Échange de tâches pour la réduction de la durée moyenne de réalisation

Ellie Beauprez, Anne-Cécile Caron, Maxime Morge, Jean-Christophe Routier

Échange de tâches pour la réduction de la durée moyenne de réalisation

Ellie Beauprez
Ellie.Beauprez@univ-lille.fr

Anne-Cécile Caron
Anne-Cecile.Caron@univ-lille.fr

Maxime Morge
Maxime.Morge@univ-lille.fr

Jean-Christophe Routier
Jean-Christophe.Routier@univ-lille.fr

Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France

Résumé

Dans cet article, nous étudions le problème de la réallocation de tâches pour l'équilibrage de charge dans les modèles distribués de traitement de données massives. Nous proposons ici une stratégie qui repose sur des agents coopératifs pour optimiser le réordonnancement de tâches dans un ensemble de jobs devant être exécutés le plus tôt possible. Elle permet aux agents de déterminer localement les prochaines tâches à exécuter, à déléguer, voire à échanger grâce à leur modèle des pairs. La nouveauté réside dans la capacité des agents à échanger des tâches à travers des négociations bilatérales concurrentes. Nos expérimentations montrent que la durée moyenne de réalisation atteinte par notre stratégie reste proche de celle obtenue avec une heuristique classique, avec un temps de réordonnancement significativement réduit.

Mots-clés : *Résolution collective de problèmes, Négociation multi-agents*

Abstract

In this paper, we study the problem of task reallocation for load-balancing in distributed data processing models that tackle vast amount of data. We propose here a strategy based on cooperative agents used to optimize the rescheduling of tasks in multiple jobs which must be executed as soon as possible. It allows agents to determine locally the next tasks to process, to delegate, eventually to swap according to their peer modelling. The novelty lies in the ability of agents to swap tasks thanks to concurrent bilateral negotiations. Our experimentation reveals that our strategy reaches a flowtime which is close to the one reached by the classical heuristic approach, and significantly reduces the rescheduling time.

Keywords: *Distributed Problem Solving, Agent-based Negotiation*

1 Introduction

La problématique de l'affectation efficace de tâches parmi des entités exécutantes est commune à de nombreuses applications réelles pour la logistique [10], la robotique collective [7, 14], le calcul distribué [13], ou comme ici le traitement de données massives [1]. Cet article traite d'une classe d'applications pratiques où : (a) des jobs (c.-à-d. des ensembles de tâches) concurrents doivent êtres exécutés le plus tôt possible, et (b) les ressources (c.-à-d. les données) requises sont distribuées. Nous considérons particulièrement le modèle de traitement le plus répandu pour traiter des données massives sur une grappe de serveurs, c.-à-d. le patron de conception MapReduce [16]. Les jobs y sont composés d'un ensemble de tâches exécutées par les différents nœuds où sont réparties les ressources. Comme plusieurs ressources sont requises pour réaliser une tâche sur un nœud, son exécution nécessite de récupérer des ressources disponibles sur d'autres nœuds, ce qui induit un surcoût.

De nombreux travaux adoptent le paradigme multi-agents pour aborder le problème de la réallocation de tâches et de l'équilibrage de charge dans les systèmes distribués [2]. L'approche centrée individus permet la distribution d'heuristiques pour des problèmes impraticables à cause de la combinatoire des ordonnancements pour permettre le passage à l'échelle. De plus, intrinsèquement réactives, les méthodes multi-agents de réaffectation s'adaptent aux estimations inexactes des temps d'exécution et aux perturbations (consommation/libération de tâches, ralentissement des exécutants, etc.) La plupart de ces travaux adoptent l'approche orientée marché [15, 7], voire en s'appuyant sur des méthodes d'apprentissage qui nécessitent un historique [14, 13]. Par contraste, nous supposons comme [1] que : (a) les agents sont coopératifs, c.-à-d. ils ont une perception locale et partielle de l'allocation, mais ils partagent le même objectif,

et (b) aucun modèle préalable n'existe, ni des données, ni de l'environnement, de par la classe d'applications pratiques visées. Nous allons ici au-delà en considérant plusieurs jobs composés d'ensembles de tâches, chacune pouvant être exécutée par un seul des agents, tous compétents. Les agents souhaitent minimiser la durée moyenne de réalisation des jobs (*mean flowtime*) qui mesure combien de temps en moyenne l'utilisateur attend le résultat d'un job. La principale difficulté réside dans la formulation de systèmes complexes d'assignation tâches-exécutants décentralisés et adaptatifs, c.-à-d. la conception des comportements individuels joués de manière asynchrone par les exécutants qui doivent aboutir à l'émergence d'affectations faisables qui combinent les objectifs des commanditaires.

Nous proposons une stratégie qui décide quelle réallocation bilatérale est suggérée ou acceptée. Elle repose sur un modèle des pairs et détermine le comportement de l'agent à chaque point de choix dans le protocole de négociation. Dans le prolongement de [3], nos contributions sont les suivantes :
1. La notion de délégation a été généralisée sous la forme de réallocation bilatérale, ce qui permet d'envisager des échanges au sens de [8] ;
2. le critère de rationalité des réallocations a été redéfini ;
3. la stratégie de contre-offre permet à un agent de compléter une proposition de délégation par un échange de tâche.

Après un aperçu des travaux connexes dans la section 2, nous formalisons le problème d'allocation multi-agents de jobs composés de tâches (cf. section 3). La section 4 décrit les opérations de consommation/réallocation et le processus de négociation. La section 5 précise comment les agents choisissent quelles tâches négocier et avec qui. Notre évaluation empirique est décrite dans la section 6. La section 7 résume notre contribution et présente nos perspectives.

2 Travaux connexes

La théorie de l'ordonnancement [6] propose des méthodes hors-ligne pour résoudre différents problèmes d'affectation efficace de tâches parmi des entités exécutantes. Par exemple, le problème d'ordonnancement qui consiste à minimiser la durée totale de réalisation, notée $C(\overrightarrow{A})$, avec m exécutants multi-tâches et n tâches mono-exécutant dont les coûts dépendent de l'entité exécutante (noté R_m) peut être formalisé par programmation linéaire —*Linear Programming* (LP). Ce problème se réduit à un problème d'appariement pondéré dans un graphe biparti avec n tâches et $n \times m$ positions. Ce problème est polynomial [9]. Reposant sur l'algorithme de Ford-Fulkerson, la complexité de l'algorithme décrit par [5] est $O(max(mn^2, n^3))$. Cette approche n'est pas toujours adaptée à la réallocation de tâches dans des systèmes distribués où décentralisation et adaptativité sont nécessaires. En effet, un contrôle global constitue un goulot d'étranglement en matière de performance, car il doit en permanence collecter des informations sur l'état du système. À l'opposé, nos agents prennent des décisions locales sur une allocation existante dans le but d'améliorer l'équilibrage de charges. De plus, les problèmes d'ordonnancement classiques sont statiques. L'estimation inexacte du temps d'exécution des tâches, aggravée par des perturbations (consommation de tâches, libération de jobs, ralentissement des nœuds, etc.) peut nécessiter d'importantes modifications de l'allocation existante pour qu'elle reste optimale.

Le paradigme multi-agents est particulièrement approprié pour la conception et l'implémentation de mécanismes distribués et adaptatifs de réaffectation de tâches-exécutants [2]. Les modèles existants se distinguent de par la nature des tâches et des agents, qu'ils représentent les exécutants ou les commanditaires des tâches. S'inspirant de théories économiques, la « programmation orientée marché » aborde les problèmes de planification distribuée à travers la recherche d'un équilibre pour un jeu non-coopératif [15]. Nous considérons l'objectif global des commanditaires, contrairement à [8] qui favorise l'objectif individuel des exécutants. Parmi les méthodes orientées marché, où les agents délèguent des tâches voire les échangent, on distingue trois familles.

DCOP. Les problèmes de réaffectation peuvent être représentés sous la forme d'un problème d'optimisation sous contraintes distribué – *Distributed Constraint Optimization Problems* (DCOP). La principale difficulté dans la mise en œuvre de ces méthodes pour la réaffectation de tâches réside dans la représentation d'un problème réaliste sous la forme d'un DCOP, car elle nécessite une expertise de la méthode de résolution (e.g. [10]).

CBBA. L'algorithme à base de consensus (CBBA – *Consensus Based Bundle Algorithm*) [7] est une méthode multi-agents d'affectation en deux phases qui consiste à : (a) sélectionner les tâches à négocier ; (b) déterminer

l'agent qui remporte ces négociations. Dans la continuité, Turner et al. étudient l'affectation en continu de tâches à une flotte de robots pour maximiser le débit de réalisation avec des ressources en carburant limitées [14]. Grâce à l'apprentissage automatique supervisé à partir des exécutions précédentes, les robots choisissent dynamiquement et de manière décentralisée la meilleure heuristique de sélection de tâche.

MARL. L'apprentissage multi-agents par renforcement (MARL – *Multi-Agent Reinforcement Learning*) nécessite une connaissance parfaite de l'environnement et requiert par essence une phase d'exploration préliminaire [13]. À l'inverse, nous ne considérons aucun modèle préalable, ni des données, ni de l'environnement, car cela n'est pas pertinent pour la classe d'applications pratiques qui nous concerne.

Baert et al. visent dans [1] un objectif égalitaire qui est la minimisation du temps nécessaire à la réalisation de l'ensemble des tâches (noté $C_{max}(\overrightarrow{A})$). Nous considérons ici le problème de la coordination des décisions entre agents pour trouver une solution globalement optimale pour des fonctions multi-objectifs. Les agents tentent de minimiser la durée moyenne de réalisation de plusieurs jobs concurrents, chacun constitué de plusieurs tâches.

Cet article est dans le prolongement de [3].
1. Nous généralisons notre cadre formel pour considérer n'importe quelle réallocation bilatérale, e.g. des échanges pour réduire la durée moyenne de réalisation.
2. Nous redéfinissons le critère de rationalité des réallocations. Précédemment défini à partir du *flowtime* local, c'est-à-dire la durée de réalisation pour les agents impliqués dans la réallocation et du *makespan*, i.e. le temps nécessaire à la réalisation de l'ensemble des jobs, dans cet article, une réallocation est rationnelle si elle permet de réduire le *flowtime* global du système. Ce critère s'avère suffisant pour garantir la convergence du processus de réallocation. Dans un souci de concision, nous ne présenterons pas les expérimentations qui montrent que passer d'un *flowtime* local au *flowtime* global réduit non seulement la durée moyenne de réalisation mais également le temps de réordonnancement.

3 Tâches situées

Nous formalisons ici le problème d'allocation multi-agents des jobs concurrents composés de tâches situées.

Un job est un ensemble de tâches indépendantes, non divisibles et non-préemptives. L'exécution de chaque tâche nécessite l'accès à des ressources distribuées sur les nœuds du système. Nous considérons les ressources transférables et non consommables.

Définition 1 (Système distribué)**.** *Un système distribué est un triplet $\mathcal{D} = \langle \mathcal{N}, \mathcal{E}, \mathcal{R} \rangle$ où :*
— $\mathcal{N} = \{\nu_1, \ldots, \nu_m\}$ *est un ensemble de nœuds ;*
— $\mathcal{E}$ *est une relation d'accointances, i.e. une relation binaire et symétrique sur* $\mathcal{N}$ *;*
— $\mathcal{R} = \{\rho_1, \ldots, \rho_k\}$ *est un ensemble de ressources de tailles* $|\rho_i|$*. La localisation des ressources, éventuellement répliquées, est déterminée par la fonction* $l : \mathcal{R} \to 2^{\mathcal{N}}$

Suivant notre application pratique (cf. section 6), nous faisons l'hypothèse ici qu'il y a exactement un agent par nœud et que toutes les ressources sont accessibles pour tous les agents, même celles sur les nœuds non locaux, car la relation d'accointances est totale.

L'exécution d'un job (sans date butoir) consiste à exécuter un ensemble de tâches indépendantes.

Définition 2 (Job/Tâche)**.** *Soit $\mathcal{D}$ un système distribué. On considère un ensemble de ℓ jobs $\mathcal{J} = \{J_1, \ldots, J_\ell\}$. Chaque job J_i est un ensemble de k_i tâches $J_i = \{\tau_1, \ldots, \tau_{k_i}\}$. Chaque tâche a un coût pour chacun des nœuds dont la valeur est déterminée par la fonction $c : \mathcal{T} \times \mathcal{N} \mapsto \mathbb{R}_+^*$*

On note $\mathcal{T} = \cup_{1 \leq i \leq \ell} J_i$ l'ensemble des n tâches sous-jacentes à $\mathcal{J}$ et $\mathcal{R}_\tau \subseteq \mathcal{R}$ l'ensemble des ressources requises pour la tâche τ. Par souci de concision, on note $\text{job}(\tau)$ le job contenant la tâche τ. Nous considérons que le nombre de jobs est négligeable par rapport au nombre de tâches, $|\mathcal{J}| << |\mathcal{T}|$.

Le coût d'une tâche est une estimation de son temps d'exécution par un nœud.

Hypothèse 1 (Coût)**.** *Soient $\mathcal{D} = \langle \mathcal{N}, \mathcal{E}, \mathcal{R} \rangle$ un système distribué et $\mathcal{T}$ un ensemble de tâches. La fonction de coût vérifie que :*

$$\begin{aligned} &\forall \tau \in \mathcal{T}, \forall \{\nu_i, \nu_j\} \subset \mathcal{N}, \\ &\Sigma_{\rho \in \mathcal{R}_\tau, \nu_i \in l(\rho)} |\rho| > \Sigma_{\rho \in \mathcal{R}_\tau, \nu_j \in l(\rho)} |\rho| \\ &\Rightarrow c(\tau, \nu_i) \leq c(\tau, \nu_j) \end{aligned} \quad (1)$$

Comme la collecte de ressources non locales représente un surcoût, une tâche est moins coûteuse si les ressources nécessaires sont « plus

locales » (cf. section 6). La fonction de coût peut être étendue à un ensemble de tâches :

$$\forall \mathrm{T} \subseteq \mathcal{T},\ \mathrm{c}(\mathrm{T},\nu_i) = \Sigma_{\tau \in \mathrm{T}} \mathrm{c}(\tau,\nu_i) \qquad (2)$$

En substance, nous considérons le problème d'allocation multi-agents de jobs composés de tâches situées.

Définition 3 (MASTA+)**.** *Un problème d'allocation multi-agents de jobs est un quadruplet* $MASTA+ = \langle \mathcal{D}, \mathcal{T}, \mathcal{J}, c \rangle$ *où :*
— $\mathcal{D}$ *est un système distribué de m nœuds ;*
— $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$ *est un ensemble de n tâches ;*
— $\mathcal{J} = \{J_1, \ldots, J_\ell\}$ *est un partitionnement des tâches en* ℓ *jobs ;*
— $c : \mathcal{T} \times \mathcal{N} \mapsto \mathbb{R}^*_+$ *est la fonction de coût.*

Une allocation de tâches est une répartition des tâches dans des lots ordonnés.

Définition 4 (Allocation)**.** *Une allocation pour un problème MASTA+ est un vecteur de m lots de tâches ordonnées* $\overrightarrow{A} = ((B_1, \prec_1), \ldots, (B_m, \prec_m))$ *où chaque lot* $(B_i, \prec_i)$ *est l'ensemble des tâches* ($B_i \subseteq \mathcal{T}$) *affectées au nœud* ν_i *associé à un ordre total strict* ($\prec_i \subseteq \mathcal{T} \times \mathcal{T}$)*.* $\tau_j \prec_i \tau_k$ *signifie que si* $\tau_j, \tau_k \in B_i$ *alors* τ_j *est exécutée avant* τ_k *par* ν_i*. L'allocation* $\overrightarrow{A}$ *vérifie :*

$$\forall \tau \in \mathcal{T},\ \exists \nu_i \in \mathcal{N},\ \tau \in B_i \qquad (3)$$
$$\forall \nu_i \in \mathcal{N}, \forall \nu_j \in \mathcal{N} \setminus \{\nu_i\},\ B_i \cap B_j = \emptyset \qquad (4)$$

Toutes les tâches sont allouées (Eq. 3) et chacune n'est allouée qu'à un seul nœud (Eq. 4). Par souci de concision, on note :
— $\overrightarrow{\mathrm{B}}_i = (\mathrm{B}_i, \prec_i)$, le lot trié de ν_i ;
— $\mathrm{jobs}(\mathrm{B}_i)$, l'ensemble des jobs affectés à ν_i, i.e. les jobs ayant au moins une tâche dans B_i ;
— $\nu(\tau, \overrightarrow{\mathrm{A}})$, le nœud chargé de τ dans $\overrightarrow{\mathrm{A}}$;
— $\mathrm{w}_i(\overrightarrow{\mathrm{A}}) = \Sigma_{\tau \in \mathrm{B}_i} \mathrm{c}(\tau, \nu_i)$, la charge de travail du nœud ν_i pour l'allocation $\overrightarrow{\mathrm{A}}$.
Comme on suppose que les nœuds sont toujours actifs, la durée de réalisation d'une tâche (*completion time*) correspond au temps d'attente avant que la tâche soit entamée plus l'estimation de son temps d'exécution :

$$\mathrm{C}_\tau(\overrightarrow{\mathrm{A}}) = \mathrm{t}(\tau, \nu(\tau, \overrightarrow{\mathrm{A}})) + \mathrm{c}(\tau, \nu(\tau, \overrightarrow{\mathrm{A}}))$$
$$\text{avec } \mathrm{t}(\tau, \nu_i) = \Sigma_{\tau' \in \mathrm{B}_i | \tau' \prec_i \tau} \mathrm{c}(\tau', \nu_i) \qquad (5)$$

Contrairement aux coûts, les durées de réalisation dépendent de l'ordre d'exécution.

Pour évaluer la qualité d'une allocation de tâches, nous considérons le *flowtime* moyen, qui mesure le temps moyen écoulé entre la date de libération des jobs [1] et leur date d'achèvement, et le *makespan* qui est le temps nécessaire à la réalisation de l'ensemble des jobs.

Définition 5 (*Flowtime*/*Makespan*)**.** *Soient MASTA+ un problème d'allocation de tâches et* $\overrightarrow{A}$ *une allocation. On définit :*
— *la durée de réalisation de* $J \in \mathcal{J}$ *pour* $\overrightarrow{A}$*,*

$$C_J(\overrightarrow{A}) = \max_{\tau \in J} \{C_\tau(\overrightarrow{A})\} \qquad (6)$$

— *le flowtime (moyen) de* $\mathcal{J}$ *pour* $\overrightarrow{A}$*,*

$$C_{mean}(\overrightarrow{A}) = \frac{1}{\ell} C(\overrightarrow{A}) \text{ avec } C(\overrightarrow{A}) = \Sigma_{J \in \mathcal{J}} C_J(\overrightarrow{A}) \qquad (7)$$

— *le makespan de* $\mathcal{J}$ *pour* $\overrightarrow{A}$*,*

$$C_{max}(\overrightarrow{A}) = \max_{\nu_i \in \mathcal{N}} \{w_i(\overrightarrow{A})\} \qquad (8)$$

Contrairement au *makespan*, le *flowtime* dépend de l'ordre d'exécution des tâches sur chacun des nœuds.

Exemple 1 (MASTA+)**.** *À partir du système distribué* $\mathcal{D} = \langle \mathcal{N}, \mathcal{E}, \mathcal{R} \rangle$ *avec* $\mathcal{N} = \{\nu_1, \nu_2, \nu_3\}$, $\mathcal{E} = \{(\nu_1, \nu_2), (\nu_1, \nu_3), (\nu_2, \nu_3)\}$ *et* $\mathcal{R} = \{\rho_1, \ldots, \rho_9\}$ *où les ressources sont répliquées sur 2 nœuds (cf. Fig. 1a), nous considérons* $MASTA+ = \langle \mathcal{D}, \mathcal{T}, \mathcal{J}, c \rangle$ *avec* $\mathcal{T} = \{\tau_1, \ldots, \tau_9\}$ *où chaque tâche* τ_i *nécessite la ressource* ρ_i*,* $\mathcal{J} = \{J_1, J_2, J_3\}$ *tel que* $J_1 = \{\tau_1, \tau_2, \tau_3\}$*,* $J_2 = \{\tau_4, \tau_5, \tau_6\}$ *et* $J_3 = \{\tau_7, \tau_8, \tau_9\}$ *et la fonction de coût donnée dans la table 1. Nous supposons que le coût d'une tâche est proportionnel à la taille des ressources et qu'il est deux fois plus important si la ressource est non locale. Nous considérons ici l'allocation* $\overrightarrow{A}$ *(cf. figure 1b) avec* $\overrightarrow{B}_1 = (\tau_5, \tau_8, \tau_3, \tau_2)$*,* $\overrightarrow{B}_2 = (\tau_4, \tau_9)$ *et* $\overrightarrow{B}_3 = (\tau_7, \tau_1, \tau_6)$*. Le makespan et le flowtime sont* $C_{max}(\overrightarrow{A}) = 12$ *et* $C(\overrightarrow{A}) = 8 + 12 + 12 = 32$*.*

	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7	τ_8	τ_9
$c(\tau, \nu_1)$	5	3	1	8	2	10	4	2	4
$c(\tau, \nu_2)$	10	3	2	4	2	5	2	2	8
$c(\tau, \nu_3)$	5	6	1	4	4	5	2	4	4

TABLE 1 – Le coût des tâches pour chaque nœud

En résumé, le coût des tâches dépend du nœud qui l'exécute en raison de la localité des ressources. Notre objectif est de minimiser le *flowtime* moyen des jobs composés de tâches.

1. Pour simplifier, nous supposons dans cet article que la date de libération est $t^0 = 0$ pour tous les jobs.

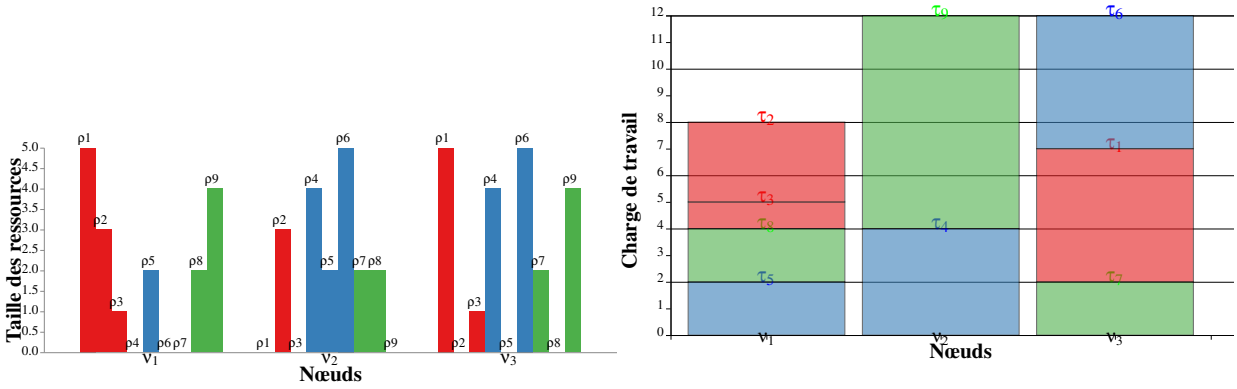


FIGURE 1 – Répartition des ressources et allocation des tâches aux nœuds pour notre exemple fil rouge où les jobs J_1, J_2 et J_3 sont respectivement représentés en rouge, bleu et vert.

4 Consommation et réallocation

Nous décrivons ici les opérations de consommation et de réallocation ainsi que le protocole de négociation.

Une **consommation** de tâche consiste pour un nœud à supprimer une tâche de son lot pour l'exécuter. La stratégie de consommation d'un agent spécifie l'ordonnancement des tâches pour le nœud dont il a la charge. Comme nous voulons minimiser le *flowtime* moyen des jobs, nous considérons ici une stratégie orientée job qui trie le lot d'abord par job puis par tâche au sein d'un même job (les tâches d'un même job sont consécutives dans le lot). En particulier, les jobs les moins coûteux sont prioritaires sur ceux plus coûteux pour minimiser localement le délai de réalisation des jobs. Par la suite, $J_1 \blacktriangleleft_i J_2$ signifie que les tâches de J_1 sont prioritaires sur celles de J_2 et $\tau_1 \lhd_i \tau_2$ que la tâche τ_1 est prioritaire sur la tâche τ_2. Plus formellement :

$$\begin{aligned} &\forall \tau_j, \tau_k \in B_i \ \tau_j \prec_i \tau_k \Leftrightarrow \mathrm{job}(\tau_j) \blacktriangleleft_i \mathrm{job}(\tau_k) \\ &\vee (\mathrm{job}(\tau_j) = \mathrm{job}(\tau_k) \wedge \tau_j \lhd_i \tau_k) \end{aligned} \quad (9)$$

L'ajout ou la suppression d'une liste de tâches T dans le lot $\overrightarrow{B}_i$ du nœud ν_i peuvent changer l'ordre d'exécution des tâches puisque ces opérations impliquent un réordonnancement du lot :
— $\overrightarrow{B_i \oplus T}$ désigne le lot qui contient l'ensemble des tâches $B_i \cup T$ trié selon $\prec_i$;
— $\overrightarrow{B_i \ominus T}$ désigne le lot qui contient $B_i \setminus T$ trié selon $\prec_i$.
— $\overrightarrow{B_i \ominus T_1 \oplus T_2}$ désigne le lot qui contient $B_i \setminus T_1 \cup T_2$ trié selon $\prec_i$.
Une **réallocation bilatérale** est une opération qui modifie l'allocation courante via l'échange de tâches entre deux agents.

Définition 6 (Réallocation bilatérale). *Soit* $\overrightarrow{A} = (\overrightarrow{B}_1, \ldots, \overrightarrow{B}_m)$ *une allocation pour le problème MASTA+* $= \langle \mathcal{D}, \mathcal{T}, \mathcal{J}, c \rangle$. *La réallocation bilatérale de la liste non vide de tâches* T_1 *allouées au proposant* ν_i *en échange de la liste de tâches* T_2 *allouées au répondant* ν_j *dans* $\overrightarrow{A}$ *(*$T_1 \subseteq B_i$ *et* $T_2 \subseteq B_j$*) aboutit à l'allocation* $\gamma(T_1, T_2, \nu_i, \nu_j, \overrightarrow{A})$ *avec les m lots* $\gamma(T_1, T_2, \nu_i, \nu_j, \overrightarrow{B}_k)$ *définis tels que :*

$$\gamma(T_1, T_2, \nu_i, \nu_j, \overrightarrow{B}_k) = \begin{cases} \overrightarrow{B_i \ominus T_1 \oplus T_2} & \text{si } k = i, \\ \overrightarrow{B_j \ominus T_2 \oplus T_1} & \text{si } k = j, \\ \overrightarrow{B}_k & \text{sinon} \end{cases} \quad (10)$$

Pour une réallocation bilatérale $\gamma(T_1, T_2, \nu_i, \nu_j, \overrightarrow{A})$, on distingue deux cas :
— un **échange** où les deux listes de tâches sont non vides ($T_1 \neq \emptyset \wedge T_2 \neq \emptyset$), noté $\sigma(T_1, T_2, \nu_i, \nu_j, \overrightarrow{A})$. Si $|T_1| = |T_2| = 1$, on parle d'échange unaire ;
— une **délégation** où un agent donne une partie de ses tâches à l'un de ses pairs sans contre-partie ($T_2 = \emptyset$), notée $\delta(T_1, \nu_i, \nu_j, \overrightarrow{A})$.
Afin d'améliorer une allocation, nous introduisons la notion de réallocation bilatérale socialement rationnelle.

Définition 7 (Réallocation bilatérale socialement rationnelle). *Soit* $\overrightarrow{A}$ *une allocation pour le problème MASTA+* $= \langle \mathcal{D}, \mathcal{T}, \mathcal{J}, c \rangle$. *La réallocation bilatérale* $\gamma(T_1, T_2, \nu_i, \nu_j, \overrightarrow{A})$ *est socialement rationnelle ssi le flowtime global décroît,*

$$C(\gamma(T_1, T_2, \nu_i, \nu_j, \overrightarrow{A})) < C(\overrightarrow{A}) \quad (11)$$

Une allocation est dite **stable** s'il n'existe aucune réallocation bilatérale socialement rationnelle.

Contrairement à [3], nous ne considérons pas comme socialement rationnelles des réallocations qui réduisent le *flowtime* local (le délai de réalisation des jobs pour les seuls nœuds impliqués dans la réallocation) qui ne permet pas de garantir la convergence du processus de réallocation, ni même comme rationnelles des réallocations qui réduisent le *flowtime* local et le *makespan* (la charge maximale des agents). La réduction du *flowtime* global permet de garantir la terminaison du processus. Dorénavant lorsque nous parlerons de *flowtime* il s'agira, sauf précision, du *flowtime* global, noté $C(\overrightarrow{A})$ (cf. Eq. 7).

Propriété 1 (Terminaison)**.** *Soit $\overrightarrow{A}$ une allocation pour un problème MASTA+ qui n'est pas stable. Toute séquence de réallocations bilatérales socialement rationnelles issue de $\overrightarrow{A}$ est finie car elle atteint une allocation stable.*

Esquisse de preuve 1. *Comme il existe un nombre fini d'allocations et que $C(\overrightarrow{A})$ décroît strictement à chaque étape, il ne peut y avoir qu'un nombre fini de telles réallocations.*

Pour réallouer les tâches, les agents réalisent de multiples négociations bilatérales à un tour. Ces négociations sont concurrentes mais un agent ne peut être impliqué à un instant donné que dans une seule négociation. Chaque négociation, reposant sur un protocole d'offres alternées [12], inclut trois étapes de décision : (a) la stratégie d'offre du proposant qui sélectionne une délégation, (b) la stratégie de contre-offre qui permet au répondant de déterminer s'il décline la délégation, l'accepte, voire fait une contre-offre et (c) l'éventuelle réallocation est confirmée ou annulée par le proposant selon les consommations qui ont eu lieu de manière concurrente.

Exemple 2 (Consommation et réallocation)**.** *Considérons le problème MASTA+ de l'exemple 1, en particulier l'allocation $\overrightarrow{A}$ représentée dans la figure 1b. Selon la stratégie de consommation adoptée par les agents, chaque lot est trié par job, c.-à-d. les jobs les moins coûteux sont prioritaires (e.g. $J_3 \blacktriangleleft_3 J_1 \blacktriangleleft_3 J_2$). L'ordre naturel sur les identifiants permet de défaire les égalités (e.g. $J_2 \blacktriangleleft_1 J_3$). Les tâches au sein d'un même job sont triées par ordre de coût croissant ($\tau_3 \lhd_1 \tau_2$). La délégation de la tâche τ_9 par le nœud ν_2 au nœud ν_1 aboutissant à l'allocation $\overrightarrow{A}' = \delta([\tau_9], \nu_2, \nu_1, \overrightarrow{A})$ (cf. figure 2a) est socialement rationnelle car elle fait décroître le flowtime de 32 à 31. Toutefois, l'échange de $\tau_9 \in B_2$ contre $\tau_5 \in B_1$ entre les nœuds ν_2 et ν_1, qui aboutit à l'allocation $\overrightarrow{A}'' = \sigma([\tau_9], [\tau_5], \nu_2, \nu_1, \overrightarrow{A})$ (cf. figure 2b), est meilleur car il fait décroître le flowtime de 32 à 29.*

5 Stratégie de négociation

Nous décrivons ici la stratégie de négociation et nous esquissons le comportement des agents.

Le **modèle des pairs** est construit à partir des messages échangés entre agents. Avant le processus de négociation, à travers ses propositions et après chaque réallocation bilatérale dans laquelle il est impliqué, l'agent ν_i informe ses pairs de ce que chaque job J lui coûte, $c(J, \nu_i)$. Le modèle de la cible ν_k par le sujet ν_i repose sur :

1. la base de croyances du sujet [2], éventuellement partielle ou obsolète, qui contient les croyances concernant les coûts des jobs pour ν_k ($c^i(J, \nu_k)$, $\forall J \in \mathcal{J}$) et donc les croyances concernant la charge de travail de ν_k ($w_k^i(\overrightarrow{A}) = \Sigma_{J \in \mathcal{J}} c^i(J, \nu_k)$) ;
2. la stratégie de consommation de la cible supposée par le sujet, notée $(\mathcal{J}, \blacktriangleleft_k^i)$.

Par souci de cohérence, on écrit $c^i(J, \nu_i) = c(J, \nu_i)$ et $w_i^i(\overrightarrow{A}) = w_i(\overrightarrow{A})$.

Le sujet peut alors déduire :

— la durée de réalisation ($C_J^i(\overrightarrow{B}_k)$) du job J pour une cible k, éventuellement lui-même ($\nu_k = \nu_i$), après l'ajout ($C_J^i(\overrightarrow{B_k \oplus T})$), la suppression ($C_J^i(\overrightarrow{B_k \ominus T})$) et le remplacement de tâches ($C_J^i(\overrightarrow{B_k \ominus T_1 \oplus T_2})$) ;

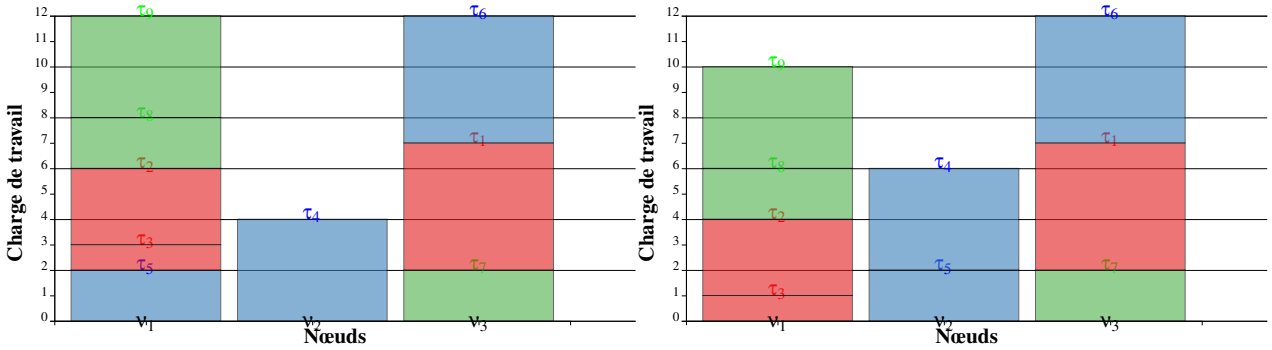
— la durée de réalisation d'un job J pour l'allocation, $C_J^i(\overrightarrow{A}) = \max_{\nu_k \in \mathcal{N}} C_J^i(\overrightarrow{B}_k)$ où $C_J^i(\overrightarrow{B}_i) = C_J(\overrightarrow{B}_i)$;

— le nœud **limitant** pour chaque job J, noté $\nu_{\max}^i(\overrightarrow{A}, J)$, i.e. le nœud ν_k pour lequel la durée de réalisation de ce job est la durée de réalisation maximale, $C_J^i(\overrightarrow{B}_k) = C_J^i(\overrightarrow{A})$;

— le *flowtime* de l'allocation $C^i(\overrightarrow{A}) = \Sigma_{J \in \mathcal{J}} C_J^i(\overrightarrow{A})$.

La **règle d'acceptabilité** est une décision locale prise par un agent pour accepter ou décliner une réallocation.

2. Les croyances du sujet ν_i sont dénotées avec *i* en exposant.



(a) Allocation après la délégation de τ_9 de ν_2 à ν_1

(b) Allocation après l'échange de τ_9 contre τ_5 entre ν_2 et ν_1

FIGURE 2 – Allocations de l'exemple fil rouge résultant de réallocations bilatérales

Définition 8 (Acceptabilité). *Soit $\overrightarrow{A}$ une allocation pour un problème MASTA+. La réallocation bilatérale $\gamma(T_1, T_2, \nu_i, \nu_j, \overrightarrow{A})$ est acceptable par l'agent $\nu_k \in \{\nu_i, \nu_j\}$ ssi l'agent croit que le flowtime diminue strictement,*

$$\Sigma_{J\in\mathcal{J}} \max_{\forall \nu_o \in \mathcal{N}\setminus\{\nu_i,\nu_j\}} (C_J^k(\overrightarrow{B_i \ominus T_1 \oplus T_2}), C_J^k(\overrightarrow{B_j \ominus T_2 \oplus T_1}), C_J^k(B_o)) < C^k(\overrightarrow{A}) \quad (12)$$

L'acceptabilité repose sur la croyance à propos des durées de réalisation des jobs pour les nœuds avant et après la réallocation (Eq. 12).

Nous proposons ici un processus où les agents initient des négociations bilatérales concurrentes qui doivent aboutir à des réallocations socialement rationnelles.

La **stratégie d'offre** d'un agent identifie une délégation en trois étapes. Il s'agit donc pour un agent ν_i de sélectionner une offre, c'est-à-dire une tâche à déléguer à un receveur dans un ensemble $\mathcal{N}'$ afin de réduire la durée de réalisation d'un job pour lequel il est limitant dans un ensemble $\mathcal{J}'$. Initialement, $\mathcal{J}' = \mathcal{J}$, $\mathcal{N}' = \mathcal{N}$. Notons que si la section précédente définit des réallocations de listes de tâches, notre stratégie d'offre se limite ici à des listes de taille 1.

1. Sélection d'un job. Afin de réduire non seulement la durée de réalisation globale d'un job qui lui est affecté mais également celles des jobs qui suivent dans son lot, notre heuristique sélectionne le job J_* le plus prioritaire parmi ceux dont il est l'agent limitant,

$$J_* = \min_{\blacktriangleleft_i}\{J \in \text{jobs}(B_i) \cap \mathcal{J}' \mid \nu^i_{\max}(\overrightarrow{A}, J) = \nu_i\} \quad (13)$$

2. Sélection d'un receveur. Les jobs d'un receveur impactés par la délégation sont ceux placés après J_* selon $\blacktriangleleft^i_j$. Afin de ne pas augmenter la durée de réalisation de ces jobs, notre heuristique sélectionne un receveur ν_* pour qui la somme des différences entre la durée de réalisation pour l'allocation et celle pour l'agent est la plus grande,

$$\nu_* = rnd\{\operatorname*{argmax}_{\nu_j \in \mathcal{N}'} \Sigma_{J_* \blacktriangleleft^i_j J}(C_J^i(\overrightarrow{A}) - C_J^i(\overrightarrow{B}_j))\} \quad (14)$$

où *rnd* est une fonction de choix aléatoire qui à chaque ensemble de nœuds associe un nœud de cet ensemble.

3. Sélection d'une délégation. Afin de réduire les durées de réalisation, l'initiateur sélectionne une tâche non locale, c.-à-d. dont la délégation réduira son coût d'exécution. Notre heuristique sélectionne la tâche du job J_* ou des jobs qui le précèdent dans $\overrightarrow{B}_i$ avec le meilleur gain en terme de coût. En cas d'égalité, c'est la tâche prioritaire du lot qui est choisie,

$$\min_{\lhd_i}\{\operatorname*{argmax}_{\tau \in \mathcal{T}' \cap B_i \cap \{J \mid J = J_* \vee (J \blacktriangleleft_i J_*)\}} c(\tau, \nu_i) - c(\tau, \nu_*)\}$$

Afin de déterminer l'offre, on distingue une stratégie **conservatrice** qui propose la délégation $\delta([\tau_*], \nu_i, \nu_*, \overrightarrow{A})$ si elle est acceptable pour l'initiateur (cf Def. 8) comme dans [3] et une stratégie **libérale** qui propose des délégations même si elles ne sont pas acceptables.

La **stratégie de contre-offre** d'un répondant ν_j à une délégation $\delta(\tau_1, \nu_i, \nu_j, \overrightarrow{A})$ lui permet de proposer une tâche dans l'ensemble $\mathcal{T}' = \mathcal{T}$ en échange de celle proposée.

1. Sélection d'une tâche. Afin de réduire les durées de réalisation, le donneur sélectionne une tâche non locale, i.e. dont la délégation réduira son coût puisqu'elle sera exécutée localement.

Notre heuristique sélectionne la tâche non locale qui est la plus prioritaire.

$$\tau_* = \min_{\lhd_j}\{\tau \in \mathcal{T}' \cap \overrightarrow{\mathrm{B}}_j \mid \mathrm{c}(\tau,\nu_j) - \mathrm{c}(\tau,\nu_i) < 0\} \quad (15)$$

2. Validation. Dans le but de garantir la convergence du processus de négociation, le déclenchement d'un échange par un répondant est soumis à son acceptabilité de l'échange qui s'appuie sur ses connaissances mises à jour. De plus, par souci d'efficacité, un échange est déclenché à condition qu'il permette de réduire le *flowtime* plus encore que la délégation initialement proposée. Formellement,

$$\begin{aligned}
&\Sigma_{\mathrm{J}\in\mathcal{J}} \max_{\forall \nu_o \in \mathcal{N}\setminus\{\nu_i,\nu_j\}} \\
&(\mathrm{C_J}^k(\overrightarrow{\mathrm{B}_i \ominus \tau_1 \oplus \tau_*}), \mathrm{C_J}^k(\overrightarrow{\mathrm{B}_j \ominus \tau_* \oplus \tau_1}), \mathrm{C_J}^k(\mathrm{B}_o)) \\
&< \Sigma_{\mathrm{J}\in\mathcal{J}} \max_{\forall \nu_o \in \mathcal{N}\setminus\{\nu_i,\nu_j\}} \\
&(\mathrm{C_J}^k(\overrightarrow{\mathrm{B}_i \ominus \tau_1}), \mathrm{C_J}^k(\overrightarrow{\mathrm{B}_j \oplus \tau_1}), \mathrm{C_J}^k(\mathrm{B}_o)) \qquad (16)
\end{aligned}$$

Dans le cas contraire, $\mathcal{T}'$ devient $\mathcal{T}' \setminus \{\tau_*\}$ dans l'étape 1. En cas d'échec, aucune contre-partie n'est proposée et le répondant évalue l'acceptabilité de la délégation pour la rejeter ou l'accepter.

Comportement d'agent. Dans notre approche, une réallocation bilatérale de tâches est le résultat de négociations entre agents qui adoptent tous le même comportement spécifié dans [4] par un automate [3] qui consiste en **2 phases** successives :
1. les proposants adoptent la stratégie d'offre conservatrice et les répondants acceptent une délégation s'ils la considèrent acceptable. Ils la rejettent sinon ;
2. les agents adoptent la stratégie d'offre libérale et la stratégie de contre-offre. Si au moins une réallocation est réalisée, ces deux phases sont réitérées. Sinon le processus est clos.

Exemple 3 (Stratégie de négociation)**.** *Considérons l'allocation* $\overrightarrow{A}$ *pour le problème MASTA+ de l'exemple 1 où l'agent* ν_1*, pour lequel on suppose les croyances à jour, reçoit de la part de l'agent* ν_2 *la proposition de la délégation* $\delta(\tau_9,\nu_2,\nu_1,\overrightarrow{A})$*. L'agent* ν_1 *sélectionne la tâche avec le gain le plus important (Eq. 15),* $\tau_* = \tau_5$*. L'échange est acceptable, le flowtime global décroît strictement et il est meilleur que celui après la seule délégation (cf. figures 1b et 2b). En résumé, le répondant* ν_1 *propose* τ_5 *en contrepartie de la tâche* τ_9 *pour atteindre l'allocation* $\overrightarrow{A}' = \sigma(\tau_9,\tau_5,\nu_2,\nu_1,\overrightarrow{A})$.

3. https://gitlab.univ-lille.fr/maxime.morge/smastaplus/-/tree/master/doc/specification

6 Évaluation empirique

Après une présentation des conditions expérimentales, nous comparons empiriquement notre approche avec une heuristique classique.

L'application pratique que nous considérons est le déploiement distribué du patron de conception MapReduce pour le traitement de jeux de données massives sur une grappe de serveurs, comme avec Spark [16]. Nous nous focalisons ici sur la phase *reduce* des jobs MapReduce que nous formalisons par un problème MASTA+ où plusieurs jobs sont soumis de façon concurrente. La fonction de coût est définie telle que :

$$\begin{aligned}
c_i(\tau,\nu_j) &= \Sigma_{\rho_j \in \mathcal{R}_\tau} c_i(\rho_j,\nu_j) \\
\text{avec } c_i(\rho_j,\nu_i) &= \begin{cases} |\rho_j| \text{ si } \nu_i \in l(\rho_j) \\ \kappa \times |\rho_j| \text{ sinon} \end{cases} \qquad (17)
\end{aligned}$$

où nous avons fixé empiriquement $\kappa = 2$ comme une valeur réaliste pour capturer le surcoût induit par la récupération des ressources non locales.

Notre prototype [4] est implémenté avec le langage de programmation Scala et la bibliothèque Akka [11] adaptée aux applications orientées messages, fortement concurrentes, distribuées et robustes. Nous supposons que : (a) le délai de transmission des messages est arbitraire mais non négligeable, (b) l'ordre des messages par pair émetteur-récepteur est préservé, (c) la distribution des messages est garantie. Les expériences ont été réalisées sur une lame munie de 20 CPUs avec 512Go de RAM.

Nous considérons des instances de MASTA+ avec $m \in [2;16]$ nœuds/agents, $\ell = 4$ jobs, $n = 3 \times \ell \times m$ tâches et 10 ressources par tâche. Chaque ressource ρ_i est répliquée 3 fois et $|\rho_i| \in [0;100]$. Nous générons 10 instances de MASTA+, et pour chacune nous générons aléatoirement 10 allocations initiales. Nous évaluons les médianes et les écarts types de deux métriques : la durée de réalisation (Eq. 7) et le temps de réordonnancement. Nous avons vu dans [1] que, lorsque le processus de consommation s'effectue de manière concurrente au processus de négociation, la communication n'affecte pas le temps de d'exécution.

Les hypothèses que nous voulons tester sont : (1) la durée de réalisation atteinte par notre stratégie est proche de celle obtenue par l'approche classique ; (2) notre approche accélère le réordonnancement.

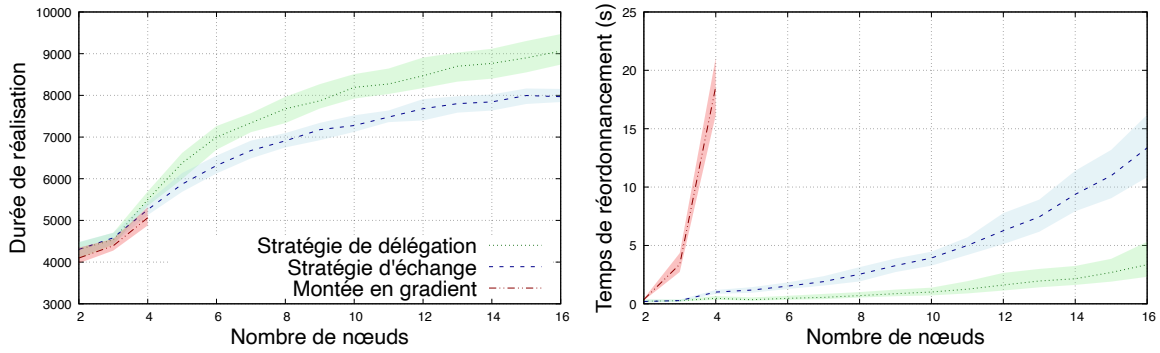


FIGURE 3 – Durée de réalisation et temps de réordonnancement pour nos stratégies

Les figures 3a et 3b comparent respectivement la durée de réalisation et le temps de réordonnancement de notre stratégie de délégation avec celle de notre stratégie d'échange ainsi qu'avec un algorithme de montée en gradient [4] qui débutent avec la même allocation initiale générée aléatoirement. À chaque étape, l'algorithme de montée en gradient sélectionne parmi tous les échanges unaires possibles celui qui minimise la durée moyenne de réalisation. Nous observons que notre stratégie d'échange atteint des solutions de qualité similaire aux solutions obtenues par la montée en gradient et, évidemment, légèrement meilleure que la stratégie de délégation. Nous observons également que, si le temps de réordonnancement de la stratégie d'échange est légèrement moins bon que pour la stratégie de délégation, il reste approximativement identique et donc largement meilleur que celui de l'algorithme de montée en gradient qui croît exponentiellement avec le nombre de noeuds. Même si, contrairement à ce dernier, notre stratégie n'évalue pas tous les échanges unaires possibles, elle s'avère efficace en sélectionnant les tâches non locales dont la délégation réduit le coût. Les échanges permettent d'échapper à des minimums locaux et de déclencher de nouvelles délégations.

La figure 4 représente, pour une exécution particulière, l'évolution du *flowtime* moyen de la stratégie d'échange. Elle opère 87 délégations dans la phase 1 comme la stratégie de délégation. Contrairement à cette dernière, la stratégie d'échange poursuit le processus de négociation en opérant 32 échanges dans la phase 2 qui, à leur tour, permettent de déclencher 23 délégations puis 14 échanges. La stratégie d'échange prolonge la stratégie de délégation en atteignant

4. Les algorithmes DCOP que nous avons évalués ne passent pas à l'échelle sur ce type de problème.

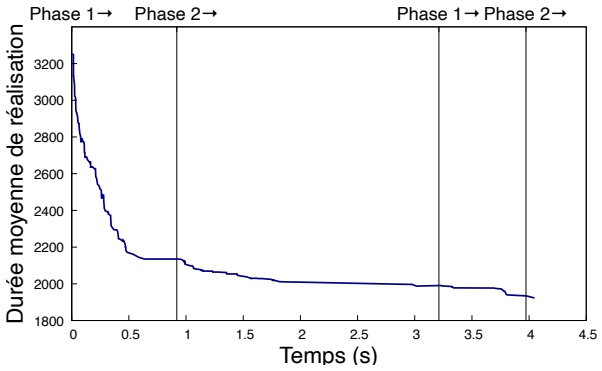


FIGURE 4 – Exécution de la stratégie d'échange

une allocation similaire avec le même temps de réordonnancement tout en permettant de l'améliorer par la suite.

7 Conclusion

Dans cet article, nous avons proposé un système multi-agents pour la réallocation de tâches sur des nœuds en fonction de la localisation des ressources nécessaires afin de réduire la durée moyenne de réalisation de jobs concurrents. Nos expériences montrent que la durée moyenne de réalisation atteinte par notre stratégie est proche de celle atteinte par l'approche heuristique classique et qu'elle réduit considérablement le temps de réordonnancement. Nous avons étendu notre cadre de négociation pour considérer n'importe quelle réallocation bilatérale, en particulier des échanges. Nous avons proposé une stratégie de contre-offre pour sélectionner une contre-partie susceptible de déclencher des échanges de tâches afin d'échapper à des minimums locaux et déclencher de nouvelles délégations. Notre nouvelle stratégie d'échange prolonge notre stratégie de délégation en atteignant une allocation similaire avec le même temps de réordonnancement

tout en permettant de l'améliorer par la suite.

Nos travaux futurs porterons sur une stratégie d'offres constituées de lot de tâches pour réduire le temps de réordonnancement et améliorer la durée moyenne de réalisation. Pour l'instant, nos expérimentations se focalisent sur la réallocation d'un ensemble de tâches appartenant à plusieurs jobs. Toutefois, comme pour [1], nous souhaitons intégrer la consommation des tâches afin d'étudier l'adaptation de notre système à des ensembles de tâches qui évoluent dans le temps. Dans le même ordre d'idées, les travaux futurs doivent également intégrer la réallocation des tâches dans un processus d'approvisionnement qui ajoute ou supprime des nœuds de calcul au cours de l'exécution en fonction des besoins des utilisateurs afin de proposer un système multi-agents élastique.

Remerciements. Nous adressons nos remerciements aux relecteurs pour leur travail minutieux et leurs précieux conseils.

Références

[1] Quentin Baert, Anne-Cécile Caron, Maxime Morge, Jean-Christophe Routier, and Kostas Stathis. An adaptive multi-agent system for task reallocation in a MapReduce job. *Journal of Parallel and Distributed Computing*, 153 :75–88, July 2021.

[2] Ellie Beauprez, Luc Bigand, Anne-Cécile Caron, Maxime Morge, and Jean-Christophe Routier. Réaffectation de tâches de la théorie à la pratique : état de l'art et retour d'expérience. In *Actes des JFSMA*, pages 51–60. Cépaduès, 2021.

[3] Ellie Beauprez, Anne-Cécile Caron, Maxime Morge, and Jean-Christophe Routier. Une stratégie de négociation multi-agents pour réduire la durée moyenne de réalisation. In *Actes des JFSMA*, pages 31–40. Cépaduès, 2021.

[4] Ellie Beauprez and Maxime Morge. Scala implementation of the Extended Multi-agents Situated Task Allocation. `https://gitlab.univ-lille.fr/maxime.morge/smastaplus`, 2020.

[5] J. Bruno, E. G. Coffman, Jr., and R. Sethi. Scheduling independent tasks to reduce mean finishing time. *Commun. ACM*, 17(7) :382–387, July 1974.

[6] Bo Chen, Chris N. Potts, and Gerhard J. Woeginger. *Handbook of combinatorial optimization*, chapter A review of machine scheduling : Complexity, algorithms and approximability, pages 1493–1641. Springer, 1998.

[7] Han-Lim Choi, Luc Brunet, and Jonathan P How. Consensus-based decentralized auctions for robust task allocation. *IEEE transactions on robotics*, 25(4) :912–926, 2009.

[8] Anastasia Damamme, Aurélie Beynier, Yann Chevaleyre, and Nicolas Maudet. The power of swap deals in distributed resource allocation. In *Proc. of AAMAS*, pages 625–633, 2015.

[9] WA Horn. Minimizing average flow time with parallel machines. *Operations Research*, 21(3) :846–847, 1973.

[10] Shijie Li, Rudy R. Negenborn, and Gabriel Lodewijks. A Distributed Constraint Optimization Approach for Vessel Rotation Planning. In *Computational Logistics*, pages 61–80. Springer, 2014.

[11] Lightbend. Akka is the implementation of the actor model on the JVM. `http://akka.io`, 2020.

[12] Ariel Rubinstein. Perfect equilibrium in a bargaining model. *Econometrica*, 50(1) :97–102, 1 1982.

[13] Andrea Schaerf, Y. Shoham, and Moshe Tennenholtz. Adaptive load balancing : A study in multi-agent learning. *Journal of Artificial Intelligence Research*, 2 :475–500, 1995.

[14] Joanna Turner, Qinggang Meng, Gerald Schaefer, and Andrea Soltoggio. Distributed strategy adaptation with a prediction function in multi-agent task allocation. In *Proc. of AAMAS*, pages 739–747, 2018.

[15] Michael P Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1 :1–23, 1993.

[16] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauly, Michael J. Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets : A fault-tolerant abstraction for in-memory cluster computing. In *Proc. of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI) ; San Jose, CA, USA*, pages 15–28, 2012.