Évaluation de l'application du paradigme holonique à un système de réservoirs

Carlos Indriago[1,2], Olivier Cardin[1], Odile Bellenguez-Morineau[3], Naly Rakoto[3], Pierre Castagna[1], Edgar Chacòn[2]

1. LUNAM Université, IUT de Nantes – Université de Nantes, IRCCyN UMR CNRS 6597 (Institut de Recherche en Communications et Cybernétique de Nantes)
2 avenue du Prof. Jean Rouxel – 44475 Carquefou
olivier.cardin@irccyn.ec-nantes.fr

2. Universidad de los Andes – La Hechicera Mérida 5101 Venezuela
echacon@ula.ve

3. LUNAM Université, Ecole des Mines de Nantes, IRCCyN UMR CNRS 6597 (Institut de Recherche en Communications et Cybernétique de Nantes)
4 Rue Alfred Kastler – 44300 Nantes
odile.morineau@mines-nantes.fr

RÉSUMÉ. Le paradigme holonique a été largement étudié dans le cadre de la production manufacturière discrète. Une classe différente de systèmes est étudiée ici, les systèmes hybrides, qui englobe l'ensemble des systèmes ayant une évolution continue par morceaux, et dont les changements sont liés à l'évolution de variables discrètes. Dans ce cas, une reconfiguration du système est généralement nécessaire, et le paradigme holonique est une réponse pertinente au besoin de flexibilité en découlant. Cet article propose l'application d'une architecture de référence connue de la littérature discrète au cas des systèmes hybrides. Le modèle proposé est composé de l'union du modèle hybride Contrôleur-Interface-Système et du modèle holonique PROSA. Un cas d'étude s'appuyant sur un système de commutations d'arrivées a été choisi et illustre la pertinence de l'architecture de référence discrète pour modéliser les systèmes hybrides. Une évaluation de performance est proposée en comparaison avec un algorithme d'ordonnancement de la position du serveur.

ABSTRACT. Holonic paradigms have been extensively studied in the context of discrete manufacturing. A different class of systems is studied here, hybrid systems, which includes systems with piecewise continuous evolution and whose changes are related to some discrete variables evolution. In this case, a system reconfiguration is usually necessary, and Holonic paradigm is an appropriate response to the need of flexibility arising. This article proposes the application of a well-known reference architecture of discrete literature to a hybrid

systems case. The proposed model consists in the union of hybrid Controller-Interface-System and Holonic PROSA model. A case study based on a system of arrivals switching was selected and shows the relevance of the discrete reference architecture to model hybrid systems. A performance evaluation is proposed by comparing the holonic control with a scheduling algorithm of the server's position.

MOTS-CLÉS : systèmes hybrides, systèmes holoniques, système de réservoirs, PROSA.

Keywords : Hybrid control systems, holonic manufacturing systems, switch arrival system, PROSA

DOI:10.3166/JESA.49.325-347 © Lavoisier 2016

1. Introduction

Les systèmes de production sont traditionnellement divisés en trois classes principales, i.e. systèmes continus, à évènements discrets et hybrides. La classe des process discrets s'intéresse aux systèmes pour lesquels des produits individuels sont fabriqués en utilisant des opérations variées associées librement (telles que des opérations d'usinage CN ou de fabrication additive par exemple) puis assemblés au sein de lignes d'assemblage de produits semi-finis pour créer un produit final. D'un autre côté, il y a les processus continus qui impliquent des flux continus de matériaux (comme des liquides, des poudres ou de l'énergie) et des services à travers des unités de traitement interconnectées via des tuyaux ou des câbles.

Ainsi, entre les systèmes discrets et continus il y a des systèmes que (Chokshi et McFarlane, 2007) appellent processus semi-continus, qui sont similaires à des processus continus impliquant des flux continus de matériaux et de services, mais dont l'état du système peut être discret, dans le sens où sa dynamique change à des instants précis correspondant à des évènements. Les processus semi-continus sont importants dans ce travail, particulièrement les systèmes appelés « systèmes hybrides (SH) », qui sont classiquement structurés en trois différents niveaux (Koutsoukos *et al.,* 2000), illustrés par la figure 1. Le contrôleur est considéré comme un système à événements discrets (SED). Il reçoit, traite et génère des événements qui sont représentés par des variables discrètes. Le système de production (SP) est un système continu généralement modélisé par des équations différentielles. Le SP reçoit des signaux, traite et génère des sorties qui sont représentées par les variables réelles qui sont généralement continues. Le contrôleur et le SP communiquent via une interface qui traduit les sorties du SP en variables discrètes pour le contrôleur, et les variables de sortie du contrôleur en signaux de commande pour l'entrée du SP. L'interface peut être considérée comme constituée de deux sous-systèmes : le générateur, qui détecte les sorties du SP et génère des variables discrètes représentant des événements du SP, et l'actionneur, qui traduit les ordres discrets générés par le contrôleur en signaux d'entrée du SP.

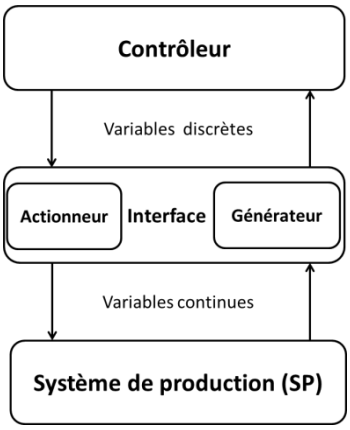

Figure 1. Architecture des systèmes hybrides

Les systèmes hybrides sont classiquement modélisés par des modèles de commutation concernant l'ensemble du système. À cause de cela, tout processus de reconfiguration est complexe parce qu'il doit prendre en compte ce système dans son ensemble. D'où la nécessité de trouver des nouvelles architectures pour ces systèmes hybrides permettant d'améliorer leur flexibilité en facilitant leur reconfiguration. Cette flexibilité a pour objectif d'augmenter l'efficience de l'utilisation des ressources, notamment dans le domaine de la production d'énergie ou de la distribution de ressources naturelles type eau potable, dans un objectif d'utilisation raisonnée et de développement durable.

Les architectures holoniques (PROSA (Van Brussel *et al.*, 1998), ADACOR (Leitão et Restivo, 2006) ou encore HCBA (Chirn et McFarlane, 2000), PROSIS (Pujo *et al.*, 2009) ou ORCA-FMS (Pach *et al.*, 2014)) sont des architectures flexibles permettant une reconfiguration des systèmes et des procédés pour la création de nouveaux produits. Une architecture de fabrication holonique doit faciliter l'(auto-) configuration, l'extension et les modifications du système, en permettant une plus grande flexibilité en augmentant l'espace de décision aux niveaux supérieurs. La structure de l'architecture de référence PROSA (Van Brussel *et al.*, 1998) est construite autour de trois types d'holons de base : holon Ordre, holon Produits, et holon Ressources. Chacun d'eux est responsable d'un aspect pilotage de la fabrication, que ce soit au niveau logistique, au niveau planification, ou au niveau des ressources. Parmi l'ensemble des architectures de référence holoniques pouvant être trouvées dans la littérature, PROSA a la particularité de proposer des holons ayant un niveau conceptuel relativement haut et, tout en étant dédié aux systèmes de production en général, indépendants de l'application visée. À la différence d'autres architectures de la littérature, la définition des holons est en effet relativement peu dédiée au domaine d'application et reste très générale. Pour ces raisons cette architecture a été choisie comme étant la première à être appliquée à un système hybride.

Le quatrième holon de PROSA est l'holon Staff. Celui-ci est plutôt dédié à une optimisation des performances du système en proposant une part de centralisation dans l'architecture distribuée native de l'architecture. La conception de ce type de holon est généralement *ad hoc*, et constitue un niveau d'exigence de performance que nous n'avons pas encore atteint dans nos travaux. Ce type de holon ne sera donc pas évoqué dans cet article et fera partie de perspectives à court terme.

Plusieurs travaux proposent déjà d'utiliser des modèles holoniques pour des systèmes continus (McFarlane, 1995 ; Chacón *et al.*, 2002 ; Chokshi et McFarlane, 2008) Une synthèse de ces travaux présentée en section 2 montre que les auteurs ont développé des holarchies très spécifiques, conçues empiriquement à partir des fonctionnalités attendues du système, sans proposer l'implémentation d'holarchies classiques. L'objectif principal de ce travail est de démontrer que les architectures de référence holoniques, conçues pour fonctionner sur des systèmes discrets, sont également performantes pour l'implémentation de systèmes hybrides.

Cet article propose l'application de l'architecture PROSA sur un système hybride et vérifie son bon fonctionnement. L'objectif est de montrer les bases de la modélisation du système hybride selon les trois holons de base de PROSA, section 3. Le cas d'étude proposé est très simple et fait partie des cas les plus utilisés de la communauté des systèmes hybrides : « système de commutation d'arrivées », proposé initialement par Chase *et al.*, (1993) et présenté dans la section 4, puis modélisé selon ce qui a été présenté précédemment dans la section 5. La section 6 propose une simulation de cet exemple en Java et les résultats qui montrent le potentiel de l'architecture utilisée pour le contrôle et la reconfiguration du système.

2. État de l'art

L'application des systèmes holoniques sur des systèmes continus a déjà été étudiée par quelques auteurs. Un cadre de contrôle pour systèmes hybrides (figure 2) est défini par (McFarlane, 1995). L'objectif du système de contrôle a la particularité de convertir en interne les objectifs ou les exigences en un ensemble de paramètres ou de trajectoires de comportement admissibles, via une de fonction de décision donnée, D. L'action de contrôle effective est alors réalisée sur le processus P. La terminologie utilisée dans la figure 2 est la suivante :

P - process (+ modèle + contrôleur)

D - Problème de prise de décision

ϕ - Principe de prise de décision

M – ensemble de décisions alternatives

U – ensemble d'évènements alternatifs

Y – ensemble de sorties possibles

$m \in M$, $u \in U$, $y \in Y$

$P: M \times U \rightarrow Y$

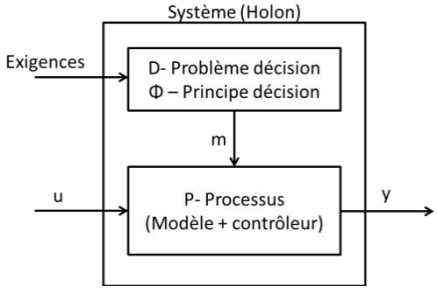

Figure 2. Cadre de contrôle pour SH

Le développement de problèmes de décision appropriés est la clé de l'intégration de l'activité de contrôle au sein des systèmes holoniques. Les instances de ces holons peuvent avoir des objectifs potentiellement différents selon l'environnement dans lequel ils opèrent. Des mécanismes de prise de décision basés sur l'optimisation distribuée sont très appropriés pour les systèmes holoniques et des méthodes de référencement pourraient aider à l'intégration des algorithmes de contrôle basés sur l'optimisation des opérations de fabrication.

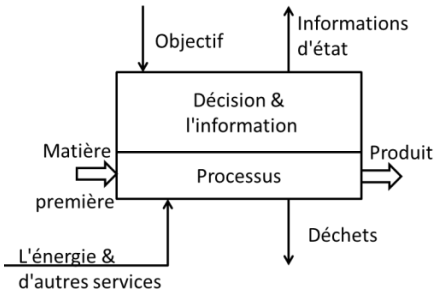

Figure 3. Unité de production

Une autre architecture holonique développée pour les systèmes continus est celle proposée par (Chacón *et al.*, 2002). Il est proposé une structure holonique basée sur un système de prise de décision appelé unité de production (figure 3). Cette unité est la base pour une structure exploitant la caractéristique fractale des systèmes holoniques. L'agrégation des unités de production est alors appelé système de production complexe. L'unité de production a des entrées et des sorties physiques (matières premières, produits finis, services) et logiques (objectifs, bulletins d'information). De plus, cette unité de production suit une structure holonique composée de trois holons de bases appelés holon ressource, holon mission-produit et holon ingénierie.

On peut enfin mentionner l'ouvrage de Chokshi et McFarlane (2007) qui développe une approche de coordination distribuée pour des processus de contrôle reconfigurables. Dans ce travail, une architecture de contrôle est développée en utilisant des éléments de base comme l'élément produit, l'élément unité, l'élément d'en-tête et l'élément de service. La figure 4 montre ces éléments et comment ils interagissent. Même si elle ne traite pas explicitement du paradigme holonique, l'objectif de cette architecture est d'avoir un système de contrôle basé sur des modèles d'interaction reconfigurables et flexibles, semblables aux systèmes holoniques, et qui peuvent être implémentés dans des procédés continus et semi-continus.

Plusieurs approches ont donc été appliquées pour le contrôle de systèmes hybrides. Ces approches étant systématiquement ad hoc, elles peuvent difficilement bénéficier des nombreux résultats obtenus sur les systèmes discrets, ce qui serait facilité en appliquant les architectures de référence classiques. L'objectif des sections suivantes est d'étudier l'opportunité d'appliquer une architecture discrète classique à un système hybride, en commençant par la description du système choisi.

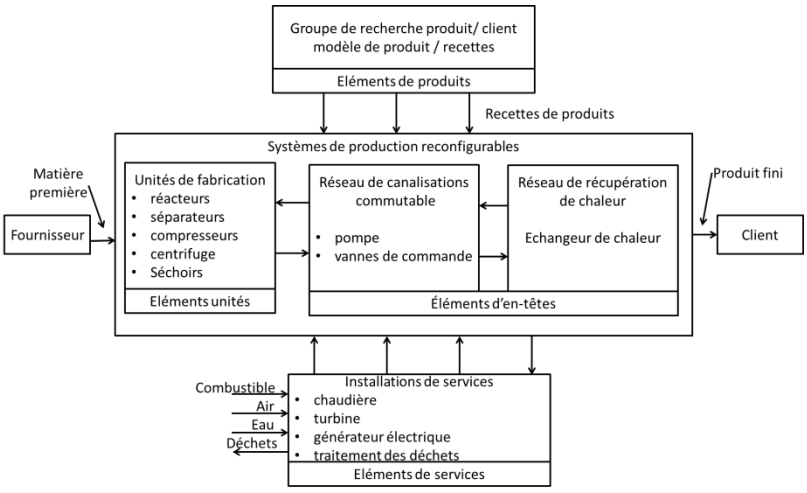

Figure 4. Éléments du processus de base dans un contrôle de processus reconfigurable

3. Méthodologie

Les caractéristiques des systèmes holoniques comme la flexibilité, la reconfiguration et l'autodiagnostic sont des caractéristiques souhaitables sur les systèmes hybrides. L'objectif de cet article est de montrer comment adapter une architecture holonique classique comme PROSA, initialement développée pour les systèmes discrets, à un système hybride. Pour cela, les sections suivantes définissent l'adaptation des holons Ordre, Produit et Ressource pour les systèmes hybrides.

3.1. Modélisation holonique du système

Une architecture hybride est composée de trois parties, comme mentionné dans les paragraphes précédents : le contrôleur, l'interface et le système de production. Dans le contrôleur du système hybride proposé le modèle est représenté par les holons Produit et Ordre, l'interface est représentée par la partie logique du holon Ressource et le système de production est représenté par la partie physique du holon Ressource. La figure 5 illustre le passage de la décomposition classique des systèmes hybrides en une modélisation suivant le paradigme holonique.

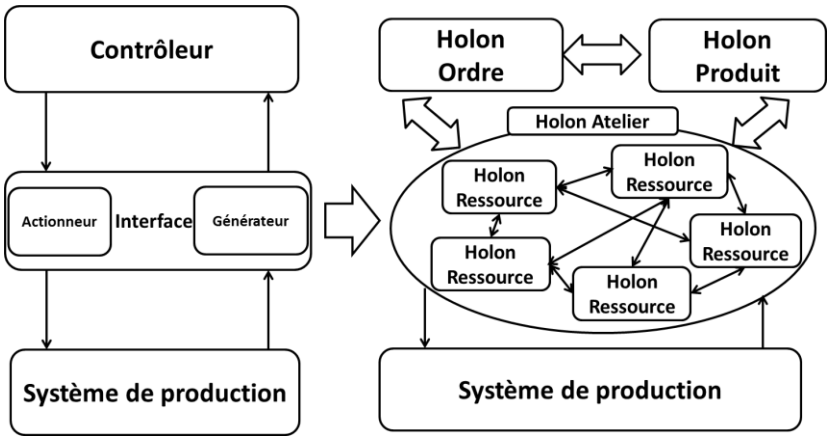

Figure 5. Architecture holonique d'un système hybride

Ainsi, les données qui transitent depuis le système de production vers la partie logique du holon ressource et vice versa sont des données continues, tandis que les données vers le holon Ordre depuis la partie logique du holon Ressource sont des données discrètes. Cette caractéristique permet à l'holon Ordre d'avoir des caractéristiques presque invariables par rapport à une utilisation classique de PROSA. En revanche, toute la partie régulation des variables continues se tient dans la partie physique du holon ressource.

3.2. Définition du holon Produit

Selon PROSA (Van Brussel *et al.*, 1998), le holon Produit contient des connaissances du processus et du produit pour assurer la réalisation correcte du produit avec une qualité suffisante. La spécification du produit utilisé pour cet article est une spécification orientée services, comme proposé dans (Quintanilla *et al.*, 2014), où le produit est défini par tous les services nécessaires pour l'obtenir (figure 6).

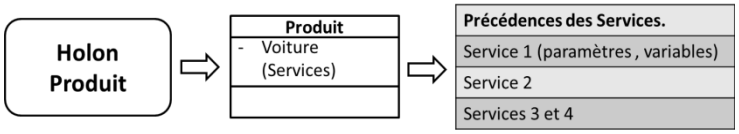

Figure 6. Modèle du produit orienté services.

3.3. Définition du holon Ressource

L'holon ressource est composé par une partie physique et une partie logique. La partie logique est une abstraction de la partie physique qui a tous les modèles de conversion d'état discret vers l'état continu et vice-versa (figure 7). Plusieurs modèles peuvent coexister dans le holon Ressource et représentent les fonctionnalités comme le modèle de détection de défaillance, le modèle de comportement souhaité (réseau de Petri hybride par exemple (David et Alla, 2010)) et tous les modèles qui sont inhérents au paradigme holonique comme le modèle de négociation par exemple.

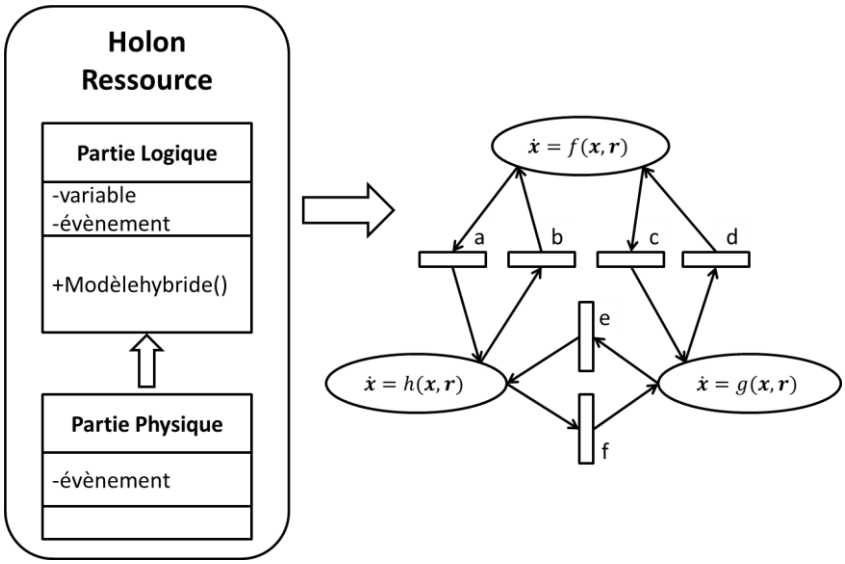

Figure 7. Exemple de modèle de holon Ressource

Par nature, les HCS sont de grands systèmes, constitués de beaucoup de composants. De ce fait, beaucoup de holons Ressource sont nécessaires pour contrôler le système. Il est nécessaire de déterminer la plus petite taille d'un holon Ressource. Ces plus petits holons Ressource, appelés ressources atomiques dans la suite de ce document, peuvent être définis comme suit :

Une ressource atomique est l'agrégation maximale d'éléments pour laquelle le système d'équations différentielles peut être inversé, dans un délai court relativement à la dynamique du système considéré.

Concernant les ressources composées, le mécanisme de négociation doit permettre de déterminer la meilleure solution en interrogeant récursivement les ressources composées jusqu'à atteindre les ressources atomiques. Les ressources sont parties prenantes de ces mécanismes de négociation. Leur fonction est d'évaluer et transmettre à l'holon Ordre les meilleures valeurs de variables possibles pour obtenir les fonctions et services désirés. Les mécanismes de négociation utilisent un modèle de communication, où le langage de négociation des ressources est basé sur le paramétrage de services, demandés et fournis par des ressources, comme proposé dans (Gamboa Quintanilla *et al.,* 2014). Les holons Ressource ont également la responsabilité de la commande en ligne des équipements, i.e. le rôle de contrôleur du système présenté précédemment dans la modélisation classique des HCS.

3.4. Définition du holon Ordre

La fonction et la structure du holon Ordre restent les mêmes que celles proposées par PROSA. En effet, l'architecture proposée dans cet article garantit que toutes les informations et les modèles à être manipulés par le holon Ordre sont discrets ou discrétisés. Le processus de discrétisation des variables de contrôle est classiquement utilisé dans le contrôle des systèmes hybrides. La différence ici est que chaque holon n'intègre que le modèle de comportement d'une partie du système, qui est plus simple à modéliser. De ce fait, le comportement global du système hybride est donc plus simple à discrétiser.

L'ordre définit l'évolution du système de production. L'ordre doit être négocié à partir d'un objectif de production en utilisant le modèle de produit. Un objectif de production est transformé en un ordre qui spécifie les services et ses paramètres vis-à-vis des services fournis par les ressources atomiques (au sens de l'holarchie), les conditions de qualité à atteindre et les indicateurs permettant de définir lorsque chaque étape de production est terminée.

Le holon Ordre est globalement responsable du planning du système. Il est très représentatif du fonctionnement dual de chaque holon : une partie du holon est dédiée au monitoring du système, tandis que l'autre est dédié aux mécanismes de négociation en mode anticipation afin de déterminer le futur planning et les holarchies à créer au sein du système.

4. Cas d'étude

Cette section décrit un exemple de système hybride, appelé système de commutations d'arrivée (de flux), et présenté initialement dans (Chase *et al.,* 1993). Plusieurs auteurs ont utilisé ce système comme illustration de travaux divers, notamment sur la détection de défaillance, car il a la particularité d'avoir un comportement chaotique lié aux conditions initiales. Dans notre cas, il a été choisi

car sa simplicité nous permet de calculer une solution optimale à laquelle se comparer et la possibilité de reconfigurer simplement le système, notamment en modifiant le nombre de réservoirs considérés, nous permet d'étendre le cas d'étude pour montrer la puissance de l'approche sur un cas plus complexe, notamment en qualifiant la flexibilité du système de contrôle proposé.

4.1. Présentation du système

Le système est constitué de n réservoirs et un serveur. Le fluide est retiré du réservoir i à un taux fixe $\rho_i > 0$. Pour compenser, le serveur fournit du fluide à un réservoir sélectionné au taux de v. Nous supposons que le système est fermé, de façon que $\sum_{i=1}^{n} \rho_i = v$. L'emplacement du serveur est une variable de commande, et peut être sélectionné en utilisant une politique de réalimentation. Le déplacement du serveur modifie la topologie du flux, et par conséquent permet le contrôle du niveau du réservoir. La figure 8 montre le système avec n = 3 et le serveur localisé sur le réservoir 1.

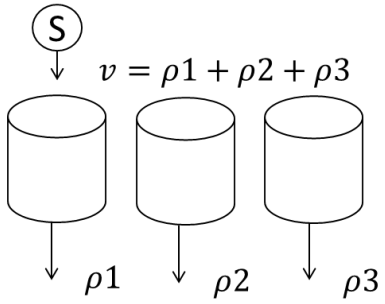

Figure 8. Système de commutations d'arrivée avec n = 3

La politique de contrôle proposée est une politique de seuil de la forme suivante : « Attribuer un seuil pour chaque réservoir, et instantanément déplacer le serveur vers un réservoir dans lequel le niveau est inférieur au seuil assigné ». Il est à noter que l'emplacement du serveur est sélectionné sur la base d'une observation quantifiée de l'état du réservoir, et le mouvement du serveur est déclenché par un « événement discret » (par exemple quand un réservoir se vide). Dans la version la plus simple de ce dispositif, on prend tous les seuils égaux à zéro, et on passe le serveur d'arrivée à chaque fois qu'un réservoir est vide.

Soit $x_i(t)$ la quantité de liquide dans un réservoir i au temps $t > 0$, et soit $x(t) = (x_1(t), \dots, x_n(t))$. A *t=0*, nous supposons que $x_i(0) \geq 0$ avec $\sum_{i=1}^{n} x_i(0) = h$. Nous appelons $x(t)$ l'état des réservoirs à l'instant *t.* Si à l'instant *t* le serveur se trouve en position *j*, alors le serveur restera à *j* jusqu'à ce qu'un autre réservoir soit vide. Cet événement aura lieu après un temps $\tau = min_{i \neq j} \{x_i(t)/\rho_i\}$. Pour $t \leq s \leq t + \tau$ l'état des réservoirs est déterminé par l'ensemble d'équations linéaires :

$$x_i(s) = \begin{cases} x_i(t) - \rho_i(s-t), & si\ i \neq j; \\ x_i(t) + (v - \rho_i)(s-t), & si\ i = j; \end{cases}$$

Ainsi, l'idée du contrôle est de fournir du fluide dans les réservoirs de manière à ce qu'aucun réservoir ne déborde et d'éviter que deux ou plusieurs réservoirs soient vides en même temps. Une analogie peut être faite entre le système de commutations d'arrivée et un système de distribution d'eau où le serveur est représenté par le château d'eau et les réservoirs sont représentés par les réservoirs dans les quartiers. L'objectif est qu'aucun réservoir de quartier ne soit vide ni ne déborde. De plus, s'il y a deux réservoirs de quartier vides, alors il y aura un problème de manque d'eau au niveau du quartier dont le réservoir ne sera pas rempli par le serveur.

4.2. Problème de pilotage

La séquence d'états de niveaux $\{x(n)\}$ appartient à l'ensemble. $X = \{x: \sum_i x_i = h,\ x_i \geq 0, et\ pour\ certains\ j,\quad x_j = 0\}$. Si $G_j: X \to X$ désigne la fonction qui décrit la transformation de X qui résulte en plaçant le serveur à l'emplacement j jusqu'à ce qu'un autre réservoir se vide, alors

$$G_j(x) = x + \left(\min_{k \neq j}\left\{\frac{x_k}{\rho_k}\right\}\right)(v_j - \rho)$$

Où ρ est le vecteur des ρ_i, et v_j est le vecteur de zéros à l'exception d'une valeur v dans la *j-ième* position. Pour n = 3 le comportement de G est représenté par la figure 9, où il y a un point p_1 de coordonnées $\left(0, \frac{h*\rho_2}{\rho_2+\rho_3}, \frac{h*\rho_3}{\rho_2+\rho_3}\right)$ et si $\left(\frac{x_2}{\rho_2}\right) = \left(\frac{x_3}{\rho_3}\right)$ alors $G = (h, 0, 0)$ où deux réservoirs sont vides en même temps.

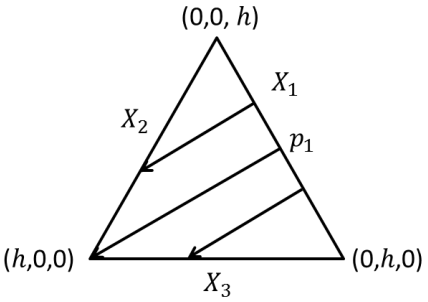

Figure 9. Comportement de G pour n = 3

4.3. Solution développée en hybride

Quand deux réservoirs sont vides en même temps, le système devient non déterministe et le contrôle classique ne donne aucune solution. Pour éviter cette situation, les systèmes hybrides utilisent le modèle du système entier. Le contrôle

utilise le modèle d'ensemble de réservoirs, (Labinaz *et al.*, 1997 ; Ushio *et al.*, 1995 ; Labinaz *et al.*, 1996), qui est un modèle complexe rigide, difficile à modifier. Ainsi, si un réservoir est ajouté, le modèle de tout le système doit être changé pour reconfigurer le contrôle, caractéristique qui montre la rigidité du système classique. Enfin ce modèle est fortement dépendant des conditions initiales (Chase *et al.*, 1993 ; Tian, 2005). Ces travaux montrent que, pour certaines conditions initiales, le contrôle hybride « classique » ne fonctionne pas, et propose donc de limiter l'espace de conditions initiales admissibles.

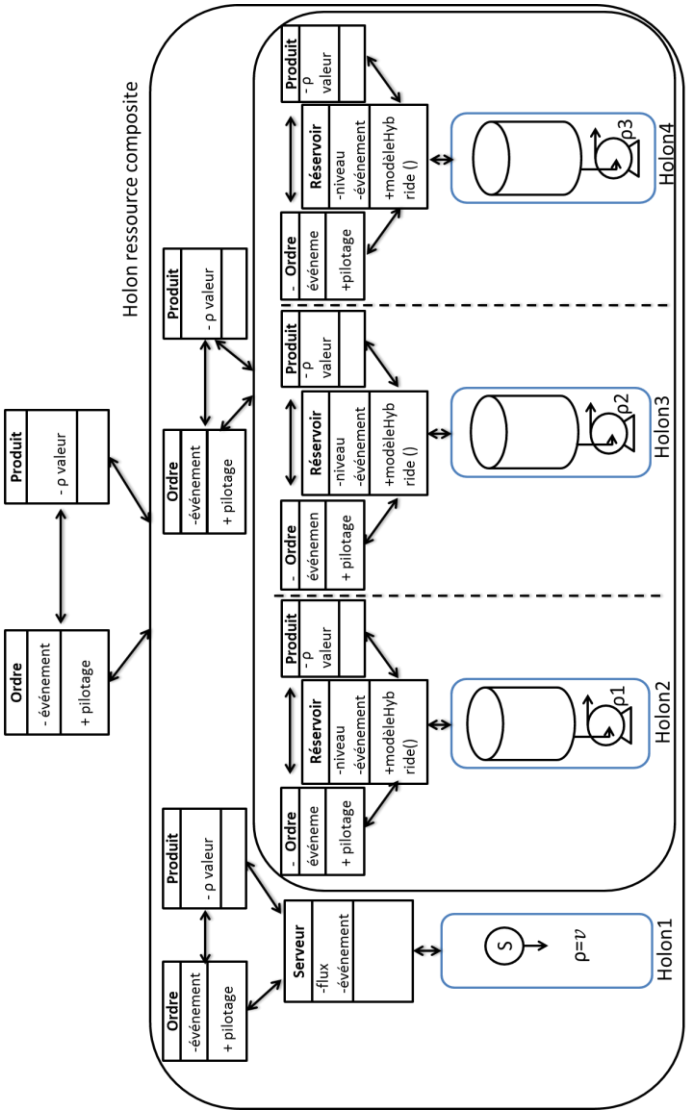

Figure 10. Modélisation holonique du système

5. Application au cas d'étude

Dans cette section, plusieurs notations sont définies :

– i : réservoir n°i ;

– N : Nombre de réservoirs ;

– Si = (i, date) : planification de la position du serveur sur le réservoir i au pas de temps noté date ;

– P : Planning, i.e. ensemble des planifications ; en corollaire, P(date) désigne le réservoir au-dessus duquel le serveur doit se déplacer au pas de temps noté date ;

– tp : date courante ;

– i_s : réservoir ayant actuellement le serveur au-dessus de lui.

Le système est modélisé suivant la représentation de la figure 10. Cette modélisation est conforme à la description faite dans (Indriago *et al.*, 2015). L'holon ordre est en charge du respect de la condition de débit de fuite des ressources réservoir exprimée dans le holon produit. Pour assurer cela, deux fonctions sont implémentées :

1) En mode prédiction, il négocie avec chaque ressource réservoir et avec la ressource serveur pour établir le planning du serveur (phases (2), (3), (7)), comme exprimé dans l'algorithme de la figure 11.

L'idée principale est d'essayer de maximiser la durée pendant laquelle le serveur reste dans la même position, et donc minimiser le nombre de commutations, tout en évitant la situation où deux réservoirs sont vides en même temps. De ce fait, les réservoirs essayent d'anticiper la réservation du serveur si la date à laquelle ils seront vides est déjà planifiée pour un autre réservoir (4). Une durée de réservation est également définie (1). Cette durée représente la durée pendant laquelle un réservoir essaie de réserver le serveur. Si aucune solution ne peut être trouvée par le système avec la durée de réservation définie a priori, alors l'algorithme boucle en réduisant la durée de réservation (5). Si la durée de réservation diminue jusqu'à être inférieure à la durée d'un pas de calcul, alors le planning est totalement vidé et un ordonnancement est totalement recommencé en diminuant la durée de réservation initiale (6).

2. En mode temps-réel, il déclenche le changement de position du serveur, suivant l'algorithme présenté dans la figure 12. Chaque fois que le serveur change de position, toutes les réservations du nouveau réservoir sont retirées du planning et une réservation pour le réservoir sur lequel le serveur était au pas de temps précédent est insérée dans le planning. Il est notable qu'il ne résulte pas de cette procédure une optimisation globale, mais seulement une prévision prenant en compte le fait que les réservations précédemment insérées dans le planning ne peuvent pas être remises en question.

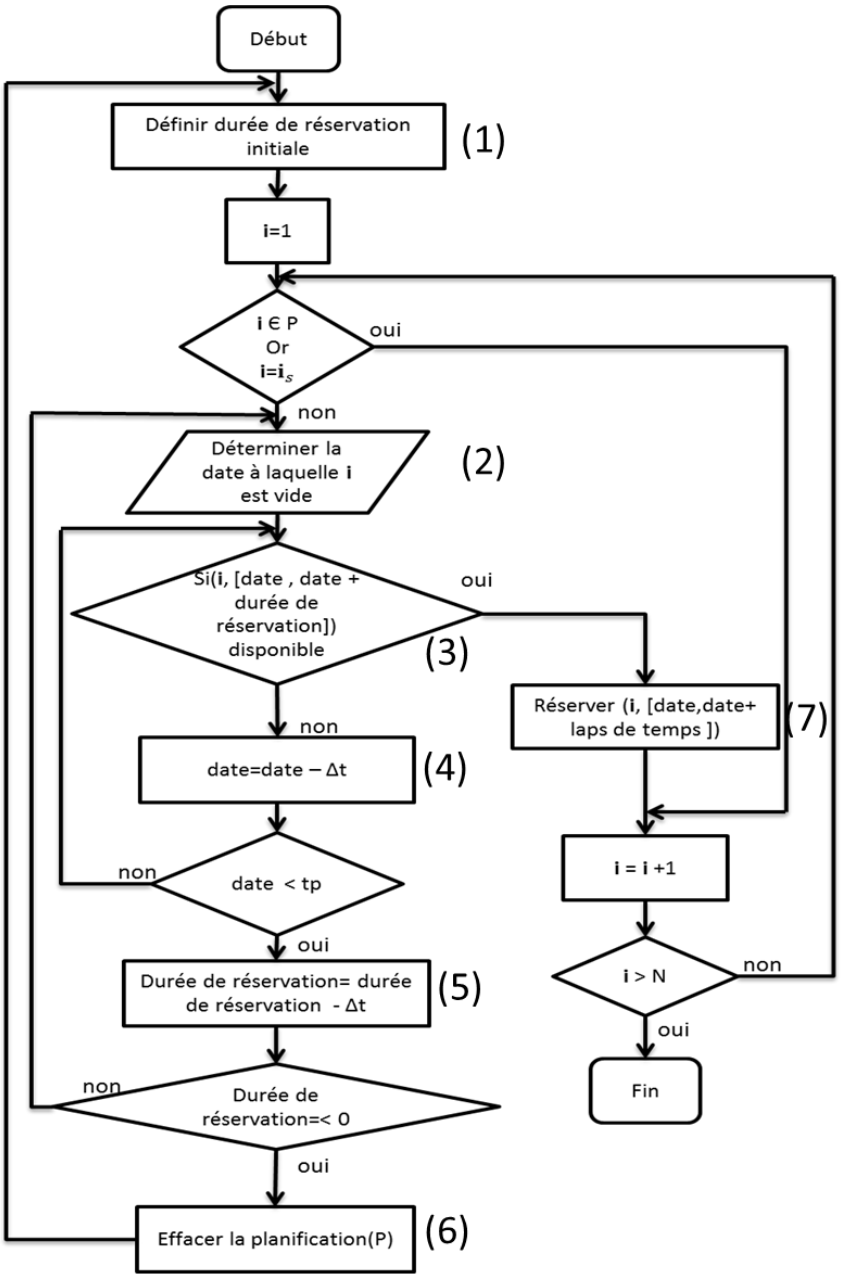

Figure 11. Algorithme d'ordonnancement de la position du serveur

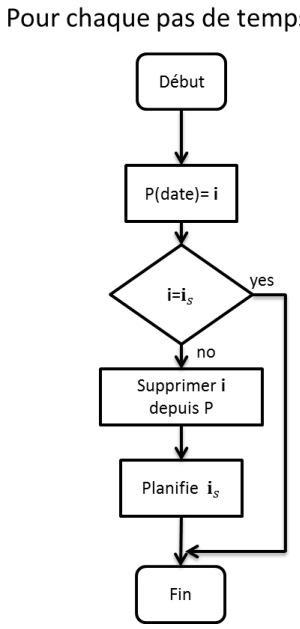

Figure 12. Algorithme de supervision de la position du serveur

6. Évaluation de performance

La section précédente a permis de présenter les heuristiques implémentées dans la structure holonique du système. L'objectif de cette section est de présenter l'application de cette heuristique sur des instances de benchmark et de vérifier que le système a un comportement correct. De plus, la seconde partie de cette section propose d'évaluer la performance du comportement vis-à-vis d'un objectif de minimisation du nombre de changements de place du serveur.

6.1. Définition des instances

Le tableau 1 présente quatre différents scénarios utilisés pour tester l'heuristique. Chaque instance est définie par le niveau initial et le débit de fuite de chaque réservoir. Les réservoirs sont supposés avoir le même volume, normalisé à 1 pour des raisons de lisibilité. Le débit de fuite est exprimé en pourcentage de volume par pas de calcul, i.e. qu'un réservoir à 10 % plein d'eau (niveau initial = 0.1) sera vide après 100 pas de calcul si le débit de fuite est de 0.1 %. Le débit d'alimentation du serveur est par définition égal à la somme de l'ensemble des débits de fuite des réservoirs.

Les scénarios 1 et 2 ont été choisis afin que deux réservoirs soient vides en même temps en partant de conditions initiales différentes. L'objectif de ces scénarios est d'observer le comportement du contrôle holonique par rapport à cette situation, généralement évitée a priori par le contrôle classique. Les scénarios 3 et 4 ont été conçus avec des conditions extrêmes de niveaux initiaux et de débits de fuite, i.e. deux réservoirs très proches du vide et un réservoir ayant une fuite très importante. La différence se fait sur les débits de fuite des réservoirs, situé soit sur le réservoir plein soit sur le réservoir vide.

Tableau 1. Scénarios étudiés

Scénario	Numéro de réservoir	1	2	3
1	Niveau initial	0	0.15	0.15
	Débit de fuite	0.1	0.1	0.1
2	Niveau initial	0	0.135	0.165
	Débit de fuite	0.1	0.09	0.11
3	Niveau initial	0	0.29	0.01
	Débit de fuite	0.28	0.01	0.01
4	Niveau initial	0	0.29	0.01
	Débit de fuite	0.01	0.28	0.01

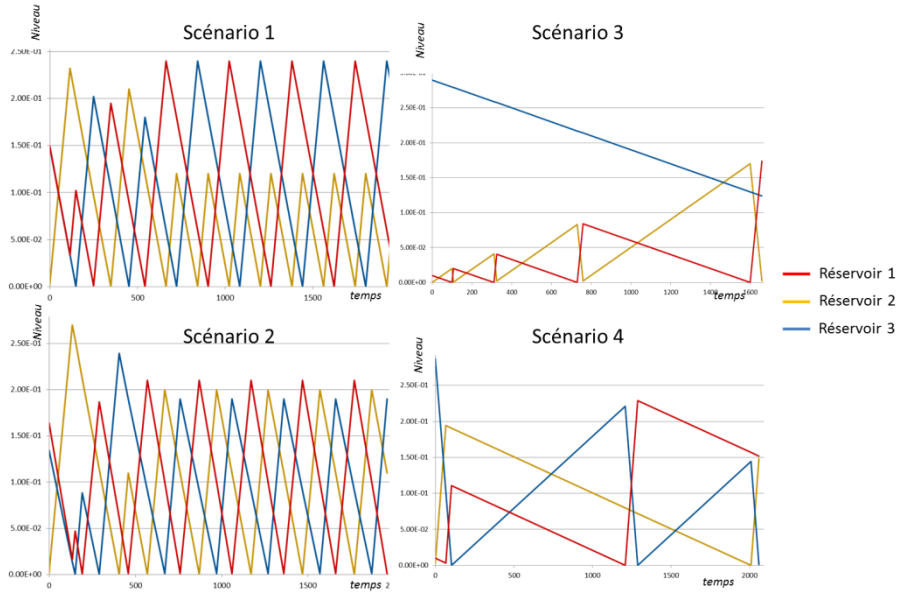

Figure 13. Comportement de l'heuristique

6.2. Comportement de l'heuristique

La première vérification a été de tester l'heuristique sur les instances et vérifier qu'elle pouvait systématiquement fournir une solution au problème. La Figure 13 illustre l'évolution du niveau de chaque réservoir au cours des 2000 premiers pas de calcul sur chaque instance pour une durée de réservation donnée. L'abscisse représente un réservoir vide. Chaque durée de réservation ayant été testée, il n'apparait pas de possibilités d'avoir un réservoir vide. Toutefois, un comportement cyclique tel que celui du scénario 1 de la figure 13 n'a pas été rencontré à chaque simulation. Il n'est donc pas possible de conclure sur une impossibilité pour l'heuristique de ne pas pouvoir trouver de solution. Toutefois, avec un si faible nombre de réservoirs, il semble hautement improbable de se retrouver dans une situation plus difficile que celle proposée en tant que situation initiale des scénarios 3 et 4.

6.3. Modèle Mathématique

Cette section est dédiée à la description de la structure du modèle *Mixed Integer Linear Programming* (MILP) appelée à calculer la solution optimale des instances précédemment présentées. L'ensemble des variables est le suivant :

– Lvi,c est le niveau d'un réservoir i au début d'une phase c ;

– Dc est la durée d'une phase c ;

– Qi,c est la quantité d'eau ajoutée à un réservoir i pendant la phase c ;

– Oi,c est la quantité d'eau perdue par un réservoir i pendant la phase c ;

– Ri,c est un booléen, égal à 1 si le serveur est au-dessus du réservoir i durant la phase c ;

– Swi,j,c est un booléen, égal à 1 si le serveur change de position entre les réservoirs i et j au début de la phase c, 0 autrement.

La formulation du modèle devient donc :

$$Min\ Z = \sum_i \sum_j \sum_c \mathrm{Sw_{i,j,c}} \quad (0)$$

S.t.

$$\forall i, \forall c, Lv_{i,c} \leq \mathrm{LevelMax} \quad (1)$$

$$\forall i, \forall c, Lv_{i,c} \geq \mathrm{LevelMin} \quad (2)$$

$$\forall i, \forall c, Lv_{i,c} = Lv_{i,c-1} + Q_{i,c-1} - O_{i,c-1} \quad (3)$$

$$\forall i, \forall c, Q_{i,c} = \begin{cases} D_c * Q, & if\ R_{i,c} = 1; \\ 0, & otherwise; \end{cases} \quad (3')$$

$$\forall i, \forall c, O_{i,c} = O_c * D_c \quad (3'')$$

$$\forall c, \sum_i R_{i,c} \leq 1 \quad (4)$$

$$\forall i \neq j, \forall c, R_{i,c-1} + R_{j,c} - \mathrm{Sw}_{\mathrm{i,j,c}} \leq 1 \quad (5)$$

La fonction objectif est de minimiser le nombre de changements de position du serveur (0). L'ensemble des contraintes (1) et (2) assure que le niveau d'un réservoir est toujours entre les limites prédéfinies. La contrainte (3) permet d'assurer la mise à jour du niveau des réservoirs durant la phase c, en considérant la fuite et l'ajout éventuel de liquide. La contrainte (3') permet de définir cette quantité ajoutée si le réservoir est en cours de remplissage, i.e. $R_{i,c}$ est égal à 1. La quantité de la fuite $O_{i,c}$ du réservoir i pendant la phase c est mise à jour à l'aide de la contrainte (3''). La contrainte (4) permet d'indiquer qu'à chaque phase, le serveur ne peut pas remplir plus d'un réservoir à la fois.

Finalement, la contrainte (5) assure le lien entre le changement de position et le remplissage des réservoirs. Des contraintes additionnelles sont définies pour linéariser les expressions, lier les variables et renforcer le modèle ; néanmoins, ce modèle atteint sa limite pour un très court horizon temporel à cause du nombre de variables considérées, le nombre de changements nécessaires à considérer augmentant le nombre de variables. Un autre modèle, directement indexé sur le temps, a également été testé, mais le nombre de variables étant encore supérieur, les limites de taille de problèmes adressables sont d'autant plus petites. Cette modélisation a été intégrée à l'architecture générale programmée en Java grâce à CPlex.

6.4. Construction des expérimentations

Un paramètre de chaque méthode a été identifié comme influençant potentiellement la qualité des résultats :

– Les temps de calcul de l'algorithme MILP sont influencés par la borne supérieure du nombre de changements de position du serveur, donnée à l'algorithme a priori des calculs. De ce fait, deux possibilités ont été testées, pour tester si les résultats de simulation pourraient être utilisés comme une borne supérieure correcte.

– Pour le contrôle holonique, la durée de réservation initiale influence le nombre de changements obtenu. Si la durée de réservation est trop courte, le contrôle tend à être nerveux, c'est-à-dire changeant de position à pratiquement chaque pas de temps. Si a contrario la durée de réservation est trop longue, le premier réservoir planifié va tellement contraindre le planning qu'il va empêcher le système de correctement anticiper les problèmes possibles.

Un programme Java a été créé de manière à utiliser chaque instance 104 fois:

1) 50 simulations sur 1 000 pas de calcul du contrôle holonique avec une durée de réservation initiale incrémentant de 1 à 50 ;

2) 50 simulations sur 2000 pas de calcul du contrôle holonique avec une durée de réservation initiale incrémentant de 1 à 50 ;

3) Calcul sur 1 000 pas de temps de la solution optimale utilisant l'algorithme MILP sans borne supérieure prédéfinie ;

4) Calcul sur 2 000 pas de temps de la solution optimale utilisant l'algorithme MILP sans borne supérieure prédéfinie ;

5) Calcul sur 1 000 pas de temps de la solution optimale utilisant l'algorithme MILP en utilisant la meilleure solution trouvée par la simulation en tant que borne supérieure ;

6) Calcul sur 2 000 pas de temps de la solution optimale utilisant l'algorithme MILP en utilisant la meilleure solution trouvée par la simulation en tant que borne supérieure.

Seuls les temps de calcul de l'algorithme MILP ont été relevés, étant donné que les temps de simulation étaient tous inférieurs à la milliseconde.

6.5. Résultats et discussion

La figure 14 montre l'évolution de la performance du contrôle holonique en fonction de la durée de réservation initiale choisie. L'influence attendue de ce paramètre est confirmée, mais ne se démontre pas dans l'ensemble des scénarios (Scénario 3 donne toujours les mêmes résultats de simulation par exemple, ce qui s'explique probablement par le faible nombre de changements nécessaires, et donc un ensemble de solutions possibles restreint).

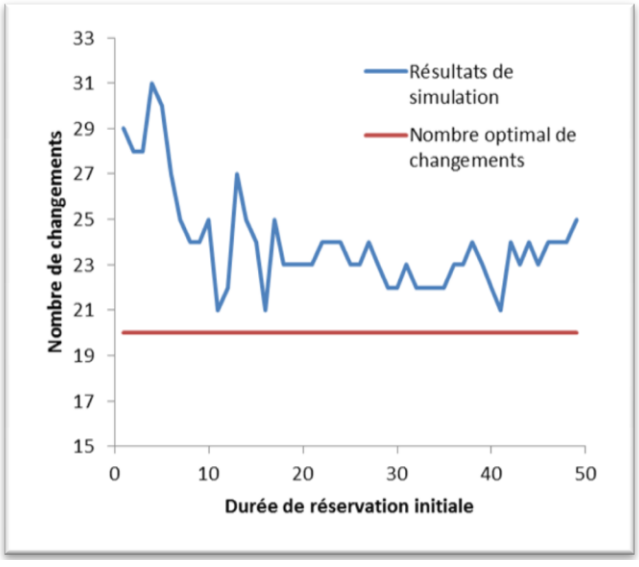

Figure 14. Influence de la durée de réservation initiale sur les résultats de simulation du scénario 2 sur 2000 pas de calcul

Le tableau 2 synthétise les résultats de simulation. Étant donné qu'ils évoluent avec la durée de réservation initiale choisie, il a été choisi de mettre en avant la ou les durées de réservation donnant la meilleure performance, puis l'évolution de la performance en fonction de la durée de réservation relatives à des dégradations de performance de respectivement 0, 20, 40, 60 et plus de 60 % par rapport à l'optimal calculé par le MILP.

Le premier résultat à noter est que le contrôle holonique trouve toujours une solution. Un autre élément important est que ce contrôle a trouvé la solution optimale au moins une fois dans 6 cas sur 8. Lorsque l'on regarde plus précisément les instances pour lesquelles les solutions optimales n'ont pas été trouvées, il est logique de remarquer que ce sont les instances où le nombre de changements optimal est le plus grand.

En ne considérant que les scénarii 1 et 2 (les 3 et 4 ont un nombre de changements trop faible), la dégradation de performance du contrôle holonique est dans 80 % des cas inférieure à 40 %. Cette performance est remarquablement bonne compte tenu du fait que les durées de réservation initiales ont été choisies arbitrairement.

Tableau 2. Résultats de simulations et de calculs

Scénario	Horizon de calcul (en nbre de pas)	Nombre optimal de switches calculés (MILP)	Meilleurs timelapses	Pourcentage de timelapses de la commande holonique classés en fonction de la dégradation du nombre de switches obtenus					Temps de calcul (sec)	
				0%	]0%; 20%]	]20%; 40%]	]40%; 60%]	>60%	Avec borne sup.	Sans borne sup.
1	1000	10	34	2.04	77.55	12.25	8.16	0	0.31	3.76
	2000	20	29/32/34/ 35/39/41/ 43/46	0	67.35	30.61	2.04	0	1053	4338
2	1000	10	29/33/34	6.12	69.39	14.29	8.16	2.04	0.75	10.47
	2000	20	16/41	0	75.51	18.37	6.12	0	869	4989
3	1000	6	Tous	100	0	0	0	0	0.37	1.48
	2000	8	Tous	100	0	0	0	0	0.48	2.86
4	1000	2	31-49	38.8	0	0	4.08	57.14	0.2	0.3
	2000	4	21-37	34.7	0	4.08	42.86	18.37	0.11	1.7

À propos de ces durées de réservation, aucune corrélation avérée n'a pu être établie entre les meilleures durées de réservation pour 1 000 pas et 2 000 pas pour chaque scénario. De ce fait, une étude plus approfondie doit être réalisée pour trouver une méthodologie appropriée de définition de la meilleure durée de réservation initiale à utiliser dans un contexte d'utilisation en ligne.

D'un autre côté, l'algorithme MILP a également toujours trouvé une solution. Toutefois, les temps de calcul obtenus sont extrêmement longs, et augmentent très rapidement entre 1 000 et 2 000 pas. Des tests plus approfondis ont montré que, sur un ordinateur i7 équipé de 4Go de mémoire RAM dont 1.5 Go disponibles au début de chaque test, l'algorithme a été à court de mémoire avant de trouver une solution sur les scénarios 1 et 2. Ceci empêche l'utilisation d'un tel algorithme en ligne, par exemple dans le cas d'une reconfiguration du système.

Plus anecdotique, comme attendu, l'utilisation des résultats de simulation en tant que borne supérieure du MILP montre une grande amélioration du comportement de l'algorithme et une diminution sensible des temps de calcul, mais finalement pas assez importante pour imaginer une utilisation réactive.

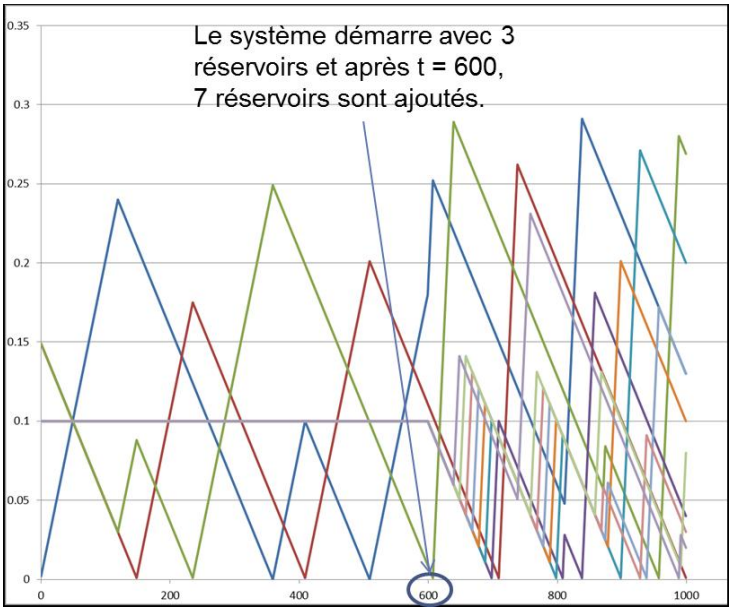

Figure 15. Comportement de l'heuristique lors d'une reconfiguration du système

Enfin, la figure 15 illustre le comportement de l'heuristique lors de la reconfiguration du système. Lors de ce scénario, le système démarre avec 3 réservoirs, puis après 600 pas de calcul, 7 nouveaux réservoirs sont ajoutés. Bien évidemment, le niveau initial de chacun des réservoirs ajoutés doit être choisi de manière à assurer à l'instant de la reconfiguration que 2 réservoirs ne soient pas vides simultanément. La configuration choisie ici est que les 7 réservoirs ajoutés sont pleins à 10 % de leur capacité. À partir de la reconfiguration, le débit d'alimentation du serveur est également modifié afin de satisfaire à la contrainte de niveau global du système constant. Le comportement de l'heuristique montre des capacités de reconfiguration permettant d'envisager une utilisation en ligne, avec

une planification en ligne permettant de prendre en compte très rapidement la nouvelle configuration du système et de remettre en cause le planning préétabli. Avant validation définitive, une étude de sensibilité sur le temps de réponse de l'heuristique dans diverses configurations pourra être menée.

7. Conclusion

Ce travail introduit une évaluation de performance d'une architecture de contrôle holonique sur un système dynamique hybride. Un cas d'étude classique de la littérature des systèmes dynamiques hybrides a été choisi, en l'occurrence un système de commutation d'arrivée appliqué à un cas de réservoirs. Un comportement prédictif-réactif simple, basé sur des mécanismes d'ordonnancement en ligne, a été défini avec l'objectif de minimiser le nombre de commutation du système. L'évaluation de performance consiste alors en une analyse de sensibilité des paramètres de l'heuristique proposée et une comparaison avec la solution optimale calculée à l'aide d'un modèle mathématiques.

Les résultats montrent une bonne performance du contrôle holonique en termes d'optimalité, mais surtout une bien meilleure efficacité que le modèle mathématiques en termes de temps de calcul, et donc en termes de flexibilité. Ceci est un point crucial pour envisager d'implémenter une solution de pilotage en ligne sur un système potentiellement reconfigurable. Les performances de l'heuristique mises en avant par les premières expérimentations présentées dans cet article sont suffisamment intéressantes pour orienter de futures investigations sur une amélioration de l'heuristique et une évaluation de performance plus poussée afin notamment de valider des éléments comme le temps de réaction de l'heuristique.

Les perspectives de ce travail s'orientent vers la définition d'un algorithme plus performant sur de plus larges instances et la définition d'une méthodologie de choix de la durée de réservation initiale, durée de réservation qui pourrait même évoluer dynamiquement dans le cas d'un changement du nombre de réservoirs du système par exemple.

Remerciements

Ce travail a été partiellement financé par le programme ECOS-Nord V11M01 (Ministère des Affaires Étrangères français et FONACIT Ministry of Science and Technology – Venezuela.)

Bibliographie

Chacón E., Besembel I., Narciso F., Montilva J., Colina E. (2002). An integration architecture for the automation of a continuous production complex. *ISA Trans.* 41, 95–113. doi:10.1016/S0019-0578(07)60205-5

Chase C., Serrano J., Ramadge P.J. (1993). Periodicity and chaos from switched flow systems: contrasting examples of discretely controlled continuous systems. *IEEE Trans. Autom. Control* 38, 70–83. doi:10.1109/9.186313

Chirn J.-L., McFarlane D.C. (2000). A holonic component-based approach to reconfigurable manufacturing control architecture. 11th *International Workshop on Database and Expert Systems Applications*, Proceedings. p. 219–223. doi:10.1109/DEXA.2000.875030

Chokshi N., McFarlane D. (2008). A distributed architecture for reconfigurable control of continuous process operations. *J. Intell. Manuf.* 19, 215–232.

Chokshi N., McFarlane D. (2007). *A Distributed Coordination Approach to Reconfigurable Process Control*. Springer.

David R., Alla H. (2010). *Discrete, continuous, and hybrid Petri nets.* Springer.

Indriago C., Cardin O., Rakoto N., Chacón E., Castagna P. (2015). Application of Holonic Paradigm to Hybrid Processes: Case of a Water Treatment Process. *Service Orientation in Holonic and Multi-Agent Manufacturing*, Studies in Computational Intelligence. Springer, p. 39-48.

Koutsoukos X.D., Antsaklis P.J., Stiver J.A., Lemmon M.D. (2000). Supervisory control of hybrid systems. *Proc. IEEE* 88, 1026–1049. doi:10.1109/5.871307

Labinaz G., Bayoumi M.M., Rudie K. (1997). A survey of modeling and control of hybrid systems. *Annu. Rev. Control* 21, 79–92. doi:10.1016/S1367-5788(97)00019-9

Labinaz G., Rudie K., Ricker L., Sarkar N., Bayoumi M.M. (1996). *A hybrid system investigation of fluid-filled tanks* (Technical Report No. 96-01). Department of Electrical and Computer Engineering, Queen's University, Canada. A summary of this work was presented at 12th International Symposium of Mathematical Theory of Networks and Systems 1996, St, Louis, MO.

Leitão P., Restivo F. (2006). ADACOR: A holonic architecture for agile and adaptive manufacturing control. *Comput. Ind.* 57, 121–130. doi:10.1016/j.compind.2005.05.005

McFarlane D.C. (1995). Holonic manufacturing systems in continuous processing: concepts and control requirements, in *Proceedings of ASI.* Citeseer, p. 282ą273.

Pach C., Berger T., Bonte T., Trentesaux D. (2014). ORCA-FMS: a dynamic architecture for the optimized and reactive control of flexible manufacturing scheduling. *Comput. Ind.* 65, 706–720. doi:10.1016/j.compind.2014.02.005

Pujo P., Broissin N., Ounnar F. (2009). PROSIS: An isoarchic structure for HMS control. *Eng. Appl. Artif. Intell.*, Distributed Control of Production Systems 22, 1034–1045. doi:10.1016/j.engappai.2009.01.011

Quintanilla F.G., Cardin O., Castagna P. (2014). Product Specification for Flexible Workflow Orchestrations in Service Oriented Holonic Manufacturing Systems, in: Borangiu T., Trentesaux D., Thomas A. (Eds.), *Service Orientation in Holonic and Multi-Agent Manufacturing and Robotics*, Studies in Computational Intelligence. Springer International Publishing, p. 177–193.

Tian Y.-P. (2005). Delayed feedback control of chaos in a switched arrival system. *Phys. Lett. A* 339, 446–454. doi:10.1016/j.physleta.2005.03.061

Ushio T., Ueda H., Hirai K. (1995). Controlling chaos in a switched arrival system. *Syst. Control Lett.* 26, 335–339. doi:10.1016/0167-6911(95)00032-1

Van Brussel H., Wyns J., Valckenaers P., Bongaerts L., Peeters P. (1998). Reference architecture for holonic manufacturing systems: PROSA. *Comput. Ind.* 37, 255–274. doi:10.1016/S0166-3615(98)00102-X