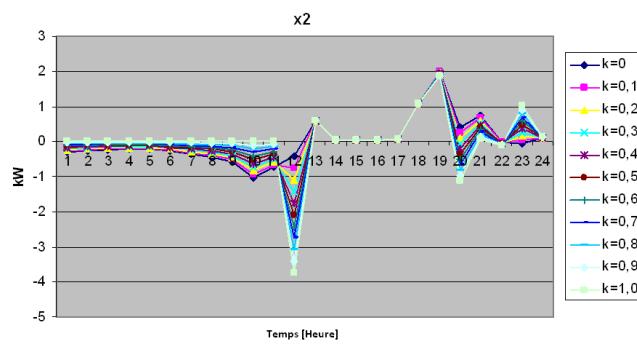
A chaque fois, on calcule la solution associée à chaque colonne de X , on valide qu'elle donne la même valeur de la fonction objectif et qu'elle ne viole pas les contraintes. Nous avons implémenté ce calcul pour la solution trouvée avec CADES, et nous avons obtenu un ensemble de solutions présenté dans la figure [9.20](#).

Remarque : Trouver des solutions intermédiaires par la discréétisation de la distance entre S^1 et S^2 n'est pas possible tout le temps. La région des solutions peut ne pas être continue ce qui peut impliquer des points à l'extérieur de la région réalisable.

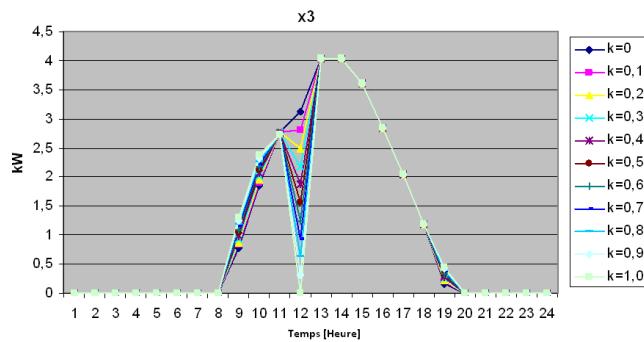
8. Dans notre exemple $m = 24$ heure



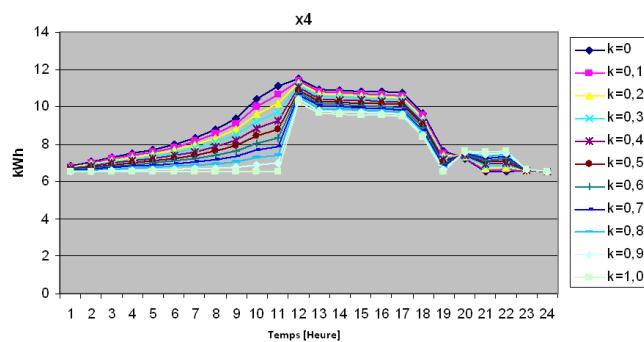
(a) Puissance retirée du réseau



(b) Puissance de la batterie



(c) Puissance du surplus



(d) Etat de la charge de la batterie

FIGURE 9.20 – Ensemble de solutions équivalentes

9.5.2 Proposition pour utiliser les solutions équivalentes

Par analogie avec les travaux de [Aubry et al. \(2009\)](#). La marge d'énergie trouvée précédemment pourrait être un moyen de faire face aux perturbations sur le plan anticipatif. Dans le domaine de la recherche opérationnelle, cette marge est appelée diamètre de stabilité ([Sotskov et al. 1998](#)) : "Pour un instant i avec la solution optimale S_i , le diamètre de stabilité $r(S_i)$ est le diamètre maximum d'une boule de centre (i) dans lequel le S_i reste toujours optimale".

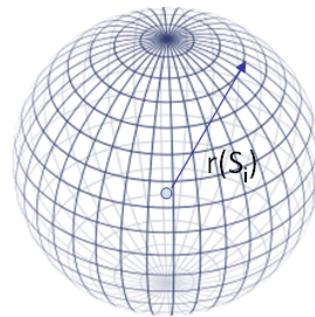


FIGURE 9.21 – Diamètre de stabilité de *Stotskov*

Comme le problème P_1 est linéaire, nous pouvons considérer que ce diamètre est la distance entre les deux solutions S^1 et S^2 précédentes (voir figure [9.22](#)).

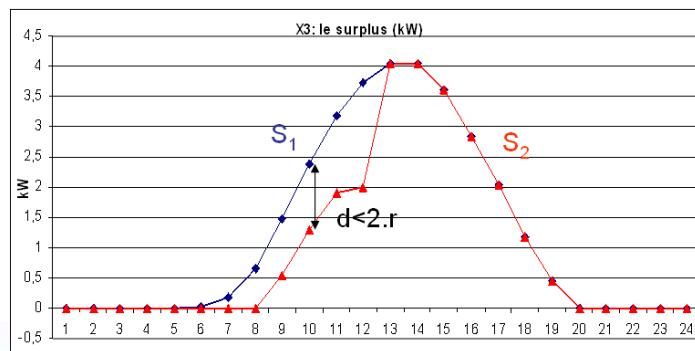


FIGURE 9.22 – Diamètre de stabilité de *Stotskov* pour le problème posé

S'il y aurait une variation à l'instant (t) de données par rapport la prévision (faite à l'instant $t-1$) cela induit un changement de plan de fonctionnement prévu, pour répondre à ce changement, nous proposons la démarche suivante :

1. chercher à nouveau avec D_{max} la marge d'énergie maximale,
2. calculer les solutions équivalentes,
3. utiliser la méthode d'évaluation de ([LE et al. 2011](#)) pour trouver la solution la plus robuste, puis appliquer cette solution.

9.6 Conclusion

La projection des problèmes de gestion des flux énergétiques dans les systèmes bâtiment vers différents environnements d'optimisation nous a permis de découvrir que le problème peut admettre un ensemble de solutions qui ont des valeurs différentes pour les variables d'optimisation tout en conservant une même valeur de la fonction objectif. Dans certains cas, il y aura un risque de basculement d'une solution vers une autre pouvant endommager les composants sensibles d'un système en imposant des stratégies de commande avec des oscillations.

Pour détecter la présence de solutions équivalentes, nous avons commencé par une approche structurelle. Il s'agit d'utiliser la décomposition Dulmage et Mendelsohn. Or, cette analyse ne peut pas suffire à détecter l'existence des solutions équivalents dans tous les cas. Cela est une limite qui nous a empêché de proposer cette méthode. Cette limite est illustrée par une analyse numérique basée sur le *pseudo-inverse de la matrice de coefficient*.

Ensuite, nous avons proposé une approche par optimisation mono-paramétrique (OMP). Avec cette approche, on détecte l'existence de solutions équivalentes lorsqu'elles concernent une variable de décision. Comme cela n'était pas assez général, nous avons proposé une approche par optimisation qui détecte à la fois l'existence des solutions équivalentes et permet de déterminer l'ensemble des solutions équivalentes. Cette approche s'appuie sur l'algorithme D_{max} que nous avons proposé. Contrairement à l'approche mono-paramétrique, avec l'algorithme D_{max} , on recherche la solution équivalente 'la plus éloignée' (si elle existe) par rapport à la solution initiale. La *distance* calculée permet de trouver des solutions intermédiaires qui peuvent être utilisées pour garantir une certaine robustesse en offrant plusieurs stratégies de gestion qui peuvent être modifiées pour tenir compte des aléas de fonctionnement. L'approche permet de distinguer les instants où il existe des degrés de liberté.

Nous avons contribué à établir les bases d'une approche permettant de garantir la robustesse du système de pilotage en présence d'aléas de fonctionnement en exploitant les solutions équivalentes lorsqu'elles existent. Ceci constitue donc une perspective intéressante au travail réalisé.

Chapitre 10

Conclusions partie III

Il y a plusieurs prototypes de bâtiments et différents acteurs, cela conduit à une grande variété de problèmes de dimensionnement et de gestion des sources. De surcroît, différents environnements d'optimisation peuvent être utilisés, ce qui n'est pas toujours facile à anticiper. En conséquence, la génération de problèmes de dimensionnement et de gestion de bâtiments pour un solveur précis présente un risque qui peut coûter cher lorsqu'il faut migrer vers un nouveau type de solveur. Le besoin de moyens automatiques pour générer les problèmes d'optimisation pour différents environnements d'optimisation est présenté dans cette partie.

Nous avons montré une approche concrétisée comme un outil métier pour générer automatiquement les problèmes de dimensionnement et de gestion des sources de bâtiments et les projeter vers différents solveurs. Nous avons défini trois types de métiers concernés : les métiers de développeur logiciel, modélisateur de type de modèles et de conception de systèmes bâtiment et de conception de gestionnaires du bâtiment. Chaque métier est représenté par un acteur jouant un rôle. Cette définition des rôles permet d'être plus efficace face aux difficultés de concevoir un problème de dimensionnement ou un problème de gestion car chacun de ces acteurs peut se spécialiser et intervenir dans des temporalités différentes.

Nous avons développé une plateforme dédiée au système bâtiment, composée d'un générateur de problèmes, de projecteurs et de générateurs d'éditeurs qui ont été adoptés pour résoudre la problématique de dimensionnement et de la gestion : c'est le *développeur logiciel* qui porte cette charge. Le *concepteur de types* peut alors créer des interfaces ou concepteurs graphiques pour chaque type de modèles. Le *concepteur de gestionnaires énergétiques* ou le *concepteur de systèmes bâtiment* peuvent alors utiliser ces interfaces pour paramétrier les types, puis récupérer automatiquement les modèles de niveau M0, pour le solveur qu'ils auront choisi.

Dans le chapitre 7, un problème de dimensionnement de sources dans lequel les aspects énergétiques et économiques sont couplés est présenté. Nous essayons avec la formulation de problèmes de trouver le dimensionnement optimal des sources électriques tout en optimisant la profitabilité économique du système global.

Dans le dernier chapitre, nous avons résolu un problème de gestion dans deux environnements de résolution différents et nous avons montré que les solutions obtenues ne coïncident pas nécessairement. Le problème d'optimisation peut admettre un ensemble de solutions qui ont des valeurs différentes pour les variables d'optimisation tout en conservant une même valeur de la fonction objectif. Dans certains cas, il y aura un risque de basculement d'une solution vers une autre pouvant endommager les composants sensibles d'un système en imposant des stratégies de commande avec des oscillations.

Pour la détection de ce phénomène nous avons proposé deux algorithmes pour prévenir le risque d'avoir un ensemble de solutions équivalentes.

Nous avons contribué à établir les bases d'une approche permettant d'appréhender ce phénomène afin de garantir la robustesse du système de pilotage en présence d'aléas de fonctionnement. Cette approche est basée sur l'exploitation des solutions équivalentes lorsqu'elles existent. Ceci constitue donc une perspective intéressante au travail réalisé.

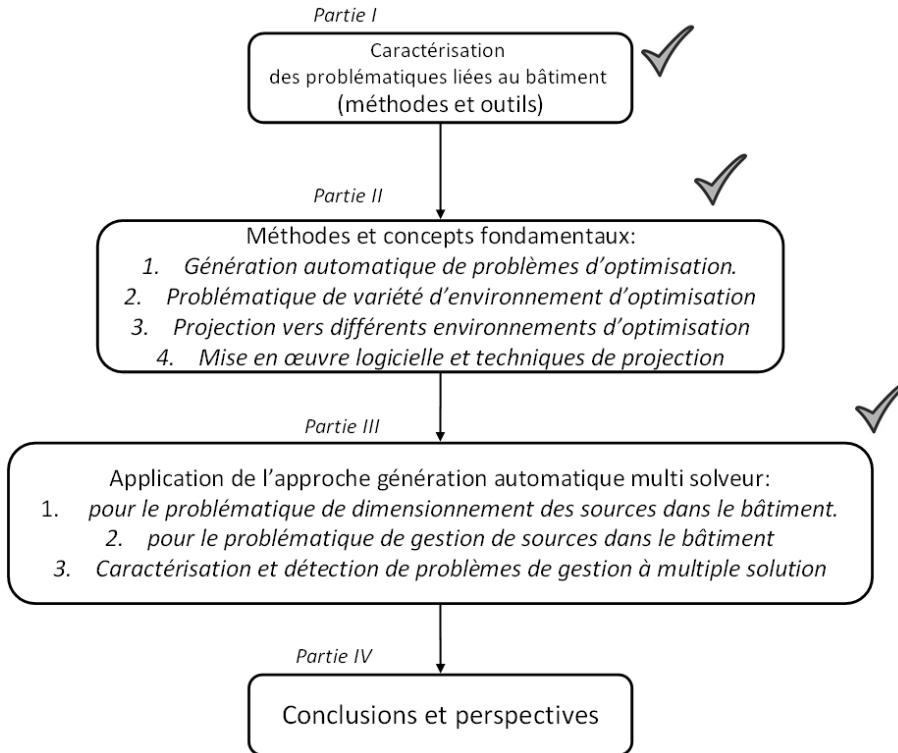


FIGURE 10.1 – Localisation dans le manuscrit de la thèse

Quatrième partie

Conclusions et Perspectives

Chapitre 11

Conclusions et Perspectives

Les travaux que nous avons présentés dans ce mémoire introduisent une solution pour la génération des problèmes de conception et de gestion de systèmes bâtiments. Cette solution est concrétisée comme un outil métier pour le dimensionnement et la gestion de sources électriques dans les bâtiments.

Pour synthétiser notre apport dans le domaine du bâtiment, nous allons le résumer dans les trois axes suivants :

Besoins actuels pour étudier les problèmes de conception et de gestion de sources :

Nous avons mis en évidence un ensemble des besoins pour la conception et la gestion de tel système électrique :

1. Vu que chaque système bâtiment peut avoir une configuration différente, le moyen de génération des problèmes de la conception et de la gestion doit être générique. Cela dans le sens que l'outil ne doit pas être limité par un certain nombre de modèles de composants de système bâtiment. L'intégration de nouveaux modèles de composants doit toujours être possible.
2. Vu le nombre important de variables et de contraintes des problèmes de conception et de gestion (environ 300 variables en prenant en compte une plage de temps de 24h) la génération de ces problèmes doit être :
 - (a) automatique, cela veut dire que la génération de problèmes de conception et de la gestion dans la syntaxe d'un environnement de résolution doit être faite par l'outil.
 - (b) multi solveur, car la capacité de résolution d'environnements est différente, ainsi le même problème peut être projeté puis résolu par différents environnements.

Ainsi pour répondre à ces besoins, nous avons proposé une approche de génération basée sur deux idées complémentaires qui sont :

1. la génération progressive de problèmes complets. Chaque composant de systèmes bâtiments ajoute ses contraintes locales individuellement sur le problème global. Puis le problème sera complété en ajoutant les contraintes globales comme le bilan énergétique et économique.
2. les problèmes (de conception et de gestion) doivent être écrits d'abord avec un formalisme neutre (indépendant) de celui d'environnements de résolution, puis une transformation soit faite sur la syntaxe selon l'environnement choisi par l'utilisateur final d'outil.

Aujourd'hui, nous ne générerons que de problèmes linéaires. Néanmoins, la génération de problèmes non linéaires est envisageable dans les perspectives. Cela demande de modi-

fications sur la structure de formalisme actuel (PIM OP-XML) et d'ajouter les nouvelles règles de projection dans le projecteurs CADES.

Développement d'outil métier :

Nous avons donné une proposition pour définir (du point de vu développement d'outil métier) les acteurs concernés par l'application bâtiment. Il s'agit de :

1. développeur logiciel.
2. concepteur de types de modèles.
3. concepteur de systèmes bâtiments et/ou concepteur de gestionnaire de systèmes bâtiments.

L'avantage de cette catégorisation est que chaque acteur intervient une fois puis il laisse le rôle à l'acteur suivant. Ainsi, le rôle de dernier acteur est réduit à utiliser que des interfaces graphiques pour composer les systèmes bâtiments puis analyser les résultats de conception ou de gestion.

La réalisation de l'outil a confirmé l'intérêt et la faisabilité de génération automatique et la transformation de modèles. Toutefois, les modèles de sources utilisés présentent des limites, comme par exemple le modèle de batterie pour lequel il manque un aspect du vieillissement. Or, cela peut être rattrapé dans les perspectives par un acteur de types de modèles qui modifiera le modèle existant voire ajouter un nouveau modèle et profiter de reste de l'outil pour la projection.

Caractérisation d'un cas de figure intéressant pour la gestion de système multi sources :

L'implémentation de cet outil nous a permis de découvrir que les problèmes de gestion peuvent avoir un ensemble de solutions équivalentes (mathématiquement) au lieu d'une solution unique. L'importance de cette découverte s'exprime physiquement. Certaines solutions peuvent être dangereuse sur les composants de stockage électrique dans le bâtiment car elles imposent de cycles rapides de charge/décharge de la batterie, ce qui réduit sa durée de vie.

Nous avons défini ce phénomène, puis nous avons proposé puis valider un algorithme pour le détecter avec deux formulations une linéaire et une autre non linéaire.

Dans la suite de cette réflexion, nous pouvons envisager de profiter de l'ensemble de solutions équivalentes pour améliorer la robustesse de système de gestion de sources. Cela peut être commencé par utiliser notre méthode pour trouver les solutions équivalentes.

Cinquième partie

Bibliographie

Bibliographie

- Abras, S. (2008), *Mutli-Agents Home Automation System for power management in buildings*, PhD thesis, Ecole Doctorale EEATS ' Electronique, Electrotechnique, Automatique et Traitement du Signal', INPGrenoble.
- ADEME (2007), Maîtrise de l'énergie et énergies renouvelables, chiffres clés 2007, Technical report, Agence de l'Environnement et de la Maîtrise de l'Énergie ADEME.
- Allain, L. (2003), *Capitalisation et traitement des modèles pour la conception en génie électrique*, PhD thesis, Ecole Doctorale EEATS 'Electronique, Electrotechnique, Automatique et Traitement du Signal', INPGrenoble.
- Angioletti, R. & Despretz, H. (2004), Maîtrise de l'énergie dans les bâtiments : Définitions. usages. consommations be 9 020, Technical report, Techniques de l'Ingénieur.
- Asratian, A., Deneley, T. & Haggkvist, R. (1998), *Bipartite graphs and their applications*, number ISBN 0-521-59345-X, Cambridge University Press.
- Atkinson1, C., Kuhne, T., Beydeda, S. & Gruhn, V. (2004), *Model-Driven Software Development, Volume II*, p119-136, Springer.
- Aubry, A., Rossi, A. & Jacomino, M. (2009), A generic off-line approche for dealing with uncertainty in production systems optimisation, 13th IFAC International Sympsum on Information Control problems in Manufacturing (IFAC-INCOM), Moscow Russia.
- Bakos, G., Soursos, M. & Tsagas, N. (2003), 'Technoeconomic assessment of a building-integrated pv system for electrical energy saving in residential sector', *Energy and Building, ELSEVIER* .
- Bartles, R. & Fiebig, D. (1996), 'Metering and modeling residential end-user electricity load curves', *Journal of Forecasting* **15**.
- Binder, H., Cronin, T., Lundsager, P., Manwell, J., Abdulwahid, U. & Baring-Gould, I. (2005), *Life Cycle Cost Analysis Handbook*, number ISBN 87-550-3441-1, Riso National Laboratory, Information Service Department.
- Blanc, X. (2005), *MDA en action, Ingénierie logicielle guidée par les modèles*, number ISBN 2-212-11539-3, EYROLLES.

- Boëda, D. (2009), *Etude de la contribution du pilotage de charges à la fourniture de services aux réseaux électriques*, PhD thesis, Ecole Doctorale EEATS 'Electronique, Electrotechnique, Automatique et Traitement du Signal', INPGrenoble.
- Boman, M., Davidsson, P., Skarmeas, N., Clark, K. & Gustavsson, R. (1998), Energy saving and added customer value in intelligent buildings, Vol. 3, Proceedings of the 3rd International Conference on the Practical Applications of Agents and Multi-Agent Systems PAAM-98, London, UK.
- BOMMÉ, E. (2009), *Modélisation et Optimisation des Machines Electroïdes à Double Entrefer*, PhD thesis, Ecole Doctorale EEATS 'Electronique, Electrotechnique, Automatique et Traitement du Signal', INP Grenoble.
- Bonnans, J. F., Gilbert, J. C., Lemaréchal, C. & Sagastizabal, C. (1997), *Optimisation Numérique*, number ISBN 3-540-63183-6, Springer.
- Brambley, M., D, H., P, H., DR, H., SC, M., KW, R. & P, T. (2005), *Advanced Sensors and Controls for Building Applications : Market assessment and Potential R&D Pathways*, U.S Departement of Energy.
- Bunus, P. & Fritzson, P. (2002), Methods for structural analysis and debugging of modelica models, 2nd International Modelica Conference, Germany.
- Bézivin, J. (2003), On the unification power of models, UML'2003 conference MDA : From Hype to Hope, and Reality, San Francisco.
- Bézivin, J. & Grebé, O. (2001), Towards a precise definition of the omg/mda framework, ASE 01, Automated Software Engineering, San Diego, USA.
- Caron, P.-A. (2007), *Ingénierie dirigée par les modèles pour la construction de dispositifs pédagogiques sur des plateformes de formation*, PhD thesis, L'UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE.
- Charles, G. & Prescod, P. (1998), *The XML Handbook*, number ISBN 0-13-081152-1, Prentice Hall PTR.
- Chenailler, H., Wurtz, F., Ploix, S., Joussellin, F. & Bontemps, A. (2010), Étude pour quantifier la part des apports internes dans bâtiment tertiaire bbc. application au bâtiment de predis, The International Building Performance Simulation Association, Paris.
- Chesn  , L., Duforestel, T., Roux, J.-J. & Rusaou  n, G. (2010), Exploitation des ressources de l'environnement par des b  timents, r  sultats et indicateurs, The International Building Performance Simulation Association, Paris.
- Chinneck, J. W. (2010), 'Practical optimization : A gentle introduction, systems and computer engineering carleton university, ottawa, ontario k1s 5b6 canada', <http://www.sce.carleton.ca/faculty/chinneck/po.html>.
- Cours (2003), Nonlinear circuits and devices, Technical report, University of Exeter, CDH Williams, CW061019/4.

- Czarnecki, K. & Helsen, S. (2003), Classification of model transformation approaches, OOPSLA 03 Workshop on Generative Techniques in the Context of Model-Driven Architecture, Anaheim, California, USA.
- Delinchant, B., Duret, D., Estrabaut, L., Gerbaud, L., Nguyen, H., Peloux, B. D., Rak-toarison, H., Verdiere, F., Bergenon, S. & Wurtz, F. (2007), ‘An optimizer using the software component paradigm for the optimization of engineering systems’, *COMPEL The International Journal for Computation and Mathematics in Electrical and Electronic Engineering* **26**.
- Despretz, H. (2009), Objectifs bâtiments 2012-2020, Technical report.
- Diestel, R. (2005), *Graph Theory*, number ISBN 978-3-642-14278-9, Springer.
- Do, T. P. (2010), *Simulation dynamique des actionneurs et capteurs électromagnétiques par réseaux de réluctances : modèles, méthodes et outils*, PhD thesis, Ecole Doctorale EEATS ‘Electronique, Electrotechnique, Automatique et Traitement du Signal’, INPGrenoble.
- Dulmage, A. & Mendelsohn, N. (1959), ‘A structure theory of bi-partite graphs of finite exterior extension’, *Transactions of the Royal Society of Canada* **53**.
- Dupeloux, B. (2006), *Modélisation des actionneurs électromagnétiques par réseaux de réluctances, création d’un outil métier dédié au prédimensionnement par optimisation*, PhD thesis, Ecole Doctorale EEATS ‘Electronique, Electrotechnique, Automatique et Traitement du Signal’, INPGrenoble.
- EDF (2010), ‘Les prix de l’électricité’, <http://bleuciel.edf.com/abonnement-et-contrat/les-prix/les-prix-de-l-electricite/tarif-bleu-47798.html>.
- Fabrizio, E. (2008), *Modelling of multi-energy systems in buildings*, PhD thesis, Politecnico di Torino et Institut National des Science Appliquées de Lyon.
- Fischer, V. (2004), *Composants logiciels pour le dimensionnement en génie électrique. Application à la résolution d’équations différentielles*, PhD thesis, Ecole Doctorale EEATS ‘Electronique, Electrotechnique, Automatique et Traitement du Signal’, INPGrenoble.
- Foundation, T. E. (2010), ‘Eclipse copyright © 2010’, <http://www.eclipse.org/>.
- Fourer, R., Gay, D. & Kernighan, B. (2003), *AMPL A Modeling Language For Mathematical Programming*, number ISBN 0-534-38809-4, Cut Hinrichs.
- Fourer, R. & Lopes, L. (2004), ‘A w3c schema for linear programming’, Department of Industrial Engineering and Management Sciences, Northwestern University.
- Ghometech (2009), ‘Un brevet avec un dépôt à l’app n° iddn.fr.001.310022.000.s.p.2009.000.30705, institut polytechnique de grenoble’.
- Guiavarch, A., Peuportier, B., Lénard, J.-D., Mudri, L., Mikolasek, R., Joanne, P., Lopez, J., Lokhat, I. & Gontier, P. (2010), Développement d’un progiciel d’aide à la conception intégrée des bâtiments, The International Building Performance Simulation Association, Paris.

- Ha, D. L., Ploix, S., Zamai, E. & Jacomino, M. (2006), Tabu search for the optimization of household energy consumption, International Conference on Information Reuse and Integration IEEE IRR : Heuristic Systems Engineering, Waikoloa, Hawaii, USA.
- Ha, D. L., Ploix, S., Zamai, E. & Jacomino, M. (2008), 'Realtimes dynamic optimization for demand-side load management', *International Journal of Management Science and Engineering Management* **3**.
- Harmon, P. (2010), 'Mda : An idea whose time has come, cutter consortium', http://www.dcc.ttu.ee/Automaatika/lap/LAP8714/MDA_Seminar_Harmon.pdf.
- Hawarah, L., Ploix, S. & Jacomino, M. (2010), User behavior prediction in energy consumption in housing using bayesian networks, in 'ICAISC (1)', pp. 372–379.
- Henze, G. & Dodier, R. (2003), 'Adaptive optimal control of a grid-independent photovoltaic system', *ASME Journal Of Solar Energy Engineering* **125**.
- IEA (2009), Key world energy statistique, international energy agency, Technical report, OECD/IEA.
- ILOG, I. (2010), 'Cplex mixed integer optimizer', <http://www.ilog.com/products/cplex/>.
- INC, O.-R. T. (2010), 'Rt-lab distributed real-time power', http://www.opal-rt.com/sites/default/files/rtlabs_brochure_en.pdf.
- Juquois, F. (2002), Guide de rédaction du cahier des charges techniques des générateurs photovoltaïques connectés au réseau, Technical report, Agence de l'Environnement et de la Maîtrise de l'Énergie ADEME. ISBN 2-86817-689-5.
- Kiehne, H. A. (2003), *Battery Technology Handbook*, number ISBN 0-8247-4249-4, MARCEL DEKKER, Inc.
- Lamotte, F. D., Berruet, P. & Philippe, J.-L. (2007), 'Using model transformation for the analysis of the architecture of a reconfigurable system', *International Journal on Manufacturing Technology and Management* **11**.
- Lamotte, F. D., Berruet, P., Rossi, A. & Philippe, J.-L. (2008), 'Control code generation using model engineering for an electric train', *Proceedings of the 17th World Congress The International Federation of Automatic Control Seoul, Korea* .
- Larousse (2010), 'Dictionnaire', <http://www.larousse.fr/dictionnaires/francais-monolingue>.
- LE, M. H., Jacomino, M. & Ploix, S. (2011), Prise en compte des incertitudes de prédiction dans la gestion des flux d'énergie dans l'habitat, 4èmes Journées Doctorales du Groupe de Recherche en Modélisation, Analyse et Conduite des Systèmes dynamiques JDMACS 2011, Marseille, France.
- Levermore, G. L. (2003), *Building Energy Management Systems, Applications to low-energy HVAC and natural ventilation control*, number ISBN 0-203-78558-4, E & FN SPON.

- Long, D. H. (2007), *Un système avancé de gestion d'énergie dans le bâtiment pour coordonner production et consommation*, PhD thesis, Ecole Doctorale EEATS ' Electronique, Electrotechnique, Automatique et Traitement du Signal', INPGrenoble.
- Mark, G. S. (2003), 'Numerical optimization courses, computer based learning unit, university of leeds', <http://www.math.mtu.edu/~msgocken/ma5630spring2003/lectures/sqp1/sqp1.pdf>.
- Mayer, C. (2001), *Matrix Analysis and Applied Linear Algebra Book and Solutions Manuel*, number ISBN 978-0-898-71454-8, SIAM : Society for Industrial and Applied Mathematics.
- MEDD (2010), Chiffres clés de l'énergie, Technical report, Ministère de l'Écologie, de l'Énergie du Développement durable et de l'Aménagement du territoire.
- Mellit, A. (2007), 'Sizing of photovoltaic systems : a review', *Revue des Energies Renouvelables* **10**.
- Merieux, P. & Pleynet, B. (1992), Chauffage et rafraîchissement : systèmes de conduite et de gestion be 2 158, Technical report, Techniques de l'Ingénieur.
- Miller, J. & Mukerji, J. (n.d.), 'Mda guide version 1.0.1', OMG Copyright © 2003.
- Missaoui, R., Warkozek, G., Abras, S., Ploix, S. & Bacha, S. (2010), Simulation temps réel pour la gestion des flux énergétiques dans l'habitat, The International Building Performance Simulation Association, Paris.
- Mondol, J. D., Yohanis, Y. & Norton, B. (2006), 'Optimal sizing of array and inverter for grid-connected photovoltaic systems', *Solar Energy, ELSEVIER* **80**.
- Moore, E. H. (1920), 'On the reciprocal of the general algebraic matrix', *Bulletin of the American Mathematical Society* **26**.
- Muselli, M., Notton, G., Poggi, P. & Louche, A. (2000), 'Pv-hybrid power systems sizing incorporating battery storage : an analysis via simulation calculations', *Renewable Energy, ELSEVIER* .
- Nafeh, A. E.-S. (2009), 'Design and economic analysis of a stand-alone pv system to electrify a remote area household in egypt', *The open Renewable energy journal* **2**.
- Nikolaou, T., Kolkotsa, D. & Stavrakakis, G. (2004), *HandBook for Intelligent Building*, <http://www.ibuilding.gr/handbook/index.html>.
- Nocedal, J. & Wright, S. (1999), *Numerical Optimization*, number ISBN 0-387-98793-2, Springer.
- Notton, G., ad I Colda, I. C. & Caluianu, S. (2010), Influence d'un ombrage partiel sur la production électrique d'un module photovoltaïque en silicium monocristallin, Vol. 13, Revue des Energies Renouvelables.

- Notton, G., Lazarov, V. & Stoyanov, L. (2009), ‘Optimal sizing of grid-connected pv system for various pv module technologies and inclinations, inverter efficieny characteristics and locations’, *Renewable Energy*, Elsevier .
- Okunski, B. (2008), The drivers of the levelized cost of electricity for utility-scale photovoltaics, Technical report, SUNPOWER CORPORATION.
- OMG (2010), ‘Object management group’, <http://www.omg.org>.
- OMG-MOF (2010), ‘Meta object facility’, <http://www.omg.org/mof>.
- Paris, B., Blanco, E. & Hazyuk, I. (2010), Programmation linéaire pour la gestion de l’énergie électrique d’un habitat, The International Building Performance Simulation Association, Paris.
- Penya, Y. (2003), Last generation applied artificial intelligence for energy management in building automation, Vol. 5, The 5th IFAC International Conference on Fieldbus Systems and their Applications, Aveiro, Portugal.
- Percebois, J. (1989), *Economie de l’énergie*, number ISBN 2-7178-1538-4, ECONOMICA.
- Pham, T. H., Wurtz, F. & Bacha, S. (2009), Optimal operation of a pv based multi-source system and energy management for household application, Proceedings of the 2009 IEEE International Conference on Industrial Technology, Australia.
- Philip, G. E., Murray, W. & Margret, W. H. (1981), *Practical Optimization*, number ISBN 0-12-283950-1, Academic Press.
- Pothen, A. & Fan, C.-J. (1990), ‘Computing the block triangular form of a sparce matrix’, *ACM Transactions on Mathematical Software* **16**.
- Powell, M. J. D. (1985), ‘On the quadtratic programming algorithm of goldfrab and idnani’, *Mathematical Programming Study* **25**.
- Presse (2010), ‘De nouveaux tarifs d’achat de l’électricité produuite a partir de la biomasse, du solaire et de la géothermie, cabinet du ministre d’etat’.
- Prunak, M. & Térouanne, E. (2010), ‘Rapport, 2e baromètre pricewaterhousecoopers sur l’état de la filière photovoltaïque en france’.
- Quenard, D. & Marcoux, Y. (2009), Bâtiments et occupants, qui consomme quoi, quand et comment ?, Technical report, TENERDIS, Journée Solaire-Bâtiments et Gestion des Réseaux Grenoble 01 Octobre.
- Sieglinde, F. K. & Stephen, P. R. (1995), *Life-Cycle Costing Manual for the federal Energy Management program*, US Departement of Commerce Technology Administration, National Institute of Standards and Technology.
- Singh, P. P. S. S. (2009), ‘Realistic generation cost of solar photovoltaic electricity’, *Renewable Energy*, Elsevier .

- Smiley, E. & Jones, J. (2000), Optimizing photovoltaic array size in a hybrid power system, Photovoltaic specialists conference, conference record of the 28th IEEE, September, Anchorage, AK, USA.
- Solar Energy Research and Consultancy* (2010), <http://www.solarbuzz.com/>.
- Soley, R. M. (2010), ‘Model driven architecture : Introduction’, www.omg.org/mda/mda_files/MDA-Seminar-Soley6.ppt.
- Sopian, K., zaharim, A., Ali, Y. & mohad Nopiah, Z. (2007), ‘Optimal operational strategy for hybrid renewable energy system using genetic algorithms’, *WSEAS TRANSACTIONS on MATHEMATICS* **7**.
- Sotskov, Y., Wagelmans, A. & Werner, F. (1998), ‘On the calculation of the stability radius of an optimal or an approximation schedule’, *Annals of Operations Research* **83**.
- TENERDIS (2010), ‘Journée collaboratives solaire l'événement de l'innovation solaire en rhône-alpes’.
- ULMER, J. S., BELAUD, J.-P. & LANN, J.-M. L. (2010), Towards a pivotal-based approach for business process alignment, 8th International Conference of Modeling and Simulation - MOSIM 10, Hammamet, Tunisia.
- Visier, J. C., Macary, M., Wurtz, E., Casamassima, M., Sanna, D., Pierroux, D., Cremona, C. & Bodin, F. (2009), ‘Vers des bâtiments à énergie positive : Proposition de structuration des actions de recherche’, *Programme de recherche et d'expérimentation sur l'énergie dans le bâtiment*.
- Warkozek, G. (2007), Anticiper l'optimisation de sources électriques par les réseaux de neurones, Master's thesis, Master EEATS INPJ/UJF Grenoble.
- Warkozek, G., Jacomino, M., Ploix, S. & Wurtz, F. (2009), Generic formulation of optimization problems for energy management : solving difficulties, practical and mathematical analysis, The 8th International Symposium on Electric and Magnetic Fields, Mondovie Italy.
- Warkozek, G., Ploix, S., Jacomino, M. & Wurtz, F. (2010), Sensitivity analysis by using optimization technique for sizing a pv grid connected home, First European Energy Conference e2c, Barcelone Spain.
- Warkozek, G., Ploix, S., Wurtz, F. & Jacomino, M. (2010a), ‘Problem formulation and analysis for optimal energy management in multi-sources systems : W effect’, *COMPEL The International Journal for Computation and Mathematics in Electrical and Electronic Engineering* **accept notification at 22 Mars 2011**.
- Warkozek, G., Ploix, S., Wurtz, F. & Jacomino, M. (2010b), Stability study for optimal energy management in multisources building, XI International Workshop on Optimization and Inverse Problems in Electromagnetism, Sophia Bulgaria.

Webster, D. & Sebastien, R. (2006), Modélisation d'une maison à énergie positive, projet de fin d'étude, Master's thesis, Institut National des science appliquées INSA Lyon, Département Génie énergétique et environnement.

Wimmer, M., Kappel, G., Kusel, A., Retschitzegger, W., Schoenboeck, J., Schwinger, W., Kolovos, D., Paige, R., Lauder, M., Schürr, A. & Wagelaar, D. (2011), A comparison of rule inheritance in model-to-model transformation languages, in 'Theory and Practice of Model Transformations, Fourth International Conference, ICMT 2011, Zurich, Switzerland, June 27-28. Proceedings, pages 31-46', Vol. 6707.

Yassine, A. (2008), *Génération des tests et placement de capteurs pour le diagnostic des systèmes physiques s'appuyant sur une modélisation structurelle*, PhD thesis, Ecole Doctorale EEATS 'Electronique, Electrotechnique, Automatique et Traitement du Signal', INPGrenoble.

Zirngibl, J. (2008), 'Cstb france, information paper on en 15316-4-6 photovoltaic systems, <http://www.iee-cense.eu/>'.

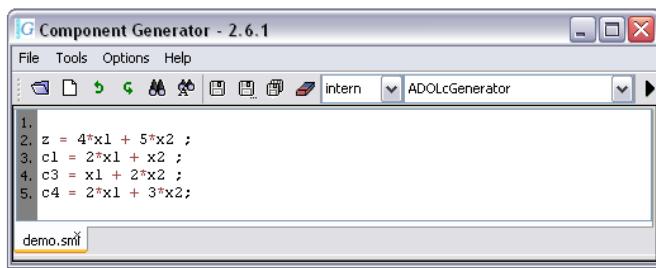
Sixième partie

Annexes

Chapitre 12

Environnements d'optimisation et algorithmes utilisés

12.1 Exemple d'un modèle en SML et ses spécifications en XML



```

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SPECIFICATIONS>
  <MODEL file="demo.icar" />
  <OPTIMIZER file="SQPVF13_V3.1.jar" />
  <CONFIGURATION>
    <MAX_ITERATION value="100" />
    <END_PRECISION value="1e-5" />
  </CONFIGURATION>
</OPTIMIZER>
<INPUT name="x2">
  <CONSTRAINT>
    <INIT value="0.0" />
    <MIN value="0.0" />
    <MAX value="3.0" />
  </CONSTRAINT>
</INPUT>
<INPUT name="x1">
  <CONSTRAINT>
    <INIT value="0.0" />
    <MIN value="0.0" />
    <MAX value="10.0" />
  </CONSTRAINT>
</INPUT>
<OUTPUT name="c4">
  <CONSTRAINT>
    <FIXED value="5.0" />
  </CONSTRAINT>
</OUTPUT>
<OUTPUT name="c3">
  <CONSTRAINT>
    <MIN value="0.0" />
    <MAX value="7.0" />
  </CONSTRAINT>
</OUTPUT>
<OUTPUT name="c1">
  <CONSTRAINT>
    <MIN value="0.0" />
    <MAX value="8.0" />
  </CONSTRAINT>
</OUTPUT>
<OUTPUT name="z">
  <CONSTRAINT>
    <MAXIMIZE />
    <MIN value="0.0" />
    <MAX value="10.0" />
  </CONSTRAINT>
</OUTPUT>
</SPECIFICATIONS>

```

(a) Le modèle analytique *demo.sml*

(b) Les spécifications des variables en *demo.xml*

FIGURE 12.1 – Le modèle analytique de problème 3.1 en sml et ses spécifications en xml

12.2 Algorithmes d'optimisation utilisés

12.2.1 L'algorithme du simplexe

La forme canonique d'un problème d'optimisation linéaire est :

$$\min_{x \in \mathbb{R}^n} f(x) \quad (12.1)$$

sachant que : $x \geq b$.

Dans ce problème, autant la fonction objectif que les contraintes sont linéaires. Il s'agit de chercher les points x qui satisfont les contraintes et qui minimisent f . Pour rechercher ces points, il faut distinguer les contraintes actives de celles qui sont inactives (une contrainte $a_i^T x \geq b_i$ est dite active si $a_i^T x = b_i$ et inactive si $a_i^T x > b_i$).

La propriété la plus importante des contraintes actives est qu'elles restreignent une perturbation solution dans le voisinage d'un point solution. Nous pouvons donc définir deux directions solutions par rapport à cette contrainte. Par exemple, pour un vecteur p vérifiant $a_i^T p = 0$, la direction p est appelée *Binding perturbation* par rapport à la contrainte i ; par contre, si p vérifie $a_i^T p > 0$, p est appelé *non binding perturbation* par rapport à la contrainte i . Ainsi les étapes de simplexe pour trouver la solution optimale se résume en cherchant le *Binding perturbation* et éviter la direction *non binding*.

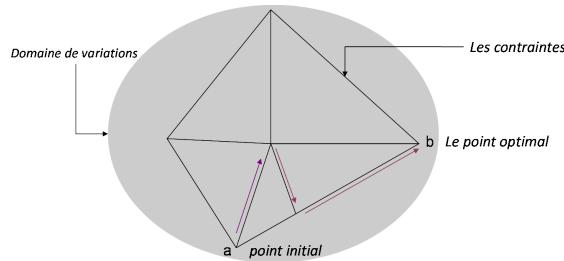


FIGURE 12.2 – Représentation schématique de la méthode du SIMPLEX

12.3 Les étapes d'algorithme du simplex

Commençons par noter k le nombre d'itérations typiques pour une méthode SIMPLEX ; pour chaque itération x_k il y a plusieurs informations associées avec la base actuelle ; plus particulièrement t_k représente le nombre de contraintes dans la base courante, puis I_k présente les indices des ces contraintes, \hat{A}_k les coefficients de ces contraintes pour x_k , \hat{b}_k le vecteur correspondant du second membre, finalement Z_k qui représente une base des vecteurs perpendiculaires avec \hat{A}_k .

Supposons que nous ayons un point faisable initial x_0 (pour trouver les techniques de calcul de ce point, nous orientons le lecteur vers (Philip et al. 1981)). A partir de ce point, nous trouvons t_0 , I_0 , \hat{A}_0 , \hat{b}_0 et Z_0 . Puis nous exécutons les étapes suivantes :

1. Validité de convergence : si les conditions de convergence sont satisfaites pour t_k , l'algorithme s'arrête ici et x_k est la solution optimale.
2. Choix de la logique de continuation : Est ce qu'il faut continuer dans la même base, ou la modifier pour supprimer une contrainte ? Si oui nous allons vers 7, sinon vers 3.
3. Calcul d'une direction de recherche faisable : calculer un vecteur non nul p_z de taille de $(n-t_k)$, puis la direction demandée sera :

$$p_k = Z_k p_z$$

4. Calcul du pas : calculer $\bar{\alpha}$ la largeur maximum non négative sur p_k , puis on détermine une largeur de pas positive α_k qui valide : $F(x_k + \alpha_k p_k) < F(x_k)$ et $\alpha_k \leq \bar{\alpha}$. Si $\alpha_k < \bar{\alpha}$ nous allons vers 6, sinon vers 5.
5. Ajouter une contrainte pour la base : selon α_k on ajoute une contrainte (son indice est r), puis on ajoute r à I_k et on modifie les autres informations associées.
6. Supprimer une contrainte de la base : choisir une contrainte (admettons que son indice est s) à supprimer, modifier les informations associées, puis retourner vers 1.
7. Mise à jour de l'estimation de solution : remplacer. $x_{k+1} \leftarrow x_k + \alpha_k p_k$, $k \leftarrow k+1$ et puis retourner vers 1.

12.4 L'algorithme de programmation séquentielle quadratique (SQP)

C'est une technique de résolution numérique de problèmes non linéaire NL, fondée sur la méthode de Newton. Historiquement elle est née avec la thèse de *Wilson 1963*, mais les travaux de *Han 1977* l'ont démocratisée ([Bonnans et al. 1997](#)).

Le principe de base de cette méthode est la linéarisation locale du problème d'optimisation afin d'obtenir un système linéaire facile à résoudre. Ainsi, SQP transforme le problème d'optimisation non linéaire en une suite de problèmes quadratiques linéarisés plus simples à résoudre avec une convergence locale rapide. Il s'agit alors d'utiliser des algorithmes linéaires comme les méthodes des points intérieurs ou du SIMPLEX.

12.5 Le principe de fonctionnement de l'algorithme SQP

Pour raison de simplification nous prenons le cas de contraintes d'égalité : Soit x^* le point de minimum local de :

$$\begin{aligned} \min \quad & f(x) \\ \text{et } \quad & g(x) = 0 \end{aligned}$$

Généralement le multiplicateur de Lagrange est intégré dans la formulation du problème pour définir les sous problèmes quadratiques à résoudre. Supposons que λ^* est le multiplicateur de Lagrange correspondant à x^* , dans ce cas nous pouvons écrire :

$$\begin{aligned} \nabla f(x^*) - \nabla g(x^*)\lambda^* &= 0 \\ -g(x^*) &= 0 \end{aligned}$$

La matrice de Jacobian est donc :

$$\begin{bmatrix} \nabla^2 \ell(x^*; \lambda^*) & -\nabla g(x^*) \\ -\nabla g(x^*)^T & 0 \end{bmatrix}$$

Cette matrice est non singulière, nous pourrons donc calculer x^* et λ^* par la méthode de Newton en résolvant le système :

$$\nabla f(x) - \nabla g(x)\lambda = 0$$

$$-g(x^*) = 0$$

En admettant qu'on ait une estimation $(x^{(k)}, \lambda^{(k)})$ de (x^*, λ^*) , la méthode Newton définit un déplacement $(x^{(k+1)}, \lambda^{(k+1)}) = (x^{(k)}, \lambda^{(k)}) + (p^{(k)}, w^{(k)})$; le pas $(p^{(k)}, w^{(k)})$ est défini par le système linéaire suivant :

$$\begin{bmatrix} \nabla^2 \ell(x^{(k)}; \lambda^{(k)}) & -\nabla g(x^{(k)}) \\ -\nabla g(x^{(k)})^T & 0 \end{bmatrix} \begin{bmatrix} p^{(k)} \\ w^{(k)} \end{bmatrix} = - \begin{bmatrix} \nabla \ell(x^{(k)}; \lambda^{(k)}) \\ -g(x^{(k)}) \end{bmatrix} \quad (12.2)$$

De plus, en supposant que $(x^{(k)}, \lambda^{(k)})$ soit suffisamment proche de (x^*, λ^*) pour que $\nabla^2 \ell(x^{(k)}; \lambda^{(k)})$ soit positif défini, le programme quadratique 12.3 aura une unique paire de solution de multiplicateur de Lagrange $(p^{(k)}, w^{(k)})$ (Mark 2003) (Nocedal & Wright 1999).

$$\min \frac{1}{2} p \nabla^2 \ell(x^{(k)}; \lambda^{(k)}) p + \nabla \ell(x^{(k)}; \lambda^{(k)}) + \ell(x^{(k)}; \lambda^{(k)}) \quad (12.3)$$

à condition que :

$$\nabla g(x^{(k)})^T p + g(x^{(k)}) = 0$$

En fait cette paire est déterminée par la condition nécessaire d'optimalité de premier ordre :

$$\nabla^2 \ell(x^{(k)}; \lambda^{(k)}) + \ell(x^{(k)}; \lambda^{(k)}) - \nabla g(x^{(k)}) w^{(k)} = 0$$

$$\nabla g(x^{(k)})^T p^{(k)} + g(x^{(k)}) = 0$$

La SQP va donc résoudre un programme quadratique sous la forme de 12.3 à chaque itération jusqu'à la convergence.

Chapitre 13

Métamodèles et générateurs de codes M0

13.1 Les métamodèles d'environnements d'optimisation (PM OP-*) et les générateurs de codes M0

13.1.1 Le métamodèle pour CADES (PM OP-CADES)

Normalement les utilisateurs de CADES écrivent leur modèle en sml (via le module *CADES Generator*), puis ils définissent manuellement les spécifications dans le module *CADES Optimizer*, puis ils les sauvegardent en XML (ce document XML est différent que le PIM du problème), enfin ils lancent l'optimisation. Ceci n'est pas pratique pour le problème du bâtiment, car nous avons un modèle qui contient des centaines de variables ou d'équations. Pour chaque cas d'étude il faut récrire les équations et en plus remettre les spécifications pour autant des variables. L'approche qu'on propose consiste à générer dès le début les variables en sml et leurs spécifications en xml au même temps, parce que toutes ces informations on les connaît grâce au PIM OP-XML et son instance OP-XML. En conséquence, il ne reste pas grande chose à modifier dans le module *CADES Optimizer*. La figure 13.1 montre l'approche classique pour utiliser CADES et notre proposition.

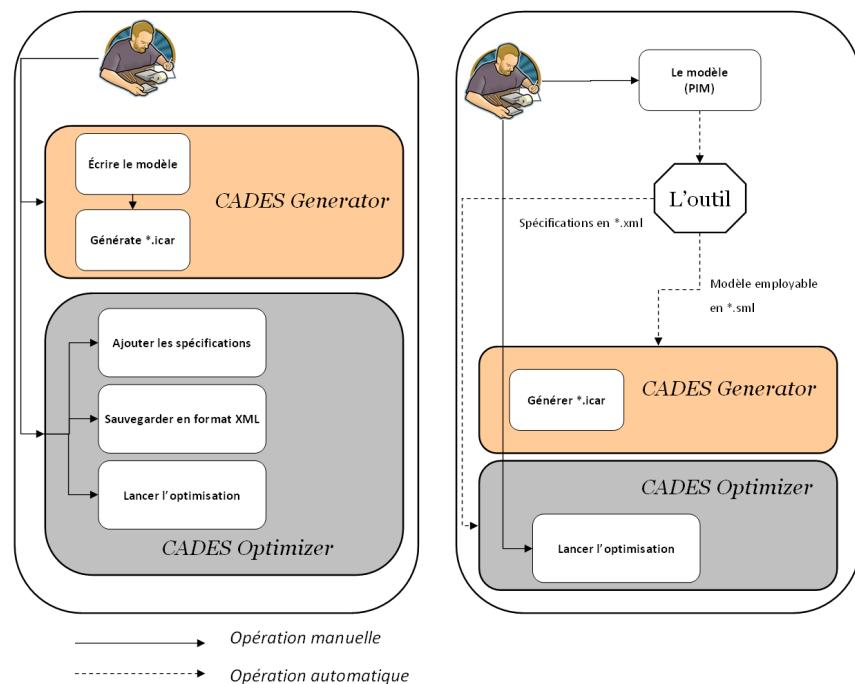


FIGURE 13.1 – Approche classique pour utiliser CADES et l'amélioration menée grâce à la transformation de modèles

Un méta modèle des variables, des contraintes en CADES est construit. Il s'agit des classes java où on décrit les règles de définitions des variables, des contraintes et de la fonction objectif (voir figure 13.2). La classe *CADESmodel* est le noyau central où on assemble tous les éléments du problème. Les projecteurs (dont on va parler dans la suite) vont instancier les modèles dépendants de plate-forme (PSM) à partir du ce méta modèle et le PIM OP-XML du problème présenté précédemment.

Nous pouvons considérer le modèle analytique comme le niveau M0 de l'architecture de l'IDM à quatre niveau (voir figure 4.2).

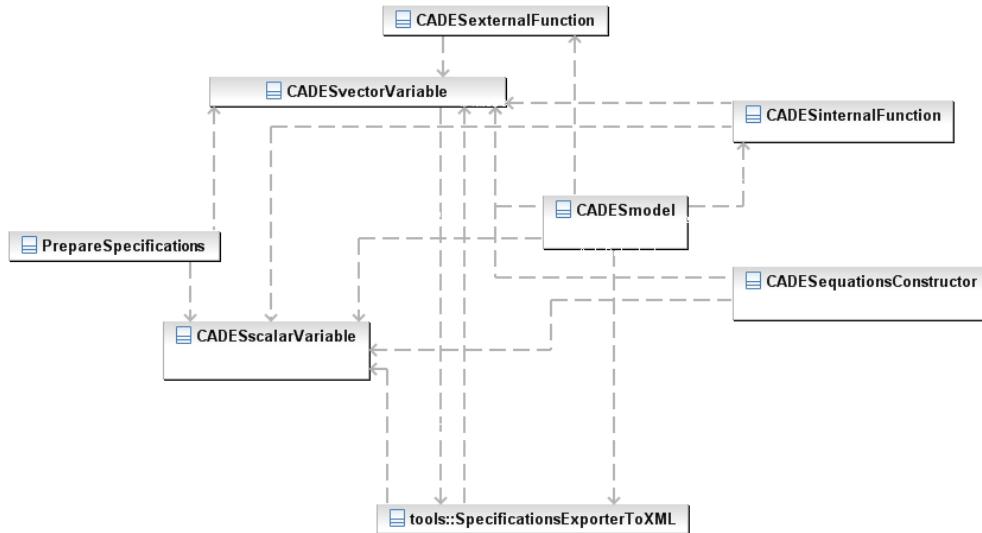


FIGURE 13.2 – Méta modèle de PSM d'un problème de type LP en CADES (PM OP-CADES)

13.1.2 Le méta modèle pour CPLEX (PM OP-CPLEX)

Le méta modèle de CPLEX est un package Java qui décrit les règles pour générer les variables, les contraintes en respectant ILOCT dans un programme utilisateur. Nous nous sommes appuyé sur la librairie du CPLEX pour définir ce méta modèle : chaque élément du problème hérite de certaines caractéristiques de son homologue dans la ILOCT. La figure 13.3 présente le diagramme des classes pour le méta modèle en CPLEX.

Le noyau central est le *CplexModel*. Le projecteur CPLEX va instancier les modèles dépendant de plate-forme (PSM) du problème à partir de ce méta modèle et le PIM de problème.

L'appel directe de solveur est possible, ceci est fait par *CplexSolver*. Grâce à la capacité interactive de CPLEX, nous pouvons aussi recevoir la solution de problème d'optimisation. Pour cela, nous avons généré un PSM de la solution obtenue par CPLEX (*CplexSolutionManager*), si la récupération de solution que nous avons implémenté n'est pas 'ergonomique' l'utilisateur pourrait fonder son propre moyen à partir du même PSM de la solution.

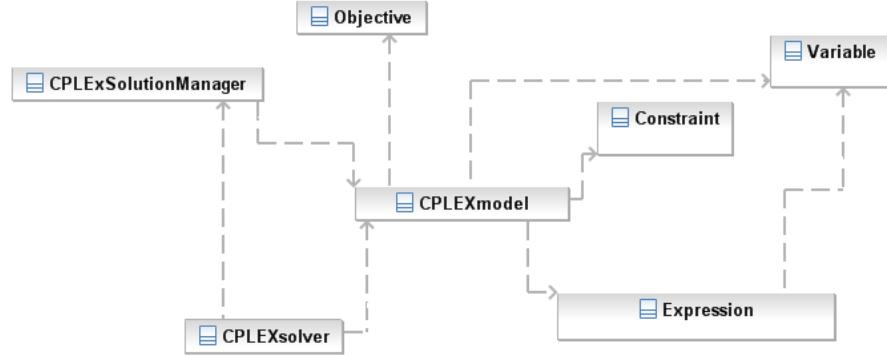


FIGURE 13.3 – Méta modèle de PSM du LP en CPLEX (PM OP-CPLEX)

13.1.3 Instancier les méta modèles PM OP-CADES et PM OP-CPLEX

Nous avons choisi parmi les trois règles de la transformation le type par programmation. Autrement dit, nous avons implémenter une classe qui pour chaque élément de PIM OP-XML va l'analyser afin de connaître son homologue dans PM de l'environnement cible, puis elle utilise les données du *concepteur de bâtiment/ de gestionnaire de bâtiment* pour générer cet homologue. Le PIM OP-XML sera représentée donc soit par *CADESmodel*, soit par *CPLEXmodel*.

Cette classe, passe par les éléments en trois étapes :

1. Construire l'arbre des contraintes : en fait c'est un patron vide dans lequel nous allons mettre les relations entre les variables. Le générateur commence par enregistrer les contraintes du PLNE, puis il les ordonne selon leurs types (d'égalité ou inégalité), ensuite il enregistre leur second membre, pour ceux qui sont d'inégalité il enregistre le type (supérieur ou inférieur), enfin selon le choix de plate-forme un objet correspondant sera généré.
2. Construire l'arbre des variables : Le générateur enregistre les variables puis il commence l'analyse. Cela veut dire qu'il cherche d'abord son type (entier ou continu) et sa taille (scalaire ou vectoriel), puis il génère l'objet correspondant selon la plate-forme. Enfin, il cherche son implantation, il s'agit d'un coefficient numérique (par exemple le tag de x_2 dans la contrainte c_1 problème 3.1 est +2)
3. Les variables vont être placées dans les branches de l'arbre 1 selon leurs *tags*.

La figure 13.4 montre l'algorithme de transformation adopté.

La dernière étape de la transformation est de générer les modèles utilisables en CADES et en CPLEX à partir de *CADESmodel* et *CPLEXmodel*. Cela fait l'objet du paragraphe suivant.

13.1.4 Générateurs de codes M0

13.1.4-1 Générateur de codes en CADES (OP-CADES)

Vu que le sml a un format textuel, alors les modèles employables vont être écrits par un générateur textuel (il va prendre le problème linéaire représenté par l'objet *CADESmodel*

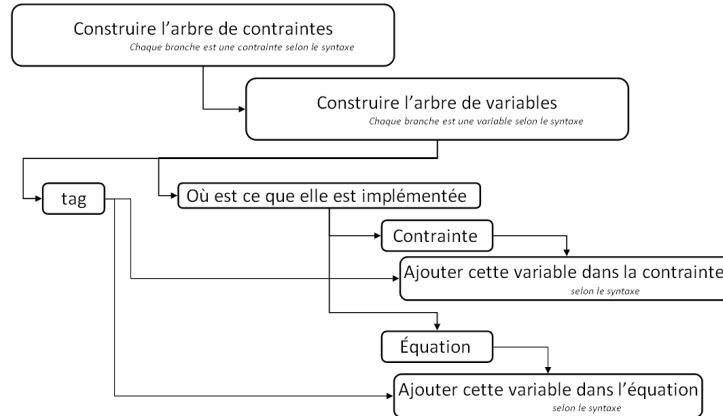


FIGURE 13.4 – Étapes de transformation de modèles par programmation

et l'écrire dans un fichier `*.sml`).

Selon le fonctionnement de CADES, nous ne pouvons pas écrire directement dans le `*.sml` les contraintes avec les signes (\geq et \leq) ; il y a que des équations dans le `*.sml` (comme nous l'avons montré dans la figure 12.1(a)). Nous transformerons donc toutes les contraintes d'inégalité en équations avant de générer le modèle utilisables. Pour cela nous appliquons la règle suivante :

Supposons que nous avons une contrainte de type $x_1 + 2 \times x_2 \geq 3$, pour intégrer cette contrainte dans le sml, nous définissons une nouvelle variable (variable artificielle) $C = x_1 + 2 \times x_2 - 3$ et on l'ajoute sur la spécification : $C \geq 0$ (la même analogie pour les contraintes de type \leq).

La figure 13.5 présente les grandes lignes pour générer les modèles employables en CADES.

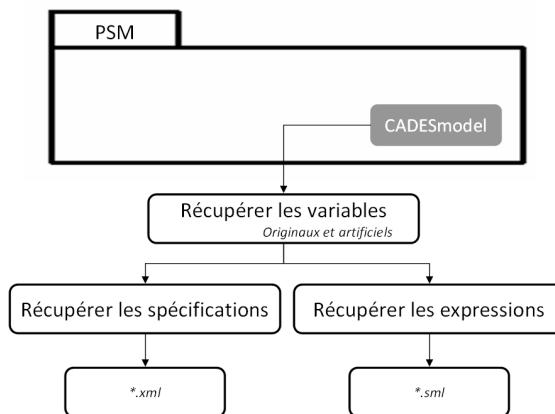


FIGURE 13.5 – Grandes lignes pour générer le modèle employable du LP pour CADES

En effet, le désavantage de cette règle est qu'elle augmente la taille du problème d'optimisation (en ajoutant les variables artificielles).

13.1.4-2 Générateur de codes en CPLEX (OP-CPLEX)

L'essentiel avec CPLEX est de construire l'objet IloCplex, parce qu'une fois que nous aurons l'objet IloCplex, la génération de *.lp s'est fait automatiquement à l'aide de ILOCT et sa bibliothèque (comme montré dans la figure 3.2).

L'algorithme pour construire l'objet IloCplex est illustré dans la figure 13.6.

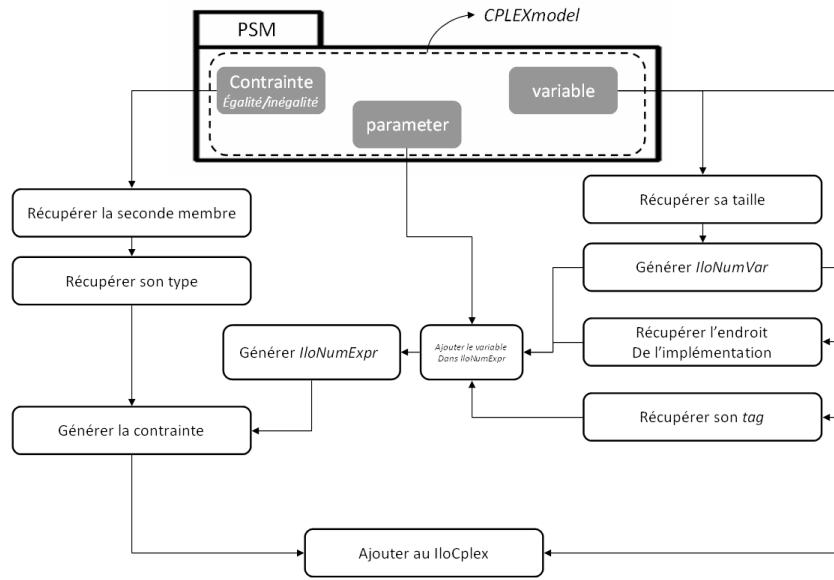


FIGURE 13.6 – Grandes lignes pour générer l'objet IloCplex du LP

Comme nous l'avons montré par la figure 3.2, le *IloNumExpr* est un objet ILOCT dans lequel nous pouvons exprimer une opération mathématique entre plusieurs variables (que l'on appelle *expression*). Nous avons profité de cet objet pour construire l'expression de la contrainte ; cela se fait en trois étapes :

1. dans le PSM il y a la déclaration d'une contrainte par exemple C. Nous définissons dans le IloCplex qu'il y a une expression sous le nom de C.
2. nous ajoutons dans cette expression les variables correspondantes venant de la branche droite de la figure 13.6.
3. selon le type de la contrainte récupérée du PSM nous déclarons que C est \geq ou \leq du second membre.

Dans le *.lp nous aurons la déclaration de C comme :
 C : *expression* \leq ou \geq second membre.

Chapitre 14

Ajouter les modèles de panneaux photovoltaïques et de batteries au PIM OP-XML

14.1 Ajouter le PIM OP-XML pour les sources électriques (partie pour le *concepteur de type de modèles*)

14.1.1 PIM OP-XML de panneau photovoltaïque

La différence avec le modèle PV entre le dimensionnement et la gestion est que le nombre de panneaux est connu dans le cas de gestion. En conséquence, la production solaire ne sera plus une variable d'optimisation. Par contre, nous optimisons la production solaire qui sera consommée localement dans le bâtiment. Le modèle analytique de PV est un ensemble de contraintes qui sont présentées dans la formulation suivante :

$$\begin{aligned} CPV_1 : \quad & x_3(t) = a_3 \times x_7(t) \\ CPV_2 : \quad & x_4 = \sum_{t=0}^{t=studyPeriod} x_5(t) \\ CPV_3 : \quad & x_5(t) = a_7(t) - x_7(t) \\ CPV_4 : \quad & x_7(t) \leq (1 - x_6(t)) \times a_7(t) \\ CPV_5 : \quad & a_7(t) = a_4 \times a_5(t) \times a_6 \\ CPV_6 : \quad & x_5(t) \leq a_7(t) \end{aligned} \tag{14.1}$$

Sachant que :

- a_3 : est le prix du surplus solaire revendu au fournisseur en euro/kWh.
- a_4 : est le rendement de l'onduleur.
- a_5 : est la production d'un seul module PV en kWh. En fait ce paramètre doit être calculé selon le type de panneaux solaires implémenté dans le système¹
- a_6 : est le nombre de modules PV. C'est un paramètre donné par l'utilisateur, alors que pour le problème de dimensionnement c'était une variable d'optimisation.
- a_7 : est la production totale des PV.
- x_3 : est la recette de la vente du surplus.
- x_4 : est l'énergie produite par les PV et consommée localement dans la période.
- x_5 : est l'énergie produite par les PV et consommée localement sur un pas de temps.
- x_6 : est la variable de commande de l'achat du surplus voir [8.4.1-1](#).
- x_7 : est le surplus en kWh revendu au réseau.

La gestion de vente du surplus est :

Si $x_6=0 \implies x_7 = 0$ (cas d'achat $x_1 \neq 0$).

Si $x_6=1 \implies x_7 \neq 0$ (cas de revente).

1. dans notre étude ce paramètre était calculé depuis un modèle physique d'un module PV de type (Module Solar-Fabrik Série SF 150/2A)

Comme dans le cas du fournisseur d'énergie, le *concepteur de type de modèles* écrit des classes Java qui représentent le modèle analytique précédent. Les ports énergétiques sont les variables x_5 et x_7 . Le calcul de gain de revendre du surplus x_3 entre dans le bilan économique comme un port économique.

Un concepteur graphique est développé par cet acteur pour que le *concepteur de gestionnaire de système bâtiment* introduire les PV dans le système, ce concepteur est donnée dans la figure 14.1.

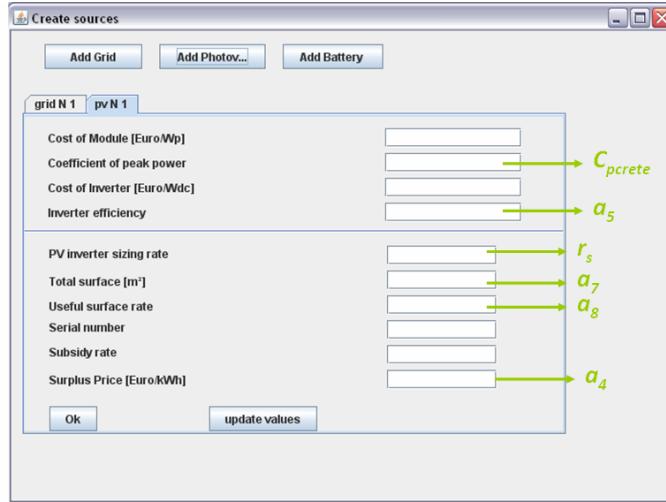


FIGURE 14.1 – Concepteur graphique développé par le *concepteur de type de modèles* pour que le *concepteur de gestionnaire du système bâtiment* puisse instancier le PIM OP-XML des panneaux photovoltaïques puis l'ajouter sur l'OP-XML complet du problème

14.1.2 PIM OP-XML de la batterie

La capacité de la batterie est un paramètre d'entrée dans le problématique de gestion. Ce modèle est illustré dans la formulation ?? :

$$\begin{aligned}
 Sup(x_{14}) &= a_{12} \times a_{13} \\
 Sup(x_{13}) &= a_{11} \times a_{13} \\
 x_{14}(t) &\leq (1 - x_{11}(t)) \times Sup(x_{14}) \\
 x_{13}(t) &\leq (1 - x_{11}(t)) \times Sup(x_{13}) \\
 x_{14}(t) + x_{12}(t-1) - a_{12} &\leq 0 \\
 x_{12}(t-1) - x_{13}(t) - a_{10} \times a_{12} &\geq 0 \\
 x_{12}(t) - a_{13} &\geq 0 \\
 x_{14}(t) - x_{13}(t) + x_{12}(t-1) - x_{12}(t) &= 0
 \end{aligned} \tag{14.2}$$

avec :

- a_9 : pourcentage pour définir la valeur initiale de l'état de la charge de la batterie.
- a_{10} : pourcentage pour définir la valeur minimale admissible de la charge de la batterie.
- a_{11} : la vitesse de décharge de la batterie.
- a_{12} : la vitesse de la charge de la batterie.

- a_{13} : la capacité maximale admissible de la batterie en kWh (Q_{max}), dans le cas de gestion de sources, elle est un paramètre.
- x_{11} : la commande de charge/décharge de la batterie : c'est une variable binaire.
- x_{12} : l'état de la charge de la batterie (*State Of Charge*).
- x_{13} : l'énergie de la décharge de la batterie en kWh.
- x_{14} : l'énergie de la charge de la batterie en kWh.
- $Sup(x_{14})$: la borne supérieure de x_{14} .
- $Sup(x_{13})$: la borne supérieure de x_{13} .

Enfin nous avons ajouté deux contraintes pour la périodicité et pour l'initialisation de la charge de la batterie. La contrainte 14.3 est pour initialiser la charge de la batterie avec une valeur de choix, alors que la contrainte 14.4 est pour imposer la périodicité en obligeant le solveur à trouver une solution avec laquelle la batterie retrouve sa charge initiale à la fin de la période d'optimisation. Nous avons, donc quatre combinaisons possibles pour piloter l'état de la charge de la batterie, en imposant ou pas ces contraintes.

$$x_{11}(0) = a_8 \times a_{12} \quad (14.3)$$

$$x_{11}(Studyperiod) = x_{11}(0) \quad (14.4)$$

Les ports énergétiques sont l'énergie de décharge et de charge de la batterie x_{10} et x_{11} . Nous n'avons pas pris en compte le vieillissement de la batterie ni le coût initial et le coût de replacement. Sinon, nous pourrons les ajouter comme des ports économiques dans le modèle.

Comme pour les autres sources, le *concepteur de type de modèles* développe un concepteur graphique pour entrer les données du modèle de la batterie. Ce concepteur est présenté dans la figure 14.2.

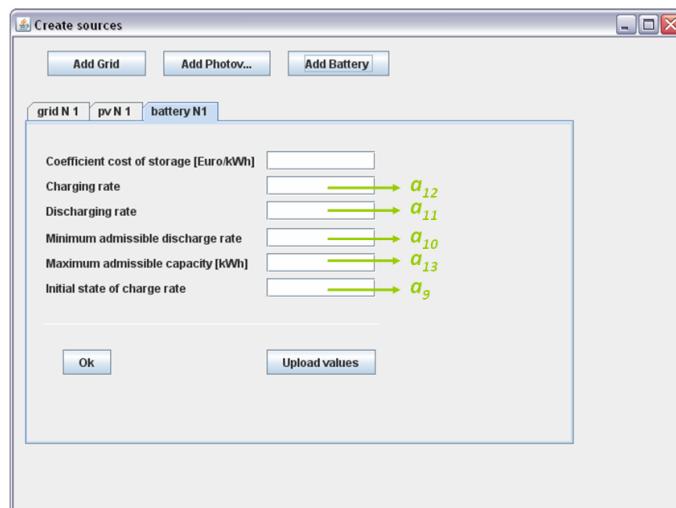


FIGURE 14.2 – Concepteur graphique développé par le *concepteur de type de modèles* pour que le *concepteur de gestionnaire de système bâtiment* puisse instancier le PIM OP-XML Batterie puis l'ajouter sur l'OP-XML complet du problème

Chapitre 15

Outil pour la gestion de bâtiment (G-HomeTech)

15.1 G-homeTech : Une implémentation de système de gestion énergétique pour un habitat

Nous présentons dans cet annexe le fruit d'une collaborations entre entreprises et laboratoires de recherche universitaires. Nous avons participé aux travaux de développement de l'outil G-homeTech, qui est sommairement un gestionnaire énergétique. Les concepts de génération automatique de problèmes d'optimisation et de projection présentés dans ce manuscrit ont été utilisés. Les équipements et composants (enveloppe par exemple) de l'habitat sont détectés et une contribution au problème de gestion optimal est ajoutée automatiquement pour chacun d'eux. Une fois généré, le problème d'optimisation peut être projeté pour être résolu soit par GLPK, soit par CPLEX.

15.1.1 Introduction

G-homeTech est une application de gestion de l'énergie dans l'habitat conçue dans le cadre des projets industriels ANR Multisol et Reactivhome. Ces projets résultent d'une collaboration entre différents partenaires : l'INES/CEA¹, Schneider Electric, le laboratoire G2ELab, le laboratoire G-SCOP, l'entreprise Armines et Orange.

Ces projets visent à concevoir un système de gestion d'énergie dans les bâtiments qui permette d'optimiser l'usage de l'énergie en cherchant les compromis optimaux entre le confort des occupants, formalisé par des fonctions de satisfaction, et un coût énergétique. Du point de vue du réseau électrique, cela permet de façonner la courbe de charge d'un habitat en agissant dynamiquement sur le tarif de l'énergie. L'architecture du système est illustrée par la figure 15.1.

15.1.2 Génération automatique de problèmes avec G-homeTech

Les principes de génération automatique de problèmes permettent, après avoir découvert les composants et équipements d'un habitat, d'ajouter dynamiquement à un problème d'optimisation initialement vierge, les contraintes propres à chaque composant ou équipement du système habitat. La génération automatique de problèmes revient à un principe de visiteur qui circule entre des générateurs de contraintes associés à chaque source ou charge électrique. Une seconde partie correspond aux contraintes de connectivité qui représentent les interactions entre les éléments structurels du système habitat, c'est-à-dire entre les composants et les équipements. Ce générateur générique a été implémenté sous forme d'une classe Java dans laquelle se trouvent des méthodes pour ajouter des variables

1. Institut Nationale d'Énergie Solaire

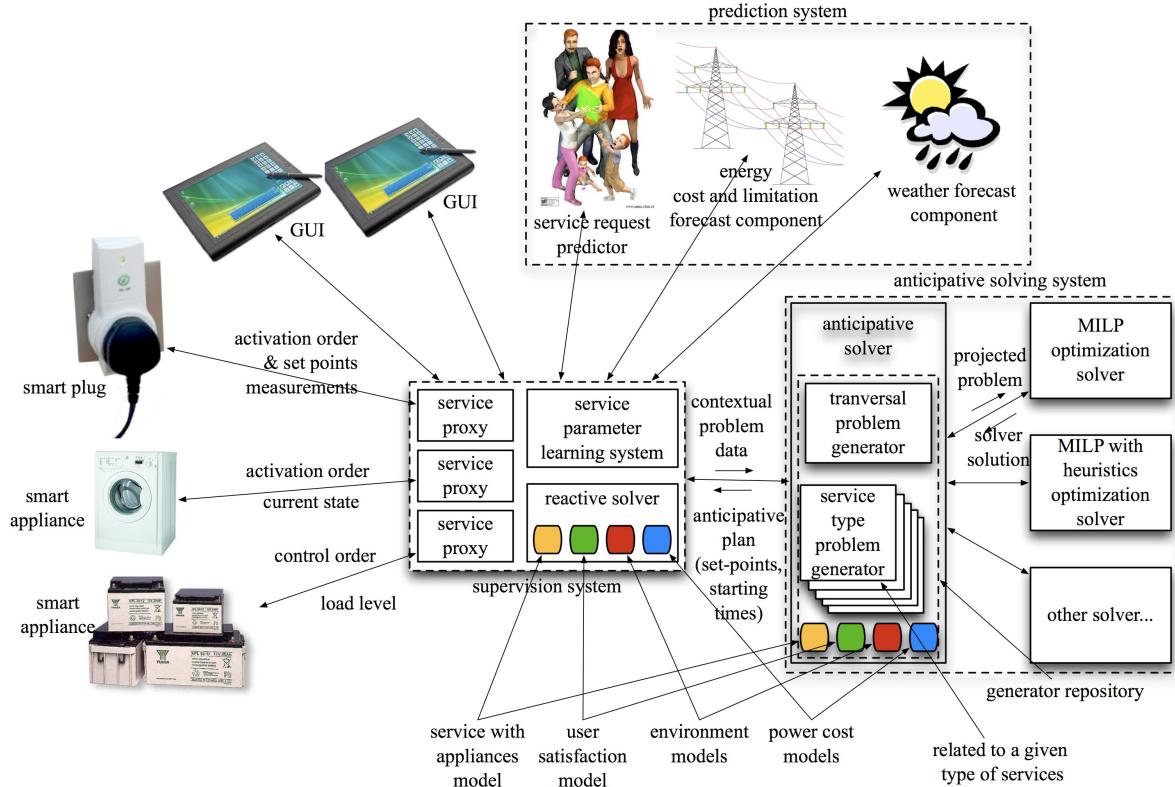


FIGURE 15.1 – Architecture de G-homeTech

et des contraintes. Les contraintes globales de connectivité, qui dépendent de plusieurs sources ou plusieurs charges, comme les contraintes de conservations de flux énergétiques, sont construites après les éléments structurels le générateur de problème va les construire selon les informations de provenance des composants.

15.1.3 Méthodes de gestion appliquées

Un système habitat non-isolé est composé de différentes sources d'énergie : de moyens de production locaux et d'un réseau électrique alimentant le système habitat. Il y a dans chaque maison ou appartement plusieurs charges, comme le chauffage ou les équipements électroménagers. Si le mécanisme de gestion d'énergie doit prendre en compte tous ces éléments, le problème de gestion devient très complexe. Pour réduire cette complexité, l'idée est de diviser la résolution en différents niveaux de résolution spécialisés (Ha et al. 2008). Les solutions optimales de ces sous-problèmes (en chaque niveau) constituent une première approche de résolution de problème complet. Une optimisation multi couches est implémentée en (Ha et al. 2008).

Les trois couches ordonnées selon le niveau d'abstraction sont :

La couche anticipative - Dans cette couche un plan anticipatif est recherché en prenant en compte des prédictions d'usages des équipements mais aussi des prédictions météorologique et de coût de l'énergie. Le calcul de plan anticipatif à pas horaire permet de réduire au maximum l'ajustement réactif (pour plus de détails voir (Ha et al. 2006) et (Long 2007)). Cette couche contient les modèles les plus abstraits de

charges et de sources. Cette couche cherche à optimiser un compromis confort / coût énergétique.

La couche réactive - L'objectif de cette couche est d'ajuster le fonctionnement des charges et des sources (selon les contraintes énergétiques) afin respecter le plan anticipative au mieux malgré l'occurrence d'événements imprévus. Elle intervient dans le cas de violation des contraintes de plan anticipative ou du confort souhaité par l'habitant. L'intervention peut être soit du type démarrer/éteindre, soit en modifiant des points de fonctionnement, comme par exemple la température de confort pour un radiateur dans une chambre. Le mécanisme réactif est beaucoup plus rapide que l'anticipative avec un pas de temps de l'ordre de la minute.

La couche locale - C'est le niveau des équipements dans l'habitat. Elle contient les systèmes de commande embarqués souvent dans les équipements par les fabricants. Cette couche effectue des ajustements permanents pour maintenir les consignes issue de la couche anticipative et éventuellement ajustée par la couche réactive. Un exemple est le mécanisme de régulation d'un climatiseur ou d'un chauffage électrique.

15.1.4 Architecture du système

G-HomeTech est constitué de quatre parties qui sont :

Le superviseur - C'est le noyau central du système, celui qui synchronise l'envoi, la réception des commandes et/ou des mesures. Il contient un *proxy* pour intégrer différentes *smart plugs*² ou/et des contrôleurs embarqués³. Le superviseur contient des données de configuration qui caractérisent les attentes des usagers par des fonctions de satisfaction pour chaque service rendu par des équipements, ou par des fonctions de coût pour les sources électriques. Il contient aussi un algorithme réactif pour décaler dynamiquement les services selon la satisfaction des occupants et le contexte énergétique. Le superviseur fait appel à un service de prévision météorologique pour demander la nouvelle prédiction. Il génère le problème d'optimisation puis il l'envoie vers le système de résolution. Il met à disposition les données pour les afficher sur les interfaces graphiques (*Graphical User Interface GUI*), ou il cherche des anciennes données d'une base correspondante.

Les interfaces graphiques - (*Graphical User Interface GUI*) sont des interfaces pour afficher les données qui sont soit stockées dans la base associée, soit acquises par des capteurs. Ces données peuvent être la consommation d'énergie, l'état des charges (ON/OFF), la satisfaction des occupants, la mesure de température, le coût d'énergie consommée ou vendue.etc.

Le système de prédiction - Ce système fournit au superviseur la prédiction de données météorologiques ainsi que la prédiction d'utilisation d'un équipement par l'usager. Il propose de plus le coût énergétique correspondant.

Le système de résolution anticipatif - Ce système résout le problème généré par le superviseur puis retourne le résultat de la résolution au superviseur.

L'architecture du système est illustrée la figure 15.1.

2. Ce sont des micro-contrôleurs connectés entre l'équipement et la prise électrique

3. Ce sont des micro-contrôleurs embarqués dans l'équipement possédant des moyens de communications

15.1.5 Procédure d'optimisation

Pour un problème typique d'anticipation, il peut y avoir plusieurs milliers de contraintes, avec des variables continues ou entières. Les solveurs de problème de type PLNE (Programmation Linéaire avec Nombre Entier) ont montré une efficacité plus importante que ceux de type programmation séquentielle quadratique (SQP). Pour cette raison, tous les modèles sont linéaires ou ont été linéarisés.

Le problème d'optimisation est généré de façon automatique, comme présenté le chapitre 2. Le générateur ajoute les contraintes transversales en fonction des composants installés dans le système (se référer à la figure 15.2).

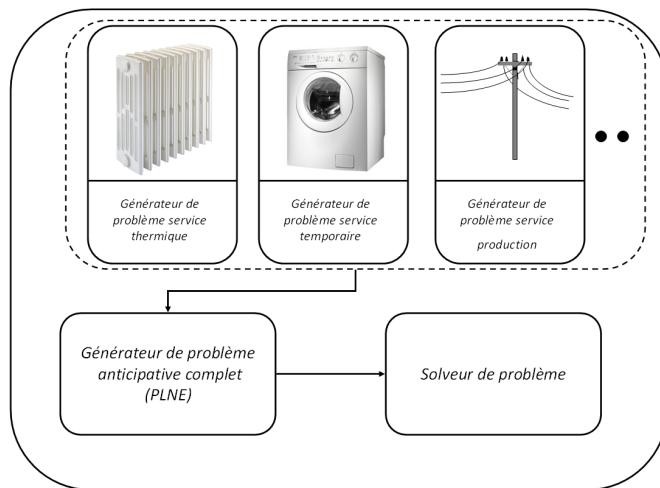


FIGURE 15.2 – Une représentation schématique pour le mécanisme anticipatif de résolution

La première étape de l'optimisation est la configuration du système. Cela est fait dans un fichier central (*setup file*). Les paramètres du système sont :

1. Une description de la base de données utilisée pour l'archivage.
2. une période de calcul pour le mécanisme anticipatif (typiquement une heure ou demi-heure).
3. un nombre de périodes à anticiper (typiquement 24 h).
4. une période de calcul pour le mécanisme réactif (typiquement quelques secondes).
5. une description de chaque service, par exemple :
 - (a) un nom de service.
 - (b) une localisation de contrôleur de ce service.
 - (c) un type de ce service (par exemple service de production d'énergie, service thermique, service temporaire, service de stockage, service non supervisé).
 - (d) un nombre booléen, *true* si le service est préemptif. Cela veut dire que le mécanisme réactif peut interrompre ce service.
 - (e) un nombre booléen, *true* si le service est contrôlable. Cela veut dire que le mécanisme anticipatif peut le piloter.

- (f) un nombre booléen, *true* si le service est prévisible. Cela veut dire qu'on peut estimer sa consommation.
- (g) une consommation/production maximale typique.
- (h) une liste des variables qui seront envoyées par le service vers le superviseur incluant les variables de consommation ou de production d'énergie depuis la dernière demande de superviseur.
- (i) une liste de paramètres liés à ce service pour aider à évaluer la satisfaction de l'utilisateur, ou le coût dans le cas d'une production d'énergie.
- (j) une liste de paramètres liés à ce service pour caractériser le modèle de comportement dans les deux cas, anticipatif et réactif.
- (k) une valeur de consigne par défaut (*set-point*)
- (l) un coefficient d'équilibrage entre le coût et la satisfaction de l'usager.
- (m) un coefficient de pondération, pour quantifier l'importance d'un service par rapport à l'ensemble des services proposés.

Globalement, le mécanisme anticipatif calcule les points de fonctionnement optimaux, puis il les envoie aux services. S'il y a eu un changement dans les données, le mécanisme réactif intervient pour modifier les valeurs de consigne et ainsi de suite (voir la figure 15.3).

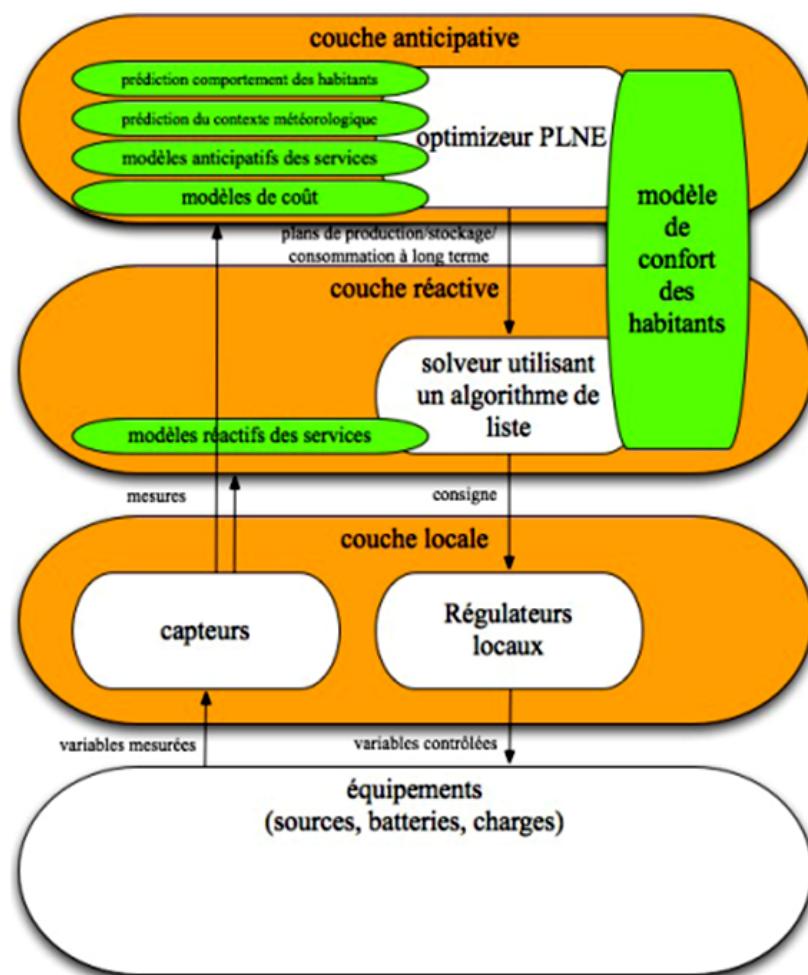


FIGURE 15.3 – Le principe d'optimisation multi couche

15.1.6 Validation en temps réel

Le système de gestion du bâtiment (G-homeTech) implémente les algorithmes anticipatifs et réactifs en temps accéléré. Cela veut dire que les équipements et les sources sont simulés dans un simulateur de vie qui permet d'implémenter différents scénarios d'utilisation des sources et des charges. Le mécanisme anticipatif applique les valeurs moyennes, alors que le mécanisme réactif applique les valeurs instantanées. Dans l'objectif d'une implémentation réelle (exploitation commerciale), il faut progresser vers une simulation temps réel, avec un pas de temps de l'ordre de la milli-seconde à la micro-seconde. Pour cela, il faut implémenter des modèles de sources électriques et de charges valides à ces échelles de temps.

Nous avons participé au travail de thèse de *Rim MISSAOUI* au sein des laboratoires GSCOP et G2Elab qui a comme objectif de tester les algorithmes de contrôle/commande dans le contexte temps-réel ([Missaoui et al. 2010](#)) et temps accéléré. Dans un premier temps, un système composé d'une source électrique traditionnelle (réseau de distribution), deux services thermiques (un réfrigérateur et un radiateur) et un service temporaire (lave linge) sont pris en compte. Pour un abonnement fixe au réseau de distribution, le superviseur trouve un plan anticipatif pour le fonctionnement de ce système tout en respectant les critères coût/confort (les températures de chauffage, le temps de démarrage du lave-linge, etc...). Un extrait des résultats de simulation temps-réel et des commandes envoyés par le superviseur sont présentés à la figure 15.4

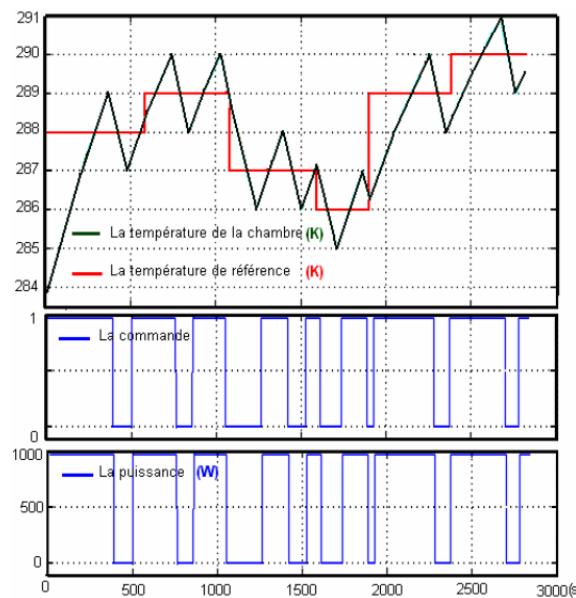


FIGURE 15.4 – Exemple de résultats de validation de commande - contrôle de chauffage électrique en temps réel

Les modèles de sources et de charges sont des modèles MATLAB avec l'interface de simulation temps-réel RT-LAB ([INC 2010](#)). Les modèles tournent sur un ordinateur appelé (PC1). Par contre, G-HomeTech, qui est un package Java, est installé sur un autre ordinateur (PC2). La communication entre G-HomeTech et les modèles RT-LAB est faite par l'intermédiaire du réseau informatique avec le protocole de communication TCP/IP (voir

figure 15.5). Nous avons crée une interface de communication à cette fin.



FIGURE 15.5 – Architecture matérielle G-HomeTech et simulateur temps réel RT-LAB

La suite de ce travail abordera une validation avec des sources d'énergie photovoltaïque ainsi qu'avec d'autres charges comme l'éclairage, le four, le micro-onde, etc... Un autre point à étudier sera une meilleure prise en compte de l'impact des occupants. Une fois la validation des algorithmes de gestion de l'énergie faite, à travers des modèles de bâtiment sur MATLAB/Simulink, on pourra réaliser une simulation *Hardware In The Loop* en utilisant de sources d'énergie électrique et des équipements réelles pour préparer une future implantation sur un site démonstrateur à l'échelle 1.

Chapitre 16

Méthode et modèle économique pour le dimensionnement

16.1 Les critères économiques pour le choix d'investissement

Nous allons étudier la faisabilité économique de bâtiment comme celle d'une centrale de production classique. En raisonnement relatif, les deux ont beaucoup de points communs sauf que le bâtiment est producteur au niveau microscopique par rapport aux centrales classiques. Ainsi nous pouvons appliquer les mêmes méthodes économiques pour calculer le taux de retour sur l'investissement puis la viabilité financière de l'investissement.

La conception de systèmes bâtiment du point de vue économique rencontre deux types de difficultés ([Percebois 1989](#)) :

1. Des difficultés liées à l'incertitude sur l'avenir : le concepteur n'aura pas d'informations complètes sur l'environnement dans le futur (par exemple, les prix de l'énergie, les prix des composants, le changement climatique.etc).
2. Des difficultés liées à l'horizon temporel considéré : par exemple, nous pourrions prendre en compte l'ensoleillement et la variation de la température prévus pour un an ou deux, mais ceci engendre des problèmes d'optimisation de très grande taille. La majorité des outils sur le marché ne prend en compte que certains jours typiques (comportement typique sur 24h) pour représenter une année.
3. Des difficultés liées à la comparaison dans le temps des diverses valeurs : pour justifier le choix de l'investissement, il faut calculer les dépenses et les recettes échelonnées dans le temps. Autrement dit, il faut trouver une équivalence entre des valeurs disponibles à différents moments. Par exemple, la recette de l'année 3 par rapport celle de l'année 0. Généralement, ce problème est résolu avec ce qu'on appelle l'actualisation : en rapprochant une valeur dans le futur à une valeur présente ([Binder et al. 2005](#)). La règle de base de la valeur présente est qu'un euro aujourd'hui n'aura pas la même valeur demain. Pour un investissement, l'intérêt d'un système multi-sources, requiert la connaissance de la valeur actuelle du bénéfice à la fin de la période de vie de l'installation. Nous calculons ce qu'on appelle le flux de trésorerie (*Cash flow*) sur la vie de l'installation. L'équation 16.1 présente le calcul du flux de trésorerie. Ainsi le calcul de la valeur présente nette (*Net Present Value NPV*) qui montre en valeur absolue le gain ou la perte de l'investissement est dans l'équation 16.3.

La formule qui donne le flux de trésorerie sur une année est :

$$CF_{year} = (Incomes_{year} - Depenses_{year}) \quad (16.1)$$

pour ramener cette valeur au présent nous utilisons la formulation suivante :

$$PCF_{year} = \frac{CF_{year}}{(1+a)^{year}} \quad (16.2)$$

soit :

PCF : la valeur présente (actualisée) du flux de trésorerie pour l'année *year*.

a : taux d'actualisation. Généralement, on prend le taux d'intérêt du marché pour une durée comparable, ou éventuellement le taux d'inflation anticipé. En effet, le choix de ce taux n'est pas technique mais économique. Souvent, on prend comme référence le taux du marché monétaire pour les durées courtes, ou celui des bons ou obligations du Trésor Public pour les durées plus longues.

Avec l'équation 16.2 nous pouvons calculer la valeur présente nette de l'investissement :

$$NPV = \sum_{year=N}^{year=0} PCF_{year} - InitialInvestment \quad (16.3)$$

La figure 16.1 montre un exemple de l'évolution de NPV sur la vie de l'installation. En fait, l'année où la courbe passe par l'axe horizontal est l'année d'amortissement des coûts totaux, réciproquement, il y a une valeur particulière de taux d'actualisation qui fait que le bénéfice total est nul. Cette valeur est appelée 'taux de rendement interne' du projet (Percebois 1989). Souvent, ce taux est utilisé pour définir la priorité de différents projets (dans le cas où nous en avons plusieurs).

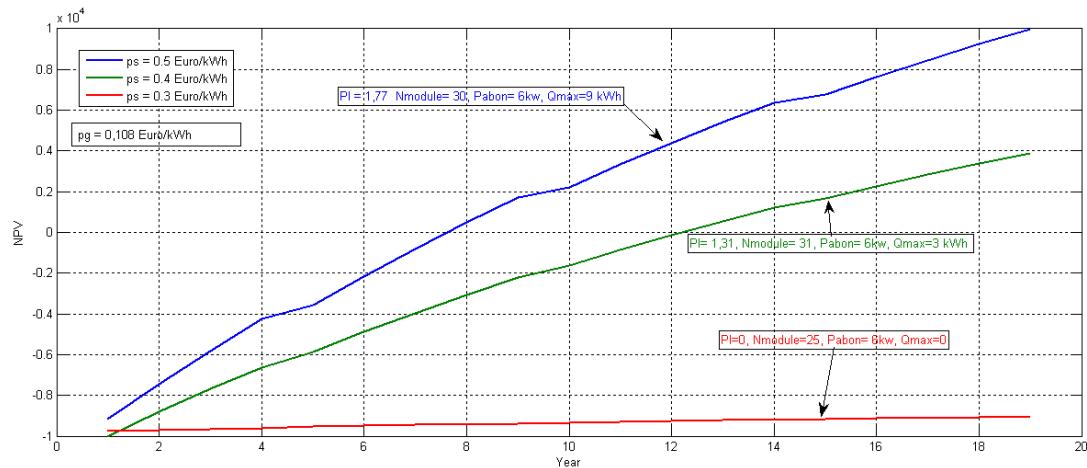


FIGURE 16.1 – Valeur présente nette NPV pour un investissement sur 20 ans pour plusieurs prix de surplus

Généralement, le choix d'un investissement se fait avec l'aide de l'indice de profitabilité PI. Il représente le pourcentage du bénéfice de l'investissement par rapport aux dépenses initiales. Pour calculer le PI l'équation 16.4 est utilisée :

$$PI = \frac{\sum PCF_{year}}{InitialInvestment} = \frac{NPV + InitialInvestment}{InitialInvestment} = \frac{NPV}{InitialInvestment} + 1 \quad (16.4)$$

Si $PI > 1$, l'investissement est rentable et le taux de rentabilité est (PI-1), par contre si $PI < 1$, l'investissement n'est pas rentable.

Le concepteur de bâtiment a besoin de critères pour l'aider à choisir économiquement son investissement. Avec les critères, nous calculons les indices présentés ci-dessus. Globalement, il y a deux types principaux de critères appliqués pour les centrales de production. Nous allons les citer rapidement puis nous présenterons comment on peut les adapter pour l'application bâtiment :

1. Critères de choix en avenir certain comme :

- Investissement par tep¹ économisée : on minimise le rapport de l'investissement total sur l'économie annuelle d'énergie réalisée.
- Délai de récupération de fonds investis : on cherche à trouver le temps nécessaire pour compenser les dépenses.
- Minimisation de coût global actualisé : on cherche à trouver la solution qui minimise les dépenses actualisées.
- Maximisation de bénéfice total actualisé : on cherche la solution qui maximise les recettes actualisées.
- Taux de rendement interne : on recherche le taux d'actualisation qui annule NPV, puis on choisit dans toutes les solutions le taux le plus élevé, car ceci veut dire qu'on va amortir les dépenses le plus vite possible.

2. Critères de choix en avenir incertain comme :

- en avenir probabilisable, on maximise l'espérance mathématique du bénéfice total actualisé.
- en avenir non probabilisable, on minimise le maximum de perte (*minimax*) ou on maximise le minimum de gain (*maximin*).

En fait, parmi ces critères nous favorisons le choix numéro (1) qui prend en compte le NPV pour l'application de bâtiment. Nous le modifions afin de maximiser le NPV selon une valeur précise du taux de rendement. En effet, pour l'application bâtiment, l'estimation du coût global uniquement avec des recettes n'est pas suffisante pour faire le choix d'investissement à cause du comportement de l'usager. Généralement quand on conçoit une centrale de production classique on sait d'avance que toute la production sera consommée (production = consommation). Alors que dans le bâtiment, le bilan n'est plus le même (production = consommation + surplus). De plus le prix de surplus est variable. En conséquence, le gain accumulé est variable. Pour toutes ces raisons nous ne pouvons pas faire le choix selon le coût uniquement ni selon le bénéfice tout seul. Le moyen qui permet de prendre en compte ces deux estimations en même temps est le plus adapté et c'est le calcul de NPV.

Dans le cas d'une grande centrale de production les décideurs ont assez d'expérience pour justifier leur choix avec des critères en avenir certain. Dans l'application bâtiment où les concepteurs bâtiment seront beaucoup plus nombreux, une expérience pareille n'est pas toujours garantie. Alors pour justifier (avec une tolérance) de travailler en avenir certain et utiliser le critère par NPV, il faut prendre en compte le plus possible de données qui décrivent l'avenir. Cela concerne les données météorologiques et la variation de prix.

1. tep : tonne d'équivalent pétrole

Chapitre 17

Tentative de mise en oeuvre d'une approche structurelle pour détecter les problèmes d'optimisation à multiples solutions équivalentes : Principaux résultats et limites mises en évidence

17.1 La décomposition Dulmage-Mendelsohn

17.1.1 Pourquoi l'approche par Dulmage-Mendelsohn (DM)

La méthode pour l'analyse structurelle est le découplage de Dulmage-Mendelsohn. Ce découplage est appliqué largement pour la résolution de systèmes linéaires, car grâce à lui nous pouvons obtenir une représentation en bloc triangulaire inférieur *Block triangular form (btf)* (Pothen & Fan 1990). Le bloc triangulaire inférieur (btf) est important pour les algorithmes qui traitent les matrices creuses parce qu'il permet de réduire le calcul et la mémoire nécessaire pour la résolution (Pothen & Fan 1990). Dans (Allain 2003) un *ordonnancement* selon la priorité de résolution d'un système d'équations a été obtenu grâce au btf.

Dans cette thèse, nous allons essayer de faire le découplage de Dulmage-Mendelsohn pour détecter le risque d'avoir l'effet W. Avec cette analyse, on examine la formulation des problèmes d'optimisation du point de vue structurel. Cela permettra de connaître quelle contrainte contient quelle variable, ainsi nous aurons une réponse pour la question : est ce que le problème est sous-déterminé ou juste déterminé ?. Pour commencer l'analyse structurelle on construit la matrice d'incidence, qui est une matrice creuse de $(m \times n)$ sachant que : m est le nombre de contraintes, n est le nombre des variables. Selon le résultat de découplage DM, nous pouvons a priori connaître si le problème est mal posé ou pas.

La différence entre notre application de découplage DM et les travaux mentionnés au-dessus est que nous l'appliquons pour un problème d'optimisation, alors que les travaux précédents ont appliqué DM pour résoudre un système d'équations. Cela impose qu'on transforme le problème d'optimisation en problème traitable par l'approche DM. Il s'agit de transformer les contraintes d'inégalité et la fonction objectif en contraintes d'égalité. Nous allons montrer les détails dans les paragraphes suivants.

17.1.2 Description de découplage Dulmage-Mendelsohn (DM)

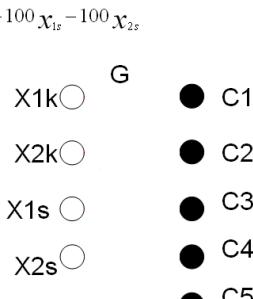
En permutant les colonnes et les lignes de la matrice d'incidence et en appliquant certaines règles classiques de la théorie des graphes, Dulmage et Mendelsohn ont développé

dans (Dulmage & Mendelsohn 1959) un algorithme pour obtenir le btf. Cet algorithme est basé sur la décomposition canonique du graphe bipartite (Asratian et al. 1998).

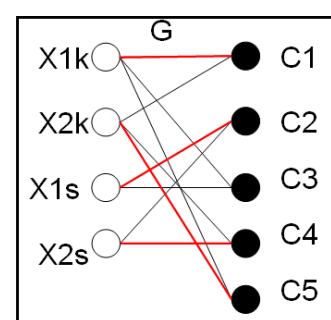
Pour comprendre le découplage DM, nous avons besoin d'utiliser certaines notions de la théorie des graphes, ces notions sont :

- Le graphe (G) : est une représentation graphique d'un système symbolisant ces individus et la relation entre eux. Les individus sont représentés par des noeuds (*Vertices v*), les relations entre ces individus sont représentées par des arcs (*Edges E*) entre les noeuds. Les graphes sont utilisés dans différents domaines tels que les circuits électriques (Bunus & Fritzson 2002), les réseaux de communications (Diestel 2005).etc. Il y a plusieurs types de graphes comme : graphe biparti, graphe linéaire, graphe orienté, etc. (Asratian et al. 1998). Nous allons utiliser dans la suite le graphe biparti.
- Le graphe biparti $G(v, E)$: est un graphe dont l'ensemble des noeuds est divisé en deux sous-ensembles. Les arcs lient les noeuds du premier sous-ensemble avec les noeuds de l'autre sous-ensemble. Par exemple dans l'application de circuits électriques, les variables sont représentées par des noeuds dans un sous-ensemble, les équations sont aussi représentées par des noeuds mais dans un autre sous-ensemble. Si une variable est contenue dans une équation, cette relation est illustrée par un arc qui lie le noeud de cette variable au noeud de cette équation. La figure 17.1 illustre qu'un problème d'optimisation peut être modélisé comme un graphe biparti (figure 17.1(a)).
- Le couplage (*Matching M*) : est un sous-ensemble des arcs E qui ne partagent pas le même noeud et qui lient les noeuds de deux sous-ensembles.
- Le couplage complet (*Maximum matching*) : est le couplage maximum possible dans un graphe ($\text{card}(E)$ est maximum). Il est représenté dans la figure 17.1(b) en traits rouges. Ce couplage signifie une dépendance forte entre les deux noeuds. Dans l'exemple de la figure 17.1(b), le noeud de x_{1k} est couplé par trois noeuds de contraintes (c_1, c_3 et c_5), mais le couplage complet est qu'avec c_1 , alors la valeur de x_{1k} ne se trouve que par la contrainte c_1 .

$$\begin{aligned}
 \min : & 150x_{1k} - 150x_{2k} - 100x_{1s} - 100x_{2s} \\
 \text{c1} : & 2x_{1k} - 8x_{2k} \geq 0 \\
 \text{c2} : & 8x_{1s} - 2x_{2s} \geq 0 \\
 \text{c3} : & x_{1k} + x_{1s} \geq 30 \\
 \text{c4} : & x_{2k} + x_{2s} \geq 70 \\
 \text{c5} : & x_{1k} + x_{2k} \geq 20
 \end{aligned}$$



(a) Graphe biparti $G(v, E)$



(b) Les arcs E en noir, en rouge couplage complet

FIGURE 17.1 – Un graphe biparti et un couplage

Supposons qu'on ait un système de $\sum = (K, V)$ qui représente un ensemble de n contraintes K liées avec un ensemble de m variables V . L'ensemble des variables V est constitué de deux sous ensembles : celui des variables connues O et celui de variables

inconnues X . Le système précédent peut être présenté structurellement par un graphe biparti $G = (K, V, A)$, où A est un ensemble d'arcs tel que $a(i, j) \in A$ si et seulement si la variable $v_i \in V$ apparaît dans la contrainte $k_i \in K$.

Avec le découplage DM, nous distinguons trois types canoniques de graphe biparti $G = (K, V, A)$ selon le couplage complet :

1. Graphe sur-déterminé (sur constraint) : s'il existe un couplage complet par rapport aux variables X et non par rapport aux contraintes K .
2. Graphe juste-déterminé (juste-constraint) : s'il existe un couplage complet par rapport aux variables X et aux contraintes K .
3. Graphe sous-déterminé (sous-constraint) : s'il existe un couplage complet par rapport aux contraintes K et non par rapport aux variables X .

Le découplage DM du graphe biparti produit trois composants canoniques K^+ , K^0 et K^- illustrés dans la figure 17.2.

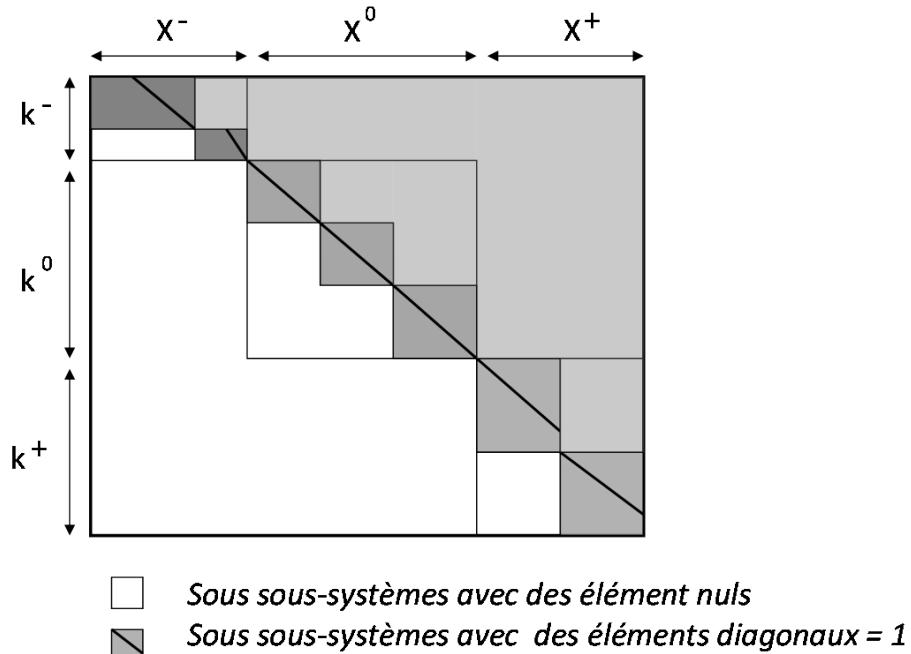


FIGURE 17.2 – Décomposition DM d'un graphe biparti en trois composants canoniques

On voit dans la figure 17.2 le bloc triangulaire inférieur (btf). On constate qu'il y a trois sous systèmes¹ :

$$\begin{aligned} S^+ &= (K^+, X^+) \\ S^0 &= (K^0, X^0) \\ S^- &= (K^-, X^-) \end{aligned} \tag{17.1}$$

Selon l'existence de ces sous-systèmes le problème analysé peut être jugé solvable ou pas. Par exemple pour qu'un système linéaire ait une solution, le sous-système sur-déterminé S^+ doit être $= \Phi$.

1. Pour savoir comment présenter graphiquement les trois sous-systèmes, directement sur le graphe biparti voir ([Bunus & Fritzson 2002](#))

L'ordre de résolution comme Allain (2003) l'a montré est défini par la représentation en bloc triangulaire inférieur. La priorité de la résolution est de trouver une solution pour le sous système où les composants sont couplés complètement (les traits rouges dans la figure 17.1(b)) selon leur apparition dans les blocs, puis calculer les valeurs des autres variables couplées en traits noires.

17.1.3 Application du découplage Dulmage-Mendelsohn pour l'analyse de systèmes linéaires

Le découplage DM est très utilisé dans le domaine de recherche opérationnelle, par exemple dans (Yassine 2008) les chercheurs optimisent l'emplacement des capteurs dans un processus industriel. Ils appliquent DM pour trouver la partie sous déterminée (S^- de 17.1) du problème, puis ils complètent cette partie avec de nouvelles contraintes qui sont les mesures des capteurs, puis ils résolvent cette partie (qui est maintenant juste-déterminée) comme un système d'équations. Les équations non valides dans cette partie représentent une erreur dans le système physique.

Dans le cadre de cette thèse, nous avons utilisé une fonction dans MATLAB (`dmperm()`) pour obtenir la décomposition DM de nos problèmes d'optimisation (les trois sous-système). Cela est réalisé après avoir transformé les contraintes d'inégalités en contraintes d'égalités comme nous allons voir dans l'exemple suivant.

17.1.3-1 Exemple de découplage Dulmage-Mendelsohn (DM) pour un problème d'optimisation linéaire

Préparer la formulation

Admettons qu'on ait le problème d'optimisation suivant :

$$\begin{aligned} & \text{Max } y \\ & c1 : x - y \leq 2 \\ & c2 : -y \leq -1 \\ & c3 : -x - y \leq -3 \end{aligned} \tag{17.2}$$

Pour faire la décomposition DM avec MATLAB, le problème d'optimisation doit d'abord être transformé en ajoutant les variables d'écart comme dans 17.3, parce que DM est applicable pour des systèmes d'équations mais pas pour des systèmes des contraintes, puis construire la matrice d'incidence qui représente le problème (voir 17.4).

$$\begin{aligned} & \text{Max } y \\ & c1' : x - y + e_1 = 2 \\ & c2' : -y + e_2 = -1 \\ & c3' : -x - y + e_3 = -3 \end{aligned} \tag{17.3}$$

$$\left[\begin{array}{ccccc} 1 & -1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ -1 & -1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{array} \right] \begin{bmatrix} x \\ y \\ e_1 \\ e_2 \\ e_3 \end{bmatrix} - \begin{bmatrix} 2 \\ -1 \\ -3 \\ \lambda \end{bmatrix} \tag{17.4}$$

Appliquer le couplage sur le problème transformé

L'appel de la fonction de décomposition DM se fait comme illustré dans la figure 17.3 :

```

clear all
clc

% c1 : x - y <= 2
% c2 : -y <= -1
% c3 : -x-y <= -3
% fo : y

% c1=[1 -1 1 0 0];
% c2=[0 -1 0 1 0];
% c3=[-1 -1 0 0 1];
% fo=[0 1 0 0 0];

c1=[1 -1 1 0 0];
c2=[0 -1 0 1 0];
c3=[-1 -1 0 0 1];
fo=[0 1 0 0 0];

k = [c1;c2;c3;fo];

[p,q,r,s] = dmperm(k) % ,q,r,s,cc,rr
rect = k(p,q) ;
spy(rect);

```

FIGURE 17.3 – Décomposition DM sous MATLAB

Le résultat de la décomposition est un groupe de vecteurs qui contient des indices des variables et des équations. Les indices permettent de trouver les trois sous-systèmes S^- , S^0 et S^+ . Les résultats et l'interprétation de ces résultats est dans le paragraphe 17.2. Les sous-systèmes S^- et S^0 de la décomposition de DM pour l'exemple 17.2 sont graphiquement présentés dans la figure 17.4.

C_1'	1	1	0	0	-1
C_3'	0	-1	1	0	-1
C_2'	0	0	0	1	-1
fo	0	0	0	0	1
	e_1	x	e_3	e_2	y

$\rightarrow S^-$

$\rightarrow S^0$

FIGURE 17.4 – Décomposition DM pour l'exemple 17.3

Remarque

Le problème posé en 17.2 a le risque de l'effet W. Comme la décomposition DM le montre, c_2 et fo sont dans le sous-système juste -déterminé avec y et e_2 . En effet, y et e_2 sont

figés alors que x non. x peut varier sans affecter la valeur de la fonction objectif fo . En conséquence, il peut y avoir deux solutions équivalentes.

17.1.3-2 Implémentation de découplage DM pour le problème de la gestion

Transformation de problème d'optimisation

Pour faire la décomposition DM pour le problème 9.1 (de chapitre 4), il faut qu'on transforme les contraintes en équations comme dans l'exemple précédent ; pour cela nous allons ajouter sept variables d'écart (e_i ; $i=1..7$). la formulation pour un instant (t) de 17.5 est donc :

$$\begin{aligned}
 c1 : & x_1(t) + x_2(t) - x_3(t) = P_{load}(t) - P_{pv}(t) \\
 c2 : & x_4(t) - x_4(t-1) - x_2(t) = 0 \\
 c3 : & x_1(t) + e_1 - P_{abon} = 0 \\
 c4 : & x_2(t) + e_2 - Q_{max} \times a_3 = 0 \\
 c5 : & x_2(t) - e_3 + Q_{max} \times a_2 = 0 \\
 c6 : & x_3(t) + e_4 - P_{pv}(t) = 0 \\
 c7 : & x_3(t) - e_5 = 0 \\
 c8 : & x_4(t) + e_6 - Q_{max} = 0 \\
 c9 : & x_4(t) - e_7 - Q_{max} \times a_1 = 0 \\
 fo : & (x_1(t) \times p_g(t) - x_3(t) \times p_s(t)) + \lambda = 0
 \end{aligned} \tag{17.5}$$

Problème de gestion découplé avec DM

L'implémentation dans MATLAB de la décomposition DM pour le problème 17.5 est présenté sur la figure 17.5.

Vu que le problème de la gestion est sur 24h, donc que le nombre de lignes de la matrice d'incidence sera (10 contraintes \times 24), et le nombre de colonnes (11 variables \times 24) et la taille de la matrice d'incidence est assez grande (240 \times 264). En appliquant les règles pour trouver les sous-systèmes nous obtenons :

1. le $S^+ = \phi$.
2. le $S^- = \{k(1 : 230, 1 : 254)\}$.
3. le $S^0 = \{k(231 : 240, 255 : 264)\}$.

En analysant le résultat du découplage DM (voir figure 17.12 et 17.11), nous pouvons extraire les points suivants :

1. Le problème structurellement a deux sous-systèmes qui sont S^- et S^0 (alors il n'y a pas sur détermination).
2. La partie juste-déterminée contient les variables et les contraintes du tableau 17.1. Les variables et les contraintes dans ce tableau sont en couplage complet, et elles sont ordonnées d'haut vers le bas selon la priorité de résolution comme trouvé par MATLAB.
3. La partie sous-déterminée contient le reste de variables et de contraintes.
4. En analogie avec la résolution du système d'équations linéaires (Allain 2003), la première étape pour résoudre le problème complet, est de trouver les valeurs de variables qui sont en couplage complet dans la partie sous-déterminé. Puis, calculer le reste des variables dans cette partie grâce à ces valeurs. Ensuite, calculer les valeurs

```

clear all
% les variables originaux et artificielles et leurs indices
% Pg 1
% Pb 2
% Ps 3
% SOC 4
% e1 5
% e2 6
% e3 7
% e4 8
% e5 9
% e6 10
% e7 11
timeConstant= 24;
%e0 =[0 0 0 1 0 0 0 0 0 0]; % initialize SOC
    % Pg Pb Ps soc    e1 e2 e3 e4 e5 e6 e7
c1 =[1 1 1 0      0 0 0 0 0 0 0];%1
c2 =[0 1 0 1      0 0 0 0 0 0 0];%2
c3 =[1 0 0 0      1 0 0 0 0 0 0];%3
c4 =[0 1 0 0      0 1 0 0 0 0 0];%4
c5 =[0 1 0 0      0 0 1 0 0 0 0];%5
c6 =[0 0 1 0      0 0 0 1 0 0 0];%6
c7 =[0 0 1 0      0 0 0 0 1 0 0];%7
c8 =[0 0 0 1      0 0 0 0 0 1 0];%8
c9 =[0 0 0 1      0 0 0 0 0 0 1];%9
f0 =[1 0 1 0      0 0 0 0 0 0 0];%10
c_tmoins1 = [0 0 0 1 0 0 0 0 0 0 0]; % pour calculer soc(t-1)

k_a=[c1;c2;c3;c4;c5;c6;c7;c8;c9;f0];
c_24 = diag(ones(timeConstant,1));
K_A = kron(c_24,k_a);
K_b = [c_tmoins1;zeros(1,11);zeros(1,11);zeros(1,11);zeros(1,11);
        zeros(1,11);zeros(1,11);
        zeros(1,11);zeros(1,11);zeros(1,11);
        K_B = kron(c_01,k_b);

K = K_A + K_B;
[p,q,r,s] = dmperm(K)    %[p,q,r,s,cc,rr]
rectang = K(p,q);
spy(rectang);

```

FIGURE 17.5 – Codes MATLAB pour décomposition DM du problème 17.5

des variables dans la partie juste-déterminée grâce aux variables connues dans la partie sous-déterminée.

La valeur de $x_1(0)$ (qui représente l'énergie achetée au réseau) par exemple, doit être calculer que par l'équation de $c_3(0)$, mais cette dernière contient aussi la variable $e_1(0)$ qui apparaît dans la partie sous-déterminée avec priorité plus importante de celle de x_1 (figure 17.12). En effet, selon cet ordre de résolution trouvée par MATLAB, il faut calculer d'abord e_1 , puis calculer x_1 . Mais e_1 est sous-déterminée, donc elle peut avoir différentes valeurs. En conséquence, $x_1(0)$ peut varier et induire l'effet W comme illustré dans la figure 9.9.

5. Malgré le fait que les deux solutions convergent sur la même stratégie après le point $t = 12$ am (comme illustré dans la figure 9.9), la décomposition par DM n'a pas mis les variables correspondantes en ces points dans la partie juste-déterminée. En effet, ceci illustre une limitation de cette approche, car elle ne prend pas en compte les valeurs numériques de variables et de seconds membres qui ont obligé les deux solveurs à converger vers la même solution.

En résultat, l'application du découplage DM pour détecter le risque de l'effet W n'est pas suffisant, vu qu'il ne prend pas en compte les valeurs numériques des variables et des seconds membres ni les valeurs numériques des coefficients des variables dans la formulation de problème d'optimisation.

Nous allons montrer dans la suite comme un exemple, la limite de cette approche au niveau des valeurs des coefficients des variables.

nom de la variable	nom d'équation
$e_6(0)$	$\longleftrightarrow c_8(0)$
$e_7(0)$	$\longleftrightarrow c_9(0)$
$x_4(0)$	$\longleftrightarrow c_2(0)$
$e_2(0)$	$\longleftrightarrow c_4(0)$
$e_3(0)$	$\longleftrightarrow c_5(0)$
$x_2(0)$	$\longleftrightarrow c_1(0)$
$e_4(0)$	$\longleftrightarrow c_6(0)$
$e_5(0)$	$\longleftrightarrow c_7(0)$
$x_3(0)$	$\longleftrightarrow fo$
$x_1(0)$	$\longleftrightarrow c_3(0)$

TABLE 17.1 – Les variables et l'équations dans la partie juste-déterminée

17.1.4 Limite de l'approche par décomposition Dulmage-Mendelsohn pour la détection de l'effet W

L'application de la décomposition de Dulmage-Mendelsohn n'est pas suffisante dans tous les cas pour détecter l'effet W. Cela vient du fait qu'elle ne tient pas compte des coefficients des variables. Les coefficients déterminent l'intersection entre la fonction objectif et les contraintes. Nous allons utiliser une notion connue dans le domaine des mathématiques appliquées pour affirmer la limite de l'approche structurelle. Il s'agit de la notion de pseudo-inverse d'une matrice (A^\dagger) (Moore 1920) qui va nous permettre de déterminer le degré de liberté, la chose que DM ne peut pas nous affirmer.

Nous allons prendre trois problèmes d'optimisation. Les trois problèmes ont les mêmes contraintes, nous avons juste changé les coefficients des variables dans la formulation de la fonction objectif. A chaque fois, on montre le résultat de la décomposition DM et on le compare avec le résultat de calcul de pseudo-inverse. Pour le premier problème d'optimisation, les deux analyses (DM et le pseudo-inverse) amènent vers la même conclusion : il y a un risque de l'effet W. Alors que les deux autres problèmes d'optimisation mettent en évidence la limite dont on parle (sans déterminer l'existence de risque d'effet W). Cela car le découplage DM donne la même décomposition pour les deux cas, alors que le pseudo-inverse présente une différence au niveau de degrés de liberté de solution (Warkozek, Ploix, Wurtz & Jacomino 2010a).

17.1.4-1 Le pseudo-inverse de la matrice de coefficient

Admettons qu'on ait le système d'équations linéaires suivant :

$$AX = B \tag{17.6}$$

Selon la régularité d'un système d'équations linéaires comme celui défini en 17.6, cette pseudo-inverse A^\dagger donne soit la solution de la norme euclidienne minimale (si le système 17.6 est régulier²) cette solution est $X^* = A^\dagger B$, soit la solution moindre carré de la norme euclidienne $X^* = A^\dagger B$ (si le système 17.6 est irrégulier) (Mayer 2001). Alors, il y a une solution générale de système 17.6 sous le format suivant :

2. Consistent veut dire le système a une solution

$$X^* = A^\dagger B + (I - A^\dagger A)\Psi \quad (17.7)$$

sachant que :

1. X^* : la solution générale du système linéaire.
2. A^\dagger : la pseudo-inverse de la matrice A.
3. B : le vecteur du second membre.
4. ψ : un vecteur de variables libres (*free variables*), il a la taille de X. Souvent, nous pouvons distinguer deux types des variables, des variables libres et des variables s'exprimant en fonction des variables libres. La solution d'un système linéaire est représentée par ces variables libres. Elles représentent donc les degrés de liberté dans le système.

Remarque

Pour avoir une solution unique la partie $(I - A^\dagger A)\Psi$ doit être nulle.

17.1.4-2 Implémentation de la pseudo-inverse

Le point qui nous intéresse avec la solution générale est la deuxième partie de la somme de la formulation 17.7 qui est $(I - A^\dagger A)\Psi$ car avec ce terme nous pouvons déterminer les degrés de liberté dans l'ensembles des variables. Le calcul de pseudo-inverse est présenté dans (Moore 1920). Nous avons utilisé dans ce travail une fonction en MATLAB s'appellant *pinv()* qui la calcule directement (cf. MATLAB Help).

1 : Cas où l'analyse structurelle et le pseudo-inverse conduit à la même conclusion

La résultat de $(I - A^\dagger A)$ pour l'exemple 17.2 est donné en 17.8 cela en supposant que toutes les variables de problème 9.1 sont libres (chose qui est impossible, mais ça ne change rien dans l'analyse) :

$$(I - A^\dagger A)\Psi = \frac{1}{3} \begin{bmatrix} 1 & 0 & -1 & 0 & 1 \\ 0 & 0 & -0 & 0 & 0 \\ -1 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ e_1 \\ e_2 \\ e_3 \end{bmatrix} \quad (17.8)$$

Analyse par le pseudo-inverse

On voit que la partie $(I - A^\dagger A)\Psi$ n'est pas nulle, il n'y a que les variables y et e_2 qui sont multipliées par une colonne de zéros (cela veut dire que quelque soit leurs valeurs, elles n'affectent pas la solution générale). En conséquence, elles ne peuvent pas donner de degré de liberté. Les variables x , e_1 et e_3 sont multipliées par des colonnes non nulles, donc elles peuvent ajouter des degrés de liberté à la résolution. Ces variables dans ce cas sont les *Free variables* mentionnées ci-dessus.

Analyse par la décomposition DM

Selon la décomposition DM, y et e_2 sont dans la partie juste-déterminée donc solution unique (voir la figure 17.4). Cela confirme le résultat d'analyse par pseudo-inverse. y est dans la formulation de la fonction objectif et aussi dans la partie juste-déterminée, ainsi

on a une seule solution pour y quelque soit les valeurs des autres variables, cela prouve l'existence des solutions équivalents, donc le risque d'avoir l'effet W.

2 : Cas où l'analyse structurelle est limitée, premier changement des coefficients

Si éventuellement le problème d'optimisation précédent 17.2 a une autre fonction objectif comme :

$$\begin{aligned} \text{Max } & x + y \\ c1 : & x - y \leq 2 \\ c2 : & -y \leq -1 \\ c3 : & -x - y \leq -3 \end{aligned} \quad (17.9)$$

La décomposition DM (après la transformation) donne la figure 17.6 :

fo	0	1	1	0	0
C_2'	1	0	1	0	0
C_1'	0	1	1	1	0
C_3'	0	1	1	0	1

$e_2 \quad x \quad y \quad e_1 \quad e_3$

FIGURE 17.6 – Décomposition DM après la transformation de l'exemple 17.9

La solution par pseudo-inverse dans ce cas contient un zéro pour la ligne de e_3 (voir la formulation 17.10). Cela veut dire qu'au contraire des autres variables, e_3 n'ajoute pas de degré de liberté au système (qui est maintenant égal à 4).

$$(I - A^\dagger A)\Psi = \frac{1}{16} \begin{bmatrix} 1 & -1 & -2 & -1 & 0 \\ -1 & 1 & 2 & 1 & 0 \\ -2 & 2 & 4 & 2 & 0 \\ -1 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ e_1 \\ e_2 \\ e_3 \end{bmatrix} \quad (17.10)$$

3 : Cas où l'analyse structurelle est limitée, deuxième changement des coefficients

Si on modifie encore le coefficient de y dans la fonction objectif par exemple $-2y$ (voir 17.11). La solution générale ne contient plus de zéro. Cela signifie qu'il y a cinq degrés de liberté (plus de degrés de liberté que dans le cas 17.9) :

$$\begin{aligned} \text{Max } & x - 2y \\ c1 : & x - y \leq 2 \\ c2 : & -y \leq -1 \\ c3 : & -x - y \leq -3 \end{aligned} \quad (17.11)$$

La décomposition DM donne dans ce cas le figure 17.7.

Le calcul de pseudo-inverse donne le résultat suivant :

<i>fo</i>	0	1	1	0	0
C_2'	1	0	1	0	0
C_1'	0	1	1	1	0
C_3'	0	1	1	0	1
	e_2	x	y	e_1	e_3

FIGURE 17.7 – Décomposition DM après la transformation de l'exemple 17.11

$$(I - A^\dagger A)\Psi = \frac{1}{16} \begin{bmatrix} 4 & 2 & -2 & 2 & 6 \\ 2 & 1 & -1 & 1 & 3 \\ -2 & -1 & 1 & -1 & -3 \\ 2 & 1 & -1 & 1 & 3 \\ 6 & 3 & -3 & 3 & 9 \end{bmatrix} \begin{bmatrix} x \\ y \\ e_1 \\ e_2 \\ e_3 \end{bmatrix} \quad (17.12)$$

Malgré la différence entre le nombre de degrés de liberté entre 17.9 et 17.11, la décomposition donne la même partie sous-déterminée (en fait la même partie présentée dans la figure 17.6) et 17.7). L'analyse structurelle n'est pas capable de trouver cet changement de degré de liberté (figure 17.8).

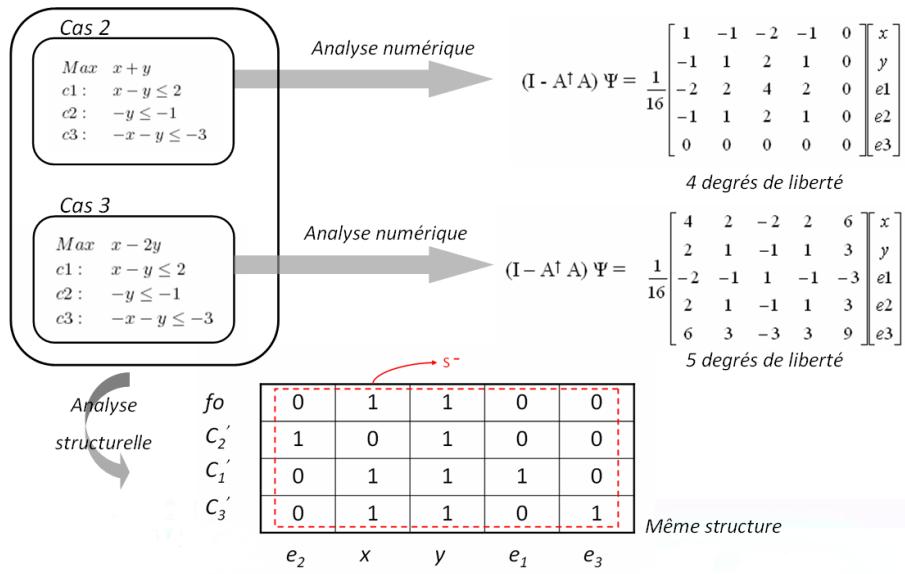


FIGURE 17.8 – Limite de l'approche structurelle pour détecter l'effet W

En résumé, comme illustré avec les exemples précédents, et avec le paragraphe 17.1.3-2 (les courbes après 12am) le problème de gestion suite à l'impact des valeurs numériques des coefficients et des second membres, peut avoir une solution unique même si l'analyse par le découplage DM donne une seule partie sous-déterminée. En conséquence, la détermination de l'existence des solutions équivalentes ainsi que le risque de l'effet W n'est pas toujours possible par le découplage DM.

Bien que le calcul de pseudo-matrice permette de trouver les degrés de liberté dans le

système, nous ne l'avons pas utilisé dans la suite pour détecter les solutions équivalentes. La pseudo-inverse donne des informations insuffisantes sur les variables libres, car si elle montre qu'elles sont libres nous ne pouvons pas savoir si ces variables peuvent prendre différents valeurs sans changer la valeur de la fonction objectif.

Remarque 1

Le couplage de l'approche de détection par DM avec le calcul de pseudo-inverse peut servir à augmenter la capacité de détecter les cas de solutions équivalents. Mais cela n'était pas envisagé dans la thèse, car nous cherchons un moyen qui est plus sûr, et qui à la fois détecte les degrés de liberté et donne une évaluation de la marge possible.

Remarque 2

Les techniques de découplage avec DM et la pseudo-inverse sont applicables pour des problèmes linéaires. Alors que le problème de gestion de tel système électrique peut être aussi non linéaire. Pour cela il faut chercher vers des techniques qui sont utilisables pour tous les cas.

17.2 Interprétation du résultat de découplage DM en MATLAB

L'interprétation du résultat pour l'exemple 17.3 à l'aide de *MATLAB Help* est :

1. p : c'est un vecteur dans lequel on trouve les indices des équations ordonnées selon la décomposition. Selon le résultat l'ordre des équations est le suivant : c1,c3,c2 et fo.
2. q : c'est un vecteur dans lequel on trouve les indices des variables ordonnées selon la décomposition. Ceci veut dire que l'ordre des variables est : e1,x,e3,e2 et y.
3. r : c'est un vecteur des indices avec lesquels nous distinguons les lignes de chaque sous-système.
4. s : c'est un vecteur des indices avec lesquels nous distinguons les colonnes de chaque sous-système.

Le résultat obtenu par MATLAB est :

Pour obtenir les trois sous-systèmes on applique les règles suivantes :

1. La partie sous-déterminée S^- est l'ensemble déterminé par :

$$k(r(1) : r(2) - 1, s(1) : s(2) - 1) \quad (17.13)$$

En appliquant les valeurs de r et s de la figure 17.9, on trouve alors que S^- est l'ensemble $k(1 : 2, 1 : 3) = \{k(1, 1), k(1, 2), k(1, 3), k(2, 1), k(2, 2), k(2, 3)\}$.

2. La partie sur-déterminée S^+ est l'ensemble déterminé par :

$$k(r(b) : r(b + 1) - 1, s(b) : s(b + 1) - 1); b = \text{length}(r) - 1 \quad (17.14)$$

En appliquant cette règle pour les valeurs de la figure 17.10, on trouve que le problème n'a pas de sous-système sur-déterminé S^+ (Il s'agit de $k(4 : 4, 5 : 5)$).

3. La partie juste-déterminée est ce qui reste de k .

Dans ce cas $S^0 = \{k(3, 4), k(3, 5), k(4, 4), k(4, 5)\}$.

```

p =
    1     3     2     4

q =
    3     1     5     4     2

r =
    1     3     4     5

s =
    1     4     5     6

cc =
    1     2     4     6     6

rr =
    1     3     5     5     5

```

FIGURE 17.9 – Résultats de la décomposition DM sous MATLAB

	e_1	x	e_3	e_2	y
C_1'	1	1	0	0	-1
C_3'	0	-1	1	0	-1
C_2'	0	0	0	1	-1
fo	0	0	0	0	1

FIGURE 17.10 – La décomposition DM

17.3 Résultat de découplage DM pour problème de gestion par MATLAB

```

p =
Columns 1 through 21
11 12 20 21 13 14 15 16 17 18 19 30 22 26 31 23 24 25 27 28 29
Columns 22 through 42
40 32 36 41 33 34 35 37 38 39 50 42 46 51 43 44 45 47 48 49 60
Columns 43 through 63
52 56 61 53 54 55 57 58 59 70 62 66 71 63 64 65 67 68 69 80 72
Columns 64 through 84
76 81 73 74 75 77 78 79 90 82 86 91 83 84 85 87 88 89 100 92 96
Columns 85 through 105
101 93 94 95 97 98 99 110 102 106 111 103 104 105 107 108 109 120 112 116 121
Columns 106 through 126
113 114 115 117 118 119 130 122 126 131 123 124 125 127 128 129 140 132 136 141 133
Columns 127 through 147
134 135 137 138 139 150 142 146 151 143 144 145 147 148 149 160 152 156 161 153 154
Columns 148 through 168
155 157 158 159 170 162 166 171 163 164 165 167 168 169 180 172 176 181 173 174 175
Columns 169 through 189
177 178 179 190 182 186 191 183 184 185 187 188 189 200 192 196 201 193 194 195 197
Columns 190 through 210

```

La contrainte pour calculer $e_i(0)$



FIGURE 17.11 – Les indices d'équations p ordonnées selon la priorité de résolution par le découplage DM

$q =$

```

Columns 1 through 21
5 30 41 52 63 74 85 96 107 118 129 140 151 162 173 184 195 206 217 228 239
Columns 22 through 42
250 261 263 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
Columns 43 through 63
31 32 33 34 35 36 37 38 39 40 42 43 44 45 46 47 48 49 50 51 53
Columns 64 through 84
54 55 56 57 58 59 60 61 62 64 65 66 67 68 69 70 71 72 73 75 76
Columns 85 through 105
77 78 79 80 81 82 83 84 86 87 88 89 90 91 92 93 94 95 97 98 99
Columns 106 through 126
100 101 102 103 104 105 106 108 109 110 111 112 113 114 115 116 117 119 120 121 122
Columns 127 through 147
123 124 125 126 127 128 130 131 132 133 134 135 136 137 138 139 141 142 143 144 145
Columns 148 through 168
146 147 148 149 150 152 153 154 155 156 157 158 159 160 161 163 164 165 166 167 168
Columns 169 through 189
169 170 171 172 174 175 176 177 178 179 180 181 182 183 185 186 187 188 189 190 191
Columns 190 through 210
192 193 194 196 197 198 199 200 201 202 203 204 205 207 208 209 210 211 212 213 214
Columns 211 through 231
215 216 218 219 220 221 222 223 224 225 226 227 229 230 231 232 233 234 235 236 237
Columns 232 through 252
238 240 241 242 243 244 245 246 247 248 249 251 252 253 254 255 256 257 258 259 260
Columns 253 through 264
262 264 10 11 4 6 7 2 8 9 3 1

```

Indice de $e_1(0)$

Indice de $x_1(0)$

FIGURE 17.12 – Les indices de variables q ordonnées selon la priorité de résolution par le découplage DM

Chapitre 18

Transformation de valeur absolue d'une expression avec l'aide des variables binaires

Soit l'expression à transformer :

$$E = \sum_{i=0}^n |dx_i| \quad (18.1)$$

avec $dx_i \in [-M_i, M_i]$

Sachant que : M_i est un majorant de dx_i . Pour remplacer la valeur absolue de dx_i , nous avons besoin de définir une variable binaire δ_i .

$$\delta_i = (dx_i \geq 0)$$

Avec cette variable nous pouvons retrouver l'intervalle précédent $[-M_i, M_i]$ avec la formulation suivante :

$$\begin{aligned} dx_i &\geq (\delta_i - 1)M_i \\ dx_i &< \delta_i M_i \end{aligned} \quad (18.2)$$

Par exemple :

$$\delta_i = 0 \implies dx_i \geq -M_i \quad (18.3)$$

$$dx_i < 0 \quad (18.4)$$

$$\delta_i = 1 \implies dx_i \geq 0 \quad (18.5)$$

$$dx_i < M_i \quad (18.6)$$

L'expression E devient alors :

$$E = \sum_{i=0}^n (2\delta_i - 1)dx_i \quad (18.7)$$

avec $dx_i \in [-M_i, M_i]$

Ainsi, l'expression (18.1) devient (18.7) avec des variables δ_i binaires vérifiant (18.2).

4818. Transformation de valeur absolue d'une expression avec l'aide des variables binaires

Il reste néanmoins un produit dans l'expression (18.7). On aboutit donc au problème final suivant :

$$\begin{aligned}
 E &= \sum_{i=0}^n (z_i - dx_i) \\
 dx_i &\geq (\delta_i - 1)M_i \\
 dx_i &< \delta_i M_i \\
 z_i &\leq 2\delta_i M_i \\
 z_i &\geq -2\delta_i M_i \\
 z_i &\leq 2dx_i + 4M_i(1 - \delta_i) \\
 z_i &\geq 2dx_i - 4M_i(1 - \delta_i)
 \end{aligned} \tag{18.8}$$

avec :

$$\begin{aligned}
 dx_i &\in]-M_i, M_i[\\
 z_i &\in]-2M_i, 2M_i[\\
 \delta_i &\in \{0, 1\}
 \end{aligned}$$

Vérification $\delta_i = 1$ ($dx_i \geq 0$) :

$$\begin{aligned}
 E &= \sum_{i=0}^n (z_i - dx_i) \\
 dx_i &\geq 0 \\
 dx_i &< M \\
 z_i &\leq 2M_i \\
 z_i &\geq -2M_i \\
 z_i &\leq 2dx_i \\
 z_i &\geq 2dx_i
 \end{aligned}$$

Vérification $\delta_i = 0$ ($dx_i < 0$) :

$$\begin{aligned}
 E &= \sum_{i=0}^n (z_i - dx_i) \\
 dx_i &\geq -M_i \\
 dx_i &< 0 \\
 z_i &\leq 0 \\
 z_i &\geq 0 \\
 z_i &\leq 2dx_i + 4M_i (< 2M_i) \\
 z_i &\geq 2dx_i - 4M_i (> -2M_i)
 \end{aligned}$$

18.1 Object Management Group OMG



FIGURE 18.1 – Les entreprises internationales participantes en OMG ([Soley 2010](#))