Réaffectation de tâches de la théorie à la pratique : état de l'art et retour d'expérience

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

Réaffectation de tâches de la théorie à la pratique : état de l'art et retour d'expérience

Ellie Beauprez ellie.beauprez@univ-lille.fr
Luc Bigand luc.bigand.etu@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

La problématique de l'affectation efficace de tâches parmi des entités exécutantes est commune à de nombreuses applications réelles. Comme les problèmes d'allocation de ressources ou les problèmes d'appariement, l'affectation de tâches à des exécutants est un problème d'anti-coordination où des agents doivent adopter des plans d'action distincts. Dans cet article, nous proposons un état de l'art des méthodes multi-agents pour l'ordonnancement dynamique de tâches réalisées en parallèle par plusieurs exécutants ainsi que notre retour d'expérience sur leur mise en œuvre.

Mots-clés : *Résolution collective de problèmes*

Abstract

The problem of efficient task assignment is common to many real-world applications. Like multi-agent resource allocations or matching problems, assigning tasks to workers is an anti-coordination problem. where agents choose distinct plans of action. In this paper, we review the multi-agents dynamic scheduling methods for the simultaneous performance of tasks by several workers and we provide our feedback on their implementation.

Keywords: *Distributed Problem Solving*

1 Introduction

La problématique de l'affectation efficace de tâches parmi des entités exécutantes est commune à de nombreuses applications réelles pour le calcul parallèle [21, 20], le traitement de données massives [2], l'informatique en nuage [34], la logistique [22, 35, 26], la robotique collective [13, 9, 38, 41] ou les plateformes de *crowdsourcing* [12]. Depuis les travaux fondateurs de Kuhn [22], il y a plus de 65 ans, de nombreuses variantes du problème d'affectation de tâches ont été étudiées, pour lesquelles des algorithmes d'approximation et des heuristiques ont été proposés. Pour l'exemple, l'algorithme polynomial de Kuhn-Munkres, appelé méthode hongroise, minimise le coût total de l'affectation de n tâches à n exécutants [1].

Les systèmes multi-agents (SMA), en tant que paradigme pour la modélisation et l'implémentation de systèmes informatiques visent à résoudre des problèmes difficiles ou impossibles à résoudre pour un système monolithique. Dans les travaux fondateurs [11, 25], l'idée d'une distribution des ressources et des tâches est renforcée par le fait que les problèmes sont eux-mêmes de nature distribuée. Comme les problèmes d'allocation de ressources [8] ou les problèmes d'appariement [28], l'affectation de tâches à des entités est un problème d'anti-coordination où des agents doivent adopter des plans d'action distincts.

Le but de cet article est d'établir un état de l'art des travaux qui traitent d'affectation de tâches en se focalisant sur les problèmes d'ordonnancement où les tâches sont réalisées en parallèle par plusieurs exécutants et sur les méthodes de réaffectation qui adoptent une approche centrée individu. Nous fournissons une grille d'analyse inspirée par [14, 16, 35, 37, 20]. Cette grille nous permet de classer les travaux existants en fonction des problèmes abordés —leurs ingrédients (ressources, exécutants, tâches) et les performances visées. Nous proposons une typologie des méthodes de (ré)affectation en fonction de leurs caractéristiques (décentralisation, dynamicité, modèle, etc.) ainsi que notre retour d'expérience sur leur mise en œuvre dans nos différents prototypes [1, 32, 33, 3, 4].

L'article est structuré comme suit. Tout d'abord, nous évoquons une application à grand succès qui souligne l'intérêt des problèmes d'affectation de tâches (cf. section 2). Ensuite, nous pré-

1. Pour une implémentation récente, voir [32].

sentons les ingrédients des problèmes d'affectation de tâches (cf. section 3) en les illustrant à l'aide d'un cas d'étude jouet qui s'inspire de cette application. Nous passons en revue les principales méthodes qui ont été proposées pour cette famille de problème : les méthodes de référence (cf. section 4) et celles qui adoptent une approche centrée individu (cf. section 5). La section 6 synthétise notre grille d'analyse et dresse quelques perspectives.

2 Application à grand succès

Kiva [13] est un SMA utilisé pour la préparation de commande dans les centres de distribution Amazon©. Il automatise la collecte de produits pour des multiples commandes en supprimant les tâches de tri manuel et de réaménagement des entrepôts. Ce système complexe est composé de véhicules autonomes (*drive unit* —DU), qui collecte des *pods* d'inventaire (étagères) sur lesquels les produits sont stockés. Dans un entrepôt, les *pods* sont initialement placés dans la zone de réserve autour de laquelle se situent des stations d'inventaire (*inventory station* —IS). Les différents produits d'une même commande sont collectés en parallèle par les DU.

Pour répondre aux commandes en temps réel, la planification de la collecte, du réapprovisionnement et du rangement des produits n'est pas le résultat d'une optimisation globale mais obtenu par une heuristique mise en œuvre par des agents DU et des agents IS qui communiquent entre eux et prennent des décisions locales à partir de connaissances limitées. Afin d'assigner les tâches de collecte des produits, l'heuristique sélectionne le DU et le *pod* en fonction de leur localisation et des produits disponibles sur le *pod*. Il est important de noter que l'heuristique n'est pas effectivement distribuée mais l'approche centrée individu constitue un cadre conceptuel pour la représentation modulaire du problème de planification.

3 Problème d'affectation de tâches

Nous proposons ici un typologie des problèmes d'affectation de tâches que nous illustrons par un cas d'étude qui parodie Kiva, c.-à-d. la collecte de colis par des livreurs.

Notre environnement est une grille de cellules, chacune contenant au plus une entité, passive (objet) ou active. Les objets sont des colis de poids différents dispersés et la destination où ils doivent être livrés. Les entités actives sont des livreurs ou des équipes de livreurs dont la taille détermine le poids maximal des colis qu'elles peuvent transporter. Une entité active peut : (a) se déplacer vers l'une des cases adjacentes ; (b) charger un colis situé dans une cellule adjacente si elle en a la capacité ; (c) déposer un colis si elle est dans une cellule adjacente à la destination.

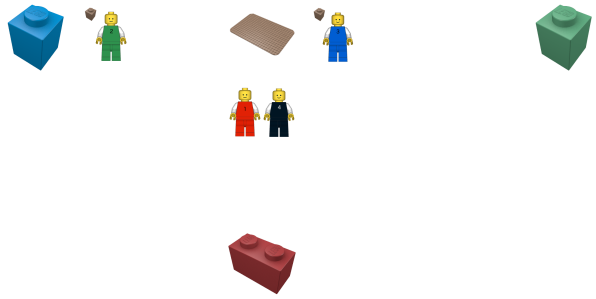

FIGURE 1 – Un colis ciblé par une entité est coloré [31]

Au delà de notre exemple, un problème d'affectation de tâches est caractérisé par les ressources utilisées, le type de tâches et les exécutants à qui elles sont affectées.

Ressources. Afin d'être exécutée, une tâche nécessite généralement l'utilisation de ressources. Leurs propriétés ont des impacts drastiques sur la nature du problème [8]. Ces ressources sont des données, des capacités de calculs, de la mémoire, des objets physiques ou de l'énergie. Elles sont présentes en quantité limitée dans le système, éventuellement munie d'une taille, en un unique exemplaire ou répliquées en plusieurs exemplaires indistinguables. Alors qu'une information est discrète, une ressource énergétique est continue et donc divisible selon une quantité. Les ressources périssables ont une valeur/taille décroissante dans le temps. Lorsqu'elle est utilisée, une ressource est soit consommée par l'exécutant et disparaît ; soit réemployée pour une autre tâche, elle est dite statique. Lorsqu'une ressource est partageable, elle peut être attribuée à plusieurs agents. Une ressource est transférable si elle est utilisable par un autre exécutant que celui qui la possède initialement. Selon la typologie du système, une ressource est locale à un exécutant s'il y accède directement. Alors qu'une ressource non locale et non partageable est inaccessible, Une ressource non locale mais partageable est dite distante.

Dans notre exemple, les colis sont des exemplaires uniques muni d'un poids. Ces ressources sont discrètes, indivisibles, non périssables et

consommables. Si une entité ne peut pas transmettre un colis à une autre entité et que l'on considère une équipe comme une seule et même entité alors les ressources ne sont ni partageables ni transférables. Bien que toutes accessibles, la localité des ressources est variable pour les différents exécutants.

Tâches. Qu'elles consistent à explorer un environnement ou à traiter des données, les propriétés des n tâches, qui sont des éléments centraux des problèmes d'affectation, ont également un impact radical sur le problème indépendamment de la méthode de résolution [37]. Qu'elle requiert ou pas des ressources, la réalisation d'une tâche après sa libération vise à produire un résultat, éventuellement avant une date butoir. Comme les ressources, les tâches sont uniques dans le système ou des instances de différents types. Alors que posséder une ressource représente un avantage (c.-à-d. une valeur positive), devoir accomplir une tâche, forcément consommable, représente une charge (c.-à-d. une valeur négative). L'estimation du coût d'une tâche, c.-à-d. son temps d'exécution, est un défi soumis à des incertitudes, des changements et des aléas. Lorsque des conditions préalables doivent être validées pour qu'elles soient réalisables, les tâches sont soumises à des contraintes de précédence induites comme cela peut être le cas dans un procédé de production. Une tâche est préemptive si son exécution peut être suspendue pour être reprise plus tard sans en altérer le résultat, éventuellement avec un surcoût voire par un autre exécutant. Un job désigne généralement une tâche divisible en plusieurs sous-tâches atomiques.

Dans notre exemple, chaque tâche consiste à collecter un colis : atteindre une case adjacente à ce colis, le charger, atteindre une case adjacente à la destination et l'y déposer. Comme certaines actions peuvent échouer (e.g. un déplacement vers une case occupée) ou être non sollicitées (e.g. une entité refoulée par une autre), le coût d'une tâche n'est qu'une estimation du nombre de pas de simulations nécessaires pour son accomplissement. Attendu que les entités peuvent ne porter qu'un colis à la fois et ne le déposer que sur la destination, les tâches sont atomiques, non-préemptives et indépendantes.

Exécutants. Qu'ils soient des robots, des machines ou des nœuds de calculs, les m exécutants doivent être compétents pour tout ou partie des tâches. Même si chaque exécutant a les compétences pour réaliser n'importe quelle tâche, leurs efficacités dans la réalisation des tâches peuvent être homogènes (noté P_m) ou hétérogènes (noté R_m). Le coût d'une tâche n'est pas nécessairement une propriété intrinsèque, mais elle varie d'un exécutant à un autre si les ressources requises pour une tâche sont inégalement réparties parmi les exécutants. De plus, la topologie est essentielle pour déterminer la faisabilité et la qualité d'une affectation. Le réseau d'accointances (qu'il soit complet, une étoile, une ligne, un anneau, une grille, un arbre ou invariant d'échelle) et les distances entre les exécutants peuvent limiter voire interdire la circulation des ressources et des tâches. D'après la taxonomie de [14], les tâches sont mono-exécutant si chacune d'elles est affectée à un seul exécutant ou multi-exécutants sinon. Réciproquement, les exécutants sont mono-tâche si chacun est affecté à une seule tâche ou multi-tâches si chacun réalise séquentiellement plusieurs tâches.

Dans notre exemple, une tâche de collecte est réalisable par une entité active ayant la taille requise. Le coût d'une collecte, qui dépend de l'entité exécutante (c.-à-d. de sa position), varie au cours de la simulation. Comme elles peuvent enchaîner les collectes, les entités actives sont multi-tâches. Nous considérons qu'une équipe de livreurs est une seule et même entité et donc, comme dans la suite de cette section, les tâches sont considérées comme mono-exécutant.

En résumé, un problème d'affectation de tâches consiste à les répartir parmi les exécutants en fonction de leur coût. Afin de définir des métriques d'évaluation, nous formalisons un tel problème de la manière suivante :

Définition 1 (Problème)**.** *Un problème d'affectation de tâches à des exécutants est un triplet* $\mathcal{P} = \langle \mathcal{S}, \mathcal{T}, c \rangle$ *où :*

- $\mathcal{S} = \langle \Omega, \mathcal{V}, \mathcal{R}, \sigma \rangle$ *est un système avec m exécutants* $\Omega = \{\omega_1, \ldots, \omega_m\}$ *dans un graphe d'accointances* $\mathcal{V} \subset \Omega \times \Omega$ *et munis de ressources* $\mathcal{R}$ *qui leur sont assignées selon une fonction* $\sigma : \mathcal{R} \to 2^{\Omega}$ *;*
- $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$ *est un ensemble de n tâches ;*
- $c : \mathcal{T} \times \mathcal{R} \mapsto \mathbb{R}_+^*$ *est la fonction de coût des tâches.*

Le coût d'une tâche pour un exécutant (éventuellement ∞ si elle n'est pas réalisable) se déduit de la localisation des ressources. Même si la plupart des travaux s'abstraient de la répartition des ressources qui est implicite dans la fonction de coût, cette répartition peut guider les heuristiques de (ré)affection (e.g. [21, 2]).

Affectation. Une affectation de tâches est une répartition parmi les exécutants des tâches selon des lots ordonnés où toutes les tâches sont affectées, ici à un seul exécutant. Formellement,

Définition 2 (Affectation)**.** *Soit $\mathcal{P}$ un problème d'affectation de tâches. Une affectation est un vecteur de m lots de tâches $\vec{A} = (\vec{B}_1, \ldots, \vec{B}_m)$ où $\vec{B}_i = (B_i, \prec_i)$ est un ensemble de tâches ($B_i \subseteq \mathcal{T}$) affectées à l'exécutant $\omega_i \in \Omega$ ordonné selon un ordre total strict ($\prec_i \subseteq \mathcal{T} \times \mathcal{T}$) tel que $\tau_j \prec_i \tau_k$ signifie que si $\tau_j, \tau_k \in B_i$ alors la tâche τ_j est exécutée avant la tâche τ_k par ω_i. Une allocation $\vec{A}$ vérifie que :*

$$\forall \tau \in \mathcal{T}\ \exists \omega_i \in \Omega,\ \tau \in B_i \quad (1)$$

$$\forall \omega_i \in \Omega, \forall \omega_j \in \Omega \setminus \{\omega_i\},\ B_i \cap B_j = \emptyset \quad (2)$$

Il est important de noter que réduire ce problème à un problème d'allocation de ressources associées à des valeurs négatives consiste à ignorer l'ordonnancement des tâches.

Définition 3 (Ordonnancement)**.** *Soit τ la tâche affectée à l'exécutant ω_i à la date de libération (release date) t^0_τ dans l'affectation de tâches $\vec{A}$ pour un problème $\mathcal{P}$. Son ordonnancement se caractérise par :*

— *un **délai d'attente** (delay),*

$$\delta(\tau, \vec{A}) = \sum_{\tau' \in B_i | \tau' \prec_i \tau} c(\tau', \omega_i) \quad (3)$$

— *une **durée de réalisation** (response time),*

$$C_\tau(\vec{A}) = \delta(\tau, \vec{A}) + c(\tau, \omega_i) \quad (4)$$

— *une **date d'achèvement** (completion date)*

$$t^E_\tau(\vec{A}) = t^0_\tau + C_\tau(\vec{A}) \quad (5)$$

Le délai d'attente d'une tâche correspond aux temps d'exécution estimés (coûts) des tâches qui la précèdent dans le lot correspondant (Eq. 3). On fait ici l'hypothèse que les exécutants ne sont jamais inactifs. La durée de réalisation de la tâche correspond au temps d'attente avant que la tâche soit entamée plus l'estimation de son temps d'exécution (Eq. 4). Elle dépend donc de la charge de l'exécutant. Contrairement au coût de la tâche, le délai d'attente et donc la durée de réalisation dépendent de l'ordre d'exécution. La date d'achèvement d'une tâche est l'instant auquel elle est terminée (Eq. 5).

Objectifs. Pour évaluer la qualité d'une affectation de tâches, au delà de la fiabilité du système [20] ou de la qualité de service [34], notre étude se focalise sur les mesures de performances envisageables.

Définition 4 (Métriques)**.** *Une affectation de tâches $\vec{A}$ pour un problème $\mathcal{P}$ se caractérise par :*

— *la charge de travail (workload) des exécutants,*

$$w_i(\vec{A}) = \Sigma_{\tau \in B_i} c(\tau, \omega_i)\ , \forall \omega_i \in \Omega \quad (6)$$

— *le débit de réalisation (throughput),*

$$W(\vec{A}) = \Sigma_{\omega_i \in \Omega} w_i(\vec{A}) \quad (7)$$

— *la durée globale de réalisation (makespan),*

$$C_{max}(\vec{A}) = \max_{\tau \in \mathcal{T}} t^E_\tau(\vec{A}) \quad (8)$$

— *la durée moyenne de réalisation (flowtime),*

$$C(\vec{A}) = \frac{1}{m} \Sigma_{\tau \in \mathcal{T}} C_\tau(\vec{A}) \quad (9)$$

Tandis que la charge de travail d'un exécutant est une mesure individuelle, les autres métriques sont macroscopiques. Le débit de réalisation est équivalent au nombre de tâches réalisées par unité de temps (Eq. 7). La durée moyenne de réalisation mesure le temps écoulé en moyenne entre la date de libération d'une tâche et sa date d'achèvement (Eq. 9). La durée globale de réalisation représente le temps nécessaire à la réalisation de l'ensemble des tâches (Eq. 8). Elle correspond donc à la charge maximale des exécutants, c.-à-d. celle de l'agent limitant (*bottleneck agent*) :

$$\mathrm{C}_{max}(\vec{\mathrm{A}}) = \max_{\omega_i \in \Omega} \{\mathrm{w}_i(\vec{\mathrm{A}})\} \quad (10)$$

Les objectifs [35, 20] qui consistent à minimiser ces métriques sont divers et variés :

— le débit de réalisation (Eq. 7) est une mesure utilitaire de la performance du système du point de vue des exécutants qui est agnostique vis-à-vis de l'ordonnancement ;

— la charge maximale des exécutants (Eq. 8-10) est une mesure égalitaire, du point de vue des exécutants, de l'équilibrage des charges qui est également agnostique vis-à-vis de l'ordonnancement ;

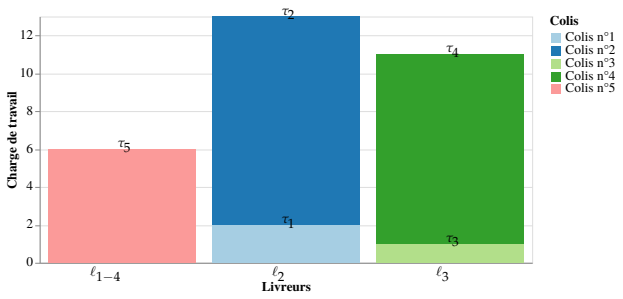

FIGURE 2 – Ordonnancement des tâches pour notre exemple

— la durée moyenne (ou totale) de réalisation est une mesure utilitaire de la performance du système du point de vue des tâches et donc de leur commanditaire. Cet objectif qui consiste à réduire les temps de réponse, c.-à-d. les délais d'attente, peut être raffiné en distinguant les différents types de tâches.

Ces objectifs parfois antagonistes peuvent également être combinés à travers des fonctions multi-objectifs.

Dans notre exemple, les métriques sont $\mathrm{W}(\vec{\mathrm{A}}) = 30$, $\mathrm{C}_{max}(\vec{\mathrm{A}}) = 13$ et $\mathrm{C}(\vec{\mathrm{A}}) = \frac{33}{5}$. L'échange de cible entre les livreurs bleu et vert fait décroître le *makespan* qui approxime (en négligeant l'ordre des collectes) le nombre de pas de simulation c.-à-d. le nombre de pas effectués par l'entité la plus sollicitée au cours de la simulation.

4 Méthodes d'affectation

La théorie de l'ordonnancement [7] propose des méthodes d'affectation hors-ligne. Nous présentons ici les méthodes qui servent de référence (*baseline*) sur les problèmes les plus classiques [2].

La règle du « plus court processus en premier » – *shortest processing time first* (SPT) – est une méthode très simple qui minimise le débit de réalisation pour un problème avec un seul exécutant multi-tâches et n tâches mono-exécutant [10]. Ce résultat se généralise au problème avec m exécutants multi-tâches si le coût d'une tâche est identique d'un exécutant à l'autre (P_m). Selon cet ordonnanceur, la plus petite tâche est affectée au premier exécutant, la seconde plus petite tâche au second exécutant, ..., la $(m+1)^{ime}$ plus petite tâche suit la plus petite tâche dans le lot du premier exécutant, la $(m+2)^{ime}$ plus petite tâche suit la seconde plus petite tâche dans le lot du deuxième exécutant, etc.

2. Pour une implémentation récente de ces ordonnanceurs, voir [32].

Le problème d'ordonnancement qui consiste à minimiser la durée totale de réalisation (*flow-time*) avec m exécutants multi-tâches et n tâches mono-exécutant dont les coûts dépendent de l'entité exécutante (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 [17]. Reposant sur l'algorithme de Ford-Fulkerson,la complexité de l'algorithme décrit par [6] est $\mathcal{O}(max(mn^2, n^3))$.

Le problème d'ordonnancement qui consiste à minimiser la durée globale de réalisation (*makespan*) avec m exécutants multi-tâches et n tâches mono-exécutant dont les coûts dépendent de l'entité exécutante est NP-difficile [18]. Les algorithmes pseudo-polynomiaux connus pour ce problème sont : une heuristique de réaffectation (*earliest completion time* – ECT) [19], des heuristiques en deux phases reposant sur la programmation linéaire [23], des méthodes de recherche locale [15], ou l'algorithme classique de séparation et évaluation [29].

D'une manière générale, les problèmes d'affectation sont des problèmes d'optimisation sous contraintes pour lesquels les résultats obtenus par des solveurs polyvalents tel que IBM® ILOG® CPLEX® sont médiocres. Toutefois, ces problèmes peuvent être résolus de manière approchée par différentes heuristiques comme la montée en gradient, le recuit simulé ou les algorithmes génétiques [30]. L'affectation obtenue avec ces algorithmes d'approximation est acceptable mais les temps d'ordonnancement sont rédhibitoires avec un grand nombre de tâches (plus de 100 000).

Les limites des méthodes d'affectation sont les suivantes.

La réactivité. Les méthodes classiques d'ordonnancement sont majoritairement des processus d'affectation instantanés (*instantaneous assignment* – IA) et non pas prolongés dans le temps (*time-extented assignment* – TA) [14]. Les modèles d'affectation tâches-exécutants pour maximiser le gain des exécutants et des commanditaires sur les plateformes *crowd-sourcing* où les tâches comme les exécutants arrivent en continu (OTA-TSA-*Online Task Assignment with Two-Sided Arrival*) suppose un motif d'arrivée des types de tâches et des types d'exécutants dont les distributions de probabilités sont indépendantes et prédites à partir de l'historique [12]. Toutefois, comme indiqué dans la section 3, l'estimation inexacte des temps d'exécution, aggravée

par des perturbations (consommation/libération de tâches, ralentissement des exécutants, etc.), peuvent nécessiter d'importantes modifications de l'affectation existante pour qu'elle reste optimale. Plutôt que de recalculer en continu une affectation optimale, quelques modifications locales au cours de l'exécution des tâches peuvent améliorer la performance de l'affectation.
Le passage à l'échelle. Un contrôleur global constitue un goulot d'étranglement en matière de performance, car il doit collecter les informations d'état de l'ensemble du système en temps réel. De plus, les problèmes d'affectation sont souvent intractables à cause de la combinatoire des ordonnancements.

5 Méthodes de réaffectation

Le paradigme multi-agents est approprié pour la conception et l'implémentation de mécanismes distribués et adaptatifs de réaffectation dynamique de tâches-exécutants, en particulier les modèles SMA dont la structure est dynamique et dénuée de relation d'autorité [16]. Ces modèles se distinguent de par la nature des tâches et des agents, qu'ils représentent les exécutants ou les commanditaires des tâches (cf. tableau 1).

Modèle	Agents	Tâches
Coalition	coopératifs	multi-exécutants
Équipe	coopératifs	divisibles
Marché	égoïstes	atomiques

TABLE 1 – Modèles pour la réaffectation

5.1 Coalition

La formation d'une coalition se justifie si la réalisation d'une tâche nécessite plus d'un exécutant comme le suggère notre exemple (cf. figure 1) ou si son coût diminue avec le nombre d'exécutants affectés. Le problème de formation des coalitions pour l'affectation de tâches est équivalent à un problème de couverture ou de partitionnement par un ensemble de coût minimal. Ces problèmes sont NP-difficiles, car le nombre de coalitions potentielles est exponentiel.

En particulier, Shehory et Kraus proposent des algorithmes décentralisés gloutons et *anytime* pour l'assignation de tâches via la formation de coalitions [40]. Les auteurs supposent que les tâches sont multi-exécutants et soumises à des contraintes de précédence. Les capacités/efficacités des exécutants sont hétérogènes. Chaque agent est membre d'une seule coalition et une coalition est mono-tâche. Les algorithmes proposés procèdent en deux étapes :

1. le calcul distribué du coût des coalitions. L'algorithme permet à un agent de choisir l'ensemble des coalitions pour lesquelles il calcule le coût et en informe ses pairs ;
2. une procédure gloutonne, itérative et distribuée où les agents choisissent les coalitions qu'ils préfèrent et les forment. Afin de choisir les coalitions avec les coûts par agent minimaux, chaque agent cherche parmi les coalitions dont il a calculé les coûts celle dont le coût par agent est minimal et l'annonce à ses pairs, puis le groupe sélectionne une coalition. Comme les coalitions doivent être disjointes, les agents impliqués sont exclus de la liste des coalitions potentielles à recalculer.

Le nombre de coalitions envisagées est réduit en limitant la taille maximale des coalitions, car les petites coalitions, moins coûteuses en communications, sont privilégiées. Il est important de noter que la plupart des algorithmes multi-agents ne sont pas opérationnels. Ils ne sont pas implémentables en l'état mais ils nécessitent d'être reformulés en termes de comportement pour être mis en œuvre [3].

5.2 Équipe

Une équipe est composée d'agents coopératifs travaillant ensemble à la réalisation d'un objectif commun. Comme pour une coalition, une équipe vise à maximiser une fonction objectif globale plutôt que la satisfaction individuelle des membres mais les tâches sont mono-exécutant. Chacun muni de compétences qui lui sont propres correspondant aux types de tâches qu'il peut accomplir, les agents communiquent avec leurs voisins dans le graphe d'accointances et jouent un rôle dans la réalisation de l'ensemble des tâches et sous-tâches. Les agents ont accès à une représentation explicite des informations sur les objectifs à accomplir, les autres membres et le plan d'actions commun afin de guider leur comportement et de s'adapter en cas d'évènements imprévus.

En particulier, Lesser et al. proposent un modèle hiérarchique de représentation des tâches pour la coordination qui est indépendant du domaine : affecter les ressources, assigner les tâches et les ordonnancer [24]. L'objectif premier de *Generalized Partial Global Planning* (GPGP) est de maximiser l'utilité combinée de l'ensemble des agents par la réalisation de ses objectifs de plus

haut niveau. GPGP est un modèle de coordination basée sur la planification des activités. La coordination des agents consiste en une recherche coopérative sur un arbre d'activité qui évolue de façon dynamique. Les activités (tâches ou objectifs) et leur relation de dépendance sont représentées dans un arbre de but de type ET/OU. De plus, les relations de dépendance entre les ressources (consommables ou pas) nécessaires à la réalisation des activités sont représentées par un graphe biparti. Le modèle GPGP vise à réduire les comportements incohérents des agents : le déséquilibrage de charges, la réalisation de sous-tâches inutiles ou redondantes, voire l'inactivité. Un agent arbitre entre son choix individuel et celui des autres agents quand il a une évaluation biaisée de la solution optimale, car il ne dispose que d'une information partielle. Le modèle GPGP permet de résoudre des problèmes en temps réel nécessitant une gestion des ressources et impliquant des interdépendances complexes entre tâches devant être réalisées par différents agents. Cette approche suppose que l'effort nécessaire à la coordination (raisonnement et communication) est négligeable par rapport aux temps d'exécution des tâches à coordonner. Dans le cas contraire, l'ordonnancement des tâches est déterminé par le système (distribué) sous-jacent.

5.3 Place de marché

S'inspirant de théorie économique, la « programmation orientée marché » aborde les problèmes de planification distribuée à travers la recherche d'un équilibre pour un jeu non-coopératif [42]. Les agents délèguent des (lots de) tâches voire les échangent. Ces transactions, effectuées via des carnets d'ordre ou des protocoles de négociation visent à réaffecter les tâches de façon efficace. Il est important de noter que, contrairement à une équipe, une place de marché suppose que les contraintes et les objectifs sont complètement distribués. Leur communication est considérée comme une manipulation stratégique qui menace l'équilibre du marché. La « programmation orientée marché » est un patron de conception pour l'ordonnancement distribué où le concepteur définit une configuration de marché, c.-à-d. les ressources, les tâches et les agents (leurs objectifs et leurs comportements) [13, 38]. Par exemple, MASTA est un système multi-agents qui exploite la localité de ressources transférables pour réaffecter en continu des tâches afin de minimiser la durée globale de réalisation (*makespan*) [2]. Parmi les méthodes multi-agents de réaffectation, on distingue trois familles.

CBBA. L'algorithme à base de consensus (CBBA – *Consensus Based Bundle Algorithm*) [9] 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 (Eq.7) avec des ressources en carburant limitées [41]. 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.

DCOP. Les problèmes de réaffectation peuvent être représentés sous la forme d'un problème d'optimisation sous contraintes distribué – *Distributed Constraint Optimization Problems* (DCOP). De nombreuses méthodes ont été développées pour la recherche d'une solution optimale à un DCOP qui est un problème NP-difficile (voir [36] pour une synthèse récente). Les principales difficultés dans la mise en œuvre de ces méthodes pour la réaffectation de tâches résident dans :

1. la représentation d'un problème réaliste sous la forme d'un DCOP, voire de plusieurs sous-problèmes COP, nécessite une expertise de la méthode de résolution [26] ;
2. la mise en correspondance de la fonction objectif avec une mesure de performance autre que le débit de réalisation (cf. définition 4). Par exemple, MGM2 suppose que cette fonction est monotone [27].

MARL. Les problèmes de réaffectation peuvent également être modélisés via des processus de décision markoviens [5], en particulier des processus de décision markoviens partiellement observables décentralisés – *Decentralized Partially Observed Markov Decision Process* (Dec-POMDP). L'optimisation d'un Dec-POMDP à horizon fini est un problème NEXPTIME. Les méthodes de résolution approchée ne peuvent être appliquées que sur de très petites instances de problèmes, elles ne passent pas à l'échelle. Au-delà de ces méthodes de planification hors-ligne, l'apprentissage multi-agents par renforcement (MARL-*Multi-Agent Reinforcement Learning*) nécessite une connaissance parfaite de l'environnement et requiert une phase d'apprentissage [39].

	Ressources	Tâches	Exécutants	Objectif	Dynamique	Décentralisé	Technique/ Modèle
(Kuhn-Munkres, 1955) [22]	—	n mono-exécutant	R_n mono-tâche	$\mathrm{W}(\vec{\mathrm{A}})$	✗	✗	LP
(SPT, 1967) [10]	—	n mono-exécutant	P_m multi-tâches	$\mathrm{W}(\vec{\mathrm{A}})$	✗	✗	heuristique
(Bruno et al., 1974) [6]	—	n mono-exécutant	R_m multi-tâches	$\mathrm{C}(\vec{\mathrm{A}})$	✗	✗	LP
(ECT, 1977) [19]	—	n mono-exécutant	R_m multi-tâches	$\mathrm{C}_{max}(\vec{\mathrm{A}})$	✗	✗	heuristique
(OTA-TSA, 2018) [12]	—	n mono-exécutant	R_m multi-tâches	$\mathrm{C}(\vec{\mathrm{A}})$ $\oplus\mathrm{W}(\vec{\mathrm{A}})$	✓	✗	LP + variable aléatoire
(Shehory et Krause, 1998) [40]	—	n multi-exécutant	R_m mono-tâche	$\mathrm{W}(\vec{\mathrm{A}})$	✓	✓	coalition
(GPGP, 2004) [24]	consommables ou pas	n mono-exécutant	R_m muli-tâches	$\mathrm{W}(\vec{\mathrm{A}})$	✓	✓	équipe
(Kiva, 2011) [13]	consommables répliquées	n mono-exécutant	R_m muli-tâches	$\mathrm{C}(\vec{\mathrm{A}})$ $\oplus\mathrm{W}(\vec{\mathrm{A}})$	✓	✓	marché
(MASTA, 2019) [2]	transférables duplicables	n mono-exécutant	R_m muli-tâches	$\mathrm{C}_{max}(\vec{\mathrm{A}})$	✓	✓	marché
(Turner et al., 2018) [41]	limitées	n mono-exécutant	R_m en ligne muli-tâches	$\mathrm{W}(\vec{\mathrm{A}})$	✓	✓	CBBA + classification
(Li et al., 2014) [26]	—	n mono-exécutant	P_m muli-tâches	$\mathrm{C}(\vec{\mathrm{A}})$ $\oplus\mathrm{W}(\vec{\mathrm{A}})$	✓	✓	DCOP
(Schaerf et al., 1995) [39]	—	n mono-exécutant	P_m muli-tâches	$\mathrm{W}(\vec{\mathrm{A}})$	✓	✓	MARL

TABLE 2 – Grille d'analyse des méthodes d'affectation (haut) ou de réaffectation (bas)

6 Synthèse

Dans cet article, nous avons proposé un état de l'art des travaux qui traitent d'affectation de tâches en nous focalisant sur les méthodes multi-agents pour l'ordonnancement dynamique de tâches réalisées en parallèle par plusieurs exécutants. Le tableau 2 synthétise l'ensemble des travaux évoqués ici selon notre grille d'analyse. La partie gauche du tableau discerne les différents problèmes abordés : leurs ingrédients (ressources, tâches, exécutants) et les objectifs visés. La partie droite révèle les caractéristiques de ces méthodes ainsi que les techniques et modèles sous-jacents. La partie supérieure du tableau contient les méthodes d'affectation alors que la partie inférieure les méthodes de réaffectation où l'assignation est dynamique et continue.

Parmi les avantages du paradigme multi-agents, nous avons mentionné la distribution d'heuristiques pour des problèmes intractables à cause de la combinatoire des ordonnancements pour permettre le passage à l'échelle. De plus, l'approche centrée individu constitue un cadre conceptuel autorisant une représentation modulaire des problèmes d'affectation combinant les objectifs antagonistes des exécutants et des commanditaires (notés $\mathrm{C}(\vec{\mathrm{A}}) \oplus \mathrm{W}(\vec{\mathrm{A}})$). Les SMAs permettent de prendre en compte la topologie du réseau et la localisation des ressources. Peuvent être abordés des problèmes d'affectation où la réalisation d'une tâche nécessite plus d'un exécutant ainsi que les problèmes complexes où les tâches sont soumises à des contraintes de précédence ou à des ressources localisées qui nécessitent de la coordination. Intrinsèquement réactifs, 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.) sans pour autant reposer sur un historique comme les méthodes d'apprentissage. Le choix du modèle sous-jacent dépend de la nature des tâches et des agents envisagées (cf. tableau 1).

Pour aborder les applications réelles, la difficulté réside dans la formulation de systèmes complexes d'assignation tâches-exécutants décentralisés et adaptatifs qui sont divers de par leurs ingrédients et leurs objectifs. La conception des comportements individuels joués de manière asynchrone par les exécutants et les commanditaires doit aboutir à l'émergence d'affectations faisables qui combinent les objectifs des exécutants avec ceux des commanditaires.

Remerciements. Nous remercions le comité de programme des JFSMA qui, par ses remarques,

nous a permis d'améliorer cet article.

Références

[1] Quentin Baert, Anne-Cécile Caron, Maxime Morge, and Jean-Christophe Routier. MAS4Data - Multiagent systems for analyzing very large data sets. https://github.com/cristal-smac/mas4data, 2019.

[2] Quentin Baert, Anne-Cécile Caron, Maxime Morge, Jean-Christophe Routier, and Kostas Stathis. Stratégie situationnelle pour l'équilibrage de charge. In *Actes des 27ièmes journées francophones sur les systèmes multi-agents*, pages 9–18. Cépaudès, 2019.

[3] Ellie Beauprez, Luc Bigand, and Maxime Morge. Musketeers : Scala library of algorithms for coalition formation. https://gitlab.univ-lille.fr/maxime.morge/musketeers, 2020.

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

[5] Aurélie Beynier, François Charpillet, Daniel Szer, and Abdel-Illah Mouaddib. DEC-MDP / DEC-POMDP. In Olivier Sigaud Olivier Buffet, editor, *Markov Decision Processes in Artificial Intelligence*, pages 277–313. Wiley-ISTE, 2010.

[6] 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.

[7] 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.

[8] Yann Chevaleyre, Paul E Dunne, Ulle Endriss, Jerome Lang, Michel Lemaitre, Nicolas Maudet, Julian Padget, Steve Phelps, Juan A Rodrguez-Aguilar, and Paulo Sousa. Issues in multiagent resource allocation. *Informatica*, 30 :3–31, 2006.

[9] 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.

[10] R.W. Conway, W.L. Maxwell, and L.W. Miller. *Theory of Scheduling*. Addison-Wesley, Reading, MA, 1967.

[11] Randall Davis and Reid G. Smith. Negotiation as a metaphor for distributed problem solving. *Artif. Intell.*, 20(1) :63–109, 1983.

[12] John P. Dickerson, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. Assigning tasks to workers based on historical data : Online task assignment with two-sided arrivals. In *Proc. of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 318–326, 2018.

[13] John Enright and Peter R. Wurman. Optimization and coordinated autonomy in mobile fulfillment systems. In *Automated Action Planning for Autonomous Mobile Robots, Papers from the 2011 AAAI Workshop, San Francisco, California, USA, August 7, 2011*, volume WS-11-09 of *AAAI Workshops*. AAAI, 2011.

[14] Brian P. Gerkey and Maja J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The international journal of robotics research*, 23(9) :939–954, 2004.

[15] A. M. A. Hariri and N. Potts, Chris. Heuristics for scheduling unrelated parallel machines. *Computers & operations research*, 18(3) :323–331, 1991.

[16] Bryan Horling and Victor Lesser. A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, 19(4) :281–316, 2004.

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

[18] Ellis Horowitz and Sartaj Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the ACM*, 23(2) :317–327, 1976.

[19] Oscar H. Ibarra and Chul E. Kim. Heuristic Algorithms for Scheduling Independent Tasks on Nonidentical Processors. *Journal of ACM*, 24(2) :280–289, 1977.

[20] Yichuan Jiang. A survey of task allocation and load balancing in distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, 27(2) :585–599, 2016.

[21] Yichuan Jiang and Zhaofeng Li. Locality-sensitive task allocation and load balancing in networked multiagent systems : Talent

versus centrality. *J. Parallel Distrib. Comput.*, 71(6) :822–836, 2011.

[22] Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2) :83–97, 1955.

[23] Jan Karel Lenstra, David B. Shmoys, and Eva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical programming*, 46(1-3) :259–271, 1990.

[24] V. Lesser, K. Decker, T. Wagner, N. Carver, A. Garvey, B. Horling, D. Neiman, R. Podorozhny, M. N Prasad, A. Raja, R. Vincent, and X. Q. Xuan, P. Zhang. Evolution of the GPGP/TAEMS domain-independent coordination framework. *Autonomous agents and multi-agent systems*, 9(1-2) :87–143, 2004.

[25] Victor R. Lesser and Daniel D. Corkill. The distributed vehicle monitoring testbed : A tool for investigating distributed problem solving networks. *AI Mag.*, 4(3) :15–33, 1983.

[26] 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.

[27] Rajiv T Maheswaran, Jonathan P Pearce, and Milind Tambe. Distributed algorithms for dcop : A graphical-game-based approach. In *Proc. of the 17th International Conference on Parallel and Distributed Computing Systems (ISCA)*, pages 432–439, 2004.

[28] David F. Manlove. *Algorithmics of Matching Under Preferences*. World Scientific, 2014.

[29] Silvano Martello, François Soumis, and Paolo Toth. Exact and approximation algorithms for makespan minimization on unrelated parallel machines. *Discrete applied mathematics*, 75(2) :169–188, 1997.

[30] Ethel Mokotoff. Parallel machine scheduling problems : A survey. *Asia-Pacific Journal of Operational Research*, 18(2) :193, 2001.

[31] Maxime Morge. Répartition des tâches pour la collecte de colis : démonstration. Actes des 27ièmes journées francophones sur les systèmes multi-agents, July 2019. Poster.

[32] Maxime Morge. ScaMATA : Scala implementation of Multi-agents Task Allocation. `https://github.com/cristal-smac/ScaMATA`, 2019.

[33] Maxime Morge. ScaSMATA : Scalable Situated Multi-Agent Task Allocation. `https://github.com/cristal-smac/ScaMATA`, 2019.

[34] Amro Najjar, Olivier Boissier, and Gauthier Picard. Négociation adaptative pour l'acceptabilité des services d'un fournisseur saas. In *Actes des ièmes journées francophones sur les systèmes multi-agents*, pages 85–94. Cépaduès, 2017.

[35] David W. Pentico. Assignment problems : A golden anniversary survey. *European Journal of Operational Research*, 176(2) :774 – 793, 2007.

[36] Gauthier Picard. Optimisation sous contraintes distribuée : une introduction au domaine. In *Actes des 26ièmes journées francophones sur les systèmes multi-agents*, pages 43–52. Cépaduès, 2018.

[37] Michael L. Pinedo. *Scheduling. Theory, Algorithms, and Systems. Third Edition*. Springer, 2008.

[38] Cyril Poulet, Vincent Corruble, and Amal El Fallah Seghrouchni. Travailler en équipe : le choix social appliqué au problème de la patrouille multi-agents. In *Actes des 20ièmes journées francophones sur les systèmes multi-agents*, pages 65–74. Cepadues Editions, 2012.

[39] 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.

[40] Onn Shehory and Sarit Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1-2) :165–200, 1998.

[41] Joanna Turner, Qinggang Meng, Gerald Schaefer, and Andrea Soltoggio. Distributed strategy adaptation with a prediction function in multi-agent task allocation. In *Proc. of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 739–747, 2018.

[42] 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.