Optimisation multi-agent par partitionnement adaptatif de l'espace de conception

D. Villanueva[†‡] villanueva@emse.fr
G. Picard[†] picard@emse.fr
R. Le Riche[†⋆] leriche@emse.fr
R. T. Haftka[‡] haftka@ufl.edu

[†]Institut Henri Fayol, ENSM.SE, Saint-Étienne, France

[‡]University of Florida, Gainesville, FL, 32611, USA

[⋆]CNRS UMR 5146, Saint-Étienne, France

Résumé

Nous proposons l'usage de systèmes multi-agents pour résoudre des problèmes d'optimisation impliquant des simulateurs numériques coûteux. Il est alors usuel de remplacer certains appels aux simulateurs numériques par des appels à des métamodèles. L'idée proposée dans cet article est d'assigner les métamodèles adéquats à chaque sous-région afin de (i) *rendre l'optimisation moins coûteuse,* (ii) *générer une méthode d'optimisation qui trouve les optima globaux et locaux et* (iii) *fournir une meilleure compréhension du problème d'optimisation et de son espace de conception. La technique utilisée est de partitionner l'espace de conception entre divers agents utilisant des métamodèles différents pour approximer leur sous-région et se coordonnant pour modifier les frontières de leur sous-région.*

Mots-clés : *Optimisation par métamodèle, partionnement adaptatif, agents coopératifs*

Abstract

In this paper we explore the use of multi-agent methods to tackle complex optimization problems that require expensive simulations. We particularly focus on surrogate-based optimization for problems in which direct optimization is too expensive. The proposed strategy partitions the design space between several agents that use different surrogates to approximate their subregion landscape. Agents coordinate by exchanging points to compute their surrogate and by modifying the boundaries of their subregions. The rationale behind this technique is to assign adequate surrogate to each subregion so that (i) *optimization is cheaper,* (ii) *the overall optimization process is not only global in scope but also stabilizes on local optima and* (iii) *the final partitioning provides a better understanding of the optimization problem.*

Keywords: *Surrogate-based Optimization, Space Partitioning, Multi-agent Systems*

1 Introduction

En ingénierie, les problèmes d'optimisation complexes, comme la conception d'avion, requièrent de nombreuses simulations coûteuses en calcul, comme la résolution des équations de la dynamique des fluides autour de l'avion. Ces simulations consistent en des appels à des logiciels qui, à un choix des variables de conception (un "point" dans l'espace de ces variables), associent des valeurs de performance de l'avion (e.g. la masse de l'avion simulé, son rayon d'action, son temps d'atteinte d'altitude de croisière, ...). De nombreux chercheurs ont contribué au développement de méthodes d'optimisation adaptées à ces simulations coûteuses. Les idées qui sous-tendent ces méthodes sont principalement *(i)* l'utilisation de *métamodèles* (ou approximations, ou *surrogates*) qui remplacent certains appels aux simulateurs coûteux par des appels à des fonctions rapides apprises sur ces simulations, *(ii)* la décomposition du problème en des problèmes de plus petite taille, *(iii)* l'approximation du problème par des problèmes plus simples à résoudre et *(iv)* la parallélisation des calculs. En Informatique, les systèmes multi-agents (SMA) sont en mesure de résoudre des problèmes complexes en les décomposant en sous-tâches autonomes [19]. En termes d'optimisation, un système multi-agent pourrait résoudre les problèmes décomposés de telle sorte que les agents ne connaissent que des sous-problèmes, plus simples à résoudre. De ce constat est né le travail pluri-disciplinaire, impliquant des informaticiens et des mécaniciens, qui est présenté dans cet article.

Dans les études antérieures [22], qui ont eut lieu pendant le projet ANR ID4CS [1], nous avons dé-

1. http://www.irit.fr/ID4CS

crit une méthodologie basée sur les systèmes multi-agents pour la conception de systèmes complexes, avec notamment la prise en compte d'incertitudes. L'optimisation avec incertitudes engendre une famille de problèmes difficiles car probabilistes et particulièrement coûteux en calcul. Dans cette première approche, chaque agent construit un métamodèle différent sur lequel il réalise une optimisation globale sur tout l'espace de conception (c-à-d. sur tout l'espace défini par les variables dont on cherche les valeurs). Grâce à la coopération (échange de points de recherche choisis), le système multi-agent définit non seulement une métamodélisaton multiple mais aussi une méthode d'optimisation globale de haut niveau dont l'efficacité et la robustesse ont été évaluées [22].

Dans cet article, nous réalisons un pas supplémentaire vers la distribution et la localité, afin d'améliorer le potentiel de distribution de nos systèmes. L'espace de conception est maintenant divisé en sous-régions (sous-espaces ayant autant de dimensions que l'espace de conception) pour l'optimisation par métamodèle avec les agents. Dans chaque sous-région, un *agent-métamodèle* va construire un métamodèle et l'utiliser pour trouver un optimum local (et non sur tout l'espace de conception comme dans [22]). Les agents se coordonnent en échangeant des points pour ajuster les métamodèles et pour repartitionner l'espace de conception. L'idée consiste donc à attribuer un métamodèle à chaque sous-région de sorte que *(i)* chaque optimisation est moins coûteuse en calculs car elle porte sur une partie seulement de l'espace de recherche, *(ii)* le processus d'optimisation se stabilise sur des optima locaux et globaux et *(iii)* le partitionnement final fournit une meilleure compréhension du problème d'optimisation (par identification des optima et des meilleurs métamodèles locaux).

Cet article est structuré comme suit. La section 2 présente le contexte théorique de notre travail. Dans la section 3, nous fournissons un aperçu de notre méthode d'optimisation par agents-métamodèles. Ensuite, nous décrivons dans la section 4 les méthodes de partitionnement de l'espace de conception permettant de localiser les optima tout en maximisant la précision des métamodèles. Enfin, dans les sections 5 et 6 ces méthodes sont illustrées sur un problème d'optimisation à deux dimensions caractérisé par des régions admissibles déconnectées, ce qui est un cas typique de difficulté rencontrée en conception optimale. Notre approche et ses résultats sont discutés en section 7. Nous concluons et dressons quelques perpectives en section 8.

2 Métamodèles, optimisation et agents

Un métamodèle est une fonction mathématique *(i)* approximant les réponses du modèle étudié (par exemple, le modèle représentant la masse d'un avion en fonction de ses différentes dimensions), construite le plus souvent à partir d'un ensemble de taille restreinte d'entrées-sorties de ce modèle *(ii)* de coût de calcul négligeable et *(iii)* ayant pour objectif de prédire de nouvelles réponses dans une partie de l'espace des entrées (éventuellement tout cet espace) [12]. L'ensemble des points initiaux est appelé plan d'expérience (ou *design of experiments*, *DOE*). Des exemples connus de métamodèles sont les surfaces de réponse polynomiales, les splines, les réseaux de neurones ou le krigeage. La préoccupation au sujet du coût est due au fait que, dans de nombreux domaines, le modèle étudié est un simulateur coûteux, pouvant nécessiter plusieurs heures (voire jours) de calcul pour produire un résultat. Un plan d'expérience et/ou la production de nouveaux points peuvent également être le résultat de mesures physiques réelles sur des prototypes ou des objets physiques existants. La figure 1 présente des exemples de métamodèles pour une fonction à une dimension et un plan d'expérience réduit. On voit ici que, pour un modèle et un sous-espace donnés, différents métamodèles peuvent avoir différents degrés de précision. L'erreur d'un métamodèle peut être mesurée. Dans cet article, nous utiliserons, par exemple, la somme des carrés des erreurs de prédiction (cf. section 3).

L'utilisation de métamodèles pour remplacer des simulations et expérimentations coûteuses en optimisation a été largement étudiée [10, 14, 16, 20]. De nombreux travaux ont proposé des stratégies d'utilisation de multiples métamodèles (*multi-surrogate*) pour l'optimisation [23, 17, 21, 8]. Notre approche par "agents-métamodèles" (*surrogate-based agents*) est liée aux approches multi-métamodèles, mais elle fait un pas supplémentaire vers des algorithmes conçus pour la parallélisation : la référence à la notion d'agent souligne le fait que les agents peuvent changer de stratégie (i.e. ici, de métamodèle, de données utilisées et de régions de l'espace de recherche considérées) en cours de résolution et que leurs actions sont asynchrones.

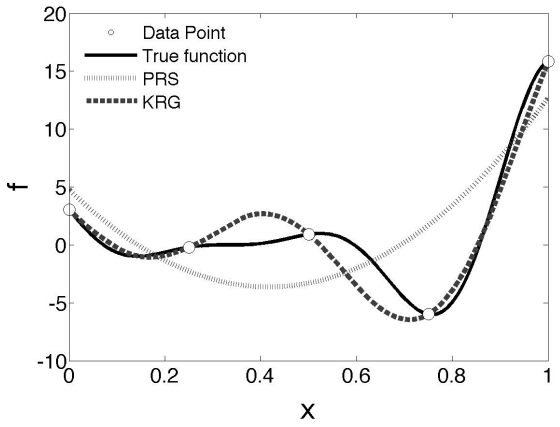


FIGURE 1 – Deux exemples de métamodèles pour une fonction à une dimension (ligne pleine) et pour un plan d'expérience de cinq points (cercles blancs) : surface de réponse quadratique (PRS) et krigeage (KRG).

Nous pouvons identifier principalement deux familles d'approches multi-agents pour l'optimisation :

Plusieurs agents – un problème global : dans ce cas les agents explorent le même espace de conception mais avec des stratégies ou des points de vue différents, comme dans les approches multi-métamodèles [22]. L'optimisation par essaim particulaire [11] et l'optimisation par colonie de fourmis [6] appartiennent également à cette famille, mais contrairement au travail présenté ici la métamodélisation n'est pas utilisée.

Plusieurs agents – différents sous-problèmes : dans ce cas le problème global est décomposé en plusieurs sous-problèmes affectés aux agents. Un problème d'optimisation peut classiquement être décomposé par dimensions –e.g. les variables comme dans [13, 15]– ou par critères –e.g. dans les approches basées sur la théorie des jeux pour l'optimisation multi-critère [19]. Nous pouvons aussi considérer une décomposition par partitionnement de telle sorte que chaque agent doit chercher dans un espace plus restreint que l'espace de conception complet. *Nous proposons une telle approche dans cet article en définissant les agents-métamodèles.*

Quelle que soit l'approche d'optimisation multi-agent, l'implémentation du SMA nécessite également de définir une politique de partage d'informations et de coordination (i.e. ici, échanger des solutions et points de données d'autres agents). De plus, des agents peuvent être détruits s'ils se trouvent inefficaces, et de même, des agents peuvent être créés pour venir en renfort. Ces créations et destruction peuvent être à l'initiatives des agents eux-mêmes, d'un agent dédié ou d'une entité oracle.

Dans notre travail actuel, nous nous intéressons donc plus particulièrement au *partionnement de l'espace de conception* (i.e., l'espace des entrées des métamodèles) pour définir des sous-régions pour des *agents-métamodèles* (des agents approximant des sous-régions par un métamodèle donné), comme le montre la figure 2. Trois motivations sous-tendent ce partitionnement. Tout d'abord chaque partition est d'une taille inférieure à l'espace de recherche complet, ce qui facilite la résolution locale du problème d'optimisation. Ensuite, la partition de l'espace avec métamodèle local permet de représenter des modèles dont le comportement varie dans l'espace (modèles non stationnaires au sens des processus aléatoires). A titre d'exemple imagé, on peut penser à un sous-sol dont les propriétés changent dans l'espace. Enfin, le partitionnement, en restreignant l'influence de l'optimum global à une sous-région, permet d'être plus sensibles aux optima locaux dans les autres sous-régions. Des méthodes existent pour ajuster des métamodèles locaux à des sous-régions de l'espace de conception partitionné en utilisant des méthodes de *clustering*. Par exemple, dans [25] une méthode est proposée pour diviser l'espace de conception en clusters en utilisant un mélange de Gaussiennes. Un point est affecté au cluster qui maximise sa probabilité d'appartenance. Des métamodèles locaux sont ensuite ajustés dans chaque cluster et un métamodèle global est construit en utilisant les métamodèles locaux et la loi de Bayes. Les travaux de [24] impliquent des surfaces de réponse, des modèles de krigeage et des points échantillonnés à partir de ces métamodèles pour réduire l'espace de conception aux seules sous-régions intéressantes. Les métamodèles permettent de générer des points non coûteux (car sans appel au vrai simulateur) qui sont ensuite regroupés par la technique des c-moyennes floues (*fuzzy c-means*). L'objectif de ce partitionnement est d'identifier les régions de bonne performance pour y concentrer les futures simulations précises.

3 Agents-métamodèles

Dans cette section nous présentons les principes de l'optimisation par métamodèles trouvés dans la littérature ainsi que l'extension à l'optimisation par des agents-métamodèles basée sur un partitionnement de l'espace de conception que nous proposons.

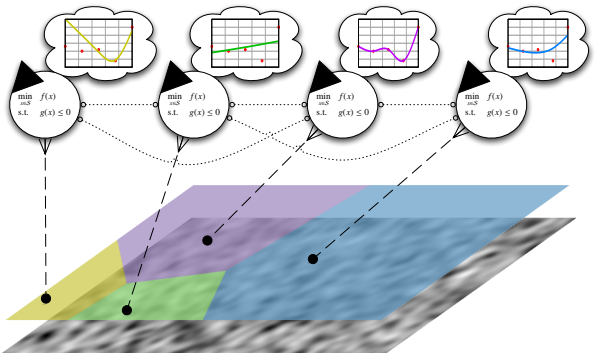


FIGURE 2 – Les agents-métamodèles optimisent une fonction commune sur une sous-région de l'espace de recherche suivant un métamodèle personnel, et communiquent des points à leurs voisins directs.

3.1 Optimisation par métamodèles

Considérons la formulation générique d'un problème d'optimisation sous contraintes présentée dans l'équation (1). f et g sont des fonctions et x un vecteur de variables.

$$\begin{aligned} &\underset{x\in\mathcal{S}}{\text{minimiser}} && f(x) \\ &\text{avec} && g(x) \leq 0 \end{aligned} \tag{1}$$

Par exemple, dans le cas de la conception d'un avion, x spécifie la géométrie et les composants de l'avion, g des performances minimales indispensables à réaliser telles qu'une altitude de croisière, une masse embarquée, un rayon d'action, et f pourrait être une quantité de carburant consommée. A un x donné, un ou plusieurs programmes de simulation coûteux numériquement doivent être appelés. En optimisation par métamodèle, un métamodèle est construit à partir d'un plan d'expérience, noté $\mathbb{X}$, qui consiste en un ensemble de points pour le vecteur de variables x. Pour le plan d'expérience, nous disposons des valeurs calculées de la fonction objectif f et des contraintes g pour ces points, notées $\mathbb{F}$ et $\mathbb{G}$, respectivement. Nous nous réfèrerons à $\mathbb{X}$ et ses valeurs associées $\mathbb{F}$ et $\mathbb{G}$ comme une base de données (ou de croyances, lorsque intégrée à un agent). Cette base est utilisée pour construire une approximation par métamodèle de la fonction objectif, notée $\hat{f}$, et $\hat{g}$ pour les contraintes. Nous pouvons approcher le problème de l'équation (1) en utilisant les métamodèles comme suit :

$$\begin{aligned} &\underset{x\in\mathcal{S}}{\text{minimiser}} && \hat{f}(x) \\ &\text{avec} && \hat{g}(x) \leq 0 \end{aligned} \tag{2}$$

Algorithme 1: Optimisation par métamodélisation

$t \leftarrow 1$ (état initial)
tant que $t \leq t^{max}$ **faire**
  Construire $\hat{f}$ et $\hat{g}$ à partir de $(\mathbb{X}^t, \mathbb{F}^t, \mathbb{G}^t)$
  `// Résoudre le problème (2) avec un algorithme d'optimisation interne:`
  $\hat{x}^* \leftarrow \underset{x\in\mathcal{S}}{\text{argmin}}\ \hat{f}(x)$, avec $\hat{g}(x) \leq 0$
  `// Appel au simulateur coûteux :`
  Calculer $f(\hat{x}^*)$ et $g(\hat{x}^*)$
  `// Mettre à jour la base :`
  $(\mathbb{X}^{t+1}, \mathbb{F}^{t+1}, \mathbb{G}^{t+1}) \leftarrow (\mathbb{X}^t, \mathbb{F}^t, \mathbb{G}^t) \cup (\hat{x}^*, f(\hat{x}^*), g(\hat{x}^*))$
  $t \leftarrow t + 1$

On note $\hat{x}^*$ la solution de ce problème approché.

L'optimisation par métamodèle classique s'approche itérativement de la solution de (1). Elle est par conséquent dépendante du nombre d'itérations t^{max}, qui correspond au budget d'appels à la fonction coûteuse. Une fois l'optimum, $\hat{x}^*$, du problème de l'équation (2) trouvé, les vraies valeurs de f et g associées sont calculées et insérées dans la base pour la prochaine itération. Ensuite, le métamodèle est mis à jour et l'optimisation (2) est recommencée. Ainsi, au temps t, nous notons $\mathbb{X}^t$ la base et l'ensemble des valeurs de fonction objectif et de contraintes associées $\mathbb{F}^t$ et $\mathbb{G}^t$, respectivement. La procédure d'optimisation par métamodèle peut ainsi être résumée par l'algorithme 1.

3.2 Agentification et coordination

L'algorithme 1 peut être pensé comme une procédure suivie par un seul agent-métamodèle lors d'une itération. Cependant, un agent i est restreint à une simple sous-région de l'espace de conception, $\mathcal{P}_i \subset \mathcal{S}$, de telle sorte qu'il ne peut considérer que les points dans sa propre sous-région disponibles dans sa base de croyances $(\mathbb{X}^t, \mathbb{F}^t, \mathbb{G}^t)_{internal}$. La sous-région d'un agent i au temps t est définie par la position de son centre. Un point de l'espace appartient à la sous-région dont le centre est le plus proche, suivant une distance euclidienne. Ceci a pour effet de diviser l'espace en cellules de Voronoi [2]. Le choix de la position du centre est le sujet de la prochaine section. Contrairement à [3], le partitionnement ne repose pas sur une discrétisation (en cellules) des dimensions de conception puis un regroupement des cellules en fonction de leur *fitness*, mais en un partage de l'espace de conception en sous-régions dépendant de la capacité pour le métamodèle de cette sous-région à trouver un optimum (et indirectement à être précis).

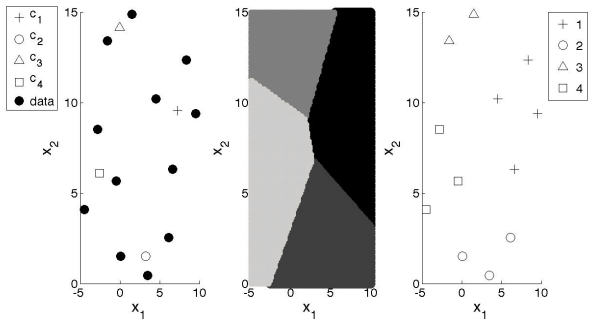


FIGURE 3 – Exemple de partionnement en deux dimensions montrant les données et les quatre centres (à gauche) correspondant à quatre agents et quatre sous-régions (au milieu), et l'affectation des données à chaque agent/sous-région (à droite).

La figure 3 illustre le partionnement d'une espace à deux dimensions en quatre sous-régions pour quatre agents, qui nécessite donc quatre centres. Dans cet exemple, nous plaçons les centres aléatoirement. Les sous-régions sont représentées par des couleurs différentes.

Une fois les sous-régions définies, chaque agent ajuste différents métamodèles proposés (cf. table 1) et choisit celui qui maximise sa précision dans la sous-région. Si plus de points sont nécessaires que ceux disponibles dans la sous-région pour régler le métamodèle, l'agent demande des points à ses voisins et les ajoute à sa base de croyances. En effet, chaque métamodèle nécessite un nombre minimum de points pour être ajusté de manière adéquate (voir les exemples de la table 1 en section 5). Les agents voisins communiquent également les informations associées à ces points. Nous définissons le meilleur métamodèle comme celui exhibant une erreur de validation croisée minimale, ici la somme des carrés des erreurs de prédiction $PRESS_{RMS}$. Elle est obtenue en enlevant un point de la base , en réajustant le métamodèle, et en mesurant l'erreur à ce point. Ceci est fait pour les p points de la sous-région de l'agent (sans tenir compte des points reçu des autres agents) afin de construire un vecteur de validation croisée e_{XV}. La valeur de $PRESS_{RMS}$ est ensuite calculée par :

$$PRESS_{RMS} = \sqrt{\frac{1}{p} e_{XV}^T e_{XV}} \qquad (3)$$

Une fois que les agents ont choisi leur métamodèle, une optimisation est effectuée, grâce à un optimiseur interne à chaque agent, pour résoudre le problème (2) à l'intérieur de la sous-région $\mathcal{P}_i$ uniquement. Notons que cette optimisation est effectuée en utilisant le métamodèle (une expression fonctionnelle), ce qui est typiquement rapide par rapport à une évaluation du vrai modèle. Si l'optimiseur fournit un point non admissible –i.e. qui ne satisfait pas les contraintes de (2) ou qui est en dehors de la sous-région– ou qu'il fournit un point déjà produit, l'agent *explore* sa sous-région. Pour explorer, l'agent ajoute à sa base un point, $\hat{x}^*$, de sa partition qui maximise la distance minimale aux points déjà présents dans sa base. Les vraies valeurs de f et g sont alors calculées, et $(\hat{x}^*, f(\hat{x}^*), g(\hat{x}^*))$ est ajouté à une base commune à tous les agents. Cette dernière est principalement utilisée pour repartitionner l'espace de conception [2]. L'itération suivante commence alors par repartionner l'espace de conception en utilisant cette base commune. La procédure représentant le comportement d'un agent-métamodèle est présentée dans l'algorithme 2.

Algorithme 2: Optimisation par un agent i dans sa sous-région

$t \leftarrow 1$ (état initial)
tant que $t \leq t^{max}$ **faire**
  Partitionner l'espace de conception (Alg. 3)
  Obtenir le centre c_i
  Mettre à jour la base de croyance à partir de la nouvelle partition
  si *pas assez de points dans la base de croyances* **alors**
    Récupérer les points d'agents voisins
  Construire $\hat{f}$ et $\hat{g}$ à partir de $(\mathbb{X}^t, \mathbb{F}^t, \mathbb{G}^t)_{internal}$
  Choisir le métamodèle qui minimise $PRESS_{RMS}$
  `// Résoudre le problème (2) avec un algorithme d'optimisation interne :`
  $\hat{x}^* \leftarrow \underset{x}{\text{argmin}}\ \hat{f}(x)$, avec $\hat{g}(x) \leq 0$
  si *pas de solution admissible OU point déjà trouvé* **alors**
    Trouver $\hat{x}^*$ dans $\mathcal{P}_i$ qui maximise la distance minimum aux points de la base de croyances
  `// Appel au simulateur coûteux :`
  Calculer $f(\hat{x}^*)$ et $g(\hat{x}^*)$
  `// Mettre à jour la base :`
  $(\mathbb{X}^{t+1}, \mathbb{F}^{t+1}, \mathbb{G}^{t+1}) \leftarrow (\mathbb{X}^t, \mathbb{F}^t, \mathbb{G}^t) \cup (\hat{x}^*, f(\hat{x}^*), g(\hat{x}^*))$
  $t \leftarrow t + 1$

2. Nous n'abordons pas la distribution de cette base dans cet article.

4 Partitionnement adaptatif et collaboratif de l'espace de conception

La méthode de partitionnement adaptatif de l'espace de conception que nous proposons se base sur le déplacement des centres des sous-régions vers des optima locaux. Ainsi, un agent peut choisir un métamodèle qui est précis autour d'un optimum local, et peut également explorer la sous-région autour de cet optimum local.

Cependant, pour la partition initiale, pour laquelle nous ne disposons d'aucune information sur les optima locaux, nous suivons une logique différente : la partition initiale est construite par un partitionnement de type k-moyennes [9]. L'algorithme des k-moyennes est une méthode de partitionnement de n données en k clusters de telle sorte que chaque point appartienne au cluster ayant la plus proche moyenne. Une fois que les points sont partitionnés lors de la première itération ($t = 1$), les agents sont affectés aux sous-régions et les points des sous-régions définissent les bases de croyances $(\mathbb{X}^t, \mathbb{F}^t, \mathbb{G}^t)_{internal}$ de chaque agent. Le processus suit ensuite l'algorithme 2 jusqu'à la fin de la première itération mis en œuvre par les différents agents.

À partir de la deuxième itération, les sous-régions sont définies par les agents de telle sorte que le centre de la sous-région soit le point de la base locale qui a la valeur minimale pour la fonction objectif f et qui satisfait la contrainte g. Si la contrainte n'est pas satisfaite, alors le centre est le point dont la contrainte est la plus faible (le point le moins infaisable).

Le positionnement du centre est déterminé en comparant l'optimum x^{*t-1} et le centre c^{t-1} de l'itération précédente. Le centre est déplacé à l'optimum de l'itération précédente s'il améliore la valeur de la fonction objectif si à la fois x^{*t-1} et c^{t-1} satisfont la contrainte g. Le centre est déplacé lorsque le dernier optimum satisfait la contrainte mais pas le centre précédent. Le centre est également déplacé s'il s'approche au mieux des contraintes alors que celles-ci ne sont respectées ni par le nouveau point, ni par le centre actuel. Enfin, si le centre est à une distance inférieure à un paramètre ε d'un autre centre (convergence de deux agents dans la même région), alors le centre est déplacé pour maximiser la distance minimale entre les centres. Ceci déplace donc l'agent vers une région non explorée précédemment. Ces différentes conditions sont résumées dans l'algorithme 3, après lequel, le processus de recherche continue avec l'algorithme 2.

Algorithme 3: Partionnement de l'espace de conception (calcul des c_i) autour de l'optimum local de la sous-région i

si $t = 1$ *(état initial)* **alors**
  k-moyennes pour trouver les centres c_i^t
sinon
  si $g(c_i^{t-1}) \leq 0$ **alors**
    // c_i^{t-1} respecte les contraintes
    si $g(\hat{x}_i^{*t-1}) \leq 0$ & $f(\hat{x}_i^{*t-1}) < f(c_i^{t-1})$ **alors**
      // $\hat{x}_i^{*t-1}$ devient centre :
      $c_i^t = \hat{x}_i^{*t-1}$
    sinon
      // c_i^t reste centre :
      $c_i^t = c_i^{t-1}$
  sinon
    // c_i^t ne respecte pas les contraintes
    si $g(\hat{x}_i^{*t-1}) < g(c_i^{t-1})$ **alors**
      // $\hat{x}_i^{*t-1}$ respecte mieux g :
      $c_i^t = \hat{x}_i^{*t-1}$
    sinon
      // c_i^t reste centre :
      $c_i^t = c_i^{t-1}$
si $\| c_i^t - c_{j\neq i}^t \| \leq \varepsilon$ *ou* $\| c_i^t - c_i^{t-1} \| \leq \varepsilon$ **alors**
  // Explorer :
  $c_i^t \leftarrow \arg\max_{c_i} min_{c \in \mathcal{P}_i}(\| c_i^t - c_{j\neq i}^t \|)$;

5 Exemple illustratif

Notre méthode d'optimisation multi-agent par partitionnement adaptatif de l'espace de conception est illustrée ici sur un problème à deux dimensions avec une contrainte [18]. La fonction objectif est quadratique et la contrainte est la fonction de test de Branin [5]. Ce problème est présenté dans l'équation (4).

$$\begin{aligned} \underset{x\in\Re^2}{\text{minimiser}} \quad & f(x) = -(x_1 - 10)^2 - (x_2 - 15)^2 \\ \text{avec} \quad & g(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right) + \cdots \\ & 10\left(1 - \frac{1}{8\pi}\right)cos(x_1) + 10 - 2 \leq 0 \\ & -5 \leq x_1 \leq 10 \\ & 0 \leq x_2 \leq 15 \end{aligned} \tag{4}$$

La figure 4 montre trois régions admissibles déconnectées qui sont définies par la contrainte g.

Ces régions admissibles couvrent environ 3% de l'espace de conception. L'optimum global est localisé en $x_1 = 3.2143$ et $x_2 = 0.9633$. Il y a aussi des optima locaux en $(9.2153, 1.1240)$ et $(-3.6685, 13.0299)$.

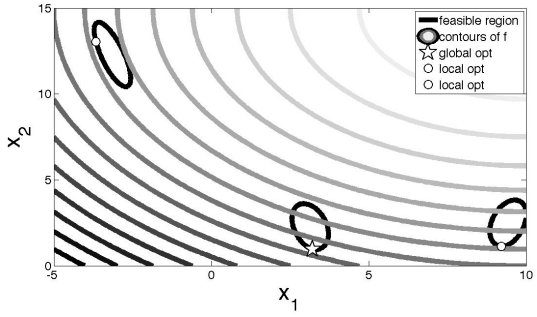


FIGURE 4 – Courbes de niveau de f pour le problème (4). Les frontières des régions admissibles sont représentées par des lignes noires pleines et l'optimum global est représenté par une étoile en $(3.214, 0.9633)$. Les optima locaux en $(9.2153, 1.1240)$ et $(-3.6685, 13.0299)$ sont représentés par des cercles blancs.

Dans cet exemple, seule la contrainte g est considérée comme étant la fonction coûteuse qui nécessite d'être approchée par métamodèlisation. Les métamodèles possibles sont trois surfaces de réponse, trois modèles de krigeage et les moindres carrés mobiles, comme décrit dans la table 1. Les agents peuvent choisir parmi ces sept métamodèles. La taille initiale du plan d'expérience (*DOE*) est 12, et le DOE été échantillonné par hypercube latin. Nous utilisons 4 agents (donc 4 sous-régions) pour résoudre le problème. Ce nombre sera discuté en section 7. Notons que, dans notre implémentation, l'algorithme d'optimisation utilisé par chaque agent pour résoudre (2) est un algorithme de programmation quadratique récurcive (en l'occurence le SQP de la fonction `fmincon` de Matlab[1]).

TABLE 1 – Métamodèles utilisés dans cette étude

ID	Description	# min. de points
1	Surface de réponse linéaire	
2	Surface de réponse quadratique	1.5 * # de cœfficients
3	Surface de réponse cubique	
4	Krigeage (tendance quadratique)	
5	Krigeage (tendance linaire)	# de param. +1
6	Krigeage (tendance constante)	
7	Moindres carrés mobiles (Shepard linéaire)	intégralité du *DOE*

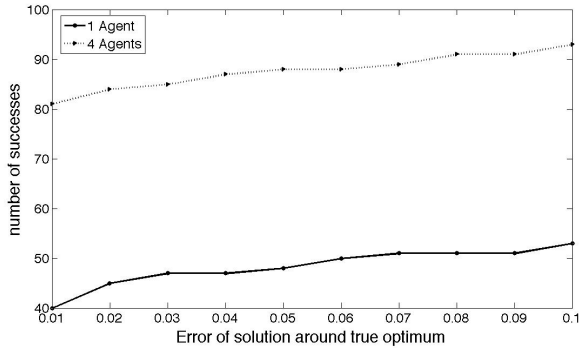
6 Résultats expérimentaux

Le processus décrit en sections 3 et 4 a été mis en œuvre pour $t^{max} = 20$ itérations avec quatre agents, ce qui représente environ 80 appels à la fonction coûteuse (g, ici). Ce processus a été répété pour 100 plans d'expérience différents. Les résultats ont été comparés à l'optimisation faite par un seul agent optimisant sur l'espace entier pour 100 itérations, ce qui représente 100 appels à la fonction coûteuse, et pour 100 expérimentations qui diffèrent par le DOE initial. L'usage d'un seul agent est équivalent à utiliser un seul métamodèle pour optimiser localement mais avec une exploration lorsque le métamodèle répète des points ou propose des points non admissibles. Ceci nous permet de comparer les deux processus avec un nombre équivalent d'appels à la fonction coûteuse.

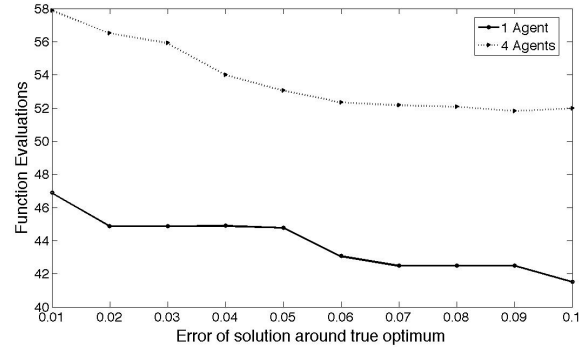
Détermination de l'optimum global. La figure 5(a) compare le nombre de succès sur 100 expérimentations, pour le SMA et pour l'agent seul, pour trouver une solution avec une certaine erreur autour de l'optimum global. Pour la plus faible erreur de 1% autour de l'optimum, le SMA a réussi 81 fois et l'agent seul 40 fois. Cette erreur de 1% est une erreur d'environ 0.03 pour x_1 et 0.01 pour x_2. Lorsque cette erreur est montée à 10%, soit environ 0.3 pour x_1 et 0.1 pour x_2, le SMA réussi 93 fois sur 100. Le SMA est donc plus performant pour trouver une solution comparé à l'agent seul, qui ne réussit que 53 fois pour une erreur de 10% autour de l'optimum.

Evaluation de la fonction coûteuse. La figure 5(b) compare le nombre d'appels à la fonction coûteuse (g) requis pour trouver une solution autour de l'optimum, dans les cas de succès. L'agent seul a demandé environ 10 évaluations de moins que le système à quatre agents. Cependant, comme montré en figure 5(a), l'agent seul n'a pas été aussi efficace que le SMA pour trouver la solution. Sur ces cas tests, le SMA est 20% plus coûteux en calcul qu'un agent seul, mais il permet de trouver l'optimum global dans plus de 80% des cas contre 40% pour l'agent seul.

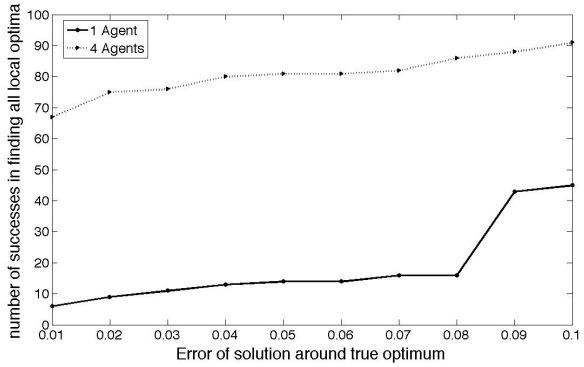
Localisation des optima locaux. Grâce au partitionnement et à la méthode de recherche coopérative, les agents sont capables de trouver les optima locaux. La figure 5(c) compare le nombre de succès moyen sur les 100 expérimentations pour trouver les trois optima locaux avec une certaine erreur pour chaque optimum. Le SMA est bien plus efficace pour trouver les trois optima que l'agent seul. À 1% d'erreur autour des optima, le système localise les trois optima dans 67% des cas, alors que l'agent seul ne les trouve que dans 6% des cas.



(a) Nombre de succès sur 100 pour trouver l'optimum global



(b) Nombre d'appels à la fonction coûteuse



(c) Nombre de succès sur 100 pour trouver tous les optima locaux

FIGURE 5 – Pour 100 expérimentations, (a) nombre de succès pour trouver l'optimum global, (b) nombre moyen d'appels à la fonction coûteuse pour des taux d'erreur variables, et (c) nombre de succès pour trouver les trois optima locaux.

Schéma de partitionnement. La figure 6(a) illustre le schéma de partitionnement décrit dans cet article sur un plan d'expérience initial aléatoire. La figure 6(b) montre les sous-régions des agents. Lors de la première itération, l'espace de conception est partitionné en utilisant l'algorithme des k-moyennes. Lors de cette itération, les agents 1 et 2 (blanc et noir, respectivement) sont incapables de localiser un optimum, d'où la représentation des optima du problème de maximisation de la distance minimale aux points existants dans la sous-région par des losanges. Les agents 3 et 4 (gris clair et gris foncé, respectivement) trouvent le même optimum, représenté par une étoile.

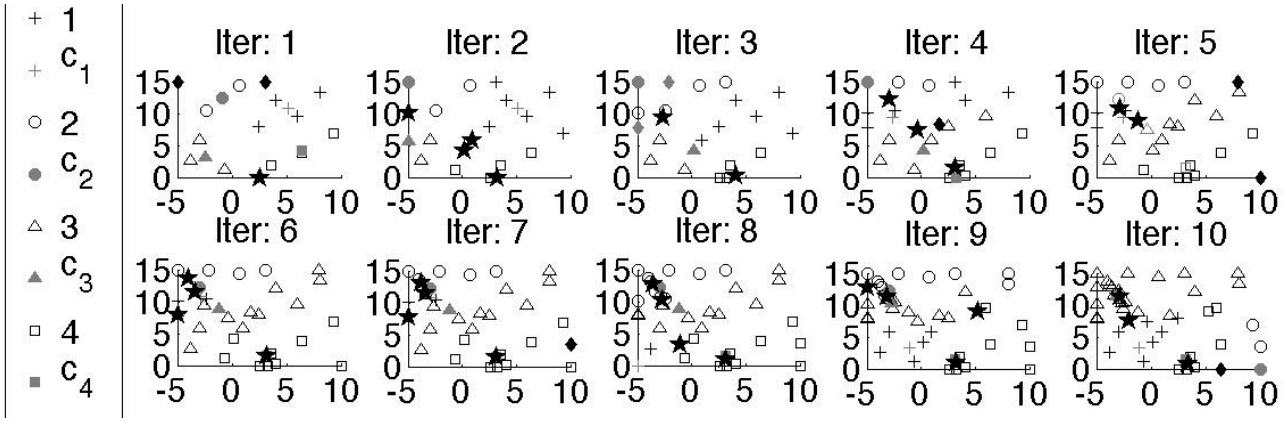
Au pas de temps 2, le dernier optimum de la première itération était une amélioration pour les agents 3 et 4. Ainsi, le centre de chacun de ces agents a été déplacé en cet optimum. Cependant, comme ces deux points étaient trop proches l'un de l'autre, l'agent 3 s'est déplacé au point maximisant la distance minimale aux autres centres. Lors du même pas de temps, l'agent 1 ne trouve aucune amélioration, donc son centre reste identique, alors que l'agent 2 trouve une amélioration donc son centre s'y déplace.

Tout au long des itérations, un regroupement de points autour de l'optimum global se forme dans la région de l'agent 1. Au pas 7, l'agent 4 fournit un point d'exploration car il avait trouvé deux points très proches en deux pas consécutifs. Les autres agents ont été attirés par la région admissible en $(-3.7, 13.0)$. Pour un certain temps, l'autre région admissible autour de $(9.2, 1.1)$ reste non explorée jusqu'au pas 10 où l'agent 2 s'y déplace avec sa sous-région. Cette expérimentation se conclut donc par la découverte de l'optimum global et des deux autres optima locaux.

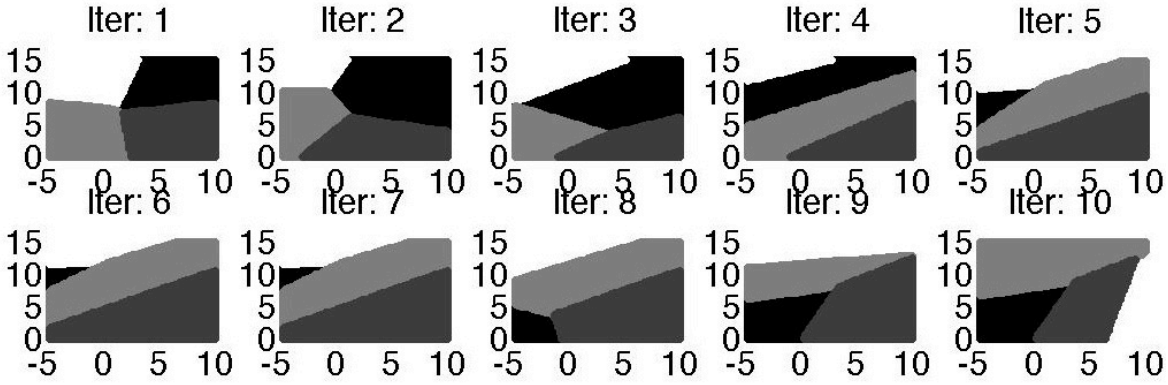
7 Discussion

Dans le modèle présenté dans cet article, nous n'avons pas discuté le nombre d'agents (fixé à 4 dans l'exemple). Ceci est dû à notre hypothèse de travail : nous considérons un budget de calcul fixe sur une grille de calcul, par exemple. Les agents étant déployés sur cette grille, nous supposons simplement qu'un nœud accueille un seul agent. Cependant, il est également possible de considérer que le nombre d'agents dépend du plan d'expérience. Il existe des méthodes découvrant le nombre de partitions en se basant sur la densité des nuages de points [7]. Ces techniques pourraient être utilisées lors des phases de repartitionnement : les zones à forte densité de points indiqueraient l'existence d'optima locaux et pourraient être associées à un agent chacune.

Un autre point important est l'usage d'une base commune pour décider d'un partitionnement, et la synchronisation qu'elle implique. Cette base est due à la phase de repartionnement qui nécessite l'ensemble des points calculés pour partitionner l'espace. Cette phase consiste en l'af-



(a) Affectation des points et positions des optima



(b) Sous-régions

FIGURE 6 – Dix itérations montrant (a) l'affectation des points de conception, les centres et les points ajoutés (les optima sont représentés par des étoiles, les points d'exploration par des losanges) et (b) les sous-régions des agents.

fectation des sous-régions aux agents et leur métamodèle. Ainsi, il pourrait être intéressant d'explorer les approches multi-agent d'allocation de ressources pour mettre en œuvre cette phase de manière distribuée [4]. De plus, le repartitionnement est un processus survenant à chaque pas de résolution. Nous pourrions considérer un repartitionnement plus flexible survenant à des moments plus appropriés au regard d'une certaine heuristique (e.g. partitionner lorsque les agents ont tendance à beaucoup explorer leur sous-région car ceci est une indication de la baisse de leur efficacité). Il faut garder à l'esprit que tout point calculé est très coûteux à obtenir et qu'il serait dommage d'en priver les autres agents, une fois obtenu. De plus, ces points sont assez peu nombreux, donc éventuellement partageables par tous les agents.

Enfin, comme de nombreuses méthodes d'optimisation par métamodèlisation, nous avons défini un temps maximum (t^{max}) comme seul critère d'arrêt. Ceci est lié à nos hypothèses de travail, ancrées dans la pratique, avec un budget limité en temps de calcul. Le temps limite est directement déduit de ce budget, en connaissant le coût numérique d'une simulation. Nous pourrions considérer un critère complémentaire basé sur la convergence du processus d'optimisation global.

8 Conclusions et perspectives

Le travail proposé dans cet article présente une méthode d'optimisation qui utilise plusieurs agents-métamodèles et un partitionnement adaptatif de l'espace de conception autour des optima locaux. Pour ce faire, les agents coopèrent en échangeant des points et en se réorganisant dans l'espace à chaque pas de résolution. Chaque agent combine optimisations locales et explorations au sein de sa sous-région. Les premiers résultats expérimentaux montrent que cette méthode permet de trouver avec une fiabilité accrue et un surcoût raisonnable l'optimum global. De plus, cette méthode localise des optima locaux. Nos études futures se focaliseront sur la décentralisation et l'asynchronisation du processus, de telle sorte que le partitionnement et la mise à jour de la base de points ne soient pas effectués à chaque itération.

D'autres perspectives incluent l'illustration de l'efficacité de cette méthode sur des problèmes à plus grandes dimensions. Nous sommes actuellement en train d'étudier un problème à 7 dimensions nécessitant un très grand nombre de simulations (de l'ordre du million avec les techniques classiques).

Ce travail est le fruit d'une collaboration pluridisciplinaire entre mécaniciens, informaticiens et mathématiciens. Il est intéressant de noter l'apport mutuel de cette collaboration. En effet, la perspective multi-agent a permis d'envisager les problèmes d'optimisation en Mécanique d'une manière originale et finalement efficace dans la recherche des optima et le nombre d'appels aux fonctions coûteuses. De même, le domaine de l'optimisation par métamodèle ouvre une voie vers de nouveaux modèles de représentations des connaissances dans le domaine de la résolution de problèmes par SMA.

Remerciements

Ces travaux ont été en partie financés par l'Agence Nationale de la Recherche (ANR) au travers du programme COSINUS (projet ID4CS ANR-09-COSI-005).

Références

[1] http://www.mathworks.fr/help/toolbox/optim/ug/brnoxzl.html#brnox01.

[2] F. Aurenhammer. Voronoi diagrams : a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3) :345–405, 1991.

[3] B. Calvez and G. Hutzler. Optimisation dichotomique adaptative : une nouvelle méthode pour le calibrage de modèles à base d'agents (présentation courte). In *16es journées francophones sur les systèmes multi-agents (JFSMA)*, pages 181–190, 2008.

[4] Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. Lemaitre, N. Maudet, J. Padget, S. Phelps, J. A. Rodriguez-Aguilar, and P. Sousa. Issues in multiagent resource allocation. *Informatica*, 30 :3–31, 2006.

[5] L. Dixon and G. Szego. *Towards Global Optimisation 2*, chapter The Global Optimisation Problem : An Introduction. North-Holland, New York, 1978.

[6] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, 2004.

[7] M. Ester, H.-S. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[8] B. Glaz, T. Goel, L. Liu, P. Friedmann, and R. T. Haftka. Multiple-surrogate approach to helicopter rotor blade vibration reduction. *AIAA Journal*, 47(1) :271–282, 2009.

[9] J. Hartigan and M. Wong. Algorithm as 136 : A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1) :100–108, 1979.

[10] R. Jin, X. Du, and W. Chen. The use of metamodeling techniques for optimization under uncertainty. *Structural and Multidisciplinary Optimization*, 25(2) :99–116, 2003.

[11] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ.*, pages 1942–1948, 1995.

[12] J. Kleijen. *Design and analysis of simulation experiments*. Springer, 2008.

[13] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. ADOPT : Asynchronous Distributed Constraint Optimization with Quality Guarantees. *Artificial Intelligence*, 161(2) :149–180, 2005.

[14] N. V. Queipo, R. T. Haftka, W. Shyy, and T. Goel. Surrogate-based analysis and optimization. *Progress in Aerospace Sciences*, 41 :1–28, 2005.

[15] D. Rajnarayan, D. Wolpert, and I. Kroo. Optimization under uncertainty using probability collectives. In *Proceedings of 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Portsmouth, VA, AIAA-2006-7033*, 2006.

[16] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4) :409–435, 1989.

[17] A. Samad, K. Kim, T. Goel, R. T. Haftka, and W. Shyy. Multiple surrogate modeling for axial compressor blade shape optimization. *Journal of Propulsion and Power*, 25(2) :302–310, 2008.

[18] M. Sasena, P. Papalambros, and P. Goovaerts. Global optimization of problems with disconnected feasible regions via surrogate modeling. In *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, GA, 2002.

[19] Y. Shoham and K. Leyton-Brown. *Multiagent Systems : Algorithmic, Game-Theoric, and Logical Foundations*. Cambridge University Press, 2009.

[20] T. W. Simpson, J. D. Peplinski, P. N. Koch, and J. K. Allen. Metamodels for computer based engineering design : survey and recommendations. *Engineering with Computers*, 17(2) :129–150, 2001.

[21] F. A. C. Viana and R. T. Haftka. Using multiple surrogates for metamodeling. In *7th ASMO-UK/ISSMO International Conference on Engineering Design Optimization*, Bath, UK, 2008.

[22] D. Villanueva, R. Le Riche, G. Picard, and R. T. Haftka. Surrogate-based agents : application to design under uncertainty. In *14th AIAA Non-Deterministic Approaches Conference*, Honolulu, HI, 2012 (forthcoming).

[23] I. Voutchkov and A. J. Keane. Multiobjective optimization using surrogates. In *7th International Conference on Adaptive Computing in Design and Manufacture*, pages 167–175, Bristol, UK, 2006.

[24] G. G. Wang and T. W. Simpson. Fuzzy clustering based hierarchical metamodeling for design space reduction and optimization. *Engineering Optimization*, 36(3) :313–335, 2004.

[25] D. Zhao and D. Xue. A multi-surrogate approximation method for metamodeling. *Engineering with Computers*, 27 :139–153, 2005.