THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National Polytechnique de Toulouse (INP Toulouse)

Discipline ou spécialité :

Réseaux, Télécommunications, Systèmes et Architecture

Présentée et soutenue par :

Mme NESRINE BADACHE

le vendredi 27 mai 2016

Titre :

ALLOCATION TEMPORELLE DE SYSTEMES AVIONIQUES MODULAIRES EMBARQUES

Ecole doctorale :

Mathématiques, Informatique, Télécommunications de Toulouse (MITT)

Unité de recherche :

Institut de Recherche en Informatique de Toulouse (I.R.I.T.)

Directeur(s) de Thèse :

M. CHRISTIAN FRABOUL

M. JEAN LUC SCHARBARG

Rapporteurs :

M. ALEXANDRE CAMINADA, UNIV TECHNO BELFORT MONTBELIAR

M. EMMANUEL GROLLEAU, ENSMA POITIERS

Membre(s) du jury :

M. EMMANUEL GROLLEAU, ENSMA POITIERS, Président

M. BENOIT VIAUD, ARTAL, Membre

M. CHRISTIAN FRABOUL, INP TOULOUSE, Membre

M. JEAN LUC SCHARBARG, INP TOULOUSE, Membre

Mme KATIA JAFFRES RUNSER, INP TOULOUSE, Membre

Remerciements

Je tiens à remercier en premier lieu mon directeur de thèse, Christian Fraboul, ainsi que mes encadrants, Jean-Luc Scharbarg et Katia Jaffres-Runser, sans qui je n'en serais certainement pas là. Merci pour leur patience, leur enthousiasme et leur confiance.

Je suis très reconnaissante envers Emmanuel Grolleau, Professeur à l'ISAE-ENSMA et membre du LIAS de Poitiers, et Alexandre Caminada, Professeur à l'Université de Technologie de Belfort-Montbéliard et membre du SET, d'avoir accepté d'être les rapporteurs de cette thèse. Je remercie également Benoît Viaud, Responsable du pôle Aero de Artal Technologies d'avoir accepté de faire partie du jury. Je leur transmets toute ma gratitude pour leurs commentaires constructifs et les perspectives ouvertes lors des discussions qui ont suivi la soutenance de la thèse.

Merci à Katia et Emmanuel de m'avoir écoutée, soutenue et "adoptée" pendant ses années de thèse. Merci à Yamine pour les discussions qui m'ont permise de voir les choses avec philosophie.

Je remercie l'ensemble du personnel de l'IRIT, avec une attention particulière pour Sylvie Eichen et Sylvie Armengaud-Metche qui ont toujours été disponibles pour m'aider à gérer les formalités administratives. Encore merci à toutes les deux pour leur bonne humeur et leur efficacité.

Une pensée pour tous les membres de l'équipe IRT et toutes les personnes avec qui j'ai partagé des cafés, des repas, des fous rires, des petites et grandes soirées et des moments précieux : Emmanuel, Andès-Luc, Gentian, Julien, Riadh, Béatrice; mes camarades et maintenant amis JB, Renaud, Clément, Aziz, Emilie, Cécile, Guigui, Farouk, Bouchra, Adnan, Xiaoting .

Je remercie les toulousains, la dream team de la bonne humeur, des sorties au lac et des barbecues au soleil: ChristophE, Yvan, Adrien, Emilie, Marj, Sandra, Pierre, Magalie, Ren, Thomas, Edd.

Aux "colocs", qui ont pris soin de moi pendant ses années de thèse, couplées à un boulot à temps plein. Vous avez su me rendre heureuse avec du Saint Nectaire, du pain maison, des repas de rois, des moments de toute beauté et un soutien sans faille. Je vous remercie infiniment.

Une pensée particulière à Barbara, ma bb, ma chtimi pour son amour et son amitié en or. J'espère t'avoir dans ma vie pour toujours. À mes amis de toujours, Sanaa, Katsu, Amel, Manel, Nassiba , Houda. Vous êtes les meilleures!

Je ne peux terminer sans remercier de tout coeur ma famille, en particulier mes parents pour leur soutien et leurs encouragements ainsi que mes frères, Sami, Réda et Djallil, qui m'ont poussée

et soutenue dans mes choix. à Thala, ma petite grande soeur, et à Loulou.

Et, enfin, je réserve mon ultime remerciement à toi, Charlie. Ce manuscrit ne serait pas ce qu'il est sans ton oeil avisé et ton sens critique. Mais par dessus tout, tu m'as accompagnée dans ces années de labeur et permis de m'y consacrer l'esprit libre. Je te dois tellement, merci d'avoir été et d'être toujours là.

Résumé

Dans la dernière décennie, les systèmes embarqués temps réel sont passés d'architectures centralisées à des architectures distribuées dites *modulaires*. Cette évolution a permis d'introduire plus de fonctionnalités grâce à l'utilisation de calculateurs (ou modules de traitement) répartis et à la définition d'interfaces de communication et de service standardisés. Nous nous intéressons dans cette thèse au cas de l'architecture avionique modulaire (Integrated Modular Avionics IMA), fondée sur les standards ARINC 653 et ARINC 651. Le passage aux architectures modulaires a introduit de nouveaux défis de conception, relatifs pour la plupart au respect des contraintes temporelles applicatives. Ces contraintes temporelles sont imposées pour garantir le bon fonctionnement des applications déployées sur ces systèmes modulaires. Elle se matérialisent, par exemple, par la définition de délais de communication maximum tolérés entre deux applications réparties sur des calculateurs distants. La conception d'un système modulaire est un problème d'intégration sous contraintes, qui regroupe plusieurs problèmes difficiles (dimensionnement, allocation de ressource spatiale et temporelle). La constante augmentation de la taille des systèmes modulaires, ainsi que de leur complexité, requiert la mise en place d'outils d'aide à l'intégration qui garantissent le passage à l'échelle. C'est dans ce cadre-là que ces travaux de thèse ont été menés.

Dans cette thèse, nous nous intéressons principalement à l'allocation *temporelle* des ressources du système. Étant donnée une répartition des fonctions sur les calculateurs, nous déterminons la périodicité d'exécution des fonctions embarquées distribuées, décrites comme des partitions communicantes, qui garantissent les contraintes temporelles applicatives et qui offrent un degré d'évolutivité du système élevé. Notre démarche prend en compte le délai de traversée variable (mais borné) du réseau embarqué de communication.

La première contribution de cette thèse est de reformuler le problème d'intégration d'un système modulaire IMA en un problème d'optimisation multicritère, sous contraintes temporelles. A partir d'une allocation des fonctions avioniques aux modules de calcul, ce problème d'intégration est décrit comme la recherche de la périodicité des partitions IMA de façon à garantir à la fois la fraîcheur des données transmises et la non-perte de la donnée dans les tampons de réception. Parmi toutes les allocations temporelles vérifiant les contraintes temporelles, nous réalisons une recherche multi-critères qui optimise conjointement un critère de charge des calcula-

teurs et de marge temporelle sur les communications. Ces deux critères ont été définis pour faciliter les évolutions futures de l'architecture.

La seconde contribution de cette thèse est la proposition de deux heuristiques de recherche multicritère adaptées à notre problème. La résolution de ce problème est difficile car le nombre d'allocations temporelles valides grandit exponentiellement avec le nombre de modules de calculs et de partitions hébergées par module. Ainsi, le grand nombre d'allocations temporelles admissibles ne permet pas d'en extraire les solutions les plus satisfaisantes sans l'aide d'un outil adapté. Dans ces travaux, nous proposons d'exploiter la nature des métriques définies pour proposer deux algorithmes d'optimisation multicritères : EXHAUST, un algorithme optimal de recherche exhaustive, et TABOU un algorithme approché basé sur une métaheuristique Tabou. Pour les deux algorithmes, la cardinalité du problème est réduite par une phase d'optimisation locale à chaque module, rendue possible par la nature des deux métriques choisies. Cette première étape d'optimisation locale permet de résoudre à l'optimal le problème d'allocation avec EXHAUST pour un système IMA de taille moyenne. Nous montrons que pour des systèmes de grande taille, l'algorithme TABOU est un très bon candidat car il extrait des solutions satisfaisantes en un temps raisonnable, tout en testant un nombre limité d'allocations valides (en comparaison avec EXHAUST qui teste toutes les allocations valides).

Ces deux heuristiques sont appliquées à un exemple simplifié de système IMA. L'analyse des solutions obtenues nous permet de mettre en exergue la qualité des solutions Pareto-optimales obtenues par les deux algorithmes. Ces solutions présentent clairement les caractéristiques recherchées d'évolutivité de la charge des calculateurs et de marge dans le réseau.

La dernière contribution de cette thèse réside dans une analyse fine des solutions Pareto-optimales. Cette analyse permet de mettre en avant différentes classes de solutions Pareto-optimales, chaque classe représentant un différent compromis entre la charge et la marge réseau. La connaissance de ces classes de solutions permet à l'intégrateur système de choisir une solution qui satisfasse au mieux ses attentes en fournissant le compromis qu'il recherche entre un critère de charge et un critère de marge réseau.

Abstract

The evolution of real-time embedded systems architectures to modular architectures has introduced more functionality through the use of distributed computers and communication interfaces and standardized service. We focus in this thesis on Integrated modular avionics architectures (IMA) standardized in ARINC 653 and ARINC 664 standard Part 7. This development has introduced new design challenges, among others, with respect to applying timing constraints mandatory for the proper functioning of such systems. The design of a modular system is an integration problem under constraints which features some difficult issues (design, spatial and temporal resource allocation). These difficulties advocate for the implementation of scalable integration and design tools. The design of such tools is the purpose of this thesis. We are interested primarily in the allocation of timing resources to the global distributed system. In particular, we determine the execution time of distributed embedded functions that guarantee the application timing constraints and offer a high degree of scalability of the system, given a distribution of avionics functions on the computers. Our approach takes into account the temporal upper-bounded variability of the communication network. The first contribution of this thesis is the formulation of the problem of integration of an IMA system in a multi-criteria optimization problem with timing constraints. For a distribution of avionics functions to computers, execution periods of IMA partitions are sought in order to ensure freshness and non-loss of transmitted data. Among all temporary allocations satisfying the timing constraints, we perform a multi-criteria search that optimizes both the load of calculators and temporal margin in the network. These two criteria facilitate a future extension of the proposed architecture. The second contribution of this thesis is the proposal of two multi-criteria search heuristics adapted to our problem. Note that the number of valid temporary allocations grows exponentially with the number of modules and partitions hosted on them. We offer two multi-criteria optimization algorithms: (i) EXHAUST, optimal exhaustive search algorithm, (ii) TABOO an approximation algorithm based on the Tabu metaheuristic. For both algorithms, the cardinality of the problem is reduced by a local optimization phase for each module, made possible by the linearity of the two selected optimization metrics. This first local optimization step helps to solve medium size problems to optimility using EXHAUST. We show that for larger systems, the TABOO algorithm is a very good candidate because it extracts satisfactory solutions in a reasonable time while testing a limited number of

valid allocations. These two heuristics are applied to an IMA system example. The analysis of the solutions obtained allows us to highlight the quality of Pareto-optimal solutions obtained by both algorithms. They have the sought characteristics of scalability and concurrent load minimization and network margin maximization. Our latest contribution lies in a detailed analysis of the trade-off solutions obtained using EXHAUST and TABOO. The analysis highlights different classes of Pareto-optimal solutions with different compromises between the load of the system and the network margin. The knowledge of these solutions allows the system integrator to choose a solution among solution classes that offer the compromise between the search criteria and network load margin.

Table des matières

Table des figures

Liste des tableaux

1 Introduction générale

Les systèmes embarqués intègrent de plus en plus des architectures dites modulaires. Ils sont composés d'un ensemble de calculateurs, interconnectés à travers des réseaux temps-réel. Cette architecture modulaire a permis le passage à l'échelle des systèmes embarqués. En effet, les anciennes architectures fédérées ne permettaient pas un ajout croissant de nouvelles fonctionnalités. Ils ne permettaient pas non plus d'interconnecter des équipements ou de faire tourner des logiciels issus de différents fabricants de façon transparente. L'architecture modulaire prône la mutualisation et la réutilisation de composants matériels et logiciels: les ressources de calcul sont partagées entre plusieurs fonctions embarquées, et les communications sont regroupées sur des bus multiplexés, par opposition aux architectures obtenues par interconnexion d'équipements dédiés par des liaisons mono émetteur. Cela réduit l'encombrement et le poids lié au matériel et au câblage.

Nous nous intéressons dans cette thèse au cas de l'architecture avionique modulaire (Integrated Modular Avionics IMA), fondée sur les standards ARINC 653 et ARINC 664 partie 7. Dans cette optique, l'industrie avionique a proposé L'Avionique Modulaire Intégrée, ou Integrated Modular Avionics (IMA) [4]. Cette dernière est fondée sur la standardisation de l'architecture matérielle et un partitionnement robuste (temporel et spatial) des applications qui partagent un calculateur. L'application est donc vue comme un ensemble de partitions logicielles communicantes. Ce type d'architecture est appliqué aussi bien dans le domaine civil, avec le Boeing B777 et l'Airbus A380 et A350, que dans le domaine militaire, avec le Rafale, l'A400M ou encore le Mirage 2000-9.

Les architectures modulaires embarquées représentent une évolution majeure des systèmes avioniques (ou automobiles), mais au prix d'une complexité accrue. Ces systèmes sont critiques, et leur validation temporelle est essentielle pour garantir la sécurité des usagers. Pour cela, il est impératif de développer des méthodes d'aide à la conception de tels systèmes qui assistent les ingénieurs de façon fiable dans les phases de développement.

Comme nous le verrons, la conception d'un système modulaire peut être vue comme un problème d'intégration sous contraintes qui regroupe plusieurs problèmes difficiles : le dimensionnement, l'allocation des ressources spatiale et temporelle. Une des difficultés majeures est la nature asynchrone de ces systèmes modulaires. Les calculateurs ne sont pas synchronisés dans leurs fonctionnement. Ainsi, on ne peut exploiter une horloge commune pour la validation tem-

porelle du système. Cet asynchronisme permet d'éviter la vulnérabilité du système à la perte de l'horloge. Par contre, il complexifie la validation du système qui doit être fiable pour tous les scénarios de lancement et de décalages possibles.

Actuellement, la conception de systèmes embarqués critiques se fonde sur l'expertise des spécialistes intégrateurs. Ces spécialistes disposent d'outils limités d'aide à la configuration des périodes d'exécution des applications embarquées pour l'intégration des systèmes complexes. Or, un système modulaire avionique comprend une centaine d'unités de calcul, qui échangent un millier de flux. Le partage des ressources de calcul et de communication complexifie le calcul des propriétés temporelles des fonctions supportées. Le partage de ressources oblige donc à considérer le système au complet dans le choix de la stratégie d'intégration adéquate. Le tout doit être réalisé en tenant compte des contraintes d'exécution des fonctions qui composent les applications, et des bornes maximales sur les délais de bout-en-bout[5]. Ces contraintes rendent la tolérance du système aux futures évolutions difficile à garantir : dans le cas de la modification ou de l'ajout d'une nouvelle fonctionnalité ou d'un nouveau matériel, les experts en intégration doivent effectuer de lourdes modifications sur la configuration existante. Ces reconfigurations sont nécessaires pour garantir la cohérence des données et le respect des contraintes temps réel d'exécution des fonctions et des échanges à travers le réseau.

Plusieurs travaux de recherche se sont penchés sur la problématique de dimensionnement et d'intégration d'une architecture temps réel. Ces travaux ont abordé le problème comme un problème d'allocation spatiale et d'ordonnancement distribué. Dans ces travaux, une allocation est choisie si celle-ci optimise un ou plusieurs critères de performance désirés par l'intégrateur (généralement, la charge des calculateurs est minimisée, entre autres). Ces solutions sont complexes et nous proposons dans cette thèse de ne pas formuler la conception comme un problème d'allocation spatiale, qui peut être guidée par des contraintes fortes de localisation des unités d'entrée/sortie, mais comme un problème d'allocation temporelle. Au lieu de rechercher un système final parmi tous les systèmes possibles, nous allons prendre une première solution qui ne remplit peut-être pas les contraintes temps-réel du systèmes. Nous allons ensuite chercher à modifier les caractéristiques temporelles de cette solution (ici la périodicité des fonctions) de façon à remplir les contraintes temps-réel et de proposer une solution proche de l'optimal.

Objectifs de l'étude

L'objectif de la présente thèse est l'aide à l'intégration des architectures temps réel avec la prise en compte des contraintes réseau dans une méthodologie d'allocation temporelle de bout-en-bout. Nous précisons que par *bout-en-bout*, nous nous intéressons à ce qui se passe entre le capteur, qui initie la fonction source d'une donnée vers l'actionneur, à la réception de la donnée. Les contraintes portent généralement sur des délais de réponse entre le stimulus et la réaction. Les architectures avioniques IMA sont le cas d'étude des travaux de cette thèse, où on considère que l'allocation spatiale des partitions sur les calculateurs a déjà été effectuée. Notre méthodologie

permet de raffiner l'intégration en choisissant les paramètres temporels qui garantissent le bon comportement le temps réel du système complet.

La méthodologie de dimensionnement proposée offre à l'intégrateur le choix dans le paramétrage des propriétés temporelles de manière à garantir les paramètres de fraîcheur (pour la cohérence des données) dans les communications inter-partitions: toute donnée reçue par sa partition destination distante après la durée de son paramètre de fraîcheur est une violation de la contrainte de fraîcheur et se voit écartée. L'objectif est de dimensionner le système de façon à éviter toute violation d'un des paramètres de fraîcheur.

En partant d'une allocation spatiale des partitions sur les calculateurs, nous montrons que l'impact des contraintes temporelles de bout-en-bout sur les communications est déterminant dans le calcul des périodicités d'exécution des partitions destinataires des communications et de la garantie de l'ordonançabilité au sein du module de calcul.

Nous formulons ce problème d'allocation temporelle, avec la prise en compte des contraintes temporelle de bout-en-bout, comme un problème d'optimisation sous contraintes. Dans ce problème d'optimisation, les variables sont les périodes des partitions destinations dans les communications entre partitions, contraintes parallèlement par des contraintes applicatives, relatives au système, et des contraintes temps réelles, relatives aux communications de bout-en-bout et à l'ordonnaçabilité dans les calculateurs.

Plan et contributions des travaux de thèse

Ce manuscrit est découpé en six chapitres qui abordent, dans un premier temps, le contexte scientifique de cette thèse, pour ensuite détailler les contributions proposées:

Chapitre 2 : Contexte industriel et scientifique. Nous présentons l'architecture modulaire avionique IMA. Après en avoir rappelé les principes fondamentaux, nous décrivons les différentes étapes de conception d'un tel système. Nous présentons la démarche standard d'allocation spatiale des fonctions embarquées et introduisons notre démarche d'allocation temporelle. La définition du délai et des contraintes de bout-en-bout nous permet par la suite de poser le problème de l'impact des choix d'allocation sur les performances temporelles. Enfin, les différents objectifs de cette thèse sont explicités. Par la suite, ce chapitre passe en revue les principaux travaux de l'état de l'art. Nous en dégageons les caractéristiques des solutions les mieux adaptées au problème d'intégration d'un système modulaire IMA. Puis, nous concluons quant à l'apport de l'étude présentée dans la suite du manuscrit.

Chapitre 3 : Formulation du problème d'allocation temporelle en IMA. Une première contribution de cette thèse est de formuler le problème d'intégration en un problème d'allocation temporelle. Ce problème est présenté comme un problème d'optimisation combinatoire où l'ensemble des allocations temporelles valides respectent les contraintes applicatives et temps-réel.

Ceci passe par la prise en compte des périodicités et les durées d'exécution des partitions embarquées, de la latence dans le réseau et de la fraîcheur des données. Nous proposons deux propriétés à respecter pour assurer un flux de transmission de données capable d'absorber les variations du réseau et qui respectent les contraintes (fraîcheur et non recouvrement des données). Un algorithme de séquencement pour la construction d'un ordonnancement valide des partitions sur une unités de calcul est proposé également.

Chapitre 4 : Métriques pour l'évaluation des performances d'une allocation temporelle. Un ensemble de métriques pour l'aide au choix d'allocations *évolutives* sont décrites dans ce chapitre. Ces critères constituent la seconde contribution de cette thèse. Ils ont été conçus dans le but d'augmenter le potentiel d'évolutivité d'une allocation. L'ajout futur de fonctions et de communications inter-partitions sera facilité de cette façon. Les métriques choisies concernent la charge des calculateurs, mais aussi la marge dans réseau. Un calculateur ayant une charge réduite admet l'ajout d'une nouvelle fonction. De même, si une solution d'allocation supporte l'ajout d'une communication sans introduire une violation d'un des paramètres de fraicheur, on dira que l'allocation est évolutive pour le réseau.

Chapitre 5 : Optimisation multiobjectif pour l'allocation temporelle. La troisième contribution s'intéresse à la conception d'algorithmes d'optimisation multicritère dédiés au problème d'allocation temporelle ainsi défini. Ce chapitre présente deux algorithmes d'optimisation multiobjectif capables de ressortir une ou plusieurs allocations temporelles Pareto-optimales pour un système IMA. En effet, il est possible qu'un même système IMA admette plusieurs allocations temporelles représentant différents compromis. Ces algorithmes sont basés sur la dominance au sens de Pareto. Le premier est un algorithme de recherche exhaustive multiobjectif. Cet algorithme calcule toutes les allocations temporelles Pareto-optimales pour une architecture IMA donnée. Le second se base sur l'heuristique tabou multiobjectif qui propose un nombre de solutions quasi-Pareto optimales. Le but est de proposer à l'intégrateur différentes allocations temporelles, chacune reflétant un compromis différent entre les métriques d'évaluation du système.

Chapitre 6 : Application des heuristiques proposées. Les deux heuristiques du Chapitre 5 sont appliquées à un système IMA de petite taille et de taille moyenne. L'analyse des solutions obtenues nous permet de mettre en exergue la qualité des solutions Pareto-optimales obtenues par les deux algorithmes. Nous montrons notamment que l'heuristique basée sur Tabou converge rapidement vers des solutions de compromis de qualité. Nous montrons également que les solutions fournies à l'intégrateur système permettent bien de trouver des solutions équilibrées entre charge des calculateurs et de marge dans le réseau.

Chapitre 7 : Conclusion et perspectives. Ce chapitre conclut les travaux de cette thèse et mets en avant les perceptives futures liées à ces travaux.

2 Contexte industriel et scientifique

Dans ce chapitre, nous présentons le contexte industriel et scientifique adressé dans cette thèse. Ceci implique les systèmes temps réel embarqués, plus particulièrement les systèmes temps réel modulaires à contraintes temporelles strictes. Nous mettons en avant les architectures embarquées modulaires implantées de nos jours dans les domaines avionique et automobile ainsi que les difficultés qui accompagnent cette évolution. Nous verrons alors que l'évolution des systèmes embarqués est telle qu'elle nécessite la mise en œuvre de nouvelles méthodologies d'allocation des ressources dans leur phase de déploiement.

Nous centrons notre travail sur les architectures embarquées modulaires du monde de l'avionique. Nous allons mettre en exergue les différentes étapes du processus d'intégration d'un tel système embarqué modulaire. Il s'en suit une présentation de la problématique de l'allocation temporelle illustrée dans le contexte avionique, pour enfin présenter le problème abordé dans cette thèse.

Les travaux de cette thèse se préoccupent de la problématique de l'allocation temporelle des fonctions et des communications sur les systèmes embarqués modulaires critiques. En effet, les systèmes temps réel modulaires doivent impérativement intégrer un ordonnancement des fonctions sur les calculateurs qui garantissent les contraintes temporelles applicatives. Comme ces fonctions communiquent par le réseau, il est aussi nécessaire de maitriser les délais de communication dans le réseau pour garantir un fonctionnement temps-réel strict des applications.

Le besoin de méthodes avancées dans les phases amont de conception, capables de proposer des allocations qui rassemblent ces exigences tout en optimisant un ensemble de critères se fait ressentir. Dans ce qui suit, un état de l'art sur les travaux connexes à la problématiques de cette thèse est donné. Des méthodes et approches proposées dans la littérature pour résoudre des problèmes d'intégration similaires sont mis en avant et sont analysés pour en préciser la problématique originale de nos travaux. Enfin, les méthodes utilisées pour la conception de la solution proposée dans ce manuscrit sont présentées.

2.1 Architecture modulaire des systèmes embarqués temps réel

Un système embarqué comporte un ensemble de calculateurs, ou modules de calculs, répartis géographiquement en différents points du système. Ces systèmes sont souvent contraints : en espace, en consommation énergétique, en délai de communication ou encore en poids. Une automobile, un avion ou un satellite sont des exemples type d'utilisation d'un système embarqué.

Pour les premières générations de systèmes embarqués, la conception d'un calculateur était réalisée par un seul constructeur ou équipementier. Il y intégrait uniquement ses propres fonctions embarquées qu'il fournissait par la suite sous la forme de boîte noire au client final qui intégrait le calculateur à son architecture. Le calculateur étant dédié aux uniques fonctions choisies par son constructeur, il n'était pas possible d'exploiter la mémoire et la puissance de calcul restantes pour servir d'autres fonctionnalités du système. Les ressources de calcul et de mémoire n'étaient pas mutualisables, et de ce fait sous-exploitées.

Les évolutions technologiques et l'augmentation des besoins clients ont fait croitre le nombre de calculateurs embarqués pour supporter un nombre croissant de fonctionnalités. L'augmentation du nombre de fonctions embarquées a eu pour conséquence directe un accroissement du nombre d'échanges des données. Pour cela, il a fallu mettre en place des réseaux sûrs, à débit de plus en plus élevé, interconnectant un nombre grandissant de calculateurs. En parallèle, beaucoup d'efforts ont été réalisés pour réduire le poids et la consommation énergétique des automobiles et des avions. La loi de Moore a permis d'augmenter la puissance des calculateurs, capables de servir un nombre croissant de fonctions à encombrement et consommation énergétique constants. Ces évolutions antagonistes ont introduit de la complexité et de réels défis dans les différentes phases de conception et de développement d'un système embarqué, comme nous le verrons.

La convergence de ces contraintes et défis a débouché sur la nécessité de repenser les architectures des systèmes embarqués critiques il y a une dizaine d'année. Cette nécessité a encouragé le passage vers une *architecture modulaire distribuée* qui prône le partage de ressources au sein de mêmes unités de calcul et le multiplexage des communications. Fortement inspirée des systèmes informatiques et de l'utilisation des API, l'architecture modulaire fournit une plateforme physique cible, modulaire et générique pouvant exécuter des fonctions embarquées développées par différents fournisseurs sur un même calculateur. Les principales caractéristiques de ces architectures modulaires sont présentées en détails dans la section suivante.

Les architectures modulaires temps réel sont soumises à des garanties temporelles strictes de par leur utilisation pour des applications critiques. Comme elles sont embarquées, elles sont aussi soumises aux contraintes matérielles de poids du matériel et du câblage. Ces architectures modulaires prônent la mutualisation des unités de calculs et des ressources réseau. Pour cela, elles se basent sur une standardisation qui permet à la fois la réutilisation des composants (modules de calcul) et l'utilisation d'interfaces de communication virtuelles pour une exploitation transparente des moyens de communication embarqués. Ainsi, il est possible de concevoir des fonctions

logicielles sans avoir à adapter le code au type de réseau utilisé. C'est dans la phase d'intégration du système complet que le lien sera établi entre l'interface de communication virtuelle utilisée par la fonction et l'interface physique du réseau réellement déployé.

Le concept des architectures modulaires repose sur les principes suivants [6] :

- **Modules de calcul génériques et partagés :** ce concept vise à accroître la possibilité de réutilisation des modules de calcul par différents acteurs et à s'éloigner des modules complètement dédiés et fermés.
- **Communications et réseaux multiplexés :** les communications sont transmises sur des bus partagés où un ensemble de flux sont multiplexés pour supporter le nombre accru d'échanges entre les fonctions.
- **Systèmes multicouches ouverts :** ce concept est inspiré des systèmes informatiques intégrant des middlewares (intergiciels) et des empilements protocolaires, similaires au modèle OSI de l'ISO. Il s'agit d'un découpage multicouche composé d'une couche exécutive standard de services et de communication, qui interface la couche applicative. Cette couche applicative contient les fonctions embarquées du système.
- **Architecture logicielle standardisée :** proposer un standard utilisable par l'ensemble des acteurs pour encourager l'interopérabilité et l'interchangeabilité des applications embarquées.
- **Architecture partitionnée :** pour assurer la fiabilité et le déterminisme des systèmes, l'environnement d'exécution d'une fonction ou d'un composant logiciel est partitionné en mémoire et en temps. A chaque fonction correspond une fenêtre temporelle fixée et périodique où le CPU lui est dédié, ainsi qu'une zone dédiée de la mémoire.

  Par exemple, dans le domaine de l'avionique modulaire, on parle de :

  * **Partitionnement spatial.** La mémoire est découpée et allouée aux différentes fonctions hébergées sur le même module. L'objectif est de prévenir les conflits d'accès à la mémoire. Ainsi, un composant logicielle ne peut modifier le code ou les données d'un autre composant logiciel.
  * **Partitionnement temporel.** Les fonctions embarquées d'un modules sont partitionnées. Chaque ensemble de fonctions se voit assigné une fenêtre d'exécution temporelle, périodique, dénommée *partition*. Les partitions d'un module sont ordonnancées statiquement sur le module de calcul comme nous le verrons dans la suite.

  Ces mécanismes de partitionnement sont très importants pour la sûreté du système. Cette ségrégation temporelle et spatiale isole les composants logiciels produits par les différents fournisseurs et leur fournit un environnement d'exécution exclusif. Ils offrent aussi la possibilité de faire coexister des applications critiques et non critiques sur un même module. Un avantage majeur du mécanisme de partitionnement est la réduction considérable de l'effort de certification pour les systèmes de criticité mixte [7] et la possibilité de réutiliser des composants.

Les deux principaux exemples actuels d'architecture modulaire temps-réels sont :

1. L'architecture modulaire intégrée (Integrated Modular Avionics), ou IMA [4], qui a été définie pour le domaine de l'avionique.
2. L'architecture AUTORSAR [8], qui a été définie pour le domaine automobile.

Cette thèse se concentre sur l'étude de l'architecture modulaire IMA. C'est une architecture temps réel, distribuée, asynchrone et modulaire, supportant des tâches mixtes (soit périodiques, soit apériodiques, etc.) et des réseaux de communication hétérogènes.

2.1.1 Architecture modulaire avionique : IMA

Les architectures modulaires intégrées (Integrated Modular Avionics ou IMA) [4] sont apparues dans les années quatre-vingt-dix pour remplacer les architectures fédérées, poussées par les considérations économiques dues à l'augmentation des coûts de production et la réduction nécessaire du poids du matériel et du câblage embarqués. L'idée derrière l'IMA est de proposer une architecture distribuée et flexible. Le concept de l'IMA est basé sur l'intégration multifonctions, c.à.d l'allocation de plusieurs fonctions de différents niveaux de criticité sur la même plateforme standardisée et communiquant à travers une couche d'interfaces standardisées.

L'architecture IMA est basée sur les normes ARINC 653 [9] et ARINC 651 [4], qui proposent une définition claire d'une répartition temporelle et spatiale robuste, pour la séparation des fonctions embarquées. ARINC 653 décrit l'usage général d'une interface applicative, dénommée APplication EXecutive (APEX). L'APEX permet le développement de la couche applicative indépendamment de son infrastructure physique. La couche APEX s'appuie sur l'isolation fonctionnelle entre les applications du système d'exploitation et les logiciels pour limiter la propagation des défaillances, et simplifier la validation du logiciel. L'APEX offre également un ensemble de services et assure l'abstraction des communications. Ceci, dans un souci de virtualisation de la couche d'implémentation physique. L'isolation et le partitionnement interviennent sur deux niveaux [10]:

Partitionnement spatial dans ARINC 653 Les unités de gestion de mémoire dans ARINC 653, dénommés MMU, s'occupent de l'allocation statique de la mémoire pour les partitions dans les phases de conception. Cette allocation reste inchangée durant toute la durée de vie de l'application.

Partitionnement temporel dans ARINC 653 A chaque partition est assignée une fenêtre d'exécution périodique. La durée de cette fenêtre est le calcul nécessaire à l'exécution des fonctions hébergées par la partition du module.

Compte tenu de la complexité de l'IMA, nous avons choisi de ne présenter que deux de ses niveaux de modélisation. Ces deux niveaux sont suffisants pour bien comprendre les études de cette thèse. Ces deux niveaux de modélisation sont:

Le niveau fonctionnel : ce niveau décrit les différentes fonctions du système et les échanges de données à travers les ports et les canaux de communication logiques (virtuels). Cette description est réalisée indépendamment de l'implantation matérielle des calculateurs et des réseaux de communication. Ce niveau ne représente que les fonctionnalités et les échanges de données entre les fonctions communicantes.

Le niveau système : ce niveau est orienté plateforme. Il décri en détail le matériel qui constituera la plateforme cible sur laquelle sont déployées les fonctions communicantes identifiées au niveau fonctionnel. La répartition des fonctions sur les différents modules de calcul y est décrite ainsi que l'allocation des canaux de communication logiques sur le réseau physique déployé. Il est possible à ce niveau de calculer la charge des modules de calcul et du réseau de communication.

Dans ce qui suit, nous nous basons sur ces deux niveaux de modélisation pour décrire plus en avant l'architecture IMA du standard ARINC 653 et les principes de l'APEX.

2.1.1.1 Niveau fonctionnel et communications logiques IMA-ARINC 653

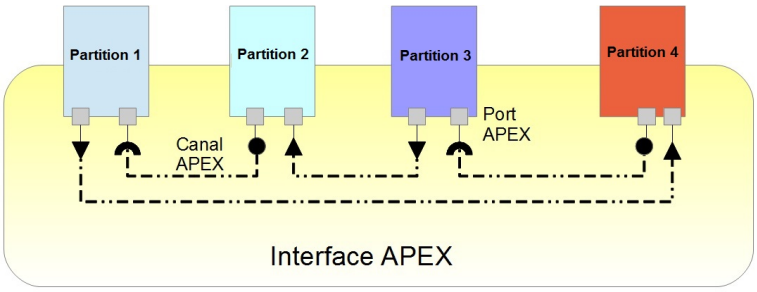


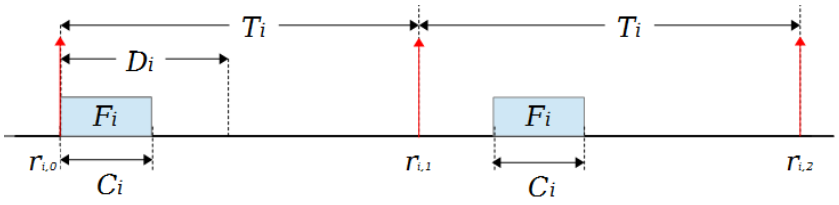
FIGURE 2.1 – Niveau fonctionnel et communications logiques IMA-ARINC 653

Ce niveau d'abstraction est une vue sur les applications ou fonctions embarquées. Ces fonctions sont réparties ici dans des ensembles, et chaque ensemble est regroupé dans une *partition*. Ce niveau fournit aussi les abstractions des communications entre les partitions représentées par des canaux logiques dits *canaux APEX*. Dans ce niveau, le système complet est représenté par des partitions communicantes comme illustré dans la Figure 2.1. L'APEX est représenté comme un intergiciel qui fournit des services aux partitions qu'il héberge. Il y a notamment des services de communication représentés par les canaux et les ports APEX. Les principaux éléments composant le niveau fonctionnel sont les suivants :

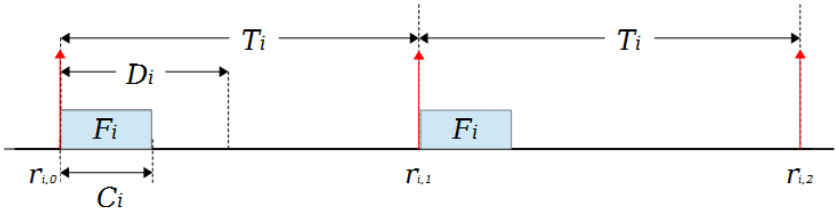
Les partitions : afin d'assurer la ségrégation spatiale et temporelle, l'ARINC 653 regroupe un ensemble de fonctions pour les exécuter dans un même environnement dédié nommé *partition*. Une partition est équivalente à un programme dans un environnement applicatif unique [11]. Une

partition possède sa propre zone mémoire. Aucune fonction différente de celles hébergées par la partition n'a accès à cette zone mémoire. Ceci assure la ségrégation spatiale de la mémoire du système.

Les tâches périodiques: Les fonctions ou tâches qui composent une partition dans un environnement temps réel sont soumises à des contraintes temporelles et fonctionnelles. Différentes données peuvent être utilisées pour caractériser une tâche temps réel. Une tâche peut s'activer périodiquement, ou de manière irrégulière en fonction de l'arrivée d'un événement. Dans cette thèse, nous nous focalisons sur les tâches dites *périodiques* qui s'exécutent de manière régulière et se répètent indéfiniment durant la vie du système. Chaque nouvelle exécution d'une tâche f_i est appelée *occurrence* ou *instance*. Le modèle d'une tâche périodique est représenté par la figure 2.2(a).



(a) Tâche périodique



(b) Tâche strictement périodique

FIGURE 2.2 – Tâches périodiques et strictement périodiques (Modèle de Liu et Layland [1])

Nous caractérisons une tâche périodique f_i par les données suivantes :

- C_i est la durée d'exécution pire-cas (ou la charge maximale) de f_i. Elle est référée par le terme WCET (*Worst Case Execution Time*). La durée d'exécution pire cas représente la borne supérieure sur la durée d'exécution de f_i sur l'unité de calcul qui l'héberge. La déterminer est un problème complexe [12]. Nous la supposerons connue.
- T_i est la période de f_i. T_i représente la durée qui sépare l'arrivée de deux demandes d'exécution consécutives de f_i.
- D_i est le délai critique de la tâche f_i. D_i représente la durée maximum entre le réveil de la tâche et la fin d'exécution de l'occurrence correspondante. Un dépassement de ce délai peut engendrer des conséquences désastreuses.

– r_i représente l'instant de la première activation de f_i sur l'unité de calcul qui l'héberge. Elle est aussi appelée *date de réveil* ou *offset*.

Dans le cas d'une tâche *périodique* standard, une occurrence $k \in \mathbb{N}$ de f_i est activée chaque $r_i+(k\text{-}1)T_i$ unités de temps. L'ordonnanceur doit ici terminer l'exécution de chaque occurrence avant que D_i unités de temps ne se soient écoulées. Il peut choisir n'importe quelle date de démarrage d'une occurrence du moment que la tâche se termine avant son échéance.

Dans un système IMA, les partitions sont *strictement périodiques*. Une tâche *strictement périodique* doit respecter une distance stricte de T_i entre l'instant de démarrage de toute instances consécutives k et $k+1$. Le modèle d'une tâche strictement périodique est représenté par la Figure 2.2(b).

Canaux et ports logiques APEX : Les partitions interagissent entre elles afin de réaliser des traitements pour assurer le bon fonctionnement du système. Pour interagir, les partitions échangent des données. Certaines partitions produisent des données et d'autres les consomment.

Dans les systèmes IMA, l'accès aux communications se fait par des *canaux logiques APEX* reliés par des *ports logiques APEX*. A chaque communication est associée la donnée dont la valeur sera échangée périodiquement. Les ports APEX sont unidirectionnels : ils sont utilisés soit en lecture, soit en écriture. Les canaux logiques et ports APEX sont représentés sur la Figure 2.1. Les messages sont sauvegardés dans des files d'attente en émission et en réception. Ces files d'attentes peuvent avoir deux modes opératoires représentatifs des deux modes de communication. Leur fonctionnement est illustré dans la Figure 2.3 et présenté ici :

Mode de communication *sampling* : Dans le mode *sampling*, la file d'attente en émission et en réception ne peut enregistrer qu'une seule donnée. Ainsi, la dernière valeur (ou occurrence) de la donnée écrite dans le port est la seule mémorisée par les files d'attente en émission ou en réception. Dans ce cas, l'arrivée d'une nouvelle valeur de la donnée écrase l'occurrence précédente dans le buffer. Une valeur est associée à une estampille temporelle qui donne la date de mise à jour de la donnée dans le port. Dans ce mode, la donnée peut être lue plusieurs fois même si sa valeur n'a pas été mise à jour. L'estampille temporelle permet de connaitre la *fraicheur* de la valeur et d'en déduire si celle-ci est trop agée ou non.

Mode de communication queueing : Le mode *queuing* gère une file d'attente de taille $N > 1$ en mode FIFO (*First In First Out*). Cette file d'attente sauvegarde N mises à jour d'une donnée. Le mode *queueing* permet de connaitre la dernière valeur disponible d'une donnée, tout en ayant un historique des $N-1$ mises à jour précédentes. La taille N est définie à la configuration.

Nous ferons l'hypothèse dans la suite que les données destinées à une partition et présentes dans un port APEX sont lues au démarrage de la partition. Ces données sont par la suite utilisées par les traitements de la partition. A la fin de l'exécution de la partition, les données sont émises à travers les ports d'émission de la partition, réalisant ainsi le schéma *Lecture-Traitement-Écriture*. En d'autres termes, il n'y a ni lecture ni écriture sur les ports APEX pendant le traitement

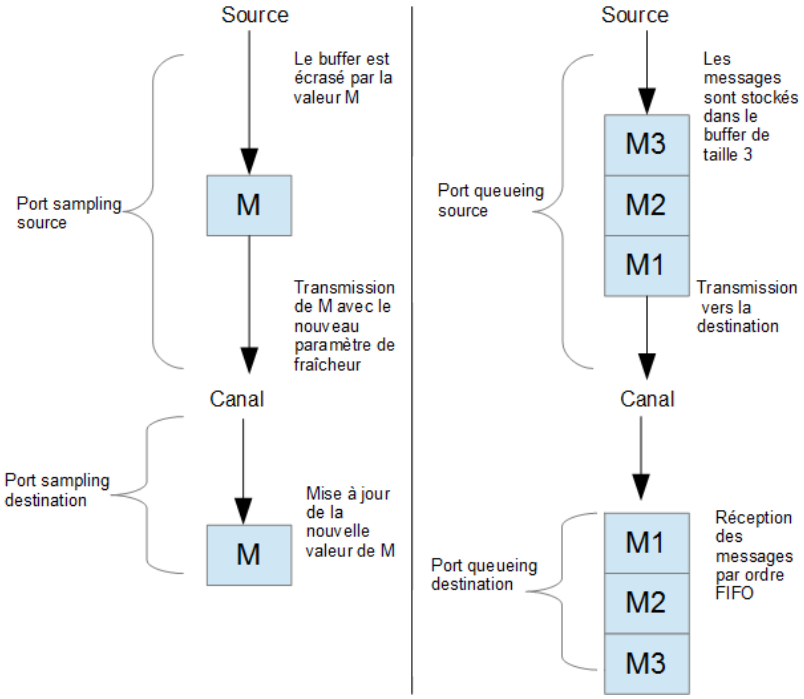


FIGURE 2.3 – Modes sampling et queueing

de la tâche. Les entrées et sorties sur le réseaux sont réalisées avant et après le traitement, respectivement.

2.1.1.2 Niveau système IMA-ARINC 653

Une fois le modèle fonctionnel APEX défini pour une application avionique donnée, il faut procéder à son déploiement sur l'architecture physique cible. Comme nous le verrons par la suite, ce déploiement comporte plusieurs problèmes difficiles à résoudre et qui constituent le cœur de cette thèse. Une fois le déploiement réalisé, on obtient le niveau de modélisation du système complet tel que schématisé sur le Figure 2.4. Ce niveau comporte les éléments suivants :

Module IMA - Core Processing Module (CPM) : le CPM, ou module IMA, est la plateforme physique responsable de l'exécution des fonctions embarquées. C'est l'unité de calcul qui héberge et s'occupe de l'exécution des partitions et leur gestion par le système d'exploitation et la couche d'interface APEX. Un CPM héberge un ensemble de partitions. Les partitions s'exécutent sur le module IMA selon une fenêtre périodique et sont ordonnancées selon une trame cyclique appelée MAjor time Frame (MAF). La longueur d'une MAF est *l'hyperpériode* [13] des partitions données par le calcul du plus petit commun multiple (*ppcm*) des périodes des partitions allouées sur le même module IMA. Une partition est préemptée lorsque le temps

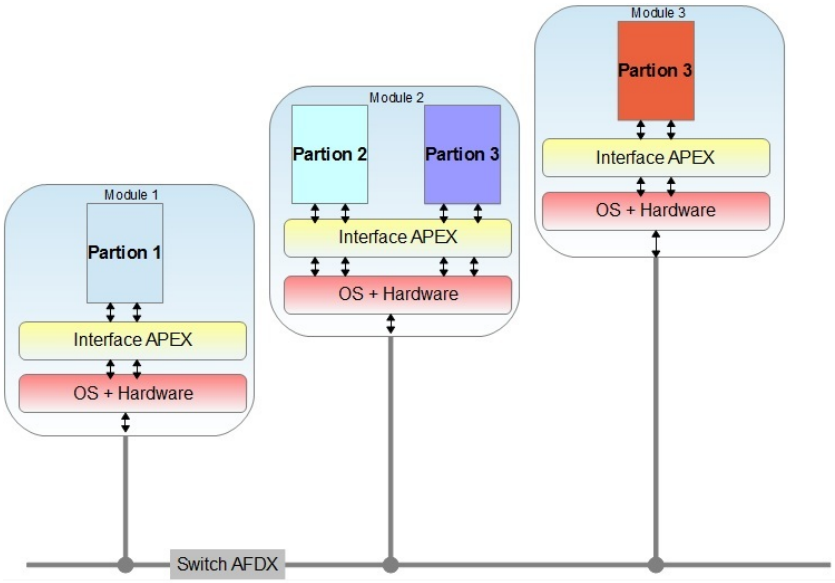


FIGURE 2.4 – Niveau système IMA-ARINC 653

d'exécution qui lui est réservé est terminé. Au démarrage d'un module les partitions qui y sont hébergées sont activées en même temps. Chaque partition s'exécute selon un déphasage statique par rapport au démarrage du module et répète son activation cycliquement selon sa période d'exécution. La Figure 2.5 représente l'ordonnancement de deux partition P_1 (de période égale à T_1 et de durée d'exécution b_1) et P_2 (de période T_2 et d'une durée d'exécution b_2) sur le module IMA M_1.

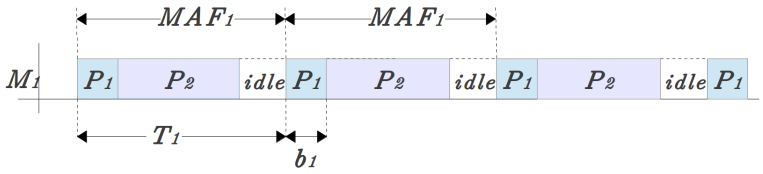


FIGURE 2.5 – Illustration d'une MAF

Les communications entre partitions IMA sont réalisées de deux façons différentes :

Communication intra-module : ce sont les communications réalisées au sein d'un même module de calcul pour l'échange de données et la synchronisation du traitement des tâches. Ces communications se font par mémoire partagée, blackboard, sémaphores et événement via l'exécutif temps réel du module. Elles peuvent aussi utiliser une interface réseau de type loopback dans certains cas.

Communication inter-module : pour communiquer avec un module distant, une ou plusieurs

piles de communication sont implantées dans chaque module. Ces piles permettent de gérer différents protocoles réseaux (AFDX, CAN, etc.).

Dans cette thèse, nous focalisons nos recherches sur les communications inter-module car elles introduisent un délai variable (mais borné) sur les échanges entre les partitions.

Réseaux de communication : A chaque donnée échangée, le modèle fonctionnel associe un canal APEX et un mode de communication. Pour le modèle système, cette communication est associée à un flux réseau réel, fourni par un des protocoles présents dans l'architecture cible. Plusieurs réseaux peuvent être utilisés tels que l'AFDX, ARINC429, CAN, etc.

Nous nous concentrons ici sur l'utilisation du réseau AFDX du standard ARINC 664 [14], partie 7 qui a été défini en même temps que l'IMA, qui apporte de meilleures performances dans la manipulation du nombre croissant des données. L'AFDX est basé sur la technologie Ethernet commuté et permet la communication entre modules IMA asynchrones. L'augmentation de la quantité des données échangées entre composants embarqués a poussé les réseaux de communication et leurs fonctionnements à évoluer dans le domaine avionique. On est passé de liens de communication point-à-point au concept de bus partagé mono-emetteur (ARINC 429), puis du concept de bus partagé mono-émetteur au réseau commuté.

L'AFDX est un réseau de données de type Ethernet commuté. Il satisfait les contraintes de criticité des fonctions temps réel tout en garantissant la qualité de service requise. Le reseau AFDX se caractérise par :

- des communications en full duplex,
- routage statique: l'acheminement des paquets dans un flux est réalisé selon un chemin unique de bout-en-bout. Le routage est statique et se base sur des tables pré-chargées dans les commutateurs,
- tout le réseau est redondé afin de garantir le service même en cas de panne d'un les deux réseaux,
- un débit élevé (100Mb/s),
- une latence et des gigues bornées dans la phase de conception pour garantir la non-perte des messages transmis.

Chaque canal logique APEX (ou flux de données) est alloué sur *un lien virtuel*, dénommé VL pour Virtual Link. Les VLs permettent de ségréguer des flux de données échangés entre les composants afin de garantir la robustesse aux fautes. Un VL simule l'unicité d'un flux dans le réseau de communication par l'allocation d'un débit unique pour ce lien et en organisant la transmission des données en flux sporadiques. Un VL relie un émetteur à une ou plusieurs destinations, par un lien unidirectionnel multicast. Un VL est principalement caractérisé par:

- une taille maximum de trame, notée $SMAX$, qui représente la taille maximum de la trame AFDX qui peut être émise sur le VL. $SMAX$ peut varier de 64 octets à 1518 octets pour un VL;

– une taille minimale de paquet, notée $SMIN$;
– un temps minimum entre deux émissions de paquets consécutifs d'un même VL. C'est le *Bandwidth Allocation Gap* noté BAG. Les valeurs des BAG sont choisies dans ensemble des valeurs qui croissent exponentiellement en puissance de 2 : {1,2,4,...,64,128}. La Figure 2.6 illustre le flux de paquets sporadiques en entrée d'un VL.

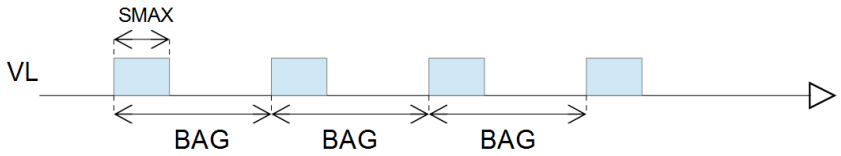


FIGURE 2.6 – Fonctionnement d'un VL

L'intégration des partitions est réalisée via la couche logicielle fournie par l'interface APEX, qui se positionne au dessus du système d'exploitation des CPMs et de leur matériel.

Implantation APEX : Cette couche APEX, présente sur chaque module, ordonnance les partitions allouées à ce module selon la MAF pré-calculée. Elle gère les accès mémoire, et fournit les ports APEX pour les communications inter-modules. Lors du déploiement, il est nécessaire de paramétrer cette couche APEX en fournissant principalement la MAF, c'est-à-dire la description complète de l'ordonnancement temporel des partitions du module.

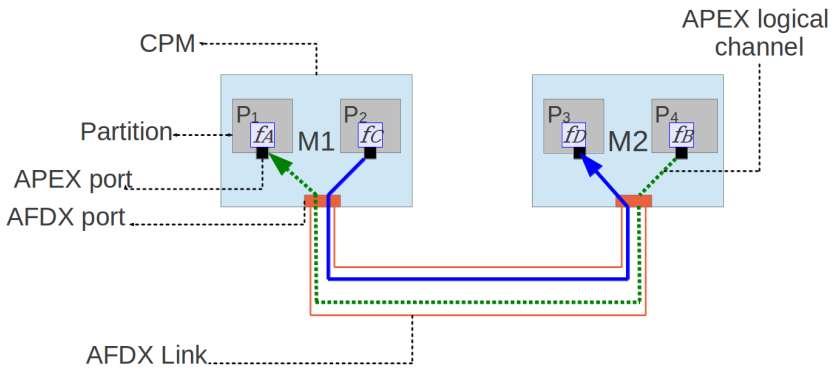


FIGURE 2.7 – Exemple d'une architecture

Exemple IMA : La Figure 2.7 donne un exemple très simple de système IMA composé de deux CPMs, qui hébergent respectivement deux partitions communicantes. Une seule tâche est allouée sur chaque partition. Les tâches f_B et f_A (resp. f_C et f_D) appartiennent à la même application avionique. f_B émet périodiquement des messages vers f_A à travers son port APEX d'émission. Celui-ci est spatialement alloué sur un lien virtuel VL AFDX (lien en pointillés verts)

connecté à la partition destination P_1, qui héberge la fonction f_A. De la même façon, f_C émet périodiquement des données vers f_D à travers un second lien virtuel AFDX (lien bleu).

2.2 Conception d'un système modulaire IMA

Comme présenté sur la Figure 2.8, une centaine de modules de calcul sont embarqués sur l'A380, interconnectés par un millier de liens virtuels AFDX.

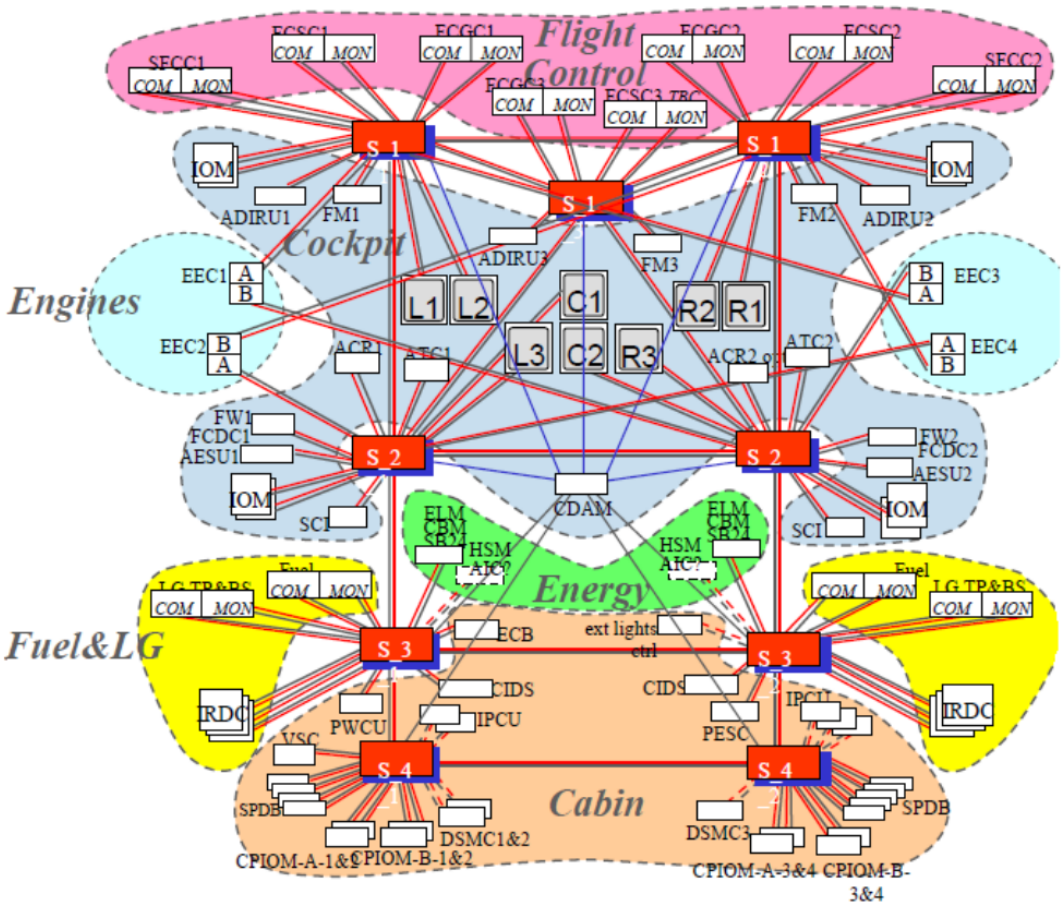


FIGURE 2.8 – Architecture IMA distribuée sur un Airbus A380

Pour concevoir un tel système, nous avons identifié les grandes étapes suivantes, illustrées par le schéma 2.9 :

1. Définition du niveau fonctionnel. La première étape de conception est liée à la spécification des éléments du niveau de modélisation fonctionnel. Ce niveau représente toutes les fonctionnalités du système final. Il est complètement indépendant de l'architecture matérielle. Nous

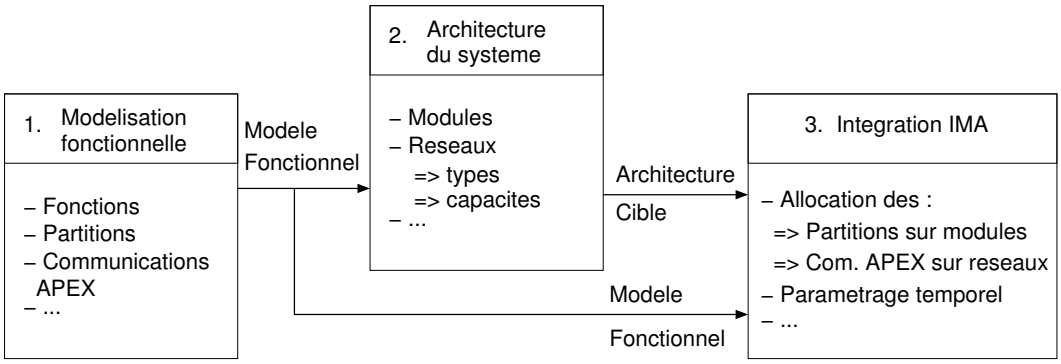


FIGURE 2.9 – Principales étapes de conception d'un système modulaire IMA

supposons ici que le niveau de modélisation système est défini par l'avionneur. Nous connaissons toutes les fonctions avioniques, les partitions qui les hébergent et les communications APEX qui les lient. Cette étape spécifie également les contraintes temporelles qui s'appliquent aux communications temps-réel.

2. Définition du niveau système. Une seconde étape consiste en une première *définition de l'architecture cible* qui devra héberger l'ensemble des fonctions avioniques. Pour définir cette architecture, une étape de dimensionnement s'impose pour déterminer le nombre de modules de calculs, ainsi que la nature et la capacité des réseaux physiques qui les interconnent. Cette étape de dimensionnement nécessite, entre autres, la prise en compte du modèle système et des contraintes industrielles de l'avionneur. Dans cette thèse, nous ne nous intéressons pas à ce problème de dimensionnement car les choix présentent des contraintes industrielles difficiles à identifier. Nous considérons que l'architecture cible est figée et capable d'héberger l'ensemble des fonctions avioniques du modèle système.

3. Intégration. La troisième étape de conception est *l'étape d'intégration du modèle fonctionnel sur l'architecture cible*, issus des deux premières étapes de conception. Cette thèse traite exclusivement de cette étape d'intégration des systèmes embarqués modulaires temps-réels. Cette étape alloue les différents fonctions et canaux APEX aux modules et liens réseaux de l'architecture cible. Pour cela, il faut prendre en compte toutes les contraintes système, et tout particulièrement respecter les contraintes temporelles imposées par les fonctions avioniques. C'est actuellement le rôle d'une équipe d'intégration. Les principales tâches sont les suivantes :

- l'allocation des partitions embarquées sur les calculateurs distribués avec définition des partitions,
- l'allocation des canaux de communication logiques sur les liens physiques existant dans le réseau de bord,
- le paramétrage temporel des partitions. C'est le choix des périodes et l'ordonnancement des partitions pour créer les MAFs de chaque module.

On peut définir deux grands types de problèmes à résoudre pour réaliser les tâches précédentes. La première famille est relative à des problématique d'allocation spatiale et la seconde s'intéresse au paramétrage temporel du système IMA.

Intégration IMA : Allocation spatiale

L'allocation spatiale prend en entrée le résultat de la modélisation de niveau système et l'architecture cible. Il s'agit ici d'allouer les partitions aux différents modules de calcul et les communications logiques inter-partitions aux canaux physiques disponibles des réseaux déployés. Chaque partition est affectée à un module de calcul connaissant l'espace mémoire disponible et la vitesse de traitement du calculateur. Le choix du lien physique (AFDX, ARINC 429, ...) pour un canal logique APEX dépend de plusieurs paramètres, tels que le débit du réseau, l'emplacement du composant source et du composant cible, le partage du réseau, ainsi que la carte de routage possible. Une fois qu'une allocation spatiale valide est obtenue, l'intégrateur connaît la cartographie des partitions sur les modules et l'utilisation du réseau. Une allocation valide est une allocation où toutes les partitions et canaux logiques sont alloués sur l'architecture cible. Les partitions possèdent leur propre périodicité et durée d'exécution. On peut donc calculer leur charge de calcul et choisir le module capable de servir cette charge.

Pour pouvoir valider une allocation spatiale, il faut vérifier que les contraintes temporelles applicatives sont respectées. Pour cela, il faut construire un ordonnancement des partitions (i.e. la MAF) pour chaque module. Connaissant ces MAFs, on doit vérifier que les contraintes temporelles sont respectées pour toutes les communications inter-modules.

Si cette validation temporelle est réalisée avec succès, l'intégration proposée est validée et peut-être déployée. Si cette validation temporelle échoue, on cherche généralement une nouvelle allocation spatiale jusqu'à en trouver une qui soit temporellement valide. Cette recherche peut s'autoriser à modifier le nombre de modules ou la capacité du réseau.

Intégration IMA : Allocation temporelle

Dans cette thèse, nous proposons un mode de résolution différent. Nous supposons qu'une première allocation spatiale a été déterminée de façon à simplement remplir la contrainte de capacité des modules et du réseau. Pour parvenir à une solution valide temporellement, nous introduisons une étape d'allocation temporelle qui ne changera pas l'allocation spatiale des partitions aux modules, ou l'allocation des canaux APEX aux liens réseaux, mais modifiera le comportement temporel du système de façon à rendre l'allocation valide temporellement parlant.

Pour cela, nous nous autorisons à modifier la périodicité des partitions qui sont uniquement réceptrices de communications APEX. La période des partitions émettrices est figée de par la modélisation système qui prend en compte le comportement in fine des fonctions avioniques. En s'autorisant à modifier les périodes des partitions en réception, nous montrons dans cette thèse qu'il est possible d'obtenir une allocation valide temporellement parlant, sans avoir à recalculer une nouvelle allocation spatiale complète.

Cette nouvelle méthode d'allocation temporelle est décrite en détails dans la suite de ce manuscrit car elle constitue le cœur de ces travaux de thèse.

2.3 Validation temporelle d'une architecture modulaire

Dans la phase d'intégration, il est nécessaire de dégager une ou plusieurs configurations valides. Ces configuration respectent les contraintes temporelles définies dans la modélisation fonctionnelle. Ces contraintes s'appliquent aux délais de communication de bout-en-bout (i.e. entre deux partitions communicantes). Pour pouvoir vérifier ces contraintes, il faut pouvoir calculer un ordonnancement temps-réel des partitions pour chaque module. Cette section définit dans un premier temps les notions de délais et de contraintes fonctionnelles de bout-en-bout. Dans un second temps, elle présente les spécificités du problème d'ordonnancement des partitions IMA. Pour finir, elle exprime comment peut se calculer ou se modéliser le délai de communication induit par le réseau de communication.

2.3.1 Contraintes et délais fonctionnels de bout-en-bout

La conception et la réalisation des fonctions embarquées suivent un cahier des charges rigoureux qui liste un nombre d'exigences et besoins de performances pour s'assurer du bon fonctionnement et de la sûreté des composants réalisés. Certaines exigences sont "traduites" en contraintes temporelles de bout-en-bout, c.à.d de la génération de la donnée par sa partition source jusqu'à sa consommation par une partition cible [15].

Nous nous intéressons au délai fonctionnel de bout-en-bout. Le délai fonctionnel de bout-en-bout est la somme de plusieurs latences nécessaires au transfert d'une donnée d'un producteur vers un consommateur. La Figure 2.10 modélise selon les deux niveaux d'abstraction présentés précédemment la décomposition d'un délai fonctionnel de bout-en-bout d'une fonction réalisée entre une partition source et une partition destination dans le contexte de l'IMA. Les deux partitions sont distribuées sur deux CPMs distants interconnectés à travers un réseau AFDX. Le délai de bout-en-bout est composé du temps passé dans le système et dans le réseaux de communication. Une donnée est générée dans une partition source. Cette donnée est transmise une fois la partition écoulée via l'interface APEX au réseau de communication. Dès que le protocole AFDX le permet, la donnée est envoyée sur le réseau et réceptionnée dans un port de réception au niveau du module destination. La donnée attendra que la partition destination s'active pour être lue dans le port de réception. En d'autres termes, le délai fonctionnel de bout-en-bout est composé d'un délai d'attente d'activation de la partition source, du délai de communication de bout-en-bout nécessaire à la traversée du réseau AFDX, ainsi que l'attente de l'activation de la partition distante cible.

Les communications dans un environnement temps réel sont soumises à des contraintes afin de s'assurer du respect des échéances. Différentes contraintes de bout-en-bout peuvent être utilisées :

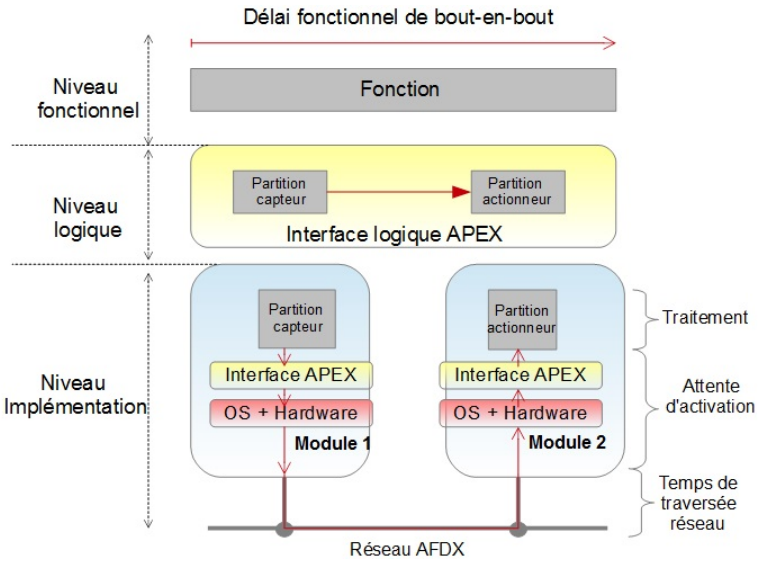


FIGURE 2.10 – Délai fonctionnel de bout-en-bout

- *La fraîcheur des données*, qui exprime que le délai entre la production d'une donnée par le producteur et sa consommation sur le port du consommateur doit être inférieur à une certaine valeur, appelée délai de bout-en-bout *pire cas*,
- *La cohérence des mise-à-jour*, qui exprime que le fait que tous les messages envoyés sont tous consommés par leur destination. Un message ne doit pas être écrasé par une mise-à-jour avant sa consommation. En effet, il ne suffit pas d'avoir un débit de communication élevé dans le réseau afin d'assurer des délais de bout-en-bout intéressant, il faut également se pencher sur les latences fonctionnelles qui affectent le délai de bout-en-bout.

Dans nos travaux de recherche, nous nous intéressons au délai fonctionnel de bout-en-bout, soumis à ces deux contraintes temporelles de fraicheur et de cohérence. Ces paramètres sont déterminant dans le calcul de l'allocation temporelle.

2.3.2 Ordonnancement des partitions

Dans le contexte des architectures distribuées modulaires, deux niveaux d'ordonnancement sont réalisés: l'ordonnancement de partitions contenues dans un même module, et l'ordonnancement des tâches ou fonctions au niveau d'une seule partition [16]. Chaque partition est composée d'une unique ou de plusieurs fonctions ordonnancées localement par un ordonnanceur spécifique. Toutes les partitions qui sont allouées sur la même unité de calcul sont par la suite ordonnancées par un ordonnanceur généralisé par le système d'exploitation temps réel global.

Ordonnancement intra-partition

Les tâches d'une même partition peuvent partager ses ressources et s'exécuter de manière concurrente. Un algorithme d'ordonnancement est utilisé localement à chaque partition. Les tâches sont suspendues quand le temps d'exécution qui a été alloué à la partition qui les héberge est écoulé.

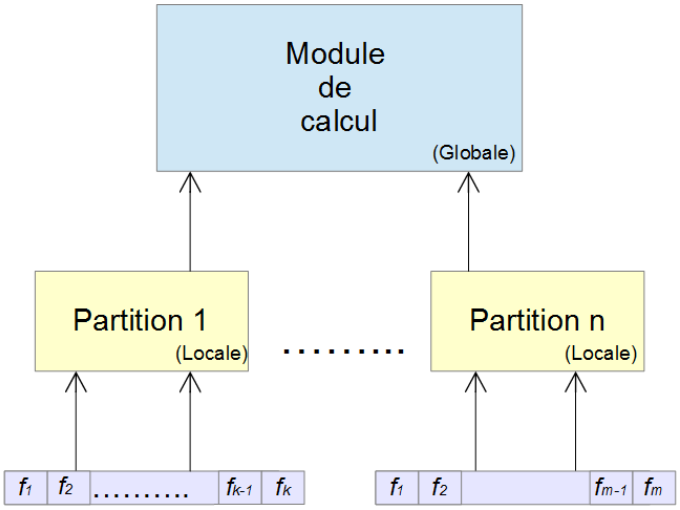


FIGURE 2.11 – Ordonnancement à deux niveaux

Ordonnancement inter-partition

Les tâches sont exécutées au sein des partitions qui sont elles même ordonnancées selon une un ordre statique, global au module. Quand une partition est activée dans la fenêtre d'exécution qui lui a été allouée, les tâches en cours d'exécution dans la partition précédemment active sont suspendues pour laisser la place aux tâches de la partition récemment activée. Dans l'ARINC 653, l'ordonnancement est déterministe [17], c-à-d. que les instants de démarrage de toutes les partitions sont connus avant la mise en route du système. Ils sont définis par la Major Time Frame ou MAF.

Dans cette thèse, nous nous intéressons à l'ordonnancement des partition sur un module IMA, c'est-à-dire à l'ordonnancement inter-partition.

2.3.3 Propriétés des ordonnancements temps-réel

L'ordonnancement des partitions sur un module est dit *valide* si toutes les contraintes temporelles sont respectées. Pour chaque partition, plusieurs contraintes temporelles peuvent être définies. La validation d'un ordonnancement de partitions partageant des ressources de calcul

et de communication consiste à vérifier que toutes les contraintes temporelles de fraicheur et de cohérence des mise-à-jour sont respectées.

Un algorithme d'ordonnancement peut être *en ligne* ou *hors ligne*. Les algorithmes en ligne décident lors de l'exécution du système de la tâche à exécuter parmi les tâches prêtes. Les algorithmes hors ligne pré-calculent un ordre valide des tâches. Une fois un ordre valide trouvé, il est configuré sur le système qui l'utilisera sans le modifier.

Un système temps-réel peut être *préemptif* ou *non-préemptif*. Dans le premier cas, une tâche peut être interrompue au cours de son exécution par le système avant sa date de fin. Dans un système non-préemptif, il est interdit d'interrompre une tâche. Une tâche s'exécute ainsi sur une unité de calcul jusqu'à sa terminaison complète.

Cas des partitions dans l'architecture avionique IMA

Un module de calcul peut héberger plusieurs partitions IMA. Ces partitions sont alors exécutées de manière concurrente, partageant ainsi les ressources de traitement et le réseau de communication. Les propriétés des partitions sont connues en amont et sont statiques (durée d'exécution, échéance, fréquence d'exécution).

L'ordonnancement des partitions d'un module est choisi de manière à garantir les contraintes temporelles des fonctions avioniques embarquées. Les partitions s'exécutent sans priorité entre elles, de manière *strictement périodique* en respectant un ordonnancement (la MAF) calculé à l'avance par un algorithme *hors ligne*. Une partition se doit d'accomplir son traitement dans la fenêtre d'exécution temporelle qui lui a été dédiée par le système d'exploitation et est suspendue à la terminaison du temps qui lui est alloué dans la MAF. Aucune préemption n'est permise. Un autre point important est le choix des périodes d'un module : ces périodes sont en général harmoniques.

Compte tenu de la nature des partitions dans un système IMA, cette thèse va se baser sur des algorithmes d'ordonnancement hors ligne, non préemptifs où les partitions sont *strictement* périodiques et harmoniques.

2.3.4 Ordonnancement dans les systèmes temps réel critique à périodes strictes non préemptives

Dans cette thèse, nous avons besoin de répondre à la question suivante :

Pour un ensemble de partitions données, existe-t-il un séquencement des partitions qui respecte les contraintes temporelles du système, sachant que les partitions sont à périodes strictes et ne peuvent être préemptées ?

Un ordonnancement strictement périodique est un ordonnancement où, pour chaque partition, deux exécutions consécutives sont espacées d'une et une seule période.

Pour déterminer si un ensemble de partitions est ordonnançable, plusieurs méthodes peuvent être utilisées :

- L'utilisation de tests d'ordonnançabilité: ces tests permettent, directement à partir de la définition des partitions, de décider de l'existence d'un ordonnancement valide ou non, et ce pour un algorithme de construction donné. Ces tests sont définis sous la forme de conditions nécessaires et / ou suffisantes.
- La construction d'un ordonnancement valide complet. Cette méthode peut-être bien plus complexe que la précédente. Une recherche exhaustive de tous les ordonnancements possibles ne passe pas à l'échelle, ce qui signifie qu'il faut définir une heuristique dédiée.

Les tests sont très rapides à calculer, néanmoins, ils peuvent être très restrictifs : ils ne détectent comme ordonnançable qu'un nombre restreint de systèmes. Quand les tests ne sont pas concluants, on a recours à la construction d'un ordonnancement valide. L'algorithme de construction peut ne pas être optimal dans le sens où il se peut qu'il n'arrive pas à construire un ordre des partitions valide alors qu'il en existe un. Néanmoins, certaines heuristiques sont bien moins pessimistes que les tests disponibles.

L'ordonnancement non préemptif a suscité l'intérêt de nombreux travaux de recherche. Jeffay et al. [18] prouvent que le problème d'ordonnancement de tâches temps réel non préemptives est classé parmi les problèmes NP-difficiles au sens fort. Les auteurs dérivent une condition nécessaire et suffisante (un test) pour l'algorithme d'ordonnancement non préemptif Earliest Deadline First (npEDF) quand les tâches sont périodiques avec des offsets de démarrage arbitraires. Toutefois, cette condition est très pessimiste [19] et risque de rejeter un nombre considérable d'ordonnancements faisables. Korst et al. [20] proposent une autre condition d'ordonnançabilité nécessaire et suffisante pour un système de deux tâches. Cette condition a été généralisée à un ensemble de tâches multiples par Kermia et al.[21]. Récemment, Nasri et al. [19] montrent que les tests d'ordonnançabilité proposés jusqu'à présent sont très pessimistes et conservateurs, tout particulièrement si les périodes des tâches sont harmoniques entre elles.

Dans la section suivante, nous citons les travaux qui abordent le cas particulier des périodes harmoniques pour l'ordonnancement strictement périodique, non préemptif.

Cas particulier des partitions à périodes harmoniques

Un ensemble de tâches (ou partitions) est dit harmonique si pour chaque couple de partitions P_i et P_i+1, il existe un k_i, tel que $T_{i+1} = k_i * T_i$ [1]. Par exemple, l'ensemble de tâches de périodes respectives {20, 40, 120} est un ensemble à périodes harmoniques avec k_1=2 et k_2=3 ici). Dans ce cas, l'hyperpériode est égale à la période maximale du jeu de tâches.

Les tâches à périodes harmoniques permettent la mise en oeuvre d'une analyse d'ordonnançabilité en un temps polynomial, pour la taille réduite de l'hyperpériode [22][23]. Les jeux de tâches à périodes harmoniques sont largement utilisés dans différents domaines industriels tels que l'avionique, les systèmes sous-marins ou les systèmes radars [24].

1. Une définition plus précise de l'harmonicité est donnée par l'équation (**??**), p. **??**

Beaucoup de travaux ont été menés afin de transformer des périodes de tâches/partitions en périodes harmoniques. Parmi les premiers, Han et Tyan avec l'algorithme *Sr*[25] qui dérive les périodes maximales (artificielles) inférieures aux valeurs originales. Dans les travaux récents de Mitra Nasri et al. [26] , les auteurs proposent un outil de construction de périodes harmoniques personnalisées pour les systèmes temps réel. Ils proposent un modèle afin de décrire les relations harmoniques entre gammes de valeurs (tâches avec une gamme de périodes comprises entre une valeur minimum et une valeur maximum). Toutefois l'ordonnancement de tâches à périodes harmoniques et non préemptives reste un problème NP-difficile au sens fort selon [27].

Récemment, Nasri et al [28] ont proposé un algorithme d'ordonnancement nommé *Precausious-RM* ou *P-RM*. Precausious-RM est un algorithme en ligne pour l'ordonnancement des tâches périodiques, non préemptives, harmoniques qui présente une complexité linéaire. Ils montrent que sous certaines conditions, *P-RM* est optimal. Un algorithme relaxé dérivé de P-RM est aussi proposé par la suite, dénommé *Lazy P-RM*. Cet algorithme améliore les performances de P-RM quand les durées d'exécution des tâches sont courtes.

Les mêmes auteurs discutent par la suite dans [19] de non-preemptif Rate Monotonic (ou npRM) et non-préemptif Earliest Deadline First (ou npEDF), pour le cas spécifique des tâches à périodes harmoniques, puis dérivent une condition suffisante d'ordonnançabilité pour npRM et npEDF. Il est montré que la condition nécessaire et suffisante d'ordonnançabilité de npEDF proposée par Jeffay et al. [18] est très pessimiste dans le cas des tâches harmoniques et est réalisée avec une complexité de calcul considérable.

L'ordonnancement de ce type de tâches (strictement périodiques, harmoniques et non préemptives) constitue un problème d'actualité. Toutefois, les travaux cités précédemment abordent le cas des ordonnancement *en ligne* et ne sont pas adaptés au cas de l'ordonnancement *hors ligne*.

Dans le cas des partitions d'un module IMA, l'ordonnancement hors ligne est réalisé par un *séquenceur* : une séquence est calculée avant la mise en route du système, et est mémorisée dans une table de séquencement que le système va suivre pendant toute sa durée de vie. La construction du séquencement hors-ligne permet à la fois de vérifier l'ordonnançabilité de l'ensemble de partitions et de fournir une séquence valide.

Il n'existe pas, à notre connaissance, d'algorithme *optimal* moins complexe qu'une recherche exhaustive similaire à la recherche de Bratley [29]. La recherche de Bratley construit l'arbre de tous les ordonnancements possibles, en élagant le plus tôt possible les branches qui violent une contrainte temporelle. La recherche de Bratley teste au pire cas l'ensemble des $!n$ ordres possibles. Cette complexité ne peut être admise pour des systèmes important. Dans ce cas, il nécessaire de recourrir à des heuristiques sous-optimales. Ces heuristiques peuvent ainsi ne pas construire un ordonnancement valide alors qu'il en existe un. Dans cette thèse, nous avons choisi d'utiliser l'heuristique Least Loaded pour le séquencement des partitions IMA car elle permet de décider de l'ordonnançabilité d'un nombre important de systèmes de tâches.

Algorithme Least Loaded

L'algorithme *Least Loeaded* ou *LL* est une heuristique proposée par Grenier et al. [30] pour réduire le temps de réponse des trames dans un réseau embarqué automobile CAN. LL ordonnance les trames CAN dans des slots d'émission des messages en introduisant des offsets au flux périodiques CAN. Cette stratégie permet de réduire considérablement les temps de réponse des messages en lissant le traffic.

L'ensemble des tâches $R_i \in \mathcal{R}$
1- Ordonner les $R_i \in \mathcal{R}$ par ordre croissant des périodes d'exécutions T_i où $T_{slot} \leq T_1 \leq ...T_n \leq T_{cycle}$
2- Pour chaque R_i $i = 1...n$:
  (a) Chercher le slot k le moins chargé parmi les premiers $\frac{T_i}{T_{slot}}$ slots,
  (b) Placer R_i tous les $\frac{T_i}{T_{cycle}}$ slots en commençant par le slot k (offset de R_i est donc $k \times T_{slot}$)
  (c) Mettre à jour la charge du coeur occupé

FIGURE 2.12 – Algorithme Least Loaded pour le placement des runnables AUTOSAR [2]

Similairement, dans un environnement multi-coeurs, Monot et al. [2] se sont inspirés de cette heuristique pour le placement de runnables AUTOSAR (i.e. tâches de niveau fonctionnel) de périodes harmoniques dans un des cœurs de l'architecture. Ce scénario est très proche de notre problème d'ordonnacement des partitions sur les modules. Nous allons expliquer LL dans le contexte du placement de tâches périodiques harmoniques sur une unité de calcul dans ce qui suit.

LL définit la longueur d'un slot comme étant la durée élémentaire. Le slot est un diviseur du cycle utilisé par le séquenceur pour ordonnancer toutes les tâches. Dans [2], ces valeurs sont de taille 5 ms pour un slot, noté T_{slot}, et de 1000 ms pour un cycle de la tâche séquenceur, noté T_{cycle}. L'idée est de répartir les tâches périodiques sur les différents slots, tout en respectant leur périodicité et dans l'objectif de répartir au mieux la charge dans le temps. Le résultat est un lissage de la charge des coeurs afin d'équilibrer l'environnement et ne pas atteindre des situations de pic de charge qui seraient dûs à une distribution moins équitable des tâches.

L'algorithme LL pour le séquencement des runnables sur un multi-coeur est illustré en une version simplifiée dans 2.12. Les tâches sont d'abord ordonnées par période croissante. Chaque tâche, dans cet ordre, est alors placée dans le slots le moins chargé parmi les premiers slots disponibles. La Figure 2.13 montre le séquencement obtenu en appliquant l'heuristique LL à l'ensemble des rennables $R_i(T_i, C_i)$ suivant : R_1(10, 2), R_2(10, 1), R_3(20,3) et R_4(20, 2) pour T_{slot}=5 et T_{cycle}=40. Nous obtenons ici un séquenceur de tâches composée de 8 slots séquencés selon la Figure 2.13.

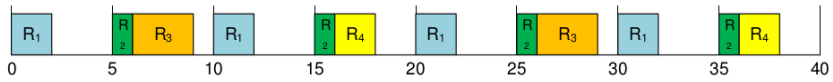


FIGURE 2.13 – Exemple d'un séquencement avec l'algorithme LL de [2]

2.4 Évaluation des délais de communication de bout-en-bout

Dans la section précédente, nous avons listé les principaux éléments qui permettent de valider temporellement une intégration IMA. Nous avons donc introduit d'une part les contraintes temporelles qui s'appliquent aux délais de bout-en-bout et d'autre part le problème d'ordonnancement des partitions sur les modules. Pour connaitre le délais de bout-en-bout, il faut être capable de calculer le délai de communication dans le réseau. Ce délai dépend bien entendu du protocole utilisé (AFDX, ARINC 429,...), mais aussi de l'ordonnancement des partitions dans les modules.

La connaissance des performances dans les réseaux des systèmes embarqués repose dans la majorité des cas sur une simplification du réseau : les communications sont modélisées au travers des paramètres de qualité de service qu'elles offrent, tel que les meilleurs et pires temps de traversée du réseau [31]. Les méthodes d'évaluation des performances des réseaux embarqués permettent d'évaluer la qualité que peuvent offrir les communications. De nombreux travaux ont été réalisés dans ce domaine. Différentes approches et méthodes existent :

- Le calcul d'une borne supérieure sûre sur les délais de communication
- Le calcul du délais de communication pire cas exact.

Ces méthodes sont brièvement décrites dans ce qui suit.

2.4.1 Borne supérieure sur les délais de communications

Ces méthodes ont fait leurs preuves pour l'obtention d'une borne supérieuse sur de traversée d'un réseau type AFDX. Elles permettent d'obtenir des bornes sûres. Les deux méthodes de calcul sont :

- **Le network calculus:** introduit par Le Boudec et Thiran [32], le network calculus est largement utilisé pour l'analyse des garanties des performance dans les réseaux informatiques. Il a été appliqué au réseau AFDX d'abord par Christian Fraboul et al. dans [33], ce qui a conduit à la certification du réseau AFDX pour l'Airbus A380.
- **L'approche par trajectoire:** méthode proposée par Steven Martin [34], elle est capable de fournir des bornes supérieures pessimistes que le network calculus, notamment pour l'AFDX comme présenté par Henri Bauer [35]. Les bornes obtenues sont pessimistes. Toutefois, des études ont montré que ce pessimisme est limité par des configurations avioniques classiques [35].

2.4.2 Délais de communication exacts pires cas

Afin de minimiser le pessimisme dans l'évaluation des réseaux temps réel, des méthodes de calcul exactes du délai pire cas ont été proposées. Ces méthodes ne passent malheureusement pas bien à l'échelle, même si des avancées importantes ont été réalisées récemment en model checking. Nous citons les méthodes suivantes :

- **La simulation exhaustive:** l'approche par simulation exhaustive analyse tous les scenarii possibles. Adnan [36] s'est intéressé à cette approche pour le calcul des délais exacts pires cas en AFDX.
- **Le Model Checking:** le Model Checking peut être utilisé pour des systèmes de petite taille pour fournir des délais exacts pire cas. Il a permis, notamment par les travaux d'Adnan [37], de dériver les délais pire cas pour des systèmes de taille moyenne.

Dans le cadre de cette thèse, nous supposerons que les bornes supérieures et inférieures sur les délais de communication dans le réseau sont connues. Elles peuvent être calculées par les méthodes présentées précédemment.

2.5 Méthodes d'aide à l'intégration des architectures modulaires

2.5.1 Principales méthodes

La complexité du processus d'intégration a poussé la recherche scientifique à proposer différentes solutions d'aide à l'intégration. Des solutions pour l'aide à l'intégration d'architectures pour le domaine automobile, plus particulièrement l'AUTOSAR, sont proposées dans [38], [39], [40]. Dans le domaine des architecture modulaires avioniques, des travaux s'intéressent à l'aide à l'intégration dans l'IMA [41] [42] [43] [44]. Une solution plus générique pour des systèmes temps réel distribués a aussi été proposée dans [45].

La recherche d'une allocation intéressante nécessite l'exploitation des propriétés et contraintes du système, de quantifier une allocation à travers des critères ou métriques (tel que le poids de l'appareil, la charge des calculateurs, la latence de bout-en-bout, ...), et enfin d'exploiter ces critères pour arriver à une allocation satisfaisante, par l'optimisation de ces critères (réduction de la charge des calculateur, augmentation de la marge dans le réseau, ...).

Dans la partie 2.2, nous avons introduit les principales problématiques relatives à l'intégration d'un système modulaire. Les travaux qui suivent se sont pour la plupart intéressés aux problème d'allocation spatiale des partitions (ou tâches) sur les modules de calculs. Cette allocation est souvent couplée au dimensionnement ou re-dimensionnement de l'architecture cible. Les solutions proposées sont bien entendu vérifiées temporellement par la création d'un ordonnancement valide des partitions sur chaque module. Ces méthodes ont pour variables les allocations possibles, et si une allocation ne respecte pas les contraintes temporelles, l'algorithme recherche une nouvelle

allocation. Plus spécifiquement, [45], [46], [47], [38] et [43] se sont intéressés à l'aide à l'intégration par la proposition d'outils et d'algorithmes d'optimisation du placement des partitions sur différents modules, afin de garantir les contraintes temporelles et l'ordonnançabilité du système. Dougherty et al [41] cherche en plus à minimiser le nombre des modules de calcul de l'architecture cible.

Les travaux d'Al Sheikh [42] et X.He et al. [45] sont à notre connaissance les seuls à avoir proposé dans la méthode d'aide à l'intégration, une solution de placement et d'allocation temporelle des paramètres des partitions embarquées. Tous les deux proposent des solutions pour l'ordonnancement multi-processeurs allié à l'assignation des périodes d'exécution des partitions dans l'architecture. Ces travaux cherchent à la fois à allouer les partitions aux modules, mais ajustent également la période d'exécution des partitions. Ces sont les travaux les plus proches de notre thèse. Néanmoins, la complexité de leur approche est bien supérieure à la notre, car ils s'attellent à la fois au problème d'allocation spatiale et temporelle.

Le problème d'intégration peut être formulé par un problème d'optimisation monocritère, où l'on cherche à optimiser une seule métrique comme le nombre de modules [45], la robustesse [47], le potentiel d'évolutivité [42], etc. Le problème a aussi été formulé pour optimiser deux critères ou plus. Par exemple, on trouve l'optimisation du nombre de calculateurs parallèlement à la réduction du débit réseau dams [41], la réduction du délai de bout-en-bout et de la consommation mémoire [38] [39], la réduction du poids du matériel et le coût des interruptions opérationnelles [43] [44], etc. Pour résoudre le problème d'intégration, des algorithmes d'optimisations sont proposés ou adaptés aux formulations choisies. Nous avons choisi dans cette thèse de formuler aussi notre problème en optimisant conjointement deux critères de performance que sont la charge des modules et la marge réseau.

Les algorithmes adaptés aux formulations multicritères sont rarement optimaux. En effet, ces algorithmes calculent des solutions de compromis en résolvant, par exemple, des programmes linéaires en nombre entiers [42], [39]. La complexité des problèmes d'allocation spatiale nécessite souvent l'utilisation de métaheuristiques : algorithme générique dans [41], recuit simulé couplé à la programmation géométrique dans [45] ou l'exploitation d'un solver branch and cut adapté au multiobjectif par calcul de front de Pareto [43] [44]. Nous allons aussi dans cette thèse nous appuyer sur une métaheuristique pour résoudre le problème d'allocation temporelle. Néanmoins, nous allons tout de même développer un algorithme de recherche optimal, multicritères qui extrait l'ensemble des allocations temporelles Pareto-optimales.

2.5.2 Description des travaux connexes

X.He et al. [45], pour l'optimisation de l'allocation distribuée de tâches dans une architecture temps réel, exploitent l'attribution des propriétés d'exécutions des tâches temps réel, l'assignation des périodes d'exécution ainsi que la configuration des accès nécessaires aux bus de communication. Ils présentent un outil d'optimisation pour les systèmes contenant des tâches périodiques distribuées. L'outil implémente la métaheuristique recuit simulé [48], couplée à de la

programmation géométrique [49] dans le but d'explorer les différentes configurations du système. Les contraintes et délais des communications entre tâches ne présentent pas une donnée de la solution proposée.

Pour des soucis du réduction du nombre de calculateur et du débit du réseau, Dougherty et al [41] introduisent et valident une méthode d'allocation spatiale, dérivée du problème de bin packing [50]. La résolution est menée en dérivant un algorithme génétique intégré à un outil nommé ScatterD [3]. Le problème d'allocation spatiale pour la flight avionics systems ou FAS est formulé comme un problème d'optimisation multiobjectif qui minimise le nombre de calculateurs et le débit du réseau. Cet outil est illustré sur la figure 2.14. La faisabilité des ordonnancements est garantie par l'allocation spatiale, peu robuste aux évolutions du système.

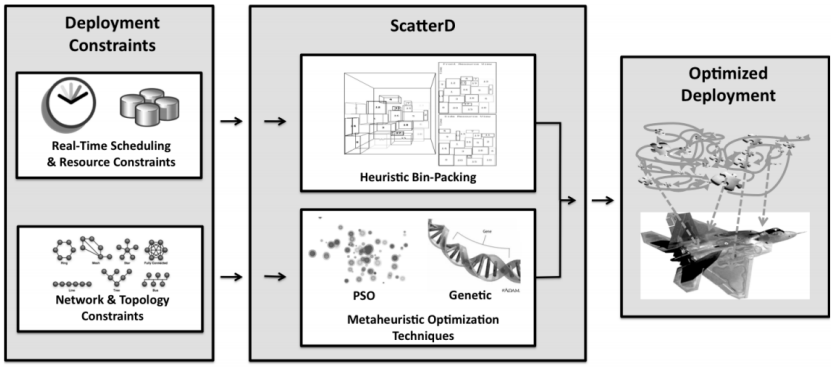


FIGURE 2.14 – Processus d'optimisation avec l'outil ScatterD [3]

M. Zhang et al. [46] s'intéressent à l'allocation spatiale dans le domaine automobile. Ils cherchent plus précisément à allouer les communications virtuelles sur le réseau embarqué, en faisant correspondre les ports de composants logiciels atomiques d'un modèle AUTOSAR avec les liens du réseau. Il formulent un problème d'optimisation multicritère qui minimise conjointement la charge des bus de communications et la charge de la mémoire utilisée, en garantissant l'ordonnançabilité de l'ensemble des tâches AUTOSAR.

Comme il a été noté par Mehiaoui et al. dans [38], très peu de travaux prennent en compte le délai de bout-en-bout comme une contrainte ou un objectif du problème d'intégration. Mehiaoui et al [38] [39] proposent formulation en programme linéaire en nombres entiers en deux étapes pour solutionner le problème multiobjectif de placement, de partitionnement et d'ordonnancement des fonctions et signaux sur un ensemble de calculateurs et communications d'une architecture AUTOSAR. Les objectifs à optimiser sont la minimisation de la latence de bout-en-bout et la minimisation de la consommation mémoire. Cette optimisation est réalisée en respectant un ensemble de contraintes telles que le rejet des groupements de fonctions dans la même tâche avec des périodes non harmoniques. Ils proposent par la suite une amélioration de la résolution par une méthode de type 'diviser pour régner' ainsi qu'une optimisation utilisant un algorithme génétique. Dans [40], Wozniak et al. utilisent la même méthode dans le but d'optimiser le temps

de réponse de bout-en-bout et la consommation mémoire dans un système AUTOSAR.

Al Sheikh [42] s'intéresse à la problématique de l'allocation des tâches strictement périodiques dans une architecture multiprocesseur dans le cadre des systèmes IMA. Ils proposent une méthode permettant de maximiser l'unique critère des durées d'exécution minimales d'inactivité entre deux exécutions de partitions en assurant qu'elles sont ordonnaçables sur le même CPM. Ce critère a été choisi pour garantir une marge d'évolution minimale des durées d'exécution des partitions. Ceci rend possible l'ajout de futures fonctions dans les partitions à moindres coût. En effet, l'intégration sera toujours valide si une partition est étendue dans le temps. La solution est modélisée sous forme d'un programme linéaire en nombres entiers. Une heuristique est ensuite proposée basée sur la théorie des jeux afin d'atteindre l'ordonnancement et le déploiement multiprocesseur avec la marge d'évolution la plus intéressante.

Annighofer et al. [43] [44] abordent, pour les architectures Distributed Integrated Modular Avionics, ou DIMA le déploiement d'applications et de composants dans un avion sous conditions de quelques contraintes : la réduction du poids du matériel et le coût des interruptions opérationnelles. Ils proposent une optimisation monobjectif avec la programmation binaire. Par la suite, ils utilisent un solver Branch-and-Cut [51] adapté au multiobjectif par calcul du Front de Pareto, avec le but de proposer des distributions de compromis pour l'optimisation du poids et des coûts. Ils montrent que le déploiement obtenu est aussi bien, voire meilleur qu'un mapping manuel. Toutefois, ces travaux n'abordent pas l'ordonnançabilité des différentes configurations

Nous avons choisi dans cette thèse de prendre en compte les délais de bout-en-bout entre fonctions embarqués comme données du problème d'optimisation, tout comme Al Sheikh [42] et Mehiaoui et al [38] [39] ont pu le faire. Dans Mehiaoui et al. [38] [39], le respect des délais de bout-en-bout est assuré par le calcul des latences pire cas de bout-en-bout et du replacement des fonctions dans l'architecture. La reconfiguration avec de nouvelles fonctions nécessite une migration répétitive des fonctions pour que le système reste ordonnançable.

Nous nous sommes aussi intéressés à la notion d'évolutivité proposée par les travaux d'Al Sheikh [42] car elle permet de créer des intégrations plus pérennes.

Comme nous l'avons déjà évoqué, les principaux travaux traitent du problème d'allocation des partitions et liens réseaux sur l'architecture cible. Ils prennent en compte les contraintes temps-réels relatives à l'ordonnançabilité des partitions sur les modules. Par contre, seuls quelques travaux intègrent des contraintes sur les délais de bout-en-bout.

Le problème d'allocation spatiale présente une complexité très élevée. Dans cette thèse, nous avons choisi de traiter le problème d'intégration différement pour en réduire la complexité. En effet, nous allons choisir une première allocation qui permet d'allouer toutes les partitions sans dépasser la capacité des modules ou du réseau. Connaissant cette allocation, nous allons modifier le paramétrage temporel des partitions (leur périodicité) pour arriver à une configuration valide temporellement parlant (ordonnançable et qui respecte les délais de bout-en-bout). Pour cela, nous allons définir un problème d'optimisation multi-critères qui favorisera le calcul de solutions évolutives. Ces solutions présenteront une certaine marge, que ce soit en puissance de calcul ou

en débit réseau, pour permettre un ajout ultérieur simple de nouvelles fonctionnalités dans le système.

2.6 L'allocation temporelle : un problème d'optimisation multicritère

Les travaux de recherche mentionnés précedemment modélisent le problème de dimensionnement d'une architecture temps réel comme un problème d'allocation spatiale et d'ordonnancement multiprocesseur. Une solution d'intégration est obtenue en optimisant un ou plusieurs objectifs de performance. Cette solution se définit comme un placement spatial des partitions (ou tâches) sur un ensemble de calculateurs distribués. Le but est généralement de minimiser la charge des calculateurs, tout en garantissant l'ordonnançabilité des partitions ou tâches pour chaque calculateur. Les contraintes liées aux réseaux et à la variabilité des délais de communication de bout-en-bout sont omis la plupart du temps.

Dans le cadre de cette thèse, nous allons prendre en compte les contraintes réseau pour dimensionner plus finement un système distribué temps réel. Pour cela, nous allons travailler dans le contexte de l'IMA et proposer une méthodologie d'intégration temporelle de bout-en-bout. On considère ici qu'une première allocation simple des partitions aux calculateurs a déjà été effectuée. Notre méthodologie permet de raffiner l'intégration en choisissant les paramètres temporels qui garantissent le comportement temps réel du système complet.

Cette méthodologie permet à l'intégrateur de définir une contrainte temporelle pour toutes les communications inter-partition que nous appelerons *paramètre de fraicheur*. Ainsi, toute donnée émise par une partition source à l'attention d'une partition destination située sur un autre calculateur devra se faire en une durée inférieure au paramètre de fraicheur. Si ce n'est le cas, la donnée est trop agée et la partition destination ne peut la traiter. L'objectif est de dimensionner le système de façon à éviter toute violation d'un des paramètres de fraicheur.

A partir d'une allocation des partitions sur les calculateurs, nous allons montrer que la prise en compte des contraintes temporelles sur les communications de bout-en-bout contraint le choix de la périodicité des partitions destination des communications. Modifier la période des partitions ne peut se faire que si l'on garanti aussi l'ordonnançabilité des partitions du module. Le problème d'allocation temporelle est donc un problème d'optimisation sous contraintes. Les variables sont les périodes des partitions destinataires des communications inter-partitions. Les contraintes sont à la fois applicatives (i.e. liées au système IMA) et des contraintes temps réelles (i.e. liées au respect du paramètre de fraicheur et à l'ordonnançabilité des partitions sur les modules).

Une première contribution de cette thèse est de re-formuler le problème d'allocation en un problème d'optimisation combinatoire où l'ensemble des allocations temporelles valides respectent les contraintes applicatives et temps réelles. Cette formulation est présentée dans le Chapitre 3.

Pour sélectionner une ou plusieurs allocations valides à proposer à l'intégrateur, ces travaux définissent un ensemble de métriques qui favorisent le choix d'allocations *évolutives*. Cette dé-

marche a pour but de simplifier la reconfiguration d'une allocation en simplifiant l'ajout futur de fonctions et de communications inter-partitions dans le système. Ceci peut s'envisager quand on cherche remettre à niveau un système existant pour intégrer de nouvelles fonctionnalités. Nous avons choisi de définir des métriques liées à la charge des calculateurs, mais aussi liées au potentiel d'évolution du réseau. Ainsi, si un calculateur présente une charge réduite, il sera relativement aisé d'y rajouter une nouvelle fonction. De même, si une solution d'allocation supporte l'ajout d'une communication sans introduire une violation d'un des paramètres de fraicheur, on dira que l'allocation est évolutive pour le réseau. Ces critères sont étudiés au Chapitre 4 et constituent la seconde contribution de cette thèse.

La troisième contribution s'intéresse à la conception d'un algorithme d'optimisation multicritère dédié au problème d'allocation temporelle ainsi défini. Nous allons définir deux heuristiques au Chapitre 5, l'une permettant la résolution exacte du problème et l'autre la résolution approchée :

- La première heuristique exploite les spécificités de l'architecture distribuée pour réduire la combinatoire de la recherche exhaustive. Ainsi, nous sommes capables, avec des ressources de calcul standard, de remonter toutes les allocations optimisant conjointement les critères de charge et d'évolutivité réseau pour des systèmes de taille moyenne composés d'une vingtaine de calculateurs.
- La seconde se base sur une version multicritère de la métaheuristique Tabou. Nous montrons au Chapitre 6 que cette heuristique permet d'identifier rapidement des allocations proches de l'optimal. Elle permet donc de remonter des solutions pour des systèmes de plus grande taille que la méthode exhaustive.

2.7 Conclusion

Ce chapitre 2 nous a permis d'introduire le contexte scientifique de cette thèse. Celle-ci s'intéresse aux systèmes temps réels modulaires, et plus particulièrement au système IMA. Après avoir décrit les travaux connexes de la littérature, nous avons pu présenter l'originalité de notre démarche d'intégration, qui se focalise sur le respect des contraintes temporelles du système et sur son évolutivité.

Une première présentation, succincte, des contributions de la thèse nous a permis d'introduire les principales étapes de la méthodologie d'allocation temporelle présentée dans ce mémoire. Les grandes étapes de la méthodologie proposée dans cette thèse peuvent se représenter schématiquement par la Figure 2.15.

Dans cette figure, on retrouve les étapes suivantes, où chaque étape est en fait une contribution de cette thèse :

- La première étape de notre méthodologie est la définition de l'ensemble des allocations temporelles admissibles, qui respectent les contraintes applicatives et les contraintes temps réel, connaissant l'allocation spatiale des fonctions embarquées sur l'architecture cible.

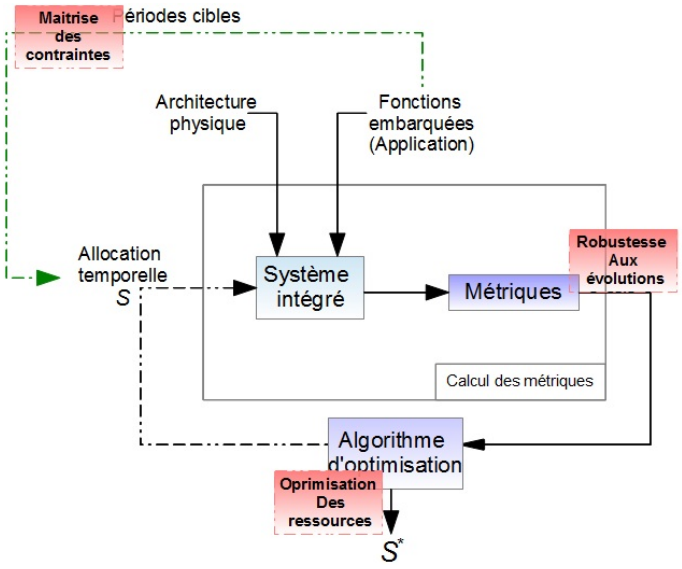


FIGURE 2.15 – Méthodologie d'allocation temporelle pour l'IMA.

- La seconde étape concerne la définition et le calcul des métriques de performance. Ces métriques ont été conçues dans un soucis d'évolutivité. Elles s'intéressent à la charge des calculateurs et au potentiel d'évolution de la marge réseau.
- La troisième étape est l'optimisation multicritère qui extrait l'ensemble des allocations qui optimisent conjointement les métriques antagonistes définies à l'étape précédente. Notre contribution réside en la définition de deux heuristiques, l'une exacte et l'autre approchée, permettant de résoudre des problèmes de taille moyenne à grande.

Cette méthodologie a été mise en place pour garantir les objectifs suivants (représentés en rouge sur la Figure 2.15) :

La maîtrise des contraintes temporelles de bout-en-bout. Cet objectif intègre les contraintes temporelles dès les premières phases de conception du système. Il est fondamental de maîtriser le temps de réponse des fonctions embarquées communicantes au vu de leur criticité dans l'avionique et l'automobile. Aussi, toutes les étapes d'intégration du système doivent prendre en compte des contraintes temporelles strictes pour ne fournir que des solutions fiables à l'intégrateur.

La conception d'un système modulaire évolutif. Il est économiquement très intéressant de pouvoir faire évoluer simplement un système aussi complexe pour le voir supporter de nouvelles fonctions embarquées communicantes. Pour cela, il faut être capable de rajouter des fonctions sur les calculateurs et des flux sur le réseau sans violer les contraintes temporelles de bout-en-bout déjà validées. Si la conception originelle offre suffisament de marge sur les calculateurs et sur le

dimensionnement du réseau, il sera bien plus aisé de faire évoluer le système. Il suffira de rajouter la fonctionnalité voulue sur un des calculateurs peu chargés du système. Il faudra ensuite calculer les performances réseau dans ces nouvelles conditions pour vérifier que les contraintes temps réel sont toujours vérifiées pour l'ensemble des flux du réseau.

L'optimisation des ressources IMA. Nous voulons définir des heuristiques d'optimisation capables d'extraire des allocations qui offrent des compromis entre la charge de calcul du système et les performances réseau. L'optimisation des ressources IMA est ici une problématique complexe, multicritère et sous contraintes. Elle est fondamentale pour résoudre notre problème, mais repose sur une combinatoire élevée. Il faut proposer des heuristiques, exactes ou approchées, permettant de résoudre ce problème pour des systèmes de taille moyenne à grande.

Le reste du manuscrit va maintenant clarifier toutes ces étapes et contributions, en commançant par présenter le problème d'allocation temporelle au Chapitre 3.

3 Formulation du problème d'allocation temporelle en IMA

Dans ce chapitre, nous présentons et formalisons le problème de l'allocation temporelle qui représente une étape cruciale de l'intégration d'un système IMA. Afin de garantir le comportement temps-réel du système, nous considérons deux contraintes sur les délais de bout-en-bout : la contrainte de fraicheur des messages dans une communication et la garantie de la réception des mises à jour des messages. Ces deux contraintes serons explicitées dans la suite du chapitre.

Dans la littérature, très peu de travaux ont abordé l'aide à l'intégration par l'allocation temporelle. Lauer et al. [52] proposent une modélisation pour une plateforme IMA basée sur des automates temporisés. Le but est de réaliser une analyse des délais fonctionnels de bout-en-bout et de vérifier les contraintes de fraicheur ainsi que les mécanismes d'échange de données entre applications. Martin et Fraboul [5][53] présentent une modélisation d'un système IMA en trois niveaux différents (application, architecture, intégration) pour la validation des performances prévisionnelles d'une architecture IMA.

Dans ce chapitre, nous exprimons deux propriétés [54], qui, si elles sont respectées, permettent de garantir la transmission des données critiques et les contraintes temporelles du système. Nous utilisons ensuite une heuristique de séquencement *Least Loaded* des partitions inspirée de [30] pour obtenir des MAFs valides et ordonnançables. Les MAFs ainsi retenues représentent les MAFs candidates à l'allocation temporelle. Nous en déduisons finalement une formulation du problème d'allocation temporelle sous la forme d'un problème d'optimisation discret. La définition du problème d'allocation sur le modèle IMA choisi avec la prise en comptes de contraintes temps-réelles et système, ainsi que le calcul des ordonnancements valides représente la première contribution de cette thèse.

La Figure 3.1 représente l'exemple applicatif sur lequel les contributions de ce chapitre sont illustrées. L'exemple illustre une application avionique répartie sur quatre CPMs interconnectés par un réseau AFDX. Chaque communication est matérialisée par les liens en noir et représente un VL AFDX. Nous considérons que l'application considérée représente une partie d'un système IMA plus large, qui inclut d'autres CPMs et d'autres applications avioniques.

La Section 3.1 décrit le modèlisation de l'IMA utilisée dans ces travaux, ainsi que le détail des paramètres de l'exemple étudié. La Section 3.3 propose la formulation du problème de l'allocation

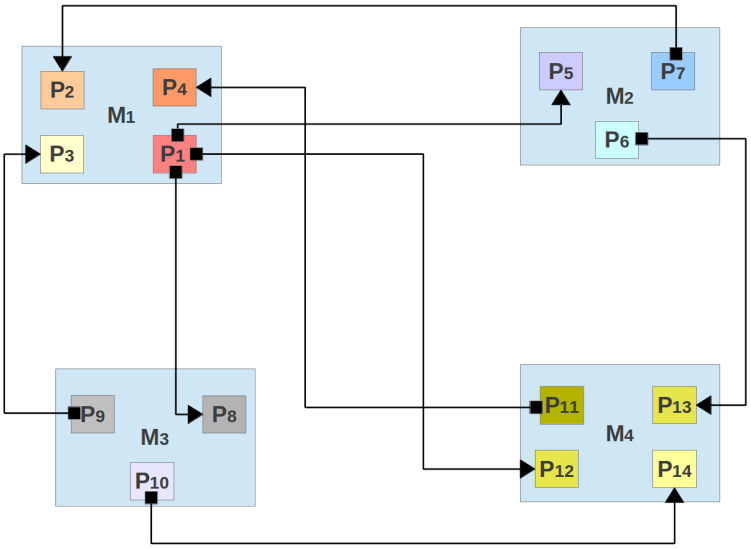


FIGURE 3.1 – Exemple d'une architecture IMA à 4 CPMs

temporelle, les contraintes du problème ainsi que la présentation de la première contribution des travaux de cette thèse. Cette Section se conclut par la formulation du problème d'optimisation sur l'ensemble discret des allocations temporelles candidates.

3.1 Modélisation d'un système IMA

Danc cette Section, nous présentons la modélication d'un système IMA utilisée dans cette these. Nous l'avons choisie de façon à pouvoir agir sur les paramètres qui influent sur les performances temporelles du système. Pour cela, nous avons suivi les recommandations de Franck Martin [6] qui indique un certain nombre d'éléments-clés pour une bonne modélisation de système:

- le choix du niveau d'abstraction pour la description du système,
- le choix des caractéristiques du système à modéliser,
- le choix des performances à mettre en exergue avec le modèle.

Nous considérons dans cette thèse qu'une application avionique est représentée par un ensemble $\mathcal{P} = \{P_1, ..., P_n\}$ de n partitions distribuées et ordonnancées sur un ensemble de m modules IMA $\mathcal{M} = \{M_1, ..., M_m\}$. Nous ne nous intéressons qu'aux communications inter-modules, c.à.d qui traversent le réseau avionique AFDX. Les communications entre des partitions hébergées dans un même module (i.e. intra-modules) ne sont pas prises en compte dans cette étude car elles ne sont pas soumises à la variabilité temporelle induite par le réseau AFDX.

3.1.1 Modèle d'un CPM

Un système IMA est une architecture avionique distribuée où les CMPs s'exécutent de manière asynchrone. Dans ce cas, nous rappelons qu'il n'existe pas d'horloge globale et que chaque CPM suit sa propre horloge locale. De plus, chaque CPM démarre à une date différente à l'initialisation. Pour modéliser le système distribué IMA dans cette thèse, nous faisons les hypothèses suivantes:

- L'horloge de référence est donnée par l'horloge locale du CPM qui démarre en premier.
- La date de démarrage d'un CPM M_ℓ par rapport à la date de démarrage du premier CPM est modélisée par son *offset* ϕ_ℓ exprimé en millisecondes.
- La dérive relative des horloges des CPMs est négligée. Bien entendu, une analyse ultérieure pourrait prendre en compte l'impact de ces dérives sur les allocations temporelles calculées par la suite.

3.1.2 Modèle d'une partition

Un module comporte N_p fenêtres temporelles appelées *partitions* qui s'exécutent de façon périodique selon un ordonnancement défini par la MAF du module. Chaque partition P_i héberge un ensemble de tâches et est caractérisée par sa période T_i et par sa durée d'exécution pire cas b_i. Nous supposons dans cette thèse qu'une seule tâche s'exécute par partition.

Une ou plusieurs partitions de l'ensemble $\mathcal{P}$ sont ordonnancées sur un module. Le résultat de cet ordonnancement hors ligne produit une date de démarrage t_i de la partition relativement à la date de démarrage du module, soit ϕ_ℓ. Ainsi, nous pouvons calculer la date absolue d_n de la $n^{iéme}$ activation périodique de la partition P_i par la relation $d_n = \phi_\ell + t_i + n \times T_i$.

On suppose aussi que toutes les données sont lues par une partition au moment de son activation. De même toutes les données émises sont écrites sur les ports de sortie à la fin de son activation, une fois toutes les opérations réalisées. Ainsi, un donnée qui arrive sur le module pendant l'exécution de la partition ne pourra pas être lue, elle devra attendre l'activitation suivante de la partition.

3.1.3 Modèle de communication

Les partitions distribuées sur une architecture IMA collaborent pour la réalisation des différentes fonctions par l'échange de messages. Une communication $Com_{i,j}$ entre une partition source P_i dans un module M_ℓ et une partition destination P_j dans un module M_k est définie par un canal logique APEX.

Un canal de communication APEX peut fonctionner selon deux modes: le mode *sampling*, où la nouvelle donnée à transmettre remplace la précédente (celle-ci a été consommée par sa destination), ou le mode *queueing* pour lequel les données sont enregistrées dans une mémoire tampon dans leur ordre d'arrivée. Nous considérons dans cette thèse les communications APEX en mode *sampling*. Si la communication est transmise sur un lien virtuel AFDX, le BAG doit être défini dans le modèle de communication.

3.1.4 Paramètres applicatifs de l'exemple étudié

Un certain nombre de paramètres sont imposés par les applications avioniques à deployer. Leurs valeurs sont des données du problème d'intégration. Parmi ces paramètres, il y a la période des partitions qui émettent des données (i.e. les partitions source). Il y a également les caractéristiques des canaux de communication (tel que le BAG en AFDX) et la durée des partitions source et destination.

Le Tableau 3.1 présente le paramétrage de l'exemple de la Figure 3.1. Chaquel ligne donne le paramétrage relatif à une partition source : la périodicité de la partition source, l'identifiant de ses partitions destination et la valeur du BAG alloué à la communication issue de cette source.

	Période T_i (ms)	Destinations	BAG (ms)
P_1	120	P_5, P_8, P_{12}	64
P_6	60	P_{13}	32
P_7	60	P_2	32
P_9	60	P_3	32
P_{10}	60	P_{14}	32
P_{11}	40	P_4	32

TABLEAU 3.1 – Paramètres applicatifs des partitions sourcepartitions source

La durée d'exécution des partitions composant l'application avionique est donnée dans le Tableau 3.2.

	Durée d'exécution b_i (ms)
P_1, P_4	5
P_2, P_3, P_{11},P_{12}, P_{13}, P_{14}	10
P_5, P_6,P_7, P_8,P_9,P_{10}	15

TABLEAU 3.2 – Durées d'exécution des partitions de l'exemple

3.2 Analyse des délais de bout-en-bout

Dans cette section, nous procédons à une analyse des latences qui composent le délai de communication de bout-en-bout entre deux partitions communicantes distantes. Nous nous appuyons sur la modélication IMA introduite précédement. Cette analyse permet de déterminer les paramètres de l'allocation temporelle ainsi que les latences qui font varier les délais de bout-en-bout. Nous définissons deux délais de bout-en-bout: *le délai fonctionnel* et *le délai de communication*, le délai de communication étant inclus dans le délai fonctionnel.

3.2.1 Délai fonctionnel de bout-en-bout

Le délai fonctionnel de bout-en-bout [52] est illustré dans la Figure 3.2. Dans cette Figure, nous considérons une communication entre une partition source P_i allouée sur le CPM M_ℓ, et la partition destination P_j allouée sur le CPM M_k. La partition source émet périodiquement une occurrence du flux de message msg. La n^{ieme} occurrence émise du flux msg est notée msg_n.

Le délai fonctionnel est défini par la durée qui sépare la date de création d'une donnée d_{msg_n} et la date où cette donnée a été traitée par la partition destination. Cette donnée est par exemple créée par un capteur et parvient au module à la date d_{msg_n} sur un port dédié. Cette donnée ne peut être lue par la partition source qu'une fois celle-ci active à la date d_n.

Nous supposons ici que la lecture des données disponibles sur tous les ports du module s'effectue en début d'activation de la partition. Les données sont produites et envoyées par l'unique tâche d'une partition une fois les traitements terminés, c.à.d à la fin de l'exécution de la partition.

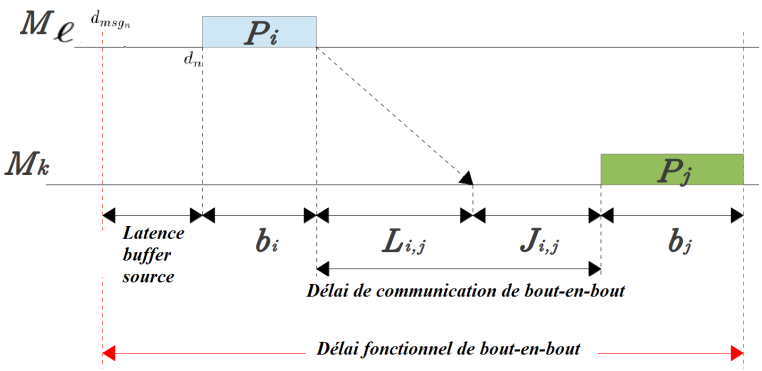


FIGURE 3.2 – Délais de bout-en-bout fonctionnel et de communication

Le délai fonctionnel se décompose en une latence de lecture dans le buffer source, de la durée d'exécution de P_i et P_j, du délai de communication dans le réseau et de la latence de lecture sur le module de destination.

Latence buffer source

Une occurrence msg_n du flux de message msg est générée par un capteur connecté au module M_ℓ. Cette occurrence doit attendre l'activation de la partition P_i pour être lue dans le buffer source. Cette latence est représentée sur la Figure 3.2 par la différence entre la date de la prochaine activation de P_i, d_n dans notre cas, et la date de la génération de la donnée, notée d_{msg_n}.

Durée d'exécution de la source

La partition P_i est activée pour une durée de temps b_i durant laquelle l'occurrence msg_n demeure au niveau de P_i, jusqu'à ce qu'elle soit transmise sur le port APEX de sortie. La latence d'exécution est donc donnée par b_i.

Latence réseau

La latence réseau est la durée entre le moment où l'occurrence d'un message msg_n quitte le port APEX en sortie au niveau de la partition source P_i, et le moment où cette occurrence msg_n arrive sur le port APEX en entrée du module destination M_k.

Chaque occurrence du flux de messages msg peut subir des délais variables dans le réseau. De ce fait, comme les réseaux embarqués sont conçus pour garantir un délai borné, nous supposons que le délai de communication, noté $L_{i,j}$, entre deux partitions distantes P_i et P_j, appartient à un intervalle borné $[L_{i,j}^{min}, L_{i,j}^{max}]$, où $L_{i,j}^{min}$ est la latence réseau meilleur cas observée et $L_{i,j}^{max}$ est la latence réseau pire cas observée sur la communication $Com_{i,j}$. La latence dans le réseau est influencée par différents facteurs. Nous pouvons citer le type de réseau, sa topologie, sa charge et son paramétrage. La latence est bornée pour chaque communication en concordance avec les études connues sur le calcul pire cas réalisé dans le contexte des réseaux embarqués avioniques (cf. [35][37] et les références détaillées dans la Section 2.4).

Latence à la destination

Une fois msg_n reçu au niveau du CPM destination, msg_n doit attendre l'activation de sa cible. La latence à la destination, notée $J_{i,j}$ est définie par la différence entre la date de la lecture de msg_n par P_j et la date de son arrivée sur le port APEX du module M_k. Les occurrences de msg qui sont reçues à la partition destination ne constituent plus un flux périodique. Ceci est la conséquence de la variabilité dans le réseau de communication et des déphasages de démarrage des CPMs communicants.

Durée d'exécution à la destination

Une fois msg_n lu par la partition destination P_j, il est utilisé pour les traitements de la tâche destination pour une durée d'exécution b_j. Une fois la tache hébergée par P_j terminée, nous supposons que la communication fonctionnelle se termine.

3.2.2 Délai de communication de bout-en-bout

Dans ces travaux, il a été choisi de travailler avec les seules composantes du délai fonctionnel de bout-en-bout qui soient directement impactées par l'allocation temporelle réalisée dans les phases d'intégration. En effet, le choix des périodes d'exécution des partitions n'impacte pas les durée d'exécution b_i et b_j. Seuls $L_{i,j}$ et $J_{i,j}$ sont conditionnés par le choix de T_i et T_j.

Nous avons donc défini le *délai de communication de bout-en-bout* $E2E_{i,j}$, comme la somme de la latence réseau $L_{i,j}$ et de la latence à la destination $J_{i,j}$. Formellement:

$$E2E_{i,j} = L_{i,j} + J_{i,j}$$

$E2E_{i,j}$ représente le délai entre l'émission de msg_n sur le port APEX d'émission de P_i et sa lecture effective par la partition destination P_j sur le module destination.

Comme il a été précisé précédemment, une partition source génère périodiquement une occurrence de message à la fin de chacune de ses exécutions. Ces occurrences sont transmises à au moins une partition destination à travers un lien virtuel AFDX (VL). Un VL AFDX est caractérisé par son BAG. Dans le tableau 3.1, les BAGs des liens virtuels sont fixés de façon à ne pas dépasser la période de leurs partitions source. Ainsi, nous pouvons assurer qu'une occurrence de message n'est pas écartée à cause du BAG correspondant au VL qui la transporte.

La tableau 3.3 détaille les bornes sur la latences réseaux des communications de la Figure 3.1..

	$L_{i,j}^{min}$ (ms)	$L_{i,j}^{max}$ (ms)
$Com_{1,5}$	2	12
$Com_{1,8}$	3	15
$Com_{1,12}$	1	6
$Com_{6,13}$	1	6
$Com_{7,2}$	2	12
$Com_{9,3}$	4	20
$Com_{10,14}$	2	10
$Com_{11,4}$	1	5

TABLEAU 3.3 – Bornes sur les latences réseau

Dans ce qui suit, nous procédons à la formulation du problème d'allocation temporelle en intégrant les contraintes applicatives temps-réelles et les contraintes technologiques propres au système IMA.

3.3 Formulation du problème

Dans ce qui suit, le délai de communication de bout-en-bout $E2E_{i,j}$ est un élément central pour la maîtrise de l'allocation temporelle. Dans la contribution proposée, nous nous appuyons sur le calcul de la période de la partition destination qui garantit la validité des contraintes de bout-en-bout et assure la qualité de service des communications.

3.3.1 Variables du problème

Les variables du problème d'allocation temporelles sont *les périodes des partitions destination.* Comme présenté ci-avant, les périodes des partitions source sont fixées au niveaux applicatif. Il est donc nécessaire de déterminer les T_j des partitions destination. Dans l'exemple de la Figure 3.1, il est nécessaire de calculer les périodicités des partitions destination P_2, P_3, P_4, hébergées par le module M_1, des partitions destination P_{12}, P_{13} et P_{14} hébergées dans le module M_4 et des partitions P_5 et P_8 hébergées respectivement par M_2 et M_3.

Ce choix n'est pas trivial car il impacte directement la valeur de la latence $J_{i,j}$. Pour que le système présente un comportement temps-réel, il faut borner le délai de communication et éviter les perte de données. Ceci se traduit par la prise en compte de contraintes temporelles sur le choix des périodes T_j des partition réceptrices. De plus, il existe un contrainte d'ordonnaçabilité des partitions sur les modules et des contraintes systèmes liées à l'IMA.

Dans ce qui suit, nous définissons les contraintes à prendre en compte pour à la fois garantir le comportement applicatif temps-réel du système, et d'autre part être conforme aux spécificités de l'IMA.

3.3.2 Contraintes applicatives temps réel

Pour maitriser la variabilité des latences qui composent le délai de communication de bout-en-bout, deux contraintes sur les périodes des partitions destinations sont introduites afin d'assurer des transferts de données fiables.

Une occurrence de message msg_n peut être perdue si un des deux cas suivants se présente :

- msg_n connaît un délai de communication de bout-en-bout supérieur à une durée maximale autorisée,
- msg_n est écrasée par une occurrence plus récente msg_{n+1} avant d'être lue par la partition destination.

Dans le premier cas, la donnée arrive trop tard et n'est plus valide. Dans le second cas, la donnée est perdue car écrasée avant d'avoir été lue par la partition en réception. Deux contraintes sur la periode en réception sont définies ci-après pour éviter ces cas de figure.

3.3.2.1 Contrainte sur la fraîcheur d'une donnée

Afin de garantir la fraîcheur et la cohérence des données dans une communication périodique $Com_{i,j}$, les messages échangés sont contraints par une durée de vie maximale appelée *paramètre de fraîcheur* : $FP_{i,j}$. Si le délai de communication $E2E_{i,j}$ est supérieur à $FP_{i,j}$, le message est détruit.

Un message est donc invalidé s'il est lu trop tard par la partition destination P_j, ce qui est illustré dans la Figure 3.3. On rappelle que les données ne peuvent être lues par une partition que si elles sont présentes dans le port APEX du module à l'activation de la partition. Si une donnée arrive pendant l'exécution d'une partition, elle ne peut être lue. Dans la Figure 3.3, l'occurrence msg_n arrive après le démarrage d'une exécution de la partition P_j, qui ne peut donc la lire. Il faut attendre l'exécution suivante de P_j pour qu'elle soit lue. On voit bien que la périodicité de P_j impacte le respect de la contrainte de fraicheur : plus la pédiode T_j est petite, plus il y a de chances pour que la contrainte de fraicheur soit respectée.

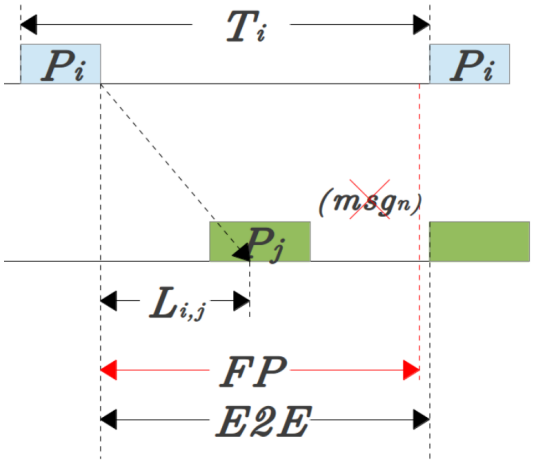


FIGURE 3.3 – Violation du paramètre de fraicheur dans une communication

Au pire cas, l'occurrence du message arrive juste après l'activation de la partition. Elle doit donc attendre une période complète ($J_{i,j} = T_j$) avant d'être consommée par P_j.

Si la Propriété 1 définie ci-après est vérifiée par la période T_j, la fraîcheur des messages transmis pour une communication donnée est assurée :

Propriété 1 *Tous les messages de $Com_{i,j}$ respectent leur paramètre de fraîcheur $FP_{i,j}$ si T_j vérifie :*

$$T_j \leq FP_{i,j} - L_{i,j}^{max} \tag{3.1}$$

Démonstration. Afin de garantir la contrainte de fraîcheur $FP_{i,j}$ associée à la communication

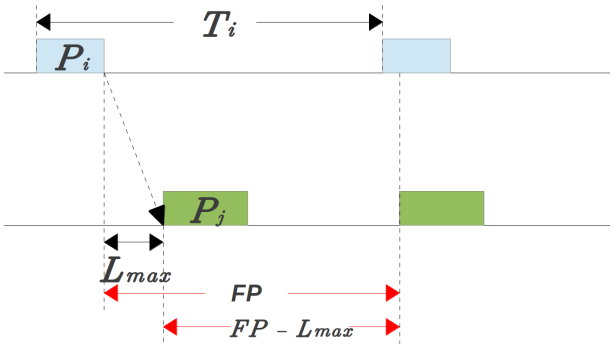


FIGURE 3.4 – Application de la propriété 1

$Com_{i,j}$, le délai de communication de bout-en-bout d'un message msg_n, noté $E2E_n$, ne doit pas dépasser $FP_{i,j}$:

$$E2E_n \leq FP_{i,j}$$

Par définition, nous avons :

$$L_{i,j} + J_{i,j} \leq FP_{i,j}$$

Deux phénomènes concomitants mènent au pires cas possible :

- la latence dans le réseau $L_{i,j}$ est égale à la latence pire cas $L_{i,j}^{max}$ pour la communication $Com_{i,j}$,
- msg_n subit une latence à la destination égale à une période entière de T_j.

Dans ce cas, nous avons :

$$L_{i,j}^{max} + T_j \leq FP_{i,j}$$

À partir de l'équation 3.1, nous pouvons déduire que si la période de la destination dépasse $FP_{i,j} - L_{i,j}^{max}$, la contrainte de fraîcheur n'est pas respectée pour msg_n. En d'autres termes, les valeurs valides pour la période T_j de la partition destination vérifier la propriété 1 :

$$T_j \leq FP_{i,j} - L_{i,j}^{max}$$

□

Pour l'exemple de la Figure 3.1, les paramètres de fraicheur des communications sont décrites dans le tableau 3.4.

Note. Si une partition P_j est la destination de K différentes partitions source, sa période T_j doit être ajustée en considérant le paramètre de fraîcheur le plus restrictif parmi les K communications. Dans ce cas précis, *Propriété 1* est conservée en remplaçant le paramètre $FP_{i,j}$ et $L_{i,j}^{max}$

	Paramètre de fraîcheur $FP_{i,j}$ (ms)
$Com_{1,5}$, $Com_{1,8}$, $Com_{1,12}$	100
$Com_{6,13}$,$Com_{7,2}$, $Com_{9,3}$, $Com_{10,14}$	60
$Com_{11,4}$	40

TABLEAU 3.4 – Paramètre de fraîcheurs des communications

comme présentée dans l'équation:

$$T_j \leq \min_{k \in K}(FP_{k,j} - L^{max}_{k,j}) \tag{3.2}$$

3.3.2.2 Contrainte sur la garantie des mises-à-jours

Un flux périodique de messages msg est cohérent si toutes les occurrences sont bien reçues, et ceci dans l'ordre de leur émission. Garantir la cohérence, c'est garantir qu'aucune occurrence msg_n ne soit perdue ou écrasée par une nouvelle occurrence avant d'être consommée par sa cible.

Dans une communication en mode *sampling*, le port APEX du module ne peut stocker qu'une donnée. Ainsi, si une nouvelle donnée arrive sur le port alors que l'ancienne n'a pas été lue par sa partition destination, il y a perte d'une mise-à-jour par écrasement. La Figure 3.5 illustre la perte de mise-à-jour par écrasement. Sur cette figure, l'occurrence msg_n est écrasée par msg_{n+1} car les deux occurrences sont arrivées entre deux activations de la partition P_j.

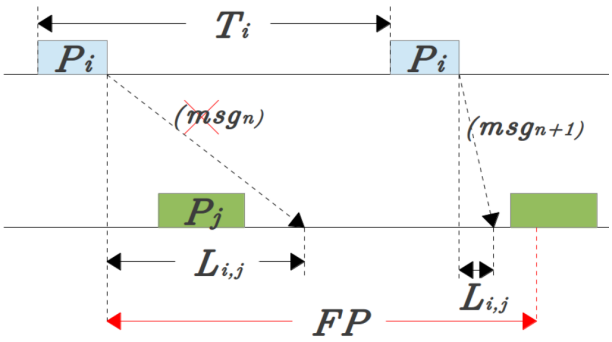


FIGURE 3.5 – Écrasement de mise-à-jour dans une communication

Pour éviter les pertes par écrasement, il faut réduire la période T_j de la partition destination. La Propriété 2 que nous proposons ci-après permet de calculer une borne sur T_j qui garantit l'absence de pertes par écrasement.

Propriété 2 Un message msg_n n'est jamais écrasé par une mise à jour plus récente msg_{n+1} avant sa lecture par la partition destination si :

$$T_j < T_i - (L^{max}_{i,j} - L^{min}_{i,j}) \tag{3.3}$$

Note. Dans les communications en mode *queueing*, une occurrence de message peut aussi être écrasée si la file d'attente est pleine. On rappelle qu'en mode *queuing* il existe une file d'attente au niveau du port APEX capable d'enregistrer N mises-à-jours consécutives. A chaque activation de la partition destination, la donnée la plus ancienne est consommée, laissant une place pour une nouvelle donnée en entrée de file. Si la file est pleine, l'arrivée d'une nouvelle occurrence écrase la dernière valeur reçue par la file. Ce cas survient si deux occurrences sont reçues entre deux activations consécutives de la partition destination, tout comme dans le mode *sampling*. L'étude qui suit est donc valable que l'on travaille en mode *sampling* ou en mode *queuing*.

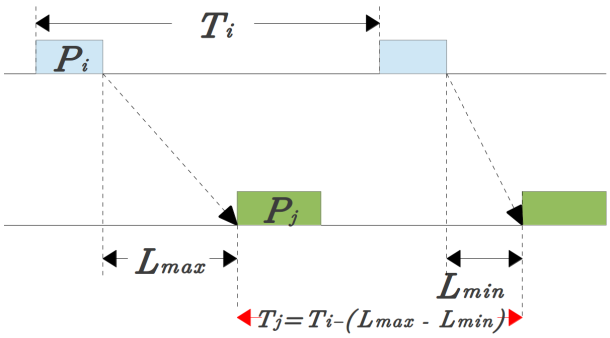


FIGURE 3.6 – Application de la propriété 2

Démonstration. Soient deux occurrences successives du flux de message msg, dénommées msg_n et msg_{n+1}. Elles sont émises par P_i à la fin de deux activations successives comme représenté sur la Figure 3.6. Les occurrences msg_n et msg_{n+1} arrivent à la partition destination P_j aux dates d_n (n^{ieme} activation de P_i) et d_{n+1} ($(n+1)^{ième}$ activation de P_i).

Par définition, ces dates valent :

$$d_n = \phi_\ell + nT_i + b_i + L_n$$

$$d_{n+1} = \phi_\ell + (n+1)T_i + b_i + L_{n+1}$$

où L_n et L_{n+1} représentent les latences réseau subies par msg_n et msg_{n+1} respectivement.

La partition destination P_j peut lire les deux occurrences consécutives si les deux dates d'arrivée sont espacées d'au moins une période d'activation T_j :

$$T_j \leq (d_{n+1} - d_n)$$

En substituant d_{n+1} et d_n par leur définition, il en suit :

$$T_j \leq T_i + L_{n+1} - L_n$$

La période de la destination doit assurer la non-perte des messages pour les cas les plus contraignants. Ce cas se produit pour la plus petite valeur de la différence $d_{n+1} - d_n$, *i.e* la plus petite valeur possible de $T_i + L_{n+1} - L_n$. Cette valeur est obtenue si $L_n = L_{i,j}^{max}$ et $L_{n+1} = L_{i,j}^{min}$ de la communication $Com_{i,j}$. Ainsi:

$$T_j \leq T_i + L_{i,j}^{min} - L_{i,j}^{max}$$
$$\Leftrightarrow T_j \leq T_i - (L_{i,j}^{max} - L_{i,j}^{min})$$

□

Note. Si la partition destination P_j est la destination de K partitions source différentes, la communication la plus contraignante est celle avec la plus petite période d'émission. Ainsi, la propriété 2 est conservée avec la formule qui suit:

$$T_j \leq \min_{k \in K}(T_k - (L_{k,j}^{max} - L_{k,j}^{min})) \tag{3.4}$$

3.3.2.3 Contrainte globale temps-réel

Les Propriétés 1 et 2 définissent des bornes maximales sur les valeurs admissibles d'une partition destination T_j. En fonction de l'architecture et de l'application, il se peut que ce soit la Propriété 1 qui soit la plus restrictive, ou il se peut que ce soit la propriété 2. Il est possible de les combiner pour obtenir une unique contrainte applicative qui garantisse le comportement temps-réel des communications.

Ainsi, la contrainte globale temps-réel qui sera utilisée dans la formulation du problème d'allocation temporelle est la suivante :

$$Tj \leq T_j^{max} \tag{3.5}$$

avec $T_j^{max} = min(FP_{i,j} - L_{i,j}^{max}, T_i - (L_{i,j}^{max} - L_{i,j}^{min}))$. T_j^{max} représente la période maximale admissible pour la partition destination P_j. Compte-tenu de cette contrainte globale, il est maintenant possible de choisir des périodes qui guarantissent la transmission des données en un temps borné. Dans la suite de ce chapitre, nous introduisons des contraintes propres au système IMA qui permettent de sélectionner un ensemble discret de valeurs admissibles des périodes en réception.

3.3.2.4 Exemple illustratif

Nous reprenons ici l'exemple de déploiement de la Figure 3.1. Compte-tenu des différents paramètres des Tableaux 3.1, 3.2, 3.3 et 3.5, il est possible de calculer les valeurs de T_j^{max} pour chaque partition destination. Nous détaillons ici le calcul de T_j^{max} et des propriétés 1 et 2 pour la partition destination P_5:

Calcul de la Propriété 1

$$\begin{aligned} T_5 &\leq FP_{1,5} - L_{max}^{1,5} \\ &\leq 100 - 12 \\ &\leq 88\ ms \end{aligned}$$

Calcul de la Propriété 2

$$\begin{aligned} T_5 &\leq T_1 - (L_{max}^{1,5} - L_{min}^{1,5}) \\ &\leq 120 - (12 - 2) \\ &\leq 110\ ms \end{aligned}$$

Ainsi la période de P_5 doit être au plus égale à T_5^{max}=88 ms. Les résultats de l'application des propriétés 1 et 2 et du calcul de T_j^{max} pour le reste des périodes des partitions destination sont donnés dans le Tableau 3.3.2.4.

	T_j^{max}	Propriété 1	Propriété 2
P_2	48	48	50
P_3	40	40	44
P_4	35	35	36
P_5	88	88	110
P_8	85	85	108
P_{12}	94	94	115
P_{13}	54	54	55
P_{14}	50	50	52

TABLEAU 3.5 – Contraintes sur les périodes des partitions destination

Nous avons des bornes sur les valeurs admissibles de période des partitions destination. La suite de ce chapitre présente notre démarche d'intégration des contraintes propres au système IMA et à l'ordonnancement des partitions sur un module.

3.3.3 Contraintes système IMA

A partir de la contrainte globale temps-réel (3.5), il est possible de définir un ensemble de valeurs admissibles pour les périodes des partitions destination. Pour cela, il faut prendre

en compte les caractéristiques du système IMA. En effet, dans un module IMA, les périodes des partitions sont harmoniques entre elles. Ainsi, il existe un entier k_i, tel que $T_{i+1} = k_i \times T_i$. Par exemple, l'ensemble de partitions de périodes $\{30, 30, 60, 120\}$ est harmonique, avec $k_1 = 1, k_2 = 2, k_3 = 2$.

Ces partitions à périodes harmoniques sont ordonnancées pour créer la *Major Time Frame* (MAF), où partitions source et destination co-habitent. Comme nous l'avons déjà énoncé, la période des partitions source est figée. Il faut donc choisir des périodes destination qui soient harmoniques avec les périodes des partitions source. C'est l'objet de la première partie de cette section.

Une fois ces valeurs harmoniques déterminées, il va falloir, pour chaque module, sélectionner les combinaisons de partitions qui sont ordonnançables. En d'autres termes, il faut vérifier quelles combinaisons permettent de créer des MAFs valides. C'est l'objet de la seconde partie de cette section. On pourra alors définir notre problème d'optimisation des périodes sous contraintes comme un problème d'optimisation discret où les solutions sont représentées par l'ensemble des MAFs admissibles d'un module.

Cette partie a donc pour objectif de déterminer l'ensemble des valeurs que les périodes destination peuvent admettre tout en :

- respectant la contrainte globale (3.5),
- restant harmonique avec les périodes des partitions source,
- guarantissant la création de MAFs valides.

3.3.3.1 Harmonisation des périodes des partitions cibles

Les périodes des partitions source sont fixées et harmoniques entre elles par construction. Il faut donc déterminer les périodes possibles des partitions destination qui respectent cette harmonie. Nous nous plaçons dans le cas où il existe *au moins* une partition source par module. Par exemple, si un module M_1 est composé d'une partition source P_1 de période 120ms, et d'une partition destination P_2 pour laquelle on a T_2^{max} égal à 72ms, T_2 peut prendre plusieurs valeurs harmoniques à 120ms et inférieures à 72ms. Dans ce cas, T_2 peut prendre 60ms ($k = 2$), 40ms ($k = 3$), 30ms ($k = 4$), 20ms ($k = 6$) ou 10ms ($k = 12$) comme valeurs entières.

La Figure 3.7 présente l'algorithme implanté pour calculer les périodes harmoniques admissibles des partitions destination. Dans cet algorithme, nous notons par A_{T_j} l'ensemble des périodes d'une partition P_j harmoniques avec les périodes sources du module. L'ensemble des MAFs harmoniques générées par l'algorithme est dénommé MAF_h et est généré par le produit cartésien des périodes candidates A_{T_j} et des périodes source du module.

L'algorithme de la Figure 3.7 se décompose en trois grandes étapes. La première étape calcule l'ensemble A_{T_j} de chaque partition destination. Si le module est composé de plusieurs partitions source, il faut prendre en compte le plus grand commun diviseur ($pgcd$) des périodes source T_c pour le calcul des périodes harmoniques.

La second étape crée le produit cartésien des ensembles A_{T_j}. Les MAFs obtenues par produit

cartésien ne sont pas forcément harmoniques. Il faut donc, dans la troisième étape, écarter les MAFs non-harmoniques pour obtenir l'ensemble MAF_h.

En appliquant l'algorithme 3.7 sur le module M_2 de la Figure 3.1, on obtient le résultat suivant: il y a deux partitions source P_6 et P_7 ayant toutes les deux une période de 60ms et une durée de 15ms. Seule la partition P_5 de durée d'exécution 15ms est une partition destination. Précédemment, nous avons calculé que T_5^{max} vaut 88ms. Nous obtenons alors l'ensemble $A_{T_j} = (60, 30, 20)$. L'ensemble MAF_h du module M_2 de notre exemple comporte trois MAFs harmoniques : $MAF_{2,1} = (T_5 = 60, T_6 = 60, T_7 = 60)$; $MAF_{2,2} = (30, 60, 60)$ et $MAF_{2,3} = (20, 60, 60)$. Connaissant les durées d'exécution ($b_5 = b_6 = b_7 = 15$ms), il faut maintenant s'assurer que ces MAFs sont bien ordonnançables sur M_2. Un autre exemple de maf, composé de trois partitions sources dont les périodes respectives sont de 120, 60, et 20, et d'une partition destination dont $T_j^{max} = 72$. Dans ce cas, $A_{T_j} = (60, 30, 20)$. Nous ne pouvons pas construire de maf harmonique avec la période destination égale à 30 de l'ensemble A_{T_j} car celle-ci n'est pas harmonique avec la partition destination de période 20. La maf (120,60,20,30) dans MAF_c n'est donc par ajoutée à l'ensemble MAF_h.

3.3.3.2 Construction d'un ordonnancement valide

Nous possédons maintenant un ensemble de MAF harmonique par module candidates pour l'ordonnancement. Nous adopterons la notation suivante dans le reste de la thèse : $MAF_{i,j}$ est la MAF candidate numéro j du module M_i.

Dans les problèmes d'ordonnancement hors ligne, un calcul de la séquence d'exécution est réalisé. Étant donné que les périodes sont harmoniques et les partitions strictement périodiques, il s'agit d'assigner les dates de début d'exécution pour la première occurrence de chaque partition d'un module. Cette séquence d'exécution est répétée de manière cyclique. Dans un système IMA, une partition ne peut être préemptée par une autre partition afin d'assurer l'isolement temporel et spatial. Il faut donc vérifier que les MAFs sont ordonnançables sans préemption.

Une première condition suffisante est que la charge du module ne soit pas supérieure à 1 :

$$U = \sum_{i=1}^{N_P} \frac{b_i}{T_i} \leq 1 \tag{3.6}$$

Cette première condition suffisante nous permet d'écarter la MAF candidate $MAF_{2,3} = (20, 60, 60)$ car sa charge est de 75/60.

Vérifier l'ordonnançabilité d'un jeu de tâches en non-préemptif est un problème difficile comme nous l'avons expliqué au chapitre précédent. Il n'existe pas de condition nécessaire et suffisante pour décider de l'ordonnançabilité d'un jeu de tâches. La seule condition nécessaire connue de Jeffay et al.[18] est très restrictive. Elle n'admet que des MAFs où toutes les partitions peuvent s'exécuter pendant la durée de la plus petite période. Pour décider plus finement, il faut utiliser une heuristique de construction de la séquence d'exécution des partitions.

```
// Calcul de la plus petite période commune T_c du module M_ℓ
T_c = pgcd(T_i des partitions source ∈ M_ℓ)

// Calcul des valeurs harmoniques possibles de chaque partition P_j
Pour chaque partition cible P_j ∈ M_ℓ :
     // A_Tj contient les harmoniques possibles de P_j
     A_Tj = ∅ ; sol_c = 0 ; x = 1 ;
     Si ((b_j ≤ T_c) ET (T_c < T_j^max))
          sol_c = ⌊T_j^max / T_c⌋ × T_c
          T_c = sol_c
     FinSi
     Faire
          Si ( pgcd(T_c, sol_c) = sol_c et ⌊sol_c⌋ × x = T_c et sol_c ≤ T_j^max )
               Ajouter(A_Tj , sol_c)
          Fsi
          x++
          sol_c= T_c / x
     Tant que (sol_c ≥ b_j)
FinPour

// Calcul du produit cartésien des A_Tj .
// MAF_c: résultat du produit cartésien des A_Tj
MAF_c = ∅ ;
// Calcul du produit cartésien pour générer MAF_c
MAF_c = ∏_j A_Tj X ∏_i T_i ;

// Sélection des MAFs harmoniques de M_ℓ
// MAF_h: ensemble des MAFs à périodes harmonique
MAF_h = ∅;
Pour chaque maf ∈ MAF_c :
Ordonner les partitions par ordre croissant des périodes d'exécution
     si pour toute partition P_k dans maf, ppcm(T_k, T_k+1) == max(T_k, T_k+1) :
//maf a des partitions à périodes harmoniques
          Ajouter(MAF_c, maf)
     sinon
          // maf n'est pas harmonique
```

FIGURE 3.7 – Algorithme pour la construction des MAFs avec des périodes harmoniques dans un module

Une recherche exhaustive n'est malheureusement pas envisageable de par l'explosion combinatoire qui en découle. Ainsi, nous avons décidé d'implanter une heuristique non-optimale. Si cette heuristique trouve une séquence valide, la MAF est réalisable. Par contre, il se peut qu'elle ne trouve pas de séquence valide alors que le jeu de tâches est ordonnançable en réalité. Nous avons choisi d'implanter une solution dérivée de l'algorithme *Least Loaded* de Grenier et al. [30], présenté en détails au Chapitre 2.3.4.

Algorithme Least Loaded pour l'allocation valide des MAFs

Dans la même philosophie que l'algorithme LL, nous recourons à une heuristique de séquencement des partitions pour un module de calcul. Les partitions sont d'abord triées par ordre croissant des périodes d'exécution. A chaque itération, l'algorithme construit le séquencement en positionnant une partition supplémentaire dans une MAF (vide au départ). Voici les principales caractéristiques de l'algorithme décrit dans la Figure 3.8 :

1. la longueur de la MAF (égale à l'hyperpériode des partitions) est découpée en un nombre de slots de taille identique, où chaque slot a une taille égale à la période de la plus petite période des partitions du module.
2. À chaque étape, la partition considérée est allouée dans le slot le moins chargé. La charge d'un slot est égale à la somme des durées d'exécution des partitions déjà placées dans le slot. La valeur de la charge d'un slot est mise à jour lorsqu'une partition est placée dans le slot en l'incrémentant de sa durée d'exécution.
3. À partir du slot sélectionné, le reste des occurrences de la partition qui doivent être exécutées dans l'hyperpériode sont placées avec le respect de la période de celles-ci. Si le placement n'est pas possible, c.à.d la charge d'un slot dépasse la charge maximale égale à la longueur du slot, l'ordonnancement est considéré comme invalide.

Cette condition de non-faisabilité peut être restrictive, mais permet de proposer des ordonnancements de MAFs faisables pour les modules de calcul. En effet, cet algorithme peut écarter des solutions réalisables du fait de sa condition restrictive, mais celle-ci est plus lâche que la condition nécessaire et suffisante de Jeffay et al. [18]. L'algorithme est décrit dans la Figure 3.8.

La Figure 3.9 montre le séquencement obtenu en appliquant l'heuristique LL à l'ensemble des partitions T_1(120,5),T_2(40,10),T_3(40,10),T_4(20,5). Pour $T_{slot} = 20$ et $T_{MAF} = 120$, cela donne une table d'ordonnancement de la table 3.6 avec 6 slots, avec une MAF de 120 ms et un slot de 20 ms. Le tableau 3.6 liste les charges des slots de la MAF à la fin de l'exécution de l'algorithme LL.

Dans ce qui suit, nous listons l'ensemble des allocations temporelles possibles pour chaque CPM avec l'heuristique LL et la condition 3.6.

A partir des T_j^{max} calculées dans le tableau 3.3.2.4, nous pouvons dériver les périodes qui sont ordonnançables (selon l'heuristique LL et la condition 3.6). Ainsi, les allocations temporelles faisables sont:

L'ensemble de $P_i \in \mathcal{P}$ et $P_i \in M_\ell$, longueur de la MAF T_{MAF}, longueur du slot T_{slot}
1- Ordonner les $P_i \in M_\ell$ par ordre croissant de leurs périodes d'exécution
2- T_{slot}= $\min_{i=1}^{i=n}(P_i)$, T_{MAF} = $\max_{i=1}^{i=n}(P_i)$
3- Pour chaque $P_i \in M_\ell$:
    (a) Chercher le slot k le moins chargé dans les $\frac{T_{MAF}}{T_{T_{slot}}}$
    (b) Placer P_i tous les $\frac{T_i}{T_{slot}}$ en commençant par le slot k
    (c) Mettre à jour la charge des slots occupés en augmentant de b_i
4-Tester la charge des slots nouvellement occupés :
    (a) Si la charge d'un des slots dépasse T_{slot}; alors le problème n'est pas ordonnaçable

FIGURE 3.8 – Algorithme de séquencement des partitions dans une MAF avec *Least Loaded*

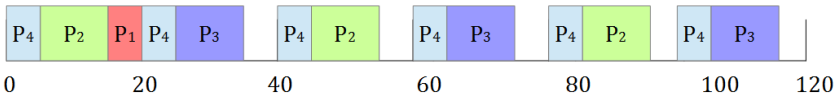


FIGURE 3.9 – Exemple du séquencement d'une MAF

M_1: pour (T_1,T_2,T_3,T_4)
- $MAF_{1,1}$ = (120,40,40,20)
- $MAF_{1,2}$ = (120,30,30,30)

M_2: pour (T_5,T_6,T_7)
- $MAF_{2,1}$ = (60,60,60)
- $MAF_{2,2}$ = (30,60,60)

M_3: pour (T_8,T_9,T_{10})
- $MAF_{3,1}$ = (60,60,60)
- $MAF_{3,2}$ = (30,60,60)

M_4: pour (T_{11},T_{12},T_{13},T_{14})
- $MAF_{4,1}$ = (40,80,40,40)
- $MAF_{4,2}$ = (40,40,40,40)

La prise en compte des contraintes temps-réél et système nous conduit naturellement à une formulation discrète du problème d'allocation temporelle traité dans cette thèse. La recherche des périodes des partitions destination équivaut à déterminer la MAF des modules dans un ensemble discret de MAFs admissibles. La formulation du problème que nous proposons dans cette thèse est présentée ci-après.

	Charge du slot(ms)
$Slot_1$	20
$Slot_2$	15
$Slot_3$	15
$Slot_4$	15
$Slot_5$	15
$Slot_6$	15

TABLEAU 3.6 – Tableau du séquencement pour la Figure 3.9

3.4 Formulation discrète de l'allocation temporelle

Nous constatons que pour un seul module CPM, il est possible d'avoir plusieurs MAFs admissibles qui respectent les contraintes temporelles et système. La question que nous pouvons nous poser est la suivante: *comment comparer ces MAFs admissibles au niveau d'un module ?*. Il serait intéressant de pouvoir départager et comparer les MAFs d'un même CPM afin de choisir la plus intéressante, c'est-à-dire celle qui propose les meilleures performances à l'intégrateur.

Dans notre exemple à quatre modules, nous avons déterminé deux MAFs admissibles par module. Il existe donc sur ce petit exemple 16 configurations admissibles du système complet. La même question se pose pour l'allocation temporelle du système dans sa totalité : quelle est l'allocation temporelle du système la plus intéressante ? Quelle configuration propose les performances les plus intéressantes pour l'intégrateur ? La Figure 3.10 présente deux allocations temporelles admissibles pour l'exemple 3.1.

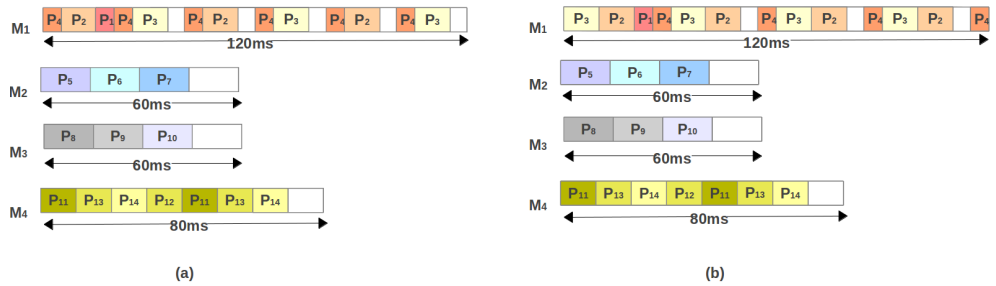


FIGURE 3.10 – Deux allocations temporelles possibles

Ces différentes interrogations font ressentir le besoin d'introduire des métriques capables d'évaluer et de quantifier les allocations temporelles selon différents critères. Ces métriques permettent la comparaison des allocations temporelles et constituent une aide fondamentale à la prise de décision dans le processus d'intégration.

Le problème d'allocation temporelle qui est un problème d'optimisation sous contrainte, a été formulé dans ce chapitre en un problème d'optimisation combinatoire grâce à la prise en compte des contraintes applicatives temps-réél et système. Dans la formulation que nous proposons,

l'ensemble des solutions est l'ensemble discret des configurations admissibles des MAFs candidates aux modules. Ces MAFs respectent, par construction, les contraintes du problème.

La taille de l'ensemble des configurations candidates, c.-à-d. le nombre de solutions de notre espace de recherche, augmente avec :

- le nombre de modules dans le système,
- le nombre de MAFs admissibles calculées pour chaque module.

Le nombre de configurations est égal à :

$$|S| = \prod_{i=1}^{n} NMAF_i$$

où $NMAF_i$ est le nombre de MAFs candidates (admissibles) du module i et n le nombre de modules dans le système.

Cet espace pour un système réel peut devenir très grand, et une recherche exhaustive n'est raisonnablement pas envisageable. En effet, une centaine de modules sont déployés sur un A380. Pour extraire des solutions intéressantes, il faut d'une part définir des métriques de performance discriminantes, et aussi construire une recherche capable d'extraire de bonnes solutions rapidement. Ainsi, dans la suite de cette thèse, nous allons montrer comment, par la définition de métriques idoines, il est possible de réduire considérablement la cardinalité de l'ensemble des configurations candidates.

Dans le chapitre suivant, nous définissons un ensemble de métriques (ou fonctions de coût) capables d'évaluer la qualité d'une allocation temporelle selon différents critères. A la fin de ce chapitre, nous dégagerons les fonctions de coût qui seront choisies pour sélectionner des configurations candidates performantes pour l'intégrateur.

3.5 Conclusion

Dans ce chapitre, nous proposons une formulation originale du problème d'allocation temporelle des partitions dans un système IMA. Ce problème concerne le calcul de la période des partitions destinations de communications inter-modules. Nous sommes partis de la définition des contraintes temporelles temps-réelles pour en déduire une borne sur les périodes admissibles. Puis, nous avons intégré les contraintes propres au système IMA que sont l'utilisation de périodes harmoniques et la contrainte d'ordonnançabilité d'une MAF. La prise en compte de la contrainte d'ordonnançabilité est un problème de décision difficile car l'ordonnancement en IMA est non-préemptif par définition. Nous avons choisi d'utiliser une heuristique proche de l'algorithme *least loaded* pour décider de la validité d'une MAF. Cet algorithme n'est pas optimal mais il a pour avantage de détecter un nombre raisonnable de MAFs ordonnançables.

L'analyse de toutes ces contraintes nous a permis de transformer le problème initial d'optimisation continue sous contraintes en un problème d'optimisation combinatoire. Ainsi, nous avons identifié pour chaque module un ensemble de MAFs candidates, ordonnançables et ne menant à

aucune perte de donnée dans les communcations inter-modules. L'espace des solutions est maintenant le produit cartésien de ces ensembles de MAFs candidates calculées pour chaque module. Cet espace croit avec le nombre de modules et de MAFs candidates par module. Nous proposons dans la suite de cette thèse un découpage du problème d'optimisation permettant de réduire sa cardinalité. Dans le chapitre suivant, nous présentons différentes métriques d'optimisation possibles du système IMA. Leur analyse nous permettra de définir les métriques les plus pertinentes et de compléter la formulation du problème d'optimisation traité dans cette thèse.

4 Métriques pour l'évaluation d'une allocation temporelle

Au chapitre précédent, nous avons modélisé le problème d'allocation temporelle en aboutissant à un ensemble de MAFs admissibles pour chaque module. Ces ensembles respectent les contraintes temps-réel et système. Une solution du problème appartient maintenant au produit cartésien des MAFs admissibles de chaque module. Pour compléter la formulation de notre problème d'allocation, il faut maintenant définir un ensemble de métriques permettant d'évaluer la qualité d'une allocation. Ces métriques permettront de choisir les solutions les plus intéressantes selon une approche multi-critères.

Plusieurs travaux ont cherché à définir des métriques d'évaluation de performances pour optimiser des architectures embarquées temps réel. Dans [46], des métriques relatives à la charge mémoire des calculateurs et à la charge des communications sont utilisées. Dans [39], il est proposé d'évaluer le déploiement d'une architecture AUTOSAR en mesurant la marge disponible sur le temps de réponse de bout-en-bout et la consommation mémoire requise pour cela.

Dans la même veine que ces travaux, nous définissons dans ce chapitre deux familles de mesures de performance, l'une visant à réduire la charge de calcul des modules et l'autre à maximiser la marge disponible sur le délai de communication des flux. Nos métriques sont introduites via l'exemple 3.1, rappelé dans 4.1. Ces métriques sont d'abord exprimées localement à un module, puis généralisées à l'ensemble du système sous la forme de métriques globales. Nous avons choisi la forme de nos métriques afin de résoudre localement puis globalement notre problème d'optimisation.

4.1 Exemple introductif

Dans le chapitre précédent, nous avons illustré la formulation du problème d'allocation temporelle sur un exemple simple de système IMA rappelé dans la figure 4.1. Après l'obtention de ces valeurs, nous avons procédé au calcul des ordonnancement faisables des MAFs pour chaque module, qui serviront à l'allocation temporelle du système IMA complet. Plusieurs MAFs sont possibles pour chaque module.

Nous nous sommes alors heurtés à la difficulté de comparer différentes allocations temporelles

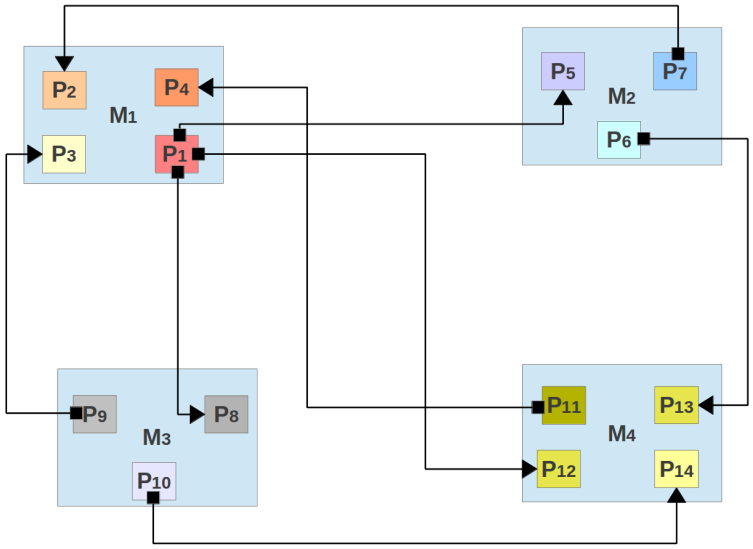


FIGURE 4.1 – Exemple d'une architecture IMA à 4 CPMs

d'un même système. Sur un système de la taille de l'exemple 4.1, la tâche semble moins complexe. Toutefois, pour des systèmes IMA de taille réelle, l'extraction de l'allocation temporelle adéquate n'est pas réalisable sans l'aide d'une méthode adaptée.

L'analyse des délais de bout-en-bout a permis de voir que le choix des valeurs des périodes des partitions destination influence la garantie ou la non-garantie des contraintes temporelles sur les communications, mais aussi la charge des modules de calcul sur lesquels elles sont allouées. Nous nous intéressons dans ce chapitre aux *performances des communications* entre partitions distantes et à la *charge des modules de calcul*, et ce pour comparer la qualités de deux allocations possibles.

Dans la section suivante, nous analysons les performances des CPMs et des communications de l'exemple décrit dans 3.1 pour deux allocations temporelles admissibles, s_1 et s_2 parmi les 16 allocations valides. Ces deux allocations sont représentées sur la Figure 4.2, où :

- s_1 représente l'allocation 3.10.(a) composée de ($MAF_{1,1}$,$MAF_{2,1}$, $MAF_{3,1}$, $MAF_{4,1}$),
- s_2 représente l'allocation 3.10.(b) composée de ($MAF_{1,2}$, $MAF_{2,1}$, $MAF_{3,1}$, $MAF_{4,1}$).

La différence entre s_1 et s_2 réside dans le choix de la MAF du module M_1. On rappelle que pour ce module, la période de la partition source P_1 est fixée par l'application à 120 ms, et que les périodes de P_2, P_3 et P_4 ne peuvent pas dépasser les périodes destination maximales respectivement égales à 48 ms, 40 ms et 35 ms. Comme il a été calculé précédemment, deux MAFs sont possibles pour M_1 :

- P_2 et P_3 utilisent des périodes de 40 ms et P_4 une période de 20 ms, c.à.d $MAF_{1,1}$=(120,40,40,20) pour s_1,
- P_2, P_3 et P_4 leurs sont assignées des périodes de 30 ms c.à.d $MAF_{1,2}$=(120,30,30,30) pour

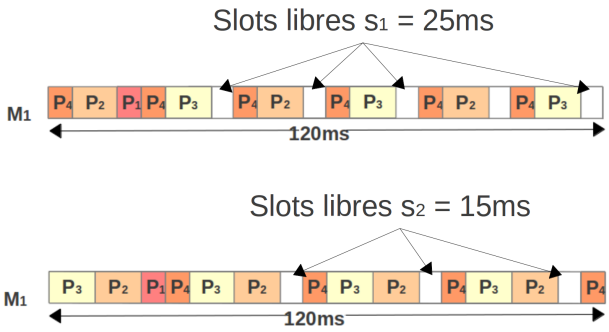


FIGURE 4.2 – Slots libres dans s_1 and s_2

s_2.

Afin de comparer les allocations s_1 et s_2, nous devons procéder comme suit:

1. Quantifier les performances d'une communication pour un flux,
2. Quantifier la charge d'un CPM, et
3. En déduire des métriques globales pour mesurer la qualité de tout le système.

4.1.1 Charge du module M_1

Les MAFs qui différencient s_1 et s_2 ($MAF_{1,1}$ et $MAF_{1,2}$) sont difficiles à comparer intuitivement compte tenu de la répartition des périodes des partitions de M_1. Afin d'évaluer les performances des deux allocations, nous calculons la charge du module. La charge est définie par le rapport entre la durée de traitement imposée par toutes les partitions et la durée de la MAF. Ainsi, une charge qui vaut 1 reflète un module saturé et une charge de 0 un module inactif.

La répartition des fenêtres d'exécution des partitions dans $MAF_{1,1}$ et $MAF_{1,2}$ est représentée sur la Figure 4.2. $MAF_{1,1}$ offre plus de slots libres (25 ms soit une charge de 79.2%) comparé à $MAF_{1,2}$ (15 ms soit une charge de 87.5%).

Si $MAF_{1,1}$ est choisie pour l'allocation temporelle de M_1, il est possible d'améliorer son évolutivité. En effet, $MAF_{1,1}$ pourrait accueillir à l'avenir plus de fonctions avioniques que M_2 compte tenu de la durée de ses slots libres.

4.1.2 Marge de communication de la partition destination P_2

Afin de montrer l'impact de $MAF_{1,1}$ et $MAF_{1,2}$ sur les communications, nous détaillons l'analyse de la communication $Com_{7,2}$. La partition source est hébergée par M_2 et la partition destination est hébergée par M_1. Le calcul pire cas du délai de communication de bout-en-bout est respectivement de $E2E_{wc}$=52 ms dans l'allocation s_1 et de $E2E_{ws}$=42 ms dans s_2. Selon la propriété 1, la contrainte de fraîcheur est garantie pour tous les délais de communication $E2E<FP$, avec $FP_{7,2}$ = 60ms.

Le calcul de la différence entre le paramètre de fraîcheur et le délai de communication de bout-en-bout pire cas (qui, rappelons le, est égal à la somme de la latence réseau pire cas L_{max}= 12 ms et de la valeur de la gigue sur le module destination, égale au pire cas à la période T_2 de la partition destination) est un moyen de comparer les performances de $Com_{7,2}$ pour les allocations temporelles s_1 et s_2.

En effet, la différence $\delta_{7,2}$=$E2E_{wc}$-$FP_{7,2}$ représente la marge disponible sur le délai pire cas de la communication avant que celui-ci ne viole le paramètre de fraicheur $FP_{7,2}$ (cf. Figure 4.3). Plus formellement, ce delta représente *la marge d'évolution possible de la communication* : si la communication connaît une baisse de débit, une augmentation dans la latence réseau pire cas L_{max} ou une augmentation de la période de la partition cible, $Com_{7,2}$ dispose d'une *marge de manœuvre* que nous notons δ, égale à cette marge. Pour les deux allocations, nous obtenons ainsi:

- dans s_1: $\delta_{7,2} = FP_{7,2}$-$E2E_{wc}$= 60 - 52 = 8,
- dans s_2: $\delta_{7,2} = FP_{7,2}$-$E2E_{wc}$= 60 - 42 = 18.

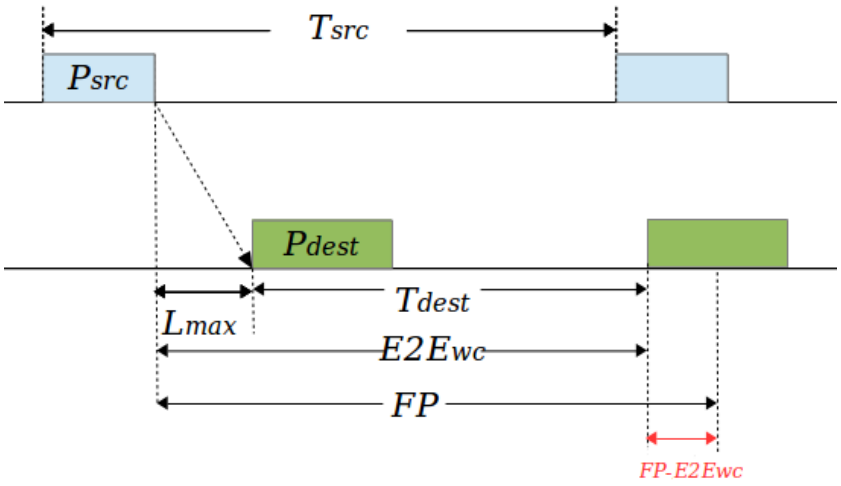


FIGURE 4.3 – Calcul de la marge δ d'une communication

Toutefois, pour être capable de comparer s_1 et s_2, il faut considérer toutes les communications qui ciblent le CPM M_1, à savoir $Com_{7,2}$, $Com_{9,3}$ et $Com_{11,4}$. Nous nous intéressons aux périodes des partitions destination étant donné que celles-ci varient d'une MAFs à l'autre. Il n'est pas utile de considérer les communications sortantes car les périodes des partitions sources sont fixées. Les tableaux 4.1 et 4.2 représentent le calcul de la marge pour les communications qui ciblent M_1 pour s_1 et s_2 respectivement.

A la lecture des tableaux 4.1 et 4.2, nous constatons que la moyenne des marges pour s_2 (11ms) est supérieure à la moyenne des marges de s_1 (7.6ms). De plus, la plus petite marge de communication observée dans s_1 est de 0, auquel cas aucune modification dans le réseau n'est admise, tandis que la plus petite marge dans l'allocation s_2 est de 5 ms. Il est clair que s_2

	$E2E_{wc}$ (ms)	Paramètre de fraîcheur $FP_{i,j}$ (ms)	Marge $\delta_{i,j}$
$Com_{7,2}$	52	60	8
$Com_{9,3}$	60	60	0
$Com_{11,4}$	25	40	15
Moyenne			7.6

TABLEAU 4.1 – $E2E_w$, Paramètre de fraîcheur et marge des communications qui ciblent M_1 avec $(T_2, T_3, T_4) = (40, 40, 20)$.

propose une meilleure marge moyenne d'évolution du réseau. Si, pour une quelconque raison, la configuration du réseau avait besoin d'être modifiée et augmentait la latence réseau L_{max} d'au plus 5 ms par VL, nous pouvons conclure que s_2 est toujours une allocation valide. Il ne faudra pas modifier la MAF de M_1. s_2 est donc l'allocation idéale pour obtenir un réseau évolutif.

	$E2E_{wc}$ (ms)	Paramètre de fraîcheur $FP_{i,j}$ (ms)	Marge $\delta_{i,j}$
$Com_{7,2}$	42	60	18
$Com_{9,3}$	50	60	10
$Com_{11,4}$	35	40	5
Moyenne			11

TABLEAU 4.2 – $E2E_w$, Paramètre de fraîcheur et marge des communications qui ciblent M_1 avec $(T_2, T_3, T_4) = (30, 30, 30)$.

Ainsi, nous avons vu que l'allocation s_1 est plus apte aux ajouts de nouvelles fonctions ou partitions sur le module M_1. Cependant elle est moins évolutive du point de vue du réseau. Les deux allocations s_1 et s_2 représentent deux compromis différents entre l'évolutivité de la charge des modules et l'évolution du réseau. En effet les performances des deux solutions sont différentes. Il revient à l'intégrateur, en fonction de l'évolutivité voulue pour le système, de choisir le compromis qui lui convienne le mieux. Dans la suite, nous allons introduire des métriques élémentaires, calculées localement aux modules, et des métriques globales, calculées pour toute l'architecture. Qu'elles soient élémentaires ou globales, ces métriques vont mesurer

1. la charge des modules, que l'on cherchera à minimiser,
2. la marge réseau, que l'on cherchera à maximiser.

Les solutions offrant les meilleurs compromis pour ces deux métriques permettront de fournir une architecture qui offrira des évolutions futures à moindres coûts.

4.2 Métriques élémentaires

Dans cette section, nous proposons d'introduire des critères, ou métriques de performances à l'échelle d'un module capable de quantifier la qualité de différentes MAFs [54]. Nous nous basons sur les analyses des performances des CPMs et des communications réalisées précédement.

Nous définissons dans cette section deux critères qui quantifient les qualités d'une unité de calcul CPM, et d'une communication entre deux partitions distantes, respectivement.

4.2.1 Charge d'un module M_ℓ

Nous définissons cette métrique, notée Q_ℓ, comme étant la charge du module M_ℓ issue de l'exécution périodique des partitions qu'il héberge. Cette métrique est à minimiser. En effet, moins le module de calcul est chargé, plus il sera en mesure d'accueillir des ajouts futurs de nouvelles fonctions. Toutefois, il est important de noter que cet ajout ne sera possible que si la durée et la période d'exécution de la nouvelle partition reste compatible avec l'ordonnancement actuel. Il se peut que l'ordonnancement actuel ne permette pas d'assurer l'exécution périodique de cette nouvelle fonction, auquel cas l'ordonnancement du module devra être revu. Ne connaissant pas les propriétés temporelles de ces futures partitions, nous avons décidé dans cette thèse de choisir la MAF qui minise simplement la charge du module.

Calcul de la charge d'un module M_ℓ. Nous calculons ce critère pour une MAF donnée (ici MAF_ℓ) par la formule:

$$Q_\ell = \frac{\sum_{P_k \in M_\ell} b_k \times \left\lfloor \frac{b_{MAF_\ell}}{T_k} \right\rfloor}{b_{MAF_\ell}} \tag{4.1}$$

où b_{MAF_ℓ} est la longueur de la MAF de l'allocation considérée, et $\left\lfloor \frac{b_{MAF_\ell}}{T_k} \right\rfloor$) est le nombre d'occurrences de P_k dans MAF_ℓ.

Nous appliquons ce calcul sur s_1 et s_2. Nous obtenons un taux d'utilisation Q_1 pour $MAF_{1,1}$ de 0.79 contre 0.86 pour $MAF_{1,2}$, ce qui correspond à l'analyse des performances des modules CPMs réalisées. Il est clair que s_1 a un facteur d'évolutivité plus intéressant que s_2 car il minimise la métrique Q_1 avec $MAF_{1,1}$

4.2.2 Marge de communication pour une partition destination

Pour l'évaluation des performances d'une communication, nous avons pu voir que le calcul de la marge du paramètre de fraîcheur FP et du délai de communication pire cas $E2E_{wc}$ (la marge δ), nous permet de déterminer une marge d'évolution permise avant que la communication n'atteigne le seuil critique où les messages sont perdus à cause du dépassement du paramètre de fraîcheur.

Nous proposons une métrique capable de quantifier la marge d'une communication qui respecte les contraintes des propriétés 1 et 2, à savoir la contrainte de fraicheur et la garantie des mises à jour. Cette métrique, notée σ_j, est illustrée sur la Figure 4.4. Elle représente la distance entre la période destination choisie et la période maximum dont elle dérive avant d'être une période destination invalide (avant d'atteindre la période pire cas T_j^{max}, auquel cas le délai de communication de bout-en-bout est égal au délai pire cas $E2E_{wc} = T_j^{max} + L_{max}^{i,j}$). Cette

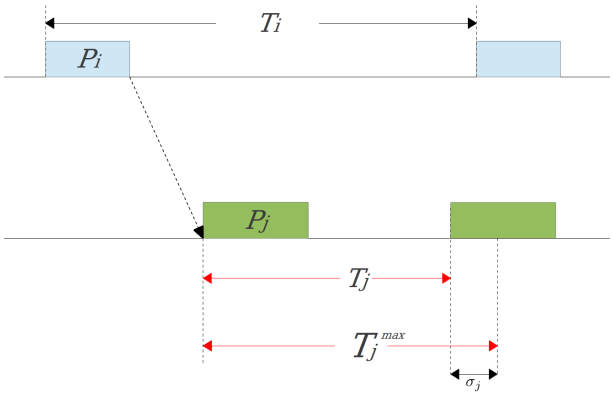


FIGURE 4.4 – Marge dans une période destination

métrique est calculée par la formule (4.2):

$$\sigma_j = T_j^{max} - T_j \tag{4.2}$$

La variabilité de cette métrique dépend de la valeur de T_j^{max} calculée dans 3.5. Deux cas sont possibles:

- $T_j^{max} = FP_{i,j} - L_{max}^{i,j}$: c.à.d que la valeur de la propriété 1 est la plus contraignante, la métrique σ_j est égale à la marge δ sur une communication,
- $T_j^{max} = T_i - (L_{max}^{i,j} - L_{min}^{i,j})$: c.à.d la valeur de la propriété 2 est la plus contraignante.

En effet, la marge δ utilisée sur notre exemple est plus lâche que la marge σ : elle ne considère que la contrainte sur la fraicheur des données, tandis que la métrique σ intègre également la contrainte de la garantie des mises à jour.

Le calcul de cette métrique σ_j, avec j l'indice de la partition destination, est une *métrique commune* pour toutes les communications dont la partition cible est P_j. Cette métrique a pour but d'agréger la quantification d'un ensemble de communications qui visent une même partition destination avec la même métrique et valeur. Pour une communication $Com_{i,j}$ et une communication $Com_{k,j}$, nous pouvons utiliser la même métrique σ_j pour quantifier les deux communications.

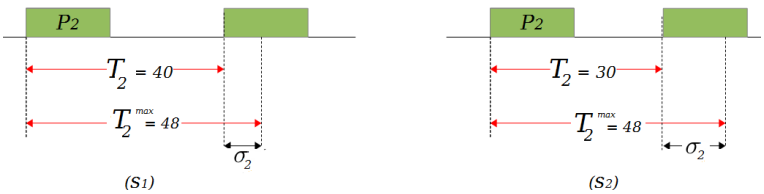


FIGURE 4.5 – Métrique σ_2 pour la partition P_2

4.2.3 Marge des communications d'un module

Afin de quantifier la qualité des communications d'un même module, nous définissons deux métriques qui rassemblent les marges σ de toutes les communications entrantes. La première métrique est la *marge moyenne du module* :

$$\sigma_{M_\ell} = \frac{\sum_{P_j \in M_\ell} \sigma_j}{nb_{Com_\ell}} \tag{4.3}$$

Cette métrique calcule la moyenne des marges des périodes des partitions destination dans le module M_ℓ au nombre de nb_{Com_ℓ}.

La seconde métrique, plus restrictive, est la *marge pire cas dans le module* (formule (4.4)):

$$\sigma^{wc}_{M_\ell} = \min_{P_j \in M_\ell} \sigma_j \tag{4.4}$$

Ici, la qualité du module est représentée par la communication qui a la plus mauvaise marge. En la maximisant, on choisit une solution qui assure une marge minimale sur toutes les communications du module.

Pour un système de grande envergure, il suffit de calculer la métrique σ pour les partitions destination, au lieu de calculer la métrique δ pour chaque communication, ce qui réduit le nombre de calculs.

4.2.4 Comparaisons des métriques élémentaires sur s_1 et s_2

Nous allons comparer les performances des allocations temporelles s_1 et s_2 selon les métriques présentées précédemment. s_1 et s_2 sont antagonistes. En effet, si nous considérons le couple de métriques (Q_1, σ_{M_1}), où la configuration la plus intéressante est celle qui minimise Q_1 et maximise en même temps σ_{M_1}, nous pouvons conclure que le choix de $MAF_{1,1}$ avec (Q_1, σ_{M_1})=(0.79,7.6) ou de $MAF_{1,2}$ avec (Q_1, σ_{M_1})=(0.87,11) sera fonction des compromis que l'intégrateur est capable de faire et des évolutions futures auxquelles le système devra faire face: si l'intégration doit garantir de M_1 la possibilité de l'ajout de nouvelles fonctions, $MAF_{1,1}$ est privilégiée. Par contre $MAF_{1,2}$ sera choisie si les communications autour de M_1 doivent offrir plus de marges à d'éventuelles modifications dans le réseau. Nous constatons qu'il est possible, au niveau de granularité d'un CPM, de réaliser des choix en amont sur les MAFs admissibles de ce CPM à l'aide des critères que nous venons de définir. Cette remarque sera exploitée au chapitre suivant.

Afin de comparer la totalité des allocations temporelles d'un système IMA, il faut maintenant définir des métriques globales. Celles-ci doivent évaluer les performances du système dans son ensemble.

4.3 Métriques globales du système

Ces métriques sont calculées à l'échelle du système complet. Nous allons étendre les métriques précédentes pour caractériser les performances globales d'un système complet IMA. Plusieurs options sont possibles que nous dérivons à partir des métriques élémentaires. Nous avons choisi de caractériser une allocation d'un système par deux catégories de métriques globales capables de quantifier d'une part, les performances et l'évolutivité du réseau, et d'autre part la marge d'évolution de ses modules de calculs. Nous dérivons les métriques globales à partir de la charge des modules Q et de la marge σ sur les périodes des modules destination d'un CPM. Après avoir présenté toutes les métriques globales possibles, nous illustrons ces dernières sur l'exemple présenté dans 4.1, pour lequel nous procédons à la comparaison des 16 allocations possibles afin d'en déterminer les allocations qui présentent les meilleurs compromis au vu des deux critères de charge et de marge réseau.

4.3.1 Charge d'un système IMA

La possibilité de l'ajout de nouvelles fonctions dans un système IMA est liée à la capacité d'évolution des modules de calcul qui le constituent. Ayant caractérisé chaque CPM par une métrique capable de quantifier son taux d'occupation, nous agrégeons ce critère élémentaire à chaque module en une métrique globale.

4.3.1.1 Charge moyenne des modules d'un système IMA.

Nous proposons de calculer la moyenne des charges CPM d'un système IMA. Ce calcul de moyenne donne une métrique capable d'apporter une connaissance de l'état de charge du système à partir des CPM qui le constituent. Cette métrique moyenne, que nous notons Q_{avg} est calculée selon la formule (4.5):

$$Q_{avg} = \frac{\sum_{i\in[1..m]} Q_i}{m} \tag{4.5}$$

Bien entendu, cette métrique de la charge moyenne est à minimiser.

4.3.1.2 Charge maximale d'un système IMA

Il est aussi intéressant d'analyser une métrique pire cas plus discriminante que la charge moyenne Q_{avg}. Pour cela, nous définission le critere de charge maximale comme étant la charge du module le plus chargé du système IMA. Nous notons cette métrique Q_w. Comme pour la charge moyenne, il faut minimiser cette métrique Q_w. Elle est définie par la formule (4.6) :

$$Q_w = \max_{l\in[1..m]} Q_\ell \tag{4.6}$$

Cette métrique assure que tous les modules d'une allocation ont une charge inférieure ou égale à Q_w.

Nous nous intéressons également à la capacité d'évolution du réseau composé des différentes communications entre les partitions du système. Nous nous basons dans ce qui suit sur les métriques élémentaires qui caractérisent les marges sur les communications, présentées dans la section 4.2.2.

4.3.2 Métriques globales réseau

Les performances d'un réseau dans un système IMA sont liées au délais de communication de bout-en-bout des différents flux. Pour qu'un système soit pérenne, il est important qu'il soit apte a intégrer facilement des changements mineurs, sans altérer les performances du système déjà déployé. Les métriques définies ci-après ont pour but de favoriser le choix d'allocations initiales robustes à de futurs changements réseau. Ainsi, si une nouvelle fonction communicante IMA doit être intégrée au système dans le futur, les impacts sur le réseau pourront être absorbés de façon transparante, dans la limite des marges temporelles calculées lors du déploiement initial.

4.3.2.1 Moyenne des marges réseau globale.

Nous proposons une métrique capable de caractériser en moyenne la marge d'évolution dont disposent les communications du système. Elle est calculée selon la formule (4.7) qui suit:

$$\Delta_{avg} = \frac{\sum_{i \in [1..n_{dest}]} \sigma_i}{n_{dest}} \tag{4.7}$$

où n_{dest} est le nombre de partitions destination appartenant à tout le système étudié. En effet, la marge réseau définie en (4.2), se calcule pour chaque partition destination du système (et non pour chaque communication). Cette métrique permet de savoir si les marges dans les communications sont uniformément distribuées et si la moyenne de ces marges permet au système de réaliser des évolutions sans impacter le fonctionnement prédéfini.

L'intérêt de cette métrique est qu'elle est capable d'offrir une vision globale et uniforme de l'état du réseau du système et de sa capacité d'évolution. Une allocation s_i est intéressante parmi un ensemble d'allocations temporelles possibles si celle-ci propose en moyenne la plus grande marge réseau. L'inconvénient de cette métrique moyenne est de cacher de potentiels extrêmes. En effet, une allocation peut proposer une grande marge moyenne tout en comportant des communications avec de très faibles marges d'évolution, compensées par des communications à marges réseau très grandes.

4.3.2.2 Marge réseau pire cas globale.

Il est également possible d'utiliser une métrique pire cas, qui a pour valeur le minimum des marges réseau observées dans le système. Elle est calculée selon la formule (4.8) qui suit:

$$\Delta_w = \min_{i \in [1..n_{dest}]} \sigma_i \tag{4.8}$$

Une allocation s_i, parmi un ensemble d'allocations temporelles candidates, est intéressante si elle propose la plus grande marge réseau pire cas. Ce critère permet de garantir que toutes les communications ont une marge réseau d'au moins Δ_w.

4.3.2.3 Application des métriques proposées

Nous proposons dans cette section d'appliquer les métriques proposées sur l'exemple illustré dans la Figure 4.1. Le but est de comparer différentes allocations temporelles admissibles pour un même système IMA et d'aider l'intégrateur à choisir l'allocation temporelle la plus intéressante au regard des métriques élémentaire et globales.

Calcul des métriques élémentaires. Le Tableau 4.3 liste les valeurs des métriques élémentaires calculées pour chacune des MAFs admissibles d'un des modules du système considéré. A la fin du chapitre 3, nous avions déterminé deux MAFs admissibles pour chacun des CPMs M_1 à M_4.

		Q_ℓ	σ_{M_ℓ}	$\sigma^w_{M_\ell}$
M_1	$MAF_{1,1}$	0.79	7.67	0
	$MAF_{1,2}$	0.88	11	5
M_2	$MAF_{2,1}$	0.75	28	28
	$MAF_{2,2}$	1	58	58
M_3	$MAF_{3,1}$	0.75	25	25
	$MAF_{3,2}$	1	55	55
M_4	$MAF_{4,1}$	0.88	12.67	10
	$MAF_{4,2}$	1	26	10

TABLEAU 4.3 – Métriques élémentaires

Les métriques élémentaires permettent de connaître à un niveau de granularité plus fin (niveau CPM) l'impact des choix des MAFs sur les performances. Elles indiquent au niveau CPM selon la MAF choisie les performances et la capacité d'évolution attendue. La métrique Q_ℓ indique le taux d'occupation du CPM selon les périodes de la MAF choisie. Dans le tableau 4.3, nous pouvons constater que pour les CPM M_1, M_2, M_3 et M_4, les MAFs respectives $MAF_{1,1}$, $MAF_{2,1}$, $MAF_{3,1}$ et $MAF_{4,1}$ proposent les taux d'occupation les plus bas.

Les métriques élémentaires des communications, σ_{M_ℓ} et $\sigma^w_{M_\ell}$ quantifient la capacité d'évolution des communications entrantes vers chaque CPM. Les MAFs les plus intéressantes sont

celles qui maximisent ces dernières métriques. Dans le tableau 4.3, nous pouvons constater que pour les CPM M_1, M_2, M_3 et M_4, les MAFs respectives $MAF_{1,2}$, $MAF_{2,2}$, $MAF_{3,2}$ et $MAF_{4,2}$ proposent les marges réseau moyennes les plus intéressantes. Nous pouvons noter que ces MAFs ne sont pas celles qui minimisent la charge moyenne des modules. Il y a donc un compromis entre les deux critères de charge et de marge réseau à trouver, cette fois-ci en calculant les métriques globales.

Calcul des métriques globale. Le Tableau 4.4 liste les valeurs des métriques globales présentées dans 4.3. Chaque solution $s_i \in S, i \in [1..16]$ est obtenue à partir d'une combinaison possible des MAFs admissibles des modules. Cette solutions est représentée par un n-uplet ordonné où n est le nombre de modules du système et où chaque élément $l \in m$ du n-uplet représente une des MAFs admissibles du le CPM M_ℓ.

	Q_{avg}	Q_w	Δ_{avg}	Δ_w
s_1	0.79	0.87	14.25	0
s_2	0.82	1	19.25	0
s_3	0.85	1	18	0
s_4	0.88	1	23	0
s_5	0.85	1	18	0
s_6	0.88	1	23	0
s_7	0.92	1	21.75	0
s_8	0.95	1	26.75	0
s_9	0.81	0.88	15.5	5
s_{10}	0.84	1	20.5	5
s_{11}	0.86	1	19.25	5
s_{12}	0.91	1	24.25	5
s_{13}	0.86	1	19.25	5
s_{14}	0.91	1	24.25	5
s_{15}	0.94	1	23	5
s_{16}	0.97	1	28	5

TABLEAU 4.4 – Métriques globales

Nous pouvons observer du Tableau 4.4 et de la Figure 4.6, que la solution s_9 présente un compris intéressant car elle présente à la fois une charge faible du système (Q_{avg}=0.81 et Q_w=0.88) et une marge réseau importante (Δ_{avg}=15.5 et Δ_w=5).

Toutefois, d'autres allocations candidates sont des solutions intéressantes car elles reflètent d'autres compromis entre nos métriques. Les métriques sur les charges des modules et les métriques d'évaluation de la marge du réseau sont antagonistes. Il est difficile de départager certaines solutions candidates ici. Par exemple, une solution peut être meilleure qu'une autre sur le premier critère mais moins bonne sur le second. Ces deux solutions fournissent deux compromis différents et ce n'est que l'intégrateur pour pourra faire un choix définitif s'il décide de privilégier la charge ou la marge réseau.

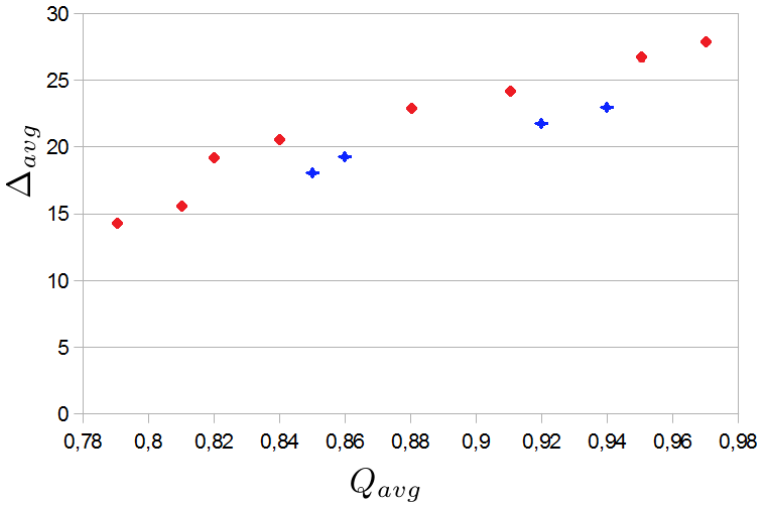


FIGURE 4.6 – Solutions candidates à l'allocation temporelle. Les meilleurs compromis sont en rouge

En comparant les solutions candidates pour les métriques Q_{avg} et Δ_{avg}, nous constatons que les allocations qui proposent les meilleurs compromis sont $s_1, s_2, s_4, s_6, s_8, s_9, s_{10}, s_{12}, s_{14}$ et s_{16}. Elles sont mises en avant dans la Figure 4.6 qui les positionne dans l'espace des critères Q_{avg} et Δ_{avg}. Avec la métrique de marge réseau pire cas Δ_w et la métrique moyenne Q_{avg}, il n'y a que deux solutions s_1 et s_9 qui présentent les meilleurs compromis possibles. Ceci est dû au fait que les métriques pire cas sont bien moins discriminantes que les métriques moyennes. C'est dans le prochain chapitre que nous motiverons la sélection de nos métriques, car ce choix est en lien direct avec l'algorithme d'optimisation multi-critère que nous avons conçu.

4.4 Conclusion

Dans ce chapitre, nous avons proposé des critères permettant l'analyse des performances des modules et des communications dans un système IMA. Les métriques ont été choisies afin d'offrir un système évolutif, où par la suite, il sera encore possible d'intégrer de nouvelles fonctionnalités aux modules et de nouveau flux au réseau. Pour cela, nous nous sommes intéressés à la charge d'un module et à la marge réseau des communications aboutissant sur une partition donnée.

Nous avons défini des métriques élémentaires, capables de quantifier la qualité d'une MAF sur un module, que ce soit pour la charge ou la MAF réseau. Nous avons étendu ces métriques pour en obtenir des définitions globales au système complet.

Ces différentes métriques sont illustrées sur un exemple simplifié de système IMA. Nous avons pu constater sur ce petit exemple qu'il n'est pas aisé de choisir une solution car plusieurs compromis entre les critères étudiés existent. Ceci est dû au fait que les critères de charge et de

marge réseau sont antagonistes. Nous allons montrer dans le chapitre suivant que l'extraction des meilleurs compromis possibles est en fait la recherche des solutions dites *Pareto-optimales* issues d'un problème d'optimisation multicritère ou multiobjectif. La résolution d'un problème multicritère sélectionne les meilleurs compromis possibles et laisse le choix définitif à l'intégrateur qui pourra, sur un ensemble restreint de solutions Pareto-optimales, utiliser des mécanismes de décision plus fins pour opérer le choix final.

5 Optimisation multiobjectif pour l'allocation temporelle en IMA

Les travaux présentés dans l'état de l'art du chapitre 2 n'abordent pas spécifiquement la problématique de l'allocation temporelle pour l'intégration des systèmes embarqués temps réel. Aussi, la plupart des heuristiques proposées dans les travaux mentionnés n'optimisent qu'une seule fonction objectif. Cependant, lorsque nous cherchons à atteindre une allocation optimale, nous cherchons souvent à satisfaire plusieurs objectifs. Nous pouvons, par exemple, chercher à minimiser la consommation énergétique d'un réseau tout en maximisant la bande passante offerte aux utilisateurs. Si les deux objectifs vont dans le même sens, on peut définir un unique critère. Par contre, si les deux objectifs sont antagonistes et ne permettent pas, individuellement parlant, de sélectionner la même solution optimale, on se trouve en face d'un problème d'optimisation multicritère (ou multiobjectif).

Nous présentons dans ce qui suit deux méthodes d'optimisations multiobjectif capables d'extraire un ensemble de solutions intéressantes pour l'allocation temporelle d'un système IMA. Ces heuristiques recherchent des solutions de compromis selon deux fonctions objectif qui optimisent la charge du module et la marge d'évolution du réseau. La première méthode est une recherche exhaustive basée sur la dominance au sens de Pareto. La seconde est une heuristique basée sur la méthode de recherche Tabou adaptée aux problèmes multiobjectifs.

5.1 Un problème d'optimisation multiobjectif

Dans le chapitre 3, nous avons réalisé une analyse détaillée des délais de bout-en-bout pour la formulation du problème de l'allocation temporelle. Nous avons pris en compte à la fois des contraintes système (ordonnancement, harmonicité des périodes) et des contraintes temps-réel pour définir un ensemble de MAFs admissibles pour chaque module du système IMA. Dans le chapitre 4, nous avons réalisé une analyse de performance des modules et des communications dans le système IMA. Des métriques de marge réseau et de charge des modules, élémentaires et globales, ont été proposées et illustrées sur un exemple. L'analyse des métriques pour les solutions de l'exemple nous a permis de constater que la marge réseau et la charge étaient des critères antagonistes. Nous devons maintenant nous tourner vers des méthodes de recherche liées

à l'optimisation multicrière pour résoudre notre problème d'allocation temporelle.

Dans la suite, nous finalisons la modélisation du problème d'allocation temporelle en formulant le problème d'optimisation multicritère résolu dans cette thèse. Cette formulation est une contribution originale de ces travaux.

5.1.1 Solutions et variables

L'étude réalisée dans le chapitre 3 nous permet de construire l'ensemble des MAFs candidates (ou admissibles) pour l'allocation temporelle d'un système composé de m modules. A chaque module $M_\ell, \ell \in 1..m$, est associé l'ensemble B_ℓ de MAFs admissibles. Leur construction assure les contraintes temps réel (contrainte sur la fraicheur d'une donnée avec *Propriété 1*, contrainte sur la garantie des mises-à-jour avec *Propriété 2*) et la faisabilité de leurs ordonnancements (contraintes système avec l'algorithme LL et l'harmonisation des périodes des partitions).

Il est désormais possible de construire l'ensemble S des allocations temporelles pour un système à partir des MAFs admissibles de chaque module. Une solution candidate est alors composée de m MAFs, chacune appartenant à l'ensemble des MAFs admissibles de son module. Le nombre des solutions candidates à l'allocation temporelle d'un système dépend du nombre de modules et de nombre des MAFs possibles par chaque module.

Formellement, une solution candidate $s \in S$ à l'allocation temporelle d'un système composé de m modules est un m-uplet. L'élément numéro ℓ de ce m-uplet est noté $MAF_{\ell,k}$. Il correspond à l'allocation de la MAF admissible numéro k de l'ensemble B_ℓ des MAFs admissibles du module M_ℓ. L'ensemble des allocations possibles S est donc égal au produit cartésien des ensembles des MAFs admissibles de tous les modules, soit:

$$S = \prod_{\ell=1}^{m} B_\ell \tag{5.1}$$

La taille de l'ensemble S des solutions candidates est donné par le produit des cardinalités des m ensembles B_ℓ, $\ell \in [1...m]$, soit:

$$Card(S) = \prod_{\ell=1}^{m} Card(B_\ell), \;\; avec\, B_\ell,\; \ell \in [1..m] \tag{5.2}$$

Le Tableau 4.4 liste l'ensemble des solutions possibles à l'allocation temporelle du système de l'exemple 3.1.

5.1.2 Choix des métriques

La section 4.2 liste un ensemble de métriques élémentaires permettant de quantifier les performances d'un unique module et des communications entrantes à celui-ci. Nous allons maintenant présenter les métriques que nous avons retenues pour l'optimisation du système IMA.

5.1.2.1 Choix des métriques locales

Nous retenons les métriques suivantes appliquées à un même module M_ℓ :

- **Métrique de charge du module** Q_ℓ calculée selon (4.1). Afin de simplifier certaines formules, on pourra aussi faire référence à cette métrique par la notation g_1 dans la suite du document.
- **Métrique de marge réseau moyenne** σ_{M_ℓ} calculée selon (4.3). On pourra aussi faire référence à cette métrique par la notation g_2 dans la suite du document.

Nous avons aussi défini une métrique de marge réseau pire cas. Cette métrique est moins discriminante que la métrique de marge moyenne. En d'autre termes, on retrouve un nombre important de solutions avec une même valeur de marge réseau pire cas. Ceci ne facilite par leur comparaison. C'est une des raisons pour laquelle nous avons choisi de travailler avec la métrique de marge réseau moyenne. L'autre raison est liée à la relation de linéarité qui existe entre la métrique de marge réseau moyenne et la métrique de marge globale que nous avons choisie. Ce point sera mis en avant au moment de la définition de nos algorithmes d'optimisation.

5.1.2.2 Choix des métriques globales

Nous retenons les métriques globales suivantes :

- **Métrique de charge moyenne** Q_{avg} calculée selon (4.5). Afin de simplifier certaines formules, on pourra aussi faire référence à cette métrique par la notation f_1 dans la suite du document.
- **Métrique de marge réseau moyenne** Δ_{avg} calculée selon (4.7). On pourra aussi faire référence à cette métrique par la notation f_2 dans la suite du document.

Le choix de métriques pire cas a été écarté car il n'est pas assez discriminant. En effet, nous avons besoin d'une vue précise de la charge globale du système. Le choix d'une métrique pire cas est pessimiste, puisque tout le système se voit résumé à la performance d'un seul CPM. Si aucune solution n'améliore la performance de ce CPM, on ne peut différencier les solutions avec ce critère.

5.1.3 Optimisation multicritère

A la fin du chapitre précédent, nous avons appliqué ces métriques à l'exemple de la Figure 4.1 afin de comparer les performances de différentes allocations temporelles possibles. Nous avons observé que les métriques de charge et de marge réseau ne permettent pas de sélectionner les mêmes solutions. Elles sont antagonistes. De ce fait, il est difficile de trouver une solution unique qui optimise conjointement les deux critères. En effet, la diminution de l'un entraîne une augmentation de l'autre objectif.

Il est possible d'après [55] de composer une nouvelle métrique unique à partir des deux fonctions de coût. Si f_1 est la fonction de charge à minimiser et f_2 la fonction de marge réseau

moyenne à maximiser, on pourrait définir une nouvelle fonction à maximiser de la forme :

$$f(s) = \alpha(1 - f_1(s)) + (1 - \alpha)f_2(s)$$

On revient alors à la maximisation d'une unique fonction sur l'espace des allocation possibles $s \in S$. Différents compromis peuvent être recherchés en modifiant le coefficient α. Par exemple, pour obtenir un compromis qui donne le même poids aux deux métriques, on pourrait optimiser $f(s)$ pour $\alpha = 0.5$.

Pourtant, cette approche n'est pas si simple à mettre en oeuvre. Toute la difficulté réside dans le choix du coefficient α. Il a été montré dans [56] que ce choix est délicat:

- En effet, il faut d'abord normaliser f_2 par la plus grande valeur possible de marge moyenne pour que les deux métriques contribuent identiquement au résultat. Cette normalisation n'est pas triviale de par la définition d'un critère 'en moyenne'.
- D'autre part, obtenir le compromis voulu en sélectionnant une valeur du paramètre α n'est pas évident. Les deux fonctions f_1 et f_2 n'ont pas la même dynamique (leur dérivée n'est pas semblable). Ainsi, un changement de solution impliquera peut-être une faible variation sur f_1 et une variation très importante sur f_2. De ce fait, on ne peut garantir que la solution obtenue pour $\alpha = 0.5$ sera le compromis où les deux critères pèsent pour moitié dans le choix.
- Pour finir, si l'intégrateur souhaite calculer des solutions qui reflètent différents compromis, il devra relancer autant de recherches que de compromis souhaités. Chaque recherche se fera pour une nouvelle valeur du coefficient α.

Une approche alternative dans la recherche de solutions à des problèmes d'optimisation multi-critères est la recherche directe de l'ensemble des compromis dits *Pareto-optimaux* [55]. Cette approche permet de calculer en une seule recherche l'ensemble (ou un sous-ensemble) des meilleurs compromis existant en optimisant conjointement les deux fonctions de coût. L'intégrateur n'a pas besoin de paramétrer sa recherche en utilisant α, et peut choisir à posteriori le compromis qui lui convient le mieux, à la vue de tous les compromis trouvés.

Ce mode de résolution d'un problème d'optimisation multiobjectif se base sur la définition de la *dominance au sens de Pareto*. Pour qu'une solution soit intéressante, il faut qu'il existe une relation de dominance entre la solution considérée et les autres solutions candidates. Cette relation se définit comme suit:

Dominance. Une solution x domine une solution y dans un problème d'optimisation si :

- x est au moins aussi bon que y pour tous les objectifs.
- x est strictement meilleur que y pour au moins un objectif.

Formellement, pour un problème de minimisation qui exploite n critères, x domine y si et seulement si [57]:

$$\forall i \in [1, n] : f_i(x) \leq f_i(y) \wedge \exists j \in [1, n] : f_j(x) < f_j(y) \tag{5.3}$$

Il est possible de diviser l'ensemble des solutions de notre problème en deux parties : l'ensemble des solutions *non dominées* et l'ensemble des solutions *dominées*. Les solutions qui dominent les autres mais ne se dominent pas entre elles sont appelées *solutions optimales au sens de Pareto* ou solutions *non dominées*. Ces solutions composent à la fin de la recherche la *surface de compromis* qui sera présentée à l'intégrateur. On peut aussi dénommer ces solutions non-dominées par le terme *front de Pareto*.

Ces ensembles de solutions dominées et non dominées peuvent être représentés sur le plan des fonctions objectif. La Figure 5.1 schématise les ensembles de solutions dominées et non-dominées dans le plan défini par deux fonctions objectif à minimiser, ici f_1 et f_2 . Chaque solution s est représentée par ses valeurs dans l'espace des fonctions objectif f_1 et f_2. Les solutions non dominées sont les solutions représentées sur le plan par des croix. Les solutions dominées sont elles représentées par des points. Afin de reconnaître pour un problème de minimisation les solutions non dominées sur un graphe, il suffit de sélectionner les points qui ne possèdent pas d'autres solutions dans les rectangles de projection [55], représentés en pointillés sur 5.1.

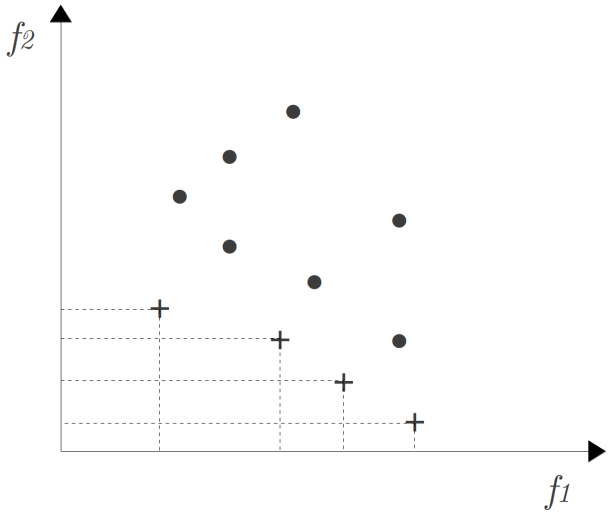


FIGURE 5.1 – Solutions dominées (points) et solutions non dominées (croix)

La recherche multicritère s'intéresse à l'obtention de l'ensemble des solutions non-dominées du problème d'optimisation. C'est celle que nous avons choisi de mettre en oeuvre dans cette thèse pour extraire un ensemble de compromis différents, que l'intégrateur pourra analyser plus finement pour effectuer son choix final.

Comme nous le verrons dans la section suivante, plusieurs heuristiques permettent d'obtenir l'ensemble des solutions non-dominées, ou un sous-ensemble qui s'en approche. Ces heuristiques peuvent exploiter la notion de *rang de dominance* ou *rang de Pareto*. Ce rang permet d'établir un classement pour toutes les solutions du problème. Les solutions de rang 1 sont les solutions non-dominées définies précédement. Les solutions de rang 2 sont les solutions non-dominées de

l'ensemble S privé des solutions de rang 1. Les solutions de rang 3 sont les solutions non-dominées de l'ensemble S privé des solutions de rang 1 et 2, etc.

L'algorithme suivant assigne les rangs de Pareto dans l'espace des solutions candidates [55], où N représente le nombre d'éléments dans l'ensemble des solutions sur lequel on effectue la comparaison.

```
RangC = 1  -- Rang courant
m = N
Tant que N ≠ 0 faire
      Pour i = 1 à m Faire
        Si x_i est non-dominée Alors
          Rang(x_i,t) = RangC
        Fin Si
      Fin Pour
      Pour i = 1 à m Faire
        Si Rang(x_i,t) == RangC Alors
          Ranger x_i dans un ensemble temporaire
          N = N − 1
        Fin Si
      Fin Pour
      RangC = RangC +1
      m = N
     Fin Pour
```

FIGURE 5.2 – Algorithme d'assignation des rangs de Pareto

5.1.4 Formulation du problème d'allocation temporelle multiobjectif

Le problème P d'allocation temporelle est formulé dans cette thèse comme un problème d'optimisation multiobjectif qui optimise conjointement deux critères : la charge moyenne des modules et la marge réseau moyenne du système IMA. Dans la suite de ce document, la fonction f_1 représentera le critère de charge moyenne et la fonction f_2 le critère de marge réseau moyenne.

L'objectif est d'extraire l'ensemble des solutions non-dominées qui à la fois minimisent la fonction objectif f_1 et maximisent la fonction objectif f_2. Le problème P est décrit par :

$$\begin{cases} \min_{s\in S} f_1(s) \\ \max_{s\in S} f_2(s) \end{cases} \text{ avec } S = \prod_{l=1}^{m} B_\ell \tag{5.4}$$

où, s est une solution candidate à l'allocation temporelle d'un système IMA dans un ensemble de solutions S obtenu par le produit cartésien des ensembles B_ℓ des MAFs admissibles.

Nous décrivons dans la section suivante les heuristiques multiobjectif que nous proposons

dans cette thèse pour solutionner le problème P.

5.2 Heuristiques multiobjectif proposées

Notre but est de proposer dans cette partie des heuristiques d'optimisation résolvant le problème d'allocation temporelle de l'équation (5.4). Nous allons donc rechercher la ou les allocations temporelles qui fournissent les compromis qui optimisent conjointement la charge et la marge réseau en moyenne.

Par exemple, pour un système composé de 6 modules, où chaque module admettrait 6 MAFs admissibles, l'ensemble des solutions du système est égal à $6^6 = 46656$ solutions. Dans les systèmes avioniques actuels, le nombre des modules de calcul est de l'ordre de la centaine avec plus d'un millier de flux de communication. Pour traiter le problème d'allocation d'un système d'une telle taille, il faut trouver une solution algorithmique qui puisse passer à l'échelle. Une recherche exhaustive qui évalue toutes les solutions de S permet de construire parfaitement l'ensemble des solutions non-dominées. Néanmoins, il est clair que dans sa forme la plus simple, elle ne peut apporter de solution pour un système IMA en un temps raisonnable.

Quand la taille du problème d'optimisation ne permet pas sa résolution exacte en un temps raisonnable, il est possible d'utiliser des heuristiques d'optimisation qui fournissent des solutions approchées. Ces heuristiques peuvent soit avoir été conçues pour répondre spécifiquement au problème posé, soit suivre un modèle d'heuristique d'optimisation plus général. Dans le second cas, le modèle suivi est une *métaheuristique*. Pour résoudre un problème d'optimisation, on spécialise cette métaheuristique en adaptant son comportement au problème donné. Des métaheuristiques ont été appliquées avec succès à différents problèmes multicritères [55][58]. Elles permettent d'obtenir un ensemble de solutions qui s'approche de l'ensemble de solutions Pareto-optimales du problème en un temps limité. Néanmoins, on ne peut garantir que les solutions fournies soient les solutions Pareto-optimales du problème.

Dans cette thèse, nous avons voulu explorer les deux pistes énoncées ici :

- La première piste est de dériver un algorithme d'optimisation qui fournisse l'ensemble Pareto-optimal complet. Pour cela, nous allons exploiter les spécificités de l'architecture IMA afin de réduire la cardinalité du problème, et ainsi être en mesure d'appliquer une recherche exhaustive sur des systèmes plus grands.
- La seconde piste est de spécialiser une métaheuristique pour qu'elle puisse fournir des solutions approchées quand le systèmes est très grand. Nous avons adopté la métaheuristique Tabou à cet effet [59].

Dans notre démarche, nous allons toujours chercher à exploiter les symétries du problème (issues de la construction des MAFs) et la linéarité des métriques.

Deux algorithmes d'optimisation multiobjectif sont présentés dans ce chapitre : **EXHAUST** et **TABOO**. Le premier est le résultat de la première piste et le second celui de la deuxième piste.

Optimisation locale aux modules Les algorithmes **EXHAUST** et **TABOO** partagent tous deux un premier niveau d'**optimisation locale** multiobjectif, présenté dans la section 5.3. Ce niveau s'effectue sur chaque module, localement et indépendamment des autres modules. C'est une recherche exhaustive locale au module qui s'appuie sur les métriques d'optimisation locales définies au chapitre 3. Cette recherche extrait les MAFs Pareto-optimales par dominance après avoir supprimé les MAFs dites *similaires*. Seules les MAFs Pareto-optimales des modules sont sélectionnées pour générer les allocations temporelles candidates s. Ainsi, la réduction du nombre de MAFs par module entraine une diminution de l'ensemble des allocations de l'ensemble S.

Les deux algorithmes diffèrent de par l'implantation du deuxième niveau d'optimisation, le niveau **d'optimisation global** :

Algorithme d'optimisation multiobjectif EXHAUST Après la phase d'optimisation locale qui permet de réduire la cardinalité de l'ensemble des allocations S, la phase d'optimisation globale recherche exhaustivement l'ensemble des solutions Pareto-optimales. Cette heuristique est adaptée aux systèmes de taille moyenne comme présenté à la section 5.4. C'est grâce à l'étape d'optimisation locale que l'algorithme **EXHAUST** permet de résoudre des systèmes de taille moyenne. L'avantage de cette approche est de fournir l'ensemble des solutions Pareto-optimales complet. Cet algorithme ne permet néanmoins pas de résoudre de grands problèmes. L'algorithme est présenté dans la section 5.4.

Algorithme d'optimisation multiobjectif TABOO Pour prendre en compte les systèmes de taille plus importante, un algorithme d'optimisation basé sur une métaheuristique a été mis en oeuvre ici. Dans sa phase d'optimisation globale, il applique la version multi-critère de Tabou proposée dans [57][58][60] aux solutions issues de l'optimisation exhaustive locale. L'algorithme est présenté dans la section 5.5.

5.3 Étape d'optimisation locale aux modules

Cette première étape d'optimisation est réalisée indépendamment pour chaque module du système. Dans cette étape, on cherche à réduire l'espace des MAFs admissibles B_ℓ de chaque module M_ℓ , $\ell \in [1...m]$ en ne sélectionnant que les MAFs non-dominées selon les critères d'optimisation locaux définis au chapitre précédent. On rappelle que ces deux critères locaux sont :

- La charge du module Q_ℓ, calculée selon l'équation (4.1),
- La marge réseau moyenne des communications entrantes σ_{M_ℓ}, calculée selon la formule (4.3).

Cette optimisation préliminaire permet de réduire le nombre des MAFs candidates par module, et par conséquent le nombre d'allocations temporelles à traiter dans l'optimisation globale.

L'extraction des MAFs optimales par module est réalisée par une recherche exhaustive des solutions non-dominées au sens des critères Q_ℓ et σ_{M_ℓ}. En pratique, le nombre de MAFs candidates par module est limité et la recherche exhaustive tout à fait envisageable.

Le fait d'avoir choisi des critères de charge et de marge réseaux *en moyenne*, nous permet de décomposer le problème en une recherche multicritere avec les critères locaux, suivie d'une recherche globale avec les critères globaux. La preuve en est donnée dans ce qui suit.

Démonstration. Les critères de charge f_1 et de marge réseaux f_2 globaux d'une solution $s \in S$ peuvent être décomposés en critères locaux aux modules. Autrement dit, chaque critère global est une composition linéaire des critères locaux.

En effet, pour f_1, le critère de la charge moyenne du système, $f_1 = \frac{1}{m}\sum_{\ell=1}^{m} Q_\ell$, avec Q_ℓ la charge moyenne du module M_ℓ. Si nous désirons minimiser uniquement le critère f_1, il faudra choisir la MAF de B_ℓ qui minimise Q_ℓ. Cette MAF est celle qui minimise également f_1 car f_1 est linéaire en Q_ℓ.

Il en est de même pour le critère f_2. En effet, f_2 est la moyenne des marges dans le réseau : $f_2 = \frac{1}{n_{dest}}\sum_{i\in[1..n_{dest}]} \sigma_i$, où n_{dest} est le nombre de partitions destination totales du système IMA. Il faudra choisir la MAF de B_ℓ qui maximise σ_i. Cette MAF est celle qui maximise le critère f_2 de part sa linéarité en σ_i également. □

Le choix des meilleurs compromis locaux aux modules permet de restreindre l'espace des allocations possibles en ne gardant que les MAFs qui mènent aux meilleurs allocations candidates possibles.

5.3.1 Algorithme de recherche des solutions Pareto-optimales par module

Le problème d'optimisation multicritere **P'** est défini pour l'ensemble B_ℓ des MAFs admissibles d'un module M_ℓ . Il est formulé ici pour conjointement :

- Minimiser $g_1 = Q_\ell$, la charge du module M_ℓ définie en (4.1), et,
- Maximiser $g_2 = \sigma_{M_\ell}$, la marge réseau moyenne des périodes des partitions destination du module M_ℓ définie en (4.3).

Les notations g_1 et g_2 sont principalement utilisées dans cette partie pour clarifier les différentes descriptions à venir.

L'algorithme de recherche locale est une recherche exhaustive. La méthode consiste à tester toutes les MAFs de B_ℓ afin de sélectionner les solutions non-dominées au sens de Pareto pour les critères g_1 et g_2. Les MAFs Pareto-optimales d'un module M_ℓ sont les solutions de rang 1 fournies par l'algorithme 5.2 appliqué à l'espace B_ℓ.

Considérons un module M_ℓ du système par exemple. Chaque $MAF_{\ell,k} \in B_\ell$, avec $k \in [1..\text{card}(B_\ell)]$, est évaluée par le calcul des métriques Q_ℓ et σ_{M_ℓ}. Dès qu'une MAF est dominée au sens de Pareto, elle est retirée de l'espace $B_{\ell,\ell\in[1...m]}$ Seules les solutions Pareto-optimales (c.-à-d.

de rang 1) sont conservées et permettront, dans l'étape d'optimisation globale, de construire des allocations temporelles pour le système complet.

Cette optimisation locale est intéressante car elle permet de réduire considérablement la taille de l'espace des allocations temporelles S issu du produit cartésien des $B_{\ell,\ell\in[1..m]}$. Néanmoins, il est nécessaire de montrer que les MAFs qui ont été écartées localement ne peuvent faire partie d'une allocation temporelle Pareto-optimale. En d'autres termes, il faut s'assurer que l'optimisation locale ne réduit pas la qualité de la recherche globale réalisée par la suite. Nous montrons que c'est bien le cas dans la preuve qui suit. Cette preuve montre par l'absurde qu'une MAF dominée localement (sur un des modules) ne peut faire partie d'une allocation Pareto-optimale compte-tenu de la relation de linéarité qui lie les critères d'optimisation locaux (g_1, g_2) et globaux (f_1, f_2).

Démonstration. Soit une solution $\mathbf{MAF^d}$ dominée localement sur le module M_ℓ. Il existe donc une MAF dans B_ℓ qui est meilleure que $\mathbf{MAF^d}$ sur les deux critères :

$$\exists \mathbf{MAF}^* \in B_\ell \ t.q \quad g_1(\mathbf{MAF}^*) < g_1(\mathbf{MAF^d}) \ \wedge \ g_2(\mathbf{MAF}^*) > g_2(\mathbf{MAF}^{\mathbf{d}}_\ell) \tag{5.5}$$

(Note : on rappelle que l'on cherche à minimiser g_1 et à maximiser g_2.)
Il est possible de construire deux allocations globales S^d et S^*, la premiere utilisant $\mathbf{MAF^d}$ pour le module M_ℓ et la seconde utilisant $\mathbf{MAF}^*$ pour le module M_ℓ. On supposera que toutes les autres MAFs de S^d et S^* sont identiques. Ainsi les allocations globales S^d et S^* ne diffèrent que par le choix de la MAF du module M_ℓ.

Supposons maintenant que l'allocation globale S^d domine l'allocation globale S^* pour les critères globaux f_1 et f_2. Nous allons montrer par l'absurde que ceci n'est pas possible.

Si S^* est dominée par S^d, la relation suivante doit être vérifiée pour être cohérent avec la première condition [1] de la définition de la dominance (5.3) :

$$\forall i \in [1,2] : f_i(S^d) \leq f_i(S^*) \tag{5.6}$$

Nous savons par définition que $f_1 = \frac{1}{m}\sum g_1$ et $f_2 = \frac{1}{m}\sum g_2$, où m est le nombre de modules du système traité.

Si nous prenons la charge f_1 à minimiser, nous avons :

$$f_1(S^d) \leq f_1(S^*)$$

avec $S^d = (MAF_1, MAF_2, \ldots, \mathbf{MAF^d_\ell}, \ldots, MAF_m)$ et $S^* = (MAF_1, MAF_2, \ldots, \mathbf{MAF}^*_\ell, \ldots, MAF_m)$

1. Note : on suppose dans cette définition que les f_i sont tous à minimiser. Il faut changer le sens de l'inégalité quand une des fonctions est à maximiser.

Par définition de f_1, on a :

$$\frac{1}{m}\sum_{k=1}^{m} g_1(MAF_k) \leq \frac{1}{m}\sum_{k=1}^{m} g_1(MAF_k)$$

Comme les MAFs des modules autres que le module M_ℓ sont identiques, on peut extraire l'évaluation du module M_ℓ du reste de la sommation :

$$\sum_{k=1,k\neq\ell} g_1(MAF_k) + g_1(\mathbf{MAF}_\ell^{\mathbf{d}}) \leq \sum_{k=1,k\neq\ell} g_1(MAF_k) + g_1(\mathbf{MAF}_\ell^{*})$$

Il est possible de simplifier cette inégalité à :

$$g_1(\mathbf{MAF}_\ell^{\mathbf{d}}) \leq g_1(\mathbf{MAF}_\ell^{*}) \tag{5.7}$$

Par le même raisonnement de maximisation sur f_2, nous obtenons aussi:

$$g_2(\mathbf{MAF}_\ell^{\mathbf{d}}) \geq g_2(\mathbf{MAF}_\ell^{*}) \tag{5.8}$$

La première condition (5.6) de la définition de dominance est vérifiée pour f_1 et f_2. Il reste à vérifier la seconde condition de cette définition de dominance (5.7). Si S^d domine S^*, nous avons forcément [1] :

$$\exists j \in [1,2] : f_j(S^d) < f_j(S^*) \tag{5.9}$$

Pour que cette inégalité soit vérifiée avec f_1 et f_2, on a

- soit $g_1(\mathbf{MAF}_\ell^{\mathbf{d}}) < g_1(\mathbf{MAF}_\ell^{*})$,
- soit $g_2(\mathbf{MAF}_\ell^{\mathbf{d}}) > g_2(\mathbf{MAF}_\ell^{*})$,
- soit $g_1(\mathbf{MAF}_\ell^{\mathbf{d}}) < g_1(\mathbf{MAF}_\ell^{*})$ et $g_2(\mathbf{MAF}_\ell^{\mathbf{d}}) > g_2(\mathbf{MAF}_\ell^{*})$.

Ceci ne peut être vrai que si $\mathbf{MAF}_\ell^{\mathbf{d}}$ domine $\mathbf{MAF}_\ell^{*}$, ce qui est en contradiction avec l'hypothèse que $\mathbf{MAF}_\ell^{*}$ domine $\mathbf{MAF}_\ell^{\mathbf{d}}$ réalisée en (5.5). □

5.3.2 Prise en compte de la similarité des MAFs

Après optimisation locale, l'ensemble des solutions Pareto-optimales par module est relativement petit. Dans cet ensemble, on trouve néanmoins des MAFs différentes qui ont exactement les mêmes performances locales. Ces MAFs présentent donc la même charge et la même marge réseau moyenne. Les MAFs équivalentes sont présentes régulièrement car dans la construction des ensembles de MAFs admissibles, on peut créer, entre autres, des MAFs qui présentent des permutations des partitions qu'elles hébergent. Ces permutations n'ont pas d'impact sur les critères de performance définis ici.

1. dans cette définition, les fonctions f_j sont à minimiser

Formellement, deux MAF, MAF_i et MAF_j, appartenant au même module M_ℓ, sont équivalentes (notation $\sim$) si et seulement si :

$$MAF_i \sim MAF_j \Leftrightarrow Q_\ell^i = Q_\ell^j \wedge \sigma_{M_\ell}^i = \sigma_{M_\ell}^j \tag{5.10}$$

Un ensemble de MAFs $X \subset B_\ell$ d'un module M_ℓ est une *classe d'équivalence* X si les MAFs la composant admettent la relation d'équivalence $\sim$. Il peut exister plusieurs classes d'équivalence dans l'ensemble B_ℓ.

Afin de réduire l'ensemble des MAFs Pareto-optimales de B_ℓ, et ainsi réduire encore plus l'ensemble des allocation temporelles candidates S, nous procédons à une réduction de l'espace des MAFs Pareto-optimales en ne gardant qu'une seule MAF par classe d'équivalence X dans B_ℓ. Il est donc possible de ne garder qu'une MAF par classe d'équivalence et réduire ainsi les ensembles B_ℓ, $\ell \in [0...m]$. Dans la suite nous gardons la notation B_ℓ pour désigner les MAFs candidates après application de l'optimisation multicritere locale et de la réduction par similarité des MAFs.

5.4 Optimisation globale : algorithme EXHAUST

La seconde étape d'optimisation consiste en la recherche des allocation temporelles Pareto-optimales du système IMA complet. De la même façon que pour le niveau d'optimisation multiobjectif par recherche locale présenté en 5.3, le calcul des allocations Pareto-optimales est réalisé selon l'algorithme d'assignation des rangs de Pareto 5.2. Dans cet algorithme, nous conservons uniquement les allocations de rang 1, i.e. les solutions non-dominées. L'algorithme EXHAUST énumère pour cela toutes les solutions, calcule leur critères f_1 et f_2, et élimine toutes les solutions dominées de l'ensemble des allocations possibles S pour obtenir *le front de de Pareto théorique* dénoté F_{PT}. En effet, les solutions Pareto-optimales obtenues par EXHAUST sont les solutions Pareto-optimales exactes du problème d'allocation : il n'existe pas de solution meilleure que celles-ci dans l'ensemble des solutions candidates S.

L'heuristique EXHAUST est un algorithme *optimal* car il trouve toutes les solutions Pareto-optimales du problème. Le désavantage de ce type de méthode est son temps de calcul proportionnel à l'espace de recherche. Or, l'espace de recherche S de notre problème grandit exponentiellement avec le nombre de modules m et le nombre de partitions par modules.

5.5 Optimisation globale : algorithme multicritere TABOO

Nous verrons qu'en pratique (cf. Chapitre 6), la réduction du nombre de partitions par module réalisée lors de la première étape d'optimisation locale nous permet de résoudre un système d'une vingtaine de modules à l'optimal. La recherche exhaustive multiobjectif nécéssite un temps de traitement important et une capacité mémoire considérable. En effet, cette heuristique ne s'arrête

pas avant d'avoir testé toutes les solutions avec la dominance au sens de Pareto. L'intérêt de cette méthode est la possibilité (en fonction des ressources de calcul et de mémoire disponibles) de ressortir les solutions Pareto-optimales exactes. Malheureusement, il n'existe pas d'heuristique multicritère autre que la recherche exhaustive pour résoudre le problème d'allocation temporelle à l'optimal.

Toutefois, l'intégrateur n'est pas forcément intéressé par toutes les solutions Pareto-optimales. Il pourra se satisfaire d'un ensemble de solutions de qualité qu'il est capable de calculer en un temps raisonnable. Dans ce cas de figure, il est possible d'utiliser des algorithmes de recherche *approchées*. Dans le domaine de l'optimisation multiobjectifs, différentes métaheuristiques ont été appliquées avec succès [55]. On retrouve plusieurs solutions basées sur des algorithmes génétiques [61], des métaheuristiques de type recuite simulé [62], Tabou [58] ou encore exploitant des colonies de fourmies [63].

Pour passer à l'échelle et fournir des allocations temporelles à des systèmes plus grands, nous avons choisi d'adapter une de ces métaheuristiques pour extraire un front de Pareto proche de l'optimal, en un temps raisonnable. Nous nous sommes basés sur une version multicritère de l'heuristique Tabou issue des travaux de thèse de Katia Jaffrès-Runser [57][58][60].

5.5.1 La recherche Tabou.

La recherche Tabou est une métaheuristique proposée par Fred Glover [59] pour l'optimisation mono-critère. La recherche Tabou est une recherche locale qui explore l'espace des solutions en faisant évoluer une solution "de proche en proche". A chaque mouvement effectué, on le mémorise dans une liste Tabou de longueur finie. Les mouvements de la liste Tabou sont interdits pendant un certain temps, afin de réduire les chances de converger prématurément vers un optimum local. Elle est considérée comme une alternative à la métaheuristique dénommée *recuit simulé*[48].

Son fonctionnement consiste à démarrer d'une configuration quelconque (solution) s_i où à chaque itération de l'algorithme, un ensemble de *voisins* de s_i est construit. L'ensemble des voisins est un sous-ensemble de S, et est composé des configurations atteignables en un seul *mouvement* élémentaire à partir de s_i [64]. Les solutions du voisinage de s_i sont par la suite évaluées selon une fonction objectif f du problème pour trouver un optimum local. Pour éviter de réévaluer les mêmes solutions, les mouvements vers ces optimums sont stockés dans une liste *Tabou* de mouvements interdits pour une durée fixée, forçant ainsi l'algorithme à explorer d'autres solutions. Cette liste stocke les mouvements inverses qui mèneraient à revenir à la solution précédente s_i *i.e* de s_{i+1} vers s_i. Il est à noter que la solution optimal s_{i+1} obtenue à la fin de l'itération i est adoptée même si la fonction objectif $f(s_i)$ est meilleure que $f(s_{i+1})$. En effet, cette propriété est celle qui permet à l'algorithme d'éviter le problème des minima locaux et de réaliser des cycles. La recherche par la suite continue dans un espace des solutions restreint qui sont atteignables sans utiliser un mouvement de la liste Tabou. La recherche Tabou s'arrête lorsqu'un nombre d'itérations donné sont réalisées sans amélioration de la solution en cours.

Cette heuristique se distingue par l'introduction d'un peu de bon sens afin que celle-ci ne

prenne pas des décisions et des directions de recherche basées uniquement sur le hasard. En effet, la recherche Tabou introduit une mémoire afin de constituer un historique de la recherche locale itérative. Nous avons choisi l'adaptation de cette heuristique au multiobjectif car il est aisé de définir un ensemble de solutions candidates. Celle-ci évite l'écueil des minima locaux et utilise la notion de voisinage précisée ci-après. Cette heuristique n'a pas pour objectif de tester toutes les solutions candidates de S puisqu'elle est dérive d'une méthode d'optimisation approchée.

La métaheuristique Tabou a été définie pour faire de l'optimisation mono-objectif. Ainsi, pour pouvoir utiliser l'heuristique Tabou en considérant plusieurs critères pour notre problème P (cf. éq. (5.4)), nous avons utilisé une version multiobjectif de Tabou présentée dans [57]. Dans cette adaptation, la notion de dominance est utilisée pour permettre à l'intégrateur de choisir parmi plusieurs solutions optimales, le compromis qu'il serait prêt à faire. L'heuristique initiale [57] a été proposée dans les travaux de thèse de Katia Jaffres-Runser [56] pour résoudre un problème de planification de réseaux locaux sans-fil. Nous allons montrer dans la suite comment nous avons adapté l'heuristique multicritère Tabou de [57] à la résolution de notre problème d'allocation temporelle en IMA.

5.5.2 Fonctionnement de l'heuristique TABOO

La métaheuristique Tabou multicritère [57] lance K recherches Tabou en parallèle et compare toutes les solutions testées avec la dominance au sens de Pareto. L'algorithme construit ainsi lors de sa recherche un ensemble de solutions non dominées appelé *front de Pareto*. Ce front de Pareto évolue à chaque itération, en fonction des solutions testées par les K recherches Tabou parallèles. A la fin de la recherche, ce front comporte l'ensemble des solutions non-dominées (au sens des critères f_1 et f_2) extraites de l'ensemble de toutes les solutions testées. Il représente donc une version approchée de l'ensemble Pareto-optimal du problème. Si le front de Pareto obtenu avec TABOO est identique au front de Pareto obtenu avec la recherche EXHAUST, la recherche Tabou a convergé vers le front optimal. L'algorithme complet est décrit dans la Figure 5.3 et décrit dans la suite de ce chapitre.

Notations Les notations suivantes sont adoptées :

S	L'ensemble de recherche des allocations valides.
K	Le nombre de recherches Tabou lancées en parallèle.
F_{PP}	Le front de Pareto
$F_c(i)$	Le front de recherche courant à l'itération i contenant K solutions courantes
s_c^k	la k^{eme} solution courante du front courant $F_c(i)$
$V(s_c^k)$	Le voisinage de la solution s_c^k
$ND(V(s_c^k))$	L'ensemble des solutions non-dominées du voisinage $V(s_c^k)$
L_{tab}^k	La liste Tabou de la solution courante k
L_{tab}	Toutes les listes Tabou

L'algorithme de la Figure 5.3 présente les principales étapes d'une itération de TABOO.

```
Pour une itération i:
    Pour k = 1 à K, faire :
      Si (i==1) {
        -- Initialiser la k^eme solution courante en choisissant aléatoirement
        -- une solution dans S
        s_c^k = Tirer_Alea(S)
      }
      -- Calculer le voisinage de s_c^k
      V(s_c^k) = Calculer_Voisinage(s_c^k, L_tab^k)
      -- Évaluer les solutions du voisinage de s_c^k
      Pour toute solution s ∈ V(s_c^k) :
        Pour chaque critère c_j :
          Évaluer(s,c_j)
        FinPour
      FinPour
      -- Extraire les solutions non-dominées de rang ≤ r
      ND(V(s_c^k)) = ExtraireND(V(s_c^k),r)
      -- Mettre à jour le front de Pareto
      F_PP = F_PP ∪ ND(V(s_c^k))
      -- Choisir la nouvelle solution courante numéro k
      F_c(i+1) = Tirer(s_c^k(i+1),ND(V(s_c^k)))
    FinPour
    F_PP = ExtraireND(F_PP,1)
    -- Mettre à jour toutes les listes Tabou
    MAJ(L_tab,F_c(i+1))
FinPour
```

FIGURE 5.3 – Algorithme de l'heuristique multiobjectif Tabou

Les modèles suivants ont été définis pour adapter la métaheuristique à notre problème :

Modèle d'une solution s. Une solution s est une allocation temporelle candidate pour le problème d'optimisation. s est représenté par un vecteur de taille m. L'élément d'indice ℓ est un entier qui donne le numéro de la MAF admissible du module M_ℓ. Cet entier appartient à l'intervalle [1, $Card(B_\ell)$], avec B_ℓ l'ensembel des MAFs admissibles du module M_ℓ.

Par exemple, $s = (3, 2, 4)$ représente une allocation temporelle d'un système composé de 3 modules, où les MAFs allouées sont respectivement :

- Pour le module 1 : la MAF d'indice 3 de B_1 ($MAF_3 \in B_1$),
- Pour le module 2 : la MAF d'indice 2 de B_2 ($MAF_2 \in B_2$),
- Pour le module 2 : la MAF d'indice 4 de B_4 ($MAF_4 \in B_3$).

Relation de voisinage. Deux solutions s et s' sont voisines si et seulement si la norme du vecteur $|s - s'|$ vaut 1. En d'autres termes, une seule des coordonnées diffère entre s et s', et la différence pour cette coordonnée vaut 1, en valeur absolue.

Par exemple, $s = (3, 2, 4)$ est voisine de $s' = (2, 2, 4)$ et de $s'' = (4, 2, 4)$ pour le module 1.

Modèle de mouvement. Nous définissons le mouvement comme l'incrémentation ou la décrémentation de 1 d'une des coordonnées du vecteur de s.

Par exemple, $s' = (2, 2, 4)$ est une solution voisine à $s = (3, 2, 4)$ obtenue par un mouvement de décrémentation de 1 de l'indice du module 1.

Définition du voisinage $V(s)$. A toute solution s de l'ensemble des solutions candidates S, nous associons l'ensemble $V(s)$ des solutions voisines de s. Cet ensemble est créé par toutes les solutions voisines de s au sens de la relation de voisinage définie ci-avant. Ces solutions sont obtenues en faisant varier un et un seul élément à la fois en effectuant un mouvement positif (+1) s'il existe, et d'un mouvement négatif (-1) si il existe dans un ensemble B_ℓ, $l \in [1..m]$.

Par exemple, si $s_c^k = (MAF_{1,4}, MAF_{2,3})$, l'ensemble $V(s_c^k)$ est composé des éléments $(MAF_{1,3}, MAF_{2,3})$, $(MAF_{1,5}, MAF_{2,3})$, $(MAF_{1,4}, MAF_{2,2})$ et $(MAF_{1,4}, MAF_{2,4})$. Le voisinage est au maximum de taille $(m \times 2)$.

Le front de recherche courant $F_c(i)$. L'algorithme lance K recherches Tabou parallèles. A chaque recherche k correspond une solution courante s_c^k. L'ensemble des k solutions courantes s_c^k compose le front de recherche courant de l'algorithme TABOO.

Mise à jour du front de recherche courant $F_c(i)$ Le front courant $F_c(i)$ évolue à chaque itération i. Toutes les k solutions sont mises à jour de la façon suivante.

Le voisinage de chaque solution s_c^k du front courant est construit. Toutes les solutions de ces voisinages sont alors évaluées par f_1 et f_2. Pour chaque voisinage, l'ensemble des solutions non-dominées de rang inférieur ou égal à r, dénommé $ND(V(s_c^k))$, est extrait. Cet ensemble comporte une ou plusieurs solutions. La nouvelle solution s_c^{k+1} est tirée aléatoirement de $ND(V(s_c^k))$ et ajoutée à $F_c(i+1)$.

Extraction des solutions Pareto optimales F_{PP}. À chaque itération, le front de Pareto optimal F_{PP} est réévalué. Il est obtenu par à partie de l'union du front de Pareto optimal de l'itération précédente et des ensembles $ND(V(s_c^k)) \in [1..K]$. L'union de ces ensembles comporte des solutions dominées. Elles sont écartées en appliquant sur cet ensemble une extraction des solutions non-dominées. Le front F_{PP} comporte à l'itération i l'ensemble de solutions non-dominées découvertes jusque là.

Les listes Tabou. Chaque solution courante s_c^k possède une liste Tabou qui lui est propre dénommée L_{tab}^k. Pour créer une liste Tabou, il faut déterminer quels sont les mouvements ou les

solutions interdites car déjà testées. Nous avons choisi le comportement suivant de la liste Tabou. Lors de la mise à jour du front courant, la solution courante s_k est remplacée par une solution voisine. A ce moment, la liste Tabou L^k_{tab} enregistre la nouvelle MAF de s_k.

Par exemple, si la solution courante $s = (3, 2, 4)$ est remplacée par la voisine$s' = (2, 2, 4)$, nous avons $MAF_{1,3}$ qui devient $MAF_{1,2}$. La nouvelle $MAF_{1,2}$ est alors enregistrée dans la liste Tabou correspondante. Cette liste est de type FIFO. A chaque ajout de solution dans liste, la plus ancienne est supprimée. La liste Tabou est de taille fixe, mais nous pouvons considérer une évolution de l'heuristique afin de manipuler des listes Tabous de tailles dynamiques. La taille d'une liste dynamique est chosie aléatoirement à chaque itération.

Lors de la construction du voisinage d'une solution s^k_c, il faut prendre en compte les informations contenues dans la liste Tabou L^k_{tab}. Plus spécifiquement, une solution voisine contenant une des MAFs appartenant à L^k_{tab} ne peut être générée.

Initialisation du front de recherche La difficulté des métaheuristiques, comme la recherche Tabou, réside dans le choix des solutions de départ pour la recherche. Il n'existe pas une règle générale qui permette de guider le choix de la solution de départ favorisant une convergence rapide vers le front Pareto-optimal. La solution de départ est communément choisie de manière aléatoire.

Conditions d'arrêt de la recherche L'algorithme Tabou multiobjectif proposé présente deux critères d'arrêt. La recherche se termine si un nombre maximal d'itérations est atteint ou si un nombre maximal d'itérations sans amélioration ou modification de F_{PP} a eu lieu.

5.6 Conclusion

Dans ce chapitre, nous avons présenté deux heuristiques multiobjectif pour l'aide à l'allocation temporelle des systèmes IMA. La particularité de ces heuristiques repose sur deux niveaux d'optimisation, niveau local au module et niveau global au système. La première heuristique, qui est une méthode optimale, repose sur une recherche multiobjectif qui teste toutes les solutions candidates du problème d'allocation temporelle, et extrait par dominance au sens de Pareto les solutions ayant les meilleurs compromis. Cette heuristique est adaptée aux systèmes de taille moyenne. La seconde heuristique est une recherche Tabou multiobjectif plus adaptée aux systèmes de grande taille. Elle permet de converger rapidement vers la surface de compromis et la possibilité d'arrêter l'algorithme à tout moment.

Dans le chapitre suivant nous présentons le résultat des algorithmes EXHAUST et TABOO sur deux systèmes IMA de taille différentes. Une analyse des résultats et une comparaison des performances y est réalisée.

6 Validation des heuristiques proposées

Le premier objectif de ce chapitre est de comparer les performances de la recherche tabou multiobjectif et de la recherche exhaustive multiobjectif. A cette fin, nous procédons à une analyse des performances des deux méthodes d'optimisation pour deux architectures IMA données. Nous commençons par montrer la validité de notre optimisation en deux temps (optimisation locale puis globale) sur un système composé d'une quinzaine de modules. Dans un second temps, nous comparons les résultats obtenus par TABOO et EXHAUST sur des exemples de systèmes composés de 15 et 20 modules. L'avantage de la recherche exhaustive est son optimalité. Ainsi, si les ressources de calcul et de mémoire le permettent, les solutions Pareto-optimales sont calculables. Le premier objectif de ce chapitre est de montrer que TABOO fournit des solutions intéressantes, proches du front optimal, et ce en un temps de recherche limité.

Le second objectif de ce chapitre est de montrer la validité de notre modélisation multi-critères par l'analyse des solutions du front obtenu après la recherche. Une répartition des solutions non-dominées en différentes classes est proposée afin de présenter un ensemble de solutions plus petit à l'intégrateur. En effet, le front de Pareto, qu'il soit obtenu par EXHAUST ou par TABOO, possède un nombre important de solutions. Pour simplifier l'analyse de ces solutions, il faut en extraire les plus significatives.

6.1 Architectures de test

Notre modélisation et nos algorithmes sont testés sur deux architectures de système IMA. La première architecture, **Exemple 1**, est composée de 15 modules IMA, interconnectés à travers un réseau AFDX. Cette architecture est détaillée dans 6.1.1. C'est une architecture de taille moyenne qui peut être solutionnée par une recherche exhaustive. Cette architecture a été choisie pour que l'on puisse à la fois calculer le *front de Pareto théorique* F_{PT} avec EXHAUST, et le *front de Pareto pratique* F_{PP} avec TABOO. Il est ainsi possible de comparer les deux fronts obtenus, mais aussi l'évolution du front de Pareto pratique au cours de la recherche TABOO.

La seconde architecture, **Exemple 2**, est composée de 20 modules, interconnectés à travers un réseau AFDX. Cette architecture est détaillée dans la section 6.1.2. L'architecture de l'exemple 2 est l'architecture la plus grande que nous ayons pu traitée avec EXHAUST avec un ordinateur standard, avec une configuration de 3.07Ghz et 8Go de RAM. La principale limitation

qui nous a empêché de tester des environnement plus grands avec EXHAUST est la mémoire. En effet, l'espace des solutions explose et sa manipulation (calcul de non-dominance) nécessite beaucoup de mémoire. Nous appliquons l'algorithme TABOO sur Exemple 2, afin de connaitre ses performances pour un système IMA plus large.

6.1.1 Exemple 1 : 15 CPM

Dans le tableau 6.1.1 sont détaillés les composants et les configurations temporelles de Exemple 1. Exemple 1 est une architecture IMA composée de 15 CPMs qui hébergent des partitions communicantes. Les partitions distantes communiquent via des VLs AFDX. Elles sont asynchrones.

	Intervalles - Durée(ms)
Nombre de CPMs	15
Nombre de partitions	55
Nombre de partitions par module	entre 3 et 5
Nombre de communications entrantes par modules	entre 1 et 4
Nombre de communications	36
Durées d'exécution des partitions	$b_i \in [5..15]$
Périodes d'exécution des partitions sources	$T_i \in [40..240]$
Paramètres de fraicheur	$FP \in [38..110]$
Bornes meilleur cas pour le débit réseau	$L_{min} \in [1..4]$
Borne pire cas pour le débit réseau	$L_{max} \in [5..20]$

TABLEAU 6.1 – Paramètres et composants de l'Exemple 1

6.1.2 Exemple 2 : 20 CPM

Dans le tableau 6.1.2 sont détaillées les composantes et les configurations temporelles de Exemple 2. Exemple 2 est une architecture IMA composée de 20 CPMs qui hébergent des partitions communicantes. Les partitions communiquent de manière inter-modulaire et sont interconnectées par des VLs AFDX.

6.2 Outils d'analyse des fronts de Pareto

Pour pouvoir réaliser les analyses suivantes, nous introduisons deux outils que sont la distance générationelle et le calcul de classes d'équivalences. Ces deux outils nous permettront soit de connaitre la 'distance' entre deux fronts de Pareto, soit de connaitre la répartition des solutions sur ce front.

	Intervalles - Durée(ms) (ms)
Nombre de CPMs	20
Nombre de partitions/Tâches	73
Nombre de partitions par module	entre 3 et 5
Nombre de communications entrantes par modules	entre 1 et 4
Nombre de communications	50
Durées d'exécution des partitions	$b_i \in [5..15]$
Périodes d'exécution des partitions sources	$T_i \in [40..240]$
Paramètres de fraicheur	$FP \in [38..110]$
Bornes meilleur cas pour le débit réseau	$L_{min} \in [1..4]$
Borne pire cas pour le débit réseau	$L_{max} \in [5..20]$

TABLEAU 6.2 – Paramètres et composants de l'Exemple 2

6.2.1 Distance générationelle

Nous allons procéder dans la suite à l'analyse des performances de la recherche tabou multiobjectif. Pour cela, nous allons comparer le front de Pareto pratique F_{PP} au front de Pareto théorique F_{PT} obtenu par la recherche exhaustive. Pour pouvoir quantifier la distance entre les deux fronts, nous utilisons mesure de *distance générationelle* définie par Van Veldhuizen [65].

Définition. La distance générationelle permet dans ces travaux de calculer la distance entre deux fronts Pareto, et donc plus spécifiquement de calculer la distance entre un front de Pareto pratique et le front de Pareto théorique. La distance générationelle G est calculée selon la formule suivante :

$$G = \frac{(\sum_{i=1}^{n} d_{i,min}^2)^{\frac{1}{2}}}{n} \tag{6.1}$$

où :

- n est le nombre d'éléments dans l'ensemble F_{PP}
- $d_{i,min} = \min_{j \in F_{PT}}(d(s_i, s_j))$ est la distance entre la solution i de F_{PP} et la solution la plus proche géométriquement de F_{PT}, dans l'espace des fonctions de coût.

La distance d dans l'espace des fonctions de coût entre une solution s_i et une solution s_j est calculée sous sa forme normalisée selon la formule suivante :

$$d_i(s_i, s_j) = \sqrt{\sum_{q=1}^{n} \left(\frac{f_q(i) - f_q(j)}{\max(f_q) - \min(f_q)} \right)^2} \tag{6.2}$$

où n représente le nombre de fonctions de coût optimisées, $\max(f_q)$ et $\min(f_q)$ la valeur maximale et minimale de la fonction de coût f_q, respectivement.

Plus les deux fronts sont éloignés dans l'espace des fonctions de coût, plus la valeur de la distance générationelle est grande. Si cette distance est nulle, les deux fronts sont identiques.

6.2.2 Classes d'équivalence

Nous verrons qu'il est très courant de trouver des solutions qui ont une évaluation identique dans les front de Pareto que nous allons manipuler. Ces solutions sont pourtant différentes. Le fait qu'elles présentent des métriques identiques est une conséquence de l'existence d'une symétrie dans l'espace des allocations temporelles candidates.

Pour l'analyse des fronts de Pareto obtenus, nous allons regrouper les solutions qui présentent une même évaluation pour les deux fonctions de coût. Nous appelons les ensembles ainsi créés des *classes d'équivalence*, où chaque classe rassemble les solutions dont les performances sont identiques pour les métriques considérées. En d'autres termes, deux solutions s_i et s_j appartiennent à la même classe d'équivalence si la condition suivante est vérifiée :

$$s_i \sim s_j \ ssi \ \forall f_q, q \in [1...n], f_q(s_i) = f_q(s_j) \tag{6.3}$$

Il est ainsi possible de proposer différents compromis à l'installateur de cette façon. Pour chaque classe d'équivalence, il suffit de présenter une seule solution à l'intégrateur, représentative du compromis entre les critères de la classe.

Cette notion de classe d'équivalence est importante pour notre évaluation car elle nous permettra de vérifier que tous les compromis du front de Pareto théorique sont présents dans le front de Pareto pratique par exemple. Si l'on représente le front de Pareto dans l'espace des fonctions de coûts, le nombre de points obtenus est égal au nombre de classes d'équivalences.

6.3 Validité de l'optimisation locale

Nous avons montré théoriquement dans le chapitre 5 que la réduction de l'ensemble des allocations temporelles par l'optimisation locale des MAFs candidates ne peut supprimer, avec les critères choisis, de solution globale Pareto optimale. Nous rappelons que cette optimisation locale consiste à ne garder que les MAFs Pareto-optimales dans l'espace des MAFs canditates d'un module, au sens des métriques d'optimisation locales.

La réduction des MAFs candidates par similarité a aussi été introduite pour réduire la cardinalité de l'espace des allocations possibles. Dans ce cas, on supprime de l'espace des MAFs du module toutes les MAFs qui présentent la même évaluation sur les deux critères. Cette réduction se rajoute à l'optimisation locale précédente. Nous allons montrer pour l'exemple 1 à 15 modules que les solutions Pareto optimales obtenues avec et sans réduction des MAFs sont identiques, cette fois-ci par calcul exhaustif et non par le raisonnement.

Pour cela, nous allons lancer une recherche exhaustive sur les ensembles de solution suivants :

1. l'espace S complet, sans réduction ;
2. l'espace S avec optimisation locale ;
3. l'espace S avec optimisation locale et réduction par similarité.

Ces recherches nous permettront de comparer les fronts de Pareto obtenus pour chaque ensemble. L'objectif est de montrer que ces choix (optimisation locale et réduction par similarité) offre la même "qualité" de front de Pareto. Un front de qualité est un front qui comporte tous les compromis possibles entre les critères d'optimisation.

Pour obtenir les résultats ci-après, l'instance du problème traité est donné par :

- Système testé: Exemple 1,
- Nombre de critères pour le niveau d'optimisation des allocations temporelles: 2,
- Critères d'optimisation des allocations temporelles : charge moyenne des modules du système (Q_{avg}), et marge moyenne des partitions destinations du système Δ_{avg},
- Nombre de critères pour le niveau d'optimisation locale : 2,
- Rang de de Pareto : 1,
- Critères pour le niveau d'optimisation locale : charge moyenne d'un module (Q_l), et moyenne des marges des partitions destinations (σ_{M_l})

Le Tableau 6.3 résume les résultats de l'impact de l'optimisation locale et de la réduction des MAFs similaires sur l'ensembles des allocations candidates, le temps de calcul et la taille des fronts de Pareto. Il donne également le nombre de classe d'équivalences trouvé. Ce nombre s'interprête aussi comme étant le nombre de compromis différents entre les fonctions de coûts qui composent le front.

	Sans réduction	Optimisation locale	Réduction similarité
Nombre d'allocations possibles	32M	2,4M	0,7M
Taille du front Pareto optimal	4155	4155	1354
Nombre de classes	56	56	56
Distance générationelle	/	0	0
Durée de l'algorithme	∼6h	28 mins	2,54 mins

TABLEAU 6.3 – Performances de l'optimisation locale et la réduction par similarité sur l'algorithme EXHAUST - Exemple 1

Les résultats présentés dans le tableau 6.7 montrent clairement que les réductions opérées par l'optimisation locale avec suppression des MAFs similaires est très intéressante. Elle permet de réduire la taille de l'espace des solutions d'un facteur 45. Le nombre des solutions candidates sans réduction est de 32 348 160. Il passe à 2 359 296 solutions avec l'optimisation locale et à seulement 746 496 solutions si on intègre la suppression des MAFs similaires sur les modules.

Validité de l'optimisation locale. Si l'on analyse le nombre de solutions du front de Pareto obtenu, on observe que l'optimisation locale fournit le même front que l'espace S sans réduction. De plus, ce front exhibe le même nombre de classes d'équivalences et une distance générationelle nulle. En d'autre termes, l'optimisation locale fournit exactement le même front que la recherche exhaustive sur l'espace sans réduction. Ce constat rejoint la preuve théorique présentée au chapitre précédent.

La Figure 6.1 représente les 2 359 296 solutions candidates à l'allocation temporelle de Exemple 1 (illustrées par des croix rouges), dans l'espace des critères de charge moyenne Q_{avg} et de marge réseau Δ_{avg}. Le front de Pareto obtenu est représenté par des croix bleues.

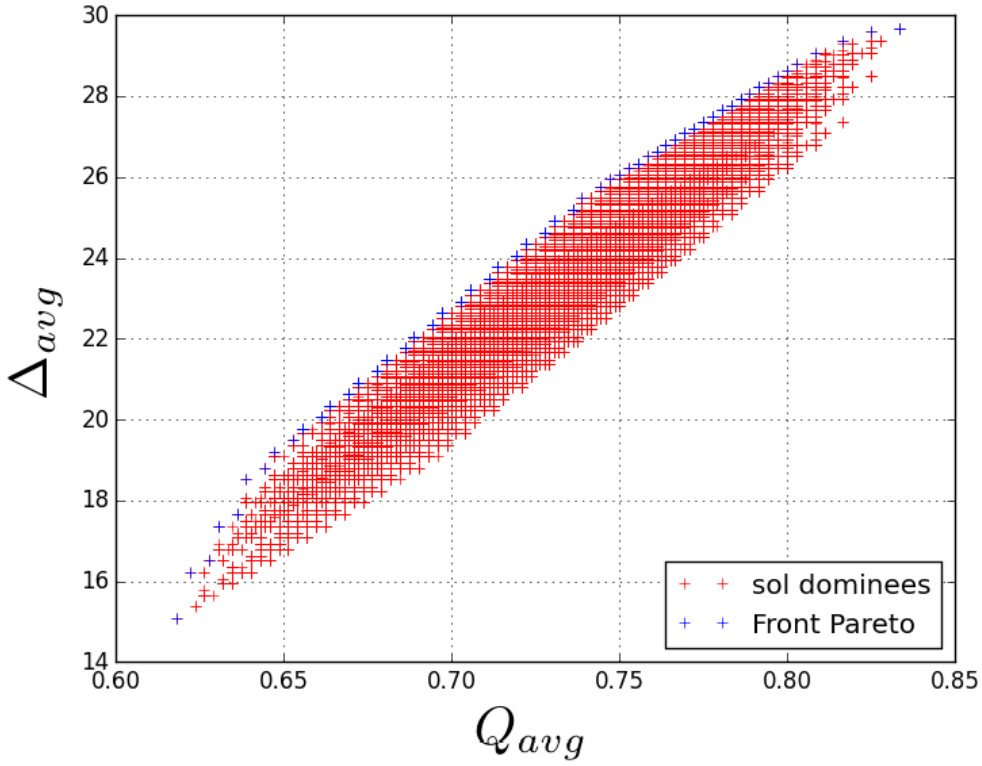


FIGURE 6.1 – Évaluation des solutions avec optimisation locale, *sans* réduction des MAFs similaires- Exemple 1

Validité de la réduction par les MAFs similaires. La réduction par les MAFs similaires réduit le nombre de solutions du front de Pareto. Ainsi, toutes les solutions Pareto-optimales ne sont pas présentes. Néanmoins, on observe que la distance générationelle est nulle. Ceci signifie que toutes les solutions obtenues par réduction des MAFs appartiennent au front de Pareto sans réduction. Un autre point très intéressant, est qu'on retrouve le même nombre de classes de similarité. Autrement dit, le front après réduction des MAFs présente la même qualité que le front sans réduction : tous les compromis y sont représentés.

La Figure 6.2 représente les 746 496 solutions candidates obtenues après optimisation locale et réduction des MAFs similaires, dans l'espace des critères de charge moyenne Q_{avg} et de marge réseau Δ_{avg}. Le front de Pareto obtenu est représenté par des croix bleues.

L'ensemble des solutions candidates a été réduit à seulement 746 496 solutions candidates. La Figure 6.3 montre l'adéquation parfaite du front de Pareto théorique de la recherche avec

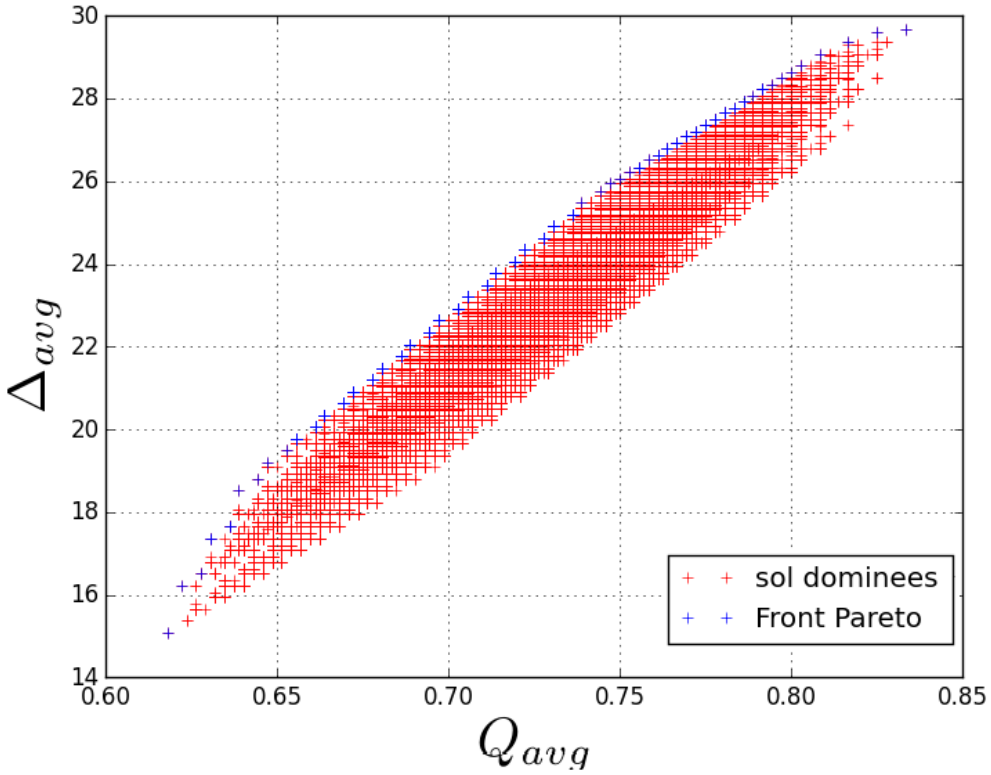


FIGURE 6.2 – Évaluation des solutions avec optimisation locale, *avec* réduction des MAFs similaires

réduction des MAFs (croix rouges) et sans réduction des MAFs (croix bleues).

Dans la suite, nous allons analyser le front optimal théorique F_{PT}. Celui-ci est égal au front de Pareto pratique obtenu par recherche exhaustive, sans réduction par similarité des MAFs. Il est représenté sur la Figure 6.1 de l'Exemple 1.

6.4 Analyse du Front de Pareto optimal

Nous réalisons dans cette section une analyse des solutions du front de Pareto théorique F_{PT}. Pour cela, nous analysons la répartition des solutions dans les classes d'équivalence. Puis, nous regardons plus en détails les caractéristiques des solutions qui appartiennent à une même classe d'équivalence. L'objectif de cette étude est de montrer que le découpage en classes d'équivalence est pertinent, et permet d'analyser efficacement un front de Pareto pour notre problème d'allocation temporelle.

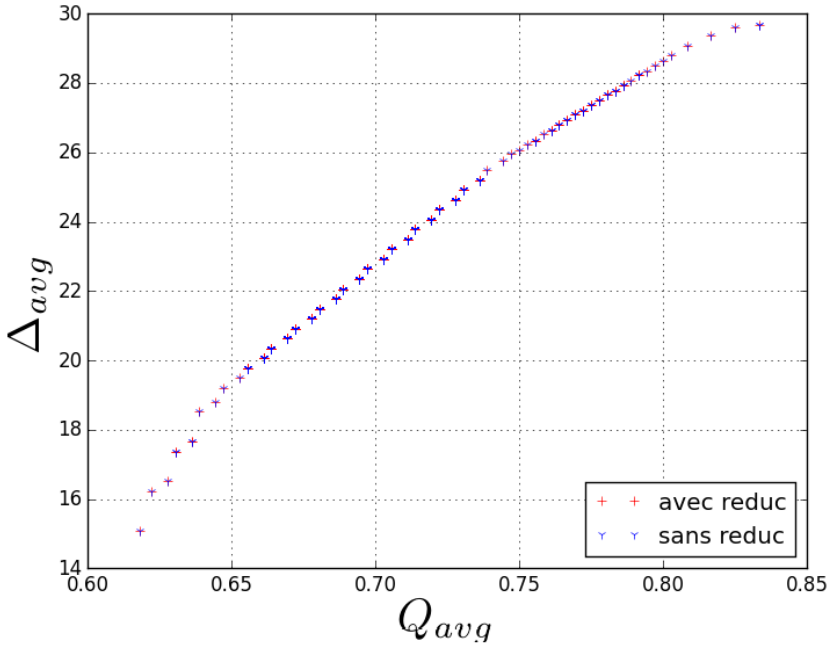


FIGURE 6.3 – Front de Pareto optimal avec réduction des MAFs et Front de Pareto optimal sans réduction des MAFs - Exemple 1

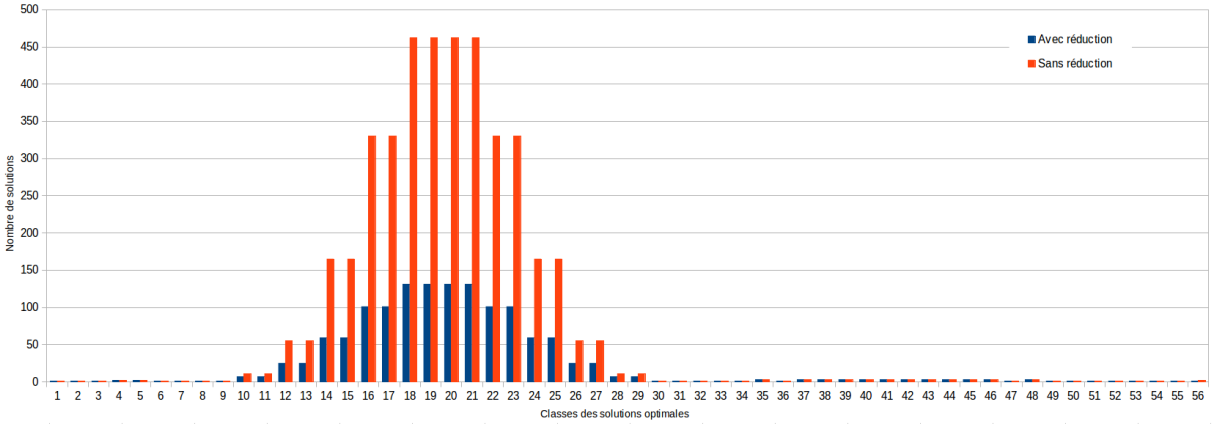


FIGURE 6.4 – Distribution des solutions de F_{PT} dans les classes d'équivalence, avec et sans réduction des MAFs similaires - Exemple 1

6.4.1 Analyse des classes d'équivalence

La Figure 6.4 illustre la distribution des solutions optimales par classe, ordonnées selon la charge Q_{avg} des charges des modules. Les évaluations de ces solutions sont représentées sur la Figure 6.3. Les bâtonnets oranges représentes la distribution du front Pareto optimal sur les 56 classes de solutions. Ils représentent le résultat de l'optimisation par recherche exhaustive sans réduction des MAFs. Les bâtonnets bleus représentent les classes des solutions optimales avec l'application de la réduction des MAFs.

Les classes représentées ont des métriques identiques: la classe 1 regroupe une solution dont les métriques sont (Δ_{avg},Q_{avg})=(0.62,15.08). Cette classe propose l'allocation temporelle qui charge le moins possible les CPM, au détriment de la marge dans le réseau. En d'autres termes, les modules de Exemple 1 pourront accepter 48% de charge supplémentaire, mais la marge dans le réseau n'est que de 15.08 en moyenne. La classe 20 regroupe 131 solutions dont les métriques proposent un compromis équilibré (Q_{avg},Δ_{avg})= (0.7,22.9). Enfin la classe 56 propose les métriques (Q_{avg},Δ_{avg})= (0.83,29.65), composée d'une seule solution optimale. Celle ci propose le compromis le plus discriminant pour la métrique Q_{avg}, mais offre la marge réseau moyenne Δ_{avg} maximale possible. Le Tableau 6.7 résume les performances des classes 1, 20 et 56.

	Charge moyenne CPMs	Marge dans le réseau	Nombre de solutions
Classe 1	0.62	15.08	1
Classe 20	0.7	22.9	131
Classe 56	0.83	29.65	1

TABLEAU 6.4 – Quelques solutions optimales par recherche exhaustive multiobjectif - Exemple 1

Toutes les solutions d'une même classe présentent la même évaluation des critères. Certaines classes possèdent un nombre très important de solutions. On peut se demander maintenant si ces solutions sont vraiment différentes structurellement ? Et comprendre pourquoi elles offrent un même compromis.

6.4.2 Analyse des solutions d'une même classe d'équivalence

Nous allons étudier plus en détails les solutions contenues dans une même classe d'équivalence. Cette étude permet de déterminer pourquoi ces solutions ont la même évaluation. Nous nous concentrons sur la classe 20 qui regroupe 131 solutions Pareto optimales.

Prenons deux solutions de la classe 20:

1. s_1=(1,0,0,1,0,0,**0,3**,0,1,0,1,2,1,1)
2. s_2=(1,0,0,1,0,0,**1,2**,0,1,0,1,2,1,1)

Toutes les MAFs de s_1 et s_2 sont identiques mis à part les MAFs des modules M_7 et M_8. Nous nous concentrons sur les évaluations de ces deux modules.

Dans s_1, nous avons les MAFs avec les périodes suivantes pour M_7 et M_8:

- M_7 = (40, 40, 240, 40, **20**) respectivement dérivées des T_j^{max} = (71, 62, 240, 53, 54)
- M_8 = (**40**, 20, 80)respectivement dérivées des T_j^{max} = (45, 99, 80)

Dans s_2, nous avons les MAFs avec les périodes suivantes pour M_7 et M_8:

- M_7 = (40, 40, 240, 40, **40**) avec les T_j^{max} = (71, 62, 240, 53, 54)
- M_8 = (**20**, 20, 80) avec les T_j^{max} = (45, 99, 80)

Pour le module M_7 les deux solutions ne diffèrent que pour l'unique partition 5, de période P_5= 20 ms pour s_1 et 40 ms pour s_2. De la même façon, pour le module M_8, s_1 et s_2 sont différentes pour la partition P_1 d'une période 40 ms pour s_1 et 20 ms pour s_2.

Si on calcule les évaluations de Q_{avg} et Δ_{avg} en ne considérant que les partitions des modules M_7 et M_8, on obtient pour les deux solutions les même valeurs de Q_{avg}= 0,614 et Δ_{avg}=30,8. Cette égalité est issue d'une symétrie 'inter-modules'. En effet, le couple (20,40) existe dans les deux solutions : pour s_1, on a une périodicité de 20 pour la 5e partition de M_7 et une périodicité de 40 pour la 1ère partition de M_8. Dans s_2 ces deux valeurs sont interchangées. Cette symétrie n'a pas d'effet sur le calcul de charge moyenne ni sur le calcul de marge réseau moyenne. On peut conclure que pour nos métriques, les solutions ayant des métriques identiques comportent soit des MAFs identiques et soit des MAFs avec une certaine propriété de symétrie. Du point de vue de notre application, cette symétrie n'a pas d'impact : s_1 ou s_2 sont intéressantes, toutes les deux. On peut donc choisir un seul candidat par classe pour le présenter à l'intégrateur.

Il serait intéressant d'exploiter et de mieux caractériser cette symétrie dans le futur. Nous n'avons pas poussé l'étude plus loin, mais identifier les solutions symétriques par construction permettrait de réduire encore un peu plus la cardinalité de l'espace de recherche.

6.4.3 Comparaison de différents compromis Pareto optimaux

Nous allons procéder à la comparaison et l'analyse de trois solutions Pareto optimales issues de trois classes d'équivalence différentes : la classe 1, la classe 20 et la classe 56. Nous étudions pour chaque solution la structure des MAFs et les performances qu'elles proposent. Pour cela, nous cherchons à caractériser les solutions en ne connaissant que les périodes des partitions destination.

Dans la Figure 6.5, pour chaque partition destination P_j en abscisse, nous présentons en ordonnée le rapport T_j/T_j^{max}. Nous dénommons ce rapport le *degré de suréchantillonnage de la partition* P_j. Si ce degré vaut 1, la période T_j est égale à la limite maximale T_j^{max}. Par contre, si le rapport est inférieur (strictement) à 1, la période T_j est plus petite que T_j^{max} : il y a sur-échantillonnage. Sur-échantilloner permet d'obtenir un système plus réactif pour les communications : les fenêtre de temps pour la réception sont plus fréquentes. On obtient donc une meilleure marge réseau. Par contre, sur-échantilloner augmente la charge du module hébergeant P_j. La charge la plus faible est obtenue quand le degré de suréchantillonnage vaut 1.

Nous définissons aussi le degré de suréchantillonnage de *l'allocation complète* comme étant la moyenne, pour les nb_{dest} partitions réception, des degrés de suréchantillonnage de chaque

partition :

$$d_{sur} = \frac{1}{nb_{dest}} \sum_{j=1}^{nb_{dest}} \frac{T_j}{T_j^{max}} \tag{6.4}$$

Dans la suite, nous allons utiliser cette mesure directement extraite des périodes des partitions destination pour analyser et comparer des solutions de classes d'équivalence différentes. Pour chaque solution, on peut représenter les degrés de suréchantillonnage des partitions réception. On pourra interpréter la relation entre le degré de sur-échantillonnage et le compromis exprimé par la classe d'équivalence. Autrement dit, on pourra vérifier si le compromis obtenue est en accord avec les degrés de suréchantillonnage des partitions d'une solutions donnée.

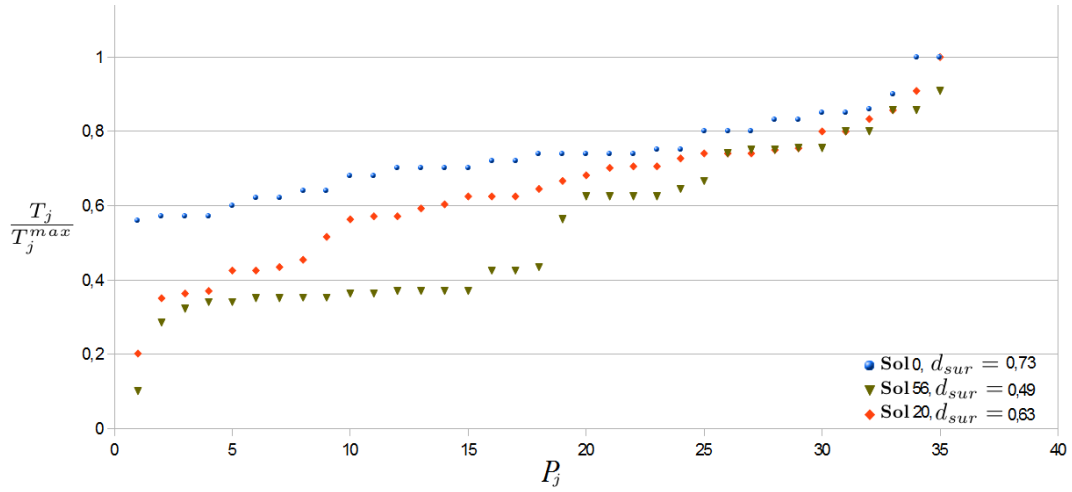


FIGURE 6.5 – Comparaison entre 3 classes d'équivalence Pareto optimale

Solution de la classe 0. La classe d'équivalence 0 est composée d'une unique solution dont les métriques sont (Q_{avg},Δ_{avg})= (0.62, 15.08) . Cette solution est la solution qui minimise la charge des modules. Par contre, cette solution ne maximise pas la marge réseau ; elle présente même un faible valeur de marge réseau. Autrement dit, si l'on cherche à *uniquement* à minimiser la charge des modules, cette solution est la solution optimale. Ainsi, les MAF de cette solution sont construites de façon à minimiser la charge du système complet, c-à-d. les périodes T_j des partitions destinations dans le système sont proches de T_j^{max}. Dans ce cas, il y a très peu de possibilités de suréchantillonnage. Dans la Figure 6.5, nous constatons que la courbe des points bleus, correspondant à la solution Sol_0 est celle qui se rapproche le plus de 1. Cette courbe bleue borne supérieurement toutes les courbes obtenues pour toutes les autres classes d'équivalence. Si on calcule le degré pour l'allocation complète, on obtient : $d_{sur} = 0.73$.

Solution de la classe 56. La classe d'équivalence 56 est composée d'une unique solution dont les métriques sont (Q_{avg},Δ_{avg})= (0.83, 29.65). Contrairement à la solution de la classe 0, la solution de la classe 56 maximise la marge dans le réseau. Cette solution charge au maximum les modules du système car les MAFs sont construites en introduisant un suréchantillonnage maximal des partitions destination. Dans ce cas, on doit observer de faibles valeurs de degré de suréchantillonnage pour les partitions destination. Dans la Figure 6.5, nous constatons bien que la courbe des triangles verts, correspondant à la solution Sol_{56}, est la plus basse. En fait, elle borne inférieurement toutes les courbes obtenues pour les autres classes d'équivalence. Dans ce cas, nous avons $d_{sur} = 0.49$.

Solution de la classe 20. Nous étudions une solution de la classe d'équivalence 20 dont les métriques sont (Q_{avg},Δ_{avg})= (0.70, 22.91). Cette solution représente un compromis équilibré pour la réalisation de l'allocation temporelle d'un système IMA. Il est possible de réaliser des suréchantillonnages en ayant une marge dans la charge des modules ou une marge dans le réseau. Dans la Figure 6.5, nous constatons que la courbe des losanges oranges, correspondant à la solution Sol_{20} est celle dont le degré de suréchantillonnage est compris entre les d_{sur} de Sol_0 et Sol_{56}, c.à.d cette solution est un compromis entre les deux classes 0 et 56. Dans ce cas, $d_{sur} = 0.62$.

En analysant le degré de suréchantillonnage, on observe que les solutions des différentes classes d'équivalence présentes des propriétes en adéquation avec les métriques calculées. Ainsi, une solution de la classe d'équivalence 20 offre un compromis intéressant entre charge et marge réseau. Nous avons pu vérifier ici que c'était bien le cas en observant la structure de la solution (les valeurs des T_j).

6.5 Passage à l'échelle

Les algorithmes exploités au début de ce chapitre sont issus d'une recherche exhaustive. Cette recherche a été améliorée par une réduction de l'espace de recherche via la prise en compte d'une propriété de similarités des MAFs et d'une optimisation locale aux modules. Cette approche nous a permis de calculer le front de Paréto théorique pour un système de 20 modules. Au delà, la mémoire fait défaut sur un ordinateur de bureau standard.

Pour pouvoir passer à l'échelle, nous avons définit une métaheuristique multicritères, dénommée TABOO, au chapitre précédent. Cette section a pour objectif de mesurer l'efficacité de TABOO dans sa recherche du front de Pareto théorique. Pour cela, on comparera les résultats de la recherche TABOO au front de Pareto Théorique.

Comme nous allons le montrer, l'algorithme TABOO permet le calcul d'un front de Pareto *de qualité* en un *temps de calcul raisonnable*. Le front de Pareto obtenu par TABOO est "proche" du front de Pareto théorique (obtenu par recherche exhaustive). Les solutions qui constituent ce front sont très intéressantes, car elles présentent des propriétés proches des solutions théoriques.

Nous procédons dans un premier temps à la comparaison des résultats de l'algorithme TABOO et EXHAUST sur l'Exemple 1, composé de 15 modules. Nous illustrons aussi les performances des deux algorithmes sur l'Exemple 2, composé de 20 modules, qui représente la taille limite que peut supporter nos ressources de calcul.

6.5.1 Exemple 1 : Recherche TABOO

Nous appliquons l'heuristique TABOO sur Exemple 1. Nous testons les fronts pratiques obtenus à différents stades de l'heuristique. La recherche tabou s'arrête après 300 itérations, et fait évoluer $K = 30$ solutions courantes. Nous avons choisis dans un premier temps de réaliser le calcul pour 300 itérations afin d'obtenir plus d'information sur le choix de la condition d'arrêt. Les listes tabou ont une taille fixée à 5 mouvements tabou et une solution est tabou pour 5 itérations consécutives avant d'être libérée. Nous pouvons choisir d'autres valeur pour les paramètres K, la taille de la liste tabou, etc. Nous avons réalisé des tests avec différentes valeurs et retenus ces paramètres car l'heuristique converge plus vite en pratique. Il est à noter que l'optimisation locale est identique à celle de la recherche exhaustive multiobjectif (dominance au sens de Pareto et réduction des MAFs équivalentes). La différence réside dans la méthode d'optimisation globale.

Pour mesurer la convergence de TABOO, nous comparons les fronts de recherche pratique de TABOO obtenus après 10 itérations, 76 itérations et 300 itérations avec le front théorique F_{PT}. Pour cela, nous traçons l'évolution de ces fronts dans l'espace des fonctions de coût. En complément, nous mesurons analytiquement la différence entre les fronts pratiques et le front théorique en calculant la distance générationelle définie précédemment.

Recherche Tabou: 10 itérations

La Figure 6.6.(a) illustre le front de Pareto pratique (croix bleus) obtenu au bout de 10 itérations de l'algorithme parmi les solutions dominées (croix rouges). La Figure 6.6.(b) illustre le Front de Pareto pratique F_{PP} de la recherche tabou après 10 itérations (croix bleus) et le front de Pareto théorique F_{PT}. Il a fallu 23,15s pour tester 6180 solutions parmi les 746 496 solutions candidates. La recherche a obtenu 53 solutions optimales réparties sur 34 classes d'équivalence de solutions (il y a 56 classes sur le front théorique). Autrement dit, on a déjà identifié 34 classes du front théorique. Ceci montre que les solutions obtenues au bout de 10 itérations sont assez proches de F_{PT} et sont réparties dans différentes classes.

La distance générationelle entre F_{PP} et F_{PT} G = 0.03 démontre de bonne performance au bout de 10 itérations seulement. Il est déjà possible à l'intégrateur de choisir une solution parmi les solutions des 34 classes.

Recherche Tabou: 76 itérations

La Figure 6.7.(a) présente le front F_{PP} obtenu au bout de 76 itérations. Les 76 itérations sont réalisées en 2,54 minutes, ce qui correspond au temps nécessaire à la recherche exhaustive

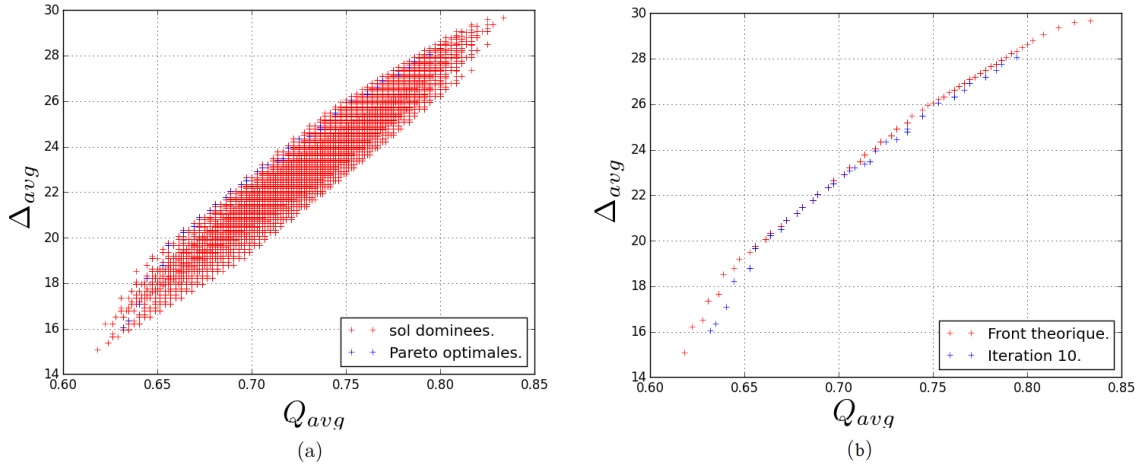


FIGURE 6.6 – Évaluation des solutions pour 10 itérations de la recherche tabou sur Exemple 1

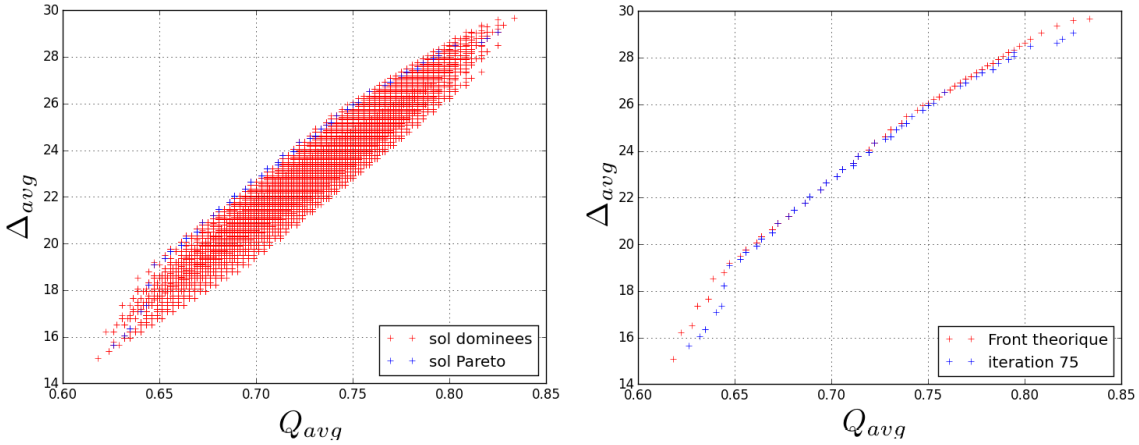
multiobjectif pour calculer le front F_{PT} de Exemple 1. L'intérêt est de connaitre les performances possibles pour la même durée avec l'application de la recherche tabou multiobjectif.

L'heuristique a testé 44370 solutions candidates (sur les 746 496 de l'espace total) pour en extraire 130 solutions du front F_{PP} pour un totale de 51 classes de solutions. Seules 5 classes du front théorique n'ont pas été identifiées. La Figure 6.7.(b) représente le front F_{PP} obtenu après 76 itérations et le front F_{PT}. La distance générationelle G = 0.02, prouve que l'heuristique converge rapidement dans la surface de compromis.

Recherche Tabou: 300 itérations

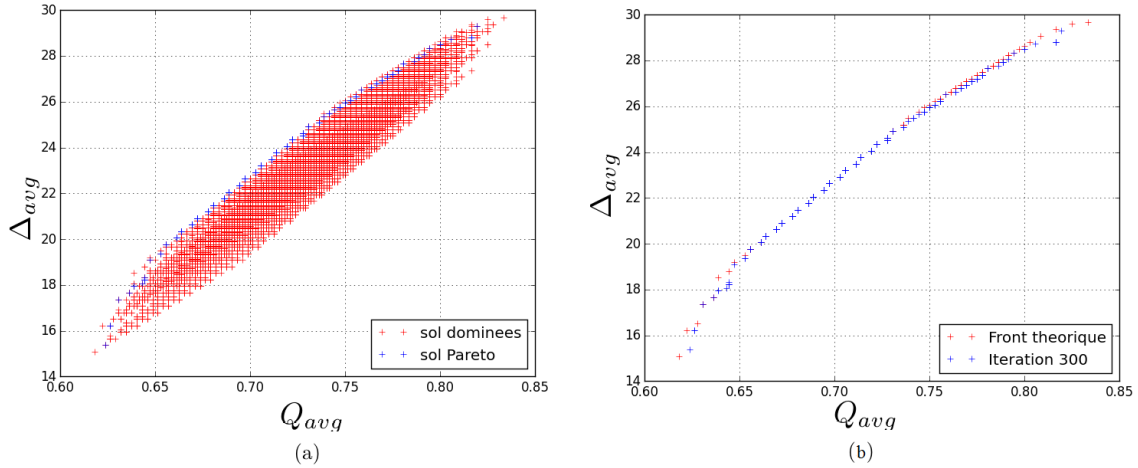
La Figure 6.8.(a) présente le front F_{PP} obtenu au bout de 300 itérations, qui correspondent au nombre d'itération maximum que nous avons fixée. Les 300 itérations sont réalisées en 10 minutes pour tester 171 480 solutions candidates (environs 1/7 de l'espace de recherche). L'heuristique a pu extraire 344 solutions optimales du front F_{PP} pour 54 classes de solutions, ce qui peut couvrir un nombre de classes de solutions optimales proches à celles du front F_{PT}. La Figure 6.8.(b) illustre la distance entre les solutions du front F_{PP} par la recherche tabou multiobjectif en 300 itérations et le front F_{PT}.

La Figure 6.9 illustre le nombre de solutions par classe du front F_{PT} et du front F_{PP} après 300 itérations de l'heuristique Tabou. Les classes sont ordonnées par ordre croissant de la métrique Q_{avg}. Il est à noter qu'à ce niveau de la recherche, les solutions obtenues sont des solutions d'approximation, c.à.d qu'elles proposent des solutions de bonnes qualités à optimales. De ce fait, les classes de la recherche exhaustive et les classes de la recherche Tabou ne sont pas identiques (ne contiennent pas forcément les mêmes solutions). Toutefois, les performances des solutions du front F_{PP} suivent la tendance des solutions du front F_{PT} : on a une distance générationelle G



(a) (b)

FIGURE 6.7 – Évaluation des solutions pour 76 itérations de la recherche tabou sur Exemple 1



(a) (b)

FIGURE 6.8 – Évaluation des solutions pour 300 itérations de la recherche tabou sur Exemple 1

= 0,011. Aussi, le nombre des classes de la recherche Tabou atteint le nombre des solutions du front F_{PT}.

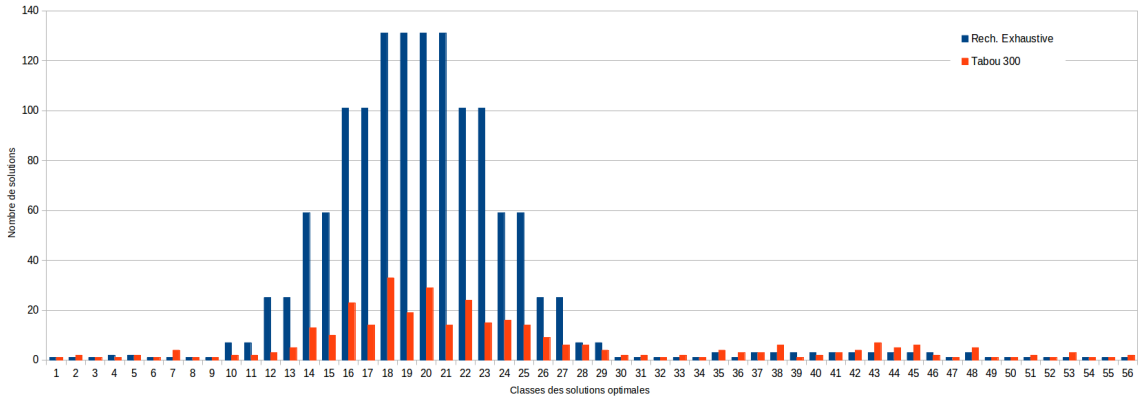


FIGURE 6.9 – Classe des solutions optimales: recherche exhaustive et recherche Tabou 300 itérations - Exemple 1

Le Tableau 6.5 résume les performances obtenues avec l'algorithme TABOO. Nous listons les performances des 3 itérations étudiées, ainsi que la comparaison avec l'algorithme EXHAUST.

	Solutions testées	nb classes	nb Pareto-optimale	G	Temps
10 itérations	6180/746456 = 0.8%	34	53	0.03	23 s
76 itérations	44370/746456 = 5.9%	51	130	0.02	2,54 min
300 itérations	171486/746456 =22.6%	54	344	0.01	10 min
EXHAUST	746456	56	1354	/	2,54 min

TABLEAU 6.5 – Performances de TABOO à différentes itérations - Exemple 1

On peut noter que les temps de recherche entre TABOO et EXHAUST n'évoluent pas de la même façon. L'implantation de TABOO n'a pas été aussi bien optimisée algorithmiquement parlant que l'implantation d'EXHAUST. TABOO est ralenti par la mémorisation du front de recherche qui grossit avec le temps, ce qui rend sa mise à jour avec les nouvelles solutions testées très longues. L'optimisation du code de TABOO est un axe d'amélioration qui permettra de réduire les temps de calcul.

Nous pouvons également connaitre la distance du front pratique obtenu par rapport au front pire cas, c.à.d les solutions qui maximisent la charge des modules et qui minimisent la marge dans le réseau. Ce front est représenté sur la Figure 6.10 par les croix bleues, ainsi que

- front de Pareto théorique (FPT), obtenu par l'algorithme EXAUST, et représenté par les croix vertes;
- front de Pareto pratique à 10 itérations de TABOO (FPP T10), représenté par les croix marrons;

- front de Pareto pratique à 300 itérations de TABOO (FPP T300), représenté par les croix rouges.

Nous pouvons constater, que le front pire cas reste proche des zones de compromis. Ceci est dû à la phase d'optimisation locale, qui permet dans un premier temps de réduire l'espace à des solutions intéressantes.

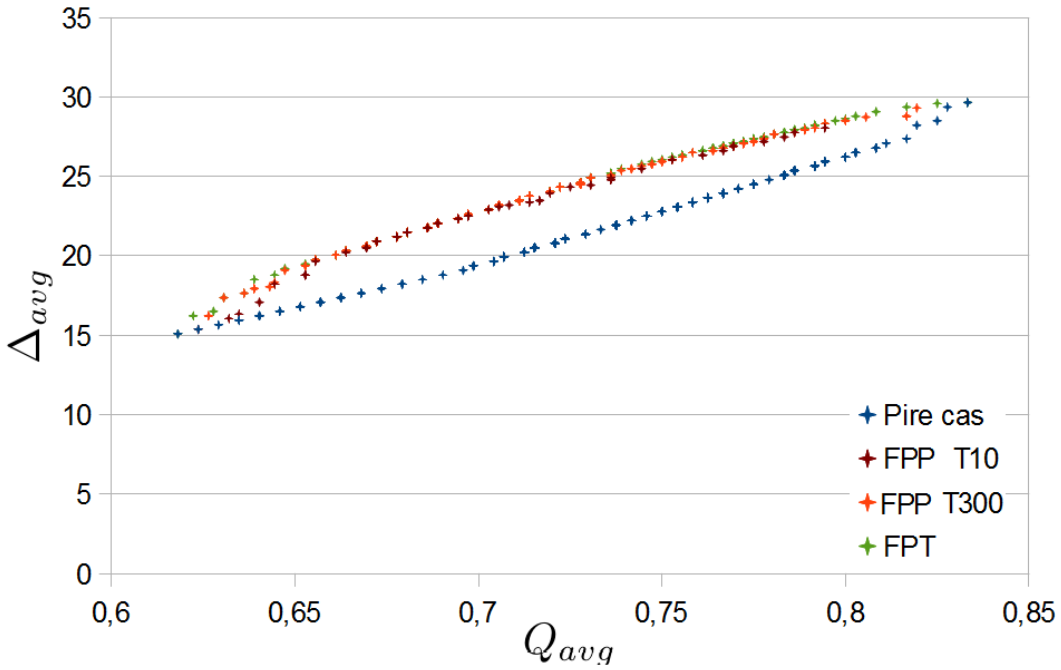


FIGURE 6.10 – Comparaison au front pire cas

6.5.2 Analyse des solutions à 10 itérations de TABOO

Nous procédons à l'analyse de la structure des solutions du front de Pareto obtenu au bout de 10 itérations de l'algorithme TABOO. La distribution des classes d'équivalence à 10 itérations est illustrée dans la Figure 6.11. Les performances sont détaillées dans le Tableau 6.5. Dans la Figure 6.12, nous analysons 3 solutions tirées respectivement des classes d'équivalence 0, 16 et 34. La but de cette étude est de montrer qu'au bout de quelques itérations de l'algorithme TABOO, nous obtenons des solutions intéressantes, même si elles ne sont pas toutes solutions Pareto-optimales.

Solution de la classe 0. La classe d'équivalence 0 est composée d'une unique solution dont les métriques sont (Q_{avg},Δ_{avg})= (0.63, 16.05). Pour ce front pratique, c'est la solution qui minimise la charge des modules au maximum. Le front théorique comporte une solution qui domine cette solution (avec (Q_{avg},Δ_{avg})= (0.62, 15.08)). Le degré moyen de suréchantillonnage est d_{sur} =

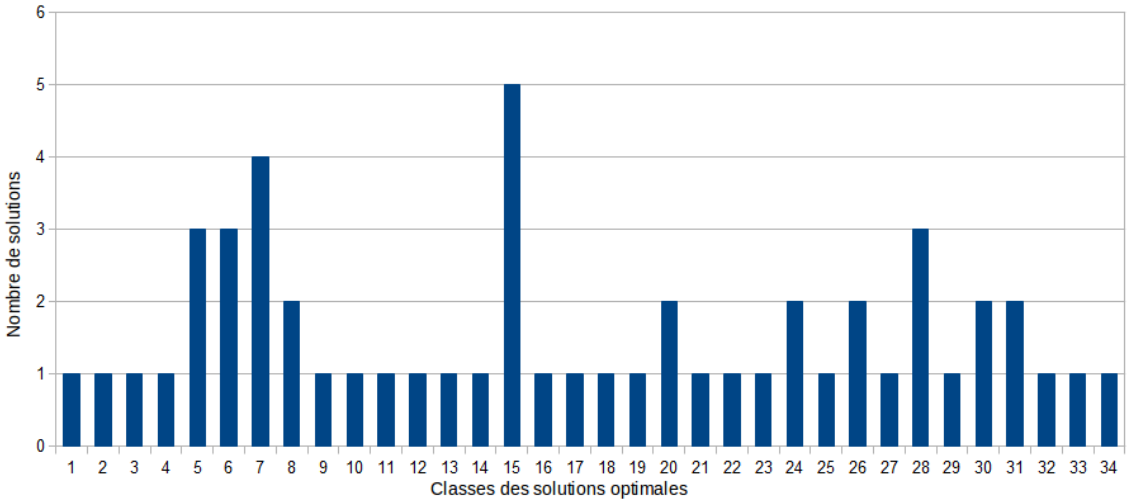


FIGURE 6.11 – Classes d'équivalence à 10 itérations de l'algorithme TABOO

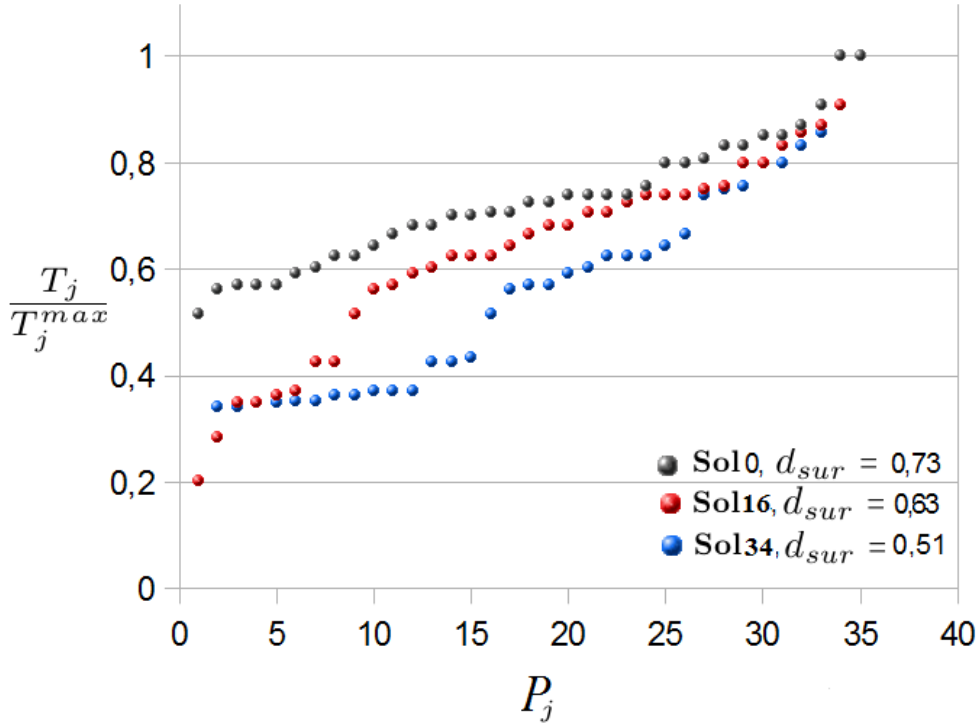


FIGURE 6.12 – Comparaison entre 3 classes d'équivalence Pareto optimale - 10 itérations de l'algorithme TABOO (Exemple 1)

0.73 pour la solution du front pratique. La solution du front théorique a le même degré de suréchantillonnage moyen. La solution du front pratique présente une structure très proche de celle du front théorique.

Solution de la classe 16. La solution de la classe d'équivalence 16 dont les métriques sont (Q_{avg},Δ_{avg})= (22.91,0.70) propose les mêmes performances que la classe 20 obtenue par l'algorithme EXHAUST. L'algorithme TABOO a pu au bout de 10 itérations ressortir une solution Pareto optimale, avec un degré de suréchantillonnage moyen $d_{sur} = 0.63$. Cette solution représente un compromis équilibré pour l'intégration d'un système IMA. Le degré de suréchantillonnage moyen pour le front optimal est de $d_{sur} = 0.63$ aussi. On remarque aussi que la solution du front pratique présente une structure très proche de celle de la solution du front théorique.

Solution de la classe 34. La classe d'équivalence 34 est composée d'une unique solution dont les métriques sont (Q_{avg},Δ_{avg})= (0.79,28.05). Cette solution est intéressante si nous nous soucions seulement de l'optimisation de la marge réseau. Pour cette solutions, $d_{sur} = 0.51$. La solution Pareto-optimale qui maximise la marge réseau a une évaluation (Q_{avg},Δ_{avg})= (0.83, 29.65). Cette solution domine la solution de la classe 34 obtenue par TABOO. Néanmoins, ces deux solutions sont proches structurellement. En effet, $d_{sur} = 0.51$ pour la solution Pareto-optimale.

Le Tableau 6.6 résume la comparaison entre les degrés moyens de suréchantillonnage entre l'algorithme TABOO et EXHAUST.

	d_{sur}
TABOO:Classe 0	0.73
TABOO:Classe 16	0.63
TABOO:Classe 34	0.51
EXHAUST:Classe 0	0.73
EXHAUST:Classe 20	0.63
EXHAUST:Classe 56	0.49

TABLEAU 6.6 – Degrés de suréchantillonnage moyens pour TABOO (10 itérations) et EXHAUST - Exemple 1

La figure 6.13 récapitule les suréchantillonages de différents classes pour les algorithmes EXHAUST et TABOO. Nous pouvons constater que les deux algorithmes convergent vers des solutions avec des degrés de suréchantillonage similaires, que ce soit en exploitant toutes les solutions possibles (algorithme EXHAUST) ou en réalisant un nombre limité d'itérations et obtenir un nombre de classes qui couvrent beaucoup de solutions pareto optimales.

FIGURE 6.13 – Comparaison du suréchantillonage dans EXHAUST et TABOO pour Exemple 1

Nous pouvons conclure qu'au bout de quelques itérations, et en ayant testé un petit ensemble de solutions candidates, TABOO trouve des solutions aux performances très proches des performances des solutions Pareto-optimales. De plus, les solutions obtenues avec 10 itérations de TABOO sont structurellement proches des solutions Pareto-optimales.

6.5.3 Critère d'arrêt de TABOO pour exemple 1

Nous avons fait le choix d'arrêter les calculs de l'algorithme au bout de 300 itérations. Pourtant, une heuristique tabou peut être stoppée de différentes façons : si celle-ci a testé tout l'espace des solutions candidates (pas facile à mettre en oeuvre), si le nombre de solutions Pareto optimales à chaque itération se stabilise et atteint un point d'inflexion, ou si l'algorithme a trouvé un nombre de solutions satisfaisantes et qu'il n'est pas nécessaire d'aller plus loin.

En observant sur la Figure 6.14 l'évolution du nombre de solutions du front de Pareto par l'algorithme au cours du temps, on observe que celui-ci évolue et augmente globalement tout le temps. Or, il serait intéressant de raisonner sur l'évolution du nombre de classes d'équivalence trouvées au cours du temps dans le front. En effet, même si on rajoute de nouvelles solutions dans le front de Pareto pratique, il est plus rare de trouver une solution qui représente un nouveau compromis. Le rajout d'un tel critère d'arrêt n'est cependant pas anodin, car le calcul du nombre de classes d'équivalence rajoute en complexité à chaque itération et plus le front est grand, plus cette recherche sera longue.

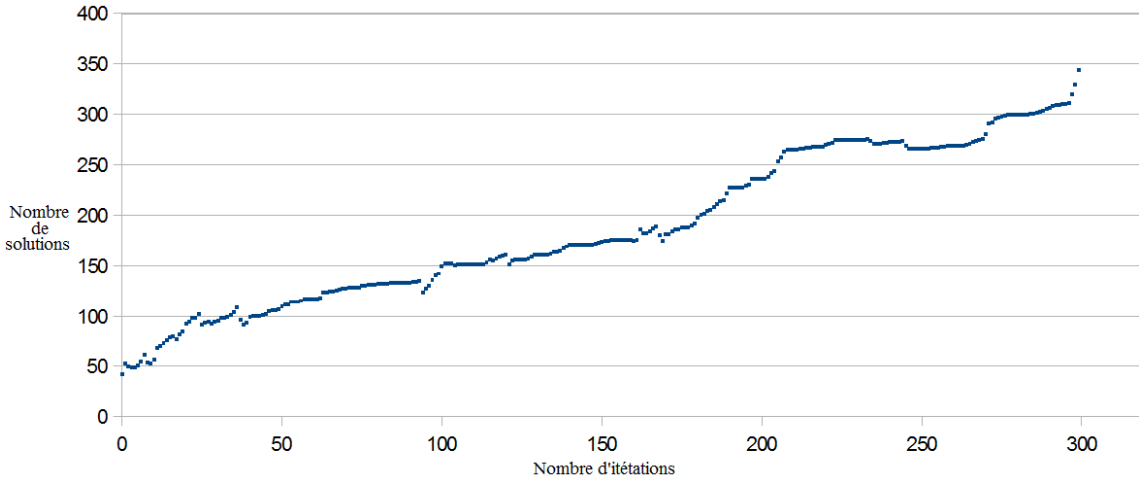


FIGURE 6.14 – Évolution de la taille du Front pratique sur 300 itérations de TABOO - Exemple 1

Trouver un front intéressant pour notre étude se traduit par la recherche d'une bonne solution par classe d'équivalence (i.e. par compromis). Ainsi, dimensionner un critère d'arrêt qui tienne compte du nombre de classes semble une approche intéressante. On pourrait définir un critère

d'arrêt qui stoppe la recherche si au moins N classes d'équivalences ont été trouvées. On pourrait définir un critère qui stoppe la recherche si l'augmentation du nombre de classes est très faible depuis T itérations. Nous n'avons pu faire cette analyse, mais elle serait intéressante à mener dans des travaux futurs.

Bilan sur l'Exemple 1

Voici les points importants que nous voulons souligner à ce stade du manuscrit.

Optimisation locale multiobjectif. L'optimisation à deux niveaux (locale et globale) a permis de considérablement réduire l'ensemble des solutions candidates. En effet, l'optimisation locale au niveau de chaque module par calcul de dominance fait passer l'ensemble des solutions candidates de 32 348 160 à 2 359 296 solutions, soit une réduction de l'espace des solutions candidates de 92.7%.

Réduction des MAFs équivalentes. De plus, après le calcul de la dominance au sens de Pareto sur les MAFs de chaque module, l'application locale de la réduction des MAFs similaires fait davantage baisser le nombre des solutions candidates. En effet l'ensemble des solutions candidates à l'allocation temporelle passe de 2 359 296 solutions à seulement 746 496 solutions, soit une réduction de 68.3%. Ces optimisations locales sont un avantage majeur pour les performances des heuristiques proposées. L'optimisation en un seul niveau (niveau global) aurait été plus demandeuse en ressources de mémoire, de calcul et de temps.

La recherche exhaustive multiobjectif globale (EXHAUST). Cette heuristique est capable de fournir des solutions optimales exactes pour des systèmes IMA de taille petite à moyenne en un temps raisonnable comme dans Exemple 1. L'algorithme a pu extraire les 1354 solutions Pareto optimales réparties sur 56 classes d'équivalence. le calcul a nécessité 2.54 minutes. Si l'intégrateur désire absolument avoir toutes les solutions exactes de compromis, et qu'il dispose également des ressources de calcul et de mémoire nécessaires, la recherche exhaustive multiobjectif est un bon outil pour l'aide à la décision pour l'intégration temporelle.

La recherche tabou multiobjectif (TABOO). Cette heuristique est approchée, c.à.d qu'elle ne garantit pas de trouver toutes les solutions du front Pareto-optimal théorique, mais offre un ensemble de solutions de qualité, très proches de celles du front optimal. En effet, l'étude des performances de la recherche tabou à différents stades de son exécution nous a permis de constater qu'après seulement 10 itérations réalisées en $23s$, le front F_{PP} est très proche du front théorique F_{PT} avec une distance générationelle G = 0.03. Il est possible pour l'intégrateur à ce stade de la recherche de choisir une stratégie d'allocation temporelle pour son problème d'intégration. Après 76 itérations, l'algorithme n'a pas été capable de retrouver toutes les solutions Pareto optimales, mais continue à améliorer la distance générationelle. Au bout de 300 itérations, l'algorithme

propose 344 solutions optimales avec une distance générationelle G =0.018 contre 1345 pour la recherche exhaustive. L'exécution des 300 itérations est réalisée en 10 min, ce qui est presque 5 fois plus long que pour la méthode optimale par recherche exhaustive. Néanmoins, les performances intermédiaires sont très intéressantes si l'intégrateur fait le choix d'arrêter la recherche tabou. Et ce temps pourra être réduit par une meilleure implantation algorithmique de TABOO.

Ces résultats illustrent les performances des heuristiques proposées pour un système de 15 modules de calcul. La recherche exhaustive est plus performante à ce niveau grâce aux optimisations intermédiaires réalisées au niveau des modules. La recherche tabou propose néanmoins plus rapidement un ensemble réduit de solutions intéressantes qui font partie du front de pareto optimal. Nous étudions dans ce qui suit les résultats de l'heuristique TABOO en augmentant la taille de l'exemple, avec Exemple 2 dont la configuration est détaillée dans 6.1.2.

6.5.4 Exemple 2 : Recherche TABOO

Nous considérons Exemple 2, détaillé dans le tableau 6.1.2, pour le problème d'optimisation P dont voici l'instance:

- Système testé: Exemple 2
- Nombre de critères pour le niveau d'optimisation des allocations temporelles: 2
- Critères d'optimisation des allocations temporelles : charge moyenne des modules du système (Q_{avg}), et marge moyenne des partitions destinations du système Δ_{avg}.
- Nombre de critères pour le niveau d'optimisation local: 2
- Rang de de Pareto: 1
- Critères pour le niveau d'optimisation local: charge moyenne d'un module (Q_l), et moyenne des marges des partitions destinations (σ_{M_l})

Recherche exhaustive multiobjectif

Nous avons prouvé précédemment l'intérêt de la réduction des MAFs dans l'optimisation locale afin de réduire le nombre de solutions candidates à l'allocation temporelle sans perdre de solutions intéressantes.

Nous détaillons pour l'exemple 2 les résultats de l'application de TABOO. Nous avons au préalable exécuté l'algorithme EXHAUST. Le front de Pareto théorique d'EXHAUST appliqué à l'exemple 2 est illustré sur la Figure 6.15.

La Figure 6.15 représente 3 981 312 solutions candidates ainsi que le front F_{PT} (croix bleues) contenant 2256 solutions Pareto optimales. Ces solutions sont réparties sur 48 classes de solutions optimales. Le calcul du front de Pareto théorique F_{PT} a été réalisé en 1.86 heures où toutes les solutions ont été testées. Nous utilisons le front F_{PT} obtenu dans cette recherche afin d'évaluer les performances de l'heuristique tabou proposée. Nous allons analyser par la suite comment TABOO réagit sur un système de plus grande taille en analysant la convergence de F_{PP} vers F_{PT}.

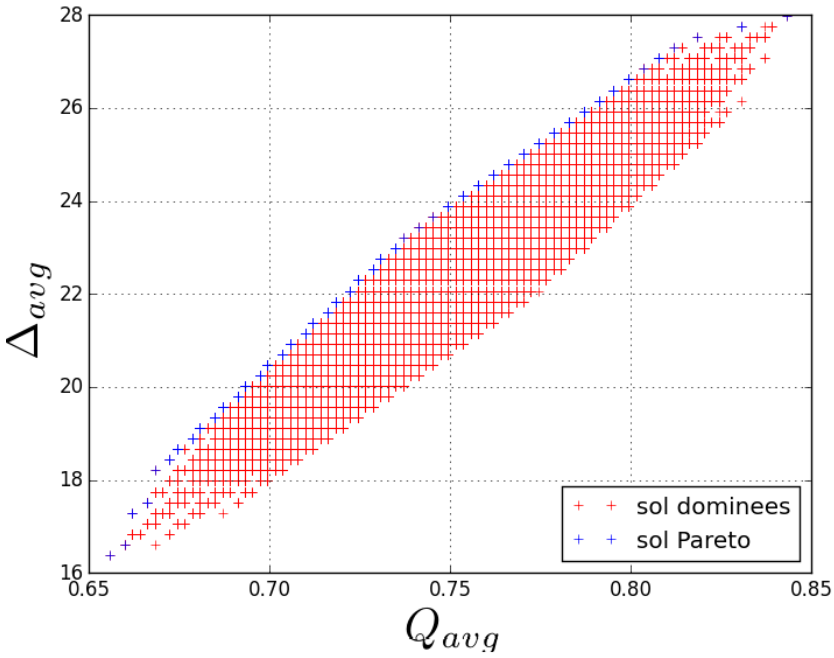


FIGURE 6.15 – Évaluation des solutions et représentations des solutions Pareto optimales; EXHAUST sur Exemple 2

Paramétrage de TABOO

Nous appliquons l'algorithme TABOO sur Exemple 2. La configuration de la recherche tabou est identique à celle appliquée à Exemple 1. L'heuristique effectue 300 itérations au maximum et fait évoluer K= 30 solutions courantes. Les listes tabou ont une taille fixée à 5 mouvements tabou et une solution est tabou pour 5 itérations consécutive avant d'être libérée. Nous testons la recherche tabou pour 10 itérations, 100 itérations et 300 itérations.

Recherche tabou: 10 itérations

La Figure 6.16 illustre le front F_{PP} obtenu au bout de 10 itérations de la recherche tabou multiobjectif. Il a fallut 2.55 minutes pour extraire 79 solutions réparties sur 29 classes de solutions. Compte tenu du nombre important de solutions à tester; et du fait que la dominance au sens de Pareto teste à chaque ajout d'une nouvelle solution optimale si celle-ci ne domine pas les solutions de l'ensemble optimal intermédiaire; il est plus difficile de trouver des solutions Pareto-optimales de F_{PT}.

En effet, même si le front F_{PP} à l'itération 10 est proche de F_{PT}, très peu de solutions sont communes aux deux fronts. Toutefois, la distance générationelle G = 0.034, ce qui est faible.

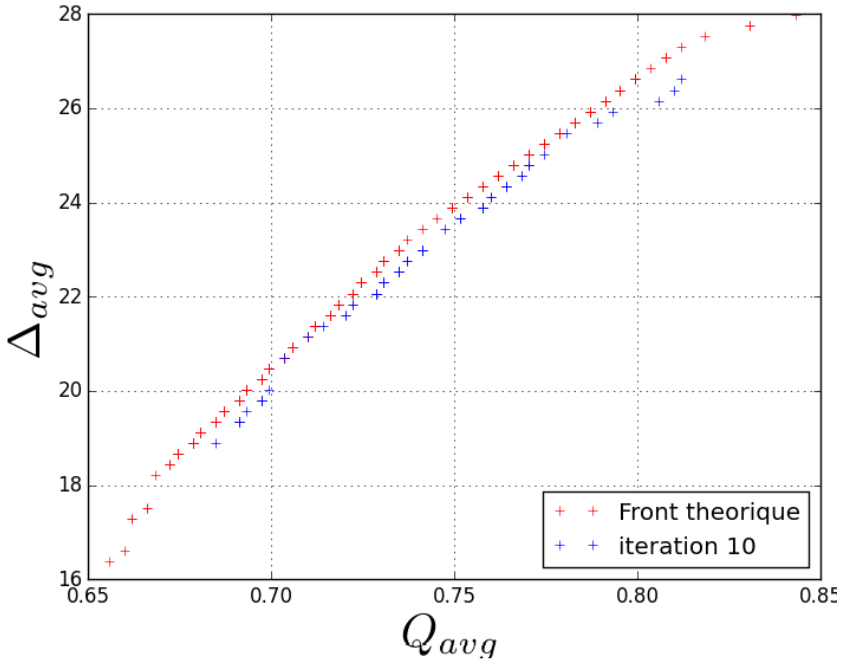


FIGURE 6.16 – Évaluation des solutions pour 10 itérations de la recherche tabou sur Exemple 2

Recherche tabou: 100 itérations

La Figure 6.17 montre le front F_{PP} obtenu au bout de 100 itérations de la recherche tabou multiobjectif qui a nécessité 21 minutes de calcul afin d'extraire 95 solutions Pareto optimales du front F_{PP} et 41 classes de solutions optimales. Les solutions du front F_{PP} sont sensiblement dans la zone de compromis du front F_{PT} avec une distance générationelle G = 0,024. Pour un temps de recherche bien plus faible que la recherche exhaustive, on a réussi a bien réduire la distance générationelle avec TABOO.

Recherche tabou: 300 itérations

La Figure 6.18.(a) représente les solutions du front F_{PP} et les solutions dominées (croix rouges). Il a fallu 1.02 heure pour extraire 140 solutions optimales et 43 classes de solutions. L'heuristique a testé 171 480 solutions. La Figure 6.18.(b) illustre le front F_{PP} et le front théorique F_{PT}. Dans cette figure, nous pouvons constater que le front F_{PP} est quasiment intégré au front F_{PT} sur la plus part des régions

La Figure 6.19 illustre les classes des solutions du front F_{PT} de Exemple 2 et du front F_{PP} au bout de 300 itérations de la recherche Tabou. Toutes les classes n'ont pas été extraites. Néanmoins, la distance générationelle G = 0.023 est faible. À ce stade l'intégrateur est capable de faire un choix d'une stratégie d'intégration temporelle. En effet, il se peut qu'une configuration

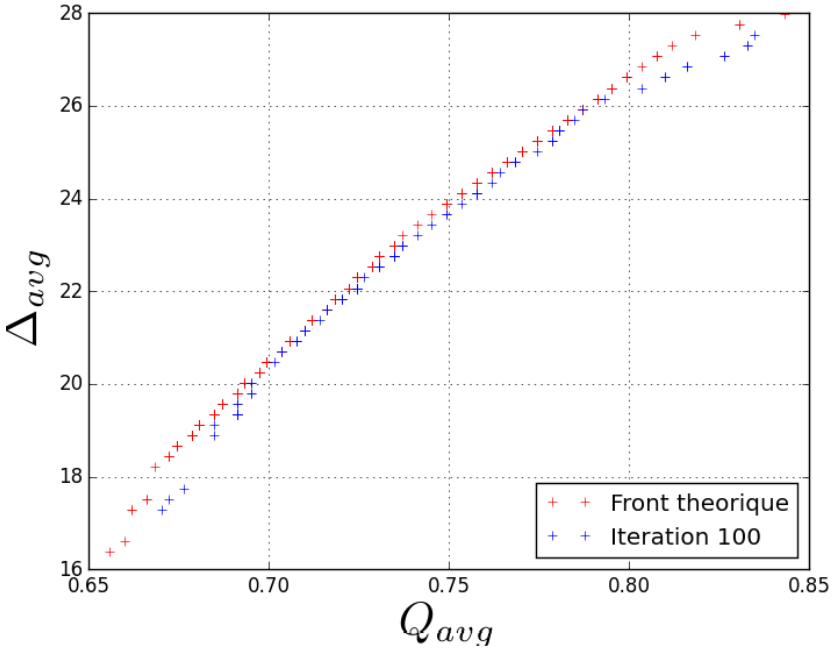
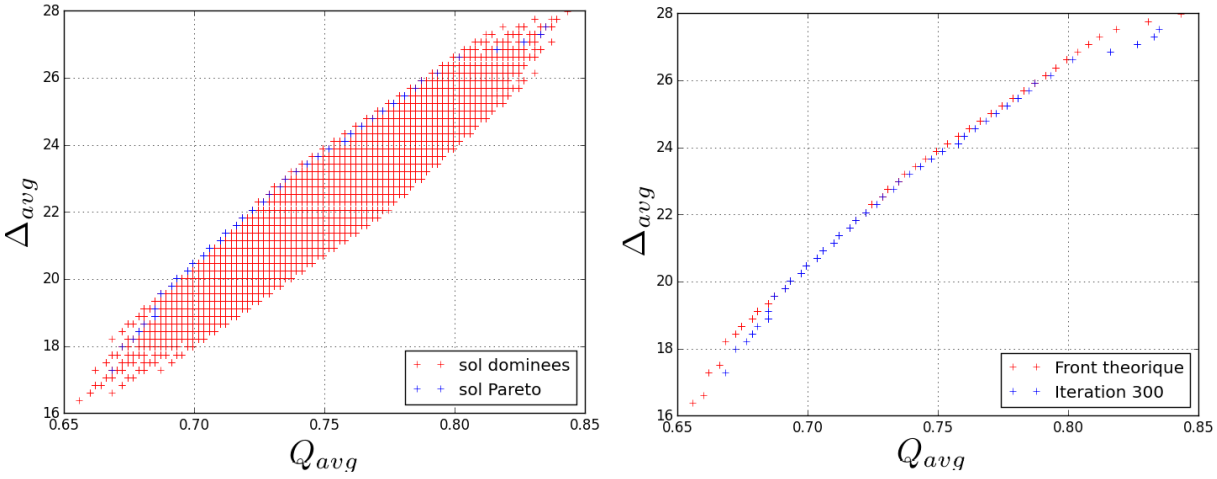


FIGURE 6.17 – Évaluation des solutions pour 100 itérations de la recherche tabou sur Exemple 2



(a) (b)

FIGURE 6.18 – Évaluation des solutions pour 300 itérations de la recherche tabou sur Exemple 2

même non optimale soit satisfaisante au vu des performances voulues pour le système. En effet, les solutions à 300 itérations sont prometteuses car elles sont très proches du front de Paréto théorique.

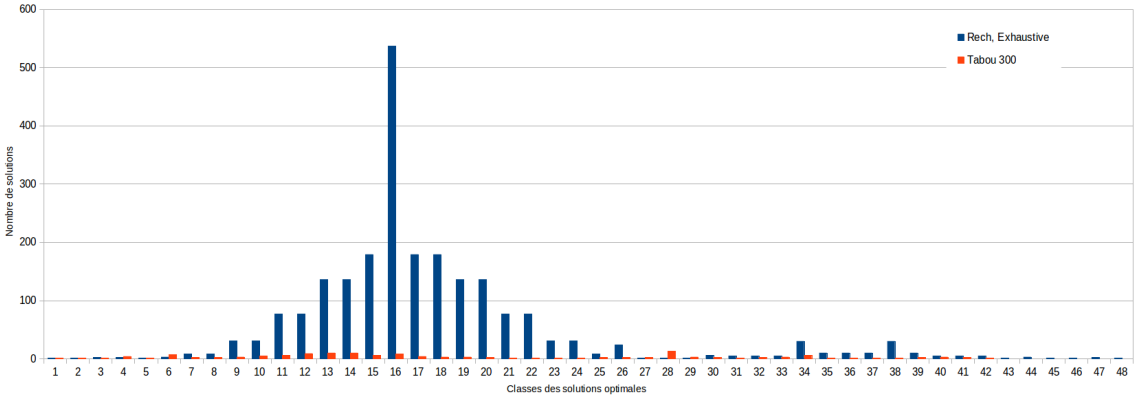


FIGURE 6.19 – Classe des solutions optimales : recherche exhaustive et recherche Tabou 300 itérations - Exemple 2

Discussion

Ces tests nous permettent de conclure, pour un problème de petite taille, que l'heuristique exhaustive multiobjectif converge et extrait la totalité des solutions de la surface de compromis optimale. La méthode tabou est plus adaptée aux problèmes de plus grande envergure. En effet, l'heuristique tabou multiobjectif réussit à converger vers la surface de compromis optimale en un nombre réduit d'itérations et exige moins de temps de calcul que l'heuristique exhaustive multiobjectif. Étant une approche d'approximation capable d'extraire des solutions de bonnes qualités à optimales, il est possible d'arrêter la recherche tabou au bout d'un certain nombre d'itérations.

Comme nous pouvons le constater sur la Figure 6.16, en peu d'itérations et en un temps de calcul très raisonnable comparé à la taille du problème, l'heuristique tabou multiobjectif à été capable d'éliminer la grande majorité des solutions non intéressantes pour le problème d'optimisation avec une distance générationelle à G=0.034 après 10 itérations, et de G=0.02 après 300 itérations. Toutefois, la recherche tabou connaît durant les dernières itérations peu d'améliorations compte tenu du nombre de solutions améliorées.

Une des difficultés de ce type d'heuristique est le choix de la solution de départ de la recherche. Il serait intéressant d'introduire des mécanismes permettant de choisir les solutions permettant de converger le plus rapidement vers la surface de compromis. Nous allons proposer quelques solutions dans les perspectives de cette thèse. La limite des performances de la machine sur laquelle sont réalisées les simulations n'a pas permis de tester des configurations capable d'atteindre les

limites de la recherche exhaustive compte tenu de la taille de l'espace des solutions, et sur lequel TABOO aurait pu extraire quelques solutions en un temps réduit.

En terme de choix des solutions intéressantes, l'intégrateur peut privilégier la solution de la classe 1 (Q_{avg}=0.61 ,Δ_{avg} = 18.43) si celui-ci tend vers un système dont l'allocation temporelle doit être robuste à l'évolution des calculateurs, une solution de la classe 41 avec (Q_{avg}=0.83 ,Δ_{avg} =27.5) si celui-ci cherche une solution robuste aux changements que peut connaître le réseau ou la solution de la classe 19 avec (Q_{avg}= 0.7 ,Δ_{avg} = 20.1) si le but est d'avoir un système plutôt équilibré sur les deux critères.

	Marge d'évolution CPMs	Marge dans le réseau	Nombre de solutions
Classe 1	0.39	18.43	1
Classe 19	0.3	20.1	13
Classe 41	0.17	27.5	1

TABLEAU 6.7 – Quelques solutions optimales par recherche Tabou multiobjectif - Exemple 2

Le fait d'avoir différentes classes de solutions optimales facilite la re-configuration du système. En effet, l'intégrateur, grâce à l'heuristique proposée, connaît des solutions alternatives pour re-configurer le système si besoin.

6.6 Conclusion

Nous avons réalisé dans ce chapitre l'application des algorithmes d'optimisation multiobjectif proposés, ainsi qu'une analyse de leurs performances. Cette analyse met en évidence l'intérêt des techniques d'optimisation multiobjectif pour l'aide à la décision d'une allocation temporelle. Ces approches fournissent à l'intégrateur un degré de liberté dans le choix de la solution qui correspond le mieux à ses attentes.

L'heuristique tabou multiobjectif mérite d'être améliorée pour accroitre ses performances. Nous envisageons de la tester sur une plateforme dont les ressources de calcul et mémoire seraient plus puissantes. De plus, il serait intéressant d'introduire un mécanisme capable de mieux choisir les K solutions qui évoluent dans l'heuristique, tel que le démarrage de l'heuristique par les solutions pires cas que nous pouvons facilement obtenir (allocation temporelle avec les MAFs les moins chargées, allocation temporelle les marges les plus larges). En effet, le choix des solutions de départ est déterminant dans la convergence d'une heuristique par exploration du voisinage. Il serait aussi envisageable d'optimiser la gestion de la liste tabou en faisant varier sa taille entre les différentes itérations.

7 Conclusion et perspectives

7.1 Conclusion

Cette thèse a pour motivation principale la maîtrise de l'allocation temporelle dans les systèmes temps réel modulaires par l'analyse des contraintes temporelles de bout-en-bout. Deux problèmes clés ont été abordés: la maîtrise des contraintes de bout-en-bout pour l'allocation temporelle, et la question du choix d'une stratégie d'allocation temporelle des ressources. Ces problèmes représentent de réels défis pour les spécialistes de l'allocation dans les étapes de l'intégration et la configuration d'un système temps réel. Ceci découle la nécessite d'une aide au paramétrage des fonctions et communications embarquées de façon à garantir le bon fonctionnement du système temps réel. Ce paramétrage devient une tâche de plus en plus complexe à achever compte tenu des exigences et contraintes imposées, nécessaire. De plus, la croissance de la taille des systèmes embarqués industriels et l'introduction de l'architecture modulaire augmentent la complexité du processus d'intégration. En effet, les architectures modulaires prônent la réutilisation et l'évolutivité, nécessitant la revérification de l'ensemble de la configuration pour l'introduction d'un nouveau composant matériel ou logiciel. Ces travaux sont appliqués au cas des architectures IMA.

Le travail présenté dans les chapitres précédents se préoccupe de la mise en place d'une démarche d'aide à l'allocation temporelle des ressources et le choix d'une stratégie d'allocation temporelle. Cette démarche conduit au choix d'une allocation temporelle optimale pour un système temps réel dans les phases amont de conception.

Dans un premier temps, une analyse en détail est réalisée des délais de bout-en-bout entre fonctions embarquées distantes et l'impact des paramètres temporels sur les performances de communication. Les communications sont soumises à des contraintes temporelles: la fraîcheur des données transmises et les délais de transmission pire et meilleure cas précalculés.

De cette analyse, nous avons proposé deux propriétés qui, si elles sont respectées, permettent de garantir les contraintes sur les délais de bout-en-bout. Ceci passe par la non-violation des paramètres de fraîcheur et la non perte des mises à jour des flux de messages. Ces deux propriétés sont utilisées pour le calcul des périodes destinations maximales. En effet, elles dérivent la période d'exécution maximale garantissant les contraintes temporelles sur les délais de bout-en-bout. Après la dérivation des périodes "pire cas" capables d'assurer les contraintes de bout-en-bout,

l'étape suivante consistait à proposer les MAFs valides et ordonnaçables. Nous avons proposé une heuristique capable de calculer toutes les combinaisons des périodes harmoniques entre partitions d'un même module. Afin de proposer des ordonnancements valides, nous nous sommes inspirés de l'heuristique Least Loaded afin de construire des séquencements faisables sur une hyperpériode. Cet algorithme n'est pas optimal mais assure de proposer pour l'allocation temporelle des MAFs ordonnaçables.

Cette thèse aborde ensuite le problème du choix d'une stratégie d'allocation temporelle comme un problème d'optimisation multiobjectif. Le but est d'extraire l'allocation temporelle la plus optimale du système, c.à.d qui propose les meilleurs compromis entre l'évolutivité des modules de calcul et du réseau aux futures mis à jour. Nous avons proposé des métriques capables de quantifier la qualité d'une allocation temporelle en matière de charge des modules et la marge d'évolution du réseau. La stratégie optimale maximise la marge dans le réseau et minimise la charge totale des calculateurs du système. Une solution est donc une configuration temporelle possible du système construite par une combinaison possible des MAFs des modules du système.

Pour solutionner notre problème d'allocation, nous proposons des heuristiques d'optimisation adaptées. Pour pouvoir prendre en compte le caractère multiobjectif du problème d'allocation temporelle, deux heuristiques multiobjectif ont été développées. Une première heuristique basée sur la recherche exhaustive multiobjectif a été implantée. Cette heuristique procède sur deux niveaux d'optimisation: niveau local des CPM et niveau global du système. Les solutions non dominées au sens de Pareto sont stockées dans un front de Pareto théorique du problème. Chaque solution du front représente un compromis différent entre les critères du problème à résoudre. Cette méthode assure de trouver les solutions optimales exactes, mais rencontre des difficultés pour les systèmes de tailles industrielles. En effet, la méthode compare toutes les solutions et est demandeuse en ressources de calcul et de mémoire.

La seconde méthode présentée est une heuristique tabou multicritère. Plusieurs recherches tabou sont effectuées simultanément pour un nombre d'itérations donné. À chaque itération, les solutions non dominées au sens de Pareto sont mémorisées dans un front de Pareto optimal. L'algorithme s'arrête au bout d'un certain nombre d'itérations. Nous avons testé la convergence de l'heuristique tabou vers le front Pareto théorique. Cette analyse montre que l'heuristique converge très rapidement vers la surface de compromis au bout des premières itérations. L'intégrateur peut ainsi exécuter l'heuristique pendant quelques itérations puis choisir une solution qui satisfait au mieux ses besoins.

Pour faciliter le choix d'une solution dû le problème parmi les solutions Pareto optimales, nous avons regroupé les solutions équivalentes en un ensemble de classes de solutions: chaque classe contient différentes stratégies qui ont des valeurs identiques des métriques d'évaluation.

7.2 Perspectives

Dans la même philosophie que l'IMA, le secteur automobile a répondu à la problématique de la croissance dans les systèmes embarqués par la création d'un consortium regroupant industriels, constructeurs de véhicules, fournisseurs technologiques et équipementiers nommés AUTomotive Open System ARchitecture (AUTOSAR) [8] en 2003. AUTOSAR a aboutit à la définition d'une infrastructure commune indépendante de la plateforme matérielle. AUTOSAR comporte un middleware et une couche applicative qui échangent des données via une interface standardisée. Elle regroupe les composants fonctionnels, les Software Component (SWC). Un SWC est composé d'un ensemble de *runnables*. Les runnables sont les plus petites entités logicielles qui s'occupent de l'exécution des fonctions voulues. Cette architecture est très complexe. Néanmoins, des problématiques similaires à celles abordées en IMA dans cette thèse y existent. Il serait très intéressant de se pencher sur cette technologie pour voir si elle peut bénéficier de nos recherches.

Dans l'objectif de couvrir les délais de bout-en-bout dans leurs intégralités, il serait important d'introduire la gestion des transactions ou chaînes de communication, où le délai de bout-en-bout s'étend sur une suite de partitions communicantes. Lauer [31] et Monot [15] ont réalisé une analyse des délais de bout-en-bout respectivement sur des transactions IMA et AUTOSAR. Le but est d'être le plus proche du système réel. Il est également intéressant de tester les heuristiques proposées sur un modèle de système AUTOSAR.

L'heuristique de placement LL utilisée pour l'ordonnancement n'est pas optimale et peut écarter des solutions faisables. Ce travail pourra être poursuivi en utilisant un algorithme plus adapté à la validation des ordonnancements. Comme il a été listé dans l'état de l'art, le problème d'ordonnancement pour ce type de système est un axe de recherche actif.

Afin de réaliser une aide à la décision plus complète, il serait intéressant de proposer une démarche d'optimisation mixte. Cette démarche serait capable d'optimiser les allocations spatiales et les allocations temporelles. Il est nécessaire de considérer d'autres contraintes temporelles telles que la cohérence des données. Ceci passe par la prise en compte des communicaation en mode queueing. Aussi, l'introduction de métriques plus adaptées pour la quantification des solutions est envisageable.

Il est possible d'améliorer le fonctionnement de l'heuristique tabou proposée. Ceci est envisageable en introduisant des mécanismes capables de guider le choix des solutions courantes à évoluer afin de diversifier au maximum les voisinages exploités et converger plus rapidement vers la surface de compromis. Il est tout a fait possible de calculer la solution qui propose la meilleure charge possible dans le système, en choisissant les MAFs les moins chargées. De même, il est possible de calculer la solution qui propose la plus grande marge dans le débit réseau. Ces deux solutions peuvent être utilisées dans l'ensemble des solutions de départs de l'heuristique TABOO, afin de mieux converger vers l'espace de compromis.

Liste des communications

Conférences internationales avec comité de lecture

- [1] Nesrine Badache, Katia Jaffrès-Runser, Jean-Luc Scharbarg and Christian Fraboul. End-to-end delay analysis in an Integrated Modular Avionics architecture (short paper). In *Emerging Technologies and Factory Automation (ETFA)*, Cagliari, Italy, 10/09/2013-13/09/2013.

- [2] Nesrine Badache, Katia Jaffrès-Runser, Jean-Luc Scharbarg and Christian Fraboul. Managing temporal allocation in Integrated Modular Avionics (regular paper). In *Emerging Technologies and Factory Automation (ETFA)*, Barcelone, 16/09/2014-19/09/2014.

Bibliographie

[1] Chung Laung Liu and James W Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 1973.

[2] Nicolas Navet, Aurélien Monot, Bernard Bavoux, and Françoise Simonot-Lion. Multi-source and multicore automotive ECUs - OS protection mechanisms and scheduling. In *International Symposium on Industrial Electronics - ISIE 2010*, Juillet 2010.

[3] J. H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, 1975.

[4] *Design Guidance for Integrated Modular Avionics. ARINC specification 651*, 1991.

[5] Christian Fraboul and Frank Martin. Modeling and simulation of integrated modular avionics. In *Parallel and Distributed Processing'98*, pages 102–110, 1998.

[6] Franck Martin. *Modélisation et évaluation de performances prévisionnelles d'architectures Avioniques Modulaires Integrées.* Thèse de doctorat, septembre 1999.

[7] Bernhard Leiner, Martin Schlager, Roman Obermaisser, and Bernhard Huber. A comparison of partitioning operating systems for integrated systems. In *Computer Safety, Reliability, and Security.* Springer Berlin Heidelberg, 2007.

[8] Autosar consortium web page, Janvier 2014. http://www.autosar.org.

[9] *Avionics application software standard in- terface. ARINC specification 653 (part 1)*, 2006.

[10] José Rufino and Joao Craveiro. Robust partitioning and composability in arinc 653 conformant real-time operating systems. In *1st INTERAC Research Network Plenary Workshop, Braga, Portugal*, 2008.

[11] S. Santos, J. Rufino, T. Schoofs, C. Tatibana, and James Windsor. A portable arinc 653 standard interface. In *Digital Avionics Systems Conference, 2008. DASC 2008. IEEE/AIAA 27th*, octobre 2008.

[12] Jakob Engblom, Ermedahl Andreas, and Hans Hansson Mikael Sjödin, Jan Gustafsson. Worst-case execution-time analysis for embedded real-time systems. *International Journal on Software Tools for Technology Transfer*, October 2003.

[13] Jan Korst. Periodic multiprocessor scheduling. 1992.

[14] *Aircraft data network, Part 1: Systems con- cepts and overview. ARINC specification 664*, 2002.

[15] Aurélien Monot. *Vérification des contraintes temporelles de bout-en-bout dans le contexte AutoSar*. Thèse de doctorat, Université de Lorraine, Novembre 2012.

[16] Arvind Easwaran, Insup Lee, Oleg Sokolsky, and Steve Vestal. A Compositional Scheduling Framework for Digital Avionics Systems. In *Proceedings of the 2009 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, RTCSA '09, pages 371–380. IEEE Computer Society, 2009.

[17] Y.-H. Lee, Daeyoung Kim, M. Younis, and J. Zhou. Partition scheduling in apex runtime environment for embedded avionics software. In *Real-Time Computing Systems and Applications, 1998. Proceedings. Fifth International Conference on*, octobre 1998.

[18] K. Jeffay, D.F. Stanat, and C.U. Martel. On non-preemptive scheduling of period and sporadic tasks. In *Real-Time Systems Symposium, 1991. Proceedings., Twelfth*, decembre 1991.

[19] Mitra Nasri, Sanjoy Baruah, Gerhard Fohler, and Mehdi Kargahi. On the optimality of rm and edf for non-preemptive real-time harmonic tasks. In *Proceedings of the 22Nd International Conference on Real-Time Networks and Systems*, RTNS '14, 2014.

[20] Jan Korst, Emile Aarts, JanKarel Lenstra, and Jaap Wessels. Periodic multiprocessor scheduling. In *PARLE '91 Parallel Architectures and Languages Europe*. 1991.

[21] O. Kermia and Y. Sorel. Schedulability analysis for non-preemptive tasks under strict periodicity constraints. In *Embedded and Real-Time Computing Systems and Applications, 2008. RTCSA '08. 14th IEEE International Conference on*, aout 2008.

[22] I. Ripoll and R. Ballester-Ripoll. Period selection for minimal hyperperiod in periodic task systems. *Computers, IEEE Transactions on*, septembre 2013.

[23] V. Brocal, P. Balbastre, R. Ballester, and I. Ripoll. Task period selection to minimize hyperperiod. In *Emerging Technologies Factory Automation (ETFA), 2011 IEEE 16th Conference on*, septembre 2011.

[24] Chi-Sheng Shih, S. Gopalakrishnan, P. Ganti, M. Caccamo, and Lui Sha. Scheduling real-time dwells using tasks with synthetic periods. In *Real-Time Systems Symposium, 2003. RTSS 2003. 24th IEEE*, decembre 2003.

[25] Ching-Chih Han and Hung ying Tyan. A better polynomial-time schedulability test for real-time fixed-priority scheduling algorithms. In *Real-Time Systems Symposium, 1997. Proceedings., The 18th IEEE*, decembre 1997.

[26] M. Nasri, G. Fohler, and M. Kargahi. A framework to construct customized harmonic periods for real-time systems. In *Real-Time Systems (ECRTS), 2014 26th Euromicro Conference on*, juillet 2014.

[27] Yang Cai and M.C. Kong. Nonpreemptive scheduling of periodic tasks in uni- and multiprocessor systems. *Algorithmica*, 1996.

[28] Mitra Nasri and Mehdi Kargahi. Precautious-rm: a predictable non-preemptive scheduling algorithm for harmonic tasks. *Real-Time Systems*, 2014.

[29] Paul Bratley, Michael Florian, and Pierre Robillard. Scheduling with earliest start and due date constraints. *Naval Research Logistics Quarterly*, 18(4):511–519, 1971.

[30] Mathieu Grenier, Lionel Havet, and Nicolas Navet. Pushing the limits of CAN - scheduling frames with offsets provides a major performance boost. In *4th European Congress on Embedded Real Time Software (ERTS 2008)*, 2008.

[31] Michael Lauer. *Une méthode globale pour la vérification d'exigences temps réel - Application à l'Avionique Modulaire Intégrée.* Thèse de doctorat, Institut National Polytechnique de Toulouse, juin 2012.

[32] Jean-Yves Le Boudec and Patrick Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet.* Springer-Verlag, 2001.

[33] Fabrice Frances, Christian Fraboul, and Jérôme Grieu. Using network calculus to optimize the afdx network. In *ERTS 2006 : 3rd European Congress ERTS Embedded real-time software*, pages pp. 1–8, 2006.

[34] Steven Martin. *Maîtrise de la dimension temporelle de la qualité de service dans les réseaux.* PhD thesis, Paris 12, 2004.

[35] H. Bauer, J. Scharbarg, and C. Fraboul. Applying and optimizing trajectory approach for performance evaluation of afdx avionics network. In *Emerging Technologies Factory Automation, 2009. ETFA 2009. IEEE Conference on*, 2009.

[36] Muhammad Adnan. *Exact Worst-Case Communication Delay Analysis of AFDX Network.* Thèse de doctorat, Institut National Polytechnique de Toulouse, novembre 2013.

[37] Muhammad Adnan, Jean-Luc Scharbarg, Jérôme Ermont, and Christian Fraboul. An improved timed automata approach for computing exact worst-case delays of AFDX sporadic flows. In *Emerging Technologies and Factory Automation (ETFA), Krakow*, septembre 2012.

[38] Asma Mehiaoui, Ernest Wozniak, Sara Tucci Piergiovanni, Chokri Mraidha, Marco Di Natale, Haibo Zeng, Jean-Philippe Babau, Laurent Lemarchand, and Sébastien Gerard. A two-step optimization technique for functions placement, partitioning, and priority assignment in distributed systems. In *SIGPLAN/SIGBED Conference on Languages, Compilers and Tools for Embedded Systems 2013*, 2013.

[39] A. Mehiaoui, S. Tucci-Piergiovanni, J. Babau, and L. Lemarchand. Optimizing the deployment of distributed real-time embedded applications. In *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2012 IEEE 18th International Conference on*, aout 2012.

[40] E. Wozniak, A. Mehiaoui, C. Mraidha, S. Tucci-Piergiovanni, and S. Gerard. An optimization approach for the synthesis of autosar architectures. In *Emerging Technologies Factory Automation (ETFA), 2013 IEEE 18th Conference on*, septembre 2013.

[41] Brian Dougherty, Jules White, Russell Kegley, and Jonathan Preston. Deployment optimization for embedded flight avionics systems 1, 2010.

[42] Ahmad Al Sheikh. *Resource allocation in hard real-time avionic systems. Scheduling and routing problems.* PhD thesis, INSA Toulouse, Septembre 2011.

[43] Björn Annighöfer and Frank Thielecke. *Supporting the Design of Distributed Integrated Modular Avionics Systems with Binary Programming.* Deutsche Gesellschaft für Luft-und Raumfahrt-Lilienthal-Oberth eV, 2013.

[44] B. Annighofer and F. Thielecke. Multi-objective mapping optimization for distributed integrated modular avionics. In *Digital Avionics Systems Conference (DASC), 2012 IEEE/AIAA 31st*, octobre 2012.

[45] Xiuqiang He, Zonghua Gu, and Yongxin Zhu. Task allocation and optimization of distributed embedded systems with simulated annealing and geometric programming. *Computer Journal*, 2010.

[46] Ming Zhang and Zonghua Gu. Optimization issues in mapping autosar components to distributed multithreaded implementations. In *Rapid System Prototyping (RSP), 2011 22nd IEEE International Symposium on*, mai 2011.

[47] A. Hamann, R. Racu, and R. Ernst. Multi-dimensional robustness optimization in heterogeneous distributed embedded systems. In *Real Time and Embedded Technology and Applications Symposium, 2007. RTAS '07. 13th IEEE*, avril 2007.

[48] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *SCIENCE*, 1983.

[49] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

[50] L. V. Kantorovich. Mathematical Methods of Organizing and Planning Production. *Management Science*, juillet 1960.

[51] Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Rev.*, Fevrier 1991.

[52] Michael Lauer, Jérôme Ermont, Claire Pagetti, and Frédéric Boniol. Analyzing End-to-End Functional Delays on an IMA Platform (regular paper). In *ISoLA Symposium On Leveraging Applications of Formal Methods, Verification and Validation, Heraklion, Greece, 18/10/2010-21/10/2010*, octobre 2010.

[53] C. Fraboul and F. Martin. Modeling advanced modular avionics architectures for early real-time performance analysis. In *Parallel and Distributed Processing, 1999. PDP '99. Proceedings of the Seventh Euromicro Workshop on*, fevrier 1999.

[54] Nesrine Badache, Katia Jaffres-Runser, Jean-Luc Scharbarg, and Christian Fraboul. Managing temporal allocation in Integrated Modular Avionics (regular paper). In *Emerging Technologies and Factory Automation (ETFA), Barcelone, 16/09/2014-19/09/2014*, 2014.

[55] Yann Collette and Patrick Siarry. *Optimisation multiobjectif.* Ed. Techniques Ingénieur, 2002.

[56] Katia Jaffrès-Runser. *Methodologies for Wireless LAN Planning.* Thèse de doctorat, INSA Lyon, Octobre 2005.

[57] Katia Jaffrès-Runser, Jean-Marie Gorce, and Stéphane Ubéda. Mono-and multiobjective formulations for the indoor wireless lan planning problem. *Computers & Operations Research*, 2008.

[58] Katia Jaffrès-Runser, Jean-Marie Gorce, and Cristina Comaniciu. A multiobjective tabu framework for the optimization and evaluation of wireless systems. In Wassim Jaziri, editor, *Local Search Techniques: Focus on Tabu Search*, pages 29–54. I-Tech Education and Publishing, September 2008.

[59] Fred Glover and Manuel Laguna. *Tabu search.* Springer, 1999.

[60] Katia Jaffres-Runser, Jean-Marie Gorce, and Stephane Ubeda. Qos constrained wireless lan optimization within a multiobjective framework. *Wireless Communications, IEEE*, 2006.

[61] N. Srinivas and Kalyanmoy Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evol. Comput.*, 2(3):221–248, September 1994.

[62] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *SCIENCE*, 220:671–680, 1983.

[63] A. Colorni, Marco Dorigo, and Vittorio Maniezzo. Distributed optimization by ant colonies. In F. J. Varela and P. Bourgine, editors, *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 134–142. MIT Press, 1992.

[64] J Dréo, A Pétrowski, P Siarry, and E Taillard. Metaheuristiques pour l'optimisation difficile. *Eyrolles, Paris*, 2003.

[65] David Allen Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Air Force Institute of Technology, 1999.

Résumé

Cette thèse s'intéresse à la conception des systèmes modulaires avioniques IMA. Une des préoccupations majeures de cette étape est l'intégration des fonctions avioniques sur l'architecture matérielle cible (calculateurs et réseaux de communication). Cette intégration doit se faire en respectant des contraintes matérielles et des contraintes temps-réel dur imposées par la criticité les applications avioniques embarquées. Dans ces travaux, nous montrons qu'il est possible d'introduire une étape d'allocation temporelle permettant de trouver plus aisément une intégration admissible. Cette étape ajuste la périodicité de certaines fonctions communicantes de façon à optimiser conjointement la charge moyenne des modules et la marge temporelle moyenne dans le réseau. Elle va produire un ensemble de solutions optimales au sens de Pareto qui respectent toutes les contraintes temps-réel du système complet. Ces solutions Pareto-optimales représentent les meilleurs compromis possibles entre nos critères de charge et de marge réseau. Elles sont obtenues en résolvant un problème d'optimisation multicritères. La résolution de ce problème est réalisée ici selon deux algorithmes : (i) EXHAUST, un algorithme optimal basé sur une recherche exhaustive ; (ii) TABOU, un algorithme basé sur une métaheuristique Tabou multicritères. Les deux algorithmes exploitent une réduction de l'espace de recherche résultant de la présence de symétries et de la définition des fonctions de coût. Nous montrons qu'EXHAUST permet de travailler sur des systèmes de taille moyenne (une vingtaine de calculateurs), et que TABOU apporte une solution performante pour des systèmes de taille supérieure. L'intérêt de l'approche multicritères est de proposer différents compromis à l'intégrateur pour qu'il puisse sélectionner la solution qui lui convient le mieux.

Mots clés : contraintes de bout-en-bout, Allocation temporelle, IMA, Optimisation multiobjectif, Tabou

Abstract

The major concern in the integration of real-time embedded systems is the choice of functional parameters. The second is the variability in end-to-end delays. We treat the case of the IMA avionics architecture. We contribute to ease the integration process by the temporal allocation, formulated as a multiobjective optimization problem. We determine, for destination partitions, the maximum execution periods with the guarantee of system and real time constraints. These periods may be subsampled and propose various options for temporal allocation. A considerable number of configurations are possible for the same system by construction. Two metrics for quantifying a system are proposed: Load rate and the margin available in the network. Two multi-criteria optimization algorithms are used: EXHAUST an exhaustive search algorithm, and TABOO, more powerful multi-criteria, based on the tabu heuristic algorithm that gives solutions in a reasonable time. Both algorithms allow to extract interesting temporal allocation that balance the two proposed criteria.

Key words : end-to-end constraints, temporal allocation, IMA, multiobjective optimization, Tabu