Relaxation lagrangienne et filtrage par coûts réduits appliqués à la production d'éléctricité.

Thierry Benoist, Maurice Diamantini, Benoît Rottembourg

Relaxation lagrangienne et filtrage par coûts réduits appliqués à la production d'électricité

Thierry Benoist BOUYGUES/e-lab
Maurice Diamantini ENSTA/LMA
Benoît Rottembourg BOUYGUES/e-lab

4 juin 2005

Résumé

Le problème ucp *(Unit Commitment Problem) consiste à planifier la production d'un parc de centrales électriques de manière à satisfaire un besoin prévisionnel donné sur une échelle de temps discrétisée (besoin horaire sur 24 heures). L'objectif consiste à définir à moindre coût d'une part, l'ordonnancement d'allumage/extinction de chaque centrale sur toute la période considérée, et d'autre part, la production de chaque centrale pour toute date où elle est allumée, de façon à satisfaire l'ensemble des deux contraintes globales (demande prévisionnelle et réserve de 10% modélisant l'incertitude de la prévision) et de trois contraintes techniques propres à chaque générateur : puissance bornée, temps minimum d'arrêt avant redémarrage, temps minimum de fonctionnement avant extinction. La fonction de coût d'un générateur comprend un coût de fonctionnement légèrement quadratique auquel s'ajoute un coût de démarrage dépendant de la durée d'arrêt d'une centrale que l'on allume.*

Nous réalisons une relaxation lagrangienne en dualisant les contraintes globales, et nous résolvons le problème ainsi relâché par programmation dynamique après avoir précalculé pour chaque date la production optimale connaissant les multiplicateurs de Lagrange. La programmation dynamique permet également le calcul du coût réduit[1] *nécessaire pour compenser le viol de certaines contraintes pour l'itération suivante. Ceci nous fournit une borne duale additive que nous exploitons d'une part pour améliorer la borne inférieure, et d'autre part pour filtrer des variables par Programmation Par Contraintes en cours du processus d'énumération implicite. Les coûts réduits sont également exploités pour guider le choix des couples (variable – valeur) dans la phase de séparation du Branch and Bound.*

1 Introduction

La production d'énergie est un enjeu économique, industriel et politique très important, plus encore avec l'internationalisation du marché de l'électricité. Malgré un souhait collectif de contrôler la consommation, le besoin a plutôt tendance à croître, non seulement en quantité, mais aussi et surtout (les intempéries aidant) en qualité de service. Cependant la ressource disponible est une quantité limitée, et n'est extensible qu'au prix de très coûteux investissements, et sur des décisions politiques ayant des horizons se chiffrant en décennies. Tous ces critères alliés conduisent à gérer au mieux les ressources disponibles.

Parmi cette problématique générale, le problème ucp consiste à coordonner la production d'unités génératrices, de façon à satisfaire une demande globale d'électricité prévisionnelle sur 24h. L'aléa est pris en compte par une contrainte de robustesse dite de « réserve ». Le problème ucp de base est classiquement bien traité par relaxation lagrangienne [1]. Cependant, du point de vue opérationnel, il existe d'autres contraintes liées par exemple à :

– la maintenance des générateurs (durée maximale d'utilisation ininterrompue, ...),

– la répartition géographique de la demande (pas moins de deux centrales allumées par région, ...).

Ces contraintes opérationnelles sont habituellement traitées – depuis de nombreuses années – dans des modèles séparés, voire par des équipes différentes.

[1] Dans ce rapport, le terme « coût réduit » désignera un coût différentiel, ou de « regret », pour passer d'une solution à une autre par le changement de valeur d'une seule variable.

Notre travail a pour but de valider une approche permettant d'appréhender le problème dans sa globalité en faisant coopérer un modèle économique et un modèle avec des contraintes opérationnelles. Pour cela, notre démarche consiste à coupler une méthode classique de relaxation lagrangienne avec un solveur à base de *Programmation Par Contraintes*, bien adapté à la prise en compte de nouvelles contraintes. En effet, l'écart Dual-Primal associé aux informations de coûts réduits permet de fixer des variables lors de l'exploration d'un arbre de recherche (*Branch and Bound*). Cette démarche, déjà validée pour la gestion d'autres actifs industriels (tels que des réseaux routiers, de l'éclairage public ou du matériel de chantiers, ...), sera appliquée ici, d'une manière originale, à la gestion d'un parc de générateurs d'électricité.

Après une description du problème UCP comprenant des besoins, des ressources disponibles, et des objectifs à satisfaire, nous présentons le modèle de résolution adopté pour le problème formalisé, en détaillant en particulier le choix des contraintes dualisées pour la relaxation lagrangienne. Cette relaxation lagrangienne nous fournit une information globale qui est un minorant de notre solution recherchée. Nous décrivons alors une technique de résolution par heuristique lagrangienne par réparation des solutions duales. Cette heuristique simple et rapide nous fournit d'une part, une solution permettant de mesurer la qualité du minorant obtenu, et d'autre part nous servira de référence pour évaluer quantitativement l'apport de notre approche multi-solveur.

Ensuite, nous définissons les ***coûts réduits***, et nous montrons comment les calculer ainsi que deux manières originales (pour ce problème) de les exploiter dans un *Branch and Bound* : à savoir pour améliorer le minorant par une ***borne duale additive***, et pour fixer des variables par ***filtrage***. Après une introduction aux principes de la *Programmation Par Contraintes* et aux techniques énumératives de type *Branch and Bound*, ainsi qu'à leur version tronquée (LDS pour *Limited Discrepency Search*) nous présenterons notre algorithme global.

Enfin, nous détaillerons nos résultats et leurs analyses avant de conclure.

---

2 Présentation du *Unit Commitment Problem* (UCP)

2.1 Le problème à résoudre

Le problème UCP (Unit Commitment Problem) consiste à planifier la production d'un parc de centrales électriques de manière à satisfaire un besoin prévisionnel donné sur une échelle de temps discrétisée (besoin horaire sur 24 heures ou besoin hebdomadaire sur 52 semaines). Cependant, l'électricité ne se stocke pas (du moins pas en une telle quantité), et il faut par conséquent produire à la demande. Or les contraintes techniques des générateurs ne permettent pas de répondre instantanément et dans n'importe quelle condition à cette demande. Il faudra donc anticiper la production en fonction du besoin prévisionnel.

Finalement le problème consiste à définir à moindre coût :

− d'une part, l'ordonnancement d'allumage/extinction de chaque centrale sur toute la période considérée,
− d'autres part, la production de chaque centrale pour toute date où elle est en marche,

de façon à satisfaire l'ensemble des contraintes.

2.2 Le besoin : prévision de la demande en électricité

Comme illustré sur la figure 1, la production totale doit satisfaire une demande globale. La demande est globalisée dans le sens où l'on ne tient pas compte de la variation des pertes en ligne qui pourraient être induites par une modification de la répartition des puissances élémentaires pour une même demande globale : tout se passe en effet comme si les centrales devaient satisfaire un seul gros client, le tout situé au même endroit (agrégation du réseau). Plus précisément, pour chaque date, la somme des productions de toutes les centrales doit égaler la demande pour cette date.

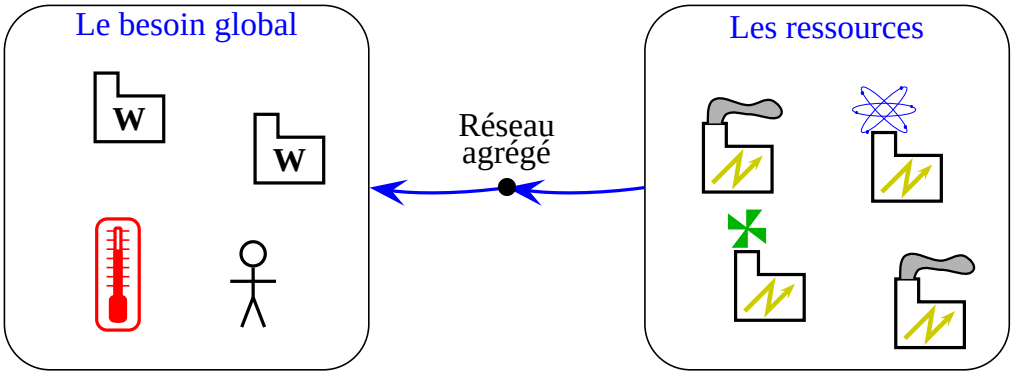

Figure 1 *globalisation du besoin en électricité et agrégation du réseau*

2.3 Les ressources : des centrales électriques

Les unités génératrices sont différentes, tant au niveau des **contraintes techniques** qu'elles doivent satisfaire, qu'au niveau de leur **coût d'exploitation**.

Aspects techniques

Tout d'abord, la capacité de production d'une centrale allumée est bornée entre deux valeurs extrêmes $p_{\min u}$ et $p_{\max u}$. Ensuite, comme l'illustre la figure **2**, la commande de mise en route d'un générateur est soumise à des contraintes d'inertie. En effet, l'extinction d'une centrale ne peut pas se faire immédiatement après un allumage : pour des raisons de fiabilité, elle doit rester allumée au minimum pendant $t_{\mathrm{up}u}$ intervales de temps. De même, une durée minimale d'arrêt $t_{\mathrm{down}u}$ doit être respectée avant un nouvel allumage.

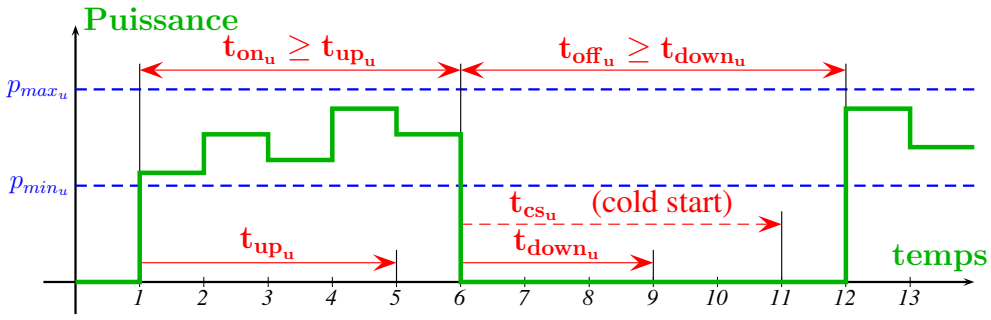

Figure 2 *Contraintes dynamiques de commande des générateurs électriques*

Du fait de la discrétisation du temps, la demande comme la réponse en production de chaque unité sont des fonctions de $\mathbb{N}$ sur $\mathbb{R}$.

Aspects économiques

Le coût d'exploitation d'une centrale est composé d'un coût de fonctionnement, et d'un coût de démarrage.

Le **coût de fonctionnement** d'une unité u peut être modélisé comme une fonction non linéaire $c_{\mathrm{f}u}$ de la puissance p_u produite (légèrement quadratique dans sa plage de fonctionnement : voir figure 3). Ce coût est donc défini par la valeur de trois coefficients spécifiques à l'unité :

$$c_{\mathrm{f}u}(p_u) = a_{0_u} + a_{1_u}.p_u + a_{2_u}.p_u^2 \tag{1}$$

Le **coût de démarrage** n'est compté qu'une seule fois lors de chaque démarrage. Sa valeur dépend de l'état de refroidissement de la centrale : une centrale chaude coûtera moins cher à démarrer qu'une centrale ayant eu le temps de se refroidir (mais il est nécessaire de respecter le temps minimum d'extinction $t_{\mathrm{down}u}$).

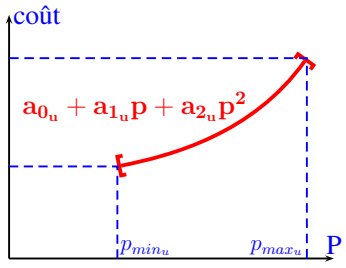

Figure 3 *Coût de fonctionnement d'une unité*

Plusieurs modèles physiques sont utilisables pour prendre en compte cette caractéristique [2]. Le modèle retenu définit deux valeurs de coût de démarrage à chaud $c_{\mathrm{hs}u}$ ou à froid $c_{\mathrm{cs}u}$ suivant que la durée d'arrêt de l'unité u est supérieure ou non à sa durée de refroidissement seuil $t_{\mathrm{cs}u}$:

$$c_{\mathrm{s}u}(t) = \begin{cases} c_{\mathrm{hs}u} & \text{si} \quad t_{\mathrm{off}u} < t_{\mathrm{cs}u} \\ c_{\mathrm{cs}u} & \text{si} \quad t_{\mathrm{off}u} \geq t_{\mathrm{cs}u} \end{cases} \tag{2}$$

avec :

– $t_{\mathrm{off}u} \in \mathbb{N}$: durée d'extinction de la centrale,

– $t_{\mathrm{cs}u} \in \mathbb{N}$: durée d'extinction au delà de laquelle la centrale u est jugée froide (*Cold Start*).

La méthode de résolution que nous proposons reste applicable quelque soit le modèle de refroidissement retenu, c'est à dire pour n'importe quelle fonction $c_{\mathrm{s}u}(t)$.

2.4 Gestion de l'incertain : garantir une réserve

L'incertitude inhérente à toute prévision est prise en compte par l'obligation de garantir, pour chaque date, une réserve de capacité de production, disponible sans délai (donc sans avoir à allumer de nouvelle centrale). Cette réserve est de l'ordre de 10% de la demande prévue pour chaque date. Lorsqu'une centrale est allumée, sa contribution à la réserve est égale à la différence entre sa production actuelle, et la production maximale qu'elle peut fournir.

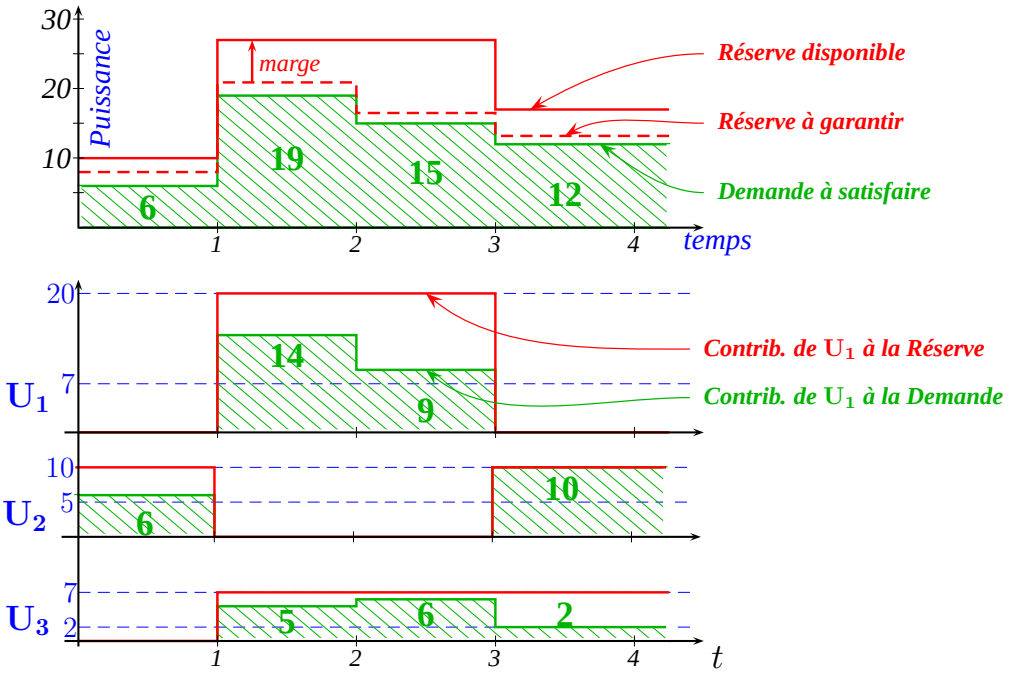

Figure 4 *exemple de satisfaction d'une demande par trois centrales*

La figure 4 illustre l'influence de la contrainte de réserve sur l'allumage des unités. On observe sur cet exemple que la demande de 19 MW à la date $t = 1$ (courbe hachurée du haut) peut être satisfaite

par l'allumage de u_1 à elle seule. Cependant la réserve de 2 MW à garantir pour cette date oblige à allumer également u_3, ce qui conduit à une solution plus chère.

De plus, on imagine facilement que les contraintes dynamiques sur les temps minimum d'allumage ou d'extinction obligent également à maintenir allumée u_3 à la date précédente, ce qui entrainerait un accroissement encore plus important du coût.

---

3 Formalisation du problème

3.1 Notations et variables utilisées

Les principaux **indices** utilisés sont :

- $t \in D = \{0, ..., t_{\max}\}$ qui représente une date : nombre d'heures ou de jours depuis le début de période. La valeur $t = 0$ représente l'instant initial pour lequel l'état du système est connu. Nous noterons également $D^+ = D \backslash \{0\} = \{1, ..., t_{\max}\}$;
- $u \in U = \{1, ..., u_{\max}\}$ qui identifie une unité (variant de 1 à $u_{\max} = |U|$) ;

Les **données globales** pour chaque instance[2] de problème sont définies pour toute date t comme suit :

- $d(t)$: demande en énergie à satisfaire pour toute date $t \in D^+$;
- $d_{\max}(t)$: demande maximale d'énergie à garantir, compte tenu des centrales allumées à la date t. Elle tient compte d'une réserve imposée qui est de l'ordre de 10% de la demande nominale.

Chaque instance définit également les **caractéristiques de chaque unité** comme suit :

- $p_{\min u}$: puissance fournie minimale (si l'unité est allumée !),
- $p_{\max u}$: puissance fournie maximale,
- t_{up_u} : durée minimale d'allumage,
- $t_{\mathrm{down}u}$: durée minimale d'extinction,
- $c_{\mathrm{hs}u}$: (*Cost Hot Start*) : coût de démarrage à chaud,
- $c_{\mathrm{cs}u}$: (*Cost Cold Start*) : coût de démarrage à froid,
- $t_{\mathrm{cs}u}$: (*Time Cold Start*) : durée de refroidissement,
- $c_{\mathrm{f}u}(p) = a_{0_u} + a_{1_u}.p + a_{2_u}.p^2$: coût de fonctionnement quadratique de l'unité u,
- $s_{\mathrm{i}u}$: état initial (e.g -4 si l'unité est arrêtée depuis 4 pas de temps).

Les **variables de décision** à déterminer sont :

- $y_u(t) \in \{0, 1\}$: indicateur de fonctionnement de l'unité u à la date t qui vaut 1 si l'unité u est active à la date t et 0 sinon. Nous noterons $\overline{y_u}(t) = 1 - y_u(t)$ pour l'indicateur complémentaire à $y_u(t)$;
- $p_u(t) \in \mathbb{R}^+$: puissance produite par l'unité u au temps t. Sa valeur est non significative si $y_u(t) = 0$.

Par ailleurs, nous utiliserons également les **variables auxiliaires** suivantes :

- $s_u(t) \in \mathbb{Z}^*$: variable d'état de l'unité u à la date t associée à la commande $y_u(t)$. La valeur d'état représente le nombre de pas de temps depuis lequel l'unité est allumée ($s_u(t) > 0$) ou éteinte ($s_u(t) < 0$). Cette valeur n'est jamais nulle car l'unité est soit allumée soit éteinte, et elle est comprise entre t_{up_u} et $t_{\mathrm{cs}u}$ (voir figure **6** page **9**). La valeur initiale $s_u(0)$ est la donnée $s_{\mathrm{i}u}$ du problème ;
- $y_u \in \{0, 1\}^{t_{\max}}$: vecteur indicateur de fonctionnement de l'unité u pour toute date ;
- $y(t) \in \{0, 1\}^{|U|}$: vecteur indicateur de fonctionnement pour toute unité à la date t ;
- ON(t) (respectivement OF(t)) : ensemble des unités allumées (respectivement éteintes) à la date t.

[2] La terminologie suivante sera utilisée : on désigne par ***instance*** d'un ***problème*** la donnée particulière d'un ***problème*** défini mathématiquement. Ainsi, cette section formalise un ***problème*** précis que nous testerons à l'aide des ***instances*** décrites en annexe. Cette terminologie est issue de la théorie de la complexité dans laquelle les ***problèmes*** sont des variables dont les valeurs sont les ***instances***.

3.2 Les contraintes

Les contraintes sont composées d'une part des deux contraintes globales de demande (C1) et de réserve (C2) liées au besoin, et d'autre part de contraintes techniques liées aux caractéristiques individuelles de chaque unité, à savoir : ses limites de production (C3) et l'inertie des changements d'état pour l'allumage (C4) ou pour l'extinction (C5).

C1	demande à satisfaire :	$\forall t \in D^+, \quad \sum_{u \in U} y_u(t).p_u(t) = d(t)$
C2	réserve à garantir :	$\forall t \in D^+, \quad \sum_{u \in U} y_u(t).p_{\max u} \geq d_{\max}(t)$
C3	puissance bornée :	$\forall t \in D^+, \ \forall u \in U, \quad y_u(t).p_{\min u} \leq p_u(t) \leq y_u(t).p_{\max u}$
C4	temps mini $t_{\text{down}u}$:	$\forall t \in D^+, \ \forall u \in U, \quad \overline{y_u}(t-1).y_u(t) = 1 \Rightarrow -s_u(t-1) \geq t_{\text{down}u}$
C5	temps mini $t_{\text{up}u}$:	$\forall t \in D^+, \ \forall u \in U, \quad y_u(t-1).\overline{y_u}(t) = 1 \Rightarrow s_u(t-1) \geq t_{\text{up}u}$

3.3 Fonction objectif

Le problème consiste à fixer pour toute unité et pour toute date les valeurs de $p_u(t)$ et de $y_u(t)$ de façon à minimiser le coût total de production du parc de centrales avec :

$$(P) \ : \begin{cases} \min\limits_{p,y} f(p,y) = \sum\limits_{u \in U} \sum\limits_{t \in D^+} c_{\text{p}u}(t) = \sum\limits_{u \in U} \sum\limits_{t \in D^+} \Big(\underbrace{y_u(t).c_{\text{f}u}(p_u(t))}_{\textit{fonctionnement}} + \underbrace{c_{\text{s}u}(t, y_u)}_{\textit{demarrage}} \Big) \\ \\ \text{sous les contraintes C1 à C5} \end{cases} \tag{3}$$

avec :

- $c_{\text{p}u}(t)$: coût de production total (fonctionnement et démarrage) de l'unité u à la date t ;
- $c_{\text{f}u}(p)$ (pour mémoire) : coût de fonctionnement de l'unité u pour produire la puissance p. La valeur de cette fonction n'est significative que si $y_u(t) = 1$;
- $c_{\text{s}u}(t, y_u)$ (pour mémoire) : coût de démarrage de l'unité u à la date t. Il dépend des états d'allumage de cette unité aux dates antérieures.

---

4 Approche duale

4.1 Choix de la relaxation

Seules les contraintes ***C1 (Demande)*** et ***C2 (Réserve)*** couplent les unités entre elles ; le principe retenu consiste donc à traiter chaque unité séparément en :

- dualisant les contraintes couplantes (méthode décrite dans [3]),
- résolvant le problème relâché par **Programmation Dynamique** [4].

Les multiplicateurs lagrangiens associés aux contraintes C1 et C2 sont respectivement notés : $\mu(t) \in \mathbb{R}$ et $\lambda(t) \in \mathbb{R}^+$.

La ***fonction de Lagrange*** correspondante est alors la suivante :

$$\begin{aligned} L(p, y, \mu, \lambda) \quad = \quad f(p,y) \quad & + \sum_{t \in D^+} \mu(t) \Big[\sum_{u \in U} \big(y_u(t).p_u(t) \big) \quad - \quad d(t) \Big] \\ & + \sum_{t \in D^+} \lambda(t) \Big[d_{\max}(t) \quad - \quad \sum_{u \in U} y_u(t).p_{\max u} \Big] \end{aligned} \tag{4}$$

d'où :

$$\boxed{\begin{aligned} L(p,y,\mu,\lambda) = &\sum_{u\in U}\sum_{t\in D^+}\Big(y_u(t).\big(c_{\mathrm{f}u}(p_u(t)) + \mu(t).p_u(t)\big) + c_{\mathrm{s}u}(t,y_u) - \lambda(t).y_u(t).p_{\max u}\Big) \\ &- \sum_{t\in D^+}\big(\mu(t).d(t) - \lambda(t).d_{\max}(t)\big) \end{aligned}} \tag{5}$$

Le ***problème relâché*** consiste à minimiser pour λ et μ fixés, la fonction de Lagrange $L(p,y,\mu,\lambda)$ en respectant les contraintes non dualisées C3, C4 et C5 qui régissent le fonctionnement de chaque générateur. On obtient alors la ***fonction duale de Lagrange*** $\omega(\mu,\lambda)$:

$$\boxed{(P_{relax}) \ : \omega(\mu,\lambda) = \begin{cases} \min\limits_{p,y} \ L(p,y,\mu,\lambda) \\ \\ \text{sous les contraintes C3, C4 et C5} \end{cases}} \tag{6}$$

Une propriété de la relaxation lagrangienne [5] est que pour toute valeur des multiplicateurs de Lagrange (μ,λ), la fonction duale de Lagrange est une borne inférieure du problème initial (P). Trouver les valeurs de (μ,λ) qui maximisent cette borne inférieure conduit à résoudre le problème dual suivant :

$$\boxed{(P_{dual}) \ : \begin{cases} \max\limits_{\mu,\lambda} \ \omega(\mu,\lambda) \\ \\ \text{sous les contraintes : } \forall t \in \{1,...,t_{\max}\}, \ \lambda(t) \geq 0 \end{cases}} \tag{7}$$

4.2 Principe de résolution de la fonction de Lagrange

Nous résolvons ce problème dual par une méthode de sous-gradient qui implique de nombreuses minimisations du problème relâché. Celui-ci doit donc être traitable efficacement. Le choix d'avoir dualisé les contraintes couplantes fait que, dans l'expression (**5**) de la fonction de Lagrange, la minimisation se fait indépendamment pour chaque unité u :

$$c(p_u,y_u) = \sum_{t\in D+}\Big[\underbrace{y_u(t).\big(c_{\mathrm{f}u}(p_u(t)) + \mu(t).p_u(t)\big)}_{\alpha} - \underbrace{\lambda(t).y_u(t).p_{\max u}}_{\beta} + \underbrace{c_{\mathrm{s}u}(t,y_u)}_{\gamma}\Big] \tag{8}$$

$c(p_u,y_u)$ représente le coût total (perturbé) de l'unité u pendant toute la période d'exploitation considérée.

Pour chaque unité u, et pour les paramètres $\lambda(t)$ et $\mu(t)$ fixés, il s'agit de trouver pour chaque date la valeur de $p_u(t)$ et de $y_u(t)$ qui minimise l'expression de $c(p_u,y_u)$ dont une composante est non linéaire (quadratique).

4.3 Précalcul des productions optimales

Nous observons que l'expression $c(p_u,y_u)$ peut être décomposée en trois termes α, β et γ indépendants dans une certaine mesure. En effet, deux cas se présentent :

- si l'unité u est éteinte à la date t (soit $y_u(t) = 0$), alors chaque membre de cette expression est nul ;
- sinon seul le terme α dépend de la puissance $p_u(t)$ produite par l'unité. Par ailleurs, $\mu(t)$ étant fixé, ce terme est totalement indépendant du passé de l'unité : seul $y_u(t)$ intervient, et non pas y_u.

Par conséquent, on peut se permettre de précalculer pour toute valeur de t la puissance $p_u(t)$ qui minimise le terme $(c_{f_u}(p_u(t))+ \mu(t).p_u(t))$.

Comme l'illustre la figure 5, la convexité du coût de fonctionnement assure l'unicité de cette puissance optimale.

De plus, la forme parabolique du coût de fonctionnement rend possible le calcul analytique de la puissance optimale $\widetilde{p_u(t)}$. En effet, si la puissance fournie par l'unité u est dans son domaine de fonctionnement, on a :

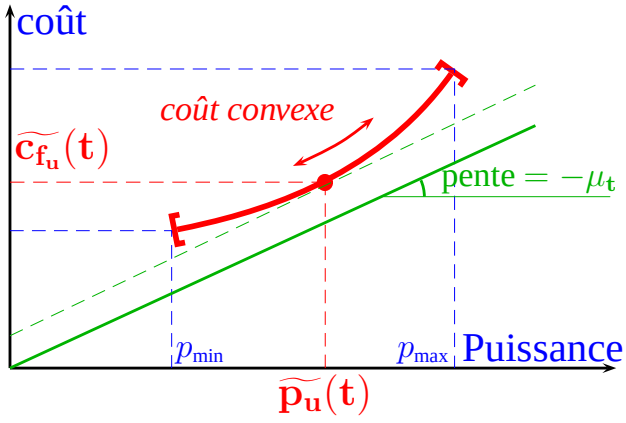

Figure 5 *précalcul du coût de production optimum* $\widetilde{c_{fu(t)}}$

$$\alpha = y_u(t).\Big(c_{f_u}\big(p_u(t)\big) + \mu(t).p_u(t)\Big) = y_u(t).\Big(a_{0_u} + a_{1_u}.p_u(t) + a_{2_u}.p_u^2(t) \; + \; \mu(t).p_u(t)\Big). \tag{9}$$

L'unité étant allumée, $y_u(t) = 1$, ce terme est donc minimum pour :

$$\frac{d\alpha}{dp_u(t)} = 2a_{2_u}p_u(t) + a_{1_u} + \mu(t) = 0 \quad \Rightarrow \quad p_u(t) = -\frac{a_{1_u} + \mu(t)}{2a_{2_u}}.$$

Mais, la puissance d'une unité en marche est bornée entre $p_{\min}$ et $p_{\max}$; nous obtenons alors la puissance optimale fournie par une unité u en marche :

$$\boxed{\widetilde{p_u(t)} = \min\left(p_{\max u}, \max\left(p_{\min u}, -\frac{a_{1_u} + \mu(t)}{2a_{2_u}}\right)\right)} \tag{10}$$

ce qui donne pour la partie α :

$$\alpha(\mu(t)) = \begin{cases} c_{f_u}(\widetilde{p_u(t)}) + \mu(t).\widetilde{p_u(t)} & \text{si} \quad y_u(t) = 1 \\ 0 & \text{sinon} \end{cases}. \tag{11}$$

Enfin, comme la partie α, la partie β de l'équation (8) n'existe que si l'unité est allumée à la date t. Le coût formé de $\alpha - \beta$ sera alors le coût de fonctionnement d'une unité active dans le problème perturbé.

Interprétation économique des multiplicateurs de Lagrange

Le multiplicateur $\mu(t)$ (réel) lié à une contrainte d'égalité de la demande, est commun à toutes les unités. Il assure la **coordination par les prix** de celles-ci en les « motivant » à produire plus ou moins selon leurs caratéristiques économiques propres de façon à satisfaire la demande à la date t (voir [6]). En ignorant la contrainte de réserve, on peut interpréter $\mu(t)$ comme un prix du KWh de la puissance fournie, et chaque centrale minimise son coût d'utilisation en fonction de ce prix. On peut donc voir la résolution du problème dual comme la recherche d'un prix au KWh $\mu(t)$ conduisant à la meilleure coordination des puissances $p_u(t)$ produites par les acteurs $y_u(t)$ allumés.

Le multiplicateur $\lambda(t)$ (≥ 0 car lié à une contrainte d'inégalité) représente un prix du KWh de réserve. Plus précisément, une grande valeur de $\lambda(t)$ incite les unités à l'allumage et donc indirectement (la production totale étant imposée par $d(t)$), à la sous-production de chaque centrale allumée, contribuant ainsi à assurerla marge $d_{\max}(t) - d(t)$ de puissance demandée.

4.4 Programmation dynamique

Dans l'équation (8), la partie γ représente le coût de démarrage et dépend de la durée d'arrêt de la centrale ainsi que des caractéristiques dynamiques propres à chacune d'elle (cf figure 2 page 3).

On définit l'état d'une unité par un entier $s_u(t)$ qui représente la durée depuis laquelle l'unité est allumée (si $s_u(t) > 0$) ou éteinte (si $s_u(t) < 0$). La date $t = 0$ correspond à l'état initial $s_{\mathrm{i}u}$ du générateur et est une donnée du problème. Les valeurs de ces états sont bornées par les valeurs extrêmes $t_{\mathrm{up}u}$ et $t_{\mathrm{cs}u}$ au delà desquelles le maintien à l'état allumé (pour $t_{\mathrm{up}u}$) ou éteint (pour $t_{\mathrm{cs}u}$) n'a plus d'influence sur les possibilités d'évolution de l'unité.

Par exemple sur la figure 6, que l'unité soit allumée depuis 7 heures ou 5 heures, elle sera toujours dans l'état +4 car $t_{\mathrm{up}u} = 4$.

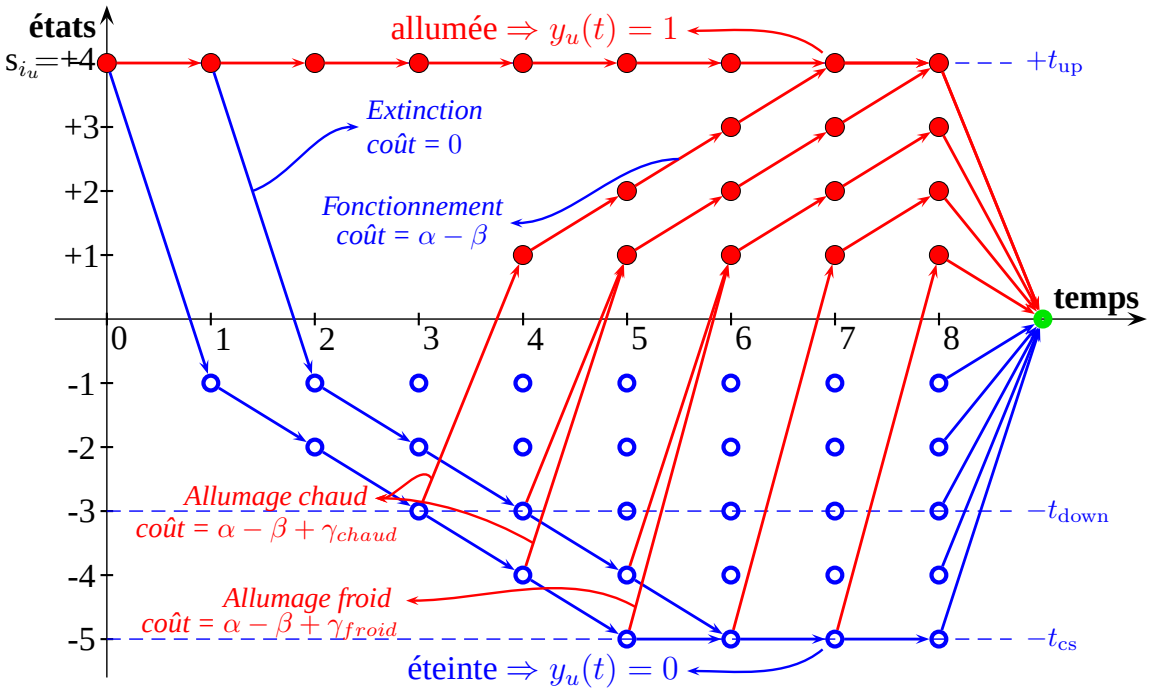

Figure 6 *graphe partiel représentant l'évolution autorisée des états d'une unité*

La **caractéristique dynamique** de chaque unité (voir figure 2) peut alors être représentée sur un graphe décrivant l'évolution possible de l'état d'une unité en fonction du temps. La pondération du graphe est effectuée de la manière suivante :

- Les arcs conduisant à un état négatif ont un coût nul,
- Les arcs conduisant à un état allumé ont un coût formé de trois composantes :
  - d'une part le coût correspondant à la production optimale précalculée (partie α en (11)),
  - d'autre part la composante négative $-\beta$ liée à la contrainte de réserve,
  - enfin un coût de démarrage non nul seulement pour un arc conduisant d'un état négatif vers un état positif et qui vaut alors :
    - $c_{\mathrm{cs}u}$ si l'état source avait atteint la limite de refroidissement $-t_{\mathrm{cs}u}$,
    - $c_{\mathrm{hs}u}$ si l'état source n'avait pas encore atteint la limite de refroidissement $-t_{\mathrm{cs}u}$.

En résumé la résolution pratique du problème perturbé s'effectue en deux phases :

- précalcul du coût de production optimal et pondération du graphe de chaque unité,
- résolution du problème de plus court chemin sur le graphe d'état de chaque unité (programmation dynamique).

5 Les coûts réduits

5.1 Description de notre démarche

Nous venons donc de réaliser une relaxation lagrangienne et nous disposons d'une solution duale (non réalisable) avec son coût associé. L'exploitation des résultats d'une relaxation se fait classiquement de deux manières différentes :

- ***exploitation de résultats partiels*** localisés dans les solutions duales en les rendant réalisables (*réparation* ou *primalisation par heuristique lagrangienne*) (section **6** page **15**),
- ***exploitation de l'information globale*** que représente la valeur de la *borne inférieure* pour élaguer les branches d'un arbre de recherche (*Branch and Bound*) (section **7.1** page **17**),

Notre démarche va consister à tenter de tirer partie à la fois de l'information globale fournie par la borne inférieure et d'informations locales fournies par les solutions duales, à travers les coûts réduits[3] des variables. Outre leurs utilisations par notre heuristique de primalisation (section **6**), nous verrons deux façons originales (globale et locale) d'utiliser ces coûts réduits en couplant la méthode de *Relaxation Lagrangienne* à la technique de *Programmation Par Contraintes*, au sein d'un *Branch and Bound* tronqué. Nous allons donc commencer par examiner comment nous déterminons et exploitons les coûts réduits.

5.2 Coût réduit d'une seule unité

Dans les graphes d'états des unités, la solution proposée par la relaxation lagrangienne, est optimale pour le problème perturbé, mais représente une solution non réalisable de notre problème inital car certaines contraintes dualisées sont violées.

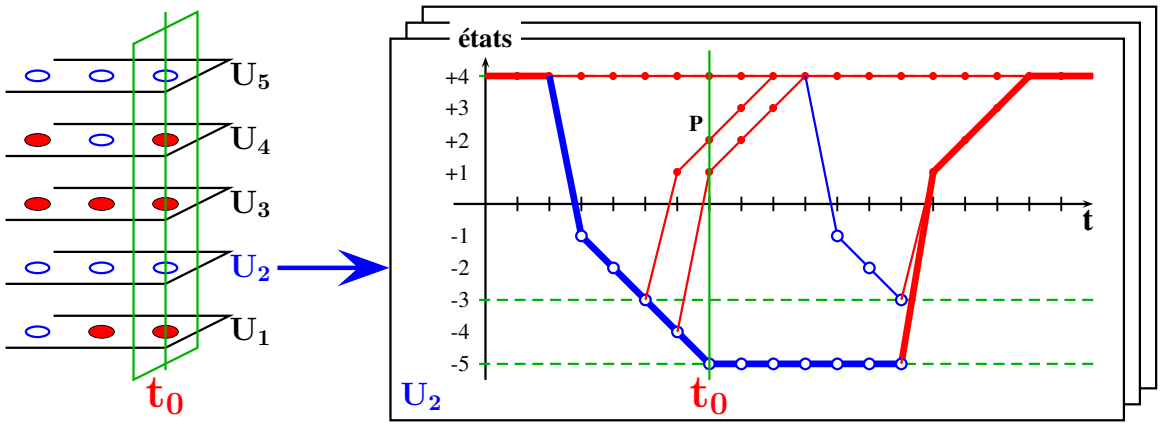

Figure 7 *coût réduit d'allumage de l'unité u_2. La séquence des états successifs de l'unité u_2 est représentée en gras sur son graphe d'états à droite. On souhaiterait la forcer à passer par un des états allumés à t_0, par exemple P.*

À titre d'exemple, la figure **7** illustre l'état des unités génératrices à une itération k donnée de la relaxation lagrangienne. Sur la partie gauche de la figure est représenté l'état allumé ou éteint de l'ensemble des unités pour toute date. On y observe qu'à la date t_0, trois des cinq unités sont allumées (en rouge-plein). La partie droite détaille le graphe de l'unité u_2 seule. La séquence proposée par la relaxation pour l'unité u_2 est représentée par le chemin du bas pour lequel l'unité est éteinte à la date t_0. Tout autre chemin dans ce graphe aura nécessairement un coût supérieur ou égal.

La possibilité d'ajuster *a posteriori* les puissances pour satisfaire la contrainte de demande C1 fait que les viols des solutions duales sont essentiellement dûs aux viols de la contrainte de réserve

[3] Dans ce rapport, le terme « coût réduit » désigne un coût différentiel, ou de « regret », pour passer d'une solution à une autre par le changement de valeur d'une seule variable.

C2 ; nous nous intéresserons donc exclusivement à la contrainte de réserve. Dans le but de rendre la solution duale réalisable, nous souhaiterions forcer l'allumage d'une unité éteinte à cette date, par exemple u_2. Une première question nous vient alors :

quel coût supplémentaire doit-on payer pour que le chemin de l'unité u_2 passe par un des états allumés à la date t_0 ?

D'où la définition suivante :

Définition 1 *Le coût réduit $rcost_u(t)$ est le coût minimum qu'il faut payer pour forcer le changement d'état[4] de l'unité u à la date t.*

Nous pouvons remarquer que, même s'il s'agît d'imposer une extinction de l'unité u à la date t, ce forçage sera toujours coûteux et l'on a : $\forall u \in U, \forall t \in \{1, ..., t_{\max}\}$: $rcost_u(t) \geq 0$.

Calcul de $rcost_u(t)$

En pratique, l'algorithme de Bellman nous donne, en plus du plus court chemin de l'état initial à l'état final, le coût absolu (valeur de Bellman) pour atteindre l'état final depuis n'importe quel nœud du graphe [4]. Nous appellerons cette phase : l'application de l'algorithme de Bellman en temps direct. Réciproquement, l'application de l'algorithme en temps rétrograde nous donne pour tout nœud, le coût du plus court chemin le joignant au nœud initial. Deux passes de programmation dynamique, l'une en temps direct et l'autre en temps rétrograde nous permettent donc, par la somme des deux chemins partiels précédents, d'obtenir le coût minimum du chemin passant par un état arbitraire. Il sera nécessairement au moins aussi élevé que celui proposé par la relaxation. Le *coût réduit* $rcost_u^{(s_u)}(t)$ d'un nœud associé à un état quelconque $s_u \in \{-t_{\text{cs}u}, ..., t_{\text{up}_u}\} \backslash 0$ s'obtient alors par différence du coût précédent avec le *coût absolu* du chemin optimum donné par la relaxation.

Finalement, nous obtenons le coût réduit $rcost_u(t)$ en calculant le coût réduit pour chaque état complémentaire à celui proposé par la relaxation (e.g. chemins passant par les états $s_u(t)$ positifs si on avait $y_u(t) = 0$), et en retenant le moins coûteux d'entre eux, soit :

$$rcost_u(t) = \begin{cases} \min\limits_{s_u \in \{1,...,t_{\text{up}_u}\}} rcost_u^{(s_u)}(t) & \text{si } y_u(t) = 0 \\ \min\limits_{s_u \in \{-t_{\text{cs}u},...,-1\}} rcost_u^{(s_u)}(t) & \text{si } y_u(t) = 1 \end{cases} \tag{12}$$

5.2.1 Notations supplémentaires basées sur les chemins

Une approche plus combinatoire dont nous aurons besoin par la suite, consiste à représenter une solution du problème relaché (P_{relax}) en (6) page 7, comme un ensemble de $|U|$ chemins $\mathcal{P}_u$ dans les graphes d'états des unités. À chaque unité u est associée un chemin $\mathcal{P}_u$ d'un certain coût $\mathcal{C}(\mathcal{P}_u) = \mathcal{C}_u$, et le coût total est la somme des coûts des chemins de chaque unité. Nous appelons $\mathcal{P}\text{off}_u(t)$ de coût $\mathcal{C}(\mathcal{P}\text{off}_u) = \mathcal{C}\text{off}_u(t)$ un plus court chemin, c'est-à-dire un des minorants parmi les chemins de l'unité u qui passe par un des états de type « éteint » à la date t (i.e. tels que $s_u(t) < 0$). De même, $\mathcal{P}\text{on}_u(t)$ de coût $\mathcal{C}(\mathcal{P}\text{on}_u) = \mathcal{C}\text{on}_u(t)$ sera un des plus courts chemins de l'unité u qui passe par un des états de type « allumé » à cette date. La valeur de $y_u(t)$ étant une décision optimale pour l'unité u à la date t, le chemin $\mathcal{P}_u$ associé à u dans la solution de (P_{relax}) est telle que :

$$\begin{aligned} \mathcal{C}(\mathcal{P}_u) &= \min\left(\mathcal{C}\text{on}_u(t), \mathcal{C}\text{off}_u(t)\right) \\ &= y_u(t).\mathcal{C}\text{on}_u(t) + \overline{y_u}(t).\mathcal{C}\text{off}_u(t), \end{aligned} \tag{13}$$

et le coût réduit $rcost_u(t)$ peut alors simplement s'exprimer par :

[4] Le coût réduit est bien défini pour un changement d'état (de 0 vers 1, ou de 1 vers 0) et non pas seulement pour un passage à l'état 1. En effet, sur le graphe d'état perturbé par les multiplicateurs lagrangiens, le chemin de l'unité u peut s'avérer moins coûteux en passant par un état allumé à une date t donnée ; et on s'intéresse alors au coût réduit nécessaire pour forcer u à passer par l'un des états éteints en cette date t.

$$\begin{aligned} rcost_u(t) &= |\mathcal{C}\text{on}_u(t) - \mathcal{C}\text{off}_u(t)| \\ &= y_u(t).\big(\mathcal{C}\text{off}_u(t) - \mathcal{C}\text{on}_u(t)\big) + \overline{y_u}(t).\big(\mathcal{C}\text{on}_u(t) - \mathcal{C}\text{off}_u(t)\big). \end{aligned} \tag{14}$$

Des chemins différents pouvant avoir un coût identique, il peut exister plusieurs chemins $\mathcal{P}_u$ associés à u dans les solutions du problème relaché (P_{relax}). Cependant, tous ces chemins sont équivalents du point de vue du problème (P_{relax}), et donc interchangeables. Les chemins $\mathcal{P}\text{on}_u(t)$ et $\mathcal{P}\text{off}_u(t)$ étant connus, on peut alors considérer que l'indicateur d'allumage $y_u(t)$ est également un sélecteur de chemin tel que :

$$\mathcal{P}_u = \begin{cases} \mathcal{P}\text{on}_u(t) \text{ si } y_u(t) = 1 \\ \mathcal{P}\text{off}_u(t) \text{ si } y_u(t) = 0 \end{cases} \tag{15}$$

5.3 Coût réduit global

Imaginons maintenant dans l'exemple de la figure 7, que pour la date $t = t_0$ la solution duale courante (pour laquelle $\{u_1, u_3, u_4\}$ sont allumées) viole la contrainte C2 de réserve. Le déficit $p_{\text{viol}}(t)$ vaut alors :

$$p_{\text{viol}}(t) \;=\; d_{\max}(t) - \sum_{u \in \text{ON}(t)} p_{\max u} \;=\; d_{\max}(t) - \sum_{u \in U} y_u(t).p_{\max u} \quad . \tag{16}$$

L'interprétation du coût réduit de l'unité u_2 de la figure 7 est qu'avec un coût $rcost_{u_2}(t_0)$, l'unité u_2 peut contribuer à la réduction du viol $p_{\text{viol}}(t_0)$ pour un montant de $p_{\max u_2}$. La question qui vient alors naturellement est :

quel coût supplémentaire doit-on payer pour combler totalement le déficit de la contrainte de réserve à cette date ?

D'où la seconde définition :

Définition 2 *Le coût réduit rcost(t) est le coût minimum qu'il faut payer pour combler totalement le déficit du viol de la contrainte de réserve à la date t.*

Calcul de *rcost*(t)

Pour modéliser le problème du calcul de $rcost(t)$, nous allons repartir de la définition 2 ci-dessus. En utilisant les notations introduites en 5.2.1, la recherche de $rcost(t)$ revient pour chaque unité u à choisir entre :

- conserver le chemin $\mathcal{P}_u$ de coût $\mathcal{C}(\mathcal{P}_u)$ correspondant à la solution du problème relaché (P_{relax}),
- inverser l'allumage de u en t pour un coût de $rcost_u(t)$ en forçant son chemin à :
  - $\mathcal{P}\text{on}_u(t)$, ce qui nous apporte une capacité de production supplémentaire de $p_{\max u}$,
  - $\mathcal{P}\text{off}_u(t)$ ce qui augmente de déficit de la réserve de $p_{\max u}$.

Nous introduisons donc les variables $e_u(t)$ dites « d'écart » par rapport à l'indicateur d'allumage $y_u(t)$ associé à la solution de (P_{relax}) :

$$\mathbf{e_u(t)} = \begin{cases} +1 \text{ si l'on force l'allumage de } u \text{ à la date } t \text{ en choisissant le chemin } \mathcal{P}\text{on}_u(t), \\ \;\;0 \text{ si l'on conserve le chemin } \mathcal{P}_u \in \{\mathcal{P}\text{off}_u(t), \mathcal{P}\text{on}_u(t)\}, \\ -1 \text{ si l'on force l'extinction de } u \text{ à la date } t \text{ en choisissant le chemin } \mathcal{P}\text{off}_u(t), \end{cases} \tag{17}$$

La solution du problème (P_{relax}) est alors représentée par le vecteur nul $e(t) = 0$. En utilisant les variables d'écart $e_u(t)$, le problème du calcul de $rcost(t)$ devient alors :

$$(\text{Rcost}_t) \;:\; \begin{cases} \min\limits_{e(t) \in \{-1,0,1\}^{|U|}} & \sum\limits_{u \in U} |e_u(t)|.rcost_u(t) \\ s.c. & \sum\limits_{u \in U} (y_u(t) + e_u(t)).p_{\max u} \geq \; d_{\max}(t) \end{cases} \tag{18}$$

Si l'on observe que :

- le critère à minimiser est une somme de termes positifs ou nuls,
- le forçage à l'extinction d'une unité u ($y_u(t) = 1$ et $e_u(t) = -1$) conduit à détériorer la réserve de puissance imposée par la contrainte d'une valeur de $p_{\max u}$;

on en déduit que la solution au problème (Rcost_t) ne peut pas contenir d'élément $e_u(t)$ négatif, et l'on a donc toujours intérêt à conserver le chemin initial $\mathcal{P}_u$ d'une unité allumée. Par conséquent, le problème (Rcost_t) se réduit à chercher parmi les unités éteintes à la date t celles qui doivent être forcées à l'allumage. En tenant compte de l'expression **16** du déficit $p_{\text{viol}}(t)$ de la réserve, nous pouvons reformuler le problème (Rcost_t) comme suit :

$$(\text{Rcost}_t) \ : \begin{cases} \min\limits_{e \in \{0,1\}^{|\text{OF}(t)|}} \sum\limits_{u \in \text{OF}(t)} e_u(t).rcost_u(t) \\ s.c. \quad \sum\limits_{u \in \text{OF}(t)} e_u(t).p_{\max u} \ \geq \ p_{\text{viol}}(t) \\ \qquad \forall u \in \text{OF}(t), \quad e_u(t) \in \{0,1\}. \end{cases} \tag{19}$$

Ce problème linéaire en nombre entier est de type ***sac à dos*** inversé dans lequel il s'agit de minimiser le coût d'allumage d'unités supplémentaires tout en ***dépassant*** la capacité du sac représentée par le montant du viol à combler. Puisque nous nous intéressons plus particulièrement à une borne inférieure de notre problème initial, l'***approximation continue*** du problème de sac à dos est utilisable : les unités sont alors classées par ordre de rentabilité décroissante (rapport $p_{\max u}/rcost_u(t)$), et on allume progressivement les unités les plus rentables jusqu'à combler le viol. L'aspect *relaché continu* du sac dos se traduit par le fait que la dernière unité nécessaire pour combler le viol est allumée « partiellement » dans le sens où elle ne coûte que la fraction correspondant à sa contribution nécessaire pour combler ce viol[5].

Maintenant que nous savons calculer les coûts réduits, voyons de quelle manière nous allons les exploiter.

5.4 Exploitation des coûts réduits pour le dual

Nous savons qu'à une itération lagrangienne k donnée, nous résolvons le problème relaché (P_{relax}) en (**6**). Le coût fourni par la solution correspondante est une borne inférieure $\omega(\mu_k, \lambda_k)$ de notre problème, et ce quelle que soit la valeur des multiplicateurs de Lagrange $\mu_k(t) \in \mathbb{R}$ et $\lambda_k(t) \in \mathbb{R}^+$. Supposons de nouveau (figure **7** page **10**) qu'à une date donnée t, la contrainte de réserve ne soit pas satisfaite. La solution duale obtenue conduit donc à une solution dont la puissance totale disponible est insuffisante à la date t :

$$P_{\max}(t) = \sum_{u \in \text{ON}(t)} p_{\max u} = \sum_{u \in U} y_u(t).p_{\max u} \quad < \quad d_{\max}(t) \tag{20}$$

Supposons encore que nous ajoutions « en dur » au problème relaché (P_{relax}) page **7** la contrainte de réserve C2 pour la date t :

$$\textbf{C2(t) :} \qquad \sum_{u \in U} y_u(t).p_{\max u} \quad \geq \quad d_{\max}(t) \tag{21}$$

nous obtenons alors le problème « moins » relaché suivant :

$$(P_{relax \setminus t}) \ : \omega_t(\mu, \lambda) \quad = \begin{cases} \min\limits_{p,y} \ L(p, y, \mu, \lambda) \\ \text{s.c.} \quad \text{C3, C4, C5 et } \textbf{C2(t)} \end{cases} \tag{22}$$

[5] Le problème de sac à dos est cependant résolu de manière exacte dans certains cas spéciaux, comme par exemple lorsque l'allumage de n'importe quelle unité à elle seule suffit pour combler le viol (cas fréquent en pratique). L'unité la moins chère u' est alors allumée entièrement, et non pas une fraction de l'unité u la plus rentable : on paye donc $rcost_{u'}(t)$ au lieu d'une fraction de $\alpha \times rcost_u(t)$ (avec $\alpha < 1$).

Nous n'avons fait qu'ajouter une contrainte redondante au problème initial (P) en (3). De plus C2(t) est déjà prise en compte dans la fonction de Lagrange $L(p, y, \mu, \lambda)$ sous la forme d'une contrainte dualisée, associée au multiplicateur λ_t. La valeur $\omega_t(\mu, \lambda)$ obtenue en résolvant le problème $(P_{relax\setminus t})$ reste donc un minorant du problème initial, quelque soit la valeur des multiplicateurs μ et λ. Résoudre le problème $(P_{relax\setminus t})$ consiste alors à résoudre le problème (P_{relax}) sur un ensemble plus petit car ne contenant que les solutions de (P_{relax}) respectant la contrainte C2(t). Nous obtenons donc un moyen d'améliorer notre borne inférieure car :

$$\boxed{\omega(\mu, \lambda) \quad \leq \quad \omega_t(\mu, \lambda) \quad \leq \quad f^*(p_u, y_u)} \tag{23}$$

Le problème consiste maintenant à calculer la valeur $\omega_t(\mu, \lambda)$, solution du problème $(P_{relax\setminus t})$. Dans le reste de cette section, nous montrons que le coût réduit $rcost(t)$ précédement calculé peut directement servir dans le calcul de $\omega_t(\mu, \lambda)$.

Nous avons vu que le chemin $\mathcal{P}_u$ associé à l'unité u à la date t dans une solution de (P_{relax}) est nécessairement un des deux chemins $\mathcal{P}\text{on}_u(t)$ ou $\mathcal{P}\text{off}_u(t)$. Le choix du chemin $\mathcal{P}_u$ est directement donné par l'indicateur d'allumage $y_u(t)$, l'autre de ces chemins étant celui qui permet de calculer le coût réduit $rcost_u(t)$. Nous allons maintenant montrer que ces mêmes chemins interviennent également dans le problème $(P_{relax\setminus t})$.

Soit $\mathcal{P}(t)$ le produit cartésien de l'ensemble des paires de chemins localement optimaux pour chaque unité :

$$\mathcal{P}(t) = \prod_{u \in U} \{\mathcal{P}\text{off}_u(t), \mathcal{P}\text{on}_u(t)\} \tag{24}$$

Théorème 1 *Il existe une solution optimale pour $(P_{relax\setminus t})$ appartenant à $\mathcal{P}(t)$.*

Preuve :
Si une solution optimale de $(P_{relax\setminus t})$ utilise pour l'unité u un chemin π_u n'appartenant pas à $\{\mathcal{P}\text{off}_u(t), \mathcal{P}\text{on}_u(t)\}$, et pour lequel $y_u(t)$ vaut 1 (respectivement 0), alors on peut toujours remplacer ce chemin π_u par $\mathcal{P}\text{on}_u(t)$ (respectivement $\mathcal{P}\text{off}_u(t)$) en préservant la faisabilité de la containte de réserve C2(t) (laquelle ne dépend que de l'indicateur allumé ou éteint $y_u(t)$). Or, par définition même de ces deux chemins, ce remplacement ne peut pas augmenter la valeur de $\omega_t(\mu, \lambda)$; donc cette nouvelle solution utilisant $\mathcal{P}\text{on}_u(t)$ est également optimale. □

Une solution aux problèmes (P_{relax}) ou $(P_{relax\setminus t})$ peut donc être représentée par un vecteur $x(t) \in \{0, 1\}^{|U|}$ dont les éléments sont :

$$x_u(t) = \begin{cases} 1 \text{ si le chemin } \mathcal{P}\text{on}_u(t) \\ 0 \text{ si le chemin } \mathcal{P}\text{off}_u(t) \end{cases} \quad \text{est dans la solution de } (P_{relax}).$$

Le problème (P_{relax}) peut se reformuler comme suit :

$$(P_{relax}) \ : \omega(\mu, \lambda) = \min_{x(t) \in \{0,1\}^{|U|}} \sum_{u \in U} \Big(x_u(t) \ . \ \mathcal{C}\text{on}_u(t) + (1 - x_u(t)) \ . \ \mathcal{C}\text{off}_u(t) \Big) \ . \tag{25}$$

Sa solution est le vecteur indicateur d'allumage $y_u(t)$, et son coût $\omega(\mu, \lambda)$ s'exprime par :

$$\omega(\mu, \lambda) = \sum_{u \in U} \Big(y_u(t). \ \mathcal{C}\text{on}_u(t) + (1 - y_u(t)) \ . \ \mathcal{C}\text{off}_u(t) \Big) \ . \tag{26}$$

Le problème $(P_{relax\setminus t})$ devient alors :

$$(P_{relax\setminus t}) \ : \omega_t(\mu, \lambda) \begin{cases} \min\limits_{x(t) \in \{0,1\}^{|U|}} \sum\limits_{u \in U} \Big(x_u(t) \ . \ \mathcal{C}\text{on}_u(t) + (1 - x_u(t)) \ . \ \mathcal{C}\text{off}_u(t) \Big) \\ s.c. \qquad \sum\limits_{u \in U} x_u(t).p_{\max_u} \geq d_{\max}(t) \\ \forall u \in U, \quad x_u(t) \in \{0, 1\} \end{cases} \tag{27}$$

La solution de $(P_{relax\setminus t})$ sera un vecteur $x(t)$, généralement différent de $y(t)$, dont il s'agît maintenant de détermininer la valeur $\omega_t(\mu, \lambda)$.

Théorème 2 $\omega_t(\mu,\lambda) = \omega(\mu,\lambda) + rcost(t)$

Preuve :

Nous allons réutiliser les variables d'écart e définies en (17) page 12 avec :

$$e_u(t) = x_u(t) - y_u(t) = \begin{cases} +1 \text{ si } x_u(t) > y_u(t) & : \text{ force l'allumage en imposant le chemin } \mathcal{P}_{on_u}(t) \text{ ,} \\ 0 \text{ si } x_u(t) = y_u(t) & : \text{ conserve le chemin } \mathcal{P}_u \text{ de } u, \\ -1 \text{ si } x_u(t) < y_u(t) & : \text{ force l'extinction en imposant le chemin } \mathcal{P}_{off_u}(t) \text{ .} \end{cases} \quad (28)$$

La solution du problème (P_{relax}) est alors représentée par le vecteur nul $e(t) = 0$. Nous allons développer le coût $(\mathcal{C}'_u)$ par unité dans le terme à minimiser dans la formulation (27) de $P_{relax\setminus t}$.

$$\begin{aligned} \mathcal{C}'_u &= x_u(t).\mathcal{C}on_u(t) + (1 - x_u(t)).\mathcal{C}off_u(t) \\ &= (e_u(t) + y_u(t)).\mathcal{C}on_u(t) + (1 - e_u(t) - y_u(t)).\mathcal{C}off_u(t) \\ &= e_u(t).\big(\mathcal{C}on_u(t) - \mathcal{C}off_u(t)\big) + \big(y_u(t).\mathcal{C}on_u(t) + (1 - y_u(t)).\mathcal{C}off_u(t)\big) \quad (29) \\ &\quad \text{voir (14) page 12} \qquad \text{et (26) page 14} \\ &= |e_u(t)|.rcost_u(t) + \omega(\mu,\lambda) \end{aligned}$$

Le problème $(P_{relax\setminus t})$ en (27) devient alors :

$$(P_{relax\setminus t}) \ : \omega_t(\mu,\lambda) = \omega(\mu,\lambda) + \begin{cases} \min\limits_{e\in\{-1,0,1\}^{|U|}} & \sum\limits_{u\in U} |e_u(t)|.rcost_u(t) \\ s.c. & \sum\limits_{u\in U} (y_u(t) + e_u(t)).p_{max_u} \geq d_{max}(t) \\ & \forall u \in U, \quad e_u(t) \in \{-1,0,1\}. \end{cases} \quad (30)$$

qui utilise directement le problème de calcul du coût réduit $(Rcost_t)$ en 18 page 12. Ceci prouve que pour un couple (λ,μ) fixé, et à partir des valeurs de Bellman obtenues en résolvant (P_{relax}), on peut calculer par un sac à dos l'optimum de $(P_{relax\setminus t})$ □

Nous recherchons finalement la date t pour laquelle ce coût réduit $rcost(t)$ est maximum, et nous obtenons une ***borne duale additive (adb)*** (voir [7]) telle que :

$$\boxed{tdb(\mu,\lambda) = \omega(\mu,\lambda) + adb(\mu,\lambda) \quad \leq \quad f^*(p_u, y_u)} \quad (31)$$

avec
- $\omega(\mu,\lambda)$: fonction duale de Lagrange
- $tdb(\mu,\lambda)$: borne duale totale
- $adb(\mu,\lambda) = \max_t(rcost(t))$: borne duale additive
- $f^*(p_u, y_u)$: optimum recherché de notre problème initial

Nous venons donc de voir une première utilisation des coûts réduits qui, exploités comme **information globale** en tant que borne duale additive, nous donne une **borne inférieure augmentée** : ce qui est original pour le problème UCP. Nous verrons au paragraphe 7.3 page 19 une autre manière d'exploiter ces coûts réduits : par filtrage de valeurs au cours d'une énumération implicite.

---

6 Primalisation

6.1 Principe

La mise en œuvre de la méthode décrite précédemment produit un minorant dont la qualité ne peut se mesurer que par rapport à l'optimum réel (généralement inconnu), ou par comparaison au coût d'une solution réelle du problème initial. Dans ce cas, la qualité mesurée pour notre borne inférieure est liée à la qualité des solutions réelles utilisées comme référence.

Indépendamment de notre algorithme principal de résolution par énumération implicite (relativement lourd à mettre en œuvre) que nous verrons ultérieurement, nous avons besoin de générer une solution réalisable pour les raisons suivantes :

– mesure de la qualité de notre minorant,
– évaluation quantitative de l'intérêt de notre algorithme principal de résolution par rapport à une technique de primalisation simple et rapide.

Les solutions duales obtenues sont rarement réalisables (jamais dans notre cas : elles violent certaines contraintes de réserve). Nous allons donc procéder à une technique de *réparation* de ces solutions duales (ou encore de *primalisation*). Une solution sera dite **primalisable** si les séquences d'allumage pour chaque unité étant déterminées, on peut trouver un réglage des centrales allumées respectant l'ensemble des contraintes du problème. Ceci ce traduit par les deux conditions[6] de primalisation suivantes :

$$\begin{cases} \textbf{C1p} & \text{(pas de surproduction) :} & \forall t \in \{1,...,t_{\max}\} & \sum_u y_u(t)\, p_{\min u} & \leq & d(t) \\ \textbf{C2p} & \text{(réserve garantie) :} & \forall t \in \{1,...,t_{\max}\} & \sum_u y_u(t)\, p_{\max u} & \geq & d_{\max}(t) \end{cases} \tag{32}$$

La condition C2p est exactement la contrainte de réserve C2. La condition C1p garantie *a priori* que sa satisfaction permettra de régler *a posteriori* les centrales allumées sans risque de surproduire. En pratique, pour les solutions duales que l'on aura à primaliser, la condition C1p sera toujours respectée et l'ensemble des unités allumées pourra toujours être réglé pour éviter la surproduction : il n'y aura jamais trop d'unités allumées.

Le respect de ces deux conditions permet d'assurer que pour une séquence d'allumage déterminée, on pourra toujours trouver un réglage des centrales respectant l'ensemble des contraintes du problème. On en déduit un algorithme de réparation qui s'exécute en deux phases :

6.2 Phase 1 : création d'une solution primalisable

Une solution duale associée au couple de multiplicateurs de Lagrange (μ, λ) est définie par l'ensemble des $u_{\max}$ plus courts chemins dans les graphes d'états des unités, ces graphes étant pondérés par les multiplicateurs de Lagrange conformément à la figure 6 page 9.

Le principe de notre primalisation consiste à forcer progressivement l'allumage d'unités éteintes tant que la condition de réserve C2p n'est pas satisfaite en toute date. On vérifie alors que les deux conditions C1p et C2p sont satisfaites. Ce forçage doit être effectué en conservant le respect des contraintes dynamique C3 et C4 qui étaient bonnes par contruction des solutions duales (car ces contraintes sont prises en compte dans le problème relâché) : une séquence d'allumage de chaque unité doit être maintenue valide.

La solution retenue est de sous-traiter ce forçage à l'algorithme de programmation dynamique (voir figure 6) en obligeant les chemins à passer par un état allumé à la date considérée. Pour imposer l'allumage d'une unité u à une date t donnée, il suffit d'affecter un poids infini à tous les arcs du graphe de l'unité u qui conduisent à une extinction à la date t. Cette nouvelle pondération du graphe peut conduire à l'allumage des unités forcées à des instants autres que t (avant et/ou après t). Mais dans tous les cas, les contraintes dynamiques C3 et C4 resteront satisfaites par construction des graphes d'états.

Deux questions se posent encore :
– quelle date doit-on traiter en priorité ?
– comment choisir l'unité éteinte dont on veut forcer l'allumage ?

Nous choisissons de traiter en priorité la date pour laquelle la condition C2p est la moins bien respectée. Nous sélectionnons alors l'unité dont l'allumage à cette date est le moins coûteux[7] au sens des coûts réduits tels qu'ils sont définis dans le paragraphe suivant. Ce forçage est pris en compte en ré-appliquant notre programme dynamique. Ceci a pour effet, non seulement de réduire ou supprimer le viol C2p pour la date en cours de traitement, mais également pour une ou plusieurs dates voisines. Nous réitérons ce processus jusqu'à ce que la condition C2p soit totalement satisfaite. Enfin nous vérifions (*a posteriori*) la satisfaction de la condition C1p[8].

[6] Nous n'utilisons pas le mot *contrainte* que nous réservons au problème initial.
[7] Le choix de l'unité à allumer pourrait facilement être amélioré en sélectionnant la première unité canditate dans la solution du problème de sac à dos décrit section 5.3 page 12. Suivant le cas, cette unité correspondrait à un compromis entre l'unité la plus *rentable* à allumer (rapport $p_{\max u}/rcost_u(t)$) et l'unité la moins chère à allumer ($rcost_u(t)$).

6.3 Phase 2 : ajustement des puissances produites

Connaissant pour chaque date l'ensemble des centrales allumées, répartir optimalement la production pour la demande $d(t)$ est connu sous le nom de *Economic Dispatch Problem* :

$$\forall t \in D^+, \qquad \begin{cases} \min\limits_{\substack{u \in \mathrm{ON}(t) \\ p_u(t) \in P_u}} \sum\limits_{u \in \mathrm{ON}(t)} c_{\mathrm{f}u}(p_u(t)) \\ \text{s.c.} \sum\limits_{u \in \mathrm{ON}(t)} p_u(t) = d(t) \end{cases} \tag{33}$$

avec $P_u = [p_{\min u}, p_{\max u}]$: intervale des puissances que peut fournir une unité u allumée.

En dualisant la contrainte d'égalité à laquelle on associe le multiplicateur $\pi(t)$, on obtient le problème dual suivant :

$$\forall t \in D^+, \qquad \max_{\pi(t) \in \mathbb{R}} \quad \min_{\substack{u \in \mathrm{ON}(t) \\ p_u(t) \in P_u}} \quad \sum_{u \in \mathrm{ON}(t)} \big(c_{\mathrm{f}u}(p_u(t)) + \pi(t).p_u(t) \big) \; - \; \pi(t).d(t) \tag{34}$$

Ce sous-problème revient à chercher un prix d'incitation à la production $\pi(t)$ (qui sera donc négatif pour favoriser la production) permettant de satisfaire la demande à coût minimum pour chaque date t. Il est comparable à celui qui nous a permis de prérégler les coûts avant chaque itération de la relaxation lagrangienne, les multiplicateurs de Lagrange $\mu(t)$ jouant le rôle du prix d'incitation $\pi(t)$. Lors de ce précalcul, $\mu(t)$ était fixé et l'on pouvait directement calculer la puissance optimale. Cependant, cette fois, le prix $\pi(t)$ est une inconnue qu'il s'agit de déterminer itérativement en fonction de la demande en puissance $d(t)$.

Pour une raison de simplicité, et compte tenu de l'aspect non critique de ce sous-problème, nous avons déterminé la valeur de $\pi(t)$ par une ***recherche dichotomique*** qui est rapide à mettre en œuvre, et dont l'efficacité est suffisante compte tenu de la précision recherchée. Une amélioration possible pourrait consister à précalculer une fois pour toute les prix $\pi_{\min_u}$ et $\pi_{\max_u}$ correspondant aux seuils de saturation de chaque unité $p_{\min u}$ et $p_{\max u}$.

Finalement, notre algorithme de primalisation est suffisamment rapide pour pouvoir être exécuté en cours de relaxation lagrangienne (toutes les une à dix itérations). L'algorithme de primalisation est donné dans l'annexe **1.3** page **27**

---

7 Énumération implicite des solutions

7.1 Branch and Bound (séparation et évaluation)

Nous avons vu que, pour une séquence d'allumage $y_u(t)$ donnée, on pouvait calculer les productions optimales $p_u(t)$ par dichotomie sur le prix $\pi(t)$. Le problème initial peut donc se résumer à trouver la séquence d'allumage conduisant au coût total de production minimum. Les méthodes de résolution consistent généralement à fixer les variables de décision d'une manière progressive.

La méthode d'énumération implicite ***Branch and Bound* (séparation et évaluation)** a pour but d'explorer l'ensemble des solutions de manière aussi économique que possible. Son principe est le suivant : on représente l'ensemble des solutions par les feuilles d'un arbre. Un nœud interne représente une solution partielle, c'est-à-dire un ensemble de décisions déjà prises (par exemple certaines valeurs de $y_u(t)$ sont déjà fixées). L'algorithme de *Branch and Bound* consiste à parcourir l'arbre des solutions en traitant chaque nœud intermédiaire de la façon suivante :

– ***séparation*** de l'ensemble des solutions correspondant aux fils de ce nœud en plusieurs branches. Dans notre cas, on choisit une variable restée libre $y_u(t)$, on crée une branche fille associée à la

[8] Si la condition C1p n'est pas remplie après satisfaction de la condition C2p, nous considérons que notre primalisation a échoué, ce qui ne s'est jamais produit en pratique.

valeur de cette variable fixée à 0, et une autre branche fille associée à sa valeur 1. On choisit une des branches filles que l'on va explorer en premier en affectant la valeur sélectionnée à cette variable,
- ***évaluation*** de la borne inférieure des solutions de cette branche fille. Si la borne inférieure associée à cette branche est pire (i.e. de coût supérieur) qu'une solution réelle déjà trouvée, on peut la couper et ainsi éviter d'explorer ce sous-arbre : on continue avec une branche non encore explorée.

Dans notre cas, la borne inférieure correspondant à une solution partielle s'obtient par quelques itérations lagrangiennes appliquées aux graphes d'états des unités dont certains arcs ont été forcés pour prendre en compte les variables $y_u(t)$ déjà fixées. Ceci conduit à :
- intégrer dans le coût de la borne inférieure du nœud courant la conséquence des décisions déjà prises,
- mettre à jour les coûts réduits associés aux variables restées libres.

Un compromis est naturellement à trouver entre la qualité d'une bonne relaxation réalisée en chaque nœud (demandant de nombreuses itérations lagrangiennes), et la vitesse d'exploration de l'arbre. On peut néanmoins affirmer qu'une bonne relaxation lagrangienne est d'autant plus souhaitable que la branche sélectionnée n'est pas celle proposée par la relaxation précédente. En effet, si l'on choisit un nœud fils conformément au conseil lagrangien, on bénéficie déjà des itérations lagrangiennes précédentes, alors que si l'on prend la décision inverse, on doit laisser plus de temps à la relaxation lagrangienne pour prendre en compte cette nouvelle décision. En pratique, nous effectuons entre 0 et 10 itérations à chaque nœud.

On constate que l'efficacité de cette technique dépend essentiellement de la qualité de la borne inférieure dont on dispose : plus elle est élevée, plus l'arbre pourra être élagué tôt ; mais il est également nécessaire de disposer d'une bonne solution réelle de notre problème, laquelle peut être obtenue :
- soit par une heuristique indépendante de notre *Branch and Bound* (comme celle calculée dans la section **6** page **15**),
- soit au cours du *Branch and Bound* en choisissant en priorité d'explorer les branches les plus prometteuses.

Il s'agit maintenant de définir une heuristique permettant d'une part de déterminer l'ordre des variables de branchement (quelle variable $y_u(t)$ va t'on examiner en premier ?), et d'autre part l'ordre des valeurs auxquelles seront fixées ces variables (va-t-on commencer par fixer $y_u(t)$ à 1 ou à 0 ?).

7.2 Guidage dynamique de l'exploration par la relaxation

Afin d'explorer l'arbre des solutions en fixant progressivement les variables $y_u(t)$, nous devons maintenant déterminer l'ordre et la valeur des variables à fixer.

L'**heuristique que nous avons choisie** consiste à nous baser sur les coûts réduits induits par la relaxation lagrangienne : nous choisissons en priorité les couples (variable $= y_u(t)$, valeur $\in \{0, 1\}$) les plus fortement encouragés par la relaxation, à savoir les variables dont l'inversion d'état coûterait le plus cher compte tenu des coûts réduits. En effet, forcer une variable à une valeur opposée à celle « fortement conseillée » par la relaxation conduira à une solution d'autant plus coûteuse que le coût réduit associé à cette variable est élevé. Par ailleurs, les variables associées à un coût réduit faible sont celles pour lesquelles les valeurs ne sont pas très bien « décidées » par la relaxation, et donc celles pour lesquelles le changement d'état correspondra à des solutions de coût voisin. Choisir de brancher en dernier sur ces variables permettra à l'algorithme de *Branch and Bound* d'explorer en priorité les solutions de coût voisin, jugées par la relaxation lagrangienne comme étant parmi les meilleures candidates.

L'algorithme de base de notre branchement est alors le suivant :

1. ***traitement d'une feuille*** : s'il n'y a plus de variable libre, on est à une feuille. On mémorise éventuellement cette solution et on remonte explorer une autre branche de l'arbre ;
2. ***séparation*** : pour chaque variable $y_u(t)$ restée libre, on calcule le coût réduit $rcost_u(t)$, que ce soit pour l'allumer ou pour l'éteindre. On sélectionne la variable $y_u(t)$ dont le changement coûterait le plus cher et on crée deux branches, une associée à la valeur binaire $\mathcal{L}(y_u(t))$ du conseil lagrangien, et une associée à la valeur complémentaire $\overline{\mathcal{L}}(y_u(t))$ (de coût $rcost_u(t)$). On commence par explorer la branche associée à la valeur $\mathcal{L}(y_u(t))$;

3. ***exploitation*** de la décision précédente pour tenter d'autres affectations de variables encore libres grâce :
   – au ***filtrage*** par coûts réduits (détaillé section **7.3**),
   – à la ***propagation*** de contraintes dans le moteur PPC (décrit section **7.4**) ;
4. ***évaluation*** : une ou plusieurs variables ont été fixées, et il reste des variables libres : on effectue donc quelques itérations lagrangiennes de façon à prendre en compte ces modifications. Ceci a pour effet :
   – de mettre à jour les multiplicateurs de Lagrange, donc les coûts perturbés associés aux arcs du graphe d'états. De nouveau coûts réduits sont calculables ;
   – d'évaluer une borne inférieure de notre sous-problème, et éventuellement de conclure à une inutilité de cette branche.
5. ***recommencer*** en 1 tant que l'arbre n'est pas entièrement exploré.

7.3 Exploitation des coûts réduits pour le filtrage

Comme nous l'avons vu à la section **7.1**, lors de la construction d'une solution partielle par une technique d'énumération, nous connaissons pour un nœud intermédiaire n donné :

– la meilleure valeur primale bpv (coût de la meilleure solution connue),

– la borne inférieure $db^{(n)}$ (duale) de la branche correspondant au nœud n en cours d'exploration.

Pour ce nœud intermédiaire n, on examine toutes les variables $y_u^{(n)}(t)$ restées libres (i.e. non affectées à une valeur). Pour chacune d'elle, la relaxation lagrangienne propose une valeur $\mathcal{L}(y_u^{(n)}(t)) \in \{0,1\}$. On calcule alors le coût réduit $rcost_u^{(n)}(t)$ pour imposer à cette variable la valeur complémentaire $\overline{\mathcal{L}}(u_u^{(n)}(t))$. Compte tenu du minorant $db^{(n)}$ calculé pour cette branche fille, on déduit que le minorant pour toute solution fille de ce nœud pour laquelle on imposerait $y_u^{(n)}(t) = \overline{\mathcal{L}}(y_u^{(n)}(t))$ vaut $db^{(n)} + rcost_u^{(n)}(t)$. Si ce minorant est supérieur à la meilleure solution connue bpv, alors on peut sans crainte suivre le conseil lagrangien en affectant la variable $y_u^{(n)}(t)$ à $\mathcal{L}(y_u^{(n)}(t))$: ceci réduit d'autant la durée d'exploration de la branche courante. C'est cette technique que nous appelons ***filtrage par coûts réduits*** [8–9].

Précisons cependant, que la valeur duale utilisée pour effectuer cette coupe, ne doit pas déjà intégrer la borne duale additive calculée ci-dessus. En effet, pour un sous-problème donné :

– ou bien nous exploitons les coûts réduits dans la borne duale additive, ce qui nous permet d'effectuer des coupes plus efficaces en comparant la meilleure solution déjà trouvée bpv et la borne duale totale $tdb = \omega(\mu,\lambda) + adb$,

– ou bien nous effectuons le filtrage de certaines variables en comparant la meilleure solution connue bpv à la valeur duale du sous-problème courant $\omega(\mu,\lambda)$ augmentée du coût réduit $rcost_u^{(n)}(t)$ associé à cette variable. La valeur duale du sous-problème ainsi utilisée ne doit donc pas déjà intégrer les coûts réduits.

En plus de l'utilisation des coûts réduits pour améliorer le **minorant global** de notre problème, nous venons d'exploiter ces coûts réduits d'une seconde manière, cette fois comme **information locale** à chaque variable. Ceci permet à l'algorithme d'énumération, conjointement avec la borne duale de la relaxation lagrangienne et une bonne solution primale, de fixer certaines variables par **filtrage** ; ce qui est également original.

La procédure de filtrage par les coûts réduits est donnée en annexe **1.4** page **27**

7.4 Programmation Par Contraintes

La **Programmation Par Contraintes** [10] est une technique issue de l'intelligence artificielle dont le but initial était de trouver une solution réalisable pour un problème soumis à de nombreuses contraintes logiques. Un **Problème de Satisfaction de Contraintes** (CSP) est composé :

– d'un ensemble de variables de décision (les $y_u(t)$ dans notre cas),
– du domaine des valeurs possibles associées à chaque variable ($\{0,1\}$ dans notre cas),
– d'un ensemble de contraintes qui lient certaines variables entre elles.

Chaque contrainte restreint la combinaison des valeurs que l'ensemble des variables peut prendre simultanément. Une solution au CSP est une affectation de l'ensemble des variables qui satisfait toutes les contraintes. La propagation de contraintes appliquée à un sous-problème peut permettre :

- de constater une contradiction (problème sans solution),
- de réduire plus ou moins le domaine d'autres variables, et de restreindre ainsi le problème.

Dans le cadre du problème UCP, une propagation de contrainte sur le problème initiale permet par exemple de conclure que pour les dates de forte demande, conjointement aux possibilités de production des unités, un certain sous-ensemble d'unités est forcément allumé, ce qui restreint (légèrement) le problème de départ. Cette propagation sera appelée par la suite **propagation initiale**.

Outre cette propagation initiale dont l'effet est relativement faible (UCP est plutôt un problème d'optimisation que de faisabilité), la **propagation de contraintes** est couramment utilisée au cours d'une énumération implicite. En effet, toute modification du domaine d'une variable suite à une prise de décision conduit à réveiller les contraintes associées, et à propager cette information aux autres variables également concernées par ces contraintes. Ceci conduit à une restriction en chaine des domaines de certaines variables, et réduit d'autant l'arbre de recherche des solutions.

Par exemple, en cours de construction d'une solution partielle, on peut être amené à fixer la variable $y_u(t)$ à 0, c'est-à-dire éteindre l'unité u à la date t. Compte tenu de la capacité ($p_{\max u}$) des unités précédement fixées à 1 à cette même date, et de la capacité des unités restées libres, on peut déduire qu'il sera impossible de satisfaire la contrainte de réserve C2 pour la date t sans utiliser la plus grosse des unités u' restées libres. La propagation de la contrainte C2 conduit alors à affecter[9] la variable $y_{u'}(t)$ à la valeur 1.

Par conséquent, en plus de la propagation initiale, nous voyons que l'utilisation de la Programmation Par Contraintes après l'affectation d'une variable lors d'une énumération (cf. item 3 de l'algorithme section **7.2**) peut permettre de :

- provoquer d'autres fixations de variables en chaine, voire conduire à une feuille dont on peut mémoriser la solution associée, avant de continuer l'exploration de l'arbre,
- conduire à une impossibilité pour la branche en cours et l'on doit remonter explorer d'autres branches (*backtrack*).

7.5 Limitation de l'exploration : LDS

Malgré les coupes obtenues par nos bornes inférieures, par l'exploitation des coûts réduits pour le filtrage et par la Programmation Par Contraintes, l'explosion combinatoire reste présente et nous sommes amenés à arrêter l'exploration avant d'avoir pu trouver l'optimum. Nous sommes donc conduits à choisir un critère d'arrêt pour limiter l'exploration. Deux techniques sont couramment employées :

- s'arrêter dès que le nombre total de nœuds internes conduisant à une impossibilité atteind un certaine valeur (*Backtrack Limit*).

  Si l'on suppose que dans l'arbre d'exploration, les décisions acceptant la proposition lagrangienne correspondent à la branche de gauche en chaque nœud, cette technique consiste alors à explorer en priorité les feuilles situées en bas à gauche de l'arbre ;
- n'explorer que les branches (et toutes les branches) contenant un nombre maximum n de choix contraires à notre heuristique lagrangienne (technique *LDS*$_n$ pour *Limited Discrepency Search*) [11], dans le but de ne pas trop nous éloigner de notre heuristique. Cela revient, dans notre arbre de recherche, à explorer une sélection de feuilles équitablement réparties sur la largeur de l'arbre.

Avec cette seconde technique, le LDS$_0$ est un algorithme glouton[10] consistant à effectuer une simple descente dans l'arbre, et à s'arrêter à la première feuille réalisable. Le LDS$_1$ quant à lui, consiste à explorer l'arbre de recherche en n'autorisant, en cours de descente dans l'arbre d'exploration, qu'un

[9] D'un point de vue plus général, la propagation de la contrainte C2 conduit alors à retirer la valeur 0 du domaine de la variable $y_{u'}(t)$. Nos variables étant binaires, le nouveau domaine devient un singleton, ce qui conduit à *instancier* la variable $y_{u'}(t)$ à la valeur restante, donc à 1.

[10] Un algorithme glouton (greedy) est un algorithme qui construit progressivement une solution sans jamais remettre en cause les choix effectués.

seul choix contraire à notre heuristique lagrangienne. Le LDS_1 est donc susceptible d'explorer N_v fois plus de nœuds que le LDS_0, N_v étant le nombre de variables à affecter, soit $u_{\max} \times t_{\max}$ pour notre problème.

Précisons que par rapport à notre technique de primalisation décrite à la section **6**, le glouton LDS_0 bénéficie (et paie le prix) de l'apport du filtrage et de la Propagation Par Contraintes. L'algorithme d'énumération implicite avec la fonctionnalité « LDS » est détaillé en annexe **1.5** page **28**.

8 Résultats obtenus et analyse

8.1 Préambule

Le développement de notre application a été effectué sur différentes plate-formes de puissance comparable (type PC fin 2003) fonctionnant sous des systèmes d'exploitation variés (Windows NT, Linux et MacosX). Les résultats et les temps de calcul donnés ci-après ont été obtenus sur station PC serveur bi-processeurs 64 bits sous linux[11]. Le langage utilisé, **Claire**[12], permet d'obtenir un exécutable compilé en s'appuyant sur un compilateur C++ externe. Le moteur de propagation de contraintes utilisé était **Choco**[13]. Cette bibliothèque écrite en Claire (principalement par François Laburthe) est en cours de port vers le langage java.

Dans les tableaux qui suivent, la première colonne représente les noms d'instances dont le détail est reporté en annexe **2** page **29**. Les colonnes $|D|$ et $|U|$ indiquent repectivement le nombre de dates et d'unités de chaque instance.

8.2 Relaxation lagrangienne et bornes inférieures

Dans cette première partie, nous présentons les bornes inférieures obtenues par *Relaxation Lagrangienne* . Nous montrons d'une part la contribution de la *borne duale additive* (voir section **5.4** page **13**) dans la borne duale totale, et d'autre part l'apport de la *Programmation Par Contraintes* initiale (voir section **7.4** page **19**) sur la valeur duale.

Le tableau suivant comprend pour chaque instance de problème deux colonnes principales. La colonne duale sans ppc correspond à la relaxation lagrangienne simple, sans utilisation de la Programmation Par Contraintes, et la colonne duale avec ppc intègre la propagation initiale. L'amélioration résultante est indiquée dans la dernière colonne.

Dans les deux cas, la valeur de la borne duale additive qui est indiquée entre parenthèses est intégrée à la borne duale totale. Liée à l'aspect discret du problème, elle est plus importante pour les petites instances, mais reste d'une contribution modeste dans tous les cas.

[11] Machines bi-processeurs 64 bits athlon opteron 2 GHz muni de 4 Goctets de RAM et fonctionnant sous linux Suze 64 bits.

[12] Ce prototype de langage dédié à l'optimisation combinatoire, est relativement lourd à utiliser pour les personnes non initiées : il nécessite de travailler dans un milieu de spécialistes ! Toute information complémentaire peut être obtenue à partir du site `http://www.claire-language.com`.

[13] La page de Choco en tant que bibliothèque Claire est située en `http://www.choco-constraints.net/`. La nouvelle page correspondant à sa réecriture en Java est désormais accessible en tant que projet (déjà opérationnel) *open-source* depuis `http://choco.sourceforge.net/`.

	$\|D\|$	$\|U\|$	duale sans PPC (dont adb)		duale avec PPC (dont adb)		amél. %
ucp0	8	4	73684.83	(dont 320.8)	**74161.27**	(dont 314.64)	0.65
ucp1a	12	5	133578.18	(dont 25.75)	**134249.35**	(dont 53.96)	0.50
ucp1b	12	10	276377.71	(dont 16.03)	**276535.28**	(dont 28.56)	0.057
ucp2	24	5	271230.74	(dont 113.33)	**273558.87**	(dont 106.44)	0.86
ucp3	24	10	557861.97	(dont 17.60)	**558018.95**	(dont 16.89)	0.028
ucp4	24	20	1115719.60	(dont 10.21)	**1116031.11**	(dont 11.21)	0.028
ucp5	24	40	2231434.39	(dont 13.97)	**2232058.84**	(dont 1.71)	0.028
ucp6	24	60	3347135.41	(dont 8.46)	**3347137.19**	(dont 6.18)	~ 0
ucp7	24	80	4462868.78	(dont 27.92)	**4462868.90**	(dont 12.62)	~ 0
ucp8	24	100	5578418.65	(dont 9.70)	**5578432.60**	(dont 5.78)	~ 0
ucp8b	24	100	5623456.23	(dont 13.91)	**5623464.01**	(dont 1.00)	~ 0
ucp9	24	200	11347576.40	(dont 0.85)	**11347576.48**	(dont 0.84)	~ 0
ucp10	48	200	**22250330.25**	(dont 23.66)	**22250330.25**	(dont 23.66)	$= 0$

Le nombre d'itérations lagrangiennes pour assurer la convergence de l'algorithme est compris entre 50 et 5000. Pour la relaxation lagrangienne et la primalisation, le temps d'exécution varie d'une fraction de seconde pour ucp0 à une heure pour la plus grosse instance ucp10.

8.3 Solutions primales

Cette seconde partie montre le coût des solutions réelles obtenues par différentes techniques. Nous distinguons d'une part les solutions fournies par notre heuristique de primalisation des solutions duales, et d'autre part les solutions obtenues en utilisant la PPC. Pour chaque méthode de résolution sont indiqués la valeur obtenue et l'écart dual-primal (gap_a ou gap_b) par rapport à la meilleure borne duale correspondante[14], obtenue précédemment.

La première colonne fournit les valeurs obtenues par ***primalisation lagrangienne*** avec l'algorithme décrit page **15**. Cet algorithme est appliqué à chaque itération de la relaxation. Le temps indiqué intègre le temps de convergence de la relaxation lagrangienne. Les résultats indiquent que cette technique est celle qui, dans le minimum de temps (1 heure sachant qu'une très bonne solution est produite en quelques minutes), atteint la meilleure solution trouvée pour la plus grosse instance.

La seconde série de résultats correspond au glouton LDS_0 (voir section **7.5** page **20**). Rappelons que la différence essentielle entre la primalisation (également un glouton) et l'algorithme LDS_0 (glouton de plus haut niveau) est que ce dernier exploite le filtrage par coûts réduits ainsi que la propagation de contraintes au fur et à mesure de la fixation des variables.

Ensuite vient le LDS_1. Celui-ci ne se termine que pour les petites instances : en effet, pour l'instance ucp5, le nombre de branches à explorer par le LDS_1 est de l'ordre de 1000 fois plus important que pour le glouton LDS_0[15].

Enfin nous présentons le résultat de l'énumération complète, laquelle ne se termine également que pour les plus petites instances.

De plus, les notations suivantes sont utilisées :

- une valeur est précédée du signe « = » si celle-ci a été prouvée optimale par une exploration complète ;
- une valeur est indiquée entre parenthèse (xxx) lorsque l'exploration a été interrompue avant la fin (au bout de 6 heures de calcul) ;
- les valeurs en gras correspondent aux meilleures solutions primales obtenues.

[14] En particulier, les bornes duales utilisées pour le calcul du gap_a des solutions obtenues sans utiliser la PPC, sont également celles qui n'exploitent pas la PPC. Les écarts gap_a et gap_b étant calculés à partir de valeurs duales différentes, ceci explique que dans le tableau suivant, certaines valeurs primales identiques conduisent à des *gap* différents.

[15] En effet, l'algorithme LDS_0 permet de n'explorer qu'une seule branche, puisque l'on s'impose de suivre le conseil lagrangien. Le LDS_1 quant à lui, autorise le non-respect de l'heuristique lagrangienne pour une seule variable au maximum. Or le nombre de variables $y_u(t)$ pour ucp5 est de $|D| \times |U| = 24 \times 40$.

					avec PPC					
	$\|D\|$	$\|U\|$	**primalisation**	(**gap**$_a$)	**LDS**$_0$	(**gap**$_b$)	**LDS**$_1$	(**gap**$_b$)	**exploration complète**	(**gap**$_b$)
ucp0	8	4	74988.56	1.769	**74906.06**	1.004	**74906.06**	1.004	**=74906.06**	1.004
ucp1a	12	5	**136231.02**	1.986	**136231.02**	1.476	**136231.02**	1.476	**=136231.02**	1.476
ucp1b	12	10	**279108.96**	0.988	**279108.96**	0.931	**279108.96**	0.931	**=279108.96**	0.931
ucp2	24	5	278155.63	2.553	277942.88	1.603	**277848.47**	1.568	**(277848.47)**	1.568
ucp3	24	10	563719.80	1.050	562877.02	0.871	562877.02	0.871	**(562837.69)**	0.864
ucp4	24	20	1122042.49	0.567	1121589.87	0.498	**1120705.34**	0.419	(1120994.44)	0.445
ucp5	24	40	2237824.45	0.286	2237191.04	0.230	**(2236808.68)**	0.213	(2237191.04)	0.230
ucp6	24	60	3355191.01	0.241	3351458.35	0.129	**(3351411.93)**	0.128	(3351458.35)	0.129
ucp7	24	80	4470455.44	0.170	4469165.93	0.141	**(4468716.75)**	0.131	**(4468716.75)**	0.131
ucp8	24	100	5587788.19	0.168	5583925.49	0.099	**(5583740.24)**	0.095	(5583925.49)	0.099
ucp8b	24	100	5629661.86	0.110	5629441.92	0.106	**(5628948.29)**	0.098	**(5628948.29)**	0.098
ucp9	24	200	11353934.20	0.056	11353839.44	0.055	**(11353650.10)**	0.054	(11353839.44)	0.055
ucp10	48	200	**22264623.41**	0.064	**22264623.41**	0.064	**(22264623.41)**	0.064	**(22264623.41)**	0.064
temps maxi			$1h$		$2h$		$\leq 6h$		$\leq 6h$	

8.4 Apport des différentes méthodes

Relaxation lagrangienne et primalisation

La relaxation lagrangienne comme la primalisation des solutions duales donnent (classiquement) d'excellents résultats, surtout pour les grosses instances qui se rapprochent d'un problème continu. Les valeurs relatives des solutions primalisées sont moins bonnes pour les petites instances. Cependant l'écart entre nos valeurs duales et nos valeurs primales semble essentiellement causé par le saut de dualité. C'est au moins le cas pour les petites instances qui ont pu être traitées par énumération complète. En fait, une heuristique lagrangienne de primalisation utilisée seule est suffisante pour les grosses instances. De plus, notre heuristique de primalisation pourrait être améliorée en utilisant une technique de figeage progressif des variables en exploitant les coûts réduits. On se rapprocherait alors du glouton LDS_0 mais sans propagation de contraintes.

Apport de l'exploitation des coûts réduits

L'apport est faible au niveau de la ***borne duale additive***. Il est d'ailleurs d'autant plus faible que la borne est excellente, c'est-à-dire pour les grosses instances. Ceci est assez logique car la borne additive exploite des aspérités des solutions duales dues à l'aspect discret d'un problème. Ces imperfections sont d'autant plus faibles qu'elles sont lissées par la relaxation pour les grandes instances.

L'apport du ***filtrage*** sur les résultats est plus délicat à mesurer, et nécessiterait des tests complémentaires. On peut cependant dire, en observant le nombre d'instanciations dues au filtrage pour le LDS_0 qu'il n'est intéressant, comme pour la propagation en générale, que pour les instances les plus discrètes, donc les plus petites.

En ce qui concerne l'apport des coûts réduits dans ***l'heuristique du choix des variables***, il est ***excellent*** : c'est le plus gros bénéfice des coûts réduits ! La preuve en est de la qualité des solutions obtenues par le LDS_0 qui peut être vu comme une technique gloutonne plutôt que comme une technique de limitation de l'ensemble des feuilles explorées (car il se limite à un singleton). En effet, **le fait de sélectionner en premier les variables dont l'inversion d'état coûterait le plus cher est un choix très fort qui prend en compte le problème dans sa globalité, et qui est donc relativement stable au cours de l'énumération.**

Notre heuristique de branchement permet d'ordonner les variables par ordre de « conviction lagrangienne » décroissante, et donc indirectement d'ordonner l'exploration des solutions (les feuilles de l'arbre) par ordre d'intérêt décroissant en les concentrant dans la partie gauche de l'arbre laquelle sera explorée en premier. Autrement dit, notre heuristique de branchement peut être vu comme un ***moyen assez générique d'ordonner les solutions explorées par ordre d'intérêt décroissant***.

Remarques sur la technique LDS

La limitation de l'exploration par la technique LDS_1 n'apporte généralement pas grand chose par rapport à une simple descente gloutonne (LDS_0) ou par rapport à une énumération limitée en temps. On peut expliquer cet échec par le principe même du LDS (voir section **7.5** page **20**). En effet, le LDS suppose que l'on dispose d'une bonne heuristique pour choisir la valeur d'une variable que l'on veut fixer, mais ne prend absolument pas en compte le fait que le choix de la variable elle-même soit ou non pertinent. Par exemple, le LDS_1 n'autorise qu'une seule discordance lagrangienne, mais ne tient pas compte du coût réduit lié à cette discordance. Ainsi, parmi les solutions explorées par le LDS_1, figure une feuille qui fait partie des 50% plus mauvaises solutions telles que prévues par la relaxation lagrangienne, à savoir celles dont on impose que la première variable soit dans l'état contraire à l'heuristique lagrangienne. Or c'est précisément cette variable pour laquelle le choix lagrangien était le plus « sûr » (et c'est pour cette raison qu'elle a été choisie en premier).

Dans notre cas, la technique LDS utilisée de manière classique (limitation en nombre de discordances lagrangiennes, le terme anglais étant « discrepancies » de L**D**S) semble donc inadaptée à notre heuristique de branchement. Cependant une amélioration serait d'utiliser un LDS limitant non pas le nombre de *backtracks*, mais le cumul des coûts réduits induit par chaque *backtrack*. Chaque *backtrack* serait ainsi pondéré par un regret en terme de coûts réduits. De cette façon, ce $\boldsymbol{LDS_{pondéré}}$ permettrait d'explorer en priorité les feuilles correspondant à une distance homogène (en terme de coûts réduits) par rapport au conseil lagrangien.

Apport de la PPC

On constate que pour les grosses instances, la propagation n'apporte rien. Notons cependant que ce problème est très facile à primaliser puisqu'il « suffit » de payer plus cher pour rendre réalisable une solution duale : il n'a pas de problème de faisabilité sous-jacent. Or le point fort de la *Programmation Par Contraintes* est précisément les problèmes fortement contraints, et elle n'a donc pas pu être mise en valeur ici. On peut donc dire que l'intérêt de la PPC s'arrête là ou la relaxation lagrangienne devient excellente. Mais elle semble très bénéfique pour les instances les plus discrètes (donc les petites instances pour le problème UCP) et pourrait permettre de prouver les solutions pour les problèmes ucp3 voire ucp4.

Dans tous les cas la technique gloutonne LDS_0 utilisant le choix des variables par coûts réduits donne de très bons résultats, à la fois pour les petites instances qu'elle résoud, et pour les plus grosses instances pour lesquelles elle trouve nos meilleures solutions

---

9 Conclusions et perspectives

En couplant une méthode de relaxation lagrangienne avec un solveur par contraintes nous sommes désormais capables d'intégrer facilement, au sein de notre application, de nouvelles contraintes opérationnelles telles que celle d'imposer en permanance l'allumage d'un sous ensemble de centrales parmi les centrales d'une même région.

Par ailleurs nous avons mis en œuvre deux manières originales d'exploiter les coûts réduits issus de la relaxation lagrangienne pour le problème UCP : outre l'amélioration (faible) du minorant global par une borne duale additive, nous avons pu les utiliser pour filtrer des variables en cours d'exploration, et ainsi aller plus loin dans une exploration incomplète.

De plus, nous avons utilisé les coûts réduits comme guide dans notre heuristique de branchement. L'efficacité de ce guide est telle que la descente en elle-même de l'arbre d'exploration (LDS_0) fournit de très bonnes solutions.

Parmi les points à approfondir ou à explorer, on peut citer :

– évaluer le $LDS_{pondéré}$ pour améliorer l'efficacité dans la sélection des solutions explorées ;

– étudier l'apport du lagrangien augmenté dans le domaine discret pour :

– améliorer le saut de dualité pour les instances de taille moyenne (les grosses étant déjà bien traitées) ;
– améliorer la vitesse de convergence lors des relaxations intermédiaires en cours d'exploration (car le nombre d'itération est limité, mais les relaxations sont fréquentes) ;
– rendre les solutions duales plus proches de solutions réalisables, ouvrant ainsi la voie à de nouvelles heuristiques de primalisation.

---

10 Bibliographie

[1] Renaud, A. (1993). Daily Generation Management at Electricité de France. *IEEE TAC V-38-7*.

[2] Valenzuela, J. and Smith, A.E. (2002). A Seeded Memetic Algorithm for Large Unit Commitment Problems. *Journal of Heuristics*, 8:173-195.

[3] Rottembourg, B. (2003). Optimisation Combinatoire : Séparation, Relaxation et Approximation. *Cours ENSTA*.

[4] Bellman, R. (1957). Dynamic Programming. *Princeton University Press*.

[5] Geoffrion, A.M. (1974). Lagrangian relaxation and its uses in integer programming. *Programming Study*.

[6] Cohen, G. (2003). Optimisation des grands systèmes. *Cours DEA MMME*.

[7] Benoist, T. (2004). *Relaxations et Décompositions Combinatoires*. Thèse de doctorat, Avignon.

[8] Foccacci, F., Lodi, A. and Milano, M. (1999). Cost-Based Domain Filtering. *In Proceedings of the 5th International Conference on Principles and Practice of Constraint Programming*, pages 189–203.

[9] Benoist, T., Laburthe, F. and Rottembourg, B. (2001). Lagrange Relaxation and Constraint Programming Collaboratives schemes for Traveling Tournament Problems. *In Proceding CP-AIRO'01 Ashford*.

[10] Barták, R. (1999). Constraint Programming : In Pursuit of the Holy Grail. *Proceedings of Workshop of Doctoral Students'99*.

[11] Harvey, W. and Ginsberg, M. (1995). Limited Discrepancy Search. *In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, 1:607-613.

Annexe 1 L'algorithme complet

La mise en œuvre ne suit pas exactement les procédures décrites ci-dessous (en particulier la procédure d'exploration). En effet l'organisation du code dépend en grande partie de l'environnement utilisé (langage claire + bibliothèque choco) qui propose, outre le moteur de propagation de contraintes, une ossature de développement (*framework*) permettant des énumérations généralisées ainsi que la gestion des sauvegardes de contextes pour les *backtrack*. Une description de ce *framework* spécifique alourdirait inutilement les algorithmes tout en en masquant la substance.

1.1 Programme principal

Entrée :
- nom de l'instance à traiter (e.g. ucp8, ...)
- LDS_Limit Nombre maximum de discordance (e.g. 1 pour LDS_1, ...)

Sortie :
- affichage des meilleures solutions primale et duale

Algorithme :
- initialisation :
  - LDS_Counter := 0 (nombre de discordances courantes)
  - lecture de l'instance et construction du modèle informatique
  - propagation initiale (par Choco)
  - initialiser les multiplicateurs $\mu(t)$ et $\lambda(t)$ à 0
- **relaxation lagrangienne** principale avec primalisations internes
  $\Rightarrow$ on connait bestDualValue, bestPrimalValue et les coûts réduits
- **explorer le nœud racine** (i.e. toutes les variables étant libres)
- afficher les résultats

1.2 Procédure de relaxation lagrangienne

Entrée :
- nbIter : le nombre souhaité d'itérations
- mainRelaxation : vrai pour la relaxation principale, faux pendant la phase d'exploration

Sortie :
- thisAdditiveValue : borne duale additive
- thisDualValue : valeur duale (sans adv)
- si mainRelaxation : modif. de *bestDualValue* et de *bestAdditiveDualValue*

Algorithme :
- pour k := 1..nbIter
  - mettre à jour le poids des graphes d'états en fonction des multiplicateurs $\mu(t)$ et $\lambda(t)$
  - résoudre les plus courts chemins (en deux passes pour accéder aux coûts réduits)
  - calculer la borne duale *dv*
  - calculer les sous-gradients (viols des deux contraintes relachées C1 et C2)
  - calculer la borne duale additive *adv* (voir section **5.4** page **13**)
  - si mainRelaxation :
    - si *dv* + *adv* > *thisDualValue* + *thisAdditiveValue*
      - mémoriser les nouvelles valeurs de *thisDualValue* et de *thisAdditiveValue*
    - faire une primalisation
  - mettre à jour les nouveaux multiplicateurs $\mu(t)$ et $\lambda(t)$

1.3 Procédure *primaliser*

Entrée :
- la valeur des coûts réduits pour toutes les variables

Sortie :
- mise à jour éventuelle de la meilleure solution primale, et de son coût bestPrimalValue

Algorithme (section 6 page 15) :
- tant que la condition de réserve (C2p page 16) n'est pas satisfaite
  - chercher la date t pour laquelle le viol de C2 est le plus important
  - chercher l'unité u éteinte la moins coûteuse à allumer à la date t (page 12)
  - forcer l'allumage de u en t (poids infini sur certains arcs)
  - relancer le calcul des plus courts chemins (avec double passe) (voir section 4.4)
  - mettre à jour le calcul des viols de la contrainte de réserve

- si condition de non-surproduction (C1p page 16) est satisfaite
  - ajuster les puissances $p_u(t)$ (section 6.3 page 17)
  - mémoriser éventuellement cette nouvelle solution

  sinon
  - ECHEC de la primalisation

1.4 Procédure *filtrage par les coûts réduits*

Entrée :
- l'ensemble des variables non affectées avec leur coûts réduits
- thisDualValue : la borne inférieure courante (sans la borne additive)
- bestPrimalValue : le coût de la meilleure solution connue

Sortie :
- affecte un certain nombre de variables $y_u(t)$

Algorithme (section 7.3 page 19) :
- pour toute variable $y_u(t)$
  - si $y_u(t)$ n'est pas encore affectée
    - soit v, la valeur de cette variable proposée par la relaxation (donnée par les plus courts chemins de l'unité u à la date t)
    - calculer $rcost_u(t)$ le coût réduit pour forcer la variable à la valeur complémentaire $1 - v$
    - si thisDualValue $+rcost_u(t) >$ bestPrimalValue
      - affecter la valeur conseillée v à la variable $y_u(t)$
- propager les contraintes

1.5 Procédure *explorer un nœud*

Entrée :
- un nœud représentant une solution partielle (car certains $y_u(t)$ sont fixés)
- bestPrimalValue : le coût de la meilleure solution connue

Algorithme (section **7.1** page **17**) :
- faire quelques itérations lagrangiennes (inutile pour le nœud racine)
  - calcul de la borne inférieure de ce nœud (thisDualValue)
  - mise à jour du coût réduit des variables libres

- si thisDualValue + thisAdditiveDualValue $>$ bestPrimalValue
  - on ignore cette branche non prometteuse

  sinon
  - filtrage par les coûts réduits (voir annexe **1.4** page **27**)
  - si on est à une feuille (toutes les $y_u(t)$ sont fixées)
    - ajuster les puissances $p_u(t)$ (section **6.3** page **17**)
    - mémoriser éventuellement cette nouvelle solution

    sinon
    - sélectionner la variable libre $y_u(t)$ la plus fermement déterminée par la relaxation lagrangienne (voir section **7.2** page **18**). Soit v la valeur de $y_u(t)$ conseillée par la relaxation

    - exploration de la branche conseillée :
      - mémoriser le contexte
      - affecter $y_u(t)$ à la valeur conseillée v
      - propager les contraintes
      - explorer ce nœud fils
      - restaurer le contexte

    - exploration éventuelle de la branche déconseillée :
      - si LDS_Counter $<$ LDS_Limit
        - mémoriser le contexte
        - incrémenter LDS_Counter
        - affecter $y_u(t)$ à la valeur déconseillée $1 - v$
        - propager les contraintes
        - explorer ce nœud fils
        - restaurer le contexte

Annexe 2 Les instances

Le tableau suivant présente quelques statistiques sur les instances utilisées :

	$\lvert D\rvert$	$\lvert U\rvert$	$p_{\max}$	$p_{\min}$	t_{up}	t_{down}	c_{hs}	c_{cs}	t_{cs}	$d(t)$
ucp0	8	4	60 à 300	20 à 75	1 à 5	1 à 4	0 à 500	0.02 à 1100	0 à 5	280 à 600
ucp1a	12	5	55 à 485	10 à 150	1 à 8	1 à 8	30 à 4500	60 à 9000	0 à 5	250 à 750
ucp1b	12	10	55 à 455	10 à 150	1 à 8	1 à 8	30 à 5000	60 à 10000	0 à 5	700 à 1500
ucp2	24	5	55 à 485	10 à 150	1 à 8	1 à 8	30 à 4500	60 à 9000	0 à 5	350 à 750
ucp3	24	10	55 à 455	10 à 150	1 à 8	1 à 8	30 à 5000	60 à 10000	0 à 5	700 à 1500
ucp4	24	20	55 à 455	10 à 150	1 à 8	1 à 8	30 à 5000	60 à 10000	0 à 5	1400 à 3000
ucp5	24	40	55 à 455	10 à 150	1 à 8	1 à 8	30 à 5000	60 à 10000	0 à 5	2800 à 6000
ucp6	24	60	55 à 455	10 à 150	1 à 8	1 à 8	30 à 5000	60 à 10000	0 à 5	4200 à 9000
ucp7	24	80	55 à 455	10 à 150	1 à 8	1 à 8	30 à 5000	60 à 10000	0 à 5	5600 à 12000
ucp8	24	100	55 à 455	10 à 150	1 à 8	1 à 8	30 à 5000	60 à 10000	0 à 5	7000 à 15000
ucp8b	24	100	55 à 455	10 à 150	1 à 8	1 à 8	30 à 5000	60 à 10000	0 à 5	7000 à 15000
ucp9	24	200	55 à 455	10 à 150	1 à 8	1 à 8	30 à 5000	60 à 10000	0 à 5	14000 à 30000
ucp10	48	200	55 à 485	10 à 150	1 à 8	1 à 8	30 à 5000	60 à 10000	0 à 5	14000 à 30200

Les instances de base suivantes sont tirées de [2][16].

- L'instance ucp0 définie page **30** est celle utilisée pour la mise au point du programme. Celle-ci est composée de quatre unités avec une prévision sur 8 heures ;
- ucp3 est la plus petite des « vraies » instances. Elle est composée de 10 unités toutes différentes et d'une prévision de besoin définie sur 24 heures. Dans la littérature, elle est considérée comme traitée à l'optimum par programmation dynamique ;
- ucp4 à ucp8 sont des réplications de ucp3 correspondant à un nombre d'unités variant de 20 à 100 ; la demande prévisionnelle étant adaptée en conséquence. Ces réplications introduisent des symétries artificielles qui peuvent perturber le comportement de notre algorithme.

Les instances suivantes ont été ajoutées pour notre propre besoin.

- Afin de tester l'influence de ces symétries, l'instance ucp8b a été spécialement créée à partir de ucp8 en perturbant très légèrement les coûts des groupes d'unités dupliquées (moins de 1%). Ces symétries artificielles sont cependant remplacées par un ordre de préférence non moins artificiel entre les unités de même caractéristiques ;
- Deux grosses instances (également dissymétriques) ont été créées à partir de ucp8b : ucp9 avec 200 unités sur 24 heures, et ucp10 avec 200 unités sur 48 heures.

Enfin, des petites instances supplémentaires dérivées de ucp3 ont été créées pour observer le comportement de l'algorithme autour de problèmes plus « discrets » :

- ucp1a et ucp1b avec une prévision sur 12 heures et respectivement 5 et 10 unités ;
- ucp2 avec 5 unités sur 24 heures.

Remarque : les instances ucp8b, ucp9 et ucp10 sont des réplications *dissymétrisées* de ucp3. Cette dissymétrisation est obtenue par ajout d'un écart e au coefficient a_0 de chaque unité (la partie constante de son coût parabolique). L'écart e vaut $i - i_{\max}/2$ où $i_{\max}$ est le nombre de réplications et i l'indice de réplication variant de 1 à $i_{\max}$.

Le nombre de réplications $i_{\max}$ valant 20 pour ucp9 ou ucp10, et compte tenu des caractéristiques économiques des unités, on obtient une perturbation maximale du coût de certaines unités de l'ordre de 1% par rapport à la même instance non dissymétrisée.
Par ailleurs, l'instance de base ucp3 semble avoir été calculée au plus juste et dans certains cas les réplications dissymétrisées n'ont pas de solution. Pour que ces instances soient réalisables, nous avons donc été ammenés à augmenter légèrement la puissance $p_{\max}$ de certaines des unités.

[16] Des ambiguités dans l'interprétation des données de ces instances nous ont conduit à apporter quelques modifications à certains paramètres ($t_{\text{up}u}$ et $t_{\text{down}u}$). Ces modifications, certe mineures, rendent impossible une comparaison précise de nos résultats, en particulier, pour la mesure de la qualité de notre borne inférieure.

ucp0

id	pmax	pmin	a0	a1	a2	tup	tdown	chs	ccs	tcs	si
1	300.0	75.0	684.74	16.83	0.0021	5	4	500.00	1100.00	5	8
2	250.0	60.0	585.62	16.95	0.0042	5	3	170.00	400.00	5	8
3	80.0	25.0	213.00	20.74	0.0018	4	2	150.00	350.00	4	-5
4	60.0	20.0	252.00	23.60	0.0034	1	1	0.00	0.02	0	-6

Avec une demande sur 8 pas de temps définie par le tableau suivant,

date	demande	date	demande
1	450.0	5	400.0
2	530.0	6	280.0
3	600.0	7	290.0
4	540.0	8	500.0

ucp3

Le bench ucp3 est composé de 10 unités avec une prévision sur 24 heures.

id	pmax	pmin	a0	a1	a2	tup	tdown	chs	ccs	tcs	si
1	455.0	150.0	1000.0	16.19	0.00048	8	8	4500.0	9000.0	5	8
2	455.0	150.0	970.0	17.26	0.00031	8	8	5000.0	10000.0	5	8
3	130.0	20.0	700.0	16.60	0.00200	5	5	550.0	1100.0	4	-5
4	130.0	20.0	680.0	16.50	0.00211	5	5	560.0	1120.0	4	-5
5	162.0	25.0	450.0	19.70	0.00398	6	6	900.0	1800.0	4	-6
6	80.0	20.0	370.0	22.26	0.00712	3	3	170.0	340.0	2	-3
7	85.0	25.0	480.0	27.74	0.00079	3	3	260.0	520.0	2	-3
8	55.0	10.0	660.0	25.92	0.00413	1	1	30.0	60.0	0	-1
9	55.0	10.0	665.0	27.27	0.00222	1	1	30.0	60.0	0	-1
10	55.0	10.0	670.0	27.79	0.00173	1	1	30.0	60.0	0	-1

Avec une demande de 24 heures définie par le tableau suivant,

date	demande	date	demande	date	demande
1	700.0	9	1300.0	17	1000.0
2	750.0	10	1400.0	18	1100.0
3	850.0	11	1450.0	19	1200.0
4	950.0	12	1500.0	20	1400.0
5	1000.0	13	1400.0	21	1300.0
6	1100.0	14	1300.0	22	1100.0
7	1150.0	15	1200.0	23	900.0
8	1200.0	16	1050.0	24	800.0

Table des matières

Annexes