

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ AKLI MOHAND OULHADJ BOUIRA

Faculté des Sciences et Sciences Appliquées

Département d'Informatique



Thèse de Doctorat en Science

Option Informatique

Optimisation Multi-Objectifs visant une Implémentation Automatique et Efficace d'Applications sur Réseau sur Puce 3D

Présenté par

Mr Maamar BOUGHERARA

Devant le jury composé de:

Président de jury	: M^r Mourad AMAD	Pr. Univ. Akli Mohand Oulhadj Bouira
Rapporteur	: M^{me} Nadia NEDJAH	Pr. State University of Rio de Janeiro
Co-Rapporteur	: M^r Djamel BENNOUAR	Pr. Univ. Akli Mohand Oulhadj Bouira
Examineur	: M^r Tahar BOUHADADA	Pr. Univ. Badji Mokhtar Annaba
Examinatrice	: M^{me} Hayet Farida MEROUANI	Pr. Univ. Badji Mokhtar Annaba
Examineur	: M^r Abbas AKLI	Dr. Univ. Akli Mohand Oulhadj Bouira
Examineur	: M^r Fouaz BERRHAIL	Dr. Univ. Ferhat Abbas Setif 1

Bouira, 2025

Dédicace

À la mémoire de mes regrettés chers père et mère,

Aucune langue ne saurait exprimer pleinement ma gratitude envers vous, ni rendre hommage à tout ce que je vous dois. Vous êtes les piliers de ma vie, les guides qui m'ont inspiré et soutenu à chaque étape de mon parcours.

Je tiens également à exprimer toute ma reconnaissance à ma cher femme Nawal, pour son soutien indéfectible, sa compréhension et son amour sans faille tout au long de cette période de recherche. Ses encouragements et son soutien inconditionnel ont été une source d'inspiration et de motivation essentielle dans les moments les plus difficiles. Son dévouement et son soutien inestimable ont rendu possible la réalisation de ce projet, et je lui en suis profondément reconnaissant. À mon fils Samy et ma fille Houda.

À mes frères et sœurs Djamel, Abdelkader, Fatma, Ilhem, et Sihem.

À toute la famille Bougherara, ainsi qu'à Fodil et Ouadah.

Remerciements

Je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué à la réalisation de cette thèse de doctorat.

Tout d'abord, je tiens à exprimer ma sincère gratitude envers ma directrice de thèse, le Professeur Nadia Nedjah, pour son soutien inestimable, son encadrement rigoureux et ses conseils éclairés qui ont grandement enrichi cette recherche. Sa passion pour la science et son dévouement envers mes progrès académiques ont été une source constante d'inspiration. Je suis reconnaissant pour le temps qu'elle a investi pour m'aider à mener à bien ce travail.

Je tiens à exprimer ma sincère gratitude envers mon co-directeur de thèse, Djamel Bennouar, pour sa précieuse contribution à ce travail de recherche. Ses conseils avisés, son expertise et son soutien constant ont été d'une aide inestimable tout au long de ce parcours académique. Ses encouragements et sa disponibilité ont été des éléments déterminants dans la réussite de cette thèse.

Je tiens à exprimer ma profonde gratitude aux membres du jury pour le temps précieux qu'ils ont consacré à lire, analyser et évaluer cette thèse. Leur engagement, leur expertise et leurs remarques constructives sont d'une grande valeur pour moi.

Je tiens à remercier chaleureusement Ouadah Saadi pour son soutien constant et ses corrections avisées, qui ont contribué à l'amélioration de ma thèse. Ses encouragements ont été précieux tout au long de ce parcours académique.

Je souhaite exprimer ma reconnaissance envers mes amis et collègues de l'École Normale Supérieure, dont la collaboration et l'enthousiasme ont créé un environnement de travail stimulant. À Rafik, Fouaz, Anis, Yazid, Ryadh, hachemi, youssef, karim sans oublier Ryma, je suis reconnaissant pour votre soutien et votre amitié.

À tous mes amis qui me connaissent et à toutes les personnes qui m'ont aidé de près ou de loin à réaliser cette thèse, je vous adresse mes sincères remerciements. Votre soutien et vos encouragements ont été d'une importance capitale tout au long de ce parcours.

Je remercie également nos étudiants sans qui cette thèse n'aurait sans doute pas vu le jour.

Abstract

Networks on chip represent a new concept in system on chip interconnections, facilitating and optimizing the integration of complex components. Although this technology is relatively new, further research is needed, especially to expedite and streamline the design phases of these networks. Two dimensional networks on chip are widely used but have limitations in terms of size and performance. To overcome these limitations, three dimensional networks on chip stack multiple layers of 2D meshes, significantly improving the network's capacity and performance. The design of networks on chip involves optimizing them for a specific application, which is divided into tasks implemented by intellectual property blocks. The design process includes intermediate steps. First, we assign and map the intellectual property blocks that implement the various tasks of the application. This allocation and placement is crucial to ensure effective use of available resources. Then, it is necessary to determine the communication routing, this step is essential to avoid congestion and minimize delays. The problems of Assignment, Mapping, and routing are classified as NP hard problems, meaning it is difficult to find an optimal solution in a reasonable time. That is why multi objective optimization algorithms are used. The approach proposed in this thesis involves using two algorithms : one based on particle swarm behavior and the other based on differential evolution. Problem modeling is performed for each of the three phases, along with the definition of objective functions. The results obtained, based on benchmarks, demonstrated the effectiveness of the proposed approaches compared to those of other algorithms. Differential evolution achieved better results in all three phases.

Keywords: System on Chip, Networks on Chip, Three dimensional Networks on Chip, Multi objective optimization, Particle swarm optimization, Differential evolution, Assignment, Mapping, Routing.

Résumé

Les réseaux sur puce représentent un nouveau concept dans les interconnexions des systèmes sur puce, facilitant et optimisant l'intégration de composants complexes. Bien que cette technologie soit relativement nouvelle, des recherches supplémentaires sont nécessaires, en particulier pour accélérer et simplifier les phases de conception de ces réseaux. Les réseaux sur puce en deux dimensions sont largement utilisés, mais ils présentent des limitations en termes de taille et de performance. Pour dépasser ces limitations, les réseaux sur puce tridimensionnels empilent plusieurs couches de maillages 2D, ce qui améliore la capacité et les performances du réseau de manière significative. La conception des réseaux sur puce implique son optimisation pour une application spécifique, qui est divisée en tâches implémentées par des blocs de propriété intellectuelle. Le processus de conception comprend des étapes intermédiaires. Tout d'abord, il faut allouer et placer les blocs de propriété intellectuelle qui implémentent les différentes tâches de l'application. Cette affectation et ce placement sont cruciaux pour garantir une utilisation efficace des ressources disponibles. Ensuite, il faut déterminer le routage des communications afin d'éviter les congestions et de minimiser les retards de transmission. Les problèmes d'affectation, de placement et de routage sont classés comme des problèmes (NP) difficiles, ce qui signifie qu'il est difficile de trouver une solution optimale en un temps raisonnable. C'est pourquoi des algorithmes d'optimisation multi objectifs sont utilisés. L'approche proposée dans cette thèse consiste à utiliser deux algorithmes, l'un basé sur le comportement des essaims de particules et l'autre basé sur l'évolution différentielle. Une modélisation du problème est réalisée pour chacune des trois phases, ainsi que la définition des fonctions objectives. Les résultats obtenus, basés sur des applications (*benchmarks*), ont montré une efficacité des approches proposées par rapport à ceux d'autres algorithmes. L'évolution différentielle a réussi à obtenir de meilleurs résultats en ce qui concerne les trois phases.

Mots clés: Système sur puce, Réseaux sur puce, Réseaux sur puce tridimensionnels, Optimisation multi objectifs, Optimisation par essaim de particules, Évolution différentielle, Affectation, Placement, Routage.

الملخص

تمثل الشبكات على الرقاقة مفهومًا جديدًا في نظام التوصيلات البينية للرقاقة، مما يسهل ويحسن تكامل المكونات المعقدة. وعلى الرغم من أن هذه التكنولوجيا جديدة نسبيًا، إلا أن هناك حاجة إلى مزيد من البحث، خاصة لتسريع وتبسيط مراحل تصميم هذه الشبكات. تُستخدم الشبكات ثنائية الأبعاد الموجودة على الرقاقة على نطاق واسع ولكن لها قيود من حيث الحجم والأداء. للتغلب على هذه القيود، تقوم الشبكات ثلاثية الأبعاد الموجودة على الرقاقة بتكديس طبقات متعددة من الشبكات ثنائية الأبعاد، مما يؤدي إلى تحسين قدرة الشبكة وأدائها بشكل كبير. يتضمن تصميم الشبكات على الرقاقة تحسينها لتنفيذ تطبيق معين، والذي يتكون من مجموعة من المهام تنفذ من طرف مجموعة من المعالجات. تتضمن عملية التصميم خطوات وسيطة. أولاً، يجب إسناد المعالجات التي تنفذ المهام المختلفة للتطبيق ثم وضعها على الشبكة. تعد عملية الإسناد و الوضع أمران بالغان الأهمية لضمان الاستخدام الفعال للمصادر المتاحة. بعد ذلك، من الضروري تحديد مسار الاتصال، وهذه الخطوة ضرورية لتجنب الازدحام وتقليل التأخير. تصنف مشاكل الإسناد و الوضع والتوجيه على أنها مشاكل جد لصعبة، مما يعني أنه من الصعب إيجاد الحل الأمثل في وقت معقول. ولهذا السبب يتم استخدام خوارزميات التحسين متعددة الأهداف. يتضمن النهج المقترح في هذه الأطروحة استخدام خوارزميتين : واحدة تعتمد على سلوك سرب الحسيمات والأخرى تعتمد على التطور التفاضلي. يتم تنفيذ نمذجة المشكلة لكل مرحلة من المراحل الثلاث، إلى جانب تعريف دوال الهدف. أظهرت النتائج التي تم الحصول عليها، بناءً على المعايير، فعالية الأساليب المقترحة مقارنة بتلك الخاصة بالخوارزميات الأخرى.

الكلمات المفتاحية : النظام على الرقاقة، الشبكات على الرقاقة، الشبكات ثلاثية الأبعاد على الرقاقة، تحسين الأهداف المتعددة، خوارزمية سرب الحسيمات، التطور التفاضلي، الإسناد، الوضع، التوجيه

Table des matières

Liste des figures	xiii
Liste des tables	xvi
Liste des algorithmes	xvii
Liste des acronymes	xviii
1 Introduction générale	1
1.1 Problématique	3
1.2 Contributions	3
1.3 Organisation	5
2 Réseaux sur puce	6
2.1 Définition d'un système embarqué	7
2.2 Évolution des interconnexions dans les SoCs	8
2.2.1 Connexion point à point	8
2.2.2 Connexion par bus	8
2.2.3 Connexion par bus hiérarchique	9
2.2.4 Réseau sur puce	10
2.3 Terminologie des réseaux sur puce	11
2.3.1 Architecture du réseaux sur puce	11
2.3.2 Caractéristiques des réseaux sur puce	14
2.3.3 Mesure des performances d'un réseaux sur puce	23
2.4 Réseaux sur puce en 3D	23
2.4.1 Avantage du 3D par rapport au 2D	25
2.4.2 Composants du réseau sur puce 3D	25
2.5 Conception des réseaux sur puce	30
2.5.1 Affectation des tâches aux IPs	31
2.5.2 Placement des IPs	32
2.5.3 Routage des communications	32
2.6 Considération finale du chapitre	34

3	Etat de l'art des travaux de recherche concernant les problèmes de conception du NoC	35
3.1	Affectation des tâches	36
3.2	Placement des IPs	37
3.2.1	Placement utilisant une méthode exacte	38
3.2.2	Placement utilisant une méta-heuristique	38
3.3	Routage	43
3.4	Discussion	45
3.5	Considération finale du chapitre	47
4	Optimisation basée sur les méta-heuristiques	48
4.1	Notions sur la complexité	48
4.2	Problème d'optimisation	49
4.3	Méthodes de résolution des problèmes d'optimisation	50
4.3.1	Méthodes exactes	51
4.3.2	Heuristiques	51
4.3.3	Méta-heuristiques	52
4.3.4	Hyper-heuristiques	54
4.4	Optimisation multi-objectifs	54
4.4.1	Concept de Pareto	55
4.4.2	Défis des problèmes multi-objectifs	56
4.4.3	Classification des méthodes d'optimisation multi-objectifs	57
4.5	Métriques de performances	59
4.5.1	Indicateurs basés sur la convergence	60
4.5.2	Indicateurs basés sur la diversité	61
4.5.3	Indicateurs hybrides	62
4.6	Considération finale du chapitre	62
5	Méta-heuristiques exploitées	63
5.1	Choix des méthodes exploitées	63
5.2	Essaims de Particules	64
5.2.1	Voisinage	65
5.2.2	Étapes des essaims de particules	66
5.2.3	PSO pour les problèmes multi-objectifs	68
5.3	Évolution différentielle	72
5.3.1	Les étapes de l'algorithme DE	74
5.3.2	DE pour l'optimisation multi-objectifs	76
5.4	Considération finale du chapitre	77
6	Méthode proposée pour la conception d'un réseau sur puce 3D	79
6.1	Phase d'affectation	79

6.1.1	Définition du problème d'affectation	80
6.1.2	Complexité du problème d'affectation	80
6.1.3	Codification du problème d'affectation	81
6.1.4	Fonctions objectifs du problème d'affectation	83
6.2	La Phase de placement	86
6.2.1	Définition du problème de placement	86
6.2.2	Complexité du problème de placement	87
6.2.3	Codification du problème de placement	89
6.2.4	Fonctions objectifs du problème de placement	89
6.3	Phase de routage	94
6.3.1	Définition du problème de routage	94
6.3.2	Complexité du problème de routage	95
6.3.3	Codification du problème de routage	95
6.3.4	Fonction objectif du problème de routage	98
6.4	Considération finale du chapitre	100
7	Tests et résultats	102
7.1	Benchmarks utilisés	102
7.1.1	Suite de synthèse pour les systèmes embarqués	102
7.1.2	Graphe de tâches gratuit	104
7.1.3	Schéma de routage	104
7.2	Caractéristiques de la machine utilisée	105
7.3	Résultats de la phase d'affectation	105
7.3.1	Résultats de MOPSO	106
7.3.2	Résultats de DEMO	108
7.3.3	Comparaison	111
7.4	Résultats de la phase de placement	116
7.4.1	Résultats de MOPSO	117
7.4.2	Résultats de DEMO	118
7.4.3	Comparaison	120
7.5	Résultats de la phase de routage	124
7.5.1	Résultats de PSO et DE	126
7.5.2	Comparaison	127
7.6	Considération finale du chapitre	133
8	Conclusion générale et perspectives	135
8.1	Conclusions	135
8.2	Recherches futures	137
	Bibliographie	139

ANNEXE A - Processeurs et Tâches du référence	155
ANNEXE B - Graphes de tâche de référence	158
ANNEXE C - Résultats de Testes	170

Liste des figures

2.1	Organisation d'un SoC et MPSoC	7
2.2	Connexion point à point	9
2.3	Connexion par bus	9
2.4	Connexion par Bus hiérarchique	10
2.5	Connexion par Réseaux d'interconnexion	10
2.6	Les éléments du NoC et couches réseau	12
2.7	Les trois types de composants d'un Réseau sur puce	12
2.8	Les Composants d'un Adaptateur réseau	13
2.9	Exemple de composants d'un Routeur	13
2.10	Topologie en anneau	15
2.11	Topologie en 2D	16
2.12	Topologie en arbre	17
2.13	Topologie irrégulière	17
2.14	Structure d'un message	18
2.15	Classification des algorithmes de routage	22
2.16	L'idée de NoC 3D	24
2.17	Un réseau sur puce 3D	24
2.18	Avantage du NoC 3D & NoC 2D	26
2.19	Routeur et liens horizontaux et verticaux dans un NoC 3D.	26
2.20	Un élément de routeur 3D	27
2.21	Crossbar du Routeur 3D	27
2.22	Crossbar 3D Du routeur 3D	28
2.23	Routeur selon Directions	28
2.24	Type de liens verticaux	29
2.25	Les étapes de conception d'une application sur un NoC	31
2.26	Le problème d'affectation	32
2.27	Le problème de placement	33
2.28	Problème de routage	33
3.1	Les classes de méthodes d'optimisation sur les NoCs	35
4.1	Classe des méthodes d'optimisation	50
4.2	Classes des Méta-heuristiques	52
4.3	Méthodes d'optimisation multi-objectifs (point de vue décideur)	57

4.4	Méthodes d'optimisation multi-objectifs (point de vue concepteur)	58
5.1	Topologie de voisinage	66
5.2	Topologie en clusters	66
5.3	Diagramme de déplacement d'une particule	67
6.1	Graphe de Taches	81
6.2	Graphe de tâches avec deux affectations possibles	81
6.3	Exemple de code en XML	82
6.4	Codification d'une solution d'affectation	83
6.5	Codification d'une solution de placement	89
6.6	Communication entre la même source et plusieurs destination	92
6.7	Communication entre plusieurs source et la même destination	93
6.8	Plusieurs chemins possibles pour une communication.	96
6.9	Chemin minimal et chemin non minimal	97
6.10	Exemple de routage avec congestion	99
6.11	Exemple de routage sans Congestion	100
7.1	Comparaison des affectations obtenues par MOPSO concernant l'énergie .	106
7.2	Comparaison des affectations obtenues par MOPSO concernant le temps .	106
7.3	Comparaison des affectations obtenues par MOPSO concernant la surface .	107
7.4	Comparaison des affectations obtenues par DEMO concernant l'énergie . .	109
7.5	Comparaison des affectations obtenues par DEMO concernant le temps . .	109
7.6	Comparaison des affectations obtenues par DEMO concernant la surface .	109
7.7	Valeurs de contribution obtenue par les cinq version de DEMO	110
7.8	Solutions non dominées obtenues par graphes de tâche de E3S	111
7.9	Analyse comparative des affectations obtenue sur les graphes de tâches E3S par DEMO et MOPSO concernant l'énergie	112
7.10	Analyse comparative des affectations obtenue sur les graphes de tâches E3S par DEMO et MOPSO concernant la surface	112
7.11	Analyse comparative des affectations obtenue sur les graphes de tâches E3S par DEMO et MOPSO concernant le temps	112
7.12	Solutions non dominées obtenues sur les graphes de tâches de TGFF	113
7.13	Comparaison des affectations obtenue sur les graphes de tâches TGFF par DEMO et MOPSO concernant l'énergie	113
7.14	Comparaison des affectations obtenue sur les graphes de tâches TGFF par DEMO et MOPSO concernant la surface	113
7.15	Comparaison des affectations obtenue sur les graphes de tâches TGFF par DEMO et MOPSO concernant le temps d'exécution	114
7.16	Valeurs de contribution obtenue par DEMO et MOPSO	115
7.17	Comparaison des placement obtenues par MOPSO concernant l'énergie . .	117

7.18	Comparaison des placement obtenues par MOPSO concernant le temps . .	117
7.19	Comparaison des placement obtenues par MOPSO concernant la surface .	118
7.20	Comparaison des placement obtenues par DEMO Rand_1 concernant l'énergie	119
7.21	Comparaison des placement obtenues par DEMO Rand_1 concernant le temps	119
7.22	Comparaison des placement obtenues par DEMO Rand_1 concernant la surface	119
7.23	Comparaison des placement obtenues par DEMO Best_1 concernant l'énergie	119
7.24	Comparaison des placement obtenues par DEMO Best_1 concernant le temps	120
7.25	Comparaison des placement obtenues par DEMO Best_1 concernant la surface	120
7.26	Solutions non dominées obtenues par DEMO et MOPSO	121
7.27	Comparaison du placement 2D obtenu en fonction des exigences de puissance	121
7.28	Comparaison du placement 2D obtenu en fonction des exigences du temps	121
7.29	Comparaison du placement 2D obtenu en fonction des exigences de surface	122
7.30	Comparaison du placement 3D obtenu en fonction des exigences de puissance	122
7.31	Comparaison du placement 3D obtenu en fonction des exigences du temps	122
7.32	Comparaison du mappage 3D obtenu en fonction des exigences de surface .	122
7.33	Valeurs obtenue de Congestion par DE et PSO	126
7.34	Valeurs obtenue de latence par DE et PSO	126
7.35	Valeurs obtenue de bande passante par DE et PSO	127
7.36	Résultat obtenu de latence pour un schéma de routage Rand	128
7.37	Résultat obtenu de latence pour un schéma de routage Hotspot	129
7.38	Résultat obtenu de latence pour un schéma de routage Local	129
7.39	Résultat obtenu de latence pour le schéma de routage Transpose_1	130
7.40	Résultat obtenu de latence pour le schéma de routage Transpose_2	130
7.41	Résultat obtenu de latence pour le schéma de routage Complement	131
7.42	Résultat de latence obtenue pour pour le graphe de tâches TG0	132
7.43	Résultat de Latance obtenue pour pour le graphe de tâches TG1	132
7.44	Résultat de latence obtenue pour pour le graphe de tâches TG2	132
7.45	Résultat de latence obtenue pour pour le graphe de tâches TG3	132
7.46	Résultat de latence obtenue pour pour le graphe de tâches TG4	133
B.1	Graphe de tâches du benchmark Auto-industriel 0	159
B.2	Graphe de tâches du benchmark Auto-industriel 1	159
B.3	Graphe de tâches du benchmark Auto-industriel 2	159
B.4	Graphe de tâches du benchmark Auto-industriel 3	160
B.5	Graphe de tâches du benchmark Consommateur 0	160
B.6	Graphe de tâches du benchmark Consommateur 1	161
B.7	Graphe de tâches du benchmark Networking 1	161
B.8	Graphe de tâches du benchmark Networking 2	162
B.9	Graphe de tâches du benchmark Networking 3	162
B.10	Graphe de tâches du benchmark Bureautique 0	162

B.11 Graphe de tâches du benchmark Télécom 0	163
B.12 Graphe de tâches du benchmark Télécom 1	163
B.13 Graphe de tâches du benchmark Télécom 2	164
B.14 Graphe de tâches du benchmark Télécom 3	164
B.15 Graphe de tâches du TG0	165
B.16 Graphe de tâches du TG1	165
B.17 Graphe de tâches du TG2	166
B.18 Graphe de tâches du TG3	166
B.19 Graphe de tâches du TG4	167

Liste des tables

2.1	Comparaison entre les structures d'interconnexion	11
2.2	Comparaison entre les différents modes de commutation	20
5.1	Statistiques sur les travaux publiés selon Google Scholar, IEEE et SCOPUS	64
5.2	Stratégies de mutation utilisées	74
6.1	Nombre d'IPs possible	81
6.2	Nombre de placements selon les processeurs et les ressources du NoCs . . .	88
6.3	Nombre de solutions de placement possibles	88
6.4	Nombre de chemins possibles	95
6.5	Plusieurs chemins d'une même paire	97
6.6	Solution complète du routage	97
6.7	Table de routage d'un exemple d'application	98
7.1	Détail des benchmarks des cinq types d'application	103
7.2	Détail des graphes de tâches	104
7.3	Résultats d'affectation des IPs avec MOPSO pour les benchmarks de E3S .	107
7.4	Resultats d'Affectation des IP avec MOPSO pour les benchmarks de TGFF	108
7.5	Resultats d'Affectation des IPs avec DEMO Rand_1 pour les benchmarks de E3S	108
7.6	Resultats d'Affectation des IPs avec DEMO Rand_1 pour les benchmark de TGFF	111
7.7	Métriques de performances obtenues pour l'affectation des benchmarks de E3S	115
7.8	Métriques de performances obtenues pour l'affectation des benchmarks de TGFF	115
7.9	Métriques de performances obtenues pour le placement en 2D	123
7.10	Métriques de performances obtenues pour le placement en 3D	123
7.11	Paramètre de configuration du simulateur Access-Noxim	125
7.12	Nœuds de destination des schémas de routage déterministes	125
A.1	Liste de processeurs de la bibliothèque	155
A.2	Liste de tâches utilisées dans la bibliothèque	156
A.3	Liste de tâches par processeur	157
A.4	Listes de processeurs par tâche utilisées dans la bibliothèque	157

B.1	Schéma de routage déterministes en 2D	168
B.2	Schéma de routage déterministes en 3D	169
C.1	Affectation des IPs avec un ordonnancement ASAP utilisant MOPSO	170
C.2	Affectation des IPs avec un ordonnancement ALAP utilisant MOPSO	171
C.3	Affectation des IPs avec un ordonnancement LS utilisant MOPSO	171
C.4	Résultats d'affectation des IPs avec MOPSO pour les benchmarks E3S	172
C.5	Résultats d'affectation des IPs avec DEMO rand_1 pour les benchmarks E3S	172
C.6	Résultats d'affectation des IPs avec DEMO best_1 pour les benchmarks E3S	172
C.7	Résultats d'affectation des IPs avec DEMO current to best pour les benchmarks E3S	173
C.8	Résultats d'affectation des IPs avec DEMO rand_2 pour les benchmarks E3S	173
C.9	Résultats d'affectation des IPs avec DEMO best_2 les benchmarks E3S	173
C.10	Résultats d'affectation des IPs avec DEMO best_1 pour les benchmarks TGFF	174
C.11	Exemple de Solutions non dominées trouvée par MOPSO pour les benchmarks E3S	174
C.12	Exemple de Solutions non dominées trouvée par DEMO best_1 pour les benchmarks E3S	174
C.13	Résultats de placement des IPs avec MOPSO en 2D pour les benchmarks de TGFF	175
C.14	Résultats de placement des IPs avec DEMO rand_1 en 2D pour les benchmarks de TGFF	175
C.15	Résultats de placement des IPs avec DEMO best_1 en 2D pour les benchmarks de TGFF	176
C.16	Résultats de placement des IPs avec MOPSO en 3D pour les benchmarks de TGFF	176
C.17	Résultats de placement des IPs avec DEMO rand_1 en 3D pour les benchmarks de TGFF	176
C.18	Résultats de placement des IPs avec DEMO best_1 en 3D pour les benchmarks de TGFF	177
C.19	Exemple de solutions de placement non dominées trouvée par MOPSO en NoC 2D	177
C.20	Exemple de solutions de placement non dominées trouvée par MOPSO en NoC 3D	177
C.21	Exemple de solutions de placement non dominées trouvée par DEMO best_1 en NoC 2D	177
C.22	Exemple de solutions de placement non dominées trouvée par DEMO best_1 en NoC 3D	178
C.23	Résultat de routage utilisant PSO en NoC 2D	178
C.24	Résultat de routage utilisant PSO en NoC 3D	178

C.25 Résultat de routage utilisant DE en NoC 2D	179
C.26 Résultat de routage utilisant DE en NoC 3D	179
C.27 Latence obtenue pour le graphe de tâches TG0	179
C.28 Latence obtenue pour le graphe de tâches TG1	179
C.29 Latence obtenue pour le graphe de tâches TG2	180
C.30 Latence obtenue pour le graphe de tâches TG3	180
C.31 Latence obtenue pour le graphe de tâches TG4	180

Liste des algorithmes

1	Les étapes principales de PSO	69
2	Les étapes principales de MOPSO modifié	71
3	Mise à jours l'archive locale	72
4	Mise à jours l'archive externe	72
5	Calcul de la distance d'agglomération	72
6	Les étapes principales de GA	73
7	Les étapes principales de DE	75
8	Le principe de l'opération de sélection	77
9	Les éptapes principales de DEMO	77
10	Temps d'exécution de k tâches d'un même niveau par le même processeur .	84
11	Ordonnancement <i>ASAP</i>	85
12	Ordonnancement <i>ALAP</i>	85
13	Ordonnancement <i>LS</i>	86

Liste des acronymes

ASAP	AS Soon AS Possible
ALAP	AS Late AS Possible
DE	Differential Evolution
DEMO	Differential Evolution for Multiobjective Optimization
EDA	Electronic Design Automation
E3S	Embedded System Synthèses Suit
IP	Intellectual Property
GA	Genetic Algorithm
LS	List Scheduling
MOPSO	Multi-Objective Particle Swarm Optimization
NoC	Network on Chip
NI	Network Interface
PSO	Particle Swarm Optimization
SoC	System on Chip
TSV	Through Silicon Via
TGFF	Task Graph For Free
WH	Wormhole

Chapitre 1

Introduction générale

L'évolution technologique dans le domaine de l'électronique et des semi-conducteurs a permis d'intégrer plus d'un milliard de transistors dans un même substrat de silicium [1]. Selon la loi de Moore, la taille physique du transistor franchira le seuil de 10 nm en 2025 [2]. La possibilité d'intégrer un grand nombre de transistors dans une même puce a incité les concepteurs de circuits intégrés à implanter beaucoup plus de fonctionnalités et d'architectures dans un même circuit afin de réduire le coût de production. En effet, un circuit de technologie récente peut remplacer des dizaines ou même des centaines de circuits avec des technologies primitives [1].

Cette croissance de l'intégration a donc poussé les concepteurs de circuits intégrés à s'orienter vers la réutilisation de composants matériels déjà conçus appelés Propriété Intellectuelle (IP – *Intellectual Property*) qui peuvent servir d'éléments de base pour concevoir de nouveaux circuits. Ceci permet de placer plusieurs composants sur la même puce pour concevoir un système sur puce (SoC – *System on Chip*) qui accélérera le développement et réduira ainsi le délai de mise sur le marché (*time to market*). Les SoCs deviennent de plus en plus un aspect essentiel de nos vies professionnelles et personnelles. Parmi tant d'autres, on les trouve dans les voitures, les avions, les systèmes de défense, les systèmes médicaux, ainsi que les smart phones, les téléviseurs intelligents et les consoles de jeux.

Le SoC peut contenir du matériel tel que des processeurs, des mémoires, des contrôleurs, des processeurs de signaux numériques. Le principal avantage du SoC est sa faible consommation d'énergie, son coût réduit et sa meilleure fiabilité que les systèmes multi puces qu'il a remplacé [3].

Avec les intégrations à grande échelle et la transition vers les systèmes intégrés sur une seule puce, le nombre des IPs augmente de jour en jour dans les SoCs et soulève ainsi de nouveaux défis de leur conception. Parmi ces défis, on trouve la communication entre les IPs. Dans un SoC, les IPs sont interconnectés par une infrastructure de communication,

comme les bus ou les canaux point à point. Du fait que les systèmes actuels se caractérisent par de fortes communications, les architectures à base de bus sont devenues inefficaces. La communication au sein de SoC est devenue un sujet de recherche important avec l'augmentation de la puissance de traitement et la diffusion des applications gourmandes en données. Cette situation incite les chercheurs à établir un nouveau moyen de communication entre les composants [4].

Les solutions utilisées dans les réseaux informatiques pour les problèmes de communication à grande échelle ont été considérées par les chercheurs comme pouvant être utilisées dans la communication sur puce. L'objectif est de fournir une communication entre les composants de la puce avec une structure de réseau simple. Cette architecture, appelée Réseau sur Puce ou (NoC – *Network on Chip*), offre une conception plus efficace et permet de connecter plusieurs composants communiquant simultanément à travers des routeurs et de courts fils d'interconnexion.

Grâce à cette architecture, les réseaux sur puce à 2 dimensions (2D) englobent l'intégration d'un grand nombre de IPs avec une consommation d'énergie meilleure par rapport aux systèmes traditionnels [5]. Cependant, et même si les réseaux sur puce 2D englobent l'intégration d'un grand nombre de IP, ils ne sont toujours pas considérés comme la solution ultime pour la communication dans les futurs systèmes sur puce à très grande échelle [6]. La distance entre les IP augmente, ce qui entraîne une augmentation des délais de transmission, de consommation d'énergie et de la taille globale de la puce, ce qui a un impact grave sur les performances. Afin de répondre à une densité de données croissante et à des besoins de performances plus élevés, il est devenu inévitable de passer de l'architecture 2D à l'architecture 3D [7].

Dans l'architecture de puce 3D, les couches de la puce sont subdivisées en segments plus petits qui sont ensuite superposés les uns sur les autres. La communication est assurée à travers des liens d'interconnexions verticaux nommés (TSV – *Through Silicon Via*). Ces TSVs ont encouragé les chercheurs à implémenter leurs réseaux en utilisant cette nouvelle technologie pour en exploiter les avantages par rapport aux circuits intégrés à 2 dimensions d'où la naissance d'un réseaux sur puces 3D (NoC 3D) [8].

Grâce aux liens courts, l'intégration 3D dans la conception des circuits consomme moins d'énergie que dans les circuits 2D. Ils réduisent, également, la latence. Dans le cadre de cette thèse, nous nous intéressons au travail de challenge consistant l'implémentation efficace d'une application quelconque sur un réseau sur puces 3D. Dans la suite, nous détaillons la problématique abordée dans cette thèse..

1.1 Problématique

Les SoC et les NoC sont implémentés pour exécuter une application spécifique. Une application peut être comprise comme un ensemble de tâches à exécuter en respectant certaines dépendances de données et de contrôle. La manière d'exprimer les caractéristiques d'une application peut se réaliser à travers un graphe de tâches (TG – *Task Graph*) orienté et acyclique.

Pour aider le concepteur à bien concevoir un réseau sur puces, des outils informatiques d'aide à la conception sont utilisés. Ils sont connus sous le nom de EDA (*Electronic Design Automation*). Le but de ces outils est d'automatiser les étapes de conception de circuits électronique afin d'optimiser l'implémentation finale à partir d'une description abstraite spécifiée par le concepteur.

L'outil EDA, utilisant la description abstraite fournie par le concepteur, doit être capable de réduire le niveau d'abstraction de la description par étapes successives d'optimisation jusqu'à arriver à une mise en œuvre finale conforme aux spécifications de conception. Parmi les étapes d'optimisation, certaines sont classées comme des problèmes complexes à résoudre dans le domaine de la conception NoC.

Dans ce contexte, notre objectif est de proposer un ensemble d'approches permettant de résoudre les trois phases essentielles dans la conception d'un NoC et ce, en les transformant en problèmes d'optimisation. Ces trois phases sont connus comme des problèmes *NP* difficiles. Ils ne peuvent pas être résolus avec un outil simple d'où la nécessité d'utiliser des méta-heuristiques d'optimisation.

1.2 Contributions

Les principales contributions de cette thèse sont en nombre de trois, une pour chaque étape de l'approche proposée, qui permet l'implémentation complète d'une application sur un réseau sur puce 3D. Ces étapes sont l'affectation des tâches aux nœuds du réseau, le placement des tâches et le routage qui permet leurs communication.

En premier lieu, nous modélisons le problème d'affectation des tâches dans le réseau en utilisant des méthodes d'optimisation, à savoir: une méthode d'optimisation basée sur le comportement des oiseaux ou essaims de particules (PSO – *Particle Swarm Optimization*) et une autre basée sur l'évolution différentielle (DE – *Differential Evolution*). Étant donné que nous définissons plusieurs objectifs contradictoires qui sont la consommation d'énergie, la superficie et le temps d'exécution, il est naturel que cela nous conduise à une optimisation multi-objectifs. Ce travail a géré trois publications. La première est basée sur l'utilisation de DEMO (*Differential Evolution for Multiobjective Optimization*) et les deux autres sur MOPSO (*Multi-Objective Particle Swarm Optimization*):

1. M. Bougherara, N. Nedjah, D. Bennouar, R. Rahmoun, A. Sadok and L. M. Mourelle, *Core/Task Associations for Efficient Application Implementation on Network-on-Chip*, Proc. of the International Conference on Computer and Applications (ICCA), Beirut, Lebanon, August 25-26, 2018, IEEE Press, pp. 18-22, 2018, doi: 10.1109/CO-MAPP.2018.8460231.
2. M. Bougherara, R. Kemcha, N. Nedjah, D. Bennouar, L. M. Mourelle, *IP Assignment Optimization for an Efficient NoC-based System using Multi-objective Differential Evolution*. In: Proc. of the 7th International Conference on Metaheuristics and Nature Inspired computing (META), Marrakech, Morocco, Oct 27-31, 2018, v. 1. pp. 435-444, 2018.
3. M. Bougherara, N. Nedjah, L. M. Mourelle, R. Rahmoun, A. Sadok, D. Bennouar: *IP assignment for efficient NoC-based system design using multi-objective particle swarm optimisation*. *Int. J. Bio Inspired Comput.* 12(4): 203-213, 2018, doi: 10.1504/IJ-BIC.2018.096483 (IF: 3.295).

Dans un deuxième lieu, nous modélisons le problème de placement multi-objectifs qui utilise les résultats du problème d'affectation comme information initiale pour trouver une solution au problème de placement. Les objectifs contradictoires sont étendus pour ce problème. Les méthodes d'optimisation utilisées sont les mêmes: MOPSO et DEMO. Ce travail agéré deux publications. La première est basée sur l'utilisation de MOPSO et la deuxième sur DEMO:

1. M. Bougherara, N. Nedjah, D. Bennouar, R. Kemcha, L. M. Mourelle: *Efficient Application Mapping onto Three-Dimensional Network-on-Chips Using Multi-Objective Particle Swarm Optimization*. Proc. of the 19th International Conference on Computational Science and Its Applications (ICCSA), Saint Petersburg, Russia July 1-4, 2019, Lecture Notes in Computer Science, Springer, Vol. 11620, Part 2, pp. 654-670, 2019, doi: 10.1007/978-3-030-24296-1_53.
2. M. Bougherara, N. Nedjah, D. Bennouar, R. Kemcha, L. M. Mourelle: *Application Mapping onto 3D NoCs Using Differential Evolution*. Proc. of the 20th International Conference on Computational Science and Its Applications (ICCSA), Cagliari, Italy July 1-4, 2020, Lecture Notes in Computer Science, Springer, Vol. 12251 (Part III), pp. 89-102, 2020, doi: 10.1007/978-3-030-58808-3_8.

Enfin, nous modélisons le problème de routage à l'aide des méta-heuristiques PSO et DE. Nous définissons également sa fonction objectif afin de l'intégrer dans le simulateur de réseau sur puce appelé Noxim [9]. Ce travail a gérer une publication:

1. M. Bougherara, N. Nedjah, D. Bennouar, L. M. Mourelle: *Routing in 3D NoCs using Genetic Algorithm and Particle Swarm Optimization*. Proc. of the 23rd International Conference on Computational Science and Its Applications (ICCSA), Athens, Grece, July 2-6, 2023, Lecture Notes in Computer Science, Springer, Vol. 14104, Part 1, pp. 1-13, 2023, doi:10.1007/978-3-031-37105-9_40.

1.3 Organisation

Le reste de cette thèse est organisée en sept chapitres. Le chapitre 2 présente le concept du SoC et du NoC ensuite, il explore la naissance du NoC 3D, incluant les problèmes de conception du réseaux sur puces qui sont l'objet de cette thèse.

Le chapitre 3 introduit un état de l'art des travaux de recherche concernant les problèmes de conceptions du NoC en 3D.

Le chapitre 4 introduit les concepts de base de l'optimisation, mono et multi objectifs, ainsi qu' une classification des algorithmes d'optimisation.

Dans le chapitre 5 nous présentons en détail les algorithmes d'optimisation exploitée dans notre thèse.

Le chapitre 6 présente les approches proposées pour les trois problèmes de conceptions. Une Modélisation est conçue ainsi que la définition des fonctions objectives utilisées pour chaque problèmes.

Le chapitre 7 est réservé à l'évaluation expérimentale pour chaque algorithme et pour chaque problème. Une comparaison de leurs performances respectives est présenté.

Le chapitre 8 permet de récapituler les principaux résultats et contributions de l'étude, de tirer des conclusions pertinentes et d'identifier des pistes de recherche futures.

Chapitre 2

Réseaux sur puce

Les systèmes embarqués sont présents en masse dans les appareils que nous utilisons quotidiennement: téléphones portables, ordinateurs, consoles de jeux, voiture, machine à laver, entre plusieurs autres. Tous ces appareils sont basés sur des SoCs et sont le résultat de l'évolution technologique, permettant l'intégration de plusieurs composants électroniques sur une seule puce. Le marché exige que les appareils soient produits de plus en plus vite. Pour répondre à cette exigence, une méthodologie de conception qui priorise la réutilisation des composants est nécessaire. Ainsi, pour la mise en œuvre de SoCs, et afin de réduire le temps de conception et faciliter la réutilisation, des composants IPs sont utilisés.

Par ailleurs, le problème de communication entre ces composants devient un facteur de conception critique lorsque le nombre de composants par SoC augmente. Une architecture de communication efficace et évolutive est donc nécessaire. Cependant, les solutions de communications actuelles, comme le bus partagé, trouvent leurs limites en termes de bande passante et d'extension à mesure que le nombre d'éléments communicants augmente. Ce type d'architecture de communication ne semblent pas s'adapter aux applications futures. C'est dans ce contexte qu'un nouveau paradigme d'interconnexion est apparu: les réseaux sur puce (NoCs), inspiré du modèle de réseau de communication informatique.

Dans la Section 2.1, nous commencerons par présenter les systèmes embarqués. Puis, nous aborderons l'évolution des types d'interconnexion dans les SoCs que nous verrons dans la Section 2.2. Les réseaux sur puce ainsi que leurs composants et leurs caractéristiques seront détaillés dans la Section 2.3. La Section 2.4 concernera les réseaux sur puce en trois dimensions. Dans la Section 2.5, nous présenterons les phases de conception des réseaux sur puce. Enfin, dans la dernière Section 2.6 nous terminons par une conclusion de ce chapitre.

2.1 Définition d'un système embarqué

Un système embarqué est un système de calcul, généralement à usage spécifique, conçu dans le même boîtier d'un circuit intégré (une puce en silicium) [10]. Un SoC est généralement composé d'un ou de plusieurs microprocesseurs, d'une mémoire et de dispositifs d'entrée et de sortie (E/S), ou de tout autre périphérique nécessaire à la réalisation de ses fonctions [11]. Tous ces composants échangent des informations entre eux à travers une structure d'interconnexion, de sorte que le SoC soit capable d'exécuter une application [12]. L'organisation d'un SoC générique est illustrée dans la Figure 2.1(a).

Pour relever le défi et répondre aux besoins des nouvelles applications, les progrès technologiques dans le domaine des circuits intégrés ont fait émerger deux tendances [13]:

1. La première, consiste à utiliser des architectures monoprocesseurs avec des performances améliorées: une fréquence de fonctionnement très élevée, un répertoire d'instructions riche, plusieurs niveaux hiérarchiques de mémoire, ainsi que l'utilisation de coprocesseurs.
2. La deuxième, quant à elle, consiste à utiliser des architectures multiprocesseurs qui sont massivement parallèles. Cela est rendu possible grâce à la croissance continue de la densité d'intégration.

En effet, il est plus facile, actuellement, d'atteindre des performances élevées en exploitant le parallélisme de plusieurs processeurs intégrés dans une même puce de silicium nommée (MPSoC – *Multi-Processor System on Chip*), qu'en augmentant la rapidité d'un seul processeur. L'organisation d'un MPSoC générique est illustrée dans la Figure 2.1(b).

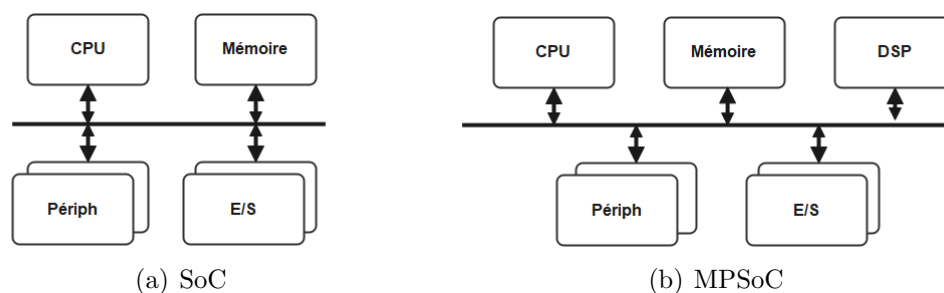


FIGURE 2.1 – Organisation d'un SoC et MPSoC

La conception du SoC et MPSoC est généralement basée sur l'utilisation de blocs de propriété intellectuelle (IP). Un IP fait référence à des composants ou modules préconçus et pré-vérifiés qui sont intégrés dans la conception d'un SoC. Ces blocs IP sont généralement développés par des sociétés tierces spécialisées dans des fonctionnalités spécifiques telles que les processeurs, les contrôleurs de mémoire, les protocoles d'interface ou les accélérateurs matériels spécialisés.

L'idée centrale dans l'utilisation des IPs est la réutilisation: un bloc développé pour un projet peut être utilisé dans des projets futurs avec un minimum d'adaptation. Ainsi, il y'a une diminution du temps total de développement et du coût du projet de système.

2.2 Évolution des interconnexions dans les SoCs

Afin que les architectures d'interconnexions s'adaptent à l'évolution des systèmes qui les intègrent, ces architectures doivent répondre aux exigences de flexibilité, extensibilité et simplicité [14]:

- a) La flexibilité des structures est une condition nécessaire pour permettre l'interconnexion de plusieurs blocs d'IPs de nature hétérogène (qui peuvent être développés par différentes équipes, avec différents outils dans différents contextes).
- b) L'extensibilité désigne la capacité à augmenter le nombre de blocs inter-connectés à tout moment, tout en conservant les mêmes performances des communications.
- c) La simplicité de mise en œuvre est caractérisé par la réduction du nombre de ressources nécessaires pour sa réalisation.

Nous citons dans cette section les quatre évolutions du système d'interconnexion utilisé dans les systèmes sur une seule puce: point à point, par bus, par bus hiérarchique, par réseau.

2.2.1 Connexion point à point

C'est la connexion la plus simple pour connecter des IP. Les blocs fonctionnels sont reliés entre eux directement sans définir un protocole de gestion de communication, comme illustré dans la Figure 2.2 [15]. Chaque connexion est dédiée et spécifique, ce qui permet d'avoir des connexions à haut débit et un parallélisme contrôlé [16]. Cependant, ce type d'interconnexion est limité, car certaines IP peuvent avoir besoin de se communiquer avec d'autre IP en fonction de l'application. Plus important, l'ensemble du SoC peut devenir inutilisable si un bloc fonctionnel est défectueux. De plus, la connexion n'est pas flexible, donc difficilement réutilisable [15].

2.2.2 Connexion par bus

Contrairement à la connexion point à point, la connexion par bus permet de connecter tous les blocs fonctionnels à un seul média de communication qui est le bus, comme illustré dans la Figure 2.3 [17]. Ce type de connexion est plus flexible par rapport au type précédent, et mieux réutilisable. Par contre l'inconvénient majeur de ce type d'interconnexion est de

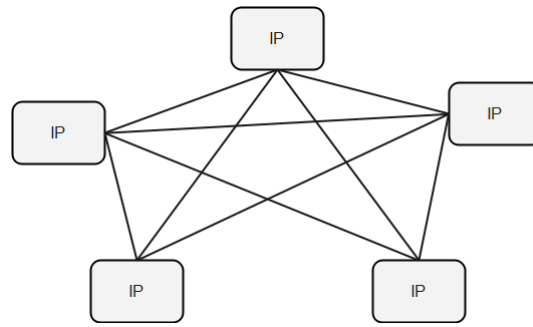


FIGURE 2.2 – Connexion point à point

ne pouvoir réaliser qu'une seule communication à la fois. Les tâches peuvent se dérouler en parallèles mais pas les communications [18]. De plus, lorsque le nombre de communications augmente ou que les contraintes de bande passante de plusieurs communications deviennent trop importantes, cela peut entraîner un goulot d'étranglement [19].

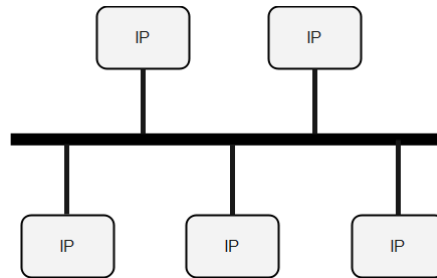


FIGURE 2.3 – Connexion par bus

2.2.3 Connexion par bus hiérarchique

Pour remédier aux problèmes rencontrés dans la structure d'interconnexion par bus, une nouvelle structure d'interconnexion, la structure de bus hiérarchique, a été proposée. Elle consiste à connecter plusieurs segments de bus deux à deux par l'intermédiaire d'un pont [20]. Le point fort de cette structure est que les IPs sont réparties dans plusieurs bus différents et non connectés à un seul bus comme dans la structure précédente. Cela permet d'équilibrer la charge des bus et de diminuer leurs longueurs. Ceci augmente la performance du bus d'une part, et permet des communications simultanées qui ne se trouvent pas dans les mêmes segments, comme illustré dans la Figure 2.4 [21]. Néanmoins, cette solution présente un point de faiblesse au niveau des ponts. En effet, c'est là que peut se créer un goulot d'étranglement lorsque le nombre d'éléments communicants inter segments augmente [19].

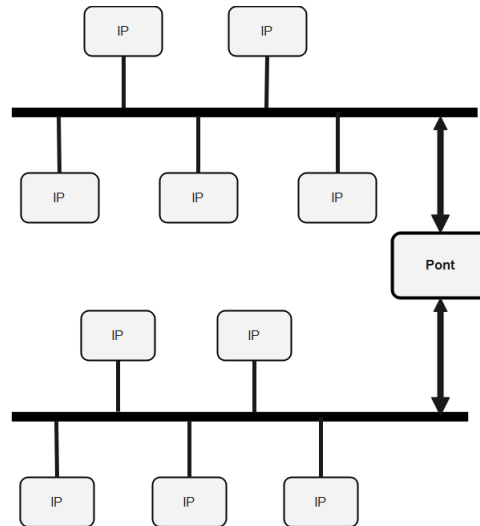


FIGURE 2.4 – Connexion par Bus hiérarchique

2.2.4 Réseau sur puce

Afin de s'affranchir des problèmes liés aux interconnexions mentionnées précédemment, à savoir le manque de flexibilité et l'impossibilité de passage à l'échelle, le concept des réseaux sur puce est apparu [22]. Ce nouveau paradigme d'interconnexion dérive des réseaux informatiques traditionnels, et qui sont adaptée pour utilisation dans des systèmes intégrés. Comme illustré dans la Figure 2.5 [3].

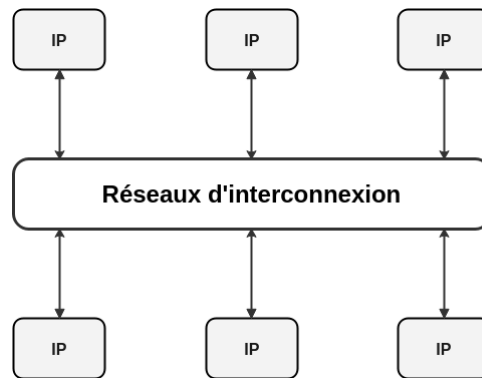


FIGURE 2.5 – Connexion par Réseaux d'interconnexion

Les principaux avantages de l'utilisation des réseaux sur puces en comparaison avec les autres infrastructures de communications sont résumés dans la Table 2.1 qui représente une comparaison entre ces différentes structures d'interconnexions [17]. Le symbole ++ est considérée comme très bonne, + comme bonne, – comme mauvaise et –– comme très mauvaise.

TABLE 2.1 – Comparaison entre les structures d’interconnexion dans les SoCs [17]

Connexion	Parallélisme	Consommation	Scalabilité	Réutilisation	Bande passante
Point à point	++	+	--	--	++
Bus partagé	--	--	--	++	--
Bus hiérarchique	+	-	-	++	+
NoC	++	++	++	++	++

2.3 Terminologie des réseaux sur puce

Les réseaux sur puce sont des infrastructures de communication utilisées dans les systèmes sur puce pour connecter et interagir avec les différents composants intégrés sur une puce. Par la suite, nous donnons la terminologie utilisée.

2.3.1 Architecture du réseaux sur puce

En raison de sa similitude architecturale avec un réseau de données informatiques, il a été considéré que l’architecture d’un NoC peut être résumé en termes de modèle de référence (OSI – *Open Systems Interconnection*) [23].

Le modèle OSI ne définit pas exactement comment un système doit être construit, mais agit comme guide conceptuel. Plus précisément, en permettant à chaque couche d’exécuter sa tâche indépendamment des autres couches et fournit des services à la couche supérieure tout en obtenant les services des couches inférieures. Les couches se communiquent entre elles à travers des interfaces standards. Le modèle OSI classique comporte sept couches: physique; liaison de données; réseau; transport; session; présentation et application. Une couche peut être considérée comme une combinaison de composants logiciels ou matériels effectuant quelques fonctionnalités.

Le modèle du NoC est comparé par rapport au modèle de référence OSI. Il comporte quatre couches qui sont placées sur les sept couches du modèle OSI [24]. La Figure 2.6 illustre la relation entre les couches NoC et le modèle de référence OSI.

- a) La couche système, qui comprend les services des deux couches OSI supérieures et représente les exigences des cœurs d’application les IPs.
- b) La couche adaptateur réseau, qui est chargée de contrôler les services de communication de bout en bout, et modélise les services des couches session et transport de l’OSI.
- c) La couche réseau, qui fournit les mêmes services que la couche réseau dans le modèle OSI. Ces services sont implémentés par le routeur matériel.
- d) La couche liaison, qui comprend les liaisons logiques et physiques qui interconnectent les composants NoC.

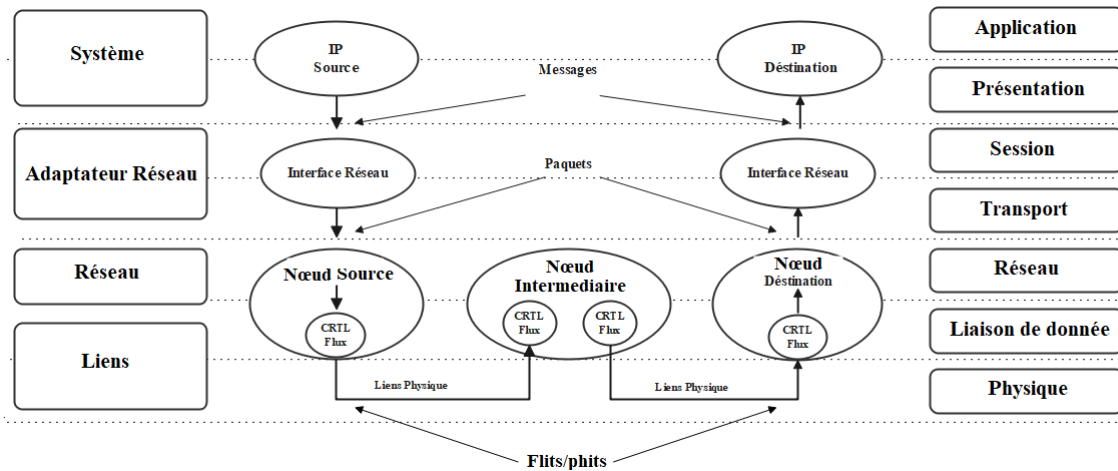


FIGURE 2.6 – Les éléments du NoC et couches réseau

Il existe quatre principaux types de composants dans un NoC [25] [26] [27] : les ressources également appelées les cœurs ou IP, les adaptateurs réseau abrégés NI (*Network Interface*), les routeurs et les liens physiques. La Figure 2.7 montre ces quatre composants.

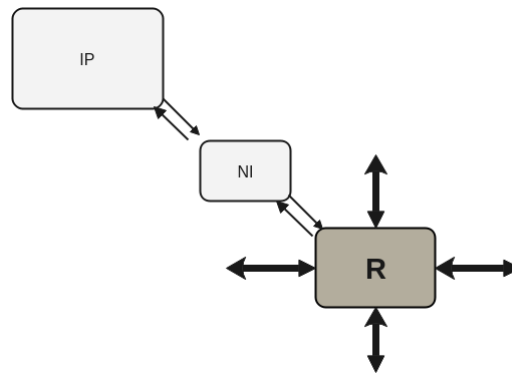


FIGURE 2.7 – Les trois types de composants d'un Réseau sur puce

Les ressources (IPs) peuvent être divers composants tels que : un processeur d'usage général, un processeur de signal numérique (DSP–*Digital Signal Processor*), qui est un microprocesseur optimisé pour exécuter rapidement des applications de traitement numérique du signal telles que le filtrage et l'extraction de signaux, une mémoire, un composant matériel d'application spécifique, un contrôleur d'E/S et un contrôleur graphique. Les IPs sont connectées aux routeurs de réseau par l'intermédiaire d'une interface de réseau.

L'adaptateur réseaux (NI–*Network Interface*) assurent l'interface entre le protocole du NoC et celui des ressources IPs [28]. Leur rôle est de séparer la fonction calcul de celle de la fonction communication [21]. Ceci augmente la flexibilité du NoC et permet la réutilisation des blocs IPs comme illustré dans la Figure 2.8. Un adaptateur réseau est constitué de deux parties : l'interface vers les routeurs et l'adaptateur vers les IPs [21].

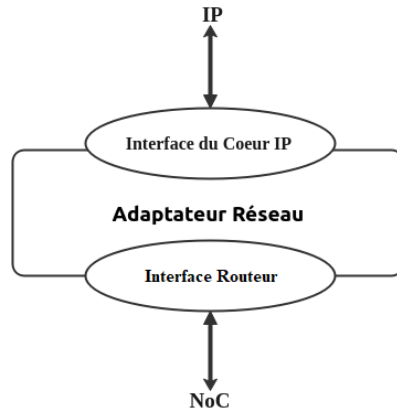


FIGURE 2.8 – Les Composants d'un Adaptateur réseau

Les routeurs permettent d'acheminer les données d'une source vers une destination dans la topologie, selon un protocole de routage choisi. Il assure la fonction de routage ainsi que celle d'arbitrage entre différentes demandes de port. Un exemple d'un routeur du NoC est illustré dans la Figure 2.9.

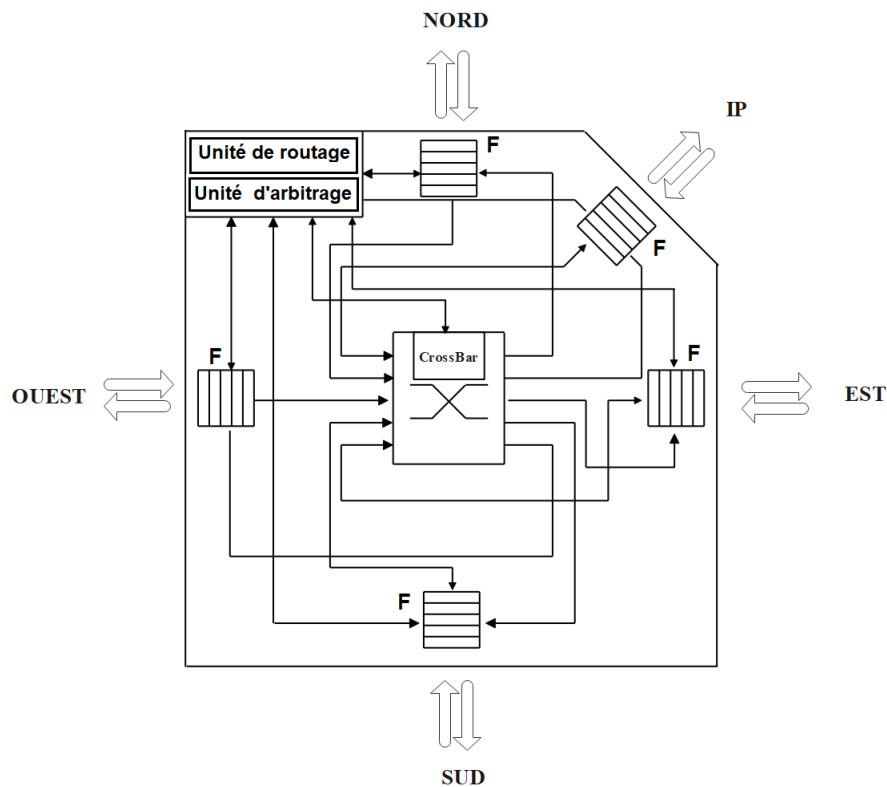


FIGURE 2.9 – Exemple de composants d'un Routeur

Il est nécessaire que la conception du routeur en NoC soit la plus simple possible, car le coût de mise en œuvre augmente de façon exponentielle avec l'augmentation de la complexité de la conception du routeur. Par conséquent, un routeur est mis en œuvre au moyen de [29]:

- a) Files d'attente (F), utilisées pour le stockage temporaire des données, celles-ci sont connectées aux différents ports de sorties par un commutateur.

- b) Un commutateur (*CrossBar*), c'est une unité de multiplexage permettant de connecter les files d'entrée aux ports de sortie pour acheminer le paquet du port d'entrée vers les portes de sortie. L'entrée de sélection est définie par l'unité de routage.
- c) Une unité de routage, qui calcule le chemin et alloue les ressources nécessaires.
- d) Une unité d'arbitrage, utilisé pour sélectionner un paquet à partir d'un port entrant vers un port sortant, elle joue le rôle d'arbitrage pour la priorité de sortie.
- e) Cinq ports de communication permettent de connecter le routeur à d'autres composants du réseau.

Les Liens physiques sont des connexions logiques établies entre deux ou plusieurs éléments communicants. Ils facilitent les connexions entre différentes entités, notamment entre les ressources IPs et les interfaces réseau NI, entre les NI et les routeurs, ainsi qu'entre les routeurs eux-mêmes. Par ailleurs, ils offrent la bande passante transportant l'information entre les éléments communicants. Il existe trois types de liens qui sont définis par le sens des transmissions. Ils peuvent être unidirectionnels en entrée, unidirectionnels en sortie ou bidirectionnels (entrée/sortie).

2.3.2 Caractéristiques des réseaux sur puce

Un réseau sur puce est défini essentiellement par sa topologie, les éléments de communication, les mécanismes de commutation, la technique de contrôle de flux exploitée et l'algorithme de routage implémenté.

La topologie spécifie l'organisation physique du réseau et définit la disposition structurale de ses éléments, c'est à dire qu'elle définit comment les ressources sont reliées entre elles. Il existe plusieurs métriques pour comparer les différentes topologies entre elles [14]. Les plus utilisées sont:

- a) Le diamètre: c'est le nombre maximal de liens qui relient deux ressources quelconques du réseau (en considérant les plus courts chemins).
- b) La distance moyenne: c'est le nombre moyen de liens entre deux ressources.
- c) La connectivité: c'est le nombre de voisins directs d'un nœud dans le réseau.
- d) La largeur de bisection: c'est le nombre minimal de liens qu'il faut couper pour séparer le réseau en deux parties égales (plus au moins un nœud).

En outre, il existe deux types de topologies, les topologies directes et les topologies indirectes [26]:

les topologies directes: dans ce type de topologie, chaque routeur est relié à un processeur formant un seul élément du système appelé nœud ou tuile (*tile*). Chaque tuile est connectée directement à un nombre fixe de tuiles voisines. Un message entre deux nœuds passe par un ou plusieurs routeurs intermédiaires. La communication est basée sur l'algorithme

de routage mis en œuvre par les routeurs. Par conséquent, Les avantages de ce type de topologies sont leur structure réutilisable et leur évolutivité. Les topologies directes les plus connues sont la topologie en anneau et la topologie en maille [30].

La Topologie en anneau Relie les nœuds d'un réseau entre-eux afin de former une boucle. Chaque nœud entre la source et la destination sert alors d'interconnexion, comme illustré dans la Figure 2.10(a). C'est une structure facilement intégrable sur silicium. Cependant elle n'est pas extensible. La topologie en anneau comporte un inconvénient majeur car pour un réseau de grande taille le message doit traverser un nombre important de routeurs, engendrant inévitablement une limite de la bande passante. Cependant, il existe une alternative proposée à la limitation de la topologie en anneau, qui connue comme topologie en anneau avec cordes, illustré dans la Figure 2.10(b). Elle permet de réduire la latence en offrant la garantie de ne traverser, au maximum, que deux routeurs pour n'importe quelle communication au sein du réseau.

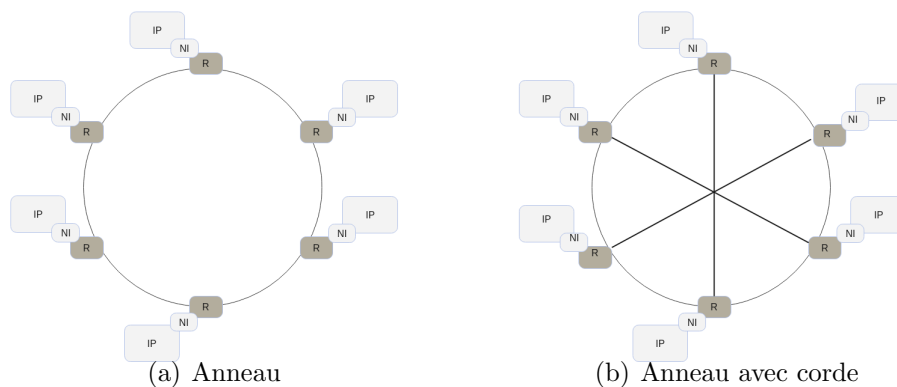


FIGURE 2.10 – Topologie en anneau

La Topologie maillée en 2D (*2D mesh*) est une topologie matricielle, l'une des plus simples et des plus étudiées, étant donné qu'elle est facile à mettre en œuvre sur silicium. Elle exige que tous les nœuds soient connectés à leurs voisins, ce qui permet une certaine fiabilité car, en cas de panne d'un lien ou d'un nœud, d'autres chemins sont possibles. Elle est représentée dans la Figure 2.11(a). La topologie maillée en tore est dérivée de la structure 2D. Elle possède la particularité d'un repliement des bords extérieurs sur eux-mêmes. Cette topologie offre une meilleure utilisation des ressources réseaux, puisque pour un même nombre des nœuds, il y a plus de liens. Cela réduit le diamètre et offre une bande passante légèrement élevée par rapport à une structure 2D. Cependant, cette structure est moins régulière et plus complexe à mettre en œuvre.

Les topologies indirectes: La structure de ce type de topologie est sous forme arborescente. Elle se base sur une hiérarchie de relation pères-fils, ou chaque feuille représente un IP. Elle a pour intérêt d'être scalable et d'offrir une latence qui peut être plus faible que la topologie 2D. Dans ce type de topologie, seuls les nœuds feuilles de l'arbre sont des nœuds de réseau qui injectent/reçoivent des paquets de données. Le reste des nœuds (y compris

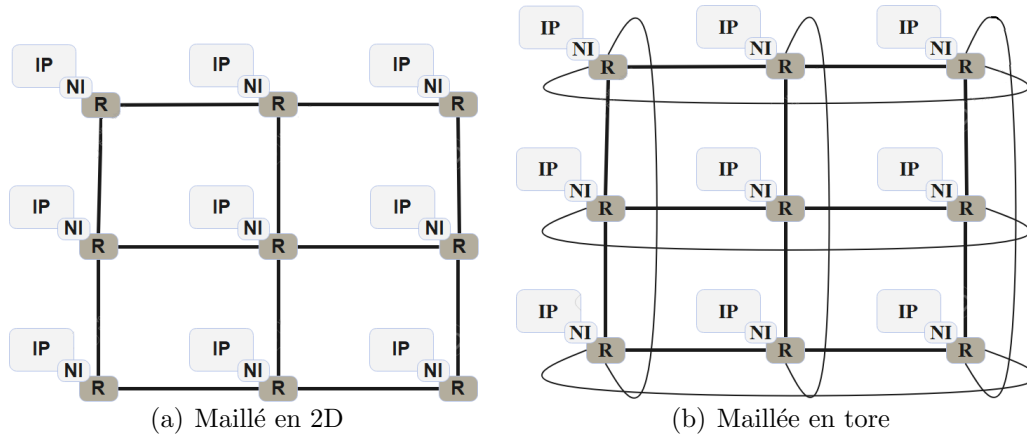


FIGURE 2.11 – Topologie en 2D

la racine) ne fonctionnent que comme commutateurs (routeurs). Un inconvénient majeur des topologies arborescentes est que, pour deux nœuds, il existe une seule route entre eux. Cela signifie qu'aucune tolérance aux pannes n'est admise. Pour ces raisons, les topologies d'arbre élargie (*fat-tree*) ont été proposées. Figure 2.12(a) illustre une topologie *fat-tree*, le lien direct d'un nœud (vers le nœud parent) a deux fois plus de bande passante que sa liaison descendante (vers le nœud enfant). Le nœud racine d'une topologie arborescente est sans doute un goulot d'étranglement dans le réseau. Il existe une autre topologie hiérarchique appelée topologie papillon qui dérive de la précédente. Celle-ci est illustré par la Figure 2.12(b). La topologie papillon a pour but de réduire la latence, mais elle a l'inconvénient d'être plus coûteuse en terme de surface silicium en raison de sa longueur plus élevée.

L'avantage des topologies indirectes est qu'elles peuvent être plus facilement adaptées aux besoins spécifiques de l'application [31]. Elle permet ainsi de répondre au mieux aux exigences souhaitées. Cependant, cela dépend de l'application cible qui peut nécessiter des contraintes de surface. D'autre part, l'un des principaux problèmes dont souffrent les topologies indirectes, consiste en la difficulté à la mise en œuvre à cause de la complexité de ses interconnexions. De plus, un mauvais placement des différentes IPs de l'application sur le réseau engendre souvent un goulot d'étranglement au niveau des nœuds de routage intermédiaires. Par conséquent, les topologies indirectes diminuent considérablement les performances du NoC.

Topologies irrégulières: Une autre classification possible pour les topologies de réseau NoC est liée à la régularité des connexions entre les routeurs. Dans les réseaux réguliers, tous les routeurs sont identiques en termes de nombre de ports. Les topologies irrégulières sont créées en intégrant différentes formes et structures régulières de manière variée

Les topologies irrégulières sont basées sur l'intégration de différentes topologies, de diverses formes, généralement des structures régulières [31]. Cela aide à réduire la distance entre les nœuds. En outre, les topologies irrégulières évoluent généralement de manière

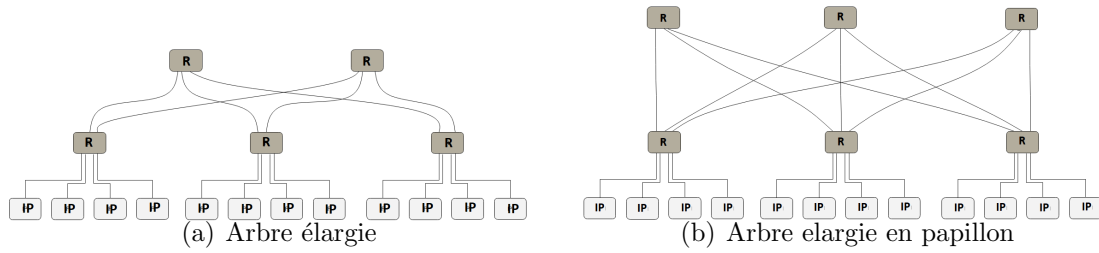


FIGURE 2.12 – Topologie en arbre

non linéaire avec la surface et la puissance. Elles sont généralement basées sur le concept de *clustering* et adaptées à des applications spécifiques. La Figure 2.13 illustre certaines topologies irrégulières, telles que le maillage réduit optimisé dans les Figure 2.13(a) et 2.13(b) et l'hybride en *cluster* maille avec anneau, dans la Figure 2.13(c).

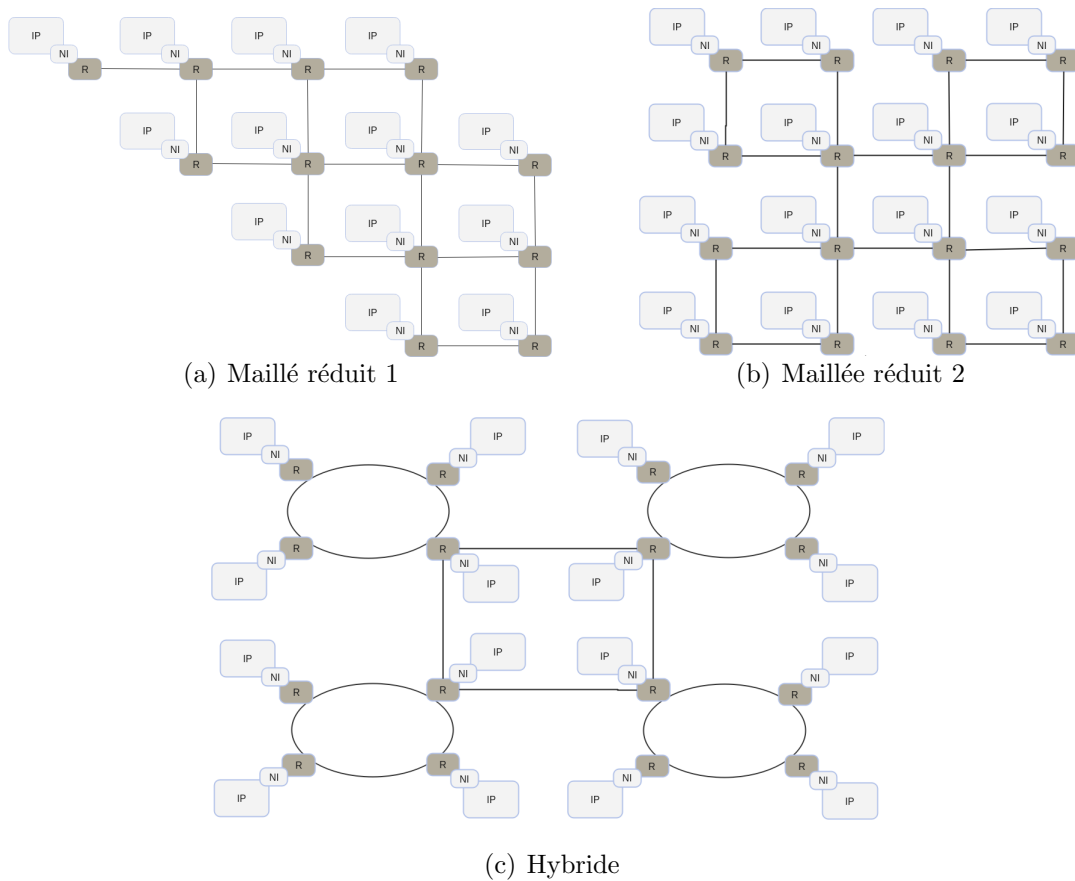


FIGURE 2.13 – Topologie irrégulière

Les éléments de communication spécifiques aux réseaux sur puce, tels que message, paquet, flit et phit [32] sont couramment utilisés. La Figure 2.14 montre la structure d'un message. Les éléments de communication sont définis comme suit:

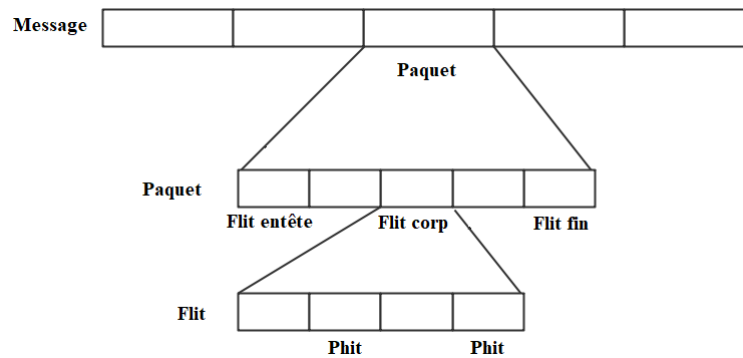


FIGURE 2.14 – Structure d'un message

- A) Message: Il représente les données qui doivent être transférées à partir de la source vers la destination. Le message est défini dans la couche application. Sa taille peut être fixe ou variable et divisée en plusieurs paquets.
- B) Paquet: Il représente une unité d'un ensemble de données. Il peut être transféré dans le réseau indépendamment des autres paquets du même message. Un paquet comporte principalement deux parties, la première partie sert à contenir des informations de contrôle et de routage et est appelée entête *header*. L'autre partie, appelée charge-utile (*payload*) contient les données. Parfois, les paquets contiennent également une queue (*tail*) indiquant la fin du paquet. Un paquet peut être de taille fixe ou variable. Il peut être décomposé en unités de données plus petites, appelées flits.
- C) Flit: Unités de contrôle de flux, ou Flit (*Flow Control Units*). Un flit est la plus petite quantité d'information (le plus petit morceau du message) pour lequel on peut définir un contrôle de flux. Il se compose d'un nombre constant de bits, et peut être formé d'un ou de plusieurs *phits*.
- D) Phit: Unités de transfert physique, ou (Phit – *Physical Transfer Units*): En fonction de la taille du flit et de la largeur des liens, plusieurs cycles peuvent être nécessaires pour transmettre un flit. Un phit représente donc la quantité d'information que l'on peut transmettre en un cycle.

Un mécanisme de commutation définit la manière dont les données sont transmises sur le réseau sur puce de la source à la destination. Il existe deux principaux mécanismes de commutation [33]: la commutation de circuits et la commutation de paquets.

La Commutation de circuits consiste à établir un lien physique unique entre deux unités de traitement durant toute la durée de la communication. Ceci est effectué en allouant un chemin (ensemble de nœuds de routage et de liens de communication) dédié à cette opération. Ce chemin est appelé circuit. Le démarrage et l'arrêt de la connexion sont contrôlés par des informations distinctes des données de communication [34]. Ce mécanisme garantit une large bande passante entre les ressources et améliore les performances du système pour les transmissions de données volumineuses. Les systèmes de commutation de

circuits sont adaptés aux communications en temps réel et sont parfois appelés réseaux de connexions orientées. Cependant, ce mode de commutation nécessite des ressources constantes tout au long du transfert, ce qui peut être contraignant.

Dans la Commutation de paquets, l'information à transmettre est découpée en paquets. Ce dernier est envoyé sans qu'il y ait une connexion établie au préalable. Le paquet est constitué de deux parties. La partie en-tête, qui contient les informations indispensables pour acheminer et reconstituer le message, et l'autre partie appelée données, elle contient l'information à transmettre. Dans ce cas, lorsque le paquet est envoyé sur le réseau, le routeur analyse l'en-tête de ce dernier afin de l'acheminer à sa destination. Ce principe de communication diminue la latence et augmente les performances du système. Dans ce mécanisme de commutation, il existe trois modes définissant comment les paquets transitent dans le réseau, et qui sont [35] [26] [36]:

- 1) Stockage et Retransmission (SAF – *Store-and-forward*): Ce mode de commutation élimine le risque de blocage des paquets dans le réseau. Cependant, un routeur doit attendre de recevoir tous les flits d'un paquet avant de le transférer au routeur suivant, ce qui nécessite des buffers de taille équivalente à celle du paquet, entraînant un coût élevé. De plus, ce mode de commutation entraîne une latence élevée, calculée en multipliant le temps de transfert d'un paquet entre deux routeurs par le nombre de routeurs traversés par le paquet. Malgré cela, ce mode de commutation est avantageux pour les messages courts et répétitifs, éliminant les risques d'interblocage, et en cas d'erreur lors de l'envoi des paquets, seul le paquet erroné sera renvoyé.
- 2) Chemin Virtuel Direct (VCT – *Virtual cut-through*): Il permet le transfert de paquets entre routeurs sans attendre la réception complète du paquet. Un routeur peut commencer à envoyer un paquet au routeur suivant avant de l'avoir entièrement reçu, à condition que le routeur suivant puisse stocker le paquet dans sa totalité. Ce mode de commutation vise à réduire la latence et à éliminer les risques d'interblocages. Cependant, il nécessite la même taille d'espace de stockage que le mode SAF.
- 3) Trou de ver (WH – *Wormhole*): Le mode wormhole est le plus populaire et le plus utilisé pour les routeurs dans les réseaux sur puce. Il permet de réduire la latence et l'espace de stockage nécessaire dans les routeurs en réservant un chemin dès que le premier flit (flit d'en-tête) sort du port de sortie du routeur. Cependant, il y a un risque d'interblocage, où les flits d'un même paquet peuvent être bloqués sur plusieurs routeurs. Ce risque est généralement évité en utilisant d'autres mécanismes tels que les canaux virtuels et les algorithmes de routage. En réduisant la capacité de mémorisation sur les ports entrants à la taille d'un flit, les besoins en mémoires dans les routeurs sont considérablement réduits, ce qui permet de réduire la surface en silicium.

Pour comparer les différents modes de commutation, le tableau 2.2 présente une comparaison entre ces différents modes.

TABLE 2.2 – Comparaison entre les différents modes de commutation

Mode	Store-and-forward	Virtual cut through	Wormhole
Taille mémoire du routeur	Un paquet complet	Un paquet complet	Fraction d'un paquet
Latence	Elevée	Faible	Faible
Inter blocage	Non	Non	Possible

Le contrôle de flux est un mécanisme qui permet de réguler le trafic et d'éviter la surcharge du réseau. Un émetteur ne peut pas envoyer plus de données que le récepteur ne peut en supporter. Ainsi, il a pour but de garantir un transport de paquets depuis l'IP source jusqu'à l'IP destination. Ceci est réalisé en utilisant des signaux de requêtes et d'acquittement échangés entre les différents routeurs. Il existe plusieurs techniques de contrôles de flux [37] [31]:

- a) Poignée de Main (*Handshake*): Cette technique consiste à acquitter chaque donnée avant d'envoyer la prochaine. Un émetteur reste en attente tant qu'il n'a pas reçu l'acquittement de l'ancienne donnée envoyée. Ceci augmente la latence du système et dégrade ses performances [37]. Cependant, elle est simple à mettre en place mais elle nécessite au moins deux cycles d'horloges pour effectuer un transfert.
- b) Basé sur le crédit (*Credit-based*): Elle consiste à utiliser des crédits qui vont indiquer à l'émetteur le nombre de places disponibles dans la mémoire du destinataire. Au départ, l'émetteur dispose de la totalité du crédit et au fur et à mesure qu'il envoie les données, le crédit est décrémenté jusqu'à ce qu'il devienne nul. Lorsque le récepteur consomme les données qui lui ont été envoyées, il informe l'émetteur du nombre de places nouvellement libérées [31]. Cette technique est simple à implémenter et améliore les performances du système en termes de bande passante et de débits car les transferts de données ne nécessitent qu'un seul cycle d'horloge.
- c) Arrêtez et partez (*Stop and Go*): Consiste à utiliser deux seuils pour chaque buffer (*Stop*) et (*Go*) [37]. Lorsque l'espace occupé dans le buffer récepteur atteint le seuil d'arrêt (*Stop*), un signal est envoyé à l'émetteur d'arrêter la transmission (*Stop*) tout en prenant en compte ce qui reste encore de la taille de la mémoire du buffer si elle est suffisante pour les flits qui sont toujours transmis par l'émetteur. Un autre signal est envoyé pour réactiver le flux de flits (*Go*), lorsque l'occupation du buffer diminue jusqu'à devenir inférieure ou égale au deuxième seuil.

L'algorithme de routage détermine la suite de nœuds que doivent emprunter les données transmises par une source afin d'atteindre leur destination. C'est un point très important lors de la conception des réseaux sur puces. Un bon algorithme de routage doit offrir les trois caractéristiques suivantes: une faible latence, un débit élevé et une facilité de mise en œuvre. Les algorithmes de routage utilisés dans les réseaux sur puces, peuvent être classés selon plusieurs critères qui sont détaillés dans ce qui suit [32] [38] [39] illustré dans la Figure 2.15:

- A) Selon l'endroit où les décisions de routage sont prises: Un algorithme de routage est dit source si le chemin emprunté par les données est entièrement défini par l'émetteur. Les informations de routage sont incluses par ce dernier dans le flit d'entête. Ainsi, les routeurs intermédiaires n'ont qu'à consulter le premier flit pour savoir à quel nœud il faudra faire suivre le paquet. Dans le cas contraire, il s'agit d'un routage distribué. Dans ce dernier, la décision peut être prise soit en exécutant une fonction qui tient compte de l'adresse cible et éventuellement de certains paramètres, soit en consultant une table de routage prédéfinie et implantée au niveau du nœud de routage.
- B) Selon la manière de prendre les décisions de routage: Si le chemin emprunté par les données reste inchangé pour chaque paire émetteur/récepteur, le routage est dit déterministe ou statique. Par contre, si l'algorithme de routage tient compte d'autres facteurs (la charge du réseau, la disponibilité des liens, etc.), alors le routage est dit adaptatif ou dynamique. Le routage statique est facile à réaliser en termes de logique et d'interaction entre les routeurs. Il est plus adapté lorsque les échanges de données sont prévisibles et stables. Dans le cas contraire, le routage adaptatif est préférable même si son implémentation est plus coûteuse. Les algorithmes de routage adaptatif peuvent être divisés en deux classes: les algorithmes adaptatifs globaux et les algorithmes adaptatifs partiels:
- Un algorithme adaptatif partiel effectue le choix du prochain nœud parmi un nombre restreint de nœuds possibles.
 - Un algorithme global, route le paquet dans n'importe quelle direction.
 - Un algorithme semi adaptatif, constitue un compromis entre le faible coût et la performance des routages statique et dynamique. Dans ce dernier, tant que les liens de communication sont disponibles, les décisions prises sont identiques à celles du routage déterministe. Par contre, dès que le réseau devient congestionné et que le lien de communication cible est occupé, le routage bascule en adaptatif. Ainsi, au lieu de stocker le paquet dans des buffers en attendant que le lien ciblé se libère, il est routé sur un autre lien disponible.
- C) Selon la longueur du chemin: Un algorithme de routage est dit minimal si les décisions sont prises de telle façon à ce que le chemin parcouru de l'émetteur au récepteur soit le plus court possible. Ce type de routage a l'avantage de réduire la latence et d'éviter les interblocages. Il est cependant moins flexible lorsque certains composants du réseau sont congestionnés. À l'inverse, si le chemin pris par les données n'est pas forcément le plus court, alors le routage est dit non minimal. Ce dernier possède l'avantage d'offrir plus de flexibilité en termes de chemins possibles, mais la latence est plus élevée et le risque du phénomène livelock est grand.

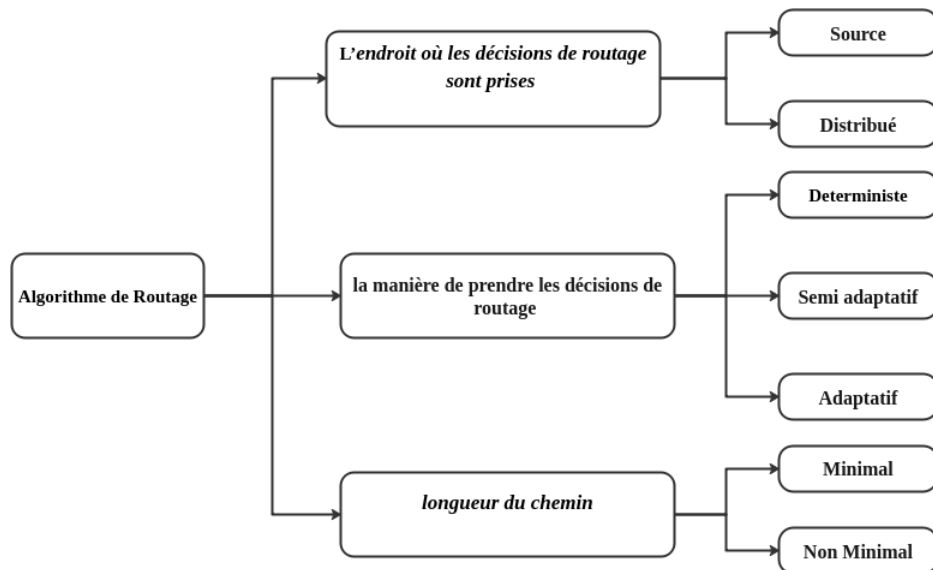


FIGURE 2.15 – Classification des algorithmes de routage [39]

Par ailleurs, il existe certaines situations de conflits liés aux algorithmes de routage et qui empêchent le transfert des paquets vers leurs destinations. Ce genre de phénomènes arrive dans le cas d'un réseau sur puce à commutation de paquets de type wormhole. Les autres modes ne sont pas exposés à ces situations [37].

- a) Inter blocage statique (*deadlock*): L'interblocage statique arrive lorsqu'un paquet ou un certain nombre de paquets sont bloqués en attente les uns des autres d'une manière cyclique. Dans ce cas, aucun paquet ne peut progresser. Certaines solutions ont été proposées pour éviter ce genre de situations. Le principe consiste à bannir certains chemins des paquets susceptibles de former des cycles éliminant aussi une condition nécessaire de l'interblocage statique.
- b) Inter blocage dynamique (*livelock*): Ce type de blocage se produit avec un algorithme de routage adaptatif. Un paquet n'arrive jamais à sa destination tout en continuant à circuler sur le réseau. Ceci revient à la non disponibilité des liens de communication parce qu'ils sont occupés par d'autres paquets.
- c) La famine (*starvation*): La situation de famine se produit lorsqu'un paquet n'obtient jamais les ressources qu'il demande, car ces ressources sont toujours utilisées par d'autres paquets. Ce type de blocage est la conséquence d'une mauvaise gestion de l'attribution des ressources aux paquets en conflit. Ce problème peut être facilement résolu par un bon algorithme d'arbitrage comme l'algorithme de *Round Robin*.

Il existe dans la littérature, de nombreux algorithmes de routage. Nous en aborderons quelques uns dans le chapitre 3.

2.3.3 Mesure des performances d'un réseaux sur puce

Comme tout réseau, les NoC possèdent des métriques de performances. Celles-ci sont fortement liées à l'architecture du réseau NoC lui-même. Il est primordial d'avoir une bonne compréhension de ces métriques et de la façon de les évaluer, afin d'être en mesure d'optimiser l'architecture du réseau. Il existe de nombreux critères de performances, nous en citons la latence, le débit, la consommation d'énergie, et la superficie. [29] [16]:

La latence est un critère essentiel pour évaluer les performances des NoCs. Elle représente le délai moyen nécessaire pour transférer une unité de données de la source vers la destination, mesuré en nombre de cycles. La latence d'un paquet spécifique est le nombre de cycles depuis l'injection de l'en-tête flit par la source jusqu'à la réception du dernier flit par la destination. La valeur de la latence dépend de plusieurs facteurs, tels que le trafic réseau (congestion), la politique de routage des paquets (statique ou dynamique), l'arbitrage d'accès aux ports des routeurs (gestion de priorité), l'utilisation de canaux virtuels, le débit local disponible, la position relative de l'émetteur et du récepteur, ainsi que la profondeur des mémoires tampons des ports des routeurs.

Le débit est une caractéristique du réseau sur puce. C'est le trafic maximum que vous pouvez accepter dans le réseau, qui représente la quantité maximale d'informations transitant dans le réseau par unité de temps. Cependant, cette valeur ne représente pas le fonctionnement réel du réseau par conflits de routage, les éléments du réseau ne peuvent jamais être utilisés à leur pleine capacité. D'où le débit de données réel ou bande passante utile qui indique le taux maximum de diffusion des données une fois que le message est dans le réseau. Son unité est le bit par seconde (*bps*), généralement le paquet entier est mesuré.

L'énergie totale consommée dans un NoC se répartie entre celle consommée pour l'exécution de l'application dédiée au NoC (l'exécution des tâches dans les IPs) et celle nécessaire à la communication. Celle-ci se décompose à son tour en énergie consommée sur les routeurs et en énergie consommée pour envoyer des données dans les liens entre les routeurs.

Le coût global d'un circuit dépend fortement de sa superficie. Une plus grande superficie entraîne des coûts élevés liés aux matériaux, à la complexité de conception et à la main-d'œuvre. Afin d'optimiser les coûts de production et de le rendre économiquement viable, la réduction de la superficie du circuit devient un objectif essentiel.

2.4 Réseaux sur puce en 3D

Même si les réseaux sur puce basés sur la 2D englobent l'intégration d'un grand nombre de IPs et offrent une meilleure évolutivité par rapport aux architectures basées sur le bus, ils ne sont toujours pas considérés comme la solution ultime pour la communication dans

les futurs systèmes sur puces à très grande échelle [5]. Lorsque la taille du réseau augmente, la distance entre les IPs augmente, ce qui entraîne à son tour une augmentation des délais de transmission, de consommation d'énergie et de la taille globale de la puce, ce qui a un impact significatif sur les performances du système implémenté.

L'apparition de la technologie des circuits intégrés à 3 dimensions (3D IC), qui est définie par de multiples couches en silicium empilées ensemble, qui se communiquent à travers des liens d'interconnexions verticaux que l'on nomme TSV (*Through Silicon Via*), a encouragé l'exploration des NoCs en utilisant cette nouvelle technologie pour en exploiter les avantages par rapport aux circuits à 2 dimensions. Liu et al [8] [40] ont proposé pour la première fois le concept de NoC 3D, dont le cœur est un NoC 2D fusionné avec les circuits intégrés à 3 dimensions formant le NoC 3D illustrée dans la Figure 2.16, dans la Figure 2.17 illustre un exemple de réseaux sur puce 3D.

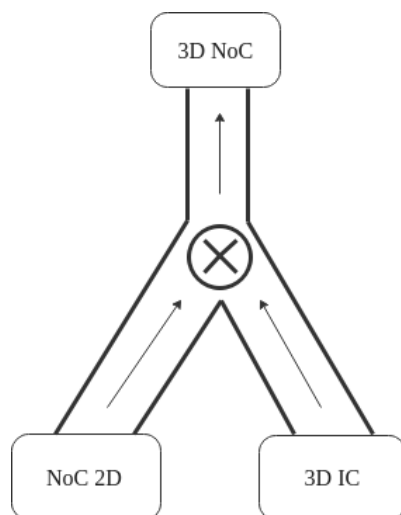


FIGURE 2.16 – L'idée de NoC 3D

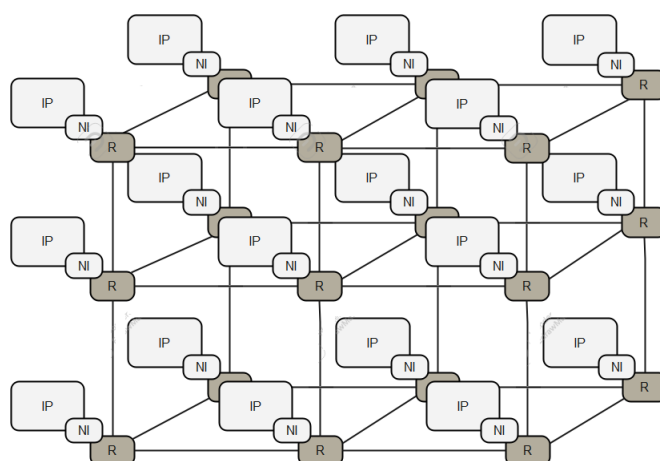


FIGURE 2.17 – Un réseau sur puce 3D

2.4.1 Avantage du 3D par rapport au 2D

L'intégration de la troisième dimension du réseau sur une puce électronique présente plusieurs avantages, notamment:

1. Réduire la longueur du fil d'interconnexion en raison de l'empilement, comme illustré dans la Figure 2.18.
2. Réduire le nombre moyen de saut (*Hops*) grâce à l'augmentation du nombre de liens d'interconnexions. Un routeur dans un réseau à 3 dimensions a au moins un lien de connexion supplémentaire par rapport à un routeur dans un réseau de deux dimensions. Le calcul du nombre de liens d'interconnexions et le nombre moyen de hops est donné par l'équation 2.1 et l'équation 2.2 [5] [41]:

$$links = N_1N_2(N_3 - 1) + N_1N_3(N_2 - 1) + N_2N_3(N_1 - 1) \quad (2.1)$$

$$HopsNocs = \frac{n_1n_2n_3(n_1 + n_2 + n_3) - n_3(n_1 + n_2) - (n_1n_2)}{3(n_1n_2n_3 - 1)}, \quad (2.2)$$

où N_i représente le nombre des commutateurs dans la dimension i .

3. La réduction des fils d'interconnexion réduit la congestion du routage et augmente finalement les performances. Ainsi, la division stratégique a un impact fort sur l'amélioration du temps de transition.
4. La réduction de la consommation d'énergie, démontrée à travers l'étude d'expérimentation menée sur l'additionneur Kogge-Stone, a montré une diminution de Puissance par 8%, 15% et 22% pour empiler 3, 2 et 4 niveau les uns sur les autres [42].
5. Atténuation des problèmes de mémoire en fournissant des liens de connexions verticales courtes et permet l'augmentation de la capacité de mémoire sur la puce.
6. L'empilement vertical réduit la distance entre les Ip ce qui améliore la latence des communications avec des connexions courtes [43].
7. Plusieurs liens d'interconnexion verticaux et horizontaux, ce qui améliorent la bande passante entre les couches. De plus elle réduit la congestions en répartissant le trafic sur différents niveaux [43].

2.4.2 Composants du réseau sur puce 3D

Les éléments clés pour transformer un NoC 2D en NoC 3D consistent d'un routeur 3D qui est une extension du routeur 2D, composé de tous les éléments du routeur 2D, auxquels s'ajoutent des éléments permettant de router les données entre les niveaux. En plus des liens horizontaux, les NoC 3D utilisent des liens verticaux pour permettre la connexion et le transport des données verticalement entre les différents niveaux de la puce empilés, la Figure 2.19 illustre les composant du NoC 3D.

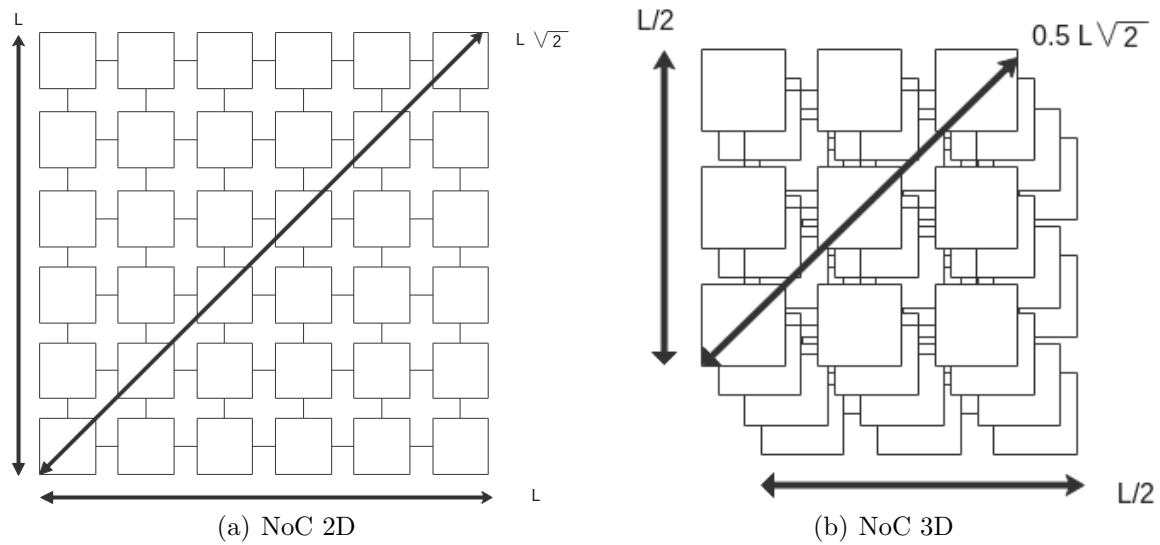


FIGURE 2.18 – Avantage du NoC 3D & NoC 2D [6]

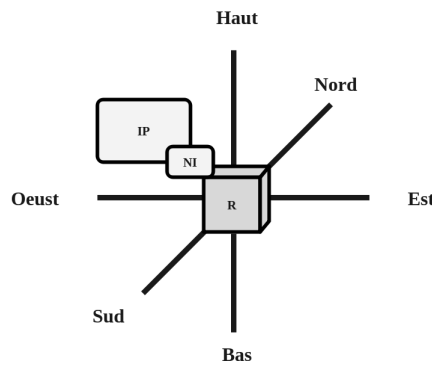


FIGURE 2.19 – Routeur et liens horizontaux et verticaux dans un NoC 3D.

Dans un routeur 3D, les éléments clés du routeur 2D, tels que les ports d'entrée et de sortie, les tables de routage, les buffers de données et les unités de contrôle, sont présents. Cependant, des éléments supplémentaires sont inclus pour faciliter le routage des données verticalement entre les différents niveaux du réseau sur puce. Ces éléments supplémentaires comprennent des ports et des canaux de communication verticaux. un exemple de routeur 3D est illustré dans la Figure 2.20.

Diverses architectures des routeurs 3D sont envisageables, dont quelques unes sont décrites ci-dessous [44] [41]

1. Routeur en 3D de base ou routeur symétrique: C'est en quelque sorte un routeur 2D qui a subi des modifications. Cela concerne l'ajout de deux ports pour la communication verticale, ce qui engendre le remplacement du crossbar (5x5) du routeur 2D par un crossbar (7x7) et une mise à jour des algorithmes de routage (pour être applicables en 3D). La Figure 2.21(a) présente un routeur symétrique. Quelque soit le lien (horizontal ou vertical), le routeur 3D le considère toujours comme un saut.

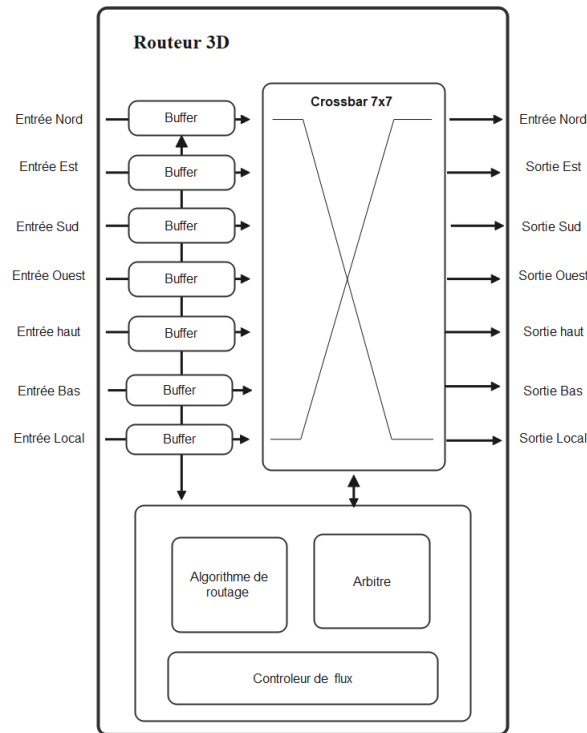


FIGURE 2.20 – Un élément de routeur 3D

Bien qu'il réduit le nombre de sauts (grâce à la réduction de la taille du NoC sur chaque niveau), sa surface se trouve augmentée ainsi que sa puissance (à cause de l'augmentation du routeur).

2. Routeur 3D hybride: Le routeur 3D hybride a pour principale caractéristique la dotation d'un bus partagé, pour assurer la communication verticale entre les routeurs 2D qu'ils soient voisins ou pas. Cela implique l'ajout d'un seul port supplémentaire pour l'accès à ce bus et donc l'utilisation d'un crossbar 6x6. La Figure 2.21(b) illustre un routeur hybride. Ce type de routeur a l'inconvénient de créer une forte contention lorsque la communication inter routeurs augmente sur le même bus. Par contre il permet de réduire la latence globale.

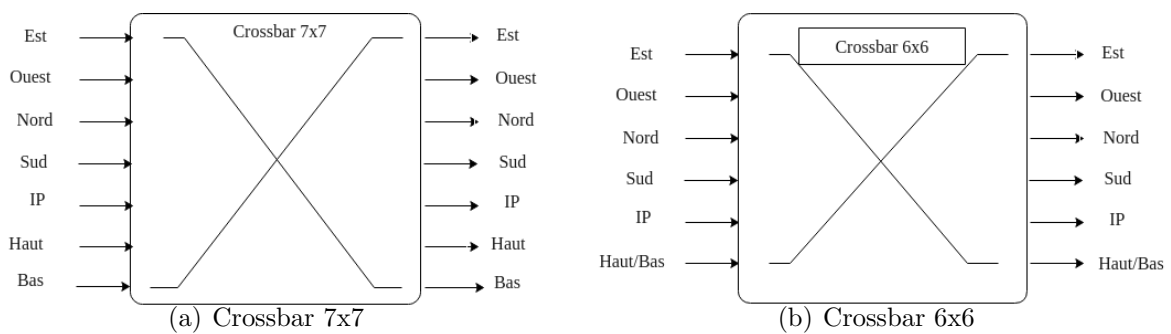


FIGURE 2.21 – Crossbar du Routeur 3D

3. Véritable routeur 3D: Il se base sur l'utilisation d'un crossbar 3D qui est une évolution du crossbar 2D. La communication intra-niveau (horizontale) se fait à travers un crossbar 2D tandis-que la communication inter-niveau (verticale) est possible grâce à l'implantation de boîte de connexion (extraction de 25 boîtes de connexions pour un crossbar 2D de 5x5). La Figure 2.22 présente un véritable routeur. L'avantage dans ce routeur 3D est que la communication dans la même ligne verticale est toujours considérée comme un seul saut même entre deux niveaux non adjacents. Cependant, cela n'est pas sans conséquence sur la complexité de la logique d'arbitrage.

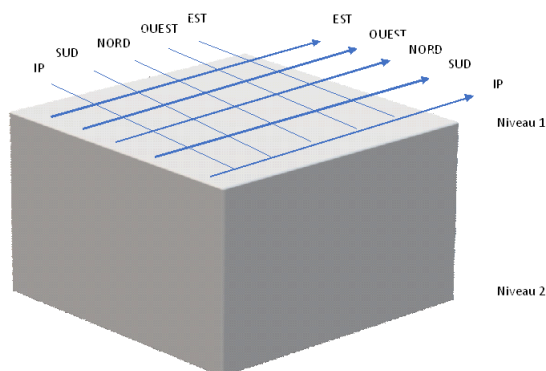


FIGURE 2.22 – Crossbar 3D Du routeur 3D

4. Routeur 3D décomposé selon les dimensions: Comme son nom l'indique, ce type de routeur se base sur la direction de déplacement pour séparer le flux de données. Ces données sont classées selon leur déplacement en trois catégories: Est-Ouest, Nord-Sud et Haut-Bas. La Figure 2.23 illustre le routeur selon les dimension. A chaque direction est dédié un module (ligne, colonne et verticale) qui reçoit les données aiguillées par un démultiplexeur 1x4. Seuls les modules ligne et colonne sont dotés d'un crossbar 4x2. Ce type de routeur 3D réduit considérablement la taille du crossbar et aussi la contention.

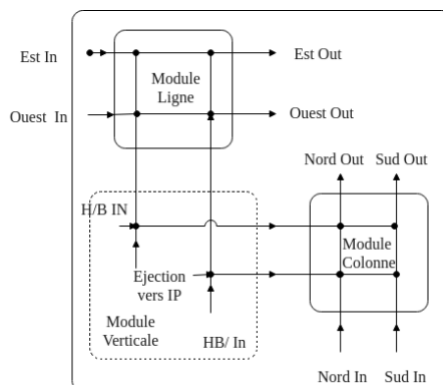


FIGURE 2.23 – Routeur selon Directions

En plus de routeurs, le principe de la structure tridimensionnelle est d'empiler plusieurs couches les unes sur les autres, ces couches sont à leur tour reliées entre elles par des liens verticaux. La Figure 2.24 montre un résumé de quelques liens possibles [45] [46]:

1. Fil collé: technique la plus connue, dans laquelle les fils relient chaque matrice dans l'empilement de la structure 3D.
2. Microbump: l'approche Microbump utilise des perles d'or ou de soudure sur la surface de chaque puce pour établir les connexions requises.
3. Sans contact: cette technique utilise un couplage capacitif ou inductif pour communiquer entre différentes matrices dans la dimension verticale.
4. Through Silicon Via: L'approche TSV fournit la densité d'interconnexion maximale possible. A travers ce lien l'interconnexion peut se faire que ce soit en face-à-face ou en face à dos. Le TSV [47], utilise des connexions électriques verticales courtes ou "vias" qui traversent une plaquette de silicium afin d'établir une connexion électrique du côté actif à l'arrière de la puce, fournissant ainsi le chemin d'interconnexion le plus court et créant une avenue pour le nec plus ultra de l'intégration 3D.

Les TSV sont des trous créés dans une plaquette de silicium à l'aide d'un processus de gravure. Les interconnexions sont formées en remplissant les TSV d'un matériau conducteur, tel que le cuivre ou le polysilicium. Le principal avantage des interconnexions TSV est le raccourcissement des chemins pour que le signal se déplace d'une puce à la suivante, ou d'une couche de circuits à la suivante. Cela permet de réduire la puissance et la possibilité d'augmenter la densité d'interconnexion, augmentant ainsi la fonctionnalité et les performances.

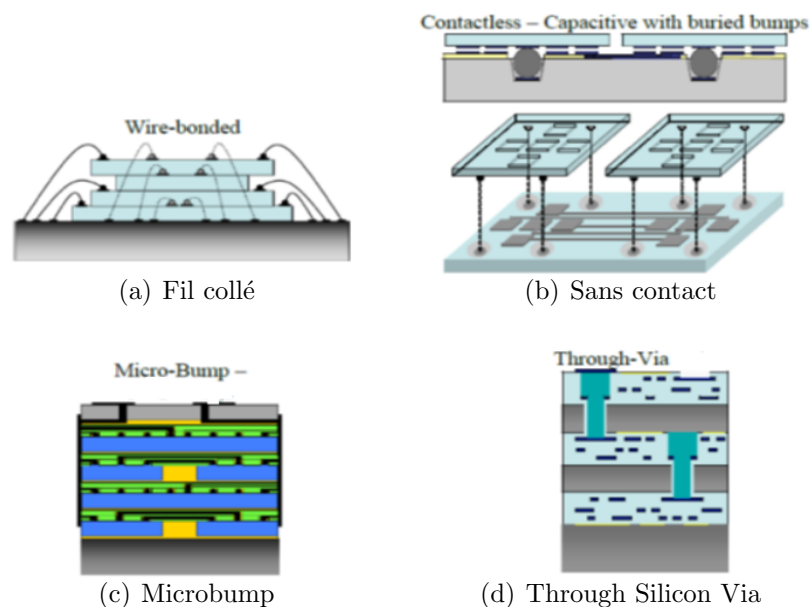


FIGURE 2.24 – Type de liens verticaux [45]

2.5 Conception des réseaux sur puce

La conception d'un réseau sur puce efficace, satisfaisant les critères de l'application, est un processus complexe et portant sur plusieurs niveaux de modélisation jusqu'à l'implémentation physique. Il est nécessaire d'automatiser les tâches de conception par des méthodes et des outils. La mise en œuvre des réseaux d'interconnexion impose différentes contraintes qui se trouvent à plusieurs niveaux [14]:

1. Niveau Architectural: Le choix de la meilleure topologie qui répond aux exigences et aux besoins de l'application ciblée, est complexe et très long à effectuer. Dans ce niveau, la performance et l'efficacité du NoC dépendent fortement de l'architecture d'interconnexion sur puces. Les commutateurs sont les composants actifs qui influencent fortement la latence et le débit de la communication sur puce, tout comme les liens qui véhiculent les données entre les commutateurs. Cependant, une affectation efficace de leur bande passante, une optimisation de leurs distances, ou une augmentation de leur nombre peut améliorer la performance des applications SoC.
2. Niveau Application: Le pouvoir d'exploiter le parallélisme des tâches et la capture des communications, est le défi principal de la conception. Au niveau application, le placement et l'ordonnancement (*scheduling*) des tâches de l'application dans les plateformes NoC, tout en optimisant les mesures de coûts et de performance, constituent le principal point à prendre en considération .
3. Au niveau de la communication: Le placement de l'application sur l'architecture de façon que les coûts soient minimisés, est délicat, car un mauvais placement peut engendrer une violation des contraintes de bande passante ou de temps réel. Au niveau communication, la commutation, le contrôle de flux et le routage des données sont des aspects essentiels de la communication. L'utilisation de techniques de routage adaptatif, de commutation adaptative et de contrôle de flux sophistiqué permet d'améliorer la latence du système en évitant les routeurs défectueux ou surchargés. Étant donné l'impact de l'architecture NoC sur les performances globales, le routage est devenu la principale stratégie à considérer lors de la conception. Les stratégies de routage peuvent être catégorisées en régimes déterministes et adaptatifs.

Cette étude de l'existant nous a permis de se positionner par rapport à ce qui se fait actuellement dans le domaine des NoCs. Ainsi, nous avons proposé trois approches d'optimisation des performances du NoC qui seront présentées en détail dans le chapitre 6 de cette thèse. Ces trois approches permettent aux concepteurs d'optimiser et de personnaliser une architecture d'interconnexion sur puce afin qu'elle corresponde à une grande charge de travail des applications SoCs. L'objectif de ces approches est d'explorer et d'évaluer les performances de l'architecture d'interconnexion sur puce pour mieux répondre aux besoins d'une maximisation de types d'applications. Le travail de cette thèse s'inscrit à deux niveaux de l'architecture NoC, le niveau physique et le niveau communication. En

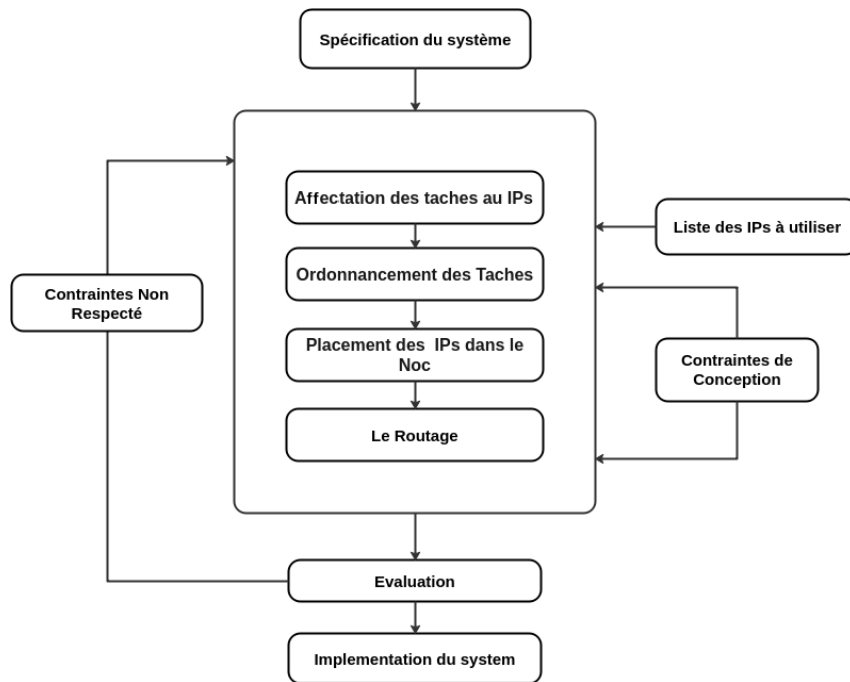


FIGURE 2.25 – Les étapes de conception d’une application sur un NoC

général, un système embarqué doit être mis en œuvre afin de répondre à des exigences telles que le coût du matériel, la consommation d’énergie et le temps d’exécution. La Figure 2.25 présente un organigramme qui simplifié les phases de conception d’un système embarqué, basé sur un réseaux sur puces.

La première étape du projet consiste à spécifier le système où des contraintes du système sont exigées, telles que le temps maximum d’exécution ou la surface du Matériel. Dans cette étape également, l’application à exécuter par le système est définie. Celle-ci est décrite par un graphe de tâches, où chaque tâche est un algorithme spécifique. L’étape suivante peut être définie comme une étape d’optimisation: à partir des spécifications du système, certaines caractéristiques du réseau sont optimisées par un outil d’aide à la conception afin de répondre aux contraintes de conception. Dans cette recherche, nous nous focalisons sur la deuxième étape où nous présentons les trois principales phases d’optimisation en lien avec la conception d’un NoC: l’affectation, le placement et le routage.

2.5.1 Affectation des tâches aux IPs

L’attribution des IPs est la première étape avant de mapper l’application sur NoC [48]. L’objectif de l’affectation des tâches aux IPs est de sélectionner, à partir d’une bibliothèque des IPs, un ensemble d’IP qui permet la réutilisation et optimise la mise en œuvre d’une application donnée en termes de temps d’exécution, de puissance et de surface. Durant cette étape, il n’est donné aucune information sur l’emplacement physique des IPs sur NoC [49]. Le processus d’optimisation doit être basé uniquement sur les fonctionnalités

du TG et des IP. Le résultat de cette étape est un ensemble d'IP qui doit maximiser les performances du NoC. C'est-à-dire minimiser la consommation d'énergie, les ressources matérielles ainsi que le temps total d'exécution de l'application.

Rappelons que le résultat de cette étape est produit sous la forme d'un graphe de caractérisation de l'application (ACG – *Architecture Characterization Graph*). Dans ce graphe chaque tâche est associée à un IP. La dynamique de cette étape d'affectation est illustrée dans la Figure 2.26.

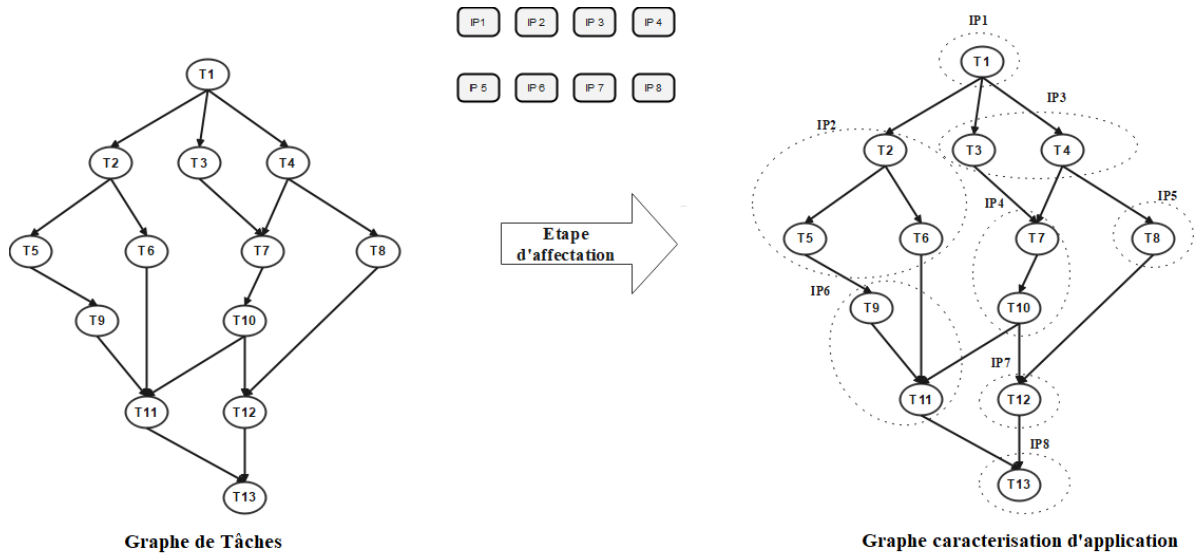


FIGURE 2.26 – Le problème d'affectation

2.5.2 Placement des IPs

Étant donné qu'il s'agit d'une application, le problème est de déterminer la manière de mapper topologiquement les IPs sélectionnées sur la structure du réseau. De cette façon les objectifs sont optimisés [50]. A ce stade et lors d'une session de communication une évaluation plus précise doit être entreprise. Celle-ci doit tenir compte de la distance entre les ressources, du nombre de routeurs et de canaux traversés par un paquet de données. Le résultat de ce processus devrait être une affectation optimale de l'attribution IPs prescrite pour exécuter l'application sur le NoC. Figure 2.27 présente l'étape de placement des IPs.

2.5.3 Routage des communications

Le routage détermine le chemin à emprunter pour le paquet entre la source et la destination. Le choix de l'algorithme de routage est une tâche critique et doit être pris en compte lors de la conception d'un NoC [51]. Parmi les problèmes de routage dans les réseaux sur puces nous pouvons citer la congestion, l'interblocage, le débit, la consommation

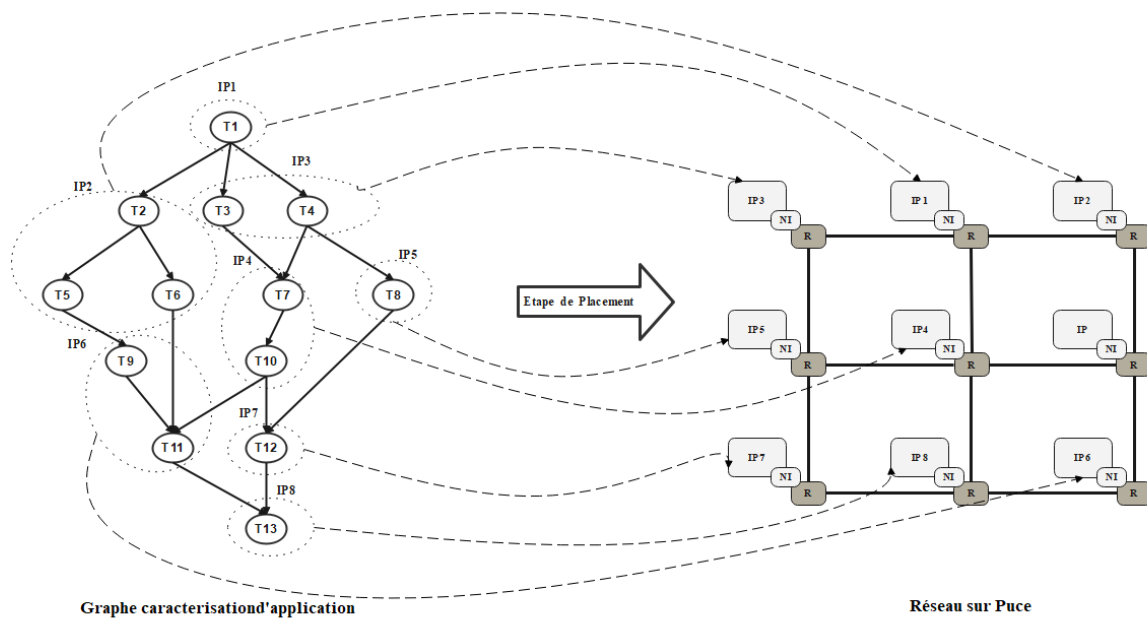


FIGURE 2.27 – Le problème de placement

d'énergie et la latence. Le chemin décidé dans l'algorithme de routage est utilisé pour réduire la congestion dans le réseau ainsi que le délai de communication entre les liens [52] [53]. En conséquence, le routage est devenu un axe de recherche depuis la proposition de NoC. Ci-dessous la Figure 2.28 qui illustre l'étape de routage.

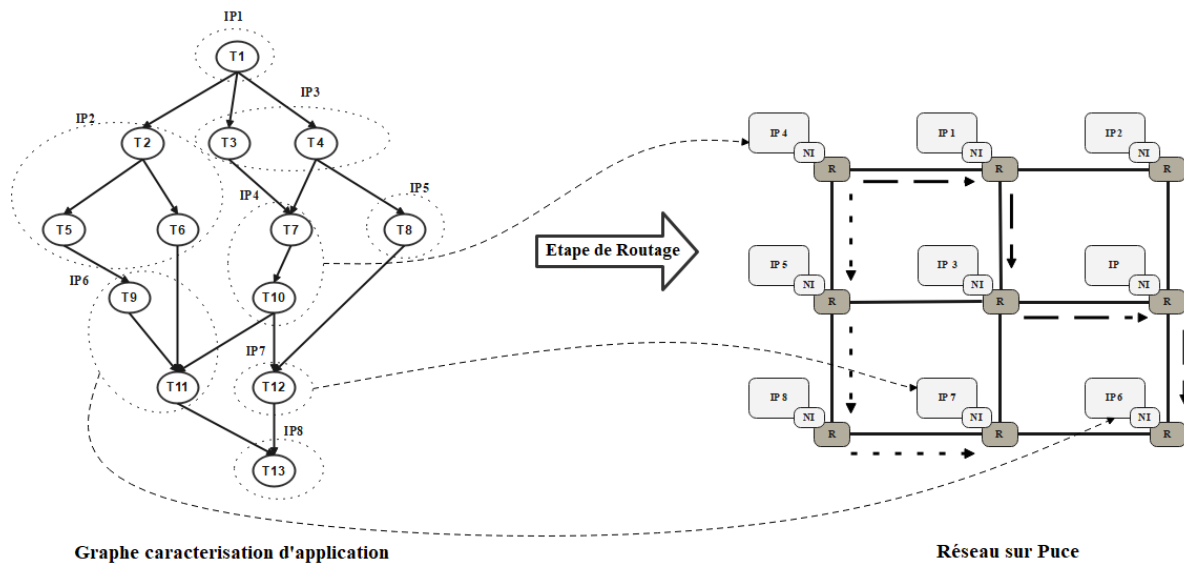


FIGURE 2.28 – Problème de routage

2.6 Considération finale du chapitre

Dans ce chapitre, nous avons présenté les réseaux sur puces (NoCs) censés résoudre les problèmes rencontrés dans les structures d'interconnexions des SoCs. Nous avons également abordé les composants d'un NoC ainsi que la conception des réseaux sur puces.

Étant donné que le problème traité est un problème d'optimisation NP-difficile, il est nécessaire de le résoudre à l'aide de méthodes d'optimisation appropriées. Dans le chapitre 3, nous nous focaliserons sur les méthodes les plus en vogue dans la littérature, en relation avec les trois principales phases d'optimisation, en lien avec la conception d'un NoC.

Chapitre 3

Etat de l'art des travaux de recherche concernant les problèmes de conception du NoC

Dans ce chapitre, une classification des méthodes d'optimisation utilisées dans la conception des réseaux sur puces, est présentée, telles qu'elles sont décrites dans la littérature. Comme expliqué dans le chapitre 4, la conception d'une application implémentée sur un NoC implique trois phases d'optimisation principales: L'affectation, le placement et le routage.

Dans le cadre de cette thèse, nous avons effectué une classification des travaux de recherche relatifs à l'optimisation des réseaux sur puces en fonction de quatre critères principaux comme illustré dans la Figure 3.1:

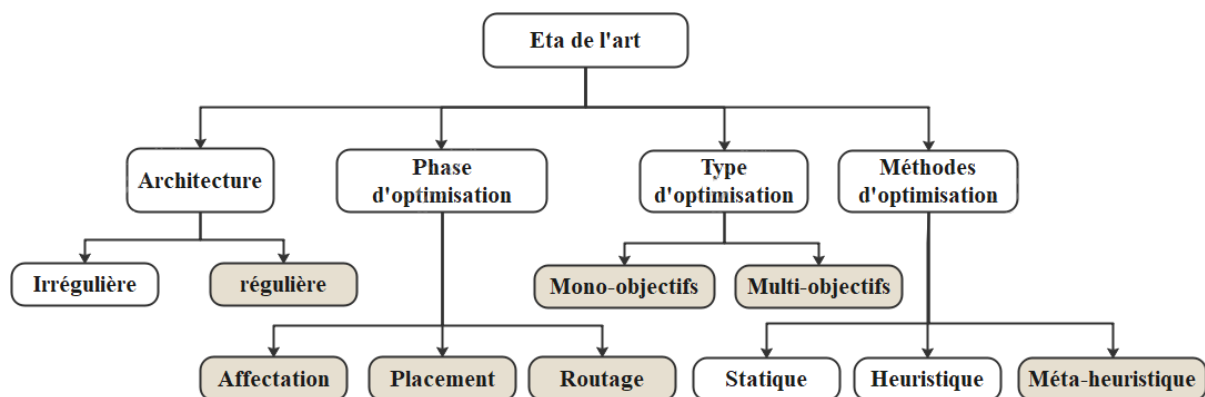


FIGURE 3.1 – Les classes de méthodes d'optimisation sur les NoCs

1. Architecture: Cette classification se base sur la topologie du réseau sur puce utilisée, qui peut être régulière, irrégulière ou reconfigurable.

2. Phase d'optimisation: Nous avons classé les méthodes en fonction de la phase d'optimisation à laquelle elles sont appliquées. Cela peut inclure la phase de routage, la phase d'affectation des ressources, la phase de placement des ressources sur l'architecture, etc. En classant les méthodes de cette manière, nous pouvons mieux comprendre comment elles contribuent à différentes étapes du processus d'optimisation des réseaux sur puces.
3. Type d'optimisation: Nous avons également classé les méthodes en fonction du type d'optimisation qu'elles cherchent à réaliser. Cela peut inclure la minimisation d'un seul objectif ou de plusieurs objectifs à la fois. Les objectifs peuvent être classés à leur tour selon la phase d'optimisation à laquelle ils se rapportent. Par exemple, la minimisation du temps d'exécution, de la consommation d'énergie et de l'espace peuvent être des objectifs pour les phases d'affectation et de placement. L'objectif de congestion, tolérance au panne et d'évitement des interblocages peut être pertinent pour la phase de routage.
4. Méthode d'optimisation: Nous avons également classé les méthodes en fonction de la méthode utilisée pour résoudre les problèmes d'optimisation. Cela peut inclure des méthodes exactes, des heuristiques et des méta-heuristique. En examinant les différentes méthodes utilisées, nous pouvons évaluer leur efficacité et leurs limitations dans le contexte des réseaux sur puces.

En ce qui concerne l'architecture des réseaux sur puces, notre intérêt se limite aux architectures régulières en 3D. Nous avons choisi de nous concentrer sur ces architectures en raison de leur popularité et de leur pertinence dans de nombreuses applications.

Cette thèse propose des solutions en ce qui concerne des NoCs dont l'architecture est régulière 3D, les étapes d'allocation, placement et routage, en utilisant l'optimisation multi-objectifs à travers des stratégies basées sur des méta-heuristiques.

3.1 Affectation des tâches

Pendant la phase d'affectation, le processus d'optimisation se base uniquement sur les fonctionnalités du graphe de tâches de l'application et des IPs [49], sans tenir compte de l'emplacement physique des IPs sur le NoC. L'objectif de cette phase consiste à sélectionner, à partir d'une bibliothèque d'IPs, un ensemble d'IPs favorisant la réutilisation et permettant une optimisation multi-objectifs.

Jena et Sharma [54] ont étudié l'affectation des tâches aux IPs dans un NoC 2D en utilisant l'algorithme multi objectifs NSGA II. L'objectif principal était de trouver une solution minimisant la consommation d'énergie due aux activités d'exécution et de communication, tout en répondant aux contraintes de performances spécifiées liées à la bande passante.

Da Silva *et al* [55] ont proposé une affectation efficace des IPs en utilisant NSGA II et microGA pour l'optimisation multi-objectifs des plateformes NoC. Les objectifs principaux étaient l'optimisation de la superficie, de la consommation d'énergie et du temps d'exécution.

Dans le travail de [34], Radu a examiné trois méthodes d'affectation des tâches aux IPs: Affectation aléatoire où plusieurs tâches peuvent être attribuées à la même IP, affectation directe où une tâche est attribuée à la première IP capable de l'exécuter et affectation minimale où la sélection est motivée par le temps minimal, et donc une tâche est toujours attribuée à l'IP qui s'exécute le plus rapidement. Dans une autre étude, Radu [56] a utilisé les algorithmes SPEA II et NSGA II avec différents méthodes de croisement et de mutation pour l'affectation et le placement des IPs. L'objectif était d'optimiser la consommation d'énergie et l'équilibre thermique.

Lee Qian-qi *et al* [57] ont proposé des algorithmes multi-objectifs améliorés basés sur les exigences et les contraintes de conception des NoC. Les algorithmes hybride, PSO avec GA et PSO avec SA, ont été développés pour bénéficier des avantages des algorithmes intelligents, tout en évitant les optima locaux. Le processus de recherche de la meilleure solution a été basé sur des stratégies plutôt que sur un caractère totalement aléatoires. Ce qui a abouti à l'obtention de meilleures solutions.

3.2 Placement des IPs

Le problème du placement dans les NoCs a fait l'objet de nombreuses études et recherches. Différentes approches ont été proposées pour comparer ses performances [58–62]. La plupart de ces travaux classent les méthodes de placement en fonction des algorithmes ou des techniques utilisées. Cette classification permet de mieux comprendre les différentes approches et méthodes utilisées pour le placement. Il en existe deux types: Statique et dynamique. Dans le placement statique, tous les IPs sont affectés aux tuiles de l'architecture avant l'exécution de l'application. Cette approche est réalisée une fois pour des plateformes spécifiques, et reste fixe pendant l'exécution de l'application. Le placement dynamique est effectué pendant l'exécution de l'application. Il permet d'ajouter, supprimer ou déplacer des tâches sur les IPs disponibles. Le but est de reconfigurer la plateforme de communication tout en minimisant le temps de remplacement des tâches. Dans cette section, nous avons présenté les méthodes qui utilisent le placement statique. Ensuite, nous présentons les méthodes de placement qui utilisent des algorithmes méta-heuristiques.

3.2.1 Placement utilisant une méthode exacte

Dans [63], la méthode statique basée sur la programmation linéaire en nombre d'entier est présentée, elle propose d'utiliser trois algorithmes de placement pour les NoCs 3D afin d'explorer les contraintes thermiques et leur impact sur la température et les performances.

Un nouvel algorithme heuristique pour le placement d'applications et les TSVs dans le NoC 3D est proposé dans [64]. Il réduit le coût global de la communication et la consommation d'énergie. L'algorithme a été comparé au coût de communication avec les méthodes existantes. Il montre une bonne amélioration en termes de coût de communication. De plus, une comparaison des paramètres de performances dynamiques tels que le débit, la latence et la consommation d'énergie a également été effectuée. Une heuristique constructive basée sur la prédiction est présentée dans [65]. Cette approche vise à réduire le coût de communication. Les résultats de ce placement, en termes de coût de communication, ont été comparés à de nombreuses techniques bien connues, rapportées dans la littérature, y compris une méthode exacte basée sur ILP, présentée dans le même article. D'après cette comparaison, dans de nombreux cas, la technique de placement produit des résultats similaires à ceux de l'ILP avec moins de temps d'exécution.

Dans [66], un placement d'applications dans une architecture NoC 3D est proposée, il est basé sur la séparation et l'évaluation. L'objectif de cette technique est de minimiser l'énergie de communication du système. Des expériences ont été menées sur différents benchmarks aléatoires afin de vérifier l'efficacité de l'algorithme. Dans [67], deux approches de placement des applications sur un NoC 3D sont présentées: Une basée sur la programmation linéaire en nombre entier et un algorithme heuristique, qui visent à minimiser l'énergie. Cette méthode privilégie l'utilisation des liens verticaux pour la communication, car ils sont plus courts et consomment moins d'énergie que les liens horizontaux. Une comparaison entre ces deux approches est réalisée en termes de consommation d'énergie et de temps d'exécution sur plusieurs benchmarks. Les résultats montrent que la méthode heuristique obtient des résultats proches de l'optimum avec des délais beaucoup plus courts.

3.2.2 Placement utilisant une méta-heuristique

Plusieurs travaux de placement qui visent à utiliser des algorithmes méta-heuristiques. Un algorithme de placement sensible à la chaleur est présenté dans [68]. Cet algorithme réalise automatiquement le placement des cœurs IPs sur l'architecture 3D en prenant principalement en compte l'écart de température. Il utilise GA pour le placement, puis le résultat obtenu est simulé en utilisant un outil de simulation du trafic ainsi qu'un autre outil de simulation pour calculer la température. L'efficacité de cet algorithme a été vérifiée en testant différentes applications multimédia de référence. L'algorithme GA est utilisé dans [69] pour trouver le meilleur placement permettant d'atteindre une

consommation d'énergie minimale pour une application donnée. Une étude de cas d'une application multicœur avec 32 microprocesseurs symétriques est présentée à titre d'exemple. L'algorithme prend moins de temps, alors qu'il a fallu plus de trois jours en utilisant une recherche exhaustive.

Un algorithme de placement est proposé dans [70] pour réduire la température maximale, tout en minimisant la consommation d'énergie de communication dans un NoC 3D. Le NoC 3D est considéré comme un ensemble de tuiles alignées verticalement, formant une super-tuile où chaque ligne verticale contient des noyaux ayant des températures similaires. L'algorithme génétique est utilisé en deux phases. La première phase prend en compte le placement des tâches en groupe, tandis que la seconde phase place chaque groupe sur une super-tuile en prenant en compte la consommation d'énergie des tâches, la consommation d'énergie de communication et les capacités d'émission de chaleur des IPs. Cela permet de réduire plus rapidement la température maximale.

Une méthode hybride basée sur l'algorithme génétique et le recuit simulé est proposée dans [71]. Elle améliore le placement en optimisant la population initiale, ce qui accélère et précise la recherche de l'individu optimal. La phase de mutation de l'algorithme génétique est remplacée par le recuit simulé pour renforcer la capacité d'optimisation globale de l'algorithme et réduire les risques de convergence vers des optimaux locaux. Les expériences montrent que la méthode hybride a un bon effet dans la résolution de placement dans les NoCs 3D à faible puissance. Dans le cas de 124 cœurs IP, la consommation électrique maximale générée est réduite de 35,3% par rapport à GA.

Dans la méthode présentée dans [72], un algorithme amélioré basé sur GA pour le placement efficace des IPs sur le NoC 3D. Pour améliorer la sélection initiale des individus, un algorithme glouton amélioré est utilisé pour générer une population plus proche de l'optimum. De plus, pour éviter les problèmes de convergence prématurée, l'algorithme de recuit simulé est intégré dans l'étape d'opération de croisement pour rechercher l'optimum global. Les résultats expérimentaux montrent que cet algorithme présente de meilleures performances de convergence et une faible consommation d'énergie par rapport à l'algorithme génétique traditionnel.

La méthode présentée dans [73], évoque un placement basé sur l'optimisation par PSO discrètes pour placer les applications sur des réseaux sur puce 2D et 3D. Les résultats obtenus avec l'approche PSO sont supérieurs à ceux des techniques rapportées, notamment pour des benchmarks de taille réduite. Un algorithme efficace en termes d'énergie est proposé dans [74]. En plus de l'approche hybride, l'algorithme PSO est utilisé. Cet algorithme a montré une réduction plus importante de la consommation d'énergie de communication et une amélioration des performances du système, lorsqu'il a été évalué sur des applications de référence générées aléatoirement.

Le travail présenté dans [75] propose un algorithme de placement à faible puissance, basé sur PSO. Les résultats montrent que cet algorithme permet une réduction significative de la consommation d'énergie. Cependant, pour des applications à grande échelle, l'efficacité de l'optimisation de la puissance est limitée. Afin de résoudre ce problème, les mêmes auteurs proposent dans [76] un algorithme de placement à faible puissance basé sur l'optimisation de PSO à comportement quantique, contrôlé par la diversité. Cette approche améliorée permet d'obtenir de meilleurs résultats en termes d'optimisation de la puissance, notamment pour des applications de plus grande envergure. Les résultats de la simulation montrent que la consommation d'énergie est réduite et que la capacité d'optimisation est améliorée par rapport à l'algorithme standard de PSO.

Dans la méthode présentée dans [77], un placement des tâches basée sur la logique floue est proposé pour atténuer les problèmes de puissance et de température dans les NoCs 3D. Cette méthode permet de modifier les poids des facteurs de placement en fonction des exigences spécifiques de la puce. Ceci conduit à une réduction du délai moyen de communication, de la consommation d'énergie et de la température maximale.

Dans [78], la méta-heuristique inspirée du comportement de chasse des chauves-souris, BA, est utilisée pour un placement à faible consommation d'énergie dans un NoC 3D. L'algorithme présente de meilleures performances en termes de convergence et de consommation d'énergie par rapport aux algorithmes traditionnels de placement basés sur GA et PSO. Cependant, l'algorithme présente certaines faiblesses. L'auteur améliore son algorithme en proposant d'utiliser un groupe de recherche de chauves-souris pour augmenter l'utilisation individuelle et la diversité du BA. Les résultats expérimentaux montrent que le BA amélioré avec la stratégie de recherche de groupe est nettement supérieur au BA standard.

Une méthode de placement d'application sensible à l'énergie et aux performances est présentée dans [79] pour les NoCs 3D. Cette méthode utilise des architectures non homogènes, combinant des routeurs 2D et 3D dans les NoCs 3D. Ce qui permet de réduire la surface et la consommation d'énergie tout en maintenant les performances des NoCs 3D homogènes. Dans [80], Un placement basé sur l'algorithme stochastique appelé allocation simulée est proposé. Il utilise un modèle de latence pour les routeurs qui prend en compte les conflits réseau résultant du partage des ressources. L'objectif de l'algorithme est d'optimiser la puissance de communication, la latence et la fiabilité des TSV. Les résultats montrent que l'algorithme peut obtenir de meilleurs résultats de puissance et de latence que les travaux antérieurs, et l'approche peut améliorer considérablement la fiabilité.

Dans cette partie, nous présentons les méthodes de placement qui vise une optimisation multi-objectifs sur les NoCs 3D. L'algorithme génétique multi-objectifs basé sur les rangs est utilisé dans [81] pour un placement sensible à la latence. L'objectif de cet algorithme est d'obtenir un front de pareto proche de l'optimal. Il prend en considération deux

objectifs avec deux modèles de latence différents pour les situations de non congestion et de congestion. L'objectif est de trouver des solutions qui minimisent à la fois la latence et la congestion du réseau.

Une nouvelle méthode de placement est introduite dans [82]. Son but est de réduire la communication inter-niveau et de minimiser le nombre de liens TSV dans les réseaux sur puce 3D. L'algorithme utilise une approche en deux étapes basées sur le recuit simulé, le placement inter-couche et intra-couche. Deux objectifs sont pris en compte, à savoir le nombre de TSV et la température, dans la même fonction objectif.

La méthode présentée dans [83] propose une technique de placement d'IPs pour les NoCs 3D hétérogènes. L'objectif principal de cet algorithme est d'avoir un placement efficace d'une application donnée. Le processus de cette méthode comporte trois étapes principales: La première étape consiste à déterminer la taille initiale du NoC et des clusters, pour réduire le coût des routeurs 3D. Ensuite, le graphe d'application est affecté aux clusters du NoC 3D, de sorte qu'un nombre optimisé de routeurs 2D et 3D sont attribués, en énumérant la bande passante de communication et les contraintes énergétiques dans chaque cluster. Enfin, les cœurs IP sont mapés sur les tuiles des clusters du NoC, en minimisant la consommation d'énergie totale et en maximisant les performances du système.

Un algorithme immunitaire adaptatif multi-objectifs est proposé dans [84] pour résoudre les problèmes de topologie et de placement dans les NoC 3D multi-applications. L'algorithme vise à réduire la latence et la consommation d'énergie du réseau en considérant des fonctions multi-objectifs. Il explore l'espace de recherche en générant un ensemble de meilleures alternatives NoC 3D. Une version modifiée de cet algorithme est également présentée dans [85], utilisant un test immunitaire multi-objectifs pour le placement des tâches dans le NoC 3D.

La méthode présentée dans [86] évoque deux approches de placement. La première est un placement 3D centralisé qui utilise une technique de parcours octaédrique pour placer les IPs sur des architectures NoCs 3D. Il s'agit d'une extension en 3D de la méthode de placement sur le NoC 2D présenté dans [87]. La seconde méthode utilise l'algorithme PSO attractif répulsif. Les algorithmes déterministes présentés dans [88] et [89] sont combinés pour obtenir un placement multi-objectifs efficace. Les deux approches exploitent efficacement l'espace de conception du NoC en minimisant la consommation d'énergie de communication et les délais. Les algorithmes proposés ont été évalués pour des benchmarks aléatoires et réels, et les résultats démontrent clairement une réduction significative de la consommation d'énergie de communication.

La méthode présentée dans [90] consiste en un algorithme de placement des tâches sensible à la latence avec des TSV partiellement remplis. L'algorithme repartie les tâches d'un graphe d'application en groupes, selon leurs communications à haut volume. Ensuite il mappe chaque groupe de tâches sur une couche de topologie NoC, basée sur un maillage 3D.

L'objectif est de réduire le trafic dans les TSV en regroupant les tâches de manière efficace. De plus, l'algorithme vise à minimiser le nombre de sauts dans le NoC en plaçant les tâches de chaque groupe sur des cœurs adjacents. Cette approche divise le problème de placement en sous-problèmes, ce qui facilite le processus de placement. Les évaluations révèlent que les améliorations de l'algorithme augmentent lorsque le nombre de TSV diminue et que le nombre de tâches augmente.

Un algorithme de placement basé sur le regroupement est utilisé dans [91]. Cet algorithme regroupe les tâches, puis les place sur une couche NoC 3D en utilisant la méthode de placement 2D présenté dans [92]. L'objectif est de minimiser à la fois le délai et la consommation d'énergie du réseau. Pour cela, les tâches ayant un poids de communication élevé sont placées à proximité les uns des autres, ce qui réduit les coûts de communication dans le réseau. L'algorithme proposé a considérablement réduit le temps d'exécution du placement. Ainsi, il convient de souligner que la méthode présentée est une approche de haute qualité et pratique pour les graphes de tâches à grande échelle.

Dans [93], une méta-heuristique inspirée de l'optimisation d'essaims de poulet auto-adaptatifs pour les NoC 3D est utilisée. Cette approche utilise une fonction objectif basée sur l'agrégation de la consommation d'énergie et du coût de communication. Elle met en œuvre une base cognitive, basée sur la méthode du regroupement partagé des voisins plus proche, pour un placement plus rapide. L'algorithme a été évalué par rapport à différentes alternatives d'algorithmes stochastiques, SA, ACO, GA, PSO et BA. Les résultats expérimentaux démontrent que cette approche surpasse les autres algorithmes bio-inspirés.

Un algorithme de placement mémétique basé sur les connaissances est proposé dans [94]. Cet algorithme combine l'algorithme évolutionnaire et la recherche locale pour atteindre un équilibre entre l'exploration et l'exploitation. L'algorithme commence avec une connaissance sur les détails du poids et des voisins de chaque nœud, ce qui rend le placement plus efficace et plus rapide. Il utilise la puissance, la surface et le délai comme critères de coût pour un placement efficace des tâches. Ces objectifs sont regroupés dans une fonction globale utilisant l'agrégation. Plusieurs topologies sont prise en compte en 3D. Les résultats de l'expérience prouvent que cet algorithme surpasse les autres en ce qui concerne l'optimisation multi-objectifs.

Dans [95], Un algorithme de placement multi-objectifs basé sur le recuit simulé est proposé, avec une fonction de coût. Celle ci prend en compte l'agrégation entre la surface, la bande passante et la latence. L'objectif est d'optimiser ces trois critères pour obtenir un placement efficace dans les applications des réseaux sur puce.

3.3 Routage

Le routage permet de trouver le chemin optimal pour acheminer un paquet de données de sa source à sa destination. Il existe plusieurs algorithmes de routage, chacun ayant des mécanismes spécifiques pour atteindre des objectifs précis. Les paramètres utilisés pour classer des algorithmes de routage sont:

1. Méthodes utilisées: Les algorithmes de routage peuvent être classés en trois catégories: Déterministe, adaptatif ou basés sur des méta-heuristiques. Les algorithmes de routage déterministes spécifient le même chemin pour chaque paire de nœuds source et destination, ce qui peut entraîner une augmentation de la latence, en particulier lorsque le réseau est encombré et subit une dégradation du débit. D'autre part, les algorithmes de routage adaptatifs permettent à un paquet de choisir parmi plusieurs chemins pour atteindre sa destination. Les chemins sont déterminés en fonction des conditions de congestion dans le réseau, offrant ainsi une certaine flexibilité et une adaptation aux variations de trafic. Enfin, certains algorithmes de routage utilisent des méta-heuristiques pour trouver le meilleur chemin entre une source et une destination. Ces méta-heuristiques utilisent un processus évolutif pour rechercher et optimiser le chemin, offrant ainsi une approche plus complexe et adaptative pour le routage.
2. Objectifs visés: Le NoC est confronté à plusieurs objectifs en matière d'optimisation, dont les principaux objectifs sont la minimisation de la latence pour réduire le délai de transmission des données, la consommation d'énergie pour optimiser l'utilisation énergétique, la maximisation du débit pour permettre des communications plus rapides, la garantie de tolérance aux erreurs avec des mécanismes de détection et de correction, l'évitement de la congestion ou des blocages pour prévenir les ralentissements, et l'équilibrage de la charge pour une répartition équitable du trafic et une meilleure utilisation des ressources disponibles.

Les travaux consacrés à trouver une solution au problème de routage dans les NoCs 3D peuvent être classés en trois catégories: déterministes; adaptatif et basé sur une méta-heuristique. Dans la suite, on fait le tour des travaux récents sur le sujet.

Les NoCs 3D utilisent souvent des algorithmes de routage déterministes en raison de leur simplicité. L'algorithme de routage le plus couramment utilisé dans ces systèmes est l'algorithme XYZ, qui est basé sur l'algorithme de routage statique d'ordre de dimension (DOR – *Dimension Order Routing*). Dans cet algorithme, les paquets sont acheminés successivement le long des dimensions X, Y et Z jusqu'à leur destination. Il est facile à implémenter. Il évite les impasses et les blocages, mais n'atténue pas le problème de congestion. L'algorithme présenté dans [96], est une amélioration du routage XYZ qui vise à réduire la latence, la consommation d'énergie et améliorer le débit. Il garantit l'absence d'interblocage et de livelock tout en conservant une approche déterministe.

Dans [97], un algorithme de routage utilisé dans les NoC 3D est présenté. Il se caractérise par sa simplicité, ses performances élevées, sa robustesse et sa déterminisme. Son objectif principal est de gérer les défaillances sur les liaisons verticales. En l'absence de défaillances, les paquets sont acheminés en utilisant le schéma ZXY. Cependant, en cas de défaillance sur les liaisons verticales, cet algorithme bascule vers le schéma XZXY. Les paquets sont d'abord acheminés vers un nœud intermédiaire accessible le long de la dimension X, à condition que le chemin ZXY de ce nœud vers la destination, soit sans défaut.

Les algorithmes adaptatifs introduisent une méthodologie dynamique pour le routage en fonction des conditions en temps réel, leur objectif est d'améliorer les performances et l'efficacité du réseau, tout en répondant aux variations de charge et aux modifications de la structure du réseau. L'algorithme adaptatif basé sur le modèle de tours pair impaire est étendue vers le NoC 3D, présenté dans [98]. Il applique des règles pour interdire certains tours dans les colonnes paires et impaires afin d'éviter les impasses.

Les algorithmes de routage 4 positifs premier et 4 négatif premier [99] sont des extensions de l'algorithme de routage basé sur le modèle de tours vers la 3D. Ils utilisent deux des trois directions positives (Nord, Est et Haut) ou négatives (Sud, Ouest et Bas) comme deux dernières directions pris par les paquets pour atteindre leur destination. Cela permet d'interdire et d'éviter les impasses. L'algorithme présenté dans [99] combine les deux et envoie une copie redondante en utilisant le schéma 4P-First en cas de défaillance du réseau.

Un algorithme tolérant aux pannes est utilisé dans un NoC 3D sans tampon présenté dans [100]. Il sélectionne la route avec le nombre minimal de sauts vers la destination le long des axes X, Y ou Z, tout en tenant compte de la charge de trafic des routes. Dans [101], un routage adaptatif non minimal est présenté, il vise à minimiser la latence et maximiser le débit dans les NoC 3D avec une bande passante verticale limitée. Il utilise des informations sur l'état du trafic pour répartir efficacement la charge sur les liaisons verticales.

L'algorithme de routage Dynamique XY est étendu à la troisième dimension dans [102]. Il choisit la direction la moins encombrée à chaque nœud intermédiaire pour acheminer les paquets de manière efficace. La congestion est évaluée en fonction de l'utilisation des tampons d'entrée des routeurs voisins. Un routage entièrement adaptatif dans un NoC 3D est présenté dans [103]. Il partitionne le réseau en groupes de clusters, connectés via un réseau de clustering léger pour distribuer les informations de congestion. Les décisions de routage sont basées sur le niveau de congestion des clusters voisins plutôt que sur les informations locales.

Un algorithme adaptatif et tolérant aux pannes dans un réseau en maillage 3D empilé est évoqué dans [104]. Il divise le réseau en quatre sous-réseaux et utilise différents canaux virtuels le long des dimensions X, Y et Z pour le routage. L'algorithme présenté dans [105] prend en compte les maillages verticaux YZ et XZ. Il utilise le routage DOR et le modèle

de tour pair-impair pour acheminer les paquets dans différentes directions en fonction de la congestion. Un algorithme de routage adaptatif minimal présenté Dans [106], Un algorithme de routage adaptatif minimal est présenté. Il se concentre sur l'efficacité de la communication intra-couche et inter-couche, prenant en compte la congestion des voisins à deux sauts lors de la sélection du chemin de transmission des données. Il compare le taux de tampon libre des chemins minimaux disponibles et choisit le chemin le moins encombré.

Un algorithme de routage tolérant aux pannes utilise un mécanisme de détection de pannes hors ligne pendant la phase de fabrication du réseau [107]. Il est globalement conscient de la congestion et sélectionne le chemin le moins encombré pour acheminer les paquets, privilégiant le routage minimal mais pouvant utiliser des routes non minimales en cas de liaisons défectueuses. Dans [108], L'algorithme présenté est appliqué dans un NoC 3D partiellement connecté verticalement. Les paquets sont d'abord acheminés vers la liaison verticale disponible la plus proche, puis vers la couche de destination. Dans [109], l'algorithme collecte des informations de congestion globale dans chaque couche du réseau. Il achemine les paquets vers les régions à faible trafic pour éviter la congestion. Les informations de congestion sont propagées via les en-têtes des paquets et des modèles de virage, assurant ainsi l'absence de blocage dans le routage.

Les méta-heuristiques sont utilisées comme approches intelligentes pour des algorithmes de routage efficaces. L'algorithme basé sur la technique d'optimisation de colonie de fourmis, est utilisé pour résoudre le problème de routage dans les réseaux de télécommunication, y compris les NoC 3D [110]. Les résultats obtenus sont supérieurs à ceux des algorithmes de routage déterministes et minimaux tels que DOR.

Dans [111], Silva Junior *et al* utilisent l'optimisation par ACO pour adapter le protocole du Système de fourmis élitiste, initialement présenté pour le routage dans un NoC 2D, afin de résoudre les problèmes de congestion sur une topologie en 3D. Dans leur approche algorithmique, ils utilisent plusieurs colonies de fourmis en interdépendance, permettant ainsi une transmission simultanée de paquets et une réduction de la latence de transmission.

3.4 Discussion

En effet, l'optimisation des réseaux sur puce présente généralement une séparation entre l'affectation et le placement, avec peu de travaux de recherche adoptant une approche intégrée où ces deux phases sont traitées conjointement. Dans certains travaux, l'affectation est réalisée en premier lieu, puis les résultats sont utilisés pour guider le processus de placement multi-objectif. Cette approche séparée permet de réduire l'espace de recherche en se concentrant sur les tâches déjà affectées et en optimisant différents objectifs tels que l'énergie, la performance et la latence. Cependant, certaines recherches récentes cherchent à résoudre le problème de manière intégrée, en combinant l'affectation et le placement dans

un seul processus d'optimisation. Ces approches intégrées visent à trouver des solutions globales et potentiellement plus efficaces en considérant simultanément les deux phases du problème. Cela rend le problème plus complexe et nécessite l'utilisation de méthodes avancées pour explorer l'espace de recherche étendu. Bien que peu de travaux de recherche aient abordé l'affectation et le placement de manière intégrée, la plupart des autres travaux adoptent une approche séparée où l'affectation est effectuée en amont et les résultats sont ensuite utilisés dans le contexte du placement multi-objectif.

En ce qui concerne le routage, il s'agit de trouver le chemin optimal pour acheminer les informations entre les différentes tâches qui sont affectées à des nœuds spécifiques. Le routage doit tenir compte de la topologie du NoC, des contraintes de communication, de la congestion éventuelle et des objectifs de performance. Dans la littérature, plusieurs algorithmes de routage ont été proposés pour le routage des IPs dans un NoC. Cependant, peu de méthodes traitent le problème en utilisant des méta-heuristiques, qui sont des techniques d'optimisation basées sur des approches probabilistes ou heuristiques.

Après avoir analysé les différentes méthodes mentionnées précédemment, nous avons identifié plusieurs observations importantes concernant les approches existantes dans le domaine des réseaux sur puce:

1. Dans certaines références, le placement est abordé comme un problème d'optimisation mono-objectif, ce qui signifie que l'accent est mis sur l'optimisation d'une seule métrique. Cependant, cette approche présente des limitations car l'amélioration d'une métrique peut entraîner une détérioration des autres métriques. Par conséquent, il devient difficile de prendre en compte plusieurs objectifs simultanément.
2. Certains travaux adoptent une approche d'agrégation qui consiste à utiliser une fonction de coût unifiée pour prendre en compte plusieurs objectifs lors du placement. Cependant, cette approche peut être inefficace car elle nécessite un ajustement approprié des poids attribués à chaque objectif, ce qui peut être difficile à réaliser sans une connaissance préalable approfondie du problème.
3. Les études antérieures se sont principalement concentrées sur un ensemble restreint de méta-heuristiques, telles que NSGA, SPEA et MOGA. Par conséquent, il existe une opportunité d'explorer de nouvelles méta-heuristiques afin d'améliorer les performances du placement dans le contexte multi-objectifs.
4. Les études antérieures se sont principalement concentrées sur l'optimisation de deux fonctions de coût, telles que les performances et la consommation énergétique. Cependant, dans un contexte multi-objectif, il est essentiel de considérer un plus grand nombre d'objectifs afin de réaliser une optimisation plus complète.
5. De plus, il existe peu de méthodes qui traitent spécifiquement du problème de routage dans un NoC en 3D. La majorité des travaux de recherche se concentrent sur des architectures NoC 2D, où les IPs sont disposés dans un réseau bidimensionnel. Cependant, avec les avancées technologiques, les architectures NoC 3D, où les IPs

sont empilés sur plusieurs couches, gagnent en popularité. Le routage dans un NoC 3D présente des défis supplémentaires liés à la gestion de la verticalité et à la minimisation des interférences.

6. En outre, il convient de souligner que peu de méthodes abordent le problème du routage dans un NoC 3D utilisant des méta-heuristiques et sous l'angle de l'optimisation multi-objectifs. Les objectifs généralement considérés dans le contexte du routage multi-objectifs comprennent la minimisation de la latence, la maximisation de la bande passante, la réduction de la consommation d'énergie, la minimisation du nombre de sauts, entre autres. En tenant compte de plusieurs objectifs, il est possible de trouver des solutions qui équilibrent ces différents critères et d'explorer un ensemble de solutions optimales potentielles.

Néanmoins, il existe encore des opportunités de recherche pour développer des méthodes d'affectation, de placement et de routage utilisant des méta-heuristiques dans une approche qui traite un, deux ou les trois problèmes à la fois. En se concentrant spécifiquement sur le NoC 3D et en abordant le problème comme un cas d'optimisation multi-objectif, cela permettrait de répondre aux exigences croissantes en termes de performances et d'efficacité des systèmes sur puce.

3.5 Considération finale du chapitre

Dans ce chapitre, une revue de l'état de l'art des méthodes d'optimisation utilisées dans la conception des Réseaux sur Puce a été réalisée. Ces méthodes ont été classées en fonction des différentes phases de conception, notamment l'affectation de tâches aux IPs, le placement des IPs et le routage. Elles ont également été classées en fonction des approches utilisées, telles que les méthodes statiques, heuristiques et méta-heuristiques, ainsi que du nombre d'objectifs pris en compte, qu'il s'agisse d'objectifs uniques ou multiples.

Dans le chapitre 4, une attention particulière sera portée à l'optimisation et à l'optimisation multi-objectifs, avec une présentation détaillée des méthodes d'optimisation utilisées dans le cadre de cette thèse.

Chapitre 4

Optimisation basée sur les méta-heuristiques

L'optimisation occupe toujours une place importante dans le domaine informatique. Elle consiste à explorer un espace de recherche, afin de maximiser ou minimiser une fonction objective. Certains problèmes d'optimisation sont impossibles à résoudre de manière exacte en un temps raisonnable, tels que le placement et l'allocation. C'est pourquoi on a souvent recours à des méthodes approchées qui ne donnent pas forcément les solutions optimales, mais gagnent considérablement en temps. Dans ce chapitre, nous présenterons les techniques d'optimisation en général, puis l'optimisation multi-objectifs.

4.1 Notions sur la complexité

La façon la plus simple de déterminer l'algorithme le plus performant, parmi une multiplicité d'algorithmes, est de comparer le temps d'exécution de chaque algorithme. Cependant, cette comparaison est injuste car elle est affectée par plusieurs facteurs tels que les caractéristiques de la machine dans laquelle les algorithmes sont mis en œuvre. Par conséquent, une autre approche indépendante de toute ressource logicielle et matérielle est nécessaire pour effectuer une comparaison rigoureuse et équitable. La complexité d'un algorithme [112] est définie comme l'estimation du nombre nécessaire d'opérations élémentaires effectuées par cet algorithme pour résoudre un problème donné. La théorie de la complexité [113] consiste à étudier formellement la difficulté d'un problème de décision. Certaines études ont montré qu'il existe une classification fondamentale des problèmes d'optimisation basée sur la complexité des algorithmes utilisés pour les résoudre [114]. Dans la suite, nous allons présenter quelques définitions qui ont une relation avec cette classification.

Un problème de recherche P1 se réduit polynomialement à un problème de recherche P2 par la réduction de Turing, s'il existe un algorithme A1 pour résoudre P1 utilisant comme sous programme un algorithme A2 résolvant P2, de telle sorte que la complexité de A1 est polynomiale, quand on évalue chaque appel de A2 par une constante. La classe P regroupe les problèmes de complexité P, qui sont ceux pouvant être résolus par des algorithmes déterministes avec un coût de calcul croissant de manière polynomiale en fonction de la taille de du problème [114].

La classe NP contient les problèmes de décision qui peuvent être décidés sur une machine non déterministe en temps polynomial. C'est la classe des problèmes qui admettent un algorithme polynomial capable de tester la validité d'une solution du problème. Intuitivement, les problèmes de cette classe ceux qui peuvent être résolus en énumérant l'ensemble de solutions possibles et en les testant à l'aide d'un algorithme polynomial [115]. La classe NP_Complet est un sous ensemble, qui contient les problèmes les plus difficiles. Ce dernier possède la propriété que tout problème dans NP peut être réduit en ce dernier en temps polynomial. Autrement dit, un problème est NP_complet quand tous les problèmes appartenant à NP lui y sont réductibles. La classe NP_Difficile contient les problèmes les plus difficile qu'un problème NP_complet, c'est à dire s'il existe un problème NP_complet se réduisant à ce problème par une réduction de Turing.

4.2 Problème d'optimisation

En termes techniques, l'optimisation est un processus de sélection visant à déterminer le meilleur plan d'action, pour un problème de décision, à partir d'un ensemble d'alternatives disponibles sous la restriction de ressources limitées. Dans ce contexte, une fonction qui est appelée fonction objectif, est minimisée ou maximisée. Selon l'espace de recherche, les problèmes d'optimisation peuvent être divisés en deux catégories: Ceux dont les variables prennent des valeurs réelles et ceux dont les variables prennent des valeurs entières, classés respectivement comme problèmes d'optimisation continue ou combinatoire.

Il existe de nombreux exemples qui entrent dans cette catégorie, trouvés dans la catégorie d'optimisation continue, utilisés dans des applications des domaines tels que l'ingénierie, l'administration, la logistique, l'économie, la biologie ou d'autres sciences. D'autre part, dans les problèmes d'optimisation combinatoire, les meilleures alternatives sont recherchées dans un ensemble de possibilités fini mais très grand. Le processus d'obtention de la meilleure solution consiste à rechercher la meilleure combinaison des valeurs entières de certaines variables. Un exemple courant d'un problème d'optimisation combinatoire c'est le problème du voyageur de commerce. Ce problème consiste à déterminer le chemin le plus court, pour traverser une série de villes. Chaque ville ne doit être visiter qu'une seule fois avant de retourner à la ville d'origine.

Les problèmes d'optimisation peuvent également être classés selon la fonction objectif, qui est la représentation mathématique du critère d'efficacité adopté dans le problème. Ce critère peut être représenté par une seule fonction objectif ou par plusieurs fonctions objectifs. Dans les problèmes d'optimisation avec une seule fonction objectif, les méthodes d'optimisation tentent de trouver une seule solution satisfaisante qui représente le minimum (ou maximum) de cette fonction. Les problèmes définis comme multi objectifs sont ceux qui cherchent à obtenir la meilleure réponse à plus d'un égard, servant plusieurs objectifs conflictuels simultanément. La plupart des travaux de recherche liés à l'optimisation considère un objectif unique alors que la plupart des problèmes d'optimisation de la vie réelle contiennent de multiples objectifs conflictuels à satisfaire. Quelques compromis sont recherchés pour solutionner un problème d'optimisation multi-objectifs, car il n'est pas possible d'obtenir une seule solution optimale aux multiples composantes conflictuelles des fonctions objectifs.

4.3 Méthodes de résolution des problèmes d'optimisation

En général les méthodes de résolution des problèmes d'optimisation combinatoires peuvent être classées, selon la taxonomie proposée par [116] en deux principales catégories: Les méthodes exactes et les méthodes approchées comme montré dans la Figure 4.1.

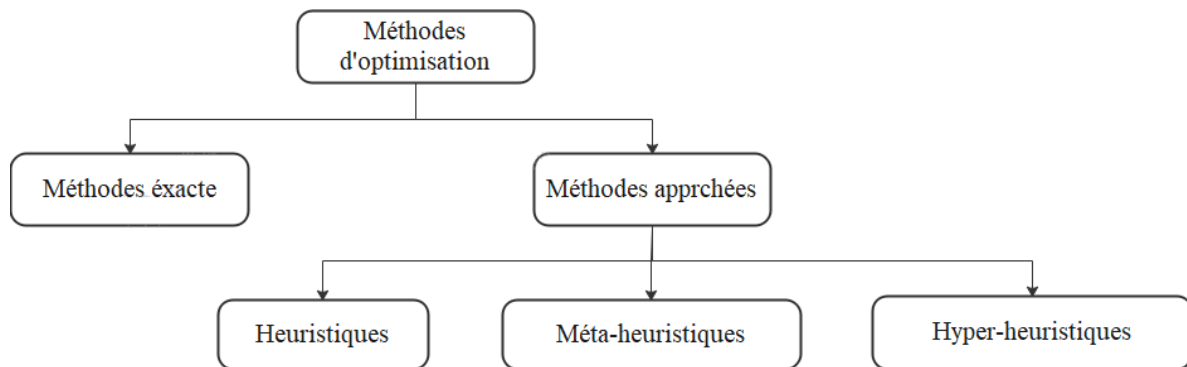


FIGURE 4.1 – Classe des méthodes d'optimisation

Les méthodes de résolution exactes permettent d'obtenir une solution dont l'optimalité est garantie. Par ailleurs, on peut s'intéresser dans certaines situations à des solutions de bonne qualité, sans garantie d'optimalité, mais au profit d'un temps de calcul réduit. Pour cela, on applique des heuristiques, des méta-heuristiques et des hyper-heuristiques. Ces deux dernières exploitent généralement des processus aléatoires dans l'exploration de l'espace de recherche pour faire face à l'explosion combinatoire engendrée par l'énumération implicite des solutions possibles. En plus de cette base stochastique, les méta-heuristiques et hyper-heuristiques sont le plus souvent itératives. Ainsi le même processus de recherche est répété lors de la résolution.

4.3.1 Méthodes exactes

L'intérêt de ces méthodes réside dans le fait qu'elles assurent l'obtention de la solution optimale du problème traité. Ces méthodes font une énumération de toutes les solutions possibles puis choisissent les meilleures d'entre elles. En fait, elles permettent de parcourir la totalité de l'espace de recherche. De cette manière, on doit assurer l'obtention de toutes les solutions ayant le potentiel d'être meilleures que la solution optimale trouvée au cours de la recherche [117]. Cependant, lorsque le problème est de grande taille, les méthodes exactes prennent trop de temps et nécessitent un coût de recherche souvent prohibitif en termes de ressources requises.

Il existe de nombreux algorithmes exacts y compris: La programmation linéaire (LP– *Linear Programming*), la programmation en nombre entier (ILP– *Integer Linear Programming*), la programmation dynamique, l'algorithme du simplexe [118], la programmation dynamique [119], l'algorithme A^* et l'algorithme de séparation et évaluation (*branch and bound*) qui a pour principe de découper (*branch*) l'espace initial de recherche en domaines de plus en plus restreints afin d'isoler l'optimum global (principe de séparation) [120].

4.3.2 Heuristiques

Afin de s'affranchir des problèmes liés aux méthodes exactes qui sont le temps d'exécution excessif et le coût exorbitant, les méthodes approchées sont apparues. Ces méthodes sont plus pratiques pour la résolution des problèmes difficiles, de grande taille et dont on recherche des solutions en un temps raisonnable. Elles donnent de bons résultats en un temps acceptable.

Les méthodes heuristiques sont des méthodes spécifiques à un problème particulier où la recherche est guidée par des astuces dépendantes de ce problème. Ces méthodes nécessitent la connaissance de domaine du problème traité. Elles se basent sur l'expérience et le résultat acquis afin d'améliorer les futures recherches.

À titre d'exemple, une heuristique présentée dans [87], propose une méthode de placement des tâches dans un réseau sur puce en deux dimensions. La stratégie consiste à débiter en priorité avec la tâche ayant le plus grand volume de communication, puis à placer les tâches qui communiquent le plus étroitement avec elle, formant ainsi une structure en forme de losange. Ce processus est réitéré jusqu'à ce que toutes les tâches soient correctement positionnées. Dans une autre heuristique décrite dans [88] et visant le placement des tâches dans un réseau sur puce 2D, la méthode initie à partir du centre et place progressivement les autres tâches, créant ainsi une disposition en forme de spirale. En revanche, dans l'approche présentée dans [89], l'heuristique amorce le placement depuis la partie inférieure et place les tâches en suivant un motif de zigzag.

4.3.3 Méta-heuristiques

Ces méthodes peuvent être vues comme des heuristiques puissantes et évoluées dans la mesure où elles sont généralisables à plusieurs problèmes d'optimisation. En fait, elles sont applicables sur une grande variété de problèmes d'optimisation de différente complexité. En outre, elles permettent de fournir des solutions de bonne qualité, pas nécessairement optimales, en un temps d'exécution raisonnable. L'ensemble des méta-heuristiques proposées dans la littérature, sont partagées en deux classes: Les algorithmes basés sur une solution unique, basés sur une population de solution et les algorithmes basés sur l'intelligence des essaims comme illustré dans la Figure 4.2.

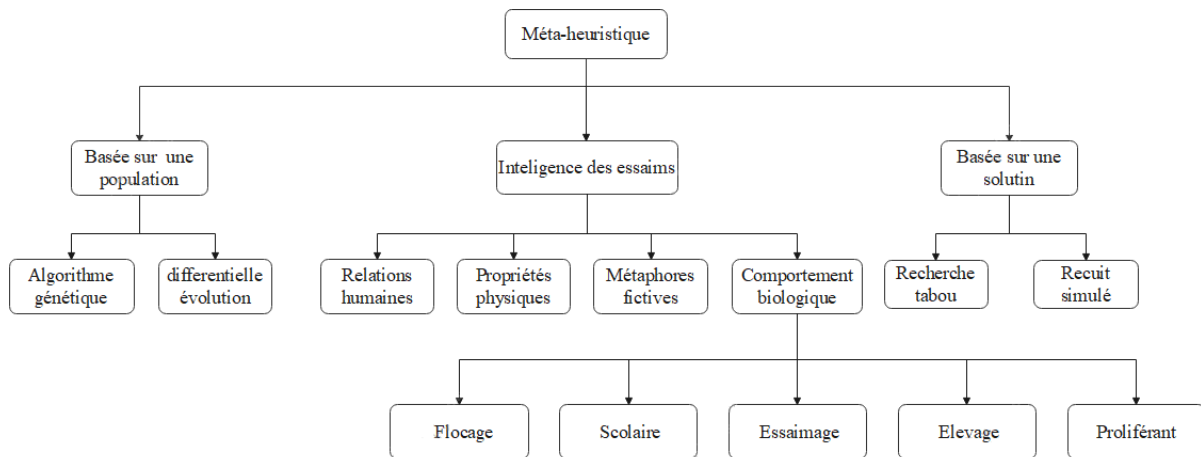


FIGURE 4.2 – Classes des Méta-heuristiques

Le principe des méta-heuristiques basée sur une solution de départ, commence par le choix d'une solution au hasard. Cette solution est améliorée au cours du processus de recherche. Étant donné que ces méthodes ne contiennent qu'une seule solution dans chacune des itérations, elles sont également appelées méthodes à point unique ou à trajectoire. Cette solution est améliorée en utilisant le mécanisme de voisinage. Parmi les méta-heuristiques les plus populaires, nous citons: Le recuit simulé (SA – *Simulated Annealing*) [121] et la recherche tabou (TS – *Tabu Search*) [122].

Contrairement aux méta-heuristiques basées sur une seule solution, celles basées sur une population génèrent un ensemble de solutions multiples appelé population à chaque exécution. Elles peuvent être classées en deux catégories principales: algorithmes évolutionnaires et essaims intelligents.

Les méta-heuristiques basée sur une population comme les algorithmes évolutionnaires imitent les mécanismes de l'évolution biologique. Elles exploitent une population générée aléatoirement pour explorer l'espace de recherche sur plusieurs générations. L'élément central de ces approches réside dans la génération successive, qui applique des opérations comme la sélection, le croisement, la mutation et la reproduction pour améliorer les

solutions candidates. Parmi les algorithmes basés sur l'évolution on cite: L'algorithme génétique (GA – *Genetic Algorithm*) [123], la programmation génétique (GP – *Genetic Programming*) [124] et L'évolution différentielle (DE–*Differential Evolution*) [125].

Les essaims intelligents sont un concept inspiré par le comportement collectif d'un groupe d'agents qui coopèrent, interagissent entre eux et avec leur environnement pour résoudre des problèmes ou accomplir des tâches de manière collective. Les agents de l'essaim doivent interagir efficacement, évaluer leur performance, s'adapter à l'inattendu, maintenir la stabilité collective et ajuster leurs stratégies en fonction de l'environnement. Les essaims intelligents se divisent en quatre catégories [126]: les méta-heuristiques basées sur des principes humains, celles basées sur des principes physiques, celles basées sur des principes fictifs et celles basées sur un comportement biologique.

Les méta-heuristiques inspirées par les relations humaines reposent sur les lois qui régissent les opérations internes et les interactions humaines. Les activités internes des systèmes complexes fonctionnent en harmonie grâce à ces régulations. En ce qui concerne les méta-heuristiques inspirées par les propriétés physiques, elles sont spécifiquement conçues pour simuler les principes fondamentaux de la théorie physique observés dans le monde réel afin d'orienter leur stratégie de recherche vers une valeur optimale. Les méta-heuristiques basées sur des métaphores fictives sont un type de technique qui repose sur une population non naturelle, utilisant des modèles mathématiques et des mesures statistiques pour rechercher une stratégie de l'optimum global.

Quant aux méta-heuristiques basées sur un comportement biologique, elles s'inspirent des comportements d'intelligence collective et de communication dans la nature. Ces comportements se manifestent notamment dans les interactions sociales des populations animales. Ces approches s'appuient sur cinq comportements spécifiques présents dans la nature [127]: Comportement de flockage, scolaire, essaimage, élevage et proliférant.

Le comportement de flockage est observé chez les oiseaux et réduit la consommation d'énergie lors des migrations. Les bancs de poissons manifestent un comportement scolaire pour se défendre, se nourrir et s'accoupler. Les insectes pratiquent l'essaimage pour trouver des partenaires et assurer leur sécurité lors de la migration. En cas de danger, les animaux adoptent un comportement d'élevage, se dirigeant vers l'avant pour se protéger. La prolifération cellulaire résulte de l'équilibre entre divisions et pertes cellulaires, avec la capacité des micro-organismes à anticiper des événements futurs grâce à l'évolution.

À titre d'exemple, parmi les méta-heuristiques les plus populaires on trouve [128]: Optimisation de l'essaim de particules [129], optimisation des colonies de fourmis (ACO – *Ant Colony Optimization*) [130], l'algorithme de colonie d'abeilles artificielles [131], l'algorithme de chauve-souris (*Bat Algorithm*) [132] et l'optimisation des loups gris (*Gray Wolf Optimisation*) [133], l'algorithme d'optimisation des baleines [134] et l'algorithme d'optimisation des sauterelles [135].

4.3.4 Hyper-heuristiques

Les hyper-heuristiques, introduites en 2000 [136], sont une catégorie de méthodes d'optimisation de haut niveau indépendantes du problème en question. Leur particularité réside dans leur capacité à coordonner l'action de différentes méta-heuristiques et algorithmes heuristiques spécifiques au problème. Elles agissent en tant que sélecteurs d'heuristiques. Leur approche pour explorer l'espace des solutions est indirecte. Plutôt que de se lancer directement dans la recherche de solutions, elles recourent à des techniques de résolution de problèmes telles que les méta-heuristiques et les algorithmes heuristiques. L'objectif principal est de prendre une décision éclairée parmi un ensemble d'heuristiques disponibles, en optant pour la méthode la plus adaptée à la résolution du problème spécifique. Cette sélection se fonde sur diverses informations, notamment le temps d'exécution requis et la qualité des solutions obtenues [137].

Il est essentiel de noter que les hyper-heuristiques cherchent à exploiter les forces de différentes méta-heuristiques et méthodes heuristiques, comblant ainsi les lacunes de certaines par les avantages d'autres. Il convient de souligner que la définition des hyper-heuristiques a évolué au fil du temps, notamment avec l'utilisation réussie de la programmation génétique comme l'une de leurs composantes [136]. En fin de compte, les hyper-heuristiques sont des outils puissants qui génèrent automatiquement ou choisissent des méthodes heuristiques (et/ou métaheuristiques) pour résoudre des problèmes complexes.

Plusieurs classifications des hyper-heuristiques peuvent être présentées. Dans [136], l'auteur divise les hyper-heuristiques en deux groupes: Les hyper-heuristiques travaillant avec des heuristiques constructives de niveau inférieur et les hyper-heuristiques utilisant des méthodes perturbatrices. Dans le cas de la première catégorie, le processus commence avec une solution vide et progresse graduellement vers une solution complète. En revanche, pour la deuxième catégorie, le processus commence avec une solution initiale complète et vise à l'améliorer à chaque étape. En outre, une autre classification, détaillée dans [138], identifie quatre classes: Les hyper-heuristiques qui choisissent aléatoirement une heuristique de bas niveau, celles qui utilisent des mécanismes d'apprentissage pour choisir des heuristiques de bas niveau, celles qui intègrent des méta-heuristique comme hyper-heuristiques et enfin celles qui exploitent les hyper-heuristiques gloutonnes.

4.4 Optimisation multi-objectifs

La plupart des problèmes rencontrés dans le monde réel nécessitent l'optimisation de plusieurs objectifs conflictuels simultanément. Chaque objectif doit être optimisé séparément pour que l'optimisation de chacun aboutisse à la solution du problème. Un problème multi-objectifs (PMO) de plusieurs variables de décision, avec m fonctions objectifs est défini par [139]:

$$F \begin{cases} F(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}), \dots, f_m(\mathbf{x})] & m > 1 \\ g_i(\mathbf{x}) \leq 0 & i = 1, \dots, n \\ h_j(\mathbf{x}) = 0 & j = 1, \dots, k \\ \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U & \mathbf{x} \in S \end{cases} \quad (4.1)$$

où $F(\mathbf{x})$ est un vecteur composé de m fonctions objectifs $f_i(x)$, $h_i(\mathbf{x})$ et $g_j(\mathbf{x})$ sont respectivement, les contraintes d'égalité et d'inégalité, \mathbf{x} représente le vecteur des variables de décision, \mathbf{x}^L et \mathbf{x}^U sont respectivement les bornes inférieure et supérieure des variables de décision, S représente l'ensemble des solutions réalisables associées à toutes les contraintes et les espaces de décision. Il est important de noter que lorsque m égale à 1, il s'agit d'un problème d'optimisation mono-objectif.

4.4.1 Concept de Pareto

Dans les problèmes d'optimisation à objectif unique, si s_1 et s_2 sont deux solutions candidates, s_1 est meilleur que s_2 si et seulement si $F(s_1) \leq F(s_2)$. Par contre dans un PMO, il n'y a pas de solution unique, mais un ensemble de solutions où une solution peut être meilleure qu'une autre pour certains objectifs et pire pour ceux qui restent. Par conséquent, la relation d'ordre total lors de la comparaison entre solutions n'est pas utile et une autre relation d'ordre partiel doit être définie.

Pareto a introduit certains concepts pour trouver une solution optimale parmi un ensemble de solutions ayant des objectifs différents [140]. Ces concepts comprennent la dominance, la solution optimale de Pareto, l'ensemble optimal de Pareto et le front optimal de Pareto.

Dans le cas d'un problème de minimisation, une solution $s_1 \in S$, domine une autre solution $s_2 \in S$ si et seulement si s_1 est partiellement inférieur à s_2 , c'est à dire $\forall i \in 1 \dots m : f_1(s_1) \leq f_1(s_2) \wedge \exists i \in 1 \dots m : f_i(s_1) < f_i(s_2)$. Cette relation sera notée $s_1 \prec s_2$. Notons que pour toute paire de solution y et z , un et un seul des cas suivants peut se présenter: y domine z , y est dominé par z , y et z sont deux solutions équivalents mais dominées pour autres solutions et y et z sont équivalents au sens de la dominance, c'est à dire ce sont des solutions qui ne sont dominées par aucune autre solutions.

On dit qu'une solution $s \in S$ est non dominée dans l'espace de solutions S s'il n'existe pas d'autre solution $s' \in S$ tel que $s' \prec s$. L'ensemble des solutions optimales de Pareto (ou encore appelées solutions non dominées) forme l'ensemble de Pareto PS: $PS = \{s \in S \mid \nexists s_1 \in S, s \prec s_1\}$. Le Front optimal de Pareto est la projection de l'ensemble optimal de Pareto (PS) par la fonction objectif F dans l'espace des objectifs Y . $FP = \{F(s) \mid s \in PS\}$ Le front de Pareto optimal contient l'ensemble des solutions correspondant aux variables de décisions qui ne sont dominées par aucune autre solution.

L'intensification et la diversification sont deux activités complémentaires utilisées pour explorer l'espace de recherche. Le premier est l'activité par laquelle l'algorithme de recherche essaie d'explorer autant d'espace de recherche que possible pour éviter de tomber dans les pièges optima locaux. Alors que le second représente l'activité dans laquelle l'algorithme de recherche essaie de développer les meilleures solutions découvertes à travers des approches ciblées [141] [142].

4.4.2 Défis des problèmes multi-objectifs

Le principal défi du PMO réside dans l'optimisation de plusieurs objectifs conflictuels. L'objectif général de la résolution du PMO est de trouver une frontière de solutions très proche du front de Pareto, c'est la convergence vers le vrai front de Pareto [139]. Cependant, une proximité avec le front de Pareto ne suffit pas à évaluer les performances de la méthode de résolution. Il est crucial que les solutions soient bien réparties le long de cette frontière. Ainsi, la diversité des solutions est essentielle. La résolution d'un PMO implique donc deux défis majeurs: La convergence pour trouver le front le plus proche du vrai front de Pareto et la diversité pour obtenir des solutions bien réparties [143].

Ainsi, un certain nombre de techniques ont été proposées pour la conservation de la diversité qui sont:

1. Exécution itérative indépendante: La première technique consiste à exécuter l'algorithme de manière itérative. Cependant, dans la plupart des applications réelles, les solutions optimales n'ont pas la même probabilité d'être trouvées. Il est donc nécessaire d'effectuer un nombre beaucoup plus élevé d'exécutions indépendantes. Une approche parallèle en utilisant des populations indépendantes peut être bénéfique pour cette raison.
2. Nichage séquentiel: C'est une méthode qui permet de localiser plusieurs niches de manière séquentielle en utilisant une exécution itérative de l'algorithme. Le principe central de cette méthode implique la modification de la fonction d'évaluation à chaque fois qu'un optimum est détecté et ce, afin de le désavantager dans les phases ultérieures de la recherche. Cependant, il convient de noter que cette approche présente l'inconvénient de perturber la structure originale du problème [144].
3. Fonction de partage (*Sharing*): La méthode de partage implique le calcul d'une valeur de partage pour chaque solution de la population, en fonction de sa proximité avec les autres solutions. Lorsqu'une solution est proche d'autres solutions, sa valeur de partage diminue, et vice versa. En conséquence, les solutions qui se regroupent dans l'espace des objectifs se voient attribuer des valeurs de fitness réduites. Cela entraîne une diminution de leur compétitivité lors de la sélection pour la génération suivante, ce qui favorise une distribution plus équilibrée des solutions sur le front de Pareto [145].

4. Agglomération (*Crowding*): Il vise à maintenir la diversité des solutions sur le front de Pareto en attribuant une valeur de distance d'agglomération à chaque solution. Les solutions plus éloignées de leurs voisins dans l'espace des objectifs sont favorisées lors de la sélection, garantissant ainsi leur répartition homogène et la préservation de la diversité. Cette méthode permet d'explorer un large éventail de solutions de compromis plutôt que de se concentrer uniquement sur les solutions optimales [123].
5. Restriction de voisinage: Pour maintenir la diversité au sein d'une population, la restriction de voisinage est utilisée. L'idée consiste à permettre la reproduction entre deux individus s'ils sont similaires, ce qui conduit à la formation de différentes espèces (*mating groups*) au sein de la population [146]. Cependant, d'autres travaux ont adopté une approche opposée en interdisant la reproduction entre individus similaires afin de prévenir l'inceste.

4.4.3 Classification des méthodes d'optimisation multi-objectifs

En raison de l'intérêt croissant pour l'optimisation multi-objectifs et de son utilité dans le monde réel, de nombreuses méthodes de résolution ont été proposées. Ces méthodes peuvent être regroupées en deux types de classifications. La première classification adopte une perspective utilisateur (ou décideur), tandis que la seconde classification adopte une perspective conceptuelle (ou du concepteur).

Dans le point de vue décideur, les méthodes d'optimisation multi-objectifs sont classées en fonction de leur utilisation souhaitée. Le décideur a la possibilité d'intervenir à différentes étapes du processus d'optimisation et de choisir la méthode qui correspond le mieux à ses préférences et objectifs. Il existe trois schémas possibles [147] [148] qui décrivent la relation entre le décideur et le concepteur en fonction du niveau d'intervention du décideur et de sa coopération avec la méthode d'optimisation, comme illustré dans la Figure 4.3.

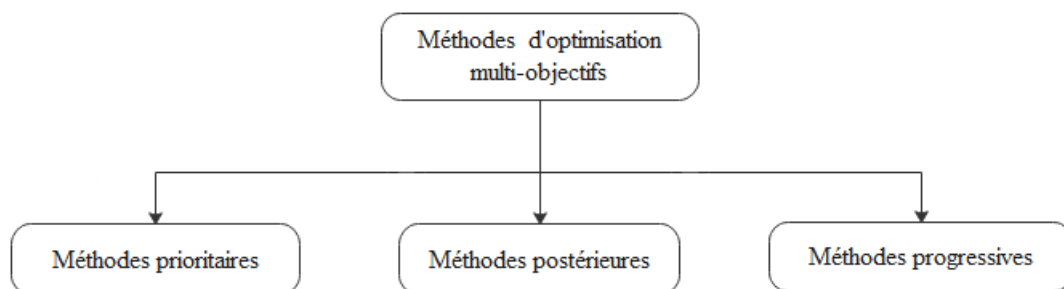


FIGURE 4.3 – Méthodes d'optimisation multi-objectifs (point de vue décideur)

Les méthodes prioritaires visent à combiner les différentes fonctions objectifs en une seule fonction d'utilité, basée sur les préférences du décideur, afin de transformer le PMO en un problème mono-objectif. Les méthodes à priori ou (prioritaires) permettent

à l'utilisateur de spécifier ses préférences en termes de buts ou d'importance relative des différents objectifs [149]. Dans ce cas, une connaissance préalable du problème et du poids de chaque objectif est nécessaire.

Les méthodes postérieures Ces méthodes présentent un ensemble de solutions parmi lesquelles le décideur peut choisir une configuration préférée parmi les alternatives proposées [150]. Pour les méthodes progressives ou interactives, la présence du décideur est indispensable lors du processus d'optimisation afin d'informer progressivement le solveur. Ces méthodes permettent au décideur d'exprimer ses préférences lors du processus d'optimisation [151] qui seront prises en considération afin que les solutions les plus adéquates puissent être obtenues.

Dans le point de vue concepteur, les méthodes sont classées en fonction du mode de traitement des fonctions objectifs. Nous distinguons les méthodes de transformation du problème en un problème mono-objectif, les méthodes non-Pareto et les méthodes basées sur la dominance de Pareto, comme illustré dans la Figure 4.4 [116].

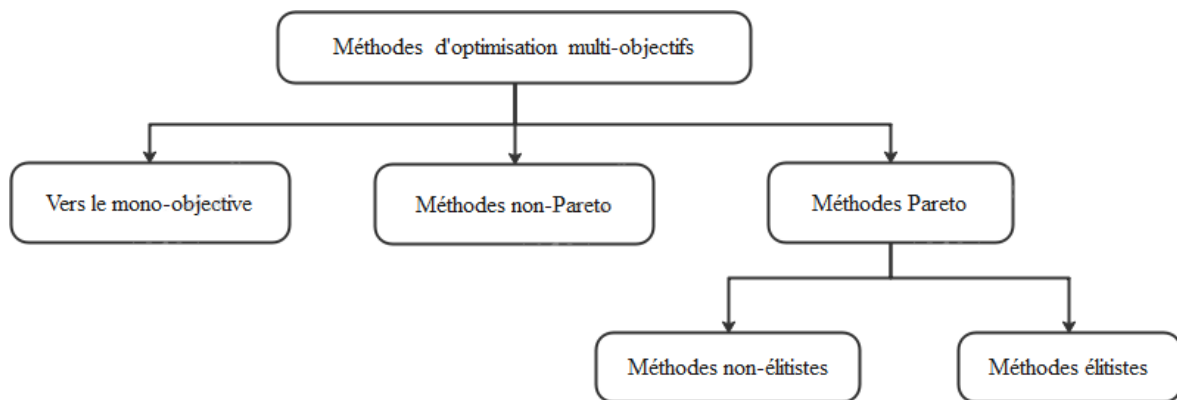


FIGURE 4.4 – Méthodes d'optimisation multi-objectifs (point de vue concepteur)

Les approches basées sur la transformation du problème vers le mono-objectif convertissent les PMOs en problèmes mono-objectif. Plusieurs méthodes ont été développées, notamment les techniques d'agrégation [152], la programmation par objectif et les méthodes ϵ contraintes [153]. Cette transformation requiert une connaissance préalable du problème en question. Bien que l'optimisation d'un problème mono-objectif puisse garantir l'optimalité Pareto de la solution obtenue, il est souvent préférable dans des contextes réels de disposer de plusieurs alternatives plutôt qu'une seule solution. Cependant, ces méthodes ne sont pas efficaces lorsque certains objectifs sont bruités ou que les données sont incertaines. De plus, ces approches sont sensibles aux coefficients de pondération des objectifs, aux contraintes ou aux vecteurs de référence, ce qui influence fortement les solutions obtenues. Dans diverses situations, des paramètres différents peuvent être nécessaires, ce qui entraîne la résolution répétée du problème. Cette répétition peut engendrer des coûts substantiels,

surtout lorsque chaque objectif doit être optimisé indépendamment. Ainsi, pour obtenir diverses solutions pseudo-optimales, il est souvent nécessaire d'exécuter les algorithmes de manière itérative.

Les méthodes non-Pareto visent à traiter les objectifs distinctement. La recherche se déroule en abordant individuellement les objectifs non comparables. Parmi ces approches, on retrouve la sélection lexicographique et la sélection parallèle. Néanmoins, une limitation significative de ces méthodes est leur tendance à générer des solutions qui sont considérablement optimisées pour certains objectifs tout en négligeant d'autres objectifs. Cela conduit à la sous-représentation des solutions de compromis [148].

Les méthodes basées sur l'optimalité de Pareto utilisent la relation de dominance pour évaluer les solutions, principalement en s'appuyant sur des algorithmes génétiques [154]. Deux catégories de méthodes Pareto se distinguent: Les méthodes non-élitistes et les méthodes élitistes.

Les méthodes non-élitistes ne conservent pas les solutions optimales de Pareto trouvées lors des itérations, ce qui peut entraîner la perte de solutions pertinentes et ralentir la convergence vers le front de Pareto. Ces méthodes utilisent le rang de dominance pour comparer les solutions et intègrent le concept de partage [145], évitant ainsi la concentration des individus de la population autour d'un même point. Parmi elles, on trouve l'algorithme génétique multi-objectifs (MOGA–*Multi-Objective Genetic Algorithm*) [155], l'algorithme génétique de tri non dominé (NSGA–*Non-dominated Sorting Genetic Algorithm*) [156] et l'algorithme génétique de Niche de Pareto (NPGA–*Niched Pareto Genetic Algorithm*) [157].

Les méthodes élitistes visent à remédier à l'inconvénient majeur des méthodes non-élitistes, qui est la lenteur dans la détermination des solutions du front de Pareto. Ces méthodes préservent les solutions non dominées dans une archive externe, permettant de sauvegarder les meilleures solutions rencontrées lors de l'optimisation. L'approche élitiste, basée sur le concept d'élitisme, peut significativement améliorer les performances de l'algorithme. Cependant, lorsque le nombre de solutions Pareto excède la taille de l'archive, des stratégies de mise à jour sont généralement employées pour gérer et maintenir la diversité.

4.5 Métriques de performances

Différentes mesures de performance sont utilisées pour évaluer l'efficacité des méthodes d'optimisation multi-objectifs [116]. Ces mesures sont basées sur trois aspects clés: La cardinalité, qui vérifie la couverture de l'ensemble des solutions non-dominées du vrai front de Pareto; la convergence, qui évalue la proximité du front obtenu au vrai front de Pareto; et la distribution et la répartition des solutions, qui garantissent une bonne dispersion

et diversité des solutions. L'utilisation d'indicateurs unaires qui mesurent ces aspects individuellement ainsi que d'une façon binaire facilitent la comparaison directe entre deux frontières Pareto. Dans la suite, les indicateurs de qualité sont présentés en fonction de leur objectif de performance, tels que la convergence et la diversité. Ces indicateurs sont basés sur des mesures de cardinalité, de distance ou de volume. L'utilisation d'une combinaison d'indicateurs de qualité dans chaque catégorie est recommandée pour une évaluation complète des méthodes d'optimisation multi-objectifs [116].

4.5.1 Indicateurs basés sur la convergence

Les métriques de convergence évaluent l'efficacité des solutions en termes de proximité avec le front de Pareto optimal [116]. De nombreux indicateurs de qualité basés sur la convergence ont été proposés dans la littérature par exemple, la contribution et la distance générationnelle.

L'indicateur de contribution est basé sur la cardinalité, qui est utilisé pour la convergence [158]. La contribution d'un front de pareto PO_1 par rapport à une autre PO_2 est le rapport des solutions non dominées produites par PO_1 dans $PO^* = PO_1 \cup PO_2$. PO est l'ensemble des solutions dans $PO_1 \cap PO_2$. Dans ce contexte, W_1 représentent les solutions dans PO_1 qui dominent certaines solutions de PO_2 , tandis que W_2 représentent les solutions dans PO_2 qui dominent certaines solutions de PO_1 . De même, L_1 englobent les solutions de PO_1 qui sont dominées par PO_2 , alors que L_2 englobent les solutions de PO_2 qui sont dominées par PO_1 . Quant aux ensembles N_1 et N_2 , ils rassemblent les solutions non comparables de PO_1 et PO_2 , définies par $N_i = PO_i / (PO \cup W_i \cup L_i)$. Le calcul de la contribution s'effectue selon les indications fournies dans l'équation 4.2:

$$Cont(PO_1/PO_2) = \frac{\frac{||PO||}{2} + ||W_1|| + ||N_1||}{||PO^*||} \quad (4.2)$$

où $||PO^*|| = ||PO|| + ||W_1|| + ||N_1|| + ||W_2|| + ||N_2||$

La distance générationnelle (GD-*Generational Distance*) est une métrique unaire de convergence qui permet de calculer la distance moyenne entre l'ensemble approché PO et un ensemble de référence R [159]. Qui est généralement représenté par le front de Pareto exact PO^* . A une itération t donnée, la distance entre les deux ensembles est moyennée sur les distances minimales par paires se calcule par l'équation 4.3:

$$GD = \frac{\sqrt{\sum_{i=1}^{|PO|} d_i^2}}{|PO|} \quad (4.3)$$

où $|PO|$ représente le nombre de solution non dominée du front Pareto PO , la variable d est la distance euclidienne entre le i ème point de $|PO|$ et le point le plus proche appartenant au front Pareto optimal PO^* . Le front ayant la plus petite GD est le meilleur.

4.5.2 Indicateurs basés sur la diversité

Les indicateurs de diversité mesurent l'uniformité de distribution des solutions obtenues en termes de dispersion et d'extension. En général, la diversité est recherchée dans l'espace objectif. Les mesures de diversité sont généralement non monotones et impliquent un coût de calcul plus élevé par rapport aux indicateurs de convergence [116]. Dans ces classes de métriques, on peut trouver l'espacement et l'entropie.

La métrique d'espacement (Sp), qui est une métrique unaire de diversité [160], est couramment utilisée pour estimer dans quelle mesure les solutions non dominées obtenues sont uniformément distribuées. La métrique Sp est définie dans l'équation 4.4:

$$Sp = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (\bar{d} - d_i)^2} \quad (4.4)$$

où d_i est défini dans l'équation 4.5:

$$d_i = \min_i \left(\sum_{k=1}^m |f_m^i - f_m^j| \right) i, j = 1 \dots n. \quad (4.5)$$

où n est le nombre de solutions dans le front obtenu par l'algorithme, m est le nombre de fonctions objectifs et \bar{d} est la moyenne de tous les d_i qui se calcule dans l'équation 4.6:

$$\bar{d} = \frac{1}{n} \sum_{j=1}^n d_j \quad (4.6)$$

La métrique d'entropie mesure la diversité d'un front Pareto PO_1 par rapport à un front PO_2 [161]. L'espace objectif à n dimensions, où n représente le nombre d'objectifs, est partitionné. Pour chaque partition i qui contient au moins un point non dominé de $PO^* = PO_1 \cup PO_2$, on calcule le nombre de points n_i appartenant à PO_1 . L'entropie $E(PO_1/PO_2)$ est calculée dans l'équation 4.7:

$$E(PO_1/PO_2) = \frac{-1}{\log(c)} \sum_{i=1}^c \frac{n_i}{c} \log \frac{n_i}{c} \quad (4.7)$$

où c représente le nombre de partitions de PO^* . Autant l'entropie est proche de 1, meilleure est la diversité de PO_1 par rapport à PO_2 .

4.5.3 Indicateurs hybrides

Certains indicateurs de qualité combinent à la fois la convergence et la diversité, tels que l'indicateur d'hypervolume [162], qui est une métrique unaire qui mesure la taille de l'espace des objectifs couverts par un front de pareto donné PO . C'est la surface où le volume dans le cas de trois objectifs totale de l'espace dominé par les solutions de l'ensemble S . Pour un problème de minimisation, l'hypervolume est à maximiser. Un point de référence est nécessaire pour le calcul de l'hypervolume comme illustré dans l'équation 4.8.

$$HV = \cup a_i | \forall \mathbf{x}_i \in PO \quad (4.8)$$

où a_i est l'hypervolume formé par la solution \mathbf{x}_i et le point de référence. L'inconvénient de cette métrique est que le résultat dépend du point de référence choisi. Ainsi, la difficulté réside dans le choix de ce point qui doit en particulier être dominé par toutes les solutions des fronts. Généralement, la pire solution possible est choisie comme point de référence [163].

4.6 Considération finale du chapitre

L'optimisation multi-objectifs est sans doute un axe de recherche primordial pour les scientifiques et les ingénieurs, non seulement à cause de la nature multi-critère de la plupart des problèmes réels, mais également parce que de nombreuses questions restent ouvertes dans ce domaine. Les approches de résolution de PMO proposées dans ce chapitre ont montré les limites des méthodes basées sur la transformation d'un PMO en un problème mono-objectif. Malgré ces limites, cette celle ci reste largement utilisée en pratique.

Dans le chapitre 5, nous allons exposer, tout particulièrement, les algorithmes: PSO et DE utilisés dans l'optimisation mono-objectif, utilisé pour réaliser le routage ainsi que MOPSO et DEMO adapté aux problèmes multi-objectifs, exploiter pour résoudre les étapes d'affectation et du placement.

Chapitre 5

Méta-heuristiques exploitées

Les problèmes d’optimisation mono et multi-objectifs sont difficiles à résoudre et nécessitent des méthodes performantes. Bien qu’il existe de nombreuses méthodes d’optimisation disponibles, il n’y a pas encore de méthode puissante capable de surmonter les difficultés des problèmes multi-objectifs. Ce domaine reste ouvert à de nouvelles idées et approches. Dans cette section, nous décrivons les algorithmes d’optimisation utilisés dans notre thèse, à savoir l’algorithme DE (*Differential Evolution*) de la classe évolutionnaire et une méthode inspirée des essaims, le PSO (*Particle Swarm Optimization*). Ces deux méthodes ont démontré leur efficacité et sont largement utilisées dans la résolution des problèmes d’optimisation.

5.1 Choix des méthodes exploitées

Le domaine de l’optimisation multi-objectifs est riche en méthodes proposées dans la littérature, rendant le choix de l’algorithme le plus adapté une tâche complexe. Ce choix repose sur des critères précis, notamment la capacité à maintenir une bonne diversité des solutions et à converger efficacement vers le front de Pareto. Dans le cadre de nos travaux, nous avons étudié différents algorithmes, en nous concentrant sur les problèmes d’optimisation multi-objectifs. À ce titre, nous avons également analysé la quantité d’articles publiés qui mettent en œuvre ces méthodes.

L’algorithme MOPSO se distingue par sa capacité exceptionnelle à explorer efficacement l’espace de recherche tout en garantissant une excellente diversité des solutions. En utilisant des mécanismes pour la sélection des leaders, MOPSO surpasse souvent des algorithmes tels que NSGA-II et SPEA2, assurant une répartition homogène des solutions sur le front de Pareto.

Par ailleurs, l'algorithme DEMO excelle dans la convergence vers des solutions Pareto-optimales grâce à la robustesse de ses opérateurs de mutation et de recombinaison différentielles. Il est particulièrement efficace pour trouver des solutions globales dans des problèmes d'optimisation complexes, où il rivalise et surpasse fréquemment d'autres algorithmes en termes de proximité au front de Pareto.

Ces deux algorithmes se démarquent par leur capacité à converger rapidement, même dans des espaces de recherche très larges, tout en maintenant une diversité remarquable des solutions. De plus, leur simplicité d'implémentation en fait des choix privilégiés pour de nombreuses applications pratiques.

Notre revue bibliographique couvrant les publications entre 2017 et 2023, présentée dans la Table 5.1, a confirmé la forte adoption de MOPSO et DEMO, démontrant leur pertinence et leur robustesse en optimisation multi-objectifs. Ces constats ont motivé notre décision d'utiliser ces deux algorithmes pour les trois phases de conception présentées dans cette thèse, afin d'exploiter leurs forces respectives et garantir des solutions optimales adaptées aux exigences de notre problématique.

TABLE 5.1 – Statistiques sur les travaux publiés selon Google Scholar, IEEE et SCOPUS

Algorithm	Google Scholar	IEEE	SCOPUS
NSGA	19600	1640	13808
SPEA	48800	124	3295
MOPSO	84800	2921	26285
DEMO	20500	659	21436

5.2 Essaims de Particules

Les essaims de particules forment une approche d'intelligence collective permettant de résoudre des problèmes d'optimisation. Les PSO sont à l'origine proposées par J. Kennedy et R. Eberhart [129] comme une simulation des animaux vivant en groupe. L'idée directrice de cette méthode est de simuler le comportement collectif des oiseaux à l'intérieur d'une nuée. Leur capacité à voler de façon synchrone et leur aptitude à changer brusquement de direction, tout en restant en une formation optimale. Les PSO sont devenu si populaire à cause de deux aspects clés [164]. Premièrement, il est apprécié pour sa simplicité. Son algorithme principal repose sur des opérations mathématiques simples et utilise un seul opérateur lié au déplacements pour générer de nouvelles solutions. Deuxièmement, les différentes variantes du PSO se sont révélées très efficaces dans de nombreux domaines d'application. Elles ont réussi à produire des solutions de haute qualité tout en réduisant considérablement le temps de calcul requis.

Dans le contexte du PSO, un essaim est une population composée de particules. Chaque particule représente une solution potentielle au problème à résoudre et est définie par sa position actuelle. Ces particules se déplacent en suivant des vitesses paramétrées par des poids qui régulent leur impact. Deux facteurs, le cognitif et le social, influencent l'attraction des particules. Le facteur cognitif est lié à l'attraction d'une particule envers son propre succès, tandis que le facteur social concerne l'attraction d'une particule envers le succès du groupe. Ces deux facteurs jouent un rôle crucial dans la dynamique du PSO, car elles permettent aux particules d'être influencées par leur propre performance et celle de leurs voisines. Chaque particule possède une meilleure position locale, représentant la position optimale qu'elle a atteinte au cours de ses recherches. De plus, il existe une meilleure position globale qui est la meilleure particule de l'ensemble de l'essaim [165].

5.2.1 Voisinage

Dans le contexte des topologies PSO, le terme "voisinage" fait référence à la description de la relation et de l'interaction entre les particules. Ces configurations influent sur la propagation des informations au sein de l'essaim de particules, ce qui a un impact direct sur la capacité d'optimisation et la convergence de l'essaim. On peut classer les topologies PSO en deux catégories distinctes : les topologies statiques et les topologies dynamiques. Les particules peuvent être reliées les une aux autres dans n'importe quel type de topologie de voisinage. Ces types sont représentés sous forme de graphe. Dans ce paragraphe, nous représentons la liste des topologies typiques utilisées dans le PSO selon la classification de [166] [167] [168]:

1. Étoile: Chaque particule est directement connectée à toutes les autres particules de l'essaim, formant ainsi un voisinage complet. Cette configuration permet un échange rapide d'informations entre les particules, favorisant une convergence rapide de l'algorithme, comme le montre la Figure 5.1(a).
2. Anneau: chaque particule est reliée directement à ses m voisines immédiates. Lorsque m est égale à deux, chaque particule a seulement deux voisines, comme le montre la Figure 5.2(a). Cette topologie permet d'explorer simultanément différentes régions de l'espace de recherche.
3. Centrale: une particule centrale est connectée à toutes les autres. Seule cette particule ajuste sa position par rapport aux autres. Si cela provoque une amélioration, l'information est généralisée comme illustré dans la Figure 5.2(b).
4. Von-Neumann: c'est une structure de grille, comme le montre la figure 5.1(d), chaque particule est connectée à ses m voisines: haut, bas, gauche et droite.

5. Clusters: Il existe m sous-groupes dans la topologie en clusters, comme le montre la Figure 5.2. Les particules de chaque sous-groupe diffusent des informations sous la topologie globale. Ces sous-groupes peuvent avoir la même topologie de communication (étoile, anneaux), ou une hybridation des topologies, où chaque cluster utilise une topologie de communication distincte des autres.

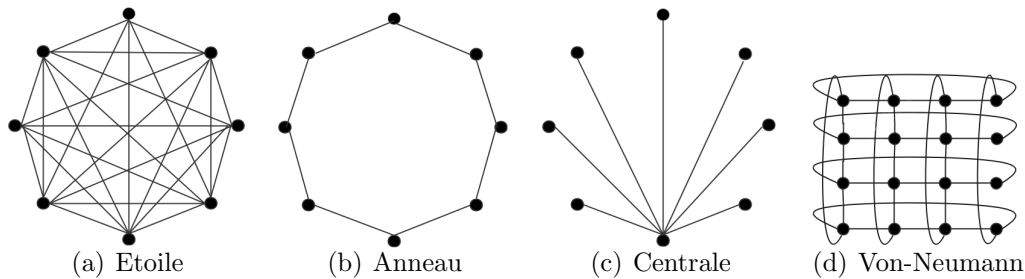


FIGURE 5.1 – Topologie de voisinage

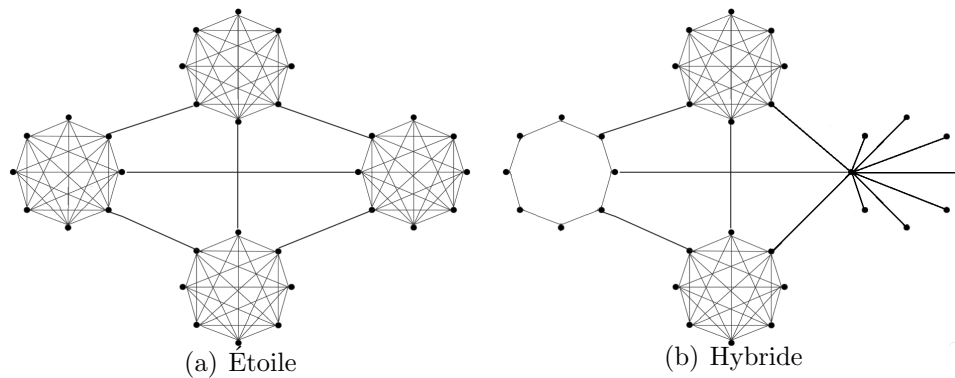


FIGURE 5.2 – Topologie en clusters

Après avoir présenté les différentes topologies, il est observé que la topologie en étoile est largement adoptée dans la littérature. Dans notre thèse, nous avons choisi d'utiliser la topologie en étoile pour favoriser un échange rapide d'informations entre toutes les particules de l'essaim.

5.2.2 Étapes des essaims de particules

Les essaims de particules traversent plusieurs étapes essentielles, chacune apportant une contribution cruciale à la recherche de la solution optimale. Ces étapes sont:

1. Le déplacement: Dans les PSO, les particules se déplacent ensemble en balayant l'espace de recherche. Les changements de directions sont basés sur le partage social de l'information entre les particules de l'essaim. Cette caractéristique lui offre un

avantage évolutionnaire [169]. À chaque itération, la nouvelle position d'une particule est calculée en fonction de sa vitesse selon l'équation 5.1:

$$x_i(t+1) = x_i(t) + v_i(t+1). \quad (5.1)$$

À chaque itération, une nouvelle vitesse est calculée selon l'équation 5.2:

$$v_i(t+1) = w.v_i(t) + c_1.r_1(t+1)(p_i(t) - x_i(t)) + c_2.r_2(t+1)(g(t) - x_i(t)). \quad (5.2)$$

La vitesse reflète l'échange social de l'information à l'intérieur de l'essaim. Elle est, en général, basée sur trois critères:

- (a) Mouvement actuel: Il est basé sur la vitesse précédente $w.v_i(t)$, son intérêt est d'empêcher la particule de subir un changement radical de direction, où w est le coefficient d'inertie.
- (b) Influence personnelle: Le terme cognitif, $c_1.r_1(t)(p_i(t) - x_i(t))$, quantifie la performance de la particule i par rapport à sa performance précédente. Cette composante est également connue sous le nom de nostalgie de la particule [170], où c_1 est le coefficient cognitif et $p_i(t)$ est la meilleure position trouvée par la particule i jusqu'à présent.
- (c) Influence sociale: La composante sociale, $c_2.r_2(t)(g(t) - x_i(t))$, quantifie la performance de la particule i par rapport à la performance de l'essaim de particules. L'effet de cette influence est d'attirer la particule vers la meilleure position trouvée par l'essaim de particules $g(t)$. Le poids c_2 est le coefficient social.

Le coefficient cognitif c_1 et le coefficient social c_2 , donnent une meilleure performance lorsqu'ils sont équilibrés [170]. Notons que $r_1(t)$ et $r_2(t)$ sont des nombres aléatoires sélectionnés à chaque itération t dans la plage de $[0 \dots 1]$. Ils sont utilisé pour caractériser la nature stochastique des contributions cognitives et sociales.

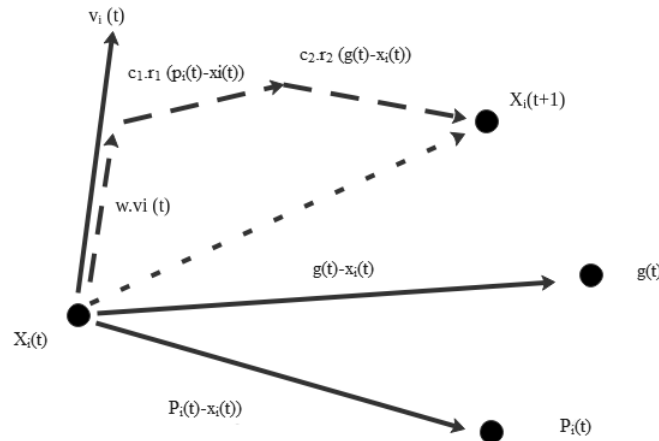


FIGURE 5.3 – Diagramme de déplacement d'une particule

2. Mise à jour de l'essaim: La mise à jour des particules de l'essaim se fait par l'actualisation de la position du record personnel p_i et celui du globale $leader\ g(t)$. Dans ce sens, chaque particule compare l'évaluation de sa position actuelle avec celle de son score personnel. Si la valeur courante est meilleure, c'est-à-dire si la particule a obtenu un nouveau score supérieur, alors la valeur de p_i sera mise à jour avec la valeur courante. À ce stade, la particule devra choisir un nouveau guide (leader). Un bon choix des guides peut aider l'algorithme à éviter la convergence vers des optima locaux.
3. Critère d'arrêt: L'amélioration des performances d'une méthode d'optimisation implique généralement l'obtention de meilleurs résultats dans un temps de calcul réduit. Le choix du critère d'arrêt est essentiel pour l'efficacité de l'algorithme d'optimisation. Trois types de critères d'arrêt de base sont couramment utilisés [171]: L'arrêt après un nombre fixe d'itérations, l'arrêt dès que la valeur de performance de la meilleure particule atteint un certain seuil et l'arrêt après un certain nombre d'itérations sans amélioration significative des résultats.

Le premier critère d'arrêt est souvent privilégié, mais il est possible de combiner plusieurs critères. Par exemple, la condition du deuxième critère peut être testée en premier. Si, après un certain nombre de générations, la valeur attendue de la performance n'est toujours pas atteinte, l'algorithme s'arrête.

L'algorithme PSO, qui regroupe les étapes présentées ci-dessus, est illustré dans Algorithme 1. Les paramètres qui interviennent dans cet algorithme sont: La dimension du problème D , le nombre de particules N , la vitesse V et les valeurs des coefficients c_1 et c_2 . Afin de raffiner la recherche, on peut trouver d'autres variantes dont les relations de mise à jour de l'essaim, sont modifiées.

5.2.3 PSO pour les problèmes multi-objectifs

L'algorithme PSO n'est pas directement applicable à un PMO. Dans un problème d'optimisation à objectif unique, chaque particule utilise une meilleure solution pour mettre à jour sa position, ce qui est clairement défini. Cependant, dans les problèmes d'optimisation multi-objectifs, chaque particule peut avoir plusieurs meilleures solutions, appelées leaders, parmi lesquelles une seule est choisie pour mettre à jour la position de la particule. Ces leaders sont généralement stockés dans une mémoire distincte de celle de l'essaim, appelée archive externe où sont conservées les solutions non dominées trouvées jusqu'à présent. Ces solutions sont utilisées comme leaders lors de la mise à jour des positions des particules. En outre, le contenu de l'archive externe est souvent considéré comme le résultat final de l'algorithme.

Algorithme 1 Les étapes principales de PSO

```
générer aléatoirement les position de chaque particule de l'essaim
évaluer tous les particules l'essaim utilisant la fonction objectif
identifier la meilleur solution  $g$ 
initialiser la vitesse et best local de chaque particule  $p_i$ , vitesse
 $t \leftarrow 0$ 
tantque  $t < \text{max d'itération}$  faire
  pour chaque particule  $i$  faire
    actualiser vitesse particule utilisant Équations 5.2
    mise à jour position particule utilisant Équations 5.1
    évaluer la particule utilisant la fonction objective
    si particule est meilleur que  $p_i$  alors
       $p_i \leftarrow \text{particule}$ 
    finsi
    si  $p_i$  est meilleur que  $g$  alors
       $g \leftarrow p_i$ 
    finsi
  fin pour
   $t \leftarrow t + 1$ 
fin tantque
retourner meilleur solution  $g$ 
```

Plusieurs variantes du PSO, pour le problème multi-objectifs (*Multiobjective Particle Swarm Optimization*), ont vu le jour [172]. Parmi ces variantes nous avons l'algorithme (NSPSO–*Non Dominated Sorting Particle Swarm Optimization*), proposé par Li [173], il combine le PSO avec la méthode principale de NSGA II. L'algorithme trie l'ensemble de la population en différents niveaux de non-domination, de sorte que les individus des meilleurs fronts puissent être sélectionnés. De cette façon, le processus de sélection pousse vers le vrai front de Pareto. Une autre version suggéré par Parsopoulos et Vrahatis [174] nommé (VEPSO–*Vector Evaluated PSO*). L'algorithme d'optimisation des essaims de particules à évaluation vectorielle est basé sur le concept de l'algorithme génétique à évaluation vectorielle. Dans cet algorithme, deux ou plusieurs essaims sont utilisés afin de rechercher l'hyperespace du problème où Chaque essaim est évalué selon l'une des fonctions objectifs. Les informations de chaque essaim sont échangées entre eux. En conséquence, les connaissances provenant d'autres essaims sont utilisées pour guider la trajectoire de chaque particule vers les points optimaux de Pareto. Coello et Lechuga ont également intégré la dominance de Pareto dans l'algorithme PSO, donnant ainsi naissance à l'algorithme (MOPSO–*Multiobjective Particle Swarm Optimization*). Dans ce cas, les solutions non dominées sont stockées dans une population secondaire. La population primaire utilise un meilleur voisinage choisi au hasard parmi cette population secondaire pour mettre à jour les vitesses de ses particules. Afin d'améliorer les capacités d'exploration de l'essaim, les auteurs ont proposé une grille adaptative pour générer des fronts de Pareto et des opérateurs de mutation bien répartis.

L'algorithme (OMOPSO–*Optimized Multiobjective Particle Swarm Optimization*), élaboré par Reyes et Coello Coello dans [175]. Cette version est basée sur la notion de dominance de Pareto avec un facteur d'agglomération (*crowding*) calculé dans l'algorithme 5 pour la sélection des leaders. Cette proposition utilise deux archives externes, une pour stocker les leaders actuellement utilisés et une autre pour stocker les solutions finales. En outre, les auteurs proposent un schéma dans lequel ils subdivisent la population en trois sous ensembles différents. Un opérateur de mutation différent est appliqué à chaque ensemble. Une version de l'algorithme présenté par Nebro *et al* en [176], nommée (SMPSO–*Speed-constrained Multiobjective PSO*), constitue une extension de l'algorithme OMOPSO présenté ci-dessus. La principale distinction de SMPSO par rapport à OMOPSO réside dans l'intégration d'un mécanisme de limitation de vitesse d'une part, et l'introduction d'un opérateur de mutation polynomiale d'autre part.

Dans cette thèse, nous présentons une variante de l'algorithme PSO axée sur une approche multi-objectifs, spécifiquement MOPSO pour l'optimisation des essaims de particules en présence de multiples objectifs. Dans l'algorithme standard du MOPSO, une seule position du meilleur local est définie pour chaque particule, représentant la meilleure position trouvée au cours de l'ensemble du processus de recherche. Notre modification de l'algorithme inclut l'ajout d'une archive locale pour chaque particule, enregistrant les meilleures solutions non dominées découvertes au cours du processus de recherche. Par conséquent, l'archive externe englobe l'ensemble des meilleures solutions non dominées trouvées par l'ensemble des particules. L'utilisation d'une archive locale, plutôt que de se limiter à une solution locale unique lors du processus de mise à jour des particules, apporte une flexibilité remarquable pour explorer de manière plus approfondie l'espace des solutions et favoriser une diversité significative.

Nous introduisons donc deux archives en plus de l'état actuel de l'essaim. Une archive stocke les meilleures solutions trouvées jusqu'à présent au cours du processus de recherche. La deuxième archive conserve l'ensemble des meilleures solutions locales pour chaque particules. Dans le processus de mise à jour de la position de chaque particule, au lieu de conserver une seule solution globale pour l'ensemble des particules, notre modification consiste à ce que chaque particule choisisse une solution globale depuis l'archive externe. De plus, le même principe est appliqué pour la meilleure solution locale. Dans notre thèse, le choix d'une meilleure solution globale et le choix d'une meilleure position locale depuis les archives sont réalisés de manière aléatoire. Il est crucial de noter que l'archive de chaque particule est limitée. Dans ce cas, lorsque l'archive d'une particule est pleine, le remplacement des solutions est effectué de manière aléatoire. En d'autres termes, le processus de mise à jour de l'archive pour chaque particule n'est pas déterministe, mais suit un schéma aléatoire, contribuant ainsi à la diversité et à la variabilité de l'algorithme. Pour préserver la diversité au sein de l'archive globale, la technique de distance d'agglomération est appliquée. L'équation qui permet de calculer la distance d'agglomération est illustré dans l'Équation 5.3.

$$d_i^j = \frac{f_j(x_{i-1}) - f_j(x_{i+1})}{f_j^{max} - f_j^{min}} \quad (5.3)$$

Où i est le numéro de la solution, j est le numéro d'objectif, $f_j(x_{i-1})$ et $f_j(x_{i+1})$ représentent la valeur de l'objectif j pour les solutions x_{i-1} et x_{i+1} respectivement, f_j est la fonction de l'objectif j , et f_j^{max} ainsi que f_j^{min} désignent respectivement les valeurs maximale et minimale de l'objectif j .

Cette technique identifie les solutions qui présentent une similitude ou une proximité, puis les élimine de l'archive globale. Cela permet de maintenir une variété de solutions représentatives et d'éviter la redondance ou la concentration excessive de solutions similaires au sein de l'archive. L'algorithme MOPSO modifié est exposé dans l'Algorithme 2.

Algorithme 2 Les étapes principales de MOPSO modifié

```

générer les particules de la population
évaluer toutes les particules de la population utilisant les fonction objectifs
initialiser la meilleure solution local dans une archive locale  $A_l$ 
initialiser la vélocité pour chaque particule
identifier les solutions non-dominées et les mettre dans l'archive externe  $A_e$ 
 $t \leftarrow 0$ 
tantque  $t < \text{max d'itération}$  faire
  pour chaque particule  $i$  faire
    choisir une meilleure solution locale  $p_i$  depuis l'archive  $A_l$ 
    choisir une meilleure solution globale  $g$  depuis l'archive  $A_e$ 
    actualiser vitesse utilisant l'Équation 5.2
    actualiser position utilisant l'Équation 5.1
    évaluer la particule utilisant les fonctions objectives
    mise à jours l'archive de  $A_l$  Algorithme 3
  fin pour
  mise à jours l'archive de  $A_e$  utilisant l'Algorithme 4
  appliquer l'Algorithme 5 pour éliminer les solutions proches
   $t \leftarrow t + 1$ 
fin tantque
retourner archive des meilleur solutions  $A_e$ 

```

Les étapes de mise à jour pour l'archive locale et l'archive externe sont présentées dans l'Algorithme 3 et l'Algorithme 4. Ces algorithmes décrivent de manière systématique les opérations nécessaires pour mettre à jour les deux types d'archives afin de conserver les meilleures solutions non dominées.

Pour maintenir une variété de solutions représentatives et éviter la redondance ou la concentration excessive de solutions similaires au sein de l'archive, l'Algorithme 5 montre comment calculer la distance de regroupement et supprimer les solutions proches.

Algorithme 3 Mise à jours l'archive locale

```

si particule est une solution non-dominée alors
  si particule domine une solution dans l'archive  $A_l$  alors
    supprimer solution dominé par particule
  finsi
  ajouter particule dans l'archive  $A_l$ 
finsi
retourner archive interne

```

Algorithme 4 Mise à jours l'archive externe

```

pour chaque particule  $i$  faire
  pour chaque solution  $j$  dans l'archive local  $A_l$  faire
    si solution  $j$  domine une solution dans l'archive externe  $A_e$  alors
      supprimer solution dominé par solution  $j$ 
    finsi
  ajouter solution  $j$  dans l'archive externe  $A_e$ 
finsi
fin pour
retourner archive externe  $A_e$ 

```

Algorithme 5 Calcul de la distance d'agglomération

```

Soit  $nb$  nombre de solution dans l'archive  $A_e$ 
pour chaque objectif  $j$  faire
  trié l'archive  $A_e$  selon la valeur de l'objectif  $j$ 
   $d_1^j = \infty$  &  $d_{nb}^j = \infty$ 
pour  $i = 2$  jusqu'à  $nb - 1$  faire
   $d_i = d_i + d_i^j$ 
fin pour
fin pour
pour chaque solution  $x_i$  dans  $A_e$  faire
  si  $d_i < \text{seuil}$  alors
    supprimé  $x_i$ 
  finsi
fin pour
retourner archive externe

```

5.3 Évolution différentielle

L'évolution différentielle est une version améliorée des algorithmes génétiques, proposée par Price et Storn en 1997 [125]. IL s'agit d'un algorithme basé sur une population initiale comme les algorithmes génétiques. Les opérations principales d'un algorithme génétique sont:

1. Sélection: consiste à choisir les individus qui participeront à la reproduction de la future population. La fonction de sélection peut être aléatoire ou suivre une méthode probabiliste, telle que la roulette et le tournoi.

2. Croisement: recombine deux parents pour produire deux nouveaux enfants dans la génération suivante. Dans un croisement en un point, les chromosomes des parents sont divisés en deux parties lors du croisement, et ces parties sont échangées pour créer les nouveaux enfants. Les positions de croisement sont sélectionnées de manière aléatoire. Il existe d'autres versions de croisement telles que le croisement à deux points ou uniforme.
3. Mutation: inverse aléatoirement une position dans le chromosome. Cela consiste à choisir un gène au hasard et à le remplacer par une autre valeur, choisie au hasard.

L'Algorithme 6 illustre les principales étapes de l'AG, qui seront répétées plusieurs fois jusqu'à l'atteinte d'un résultat optimal ou du nombre d'itérations demandé.

Algorithme 6 Les étapes principales de GA

```

initialiser les individus de la population
évaluer toute la population utilisant la fonction objectif
 $t \leftarrow 0$ 
tantque  $t < \text{max d'itération}$  faire
    sélectionner deux individus utilisant opération de sélection
    appliquer le croisement sur les deux individus
    appliquer la mutation sur les deux individus
    évaluer les nouveaux individus utilisant la fonction objectif
    remplacer les deux anciens individus par les nouveaux dans la population.
     $t \leftarrow t + 1$ 
fin tantque
identifier la meilleure solution  $g$ 
retourner  $g$ 

```

Dans l'évolution différentielle, non seulement les opérations ont été utilisées avec un ordre différent, mais le contenu des opérations a également changé. La différence principale en construisant de meilleures solutions est que les algorithmes génétiques se fondent sur le croisement tandis que le *DE* se fonde sur l'opération de mutation. Cette opération principale est basée sur la différence des paires de solutions aléatoirement tirées depuis la population. L'algorithme utilise l'opération de mutation comme un mécanisme de recherche et l'opérateur de sélection pour diriger la convergence vers les régions éventuelles dans l'espace de recherche.

La notation de l'algorithme (*DE*/ $x/y/z$) est généralement utilisée pour définir une stratégie de *DE* [125] tel que: x spécifie le vecteur à muter, qui peut être un vecteur de population choisi au hasard ou le meilleur vecteur de la population actuelle, y est le nombre de vecteurs différentiels utilisés pour perturber le vecteur cible et z désigne le schéma de croisement qui peut être exponentiel ou binomial.

5.3.1 Les étapes de l'algorithme DE

Le DE est un algorithme d'optimisation globale basé sur la population. Il commence avec une population de N individus de dimension D . Chaque individu représente une solution candidate, $X_{i,t} = \{x_{i,t}^1, \dots, x_{i,t}^D\}$, $i = 1, \dots, N$, où G désigne la génération à laquelle appartient la population, [125]. La population initiale est générée aléatoirement à partir de l'ensemble de l'espace de recherche. Les principales étapes de l'algorithme DE sont les suivantes [177]:

1. Opération de mutation: L'opérateur de mutation permet de modifier ou de perturber la population avec le vecteur mutant $v_{i,t}$ pour chaque individu $x_{i,t}$ dans la population à la génération t . L'opérateur de mutation peut être généré à l'aide d'une stratégie spécifique. Les stratégies les plus fréquemment utilisées sont illustré dans la Table 5.2 [178].

TABLE 5.2 – Stratégies de mutation utilisées

Mutation	Fonction
DE/rand/1	$v_{i,t} = x_{r_1,t} + F.(x_{r_2,t} - x_{r_3,t})$
DE/best/1	$v_{i,t} = x_{best,t} + F.(x_{r_1,t} - x_{r_2,t})$
DE/best/2	$v_{i,t} = x_{best,t} + F.(x_{r_1,t} - x_{r_2,t}) + F.(x_{r_3,t} - x_{r_4,t})$
DE/rand/2	$v_{i,t} = x_{r_1,t} + F.(x_{r_2,t} - x_{r_3,t}) + F.(x_{r_4,t} - x_{r_5,t})$
DE/current-to-best/2	$v_{i,t} = x_{i,t} + F.(x_{best,t} - x_{r_1,t}) + F.(x_{r_2,t} - x_{r_3,t})$
DE/current-to-rand/2	$v_{i,t} = x_{i,t} + F.(x_{r_1,t} - x_{r_2,t}) + F.(x_{r_3,t} - x_{r_4,t})$

Le vecteur $v_{i,t}$ représente un vecteur mutant à produire. Les constantes entières r_1, r_2, r_3, r_4 et r_5 sont générées aléatoirement dans la plage de $[1 \dots N]$, qui sont différentes de l'indice j . $x_{best,t}$ est le meilleur individu à la génération t . Le paramètre F est le facteur d'échelle qui est une constante réelle généralement choisie dans la plage de $[0 \dots 1]$. Il contrôle l'amplification de la variation de différence.

2. Opération de croisement: L'opération de croisement améliore la diversité de la population qui s'applique après la phase de mutation. D'autre part le croisement utilise la mutation du vecteur mutant $v_{i,t}$ pour échanger ses composants avec le vecteur cible $x_{i,t}$ et former le vecteur d'essai $u_{i,t}$. L'opération de croisement est définie dans l'équation 5.4 [179]:

$$u_{i,t}^j = \begin{cases} v_{i,t}^j & \text{Si } (rand_j \leq CR) \text{ ou } (j = j_{rand}) \\ x_{i,t}^j & \text{Sinon} \end{cases} \quad (5.4)$$

où $j = 1, 2, \dots, D$ et $rand_j$ est la j ème évaluation d'un générateur de nombres aléatoires uniforme dans $[0 \dots 1]$ [180]. Le taux de croisement CR est une constante spécifiée par l'utilisateur dans la plage $[0 \dots 1]$ et j_{rand} est un entier choisi au hasard dans la plage $[1 \dots D]$ [180].

3. Opération de sélection: Afin de maintenir la taille de la population constante au cours des générations suivantes, la phase de sélection est effectuée. Le vecteur d'essai est évalué selon la fonction objectif et comparé à son correspondant du vecteur cible dans la génération actuelle. Si le vecteur d'essai est meilleur que le vecteur cible, le vecteur d'essai remplacera le vecteur cible, sinon le vecteur cible restera dans la population. L'opération de sélection est représentée dans l'équation 5.5 [181]:

$$x_{i,t+1} = \begin{cases} u_{i,t} & \text{Si } f(u_{i,t}) \leq f(x_{i,t}) \\ x_{i,t} & \text{Sinon} \end{cases} \quad (5.5)$$

Les trois étapes, mutation, croisement et sélection sont répétées pour chaque génération jusqu'à un critère de terminaison. L'algorithme DE, qui regroupe les étapes présentées ci-dessus, est illustré dans l'algorithme 7.

Les paramètres utilisés dans l'algorithme sont: La dimension du problème D , le nombre N de particules, le nombre maximum d'itération $MaxT$, le choix du schéma de mutation et l'initialisation des valeurs du facteur d'échelle F et CR .

Algorithme 7 Les étapes principales de DE

```

générer les individus de la population
évaluer les individus avec la fonction objective
initialiser la meilleure solution globale  $g$ 
 $t \leftarrow 0$ 
tantque  $t < \text{max d'itération}$  faire
  pour chaque individu  $i$  faire
    générer le vecteur mutant utilisant les opérations de mutation décrites dans la
    Table 5.2
    générer le vecteur d'essai utilisant l'opération de croisement 5.4
    sélectionner l'individu utilisant l'opération de sélection 5.5
    si individu est meilleur que  $g$  alors
       $g \leftarrow$  l'individu
    finsi
  fin pour
   $t \leftarrow t + 1$ 
fin tantque
retourner meilleure solution  $g$ 

```

5.3.2 DE pour l'optimisation multi-objectifs

Dans l'optimisation à objectif unique, la décision dans l'opération de sélection est simple. Le candidat remplace simplement le parent lorsque le candidat est meilleur que le parent. Mais dans les problèmes multi-objectifs, nous pourrions utiliser le concept de pareto pour traiter plusieurs objectifs, afin de sélectionner la meilleure solution dans le mécanisme de sélection.

L'algorithme DE a été étendu pour résoudre des problèmes PMO en raison de sa grande robustesse et de sa rapidité de convergence [181]. Parmi les algorithmes d'optimisation multi-objectifs basés sur le DE, on trouve l'algorithme (PDE– *Pareto Differential Evolution*), utilisé pour créer de nouvelles solutions [182]. Celles non dominées sont conservées en tant que base pour la génération suivante. Les résultats obtenus dans ce cadre ont montré la supériorité de l'algorithme PDE pour la résolution des PMO de complexité accrue. Une autre version d'algorithme proposée par Madavan *et al* [183] (PDEA–*Pareto Differential Evolution Approach*). C'est autre version similaire à celui de PDE. Il applique la technique DE pour créer de nouvelles solutions et les maintenir dans une population auxiliaire.

L'algorithme (VEDE–*Vector Evaluated Differential Evolution*) représente une approche de Differential Evolution (DE) parallèle et multi-population fondée sur l'algorithme génétique d'évaluation vectorielle VEGA posé dans [184]. Une autre version nommée (NSDE–*Non-dominated Sorting Differential Evolution*), introduit une simple modification dans l'algorithme NSGA II [185]. Les opérateurs de croisement et de mutation codés en réel de la NSGA II sont remplacés par le DE [186]. Une variante nommée (MODE–*Multi-Objectifs Differential Evolution*), présentée par Xue *et al* [187], utilise la notion de Pareto et de distance de crowding. Cependant, son approche diffère de celle de l'algorithme PDEA, car la solution trouvée est utilisée pour sélectionner les meilleures solutions pour la prochaine génération. Enfin, la version nommée (DEMO–*Differential Evolution for Multi-Objectifs*), proposée par Robic et Filipic [181], présente des améliorations de l'algorithme DEMO visant à obtenir de bons résultats. Bien que son algorithme présente des similitudes avec celui de PDEA, DEMO adopte une stratégie différente de sélection pour la mise à jour de la population générée.

Comme déjà exposé dans l'algorithme MOPSO, nous avons également apporté des modifications à l'algorithme DEMO afin de le rendre conforme à nos exigences. Pour chaque individu, nous avons introduit une archive de taille fixe regroupant toutes les meilleur solutions locales rencontrées par cet individu, ainsi qu'une archive externe répertoriant les meilleures solutions trouvées par l'ensemble de la population au cours du processus de recherche.

Après avoir appliqué les deux opérations de mutation et de croisement pour obtenir un vecteur d'essai, nous utilisons la stratégie de sélection suivante: si le vecteur d'essai domine l'individu parent, le vecteur d'essai remplacera le parent. Si le vecteur parent domine le

vecteur d'essai, il sera rejeté. Lorsque les vecteurs d'essai et parent ne sont pas liés les uns aux autres, le vecteur d'essai est ajouté à l'archive interne de l'individu. Il sera utilisé pour la mise à jour de l'archive externe. Algorithme 8 explique ce détail.

Algorithme 8 Le principe de l'opération de sélection

```

si vecteur d'essai domine individu alors
    individu  $\leftarrow$  vecteur d'essai
sinon
    si individu domine vecteur d'essai alors
        supprime le vecteur d'essai
    sinon
        ajouter vecteur d'essai a l'archive interne  $A_i$ 
    finsi
finsi

```

L'algorithme DEMO modifié est illustré dans l'Algorithme 9, présentant en détail les étapes principales de l'algorithme DE pour l'optimisation multi objectifs.

Algorithme 9 Les étapes principales de DEMO

```

générer les individus de la population
initialiser les meilleure solutions dans une archive externe  $A_e$ 
initialiser la meilleure solution local dans une archive interne  $A_i$ 
 $t \leftarrow 0$ 
tantque  $t < \text{max d'itération}$  faire
    pour chaque individu  $i$  faire
        générer le vecteur mutant utilisant les opérations de mutation décrites dans la Table 5.2
        générer le vecteur d'essai utilisant l'équation 5.4
        sélectionner l'individu utilisant l'Algorithme 8
        mise à jours l'archive de l'individu  $A_i$  utilisant l'Algorithme 3
    fin pour
    mise à jours l'archive externe  $A_e$  utilisant l'Algorithme 4
    appliquer Algorithme 5.3 pour éliminer les solutions proches
     $t \leftarrow t + 1$ 
fin tantque
retourner l'archive externe  $A_e$ 

```

5.4 Considération finale du chapitre

L'optimisation multi-objectifs est sans doute un axe de recherche primordial pour les scientifiques et les ingénieurs, non seulement à cause de la nature multi-critère de la plupart des problèmes réels, mais également parce que de nombreuses questions restent ouvertes dans ce domaine. Les approches de résolution de problèmes multi-objectifs proposées dans

ce chapitre ont montré les limites des méthodes basées sur la transformation d'un problème multi-objectif en un problème mono-objectif. Malgré ces limites, cette classe de méthodes reste largement utilisée en pratique.

Dans le chapitre 6, nous allons exposer tout particulièrement les trois problèmes d'optimisation: l'affectation des tâches, le placement des IPs ainsi que le problème de routage, ainsi que leur modélisation pour être exploitée par les méthodes d'optimisation présentées dans le chapitre 5. Cette adaptation s'insère dans le problème de conception des réseaux sur puces en trois dimensions.

Chapitre 6

Méthode proposée pour la conception d'un réseau sur puce 3D

Le projet de conception d'une plateforme NoC utilise une méthodologie qui divise le processus en plusieurs étapes. Chaque étape est caractérisée par un degré d'abstraction du matériel à mettre en œuvre. Au cours de ces étapes, différents modèles de plateformes sont développés. Les modèles initiaux sont fonctionnels et ne précisent pas de détails matériels. Tandis que les modèles finaux sont plus structurels et présentent les caractéristiques du matériel à mettre en œuvre. Sur la base de modèles plus ou moins abstraits, des processus d'optimisation sont menés afin d'affiner de plus en plus les solutions intermédiaires, pour obtenir un résultat opérationnel.

6.1 Phase d'affectation

Cette phase permet de choisir les IPs utilisées pour la mise en œuvre de l'application dans le NoC. La méthode d'affectation vise à réutiliser les ressources existantes qui ont été développées précédemment. D'autre part, la réutilisation des ressources se fait lorsque la même IP effectue différentes tâches d'une application. Ceci est particulièrement possible dans les projets qui utilisent plusieurs processeurs, où chaque processeur est capable d'exécuter plusieurs tâches d'une application. Comme chaque processeur a ses propres caractéristiques, la combinaison de différents processeurs avec la réutilisation de certains pour effectuer des tâches spécifiques, se traduit par différentes solutions au problème de l'affectation.

6.1.1 Définition du problème d'affectation

Le problème d'affectation d'IPs consiste à sélectionner un ensemble IPs pour exécuter les tâches d'une ou plusieurs applications données. Ceci repose sur un modèle de graphe de tâches $GT(T, E)$ et sur une bibliothèque d'IPs (IP). La recherche de solutions valables à ce problème se fait à partir de la sélection des IPs en tenant compte des caractéristiques de tâches qui sont fournies dans le cahier des charges de l'application et des caractéristiques des IPs. Mathématiquement on peut définir le problème d'affectation comme suit:

$$Affecter(T, IP) = \left\{ \begin{array}{l} (t_i, ip_j) | \forall t_i \in T, \exists ip_j \in IP, \text{ avec } type(t_i) \in execute(ip_j), \\ i = 1, 2, \dots, n, \text{ et } j = 1, 2, \dots, m, \end{array} \right. \quad (6.1)$$

où $type(t_i)$ représente le type de la tâche t_i , $execute(ip_j)$ est l'ensemble des tâches qui peuvent être exécutées par l'IP ip_j . De plus, n est le nombre de tâches dans le graphe GT et m est le nombre d'IPs dans la bibliothèque d'IPs.

Le résultat de l'affectation est un graphe qui associe les IPs au graphe de tâches GT. Ce graphe est appelé graphe de caractérisation architecturale, noté GCA. L'objectif principal est de trouver un ensemble d'IPs capable d'exécuter l'application tout en minimisant certains objectifs concurrents. Parmi ces objectifs, nous prenons en considération la minimisation de la surface occupée, du temps d'exécution de l'application et de la puissance nécessaire pour cette exécution.

6.1.2 Complexité du problème d'affectation

Considérons une application dont le graphe de tâche comprend n tâches, t_0, t_1, \dots, t_{n-1} et une bibliothèque de m IPs. Soit a_i le nombre d'IP pouvant être allouées à la tâche t_i . Le nombre d'affectation possibles A peut être calculé selon l'Equation 6.2:

$$A = a_0 \times a_1 \times \dots \times a_{n-1} \quad (6.2)$$

notons que nous avons $1 \leq A \leq m^n$. Le cas limite avec $A = 1$ se produit lorsque on utilise un seul IP et cela peut être attribué à n'importe quelle tâche incluse dans le graphe d'application. L'autre cas limite $A = m^n$ se produit lorsque la bibliothèque utilisée couvre m IP et que toutes celles-ci peuvent être attribuées à n'importe quelle tâche du graphe.

L'affectation des IPs a une importance primordiale pour la mise en œuvre d'une application sur une plateforme NoC. Pour illustrer cela, considérons le graphe de tâche illustré dans la Figure 6.1 composé de six tâches. Le nombre d'IPs pouvant être attribués à chacune des tâches varie d'une tâche à une autres, illustré dans la Table 6.1.

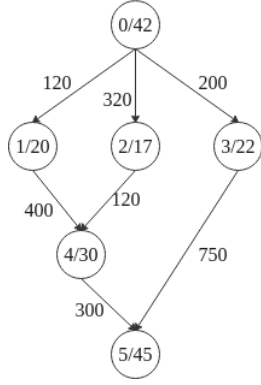


FIGURE 6.1 – Graphe de Taches

Tâche	#IPs possible
0	17
1	17
2	14
3	12
4	14
5	17

TABLE 6.1 – Nombre d'IPs possible

Le nombre total d'affectation correspondant à ce graphe de tâches est: $A = 17 \times 17 \times 14 \times 14 \times 17 = 11555376$, soit plus de 11 millions de combinaisons possibles. La Figure 6.2 présente le graphe de tâches ainsi deux exemple d'affectation possibles. Chaque affectation possible est différente de l'autre en terme d'objectifs. La procédure de calcul des objectifs choisis qui sont la consommation de l'énergie, la surface et le temps d'exécution, est détaillée dans la Section 6.1.4.

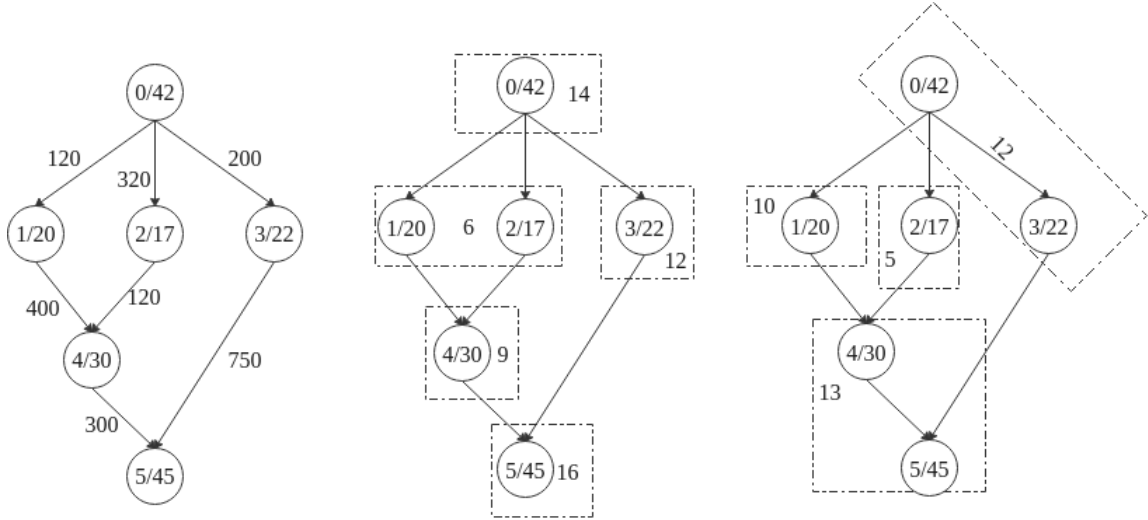


FIGURE 6.2 – Graphe de tâches avec deux affectations possibles

6.1.3 Codification du problème d'affectation

Différents codages sont utilisés pour représenter les objets qui manipulent le processus d'affectation des *IPs*. Ces objets sont la bibliothèque de référence des *IPs*, le graphe des tâches et le représentant des solutions au problème.

1. Bibliothèque de référence des *IPs*: En raison de la capacité à accélérer le développement grâce à la réutilisation des *IPs* dans la conception de systèmes embarqués, tels que les SoCs, MPSoC. Une bibliothèque de référence appelée E3S (*Embedded System*

Synthèses Suit) est utilisée pour valider les approches proposées. Cette bibliothèque comprend 17 processeurs et 46 tâches, chacun associé à des propriétés telles que le temps d'exécution et la consommation d'énergie. Les données sont structurées en format XML pour améliorer la lisibilité et l'intégration dans des outils de calcul. La bibliothèque est divisée en IPs dédiées ou partagées entre tâches. Chaque IP est définie par des attributs liés au processeur et à la tâche, facilitant ainsi l'identification et la représentation dans un format standardisé. La Figure 6.3(a) montre un extrait du document XML utilisé pour représenter le référentiel des IPs.

2. Graphe de tâches: Les NoCs sont généralement conçus pour des applications spécifiques. Les fonctionnalités de ces applications sont définies par des tâches fondamentales et des flux de données, souvent représentés par un graphe de tâches. Dans ce contexte, un graphe de tâches GT(T, E) est un graphe orienté acyclique où chaque nœud représente un module de traitement applicatif représentant une tâche $t_i \in T$. Ces tâches peuvent englober diverses opérations comme des calculs en virgule flottante, des traitements audio ou vidéo, des opérations matricielles et entre autres. Les arêtes orientées $e_{i,j} \in E$ entre les tâches t_i et t_j expriment des dépendances de données ou de contrôle. Une tâche peut être exécutée uniquement lorsque ses tâches dépendantes sont terminées et que les données d'entrée sont accessibles. La Figure 6.3(b) montre le document XML utilisé pour représenter un graphe de tâches.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<repository>
  <ips>
    <ip procName="AMD_ElanSC520-133_MHz--square"
      price="33.0" taskTime="9e-06" taskPower="1.6"
      area="9.61E-6" taskName="Angle2Time Conversion"
      type="0" procID="0" id="0"
    />
    <ip procName="AMD_ElanSC520-133_MHz--square"
      price="33.0" taskTime="2.3e-05" taskPower="1.6"
      area="9.61E-6" taskName="Basic floating point"
      type="1" procID="0" id="1"
    />
    <ip procName="AMD_ElanSC520-133_MHz--square"
      price="33.0" taskTime="0.00049" taskPower="1.6"
      area="9.61E-6" taskName="Bit Manipulation"
      type="2" procID="0" id="2"
    />
    . . .
  </ips>
</repository>
```

(a) Bibliothèque des IPs

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<task_graph>
  <tasks>
    <task code="0" name="src" type="45" />
    <task code="1" name="text" type="44" />
    <task code="2" name="sink" type="45" />
    <task code="3" name="rotate" type="43" />
    <task code="4" name="dith" type="42" />
  </tasks>
  <EDGES>
    <edge name="a0_0" from="0" to="1" cost="1000"/>
    <edge name="a0_1" from="0" to="3" cost="7070"/>
    <edge name="a0_2" from="3" to="4" cost="7070"/>
    <edge name="a0_3" from="4" to="2" cost="7070"/>
    <edge name="a0_4" from="1" to="2" cost="1000"/>
  </edges>
</task_graph>
```

(b) Code du graphe de tâches

FIGURE 6.3 – Exemple de code en XML

3. Codification d'une solution: Une solution d'affectation est représentée sous la forme d'un chromosome ou particule. La solution doit associer un ensemble d'IPs aux tâches du GT de l'application. La Figure 6.4 illustre le codage du chromosomique utilisé. La position du gène sur le chromosome correspond à l'identifiant de la tâche dans le GT. Si nous avons une application de n tâches, alors le gène disposant de la position 0 du chromosome, correspond à la tâche dont l'identifiant est 0 ainsi de suite, jusqu'au gène $n-1$ qui correspond à la tâche d'identifier $n-1$. Chaque gène porte

en lui deux données distinctes: Le type de tâche correspondante et l'identifiant de l'IP attribuée. Ces informations sont essentielles pour déterminer la correspondance entre les gènes et les tâches du système, ainsi que pour spécifier quelle IP est assignée à chaque tâche.

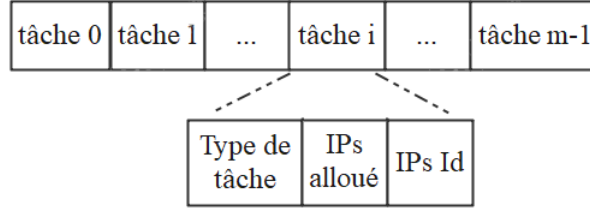


FIGURE 6.4 – Codification d'une solution d'affectation

6.1.4 Fonctions objectifs du problème d'affectation

Pour valider une solution, il est nécessaire d'évaluer les solutions obtenues dans la population actuelle. L'évaluation se fait en fonction des objectifs. Certains objectifs sont collaboratifs, d'autres sont conflictuels ou concurrents. Parmi les objectifs collaboratifs, on peut citer la réduction des IPs, qui introduit une réduction de la surface réelle et donc entraîne une réduction proportionnelle des coûts. Parmi les objectifs contradictoires, on retrouve la minimisation du temps d'exécution, qui est essentiellement liée à l'augmentation de la fréquence de fonctionnement. Cette augmentation, à son tour, provoque une augmentation de la consommation de l'énergie. L'évaluation des objectifs contradictoires est nécessaire pour obtenir des solutions présentant un meilleur équilibre d'efficacité. Afin d'obtenir ces solutions et compte tenu de la relation entre les objectifs, il est nécessaire de considérer l'ensemble minimum formé par trois objectifs principaux: Réduire la surface réelle en minimisant le nombre d'IPs utilisés. Ensuite, réduire la consommation d'énergie en minimisant la consommation de tous les processeurs utilisés. Enfin, il est essentiel de réduire la durée d'exécution de toutes les tâches de l'application. Il convient de noter que le temps consacré à la communication entre les tâches doit être ignoré dans cette phase. Ce sujet sera abordé lors de la phase de placement.

Afin de calculer la surface requise par une solution donnée, il est nécessaire de connaître la surface de chacun des processeurs ou IP. Comme un processeur peut exécuter plus d'une tâche, nous devons vérifier chaque tâche du graphe T et l'associer au processeur sélectionné. La surface est calculée en additionnant toutes les surfaces des IPs utilisés. Cependant, lorsqu'un même IP est associé à plusieurs tâches, il n'est comptabilisée qu'une seule fois. L'Equation 6.3 définit comment calculer la surface requise à utiliser pour une affectation donnée A:

$$Surface(A) = \sum_{ip \in Proc(GCA_A)} Surface_{ip}, \quad (6.3)$$

où $Proc(GCA_A)$ retourne l'ensemble des IPs utilisées dans le graphe GCA pour une affectation donnée A , et $Surface_{ip}$ représente la surface occupée par le processeur ip [53].

Pour évaluer la consommation d'énergie totale d'une application représentée par un graphe de tâches GT, les consommations énergétiques des IPs utilisés dans l'affectation sont additionnées. Dans l'Equation 6.4, $puissance_{t_i}^{ip}$ représente la consommation d'énergie nécessaire pour exécuter une tâche de type t_i à l'aide du processeur d'identifiant ip .

$$Puissance(A) = \sum_{t_i \in T; ip \in Proc(GCA_A)} Puissance_{t_i}^{ip}, \quad (6.4)$$

Afin de calculer le temps d'exécution imposé par une solution d'affectation A , il est nécessaire de visiter toutes les tâches du graphe T de l'application et de les ordonner selon leur propre cycle d'exécution. L'Equation 6.5 définit le calcul du temps d'exécution:

$$Temps(A) = \sum_{ip \in GCA_A; t_i \in T} \max_{t_i \in Steps(T)} (temps_i^{ip}), \quad (6.5)$$

où $temps_i^{ip}$ renvoie le temps nécessaire à l'exécution de la tâche t_i par le processeur ip . $Steps(T)$ renvoie les tâches qui s'exécutent dans le même cycle d'exécution. Lorsque plusieurs tâches parallèles sont attribuées à un même processeur partagé, elles ne peuvent pas être exécutées simultanément et doivent être effectuées séquentiellement. Si nous considérons un scénario où des tâches t_1, t_2, \dots, t_k peuvent être parallélisées, mais sont toutes affectées au même processeur partagé, le temps total d'exécution est la somme des durées d'exécution de toutes les tâches affectées pour ce processeur. L'Algorithme 10 illustre la méthode pour calculer ce temps.

Algorithm 10 Temps d'exécution de k tâches d'un même niveau par le même processeur

```

 $te \leftarrow 0$ 
pour tout  $t_i \in T$  faire
  pour tout  $t_j \in T$  faire
    si  $level(t_i) = level(t_j)$  alors
      si  $processeur(t_i) = processeur(t_j)$  alors
         $te \leftarrow te + temps_j^{ip}$ 
      fin si
    fin si
  fin pour
fin pour
retourner  $te$ 

```

$level(t_i)$ renvoie le cycle d'exécution de la tâche t_i et $processeur(t_i)$ renvoie le processeur qui exécute la tâche t_i . Pour ordonner les tâches dans leur propres cycles d'exécution, un algorithme d'ordonnancement doit être appliqué. Dans cette thèse, trois algorithmes d'ordonnancement sont utilisés:

1. L'ordonnancement (ASAP – *As-Soon-As-Possible*): Consiste à placer les nœuds (en commençant par les nœuds racines) dès la disponibilité des ressources demandées, tout en tenant compte des dépendances de données issues de l'ordonnancement de leurs successeurs [188] comme illustré dans l'Algorithme 11.

Algorithme 11 Ordonnancement *ASAP*

```

 $T \leftarrow \emptyset$ 
 $s \leftarrow 1$ 
pour  $\forall t \in T | Pred(t) = \emptyset$  faire
    Ordonne  $t$  dans niveau  $s$ 
     $T \leftarrow T \cup \{t\}$ 
fin pour
tantque  $\forall t \in GT | t \notin T \& Pred(t) \in T$  faire
     $s \leftarrow s + 1$ 
    Ordonne  $t$  dans niveau  $s$ 
     $T \leftarrow T \cup \{t\}$ 
fin tantque

```

2. L'ordonnancement de type (ALAP – *As-Late-As-Possible*): Consiste à placer les nœuds, en considérant d'abord les nœuds feuilles (en commençant au plus tard). Le placement des prédécesseurs des nœuds ordonnancés, se fait de la même façon, c'est à dire au plus tard, tout en tenant compte des dépendances de données [188] comme illustré dans l'Algorithme 12.

Algorithme 12 Ordonnancement *ALAP*

```

 $T \leftarrow \emptyset$ 
 $s \leftarrow derniercycle$ 
pour  $\forall t \in T | Succ(t) = \emptyset$  faire
    Ordonne  $t$  dans niveau  $s$ 
     $T \leftarrow T \cup \{t\}$ 
fin pour
tantque  $\forall t \in T | t \notin T \& Succ(t) \in T$  faire
     $s \leftarrow s - 1$ 
    Ordonne  $t$  dans niveau  $s$ 
     $T \leftarrow T \cup \{t\}$ 
fin tantque

```

3. L'ordonnancement avec liste (LS – *List-Scheduling*): L'algorithme d'ordonnancement par LS [189], est une extension de l'ordonnancement de type ASAP et ALAP. Ce type d'algorithme utilise une liste dans laquelle sont classés les nœuds. Le classement d'un nœud peut dépendre de plusieurs critères dont: Sa priorité, les données disponibles, les ressources disponibles et entre autre. Il existe plusieurs variantes de ce type d'algorithme. La liste peut être statique, construite une fois pour toute au début

ou dynamique, mise à jour à chaque nouvelle configuration. Dans notre thèse, nous utilisons la liste statique qui est basée sur le résultat obtenu des deux algorithmes ASAP et ALAP, le détail de l'algorithme est dans l'Algorithme 13.

Algorithme 13 Ordonnancement LS

```

 $S_{GT}^{asap} \leftarrow ASAP(GT)$ 
 $S_{GT}^{alap} \leftarrow ALAP(GT)$ 
pour  $\forall t \in T$  faire
     $S_1 \leftarrow S_{tasap}(t)S_T^{asap}(t)$ 
     $S_2 \leftarrow S_{talap}(t)S_T^{asap}(t)$ 
    Ordonne  $t$  dans niveau random( $S_1, S_2$ )
fin pour

```

où S_{GT}^{ASAP} désigne l'ordonnancement du graphe de tâches GT avec l'algorithme ASAP et $S_t^{ALAP}(t) \in S_T^{ASAP}(t)$ renvoie le niveau d'ordonnancement de la tâche t en utilisant l'algorithme ALAP.

Lorsque les trois fonctions objectifs sont appliquées à l'exemple illustré dans la Figure 6.1, les tâches sont affectées aux processeurs selon la séquence suivante: 6, 10, 10, 5, 10, 13. Les résultats obtenus pour les fonctions objectives sont les suivants:

- Puissance: 9,01 *Watts*.
- Surface: $1.218 \cdot 10^{-5} \text{metre}^2$
- Temps d'exécution: $6 \cdot 10^{-3} \text{Seconde}$ (après application de l'algorithme ALAP).

6.2 La Phase de placement

Le résultat obtenu de la phase d'affectation était un graphe qui est une association entre les IPs et le graphe de caractérisation architecturale notée GCA. L'étape suivante est l'étape de placement. Après avoir sélectionné les meilleures ressources IPs, en fonction des caractéristiques du T et de la bibliothèques des IPs, il est nécessaire de sélectionner le meilleur emplacement pour ces ressources dans l'implémentation physique du NoC.

6.2.1 Définition du problème de placement

Le placement des IPs est l'un des principaux problèmes de conception d'un NoC [190]. Il consiste à placer les IPs d'un graphe GCA dans une architecture fixe, de sorte que certains objectifs soient optimisés. L'architecture est donnée par le graphe d'architecture $NT(R, E)$, où R est l'ensemble des ressources, et E l'ensemble des canaux de communication entre ces ressources. Le résultat obtenu par la première phase, comme présenté dans la Section

6.1, est donné par le graphe $GCA(I, E)$, correspondant au graphe de tâches $GT(T, E)$. Ce graphe orienté est défini par chaque nœud $ip \in I$ étant associé à une tâche $t_i \in T$, où ip représente l'IP alloué pour exécuter la tâche t_i .

À noter que le GCA n'est qu'un graphe de tâches GT applicatif avec la sélection des IPs allouées aux tâches. Ces IPs peuvent être dédiées à une seule tâche ou partagées par plusieurs d'entre elles.

Mathématiquement, on peut définir le problème de placement comme suit:

$$Placer(I, R) = \left\{ (ip_i, r_j), \mid \forall ip_i \in I, \exists r_j \in R, i = 1, 2, \dots, p, \text{ et } j = 1, 2, \dots, q \right\} \quad (6.6)$$

où ip_i représente l'IP qui exécute la tâche t_i , r_j représente la ressource où l'IP ip_i va être placée dans l'architecture NoC. De plus, p représente le nombre d'IPs dans le graphe de tâches CGA, et q représente le nombre de ressources disponibles dans l'architecture du NoC.

Les objectifs de placement peuvent être le coût de mise en œuvre, la consommation d'énergie, le temps d'exécution et entre autres.

6.2.2 Complexité du problème de placement

Le nombre de placement possible augmente d'une façon factorielle, avec la quantité de ressources réseau. Le problème de placement est un problème d'affectation quadratique. Ce type de problème est aussi difficile à résoudre qu'un problème NP-complet. Considérons une application dont le graphe comprend n tâches, t_0, t_1, \dots, t_{n-1} et une plateforme NoC de q ressources. Le nombre de placement possibles différents M , dépend du nombre de ressources disponibles dans le NoC et varie également avec le nombre des IP. Pour les IP utilisées dans l'affectation, soit p le nombre de processeurs, dont certains à usage dédié et d'autres à utilisation partagée.

Dans le cas où chaque IP est dédiées à une seule tâche, le nombre d'IP p est égal au nombre de tâches n , tandis que dans le cas de l'utilisation d'IP partagées par plusieurs tâches, le nombre d'IPs est inférieur au nombre de tâches. En supposant que la plateforme NoC dispose de ressources suffisantes pour mettre en œuvre l'application, nous aurons donc $q \geq p$. Dans ce cas, le nombre de placement possibles peut être calculé selon l'Equation 6.7.

$$M = \frac{q!}{(q-p)!}, \quad (6.7)$$

notons que $1 \geq M \geq q!$. Le cas minimum se produit lorsque la plateforme du NoC est constituée par d'une seule ressource et l'affectation comprend une seule IP avec $M = 1$, tandis que le cas maximum $M = q!$ se produit lorsque le nombre des tâches, des IPs et les ressources coïncident. Nous aurons $n = p = q$. Pour illustrer cela, nous donnons quelques

exemples du nombre d'affectations réalisables pour certaines valeurs de q et p indiqués dans la Table 6.2. L'étape de placement utilise le résultat de la première phase (l'affectation

TABLE 6.2 – Nombre de placement selon le nombre de processeurs et le nombre de ressources du NoCs [17]

Nombre de processeurs p	Nombre de ressources q		
	4	9	16
2	12	72	240
3	24	504	3.360
4	24	3.024	43.680
5	–	15.120	524.160
6	–	60.480	5.765.760
7	–	181.440	57.657.600
8	–	362.880	518.918.400
9	–	362.880	4.151.347.200

), qui se compose de plusieurs solutions non dominées. Par conséquent, soit A le nombre d'affectations distinctes obtenues et p_i le nombre de processeurs alloués dans la solution i , et, n_i le nombre minimum de ressources dans le NoC pour la mise en œuvre de l'application à l'aide de l'affectation i . Dans ce cas, le nombre total de placement à considérer est défini par la somme des placement possibles se référant à chacune des affectations, comme décrit dans l'Equation 6.8.

$$M_A = \sum_{i=1}^s \frac{q!}{(q - p)!} \quad (6.8)$$

À titre d'exemple, considérons 4 solutions d'affectations valides d'une application donnée. Le calcul du nombre de solution de placement possibles est illustré dans la Table 6.3.

TABLE 6.3 – Nombre de solutions de placement possibles [17]

Affectation i	Nombres de		
	processeur p	ressources q	placements possible
1	3	9	504
2	6	9	60.480
3	4	9	3.024
4	9	9	362.880
Nombre Total de placement			426.888

6.2.3 Codification du problème de placement

La codification du problème de placement utilisée, est une extension du codage adopté dans l'étape d'affectation des IPs. La Figure 6.5 représente une solution de placement d'une application de n tâches. Dans ce cas, en plus des attributs utilisés dans l'étape d'affectation des tâches, un nouvel attribut d'identification est introduit nommé ressource. Cet attribut indique l'adresse IP sur laquelle la tâche doit être placée. Rappelons que l'indice, dans cette codification, identifie la tâche du graphe. Lorsque deux tâches t_1, t_2 sont affectées dans le même IP mais physiquement placées dans des ressources différentes cela, signifie qu'il existe deux IPs de même type pour exécuter ces tâches.

Au niveau de la topologie de réseau adoptée *3D mesh*, chaque ressource est identifiée par un indice entier et séquentiel. Il commence par le coin inférieure gauche et allant vers le coin supérieur droit, ligne par ligne, colonne par colonne et niveau par niveau. La première ressource est identifiée avec l'index 0, car la ligne du bas est la ligne 0 et la colonne la plus à gauche est la colonne 0 avec le niveau le plus bas 0.

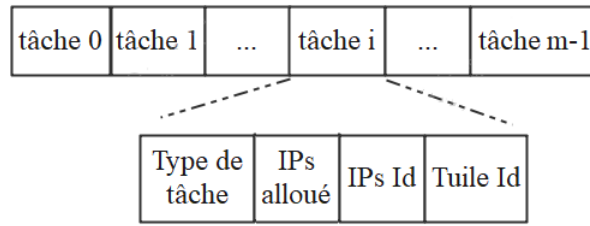


FIGURE 6.5 – Codification d'une solution de placement

Dans un NoC maillé $N \times N \times N$ et à partir de l'index des ressources, le calcul du niveau z_i , de la ligne x_i et de la colonne y_i de la i ème ressource se fait, respectivement par les Equations 6.9–6.11:

$$z_i = i \text{ div } N \times N \quad (6.9)$$

$$x_i = (i - (N \times N) \times z_i) \text{ div } N \quad (6.10)$$

$$y_i = (i - (N \times N) \times z_i) \text{ mod } N \quad (6.11)$$

dans le cas où $i = 23$ et la taille du Noc est $3 \times 3 \times 3$, alors $z_i = 2$, $x_i = 1$ et $y_i = 2$.

6.2.4 Fonctions objectifs du problème de placement

Chaque solution doit être évaluée en fonction des objectifs visés. Dans l'étape d'attribution des tâches, trois objectifs à minimiser ont été définis: La surface occupée, la consommation d'énergie et le temps d'exécution de l'application comme présenté dans la Section 6.1.4. Dans l'étape de placement, ces trois objectifs seront à nouveau considérés, étant donné que le calcul de chacun est basé sur les données du référentiel et le profil de

communication de la demande. La solution de placement fournit une vue de la mise en œuvre matérielle. Elle permet l'évaluation des aspects qui ne peuvent l'être lors de l'étape d'affectation d'IPs à savoir: La consommation d'énergie pour assurer la communication, la surface nécessaire à cette communication et le temps de communication dans le temps total d'exécution.

Afin de calculer la surface requise par une solution de placement donnée, il est nécessaire de connaître la surface nécessaire aux processeurs sélectionnés et celle requise par les liens et routeurs utilisés. Comme un processeur peut être responsable de plus d'une tâche, chaque nœud du graphe GCA, qui représente une solution de placement M , doit être visité afin de vérifier l'identification du processeur dans l'élément $E3S$ approprié.

Le nombre total de liens et de routeurs peut être obtenu en considérant tous les chemins de communication entre les ressources exploitées. Notons qu'un placement des IPs donné, peut ne pas utiliser la totalité des ressources, des liens et des routeurs disponibles. En revanche, certains canaux de communication peuvent être réutilisés plusieurs fois. L'algorithme de routage utilisé, influence fortement l'utilisation des canaux de communication et de routeurs de réseau. Pour évaluer les composants de l'architecture de communication utilisée, il est nécessaire de connaître l'algorithme de routage utilisé. Adopter un algorithme de routage statique tel que XYZ, permet de calculer le nombre de canaux de communication utilisés entre deux Ressources. Le nombre de liens est calculé par l'Equation 6.12. Cela représente également la distance entre les ressources r_i et r_j et elle est appelée *Manhattan distance* [191].

$$nLinks(r_i, r_j) = |x_i - x_j| + |y_i - y_j| + |z_i - z_j| \quad (6.12)$$

où (x_i, y_i, z_i) et (x_j, y_j, z_j) sont les coordonnées tridimensionnelles des ressources r_i et r_j . Dans un NoC 3D l'Equation 6.12 est redéfinie dans l'Equation 6.13:

$$nLinks(r_i, r_j) = nLiensH(r_i, r_j) + nLiensV(r_i, r_j), \quad (6.13)$$

où $nLiensH(r_i, r_j)$ et $nLiensV(r_i, r_j)$ représentent respectivement le nombre de liens traversés horizontalement et verticalement, calculé selon l'Equation 6.14 et 6.15 respectivement:

$$nLiensH(r_i, r_j) = |x_i - x_j| + |y_i - y_j| \quad (6.14)$$

$$nLiensV(r_i, r_j) = |z_i - z_j|, \quad (6.15)$$

le nombre de routeurs traversés est défini par $nLinks(r_i, r_j) + 1$.

La surface totale utilisée est calculée en additionnant les surfaces requises par la mise en œuvre de tous les processeurs, routeurs et liens distincts. L'Equation 6.16 décrit le calcul nécessaire pour obtenir la surface totale pour la mise en œuvre d'un placement des IPs donné M :

$$Surface(M) = surface_p(M) + Surface_c(M), \quad (6.16)$$

où la fonction $surface_p(M)$ fournit la surface nécessaire utilisée par les processeurs pour la mise en œuvre d'une solution de placement M , présentée dans l'Equation 6.3 et $Surface_c(M)$ est calculée selon l'Equation 6.17:

$$Surface_c(M) = \sum_{\forall e_{i,j} \in E} (A_{LH} \times nLiensH(r_i, r_j) + A_{LV} \times nLiensV(r_i, r_j) + A_r \times (nLinks(r_i, r_j) + 1)), \quad (6.17)$$

où $e_{i,j}$ représente une communication entre la tâche t_i et t_j , A_{LH} est la surface d'un lien horizontal, A_{LV} est la surface d'un lien vertical, A_r est la surface d'un routeur donné et r_i, r_j représentent les ressources exploitées pour une communication donnée par $e_{i,j}$.

La consommation d'énergie totale d'une application sur le NoC comprend la consommation électrique des processeurs lors du traitement du calcul effectué par chaque IP, et celle due au transport de données entre les ressources, comme illustré dans l'Equation 6.18:

$$Puissance(M) = Puissance_p(M) + Puissance_c(M) \quad (6.18)$$

où $Puissance_p(M)$ et $Puissance_c(M)$ sont respectivement la consommation d'énergie de traitement et de communication.

Le premier élément de l'Equation est décrit dans l'Equation 6.4, qui représente la puissance nécessaire pour exécuter les tâches d'une application représentée par le graphe GT. Le deuxième élément $Puissance_c(M)$ représente la puissance nécessaire pour assurer la communication d'une application, qui est caractérisé par un ensemble de bits qui circule entre plusieurs ressources.

Soit E_{bit} , l'énergie consommée par un bit lorsqu'il se déplace entre routeurs. L'énergie consommée pour un seul bits E_{bit} est calculée à l'aide de l'Equation 6.19:

$$E_{bit} = E_{R_{bit}} + E_{LV_{bit}} + E_{LH_{bit}} \quad (6.19)$$

où $E_{R_{bit}}$, $E_{LH_{bit}}$ et $E_{LV_{bit}}$ représentent respectivement la consommation d'énergie du routeur et des liens de communication physique horizontaux et verticaux. Par conséquent, la consommation d'énergie totale de l'envoi d'un bit de données de la ressource r_i à la ressource r_j est calculée en prenant en compte le nombre de routeurs et de liens que le bit traverse sur son chemin. Cette consommation d'énergie est calculée dans l'Equation 6.20:

$$E_{bit}^{r_i, r_j} = nLiensH(r_i, r_j) \times E_{LH_{bit}} + nLiensV(r_i, r_j) \times E_{LV_{bit}} + (nLinks(r_i, r_j) + 1) \times E_{R_{bit}} \quad (6.20)$$

La puissance totale pour un placement donné M est calculée en utilisant l'Equation 6.21.

$$Puissance_c(M) = \sum_{\forall e_{i,j} \in E} V(d_{t_i, t_j}) \times E_{bit}^{r_{t_i}, r_{t_j}} \quad (6.21)$$

où $V(d_{t_i,t_j})$ fournit le nombre de bits pour une communication $e_{i,j}$ entre la tâche t_i et la tâche t_j et r_t renvoie le numéro de la ressource dans lequel la tâche t est placée.

Pour calculer le temps d'exécution d'une solution de placement donnée, nous considérons le temps d'exécution de chaque tâche, leur ordonnancement et le temps supplémentaire dû au communication de ces données. Le temps d'exécution de chaque tâche est défini par l'attribut *taskTime* dans le graphe de tâche TG. Il est nécessaire de visiter toutes les tâches de TG et de les ordonnancer dans son propre cycle d'exécution. En ce qui nous concerne, Nous avons utilisé les algorithmes d'ordonnancement présentés dans la Section 6.1.4 qui sont: ASAP, ALAP et LS. Les liens et les routeurs peuvent être évalués à l'aide d'un algorithme de routage. Nous avons identifié deux situations qui peuvent dégrader les performances de mise en œuvre, allongés ainsi le temps d'exécution de l'application et sa consommation d'énergie:

1. Tâches parallèles avec chemin de communication partiellement partagé: Lorsqu'une tâche dans une ressource doit envoyer des données à des tâches soi-disant parallèles dans différentes ressources via le même lien initial, les données des autres ressources ne peuvent pas être envoyées en même temps Figure 6.6.

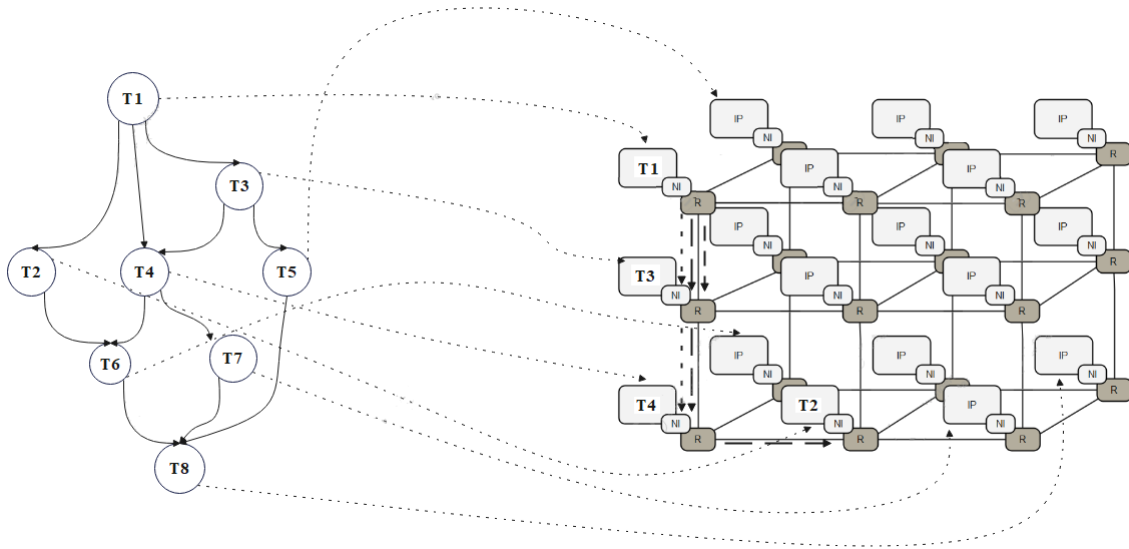


FIGURE 6.6 – Communication entre la même source et plusieurs destinations

2. Tâches parallèles avec cible commune utilisant le même chemin de communication: Lorsque plusieurs tâches doivent envoyer des données à une tâche cible commune, un ou plusieurs liens partagés seraient nécessaires. Les données des autres tâches doivent ensuite être mises en attente et n'arriveront donc pas en même temps à la tâche cible comme illustré dans la Figure 6.7.

L'Equation 6.22 donne le temps d'exécution compte tenu du temps de calcul et le temps de la communication pour une solution de placement donnée M .

$$Temp(M) = Temps_{M_p} + Temp_{M_c} + (F_1 + F_2) \quad (6.22)$$

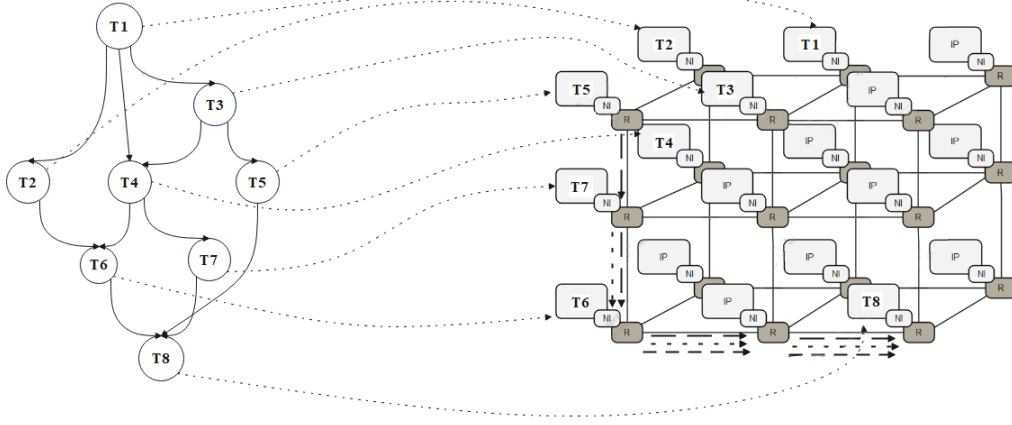


FIGURE 6.7 – Communication entre plusieurs source et la même destination

où $Temp_{M_p}$ représente le temps nécessaire pour exécuter les tâches du graphe TG présenté dans l'Equation 6.5. $Temp_{M_c}$ est le temps passé à communiquer entre les tâches.

La fonction F_1 calcule le temps supplémentaire dû à la situation 1, tandis que la fonction F_2 calcule le temps supplémentaire dû à la situation 2.

Le modèle de temps de communication utilisé dans cette étude est basé sur le délai de commutation wormhole, qui comprend le délai de routage. Le délai de routage correspond au temps nécessaire pour acheminer les données de la source à la destination. Un phit représente la quantité de bits transmis simultanément. L'algorithme de routage utilisé est XYZ et le délai de routage dépend du nombre de routeurs dans la route du phit entre sa source et sa destination.

Le temps de routage $T_{phit}^{r_i, r_j}$ d'un *phit*, envoyé par la ressource r_i vers la ressource r_j , considérant que la route contient $nLinks(r_i, r_j)$ routeurs et qu'il n'y a pas de conflit dans les routeurs, est défini dans l'Equation 6.23.

$$T_{phit}^{r_i, r_j} = nLiensH(r_i, r_j) \times T_{LH_{phit}} + nLiensV(r_i, r_j) \times T_{LV_{phit}} + (nLinks(r_i, r_j) + 1) \times T_{R_{phit}} \quad (6.23)$$

où $T_{LH_{phit}}$, $T_{LV_{phit}}$ et $T_{R_{phit}}$ sont respectivement le temps pour envoyer un phit dans un lien horizontal, vertical et le routeur. Le temps passé à cause de la communication $Time_c$ entre les tâches est calculé dans l'Equation 6.24.

$$Temp_c = \sum_{e_{i,j} \in E} \left\lceil \frac{V(d_{t_i, t_j})}{phit} \right\rceil * T_{phit}^{r_i, r_j} \quad (6.24)$$

où $V(d_{t_i, t_j})$ est le volume de bits transmis de la tâche t_i à la tâche t_j associé aux ressources r_i et r_j .

Lorsque les trois fonctions objectifs sont appliquées à l'exemple illustré dans la Figure 6.1, les tâches sont affectées aux processeurs selon la séquence suivante: 6, 10, 10, 5, 10, 13. avec un placement dans le réseau sur puce selon la séquence: 4, 0, 1, 5, 0, 2. Les résultats obtenus pour les fonctions objectives sont les suivants:

- Puissance : 17,24 *Watts*.
- Surface : $2.47 \cdot 10^{-5} \text{metre}^2$.
- Temps d'exécution : $5.6 \cdot 10^{-3} \text{Seconde}$.

6.3 Phase de routage

La recherche d'un routage optimal dans un NoC appartient à la classe des problèmes de routage. Plus précisément, il s'agit du cas particulier d'un problème de routage de paquets. Parmi les problèmes d'optimisation combinatoire, les problèmes de routage sont de complexité NP complète. Cette classe comprend également le problème du voyageur de commerce, le problème de tournée des véhicules et le problème du plus court chemin.

6.3.1 Définition du problème de routage

Une application implémentée dans un NoC est caractérisée par un graphe de tâches $GT(T, E)$, où T est l'ensemble des tâches et E est l'ensemble des communications entre ces tâches. Chaque communication est définie par une tâche source, une tâche destination et un nombre de bits à injecter. Les communications sont représentées par l'ensemble des arêtes $E = e_1, e_2, e_3, \dots, e_l$ dans le graphe GT . Chaque arête e_k est composée d'un nœud source $src(e_k) \in T$, d'un nœud destination $dst(e_k) \in T$, et d'un volume de données à envoyer $comm_k^{t_i, t_j}$, où $t_i = src(e_k)$ et $t_j = dst(e_k)$. Le graphe $NT(R, C)$ représente la topologie du réseau, où R est l'ensemble des ressources et C est l'ensemble des canaux de communication entre ces ressources. Un ensemble de chemins P assure la transmission de paquets entre la ressource source et la ressource destination.

Mathématiquement, le routage est définie dans l'Equation 6.25.

$$route(E, P) = \begin{cases} (e_k, p_a) \mid \forall e_k \in E, \exists p_a = (r_0, r_1, \dots, r_v) \in P, \\ \text{avec } src(e_k) = r_0 \text{ et } dst(e_k) = r_v, \\ k = 1, 2, \dots, l \text{ et } a = 1, 2, \dots, b \end{cases} \quad (6.25)$$

où $src(e_k)$ et $dst(e_k)$ sont les sources et destinations de l'arête e_k , l est le nombre d'arêtes dans le graphe de tâches GT , et b représente le nombre de chemins existant dans l'ensemble P . De plus, r_0 et r_v représentent la ressource source et la ressource destination d'un chemin p_a .

6.3.2 Complexité du problème de routage

Considérons une application dont le graphe comprend n tâches, t_0, t_1, \dots, t_{n-1} et l arrêtes. Soit s_k le nombre de chemins différents pouvant être allouées à chaque arrête $e_{i,j}$. Le nombre des différentes combinaisons possibles S peut être calculé selon l'Equation 6.26.

$$S = s_0 \times s_1 \cdots \times s_{l-1} \quad (6.26)$$

Notons que nous avons $1 \leq S \leq s^l$. Le cas limite avec $S = 1$ se produit lorsqu'il existe une seule route pour chaque arrête. $S = s^l$ se produit lorsque chaque arrête $e_{i,j}$ peut avoir s routes différentes. Pour illustrer cela, considérons le graphe des tâches composé de 8 arêtes, où le nombre chemins possibles est illustré dans la Table 6.4.

TABLE 6.4 – Nombre de chemins possibles

L'arête	Chemins possible
0	5
1	6
2	9
3	4
4	5
5	7
6	8
7	6
8	4

le nombre total de routes correspondantes à ce graphe de tâches est $S = 5 \times 6 \times 9 \times 4 \times 5 \times 7 \times 8 \times 6 \times 4 = 7257600$, soit plus de 7 millions de combinaisons de routage possibles. La Figure 6.8 présente un exemple de plusieurs routes possibles sur la même arête $e_{i,j}$. Chaque chemins possible est différent de l'autre en terme d'évaluation (selon les objectifs choisis pour évaluer une solution de routage).

6.3.3 Codification du problème de routage

Pour codifier le problème du routage, certains objets doivent être codés et utilisés pour le processus de routage, ces objets sont:

1. Chemin: Un chemin entre une source (Src) et une destination (Dst), est un ensemble de nœuds intermédiaires qui assurent le transport des paquets depuis la source jusqu'à la destination. Dans cette thèse, la génération du chemin se fait d'une manière aléatoire. Partant de la source Src , un prochain saut est choisi d'une façon aléatoire parmi les nœuds voisins du nœud courant jusqu'à l'atteinte la destination

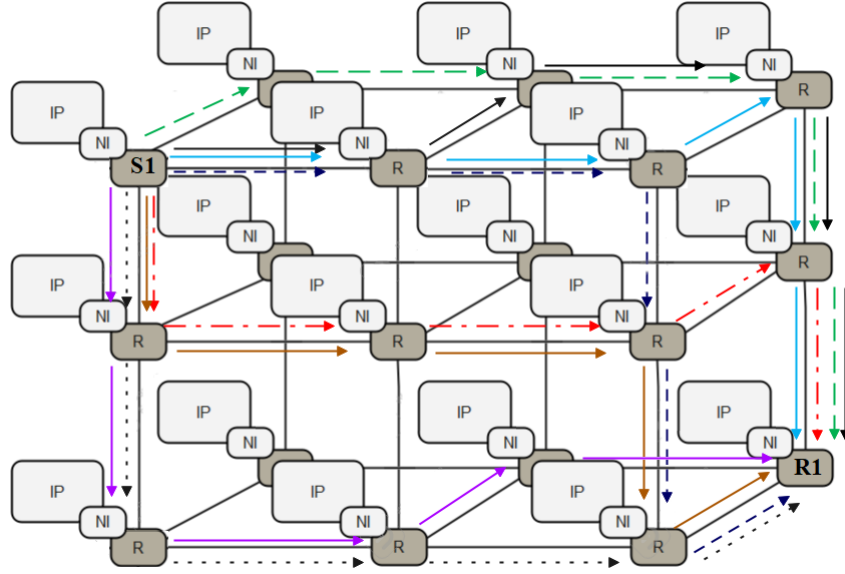


FIGURE 6.8 – Plusieurs chemins possibles pour une communication.

Dst. Pour éviter une situation de blocage, il est recommandé d'éviter de revenir à un nœud déjà visité. Dans le cas où tous les voisins du nœud courant ont été déjà utilisés dans le chemin en cours de construction, aucun prochain saut ne sera possible. Dans ce cas là, ce chemin sera annulé et nous revenons à la case départ.

On peut créer deux types de chemins: Minimal et non minimal. Le chemin minimal relie une source de coordonnées (x_i, y_i, z_i) à une destination de coordonnées (x_j, y_j, z_j) est considéré comme minimal si sa longueur est égale à la distance de Manhattan définie dans l'Equation 6.12. Tandis qu'un chemin non minimal est celui dont sa longueur est donnée par l'Equation 6.27.

$$longueur(r_i, r_j) = nLinks(r_i, r_j) + x \quad (6.27)$$

où x est un nombre défini par l'utilisateur afin de limiter le nombre de sauts supplémentaires nécessaires pour construire un chemin non minimal. La Figure 6.9 illustre les deux types de chemin.

2. Ensemble de chemins: Pour un routage optimal, la paire (Src, Dst) représente une communication entre une source Src , et une destination Dst , elle ne doit pas être basée sur un seul chemin. Il faut créer plusieurs chemins différents de la même paire, afin de pouvoir choisir le chemin efficace. Malgré l'aspect aléatoire de notre approche, nous avons proposé de générer s chemins différents et aléatoires entre chaque paire (Src, Dst) . Parmi lesquels, le meilleur chemin sera maintenu dans l'ensemble final des chemins de la solution. La Table 6.5 montre 5 chemins possible pour la même paire (Src, Dst) .

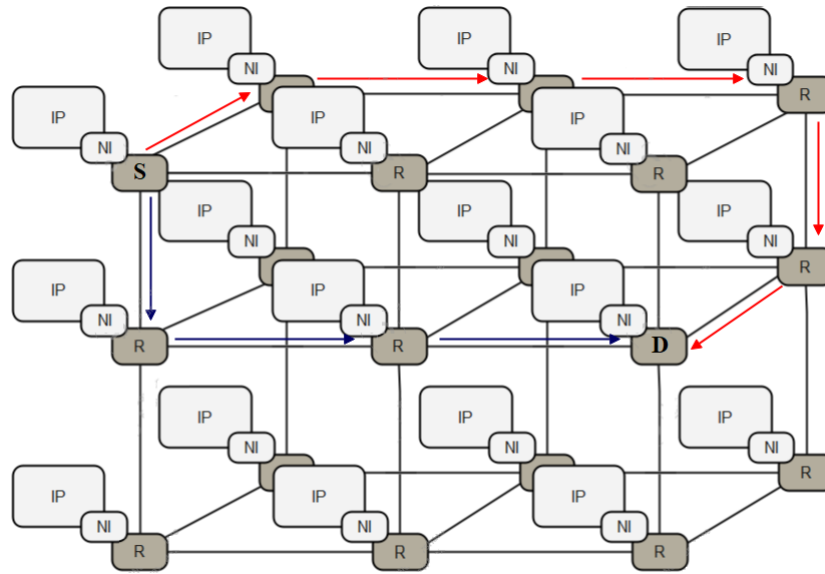


FIGURE 6.9 – Chemin minimal et chemin non minimal

TABLE 6.5 – Plusieurs chemins d'une même paire

Paire (Src, Dst)	ID Chemin	Chemin
(13, 8)	1	[13, 9, 5, 6, 7, 8]
	2	[13, 9, 10, 11, 12, 8]
	3	[13, 14, 15, 16, 12, 8]
	4	[13, 14, 10, 11, 7, 8]
	5	[13, 9, 10, 6, 7, 8]

3. Solution complète: Une solution complète comporte l'ensemble de chemins reliant plusieurs paires de nœuds concernées par le routage selon le type de l'application à implémenter dans le *NoC*. À la première étape, le choix du meilleur chemin parmi les s chemins générés est effectué d'une manière aléatoire. Ensuite l'algorithme utilisé prend le relais pour pouvoir choisir la meilleure solution selon les objectifs fixés par le concepteur. La Table 6.6 montre un exemple d'une solution complète choisie parmi les chemins qui existent sur chacune des paires (Src, Dst).

TABLE 6.6 – Solution complète du routage

Paire (Src, Dst)	ID Chemin choisi	Détail du chemin
(6, 16)	4	[6, 10, 11, 12, 16]
(1, 11)	1	[1, 2, 3, 7, 11]
(10, 4)	5	[10, 11, 12, 8, 4]
(9, 15)	2	[9, 10, 14, 15]
(13, 8)	3	[13, 14, 15, 16, 12, 8]

4. Table de routage: La solution finale obtenue à ce niveau est constituée d'un ensemble de chemins reliant les différentes paires, (Src, Dst) , selon le type de l'application. Ces chemins sont sauvegardés dans un fichier sous forme de tables de routage, utilisé par le routeur. Pour chaque paires (Src, Dst) , on associe la liste des nœuds intermédiaires, qui séparent le nœud source de nœud destination. Un exemple de table de routage est indiqué dans la Table 6.7.

TABLE 6.7 – Table de routage d'un exemple d'application

Source	Destination	Chemin choisi
1	9	[1, 10, 9]
2	18	[2, 1, 10, 19, 18]
3	1	[3, 0, 1]
4	10	[4, 13, 10]
5	19	[5, 14, 13, 10, 19]
6	2	[6, 7, 4, 1, 2]
7	11	[7, 4, 1, 10, 11]
8	20	[8, 17, 26, 23, 20]
9	3	[9, 12, 3]
10	12	[10, 9, 12]
11	21	[11, 14, 23, 22, 21]
12	4	[12, 13, 4]
14	22	[14, 13, 22]
15	5	[15, 6, 7, 8, 5]
16	14	[16, 17, 14]
17	23	[17, 14, 23]
18	6	[18, 21, 24, 15, 6]
19	15	[19, 22, 13, 12, 15]
20	24	[20, 19, 18, 21, 24]
21	7	[21, 12, 13, 4, 7]
22	16	[22, 25, 16]
23	25	[23, 26, 25]
24	8	[24, 25, 16, 17, 8]
25	17	[25, 26, 17]

6.3.4 Fonction objectif du problème de routage

L'optimisation du routage dans les NoCs vise à réduire la latence de communication. Elle permet ainsi un temps d'exécution plus court de l'application embarquée. Dans une application embarquée et pour un ensemble de communications envoyées simultanément sur le réseaux, deux objectifs principaux à minimiser peuvent être pris en considération: Le délai de transmission pour les paquets et le nombre de nœuds traversés entre la source et la destination.

La congestion d'un réseau est une augmentation du trafic provoquant un ralentissement global de celui-ci. Le contrôle de la congestion est l'un des problèmes les plus difficiles lors de la conception d'un NoC à haut débit et à faible latence [192]. C'est le fait que le trafic entrant dépasse la bande passante sortante qui produit : Un délai de mise en file d'attente, une perte de paquets et un blocage des nouvelles communication [193]. En fait, pour faire face à la congestion des liens, il est possible d'augmenter la taille des tampons, mais cela entraîne une surcharge de zone. Sinon, limiter la taille des tampons et donc dégrader les performances du NoC en termes de latence et de débit. La Figure 6.10 illustre trois paquets dont les nœuds d'origine sont les nœuds Src_1 , Src_2 et Src_3 . Ils doivent être envoyés au nœud situé en Dst_1 , Dst_2 et Dst_3 . Les flèches indiquent les chemins trouvés avec l'algorithme de routage. Les trois paquets doivent passer par le même canal de communication pour atteindre leur destination. Étant donné que chaque canal de communication prend en charge la transmission d'un seul flit par cycle d'horloge, un seul paquet sera transmis (choisi par une politique d'arbitrage), tandis que les autres sont bloqués, comme le montre la Figure 6.10. Malgré que les chemins soient minimaux, l'algorithme ne peut empêcher la congestion de se produire.

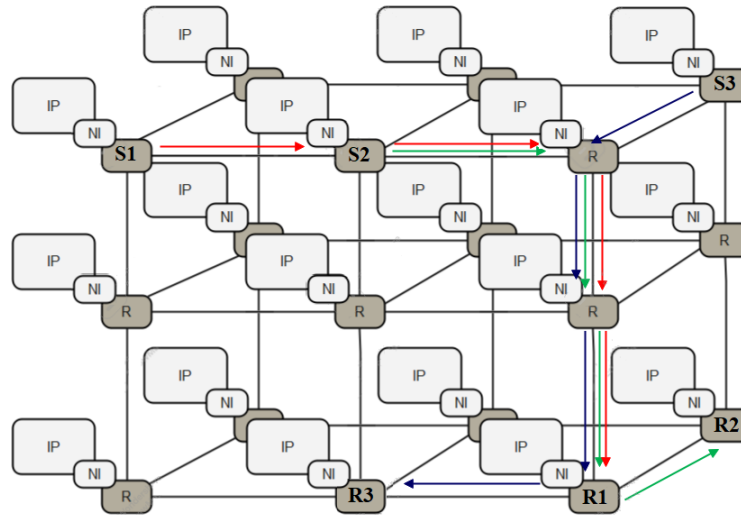


FIGURE 6.10 – Exemple de routage avec congestion

D'autre part, d'autre type d'algorithmes de routage peuvent être utilisés pour éviter les situations de congestion du réseau. De tels algorithmes utilisent non seulement la position des nœuds d'origine et de destination, mais également l'état de charge actuel du réseau pour calculer les chemins les moins congestionnés. Lors de la recherche d'une région du réseau en cours d'utilisation, le routage peut définir le paquet pour qu'il suive un autre chemin. Ce chemin sans congestion peut toutefois ne pas être minimal. Comme le montre l'exemple de la Figure 6.11.

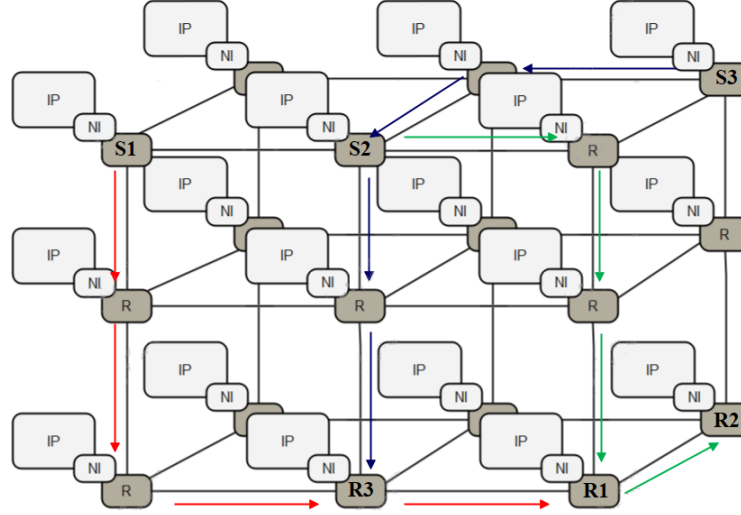


FIGURE 6.11 – Exemple de routage sans Congestion

La congestion, peut être considérée comme un coût que nous avons défini pour estimer un taux d'utilisation des liens du réseau. Initialement, les coûts de congestion de tous les liens sont égaux à 0. À chaque fois qu'un lien apparaît dans un chemin, son coût est incrémenté de 1. Le coût de congestion d'un chemin de longueur L est donné par l'Equation 6.28.

$$Congestion_chemin_i = \sum_{j=1}^{liens(i)} C_{i,j} \quad (6.28)$$

avec $liens(i)$ le nombre de liens utilisés dans le chemin i et $C_{i,j}$ le coût de congestion associé au $j^{ième}$ lien du chemin i dans la table de routage. Le coût total de congestion sera calculé selon l'Equation 6.29.

$$Congestion(S) = \sum_{i=1}^l Congestion_chemin_i \quad (6.29)$$

6.4 Considération finale du chapitre

Dans ce chapitre, nous avons présenté dans un premier temps notre modélisation pour résoudre le problème d'affectation des tâches aux ressources, et les objectives utilisés pour l'évaluation dans un modèle de simulation. Par la suite, nous avons montré comment le résultat de ce dernier sera utilisé pour l'étape suivante qui est le placement, ainsi que sa modélisation avec ses caractéristique et ses objectives et les algorithmes utilisés pour l'évaluation. Ces objectifs qui sont le temps d'exécution, l'énergie et la superficie ont été appliqués dans les deux phases l'affectation et le

placement, mais d'une manière différente. La dernière étape consistait à modéliser la phase de routage et à présenter ses caractéristiques et ses objectifs afin de les appliquer au problème.

Dans le chapitre 7, notre modélisation sera utilisée lors de l'exploration des solutions pour obtenir des résultats. Un ensemble de tests seront réalisés à la fin pour évaluer l'efficacité du modèle utilisé et les métaheuristiques proposées.

Chapitre 7

Tests et résultats

Après avoir implémenté les différentes techniques étudiées dans le Chapitre 6, nous présentons les résultats obtenus lors de nos expérimentations. Nous commençons par présenter les différents benchmarks sur lesquels nous avons effectué nos tests. Ensuite nous étudions les résultats obtenus suite à la réalisation des tests sur ces benchmarks.

7.1 Benchmarks utilisés

Les benchmarks sont des graphes de tâches, utilisés pour effectuer les tests, nous en avons utilisé trois types de benchmarks:

- (a) Applications embarquées de différentes complexités obtenues à partir de la suite de synthèse pour les systèmes embarqués E3S.
- (b) Graphes de tâches de différentes complexités générés en utilisant l'outil TGFF (*Task Graph For Free*) [194].
- (c) Différents Schémas de routage.

7.1.1 Suite de synthèse pour les systèmes embarqués

Cette suite de benchmarks contient des graphes de tâches pour cinq catégories d'applications embarquées: Automobile/industriel, grand public, réseaux, bureautique et télécommunications. Chacune de ces catégories est caractérisée par un ensemble de graphes de tâches spécifiques, avec trois versions distinctes pour chaque type: Système distribué, client-serveur sans fil et système sur puce.

Dans nos recherches, nous n'utilisons que les graphes de tâches conçus pour les systèmes sur puce. Cette suite de benchmarks est basée sur la suite de EEMBC (*Embedded Microprocessor Benchmark Consortium*) [195]. Elles contiennent 17 processeurs embarqués, qui se caractérisent par: Le temps d'exécution, la consommation d'énergie (mesurés sur 46 tâches), la taille sur la puce, le prix, la fréquence de fonctionnement et d'autres informations. Les cinq catégories d'applications embarquées sont:

- (a) Automotive/industrial: Cette application se constitue de 4 graphes de tâches de communication, afin de modéliser un système automobile intégré.
- (b) Consumer: Cette application est composée de deux graphes de tâches, pour le traitement d'images.
- (c) Networking: Ce modèle d'applications est utilisé pour le routage dans le réseau internet. Pour construire la table de routage.
- (d) Office automations: Elle se constitue d'un seul graphe de tâches pour le traitement des images et de textes.
- (e) Télécommunications: Ces applications se constituent de 5 graphes de tâches pour le traitement de signal.

La Table 7.1 présente les cinq catégories d'applications, le nombre de tâches, le nombre d'arcs ainsi que le nombre d'allocations possibles (nombre de solutions possibles).

TABLE 7.1 – Détail des benchmarks des cinq types d'application

Application	Benchmark ID	S	P	#Tâche	#Arcs	#A
auto-indust-tg0	1	✓		6	4	1183744
auto-indust-tg1	2	✓		4	3	18496
auto-indust-tg2	3		✓	9	9	606076928
auto-indust-tg3	4	✓		5	4	8704
consumer-tg0	5	✓		7	8	2247264
consumer-tg1	6		✓	7	5	176868
networking-tg1	7	✓		4	3	41616
networking-tg2	8	✓		4	3	41616
networking-tg3	9	✓		4	3	41616
Office-tg0	10		✓	5	5	210681
Telecom-tg0	11	✓		4	4	56644
Telecom-tg1	12		✓	6	6	9516192
Telecom-tg2	13		✓	6	6	9516192
Telecom-tg3	14	✓		3	2	4046
Telecom-tg4	15	✓		3	2	3468

Notons que la lettre S indique que l'application est représentée par un graphe séquentiel, tandis que la lettre P indique qu'elle est parallèle.

7.1.2 Graphe de tâches gratuit

L'outil TGFF, développé par R.P. Dick et D.L. Rhodes [194], facilite la standardisation des benchmarks aléatoires pour le problème d'ordonnancement, la recherche d'allocation et la recherche coopérative hardware-software. Cet outil permet à l'utilisateur de générer des graphes de tâches de taille théoriquement illimitée de manière aléatoire. Avec TGFF, les nœuds du graphe sont considérés comme des tâches et les arcs du graphe représentent la communication entre les tâches (nœuds). Chaque nœud du graphe peut également être vu comme un processeur *IP*, tandis que chaque arc du graphe peut être considéré comme une ressource de communication. De plus, chaque arc peut être associé à un numéro, représentant la quantité de communication. L'outil TGFF permet à l'utilisateur de spécifier le nombre de tâches qu'un graphe doit posséder, ainsi que le degré maximum d'entrée et de sortie des nœuds du graphe.

Ce type de graphes convient à plusieurs applications nécessitant la production de graphes de pseudo-aléatoire [196]. Nous avons généré 15 graphes de tâches avec TGFF. Ils sont représentés par la Table 7.2.

TABLE 7.2 – Détail des graphes de tâches

Graphe	Benchmark ID	#Tâche	#Arcs	#A
TG0	15	16	22	$56.581 * 10^{19}$
TG1	16	18	23	$60.810 * 10^{21}$
TG2	17	12	16	$10.023 * 10^{15}$
TG3	18	20	27	$357.563 * 10^{23}$
TG4	19	24	32	$127.581 * 10^{29}$
TG5	20	32	39	$27.306 * 10^{42}$
TG6	21	64	93	$322.200 * 10^{79}$
TG7	22	40	51	$577.247 * 10^{48}$
TG8	23	54	74	$406.906 * 10^{69}$
TG9	24	47	69	$400.564 * 10^{59}$
TG10	25	75	100	$340.760 * 10^{96}$
TG11	26	87	118	$999.536 * 10^{110}$
TG12	27	93	123	$433.529 * 10^{119}$
TG13	28	67	86	$513.443 * 10^{84}$
TG14	29	117	176	$759.028 * 10^{148}$

7.1.3 Schéma de routage

L'utilisation des Schémas de routage est une pratique courante dans l'évaluation des algorithmes de routage dans les NoCs. Leur comportement varient en fonction des algorithmes, utilisant différents types de modèles de trafic. Afin d'évaluer les NoCs, les chercheurs ont utilisé différents générateurs de schémas de routage qui

représentent différents types de modèles de trafic. Ceux-ci sont largement utilisés pour l'analyse de la consommation d'énergie et de la latence dans les réseaux. Deux classes de schémas existent:

- (a) Schéma aléatoire: Dans ces cas, tous les nœuds sont choisis de manière aléatoire. Nous citons:
 - Rand: dans ce modèle, chaque nœud peut être utilisé comme source ou destination avec la même probabilité.
 - Point chaud (*Hotspot*): Dans ce point chaud, tous les nœuds peuvent être sélectionnés comme source avec la même probabilité. Cependant, certains nœuds hotspots, ont plus de chance d'être sélectionnés comme destination.
 - Local: Dans ce modèle, tous les nœuds peuvent être sélectionnés comme source, mais seuls les nœuds voisins peuvent être sélectionnés au hasard comme destination.
- (b) Schéma déterministes: le nœud de destination est déterminé en fonction de la position du nœud source. Nous citons:
 - Transpose: Chaque nœud envoie des données uniquement vers une destination avec la transposée de sa propre adresse dans le réseau.
 - Complément: Chaque nœud envoie des messages uniquement à celui qui est le complément de sa propre adresse dans le réseau.

7.2 Caractéristiques de la machine utilisée

Dans le cadre de cette étude, tous les tests ont été réalisés sur une machine équipée d'un processeur Intel Core i5- 2520M, cadencé à 2.5 GHz avec 2 cœurs physiques et quatre threads. La machine est équipée de 12 Go de mémoire RAM et fonctionne sous le système d'exploitation Windows 10 Professionnel. Les tests ont été effectués en utilisant l'environnement de développement Netbeans IDE 8.2. Pour le test du routage le simulateur utilise la même machine sous le système d'exploitation linux (ubuntu).

7.3 Résultats de la phase d'affectation

A ce niveau, et avant de présenter les résultats obtenu par l'implémentation des deux algorithmes, MOPSO et DEMO, un nombre de tests a été effectué avant d'obtenir les vrais paramètres de chaque algorithm. Pour MOPSO, les paramètres c_1 , c_2 , et w sont étudiés. Ils sont fixés à des valeurs spécifiques pour produire de bons résultats lors de la simulation: w est fixé à 0.6, c_1 et c_2 sont tous deux fixés à 2, la taille de l'essaim

est de 1000, et le nombre d'itérations est de 2000. Pour DEMO, les paramètres F et CR sont étudiés. Une analyse de sensibilité est réalisée pour la constante de mutation F et CR. Les paramètres suivants sont trouvés pour donner de bons résultats: la population initiale est fixée à 100, CR est fixé à 0.1 et F est fixé à 0.5, et le nombre maximum d'itérations est fixé à 200.

7.3.1 Résultats de MOPSO

Parmi les 14 applications courantes présentées dans la Table 7.1, nous avons utilisé 5 graphes de tâches parallèles [197]. Dans ces graphes, l'ordonnancement joue un rôle crucial dans la détermination du temps d'exécution. De plus, les 15 graphes de tâches présentés dans la Table 7.2 générés par TGFF ont également été testés. Pour cette raison, les trois algorithmes d'ordonnancement présentés dans le Chapitre 6 sont utilisés uniquement sur les applications dont les numéros sont les suivants: 3, 6, 10, 12, 13 et 16-30. Le reste des applications correspondent à des graphes de tâches séquentiels

Les résultats des affectations obtenus avec les stratégies d'ordonnancement ASAP, ALAP et LS sont présentés dans les Figures 7.1, 7.2 et 7.3, respectivement, concernant les objectifs de puissance, de temps d'exécution et de surface. Ces résultats sont publiés dans [198] et sont évalués selon les trois objectifs mentionnés, mesurant l'énergie en watts, le temps en 10^{-3} secondes et la surface en 10^{-6} mètre carré. Les figures illustrent la meilleure valeur obtenue pour chaque objectif.

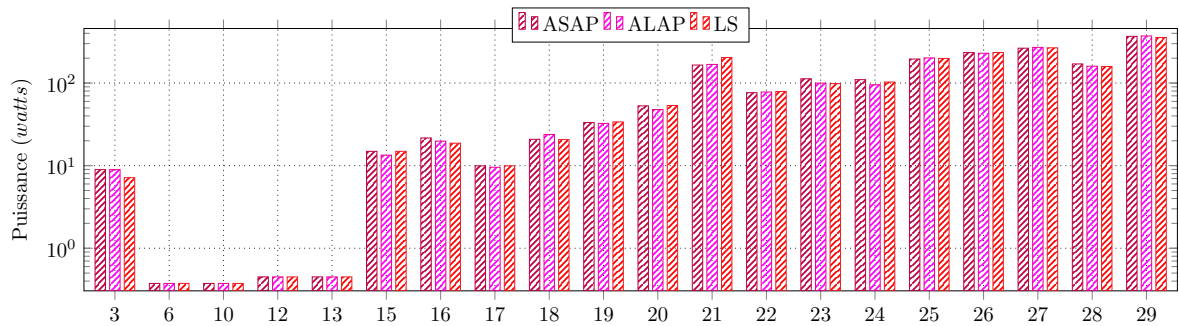


FIGURE 7.1 – Comparaison des affectations obtenues par MOPSO concernant l'énergie

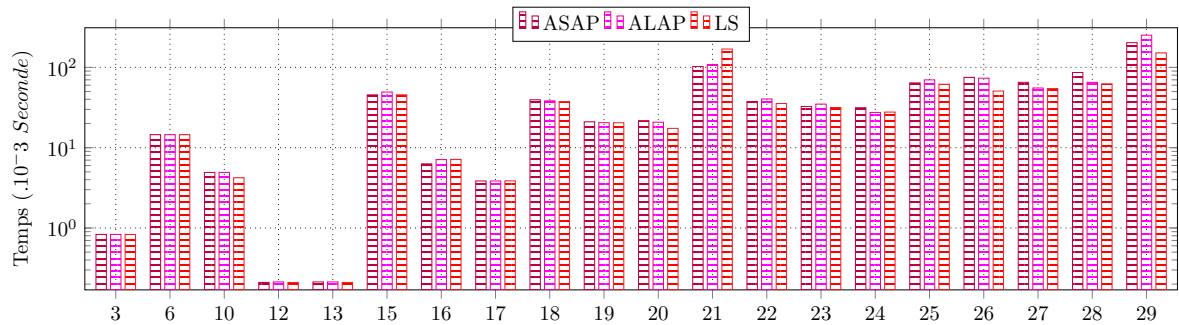


FIGURE 7.2 – Comparaison des affectations obtenues par MOPSO concernant le temps

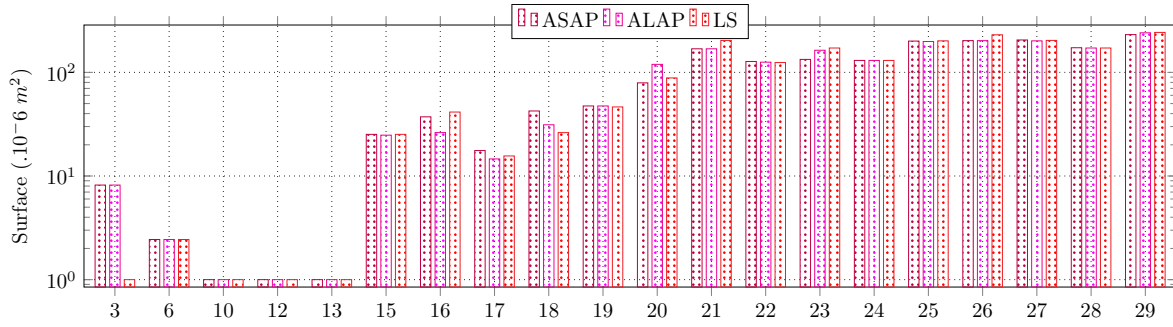


FIGURE 7.3 – Comparaison des affectations obtenues par MOPSO concernant la surface

Les résultats obtenus mettent en évidence que l’algorithme d’ordonnancement LS permet d’obtenir une meilleure affectation dans la plupart des cas de simulations et montre une plus grande efficacité par rapport aux deux autres algorithmes d’ordonnancement ALAP et ASAP.

En conséquence, dans le reste de cette étude, nous avons choisi de maintenir l’algorithme LS comme l’algorithme d’ordonnancement par défaut pour les tests ultérieurs. Les Tables 7.3 et 7.4 présentent les résultats obtenus avec l’affectation des IPs en utilisant l’algorithme MOPSO appliqué à tous les benchmarks de la suite E3S ainsi que les graphes de tâches générés par TGFF.

TABLE 7.3 – Résultats d’affectation des IPs avec MOPSO pour les benchmarks de E3S

App ID	solutions	IPs			Meilleur Affectation			Affectation Moyenne		
		Minimum	Maximum	Moyenne	Puissance	Temps	Surface	Puissance	Temps	Surface
1	3	1	2	1.6	4.15	0.0221	1.0	4.83	0.0221	2.14
2	3	1	2	1.6	2.15	0.0475	1.0	2.83	0.0475	2.14
3	8	2	4	3.25	8.34	0.832	8.18	17.08	0.852	42.84
4	3	1	2	1.6	3.15	0.0395	1.0	3.83	0.0395	2.14
5	12	1	3	2.3	0.525	22.12	2.44	5.34	54.63	9.48
6	10	1	3	2.3	0.375	14.52	2.44	3.395	44.973	8.39
7	14	1	3	2	0.3	0.282	2.44	2.48	1.459	17.90
8	14	1	3	2	0.3	0.311	2.44	2.48	1.88	17.9
9	15	1	3	2	0.3	0.39	2.44	2.38	2.794	19.43
10	33	1	4	2.36	0.375	4.22	1.0	4.72	401.6	14.49
11	8	1	3	1.75	0.3	0.0245	1.0	3.008	0.347	3.008
12	32	2	4	2.625	1.145	0.213	2	4.164	2.16	10.65
13	25	2	4	2.96	1.049	0.209	2	3.88	1.974	10.97
14	6	1	2	1.5	0.224	0.029	1.0	2.11	0.277	1.98

La Table C.11 décrite dans l’Annexe C évoque les allocations non dominées, définies par MOPSO, pour un échantillon d’applications *E3S*. Parmi les 14 applications utilisées, cinq benchmarks, utilisés, dans l’ordonnancement, ont été sélectionnés. La présence du symbole \times dans une des dernières colonnes indique que la solution de l’allocation indiquée a permis d’atteindre la valeur minimale de surface (S), de consommation d’énergie (P) et de temps (T).

TABLE 7.4 – Resultats d’Affectation des IP avec MOPSO pour les benchmarks de TGFF

App ID	solutions	IPs			Meilleur Affectation			Affectation Moyenne		
		Minimum	Maximum	Moyenne	Puissance	Temps	Surface	Puissance	Temps	Surface
15	58	5	10	7.603	13.395	49.328	24.778	38.511	272.505	98.464
16	43	4	10	7.906	19.805	7.111	26.225	58.863	32.675	135.003
17	60	4	9	6.833	9.560	3.850	14.615	29.883	114.792	82.384
18	48	6	12	8.979	23.850	38.296	31.211	65.588	123.211	160.048
19	60	7	13	9.433	32.480	20.480	47.402	76.408	90.305	154.729
20	54	9	14	11.407	47.830	20.771	119.452	116.731	51.555	227.714
21	90	10	16	13.211	168.125	108.225	168.823	272.773	579.919	263.743
22	92	9	15	11.793	77.910	40.446	125.397	142.692	157.721	232.906
23	45	11	16	13.511	99.839	34.895	163.442	184.094	267.171	257.295
24	81	10	15	12.913	95.394	27.483	130.383	181.478	145.844	257.686
25	63	11	16	13.968	201.414	69.903	198.447	319.903	562.617	293.007
26	81	12	16	14.259	229.490	73.343	202.440	370.410	409.856	305.751
27	95	12	17	14.694	270.334	55.358	201.440	415.590	602.240	322.112
28	97	11	16	13.587	160.054	65.121	171.816	271.611	229.562	286.744
29	96	13	17	14.625	371.829	251.852	240.880	537.986	1225.420	321.340

7.3.2 Résultats de DEMO

Pour l’algorithme d’allocation DEMO cinq variants de mutation ont été utilisés pour les tests. La Table 7.5 montre les résultats obtenus avec l’affectation des IPs, utilisant la méthode de mutation Rand_1, appliquée à tous les benchmarks de E3S. Notant que l’algorithme d’ordonnancement utilisé est LS. Les résultats des autres

TABLE 7.5 – Resultats d’Affectation des IPs avec DEMO Rand_1 pour les benchmarks de E33

App ID	solutions	IPs			Meilleur Affectation			Affectation Moyenne		
		Minimum	Maximum	Moyenne	Puissance	Temps	Surface	Puissance	Temps	Surface
1	3	2	2	2	4.15	0.0221	1.0	4.83	0.0222	2.14
2	3	2	2	2	2.15	0.0475	1.0	2.83	0.0474	2.14
3	8	2	3	2.62	7.15	0.832	1.0	8.208	0.833	7.38
4	3	2	2	2	3.15	0.0395	1.0	3.83	0.0395	2.14
5	16	2	3	2.87	1.745	22.12	6.432	6.147	38.657	9.33
6	12	2	4	3	3.44	0.47	14.52	4.009	21.44	9.0072
7	13	2	3	2.46	0.3	0.282	2.44	2.4	1.44	2.408
8	14	2	3	2.53	0.3	0.311	2.44	2.15	1.908	19.049
9	14	2	3	2.57	0.3	0.39	2.44	2.067	2.835	20.6
10	24	2	4	2.87	0.565	4.22	1.0	5.664	236.90	16.309
11	5	2	3	2.4	0.3	0.0245	2.0	2.373	0.0593	2.864
12	7	3	4	3.42	0.45	0.209	2.0	3.554	0.646	8.377
13	6	3	5	3.83	0.45	0.209	2.44	3.54	0.515	3.2
14	5	2	2	2	0.224	0.029	1.0	1.993	0.1894	2.176

stratégies de mutation de DEMO sont listés dans l’Annexe C. Nous avons regroupé ces résultats dans trois figures, déjà publiées dans [199]. La Figure 7.4 permet une comparaison visuelle de la meilleure consommation d’énergie des affectations utilisant les stratégies de mutation comparées. De plus, la Figure 7.5 offre une meilleure comparaison visuelle des résultats concernant le temps d’exécution. Enfin, la Figure 7.6 montre la comparaison de la surface par les affectations obtenues par les stratégies comparées. Afin de comparer les opérations de mutations, différentes mesures de performance existent pour la comparaison. Nous avons choisi d’utiliser l’indicateur

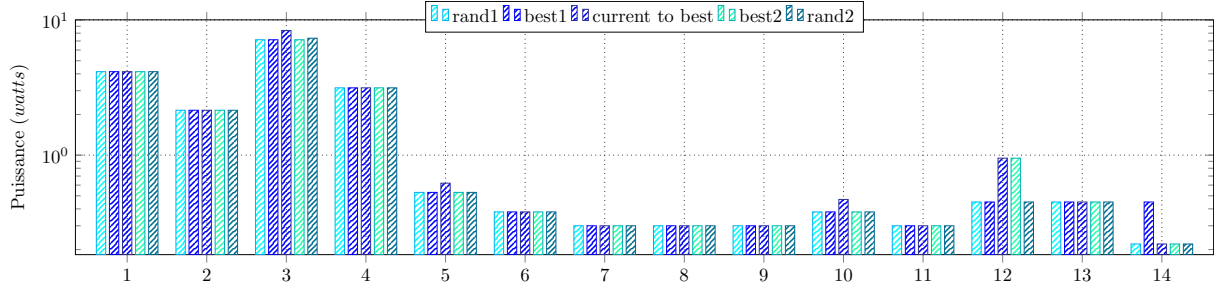


FIGURE 7.4 – Comparaison des affectations obtenues par DEMO concernant l'énergie

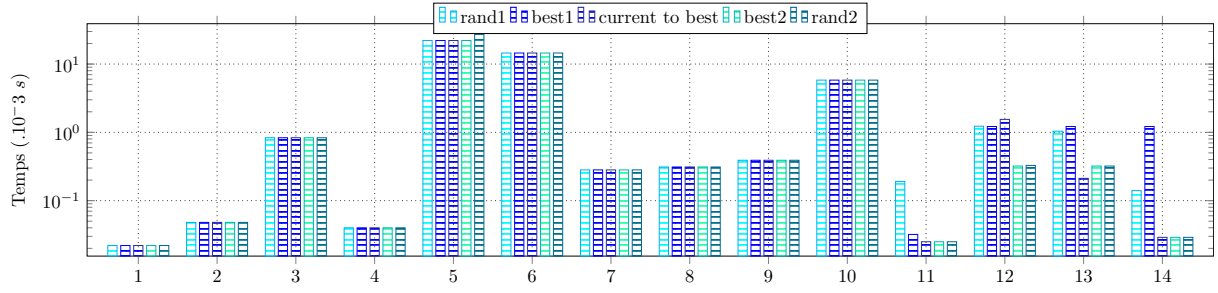


FIGURE 7.5 – Comparaison des affectations obtenues par DEMO concernant le temps

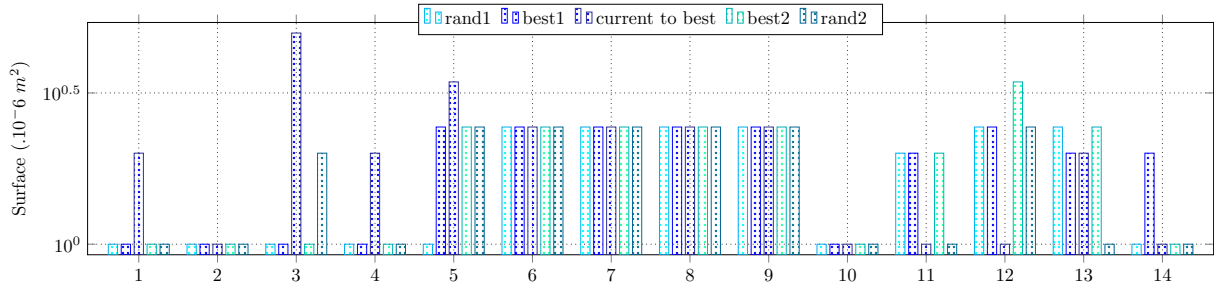


FIGURE 7.6 – Comparaison des affectations obtenues par DEMO concernant la surface

de convergence nommé Contribution. Par ailleurs il permet de détecter, entre deux fronts pareto, lequel participe mieux et avec quels pourcentage dans le vrai front pareto. Cet indicateur est présenté dans le Chapitre 4. Les différentes stratégies de Mutation dans DEMO sont notées comme: A=DEMO/rand/1/, B=DEMO/rand/2/, C=DEMO/current-to-best/1/, D=DEMO/best/1/et E=DEMO/best/2/. Dans la Figures 7.7, les cinq variantes de l'opération de mutation sont comparées. Elles montrent la contribution entre deux stratégies. La valeur la plus élevée, indique les meilleures performances. On peut observer que les stratégies A (Rand_1) et B (Best_1) sont les plus performantes dans la plupart des applications. Dans le benchmark 11 la stratégie A (Rand_1) est plus performante que les autres. En ce qui concerne les benchmarks 2, 4 et 5, toutes les stratégies de mutation sont équivalentes. La Table 7.6 présente les résultats de l'affectation selon la stratégie de mutation Rand_1 appliquée au graphe de tâches généré par l'outil TGFF. Les résultats de l'affectation des IPs par l'utilisation de DEMO avec la stratégie Best_1, présentés dans la Table C.10, sont inclus dans l'Annexe C.

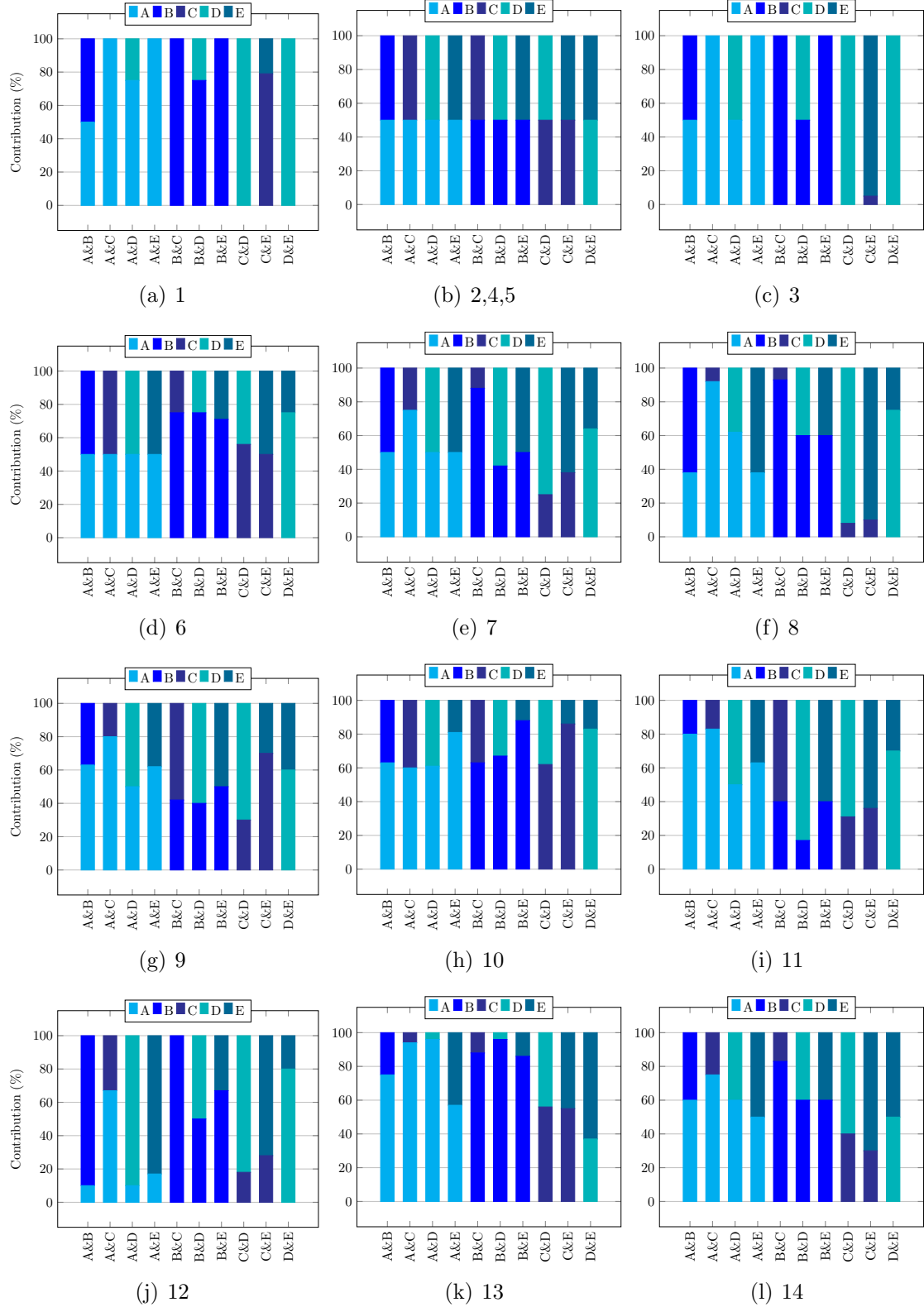


FIGURE 7.7 – Valeurs de contribution obtenue par les cinq version de DEMO

La Table C.12 présenté dans l'Annexe C montre les allocations non dominées trouvées par DEMO pour un échantillon d'applications de E3S. Parmi les 16 applications utilisées, Les cinq benchmarks parallèles ont été sélectionnés.

TABLE 7.6 – Resultats d’Affectation des IPs avec DEMO Rand_1 pour les benchmark de TGFF

App ID	solutions	IPs			Meilleur Affectation			Affectation Moyenne		
		Minimum	Maximum	Moyenne	Puissance	Temps	Surface	Puissance	Temps	Surface
15	85	5	8	6.729	5.500	45.204	7.433	17.781	105.730	40.502
16	95	4	8	6.116	11.255	6.581	7.433	18.681	89.711	35.665
17	87	4	8	6.057	5.200	3.764	4.440	10.218	155.535	30.814
18	72	5	9	7.125	9.985	37.054	10.622	18.622	68.578	34.207
19	63	5	9	7.825	13.575	19.373	10.622	20.364	35.180	30.728
20	20	7	12	9.850	22.530	18.951	17.608	33.954	64.314	62.871
21	21	11	14	12.333	105.280	86.730	88.393	143.201	146.830	208.926
22	24	9	13	11.125	39.765	36.620	25.225	52.921	51.151	88.499
23	12	12	14	12.667	64.475	33.481	84.843	84.686	255.640	149.914
24	16	9	14	11.375	46.505	26.160	46.403	70.303	185.732	93.358
25	20	7	12	9.850	22.530	18.951	17.608	33.954	64.314	62.871
26	21	11	14	12.333	105.280	86.730	88.393	143.201	146.830	208.926
27	24	9	13	11.125	39.765	36.620	25.225	52.921	51.151	88.499
28	12	12	14	12.667	64.475	33.481	84.843	84.686	255.640	149.914
29	16	9	14	11.375	46.505	26.160	46.403	70.303	185.732	93.358

7.3.3 Comparaison

Pour renforcer l’approche proposée par les algorithmes MOPSO et DEMO, nous avons comparé les résultats obtenus à ceux rapportés dans la référence [55]. Dans cette référence, les algorithmes NSGA-II et MicroGA ont été utilisés pour générer des affectations. Nous avons utilisé les solutions non dominées, produites par chaque algorithme afin de les comparer. Cette approche permet d’évaluer la performance des algorithmes MOPSO et DEMO par rapport à des méthodes déjà établies, et largement utilisées dans la littérature. Pour DEMO, étant donné que les deux stratégies de mutation, Rand_1 et Best_1, sont les meilleures, seule la stratégie Best est utilisée dans la comparaison.

La Figure 7.8 illustre le nombre total de solutions non dominées obtenues pour chaque front pareto dans le cadre de chaque benchmark, en utilisant les deux approches, MOPSO et DEMO.

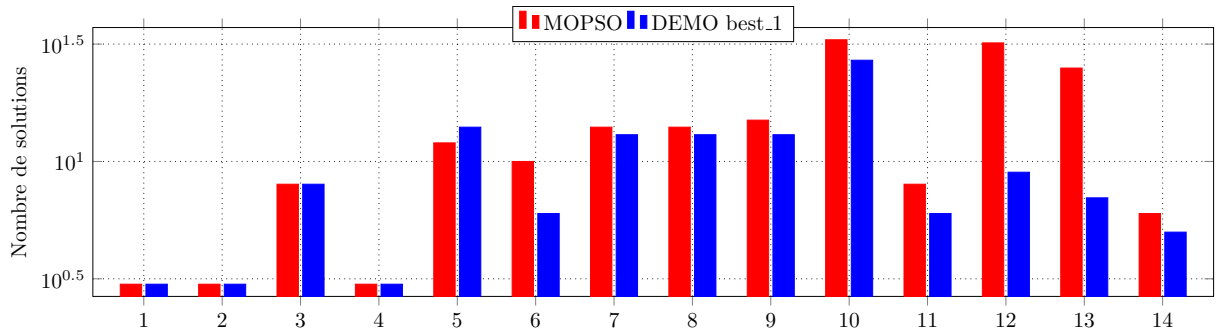


FIGURE 7.8 – Solutions non dominées obtenues par graphes de tâche de E3S

Les Figures 7.9, 7.10 et 7.11 illustrent les résultats optimaux obtenus pour chacune des méthodes, en ce qui concerne les trois objectifs l’énergie, la surface et le temps d’exécution.

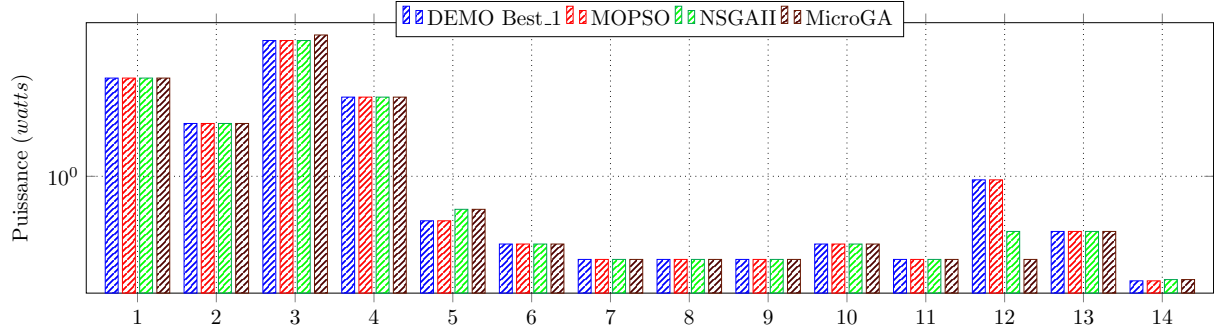


FIGURE 7.9 – Analyse comparative des affectations obtenue sur les graphes de tâches E3S par DEMO et MOPSO concernant l'énergie

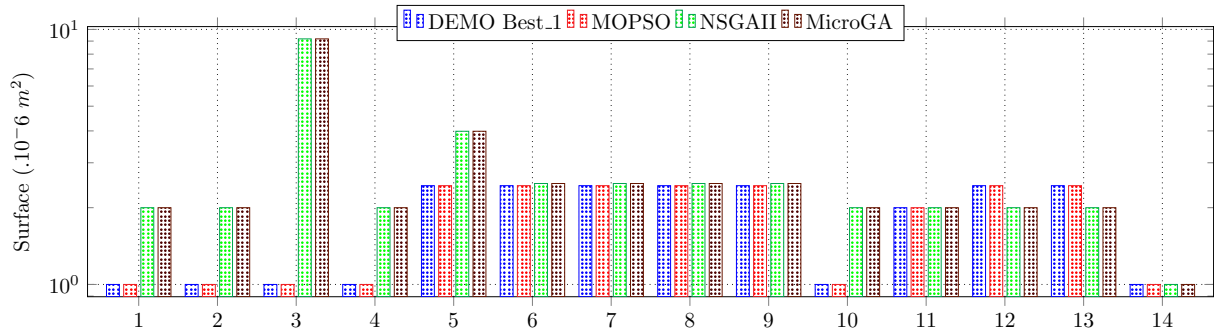


FIGURE 7.10 – Analyse comparative des affectations obtenue sur les graphes de tâches E3S par DEMO et MOPSO concernant la surface

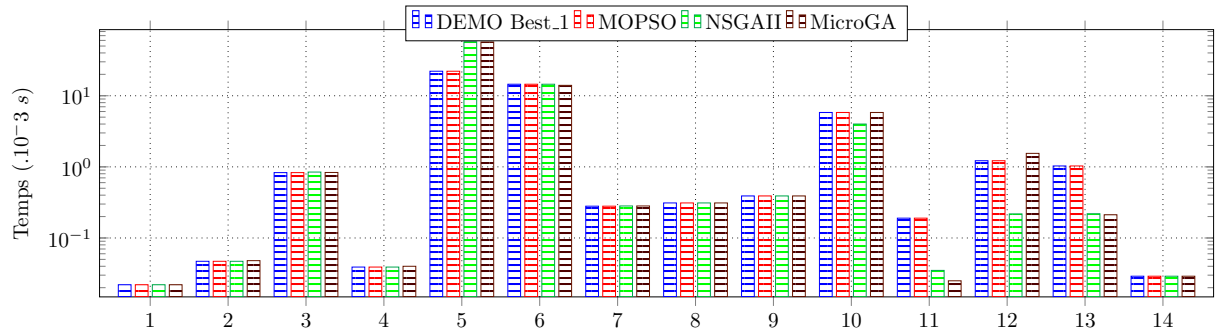


FIGURE 7.11 – Analyse comparative des affectations obtenue sur les graphes de tâches E3S par DEMO et MOPSO concernant le temps

La Figure 7.12 présente le nombre total de solutions non dominées obtenues pour chaque front pareto dans le cadre de chaque graphe de tâches généré par TGFF, en utilisant les deux algorithmes, MOPSO et DEMO.

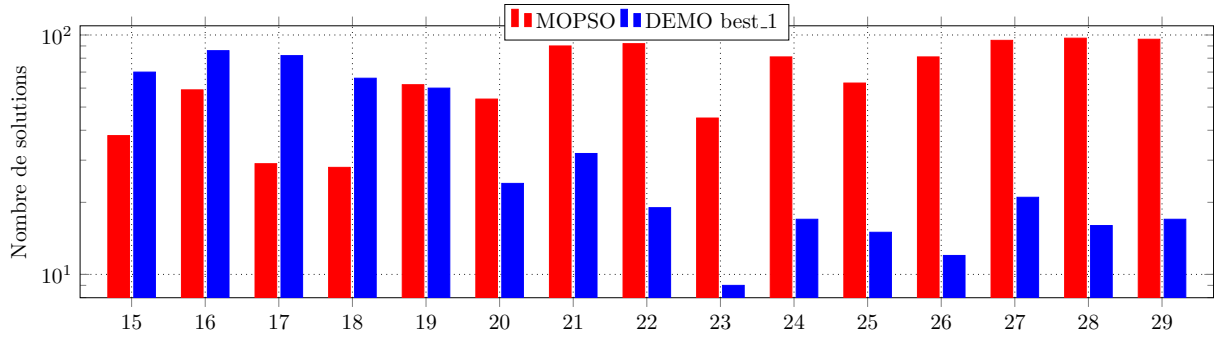


FIGURE 7.12 – Solutions non dominées obtenues sur les graphes de tâches de TGFF

Les Figures 7.13, 7.14 et 7.15 présentent les résultats atteints par chacune des méthodes, en mettant en évidence leurs performances en termes de consommation d'énergie, de surface et de temps d'exécution

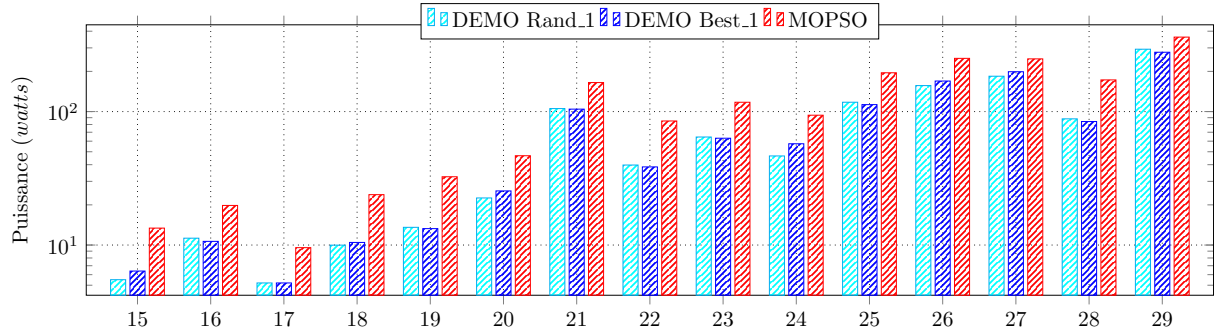


FIGURE 7.13 – Comparaison des affectations obtenue sur les graphes de tâches TGFF par DEMO et MOPSO concernant l'énergie

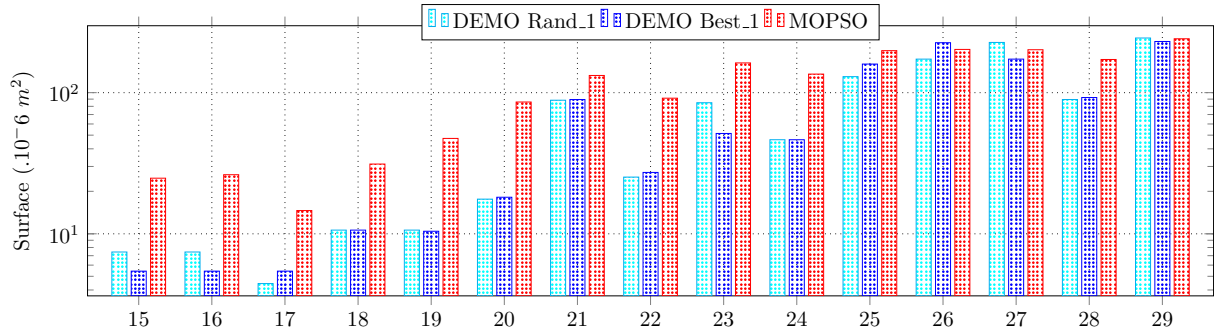


FIGURE 7.14 – Comparaison des affectations obtenue sur les graphes de tâches TGFF par DEMO et MOPSO concernant la surface

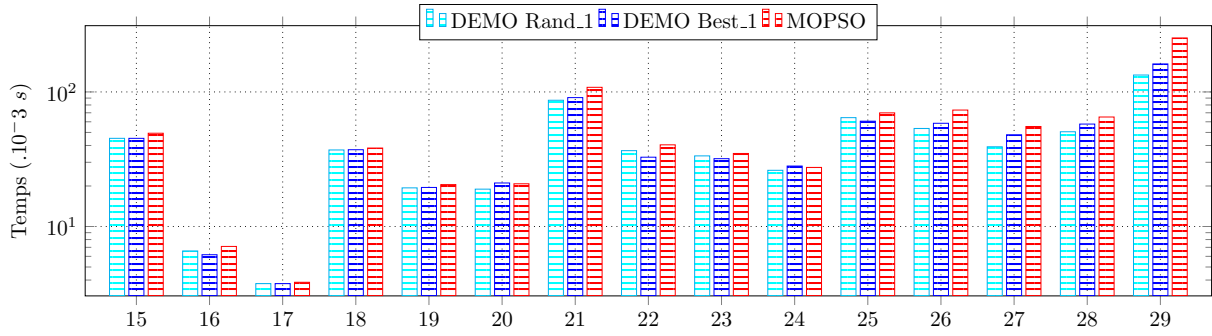


FIGURE 7.15 – Comparaison des affectations obtenue sur les graphes de tâches TGFF par DEMO et MOPSO concernant le temps d'exécution

Tout d'abord, l'algorithme DEMO a obtenu de meilleurs résultats que MOPSO pour la plupart des benchmarks. Cela indique que DEMO a réussi à trouver des solutions plus performantes en termes de consommation d'énergie, de surface et de temps d'exécution par rapport à MOPSO. De plus, les résultats obtenus par nos deux algorithmes (DEMO et MOPSO) sont meilleurs que ceux rapportés par les algorithmes NSGA-II et MicroGA dans la référence [200]. Cela démontre l'efficacité et la pertinence des approches basées sur l'optimisation multi-objectifs (DEMO et MOPSO) pour le problème d'allocation d'IPs. En ce qui concerne la comparaison entre DEMO et MOPSO, les deux algorithmes ont produit des résultats similaires. Cependant, DEMO s'est avéré être plus rapide que MOPSO pour trouver les résultats optimaux. Cela met en évidence les performances supérieures de DEMO par rapport à MOPSO en termes d'efficacité de recherche et de temps de calcul pour résoudre le problème d'allocation d'IPs.

Cependant, afin d'étudier en profondeur l'efficacité des deux algorithmes, nous avons analysé diverses métriques d'optimisation, dont la convergence, la divergence, et une métrique hybride mentionnée dans le Chapitre 4. Cette approche analytique fournit des informations détaillées sur le comportement de chaque algorithme. Ces métriques multiples contribuent à une compréhension approfondie des points forts et des points faibles de chaque algorithme dans des contextes variés. Les Table 7.7 et 7.8 présentent les valeurs obtenues pour chaque algorithme, MOPSO et DEMO, en ce qui concerne ces trois métriques, ainsi que le nombre de solutions non dominées trouvées. Afin d'améliorer la lisibilité, la Figure 7.16 illustre les résultats obtenus spécifiquement pour la métrique de contribution.

TABLE 7.7 – Métriques de performances obtenues pour l'affectation des benchmarks de E3S

Benchmark	solutions	MOPSO			solutions	DEMO		
		Contribution	Espacement	Hyper Volume		Contribution	Espacement	Hyper Volume
1	3	50	1.39	$1*10^{-10}$	3	50	1.72	$1*10^{-10}$
2	3	50	3.08	$1*10^{-10}$	3	50	1.72	$1*10^{-10}$
3	12	0	2.76	$8.6*10^{-5}$	12	100	1.24	$2.3*10^{-9}$
4	3	50	1.73	$1*10^{-10}$	3	50	1.73	$1*10^{-10}$
5	10	13	53.10	$3.2*10^{-5}$	14	87	15.57	$355*10^{-6}$
6	8	0	18.92	$1.7*10^{-5}$	5	100	5.62	$5.4*10^{-8}$
7	14	53	5.49	$1.8*10^{-6}$	12	47	2.41	$3.9*10^{-6}$
8	14	57	7.10	$2.1*10^{-6}$	12	43	4.49	$3.7*10^{-6}$
9	15	57	4.3	$2.7*10^{-6}$	13	43	6.04	$2.2*10^{-5}$
10	36	46	46.68	$5.86*10^{-2}$	28	54	85.82	$5.57*10^{-2}$
11	8	62	6.96	$1.3*10^{-7}$	6	37	3.96	$3.7*10^{-7}$
12	12	12	10.48	$1.05*10^{-4}$	9	88	14.61	$8.1*10^{-6}$
13	18	14	9.27	$1.73*10^{-4}$	11	86	5.33	$8.8*10^{-6}$
14	6	67	3.83	$5.4*10^{-8}$	4	33	1.83	$4.57*10^{-6}$

TABLE 7.8 – Métriques de performances obtenues pour l'affectation des benchmarks de TGFF

Benchmark	solutions	MOPSO			solutions	DEMO		
		Contribution	Espacement	Hyper Volume		Contribution	Espacement	Hyper Volume
15	38	0.00	343.128	0.545	70	100.00	89.611	0.040
16	59	0.00	2240.208	0.551	86	100.00	161.035	0.980
17	29	0.00	17.988	0.000019	82	100.00	10.875	0.006
18	28	0.00	1781.510	0.137	66	100.00	36.450	0.033
19	62	0.00	3820.424	0.388	60	100.00	14.557	0.0029
20	54	20.69	2262.469	0.226	24	79.31	52.475	0.0007
21	90	0.00	5956.719	5.797	32	100.00	1218.352	0.037
22	92	0.00	2203.557	0.656	19	100.00	586.877	0.008
23	45	0.00	4540.391	0.757	9	100.00	1429.829	0.011
24	81	10.53	5152.411	2.792	17	89.47	709.022	0.063
25	63	0.00	9342.037	3.777	15	100.00	2333.614	0.070
26	81	25.00	9665.748	3.263	12	75.00	2735.646	0.023
27	95	0.00	11733.066	5.397	21	100.00	6473.030	0.399
28	97	0.00	9497.889	1.215	16	100.00	1316.258	0.000892
29	96	11.11	15130.988	8.093	17	88.89	4754.644	0.018

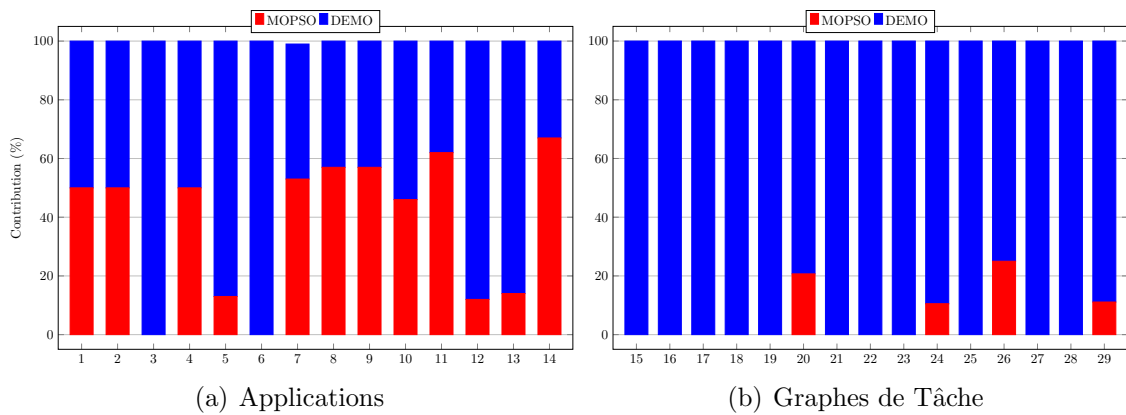


FIGURE 7.16 – Valeurs de contribution obtenue par DEMO et MOPSO

Selon les résultats obtenus, l'algorithme DEMO et son impact par rapport à l'algorithme MOPSO sont significatifs. Dans la plupart des tests, il démontre des performances élevées en termes de convergence (métrique de contribution) et de

rapidité pour atteindre le front pareto optimal. Cependant, ses résultats révèlent une distribution moins satisfaisante par rapport à la métrique de diversité (hypervolume et espacement). En revanche, MOPSO assure une bonne diversité des résultats, tandis que DEMO concentre ses résultats sur une petite surface. MOPSO présente une diversité importante des résultats obtenus, avec une métriques de convergence jugées faibles par rapport à DEMO.

Dans le contexte de l'allocation d'IP, les résultats obtenus mettent en évidence que, malgré des solutions d'allocation différentes générées par les algorithmes MOPSO et DEMO, celles-ci ne présentent pas de différences significatives du point de vue des objectifs évalués. Graphiquement, ces solutions se situent au même point dans l'espace des objectifs, réduisant ainsi le nombre de points distincts sur le front pareto. Cette convergence résulte de la nature discrète de l'espace de recherche du problème d'affectation, limitant les choix possibles à un ensemble déterminé de solutions discrètes. Bien que la distinction d'un front pareto clair soit difficile, les deux algorithmes ont réussi à fournir des solutions de qualité, améliorant ainsi les performances par rapport à d'autres approches. Cela souligne l'efficacité des algorithmes MOPSO et DEMO dans la résolution complexe du problème d'allocation d'IP, malgré des différences potentielles dans leurs configurations finales.

7.4 Résultats de la phase de placement

Dans cette section, nous présentons les résultats obtenus par les implémentations de la méthode de placement des IPs dans la plateforme NoC en utilisant les algorithmes MOPSO et DEMO. Le NoC est basé sur une architecture 3D Mesh et 2D Mesh. Nous avons utilisé les 15 graphe de tâches créés à l'aide de l'outil TGFF. Rappelons aussi que l'algorithme d'ordonnancement utilisé est toujours LS.

Avant de présenter les résultats du placement, les résultats de l'étape d'affectation sont injectés pour être placés sur la plateforme NoC selon deux topologies, 2D et 3D. L'objectif de cette étude est d'analyser le comportement de l'utilisation des topologies 2D et 3D. Pour cette raison les résultat du placement seront présenter dans cette section seulement pour les benchmarks présentés dans la Table 7.2. Après plusieurs tentatives de tests, les paramètres ont été ajustés calibrer nos algorithme sont: Pour MOPSO, les paramètres c_1 , c_2 , et w sont étudiés. Ils sont fixés à des valeurs spécifiques pour produire de bons résultats lors de la simulation: w est fixé à 0.6, c_1 et c_2 sont tous deux fixés à 1.49, la taille de l'essaim est de 1000, et le nombre d'itérations est de 500. Pour DEMO, les paramètres F et CR sont étudiés. Une analyse de sensibilité est réalisée pour la constante de mutation F et CR. Les

paramètres suivants sont trouvés pour donner de meilleurs résultats: la population initiale est fixée à 200, CR est fixé à 0.9 et F est fixé à 0.5, et le nombre maximum d'itérations est fixé à 500.

7.4.1 Résultats de MOPSO

Nous présentons les résultats de placement lors de l'utilisation de l'algorithme MOPSO. Ces résultats sont déjà publiés dans [201]. Deux topologies sont utilisées dans les maillages 2D et 3D pour les tests. Les Figures 7.17, 7.18 et 7.19 présentent les résultats obtenus par l'algorithme MOPSO pour les graphes de tâches utilisés lors des tests. De plus, chaque figure indique les valeurs minimales atteintes pour la consommation d'énergie, la surface matérielle et le temps d'exécution pour chaque graphe de tâche.

Les valeurs minimales présentées dans les tables ne sont pas nécessairement issues de la même solution, mais représentent les résultats les plus bas trouvés parmi les solutions non dominées. Il est important de noter que les valeurs de puissance sont exprimées en *watts*, le temps est mesuré en millisecondes (10^{-3} secondes) et la surface est en mètres carrés (10^{-6}).

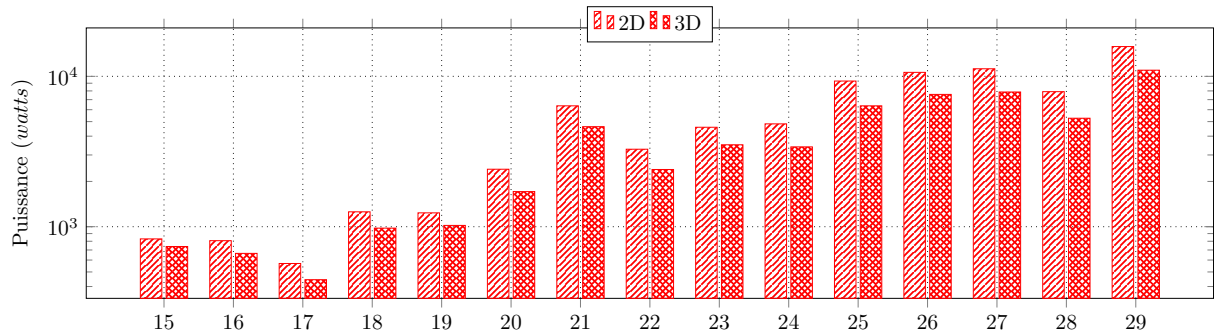


FIGURE 7.17 – Comparaison des placement obtenues par MOPSO concernant l'énergie

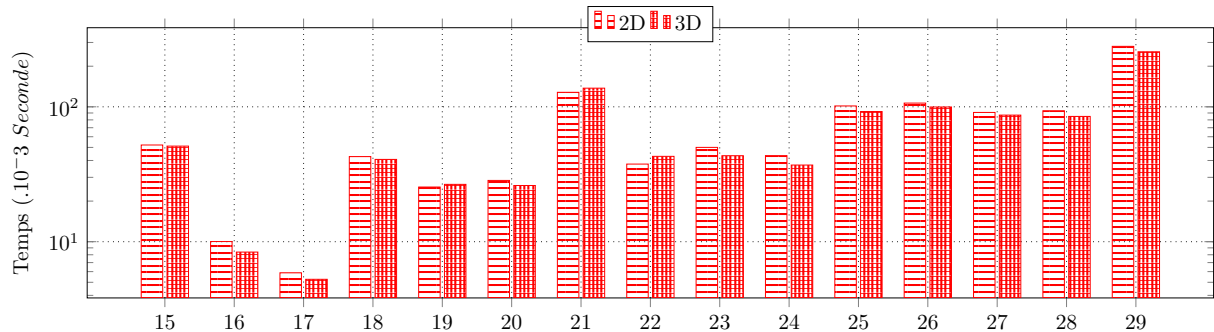


FIGURE 7.18 – Comparaison des placement obtenues par MOPSO concernant le temps

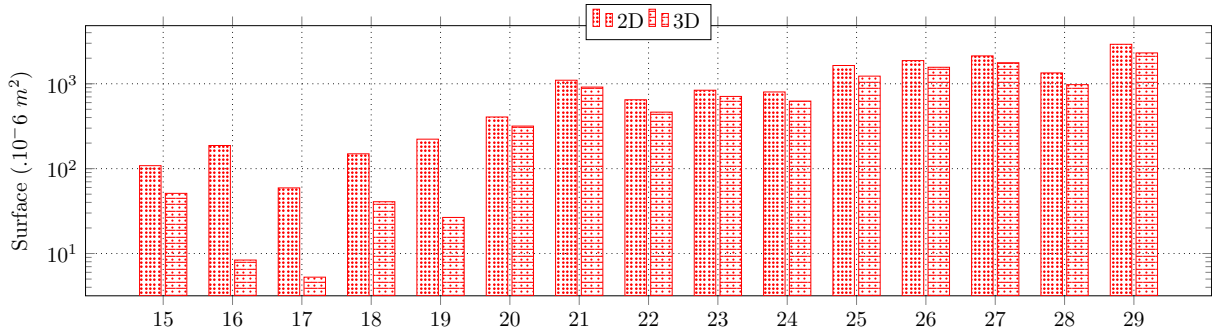


FIGURE 7.19 – Comparaison des placement obtenues par MOPSO concernant la surface

Il est clairement conclu que le placement dans un NoC 3D donne de meilleurs résultats que le NoC 2D en raison du grand nombre de solutions non dominées découvertes dans le maillage 3D par rapport au maillage 2D. Cette conclusion est étayée par les données fournies dans les graphiques. Ces résultats suggèrent que le maillage 3D offre des avantages significatifs en termes de consommation d'énergie, de surface et de performances temporelles pour le rendement de placement par rapport au maillage 2D.

Les Tables C.19 et C.20 présentées dans l'Annexe C montrent quelques solutions de placement non dominées trouvées par MOPSO pour les cinq graphes de tâches utilisés dans le test. Dans cette table, nous exposons la solution d'affectation ainsi que son emplacement dans le NoC 2D et 3D respectivement.

7.4.2 Résultats de DEMO

Dans cette étude, nous avons présenté des résultats différents pour les benchmarks en utilisant les algorithmes de placement DEMO sur deux topologies, le maillage 2D et 3D, dans nos tests. Rappelons que la stratégie de mutation Rand_1 et Best_1, qui ont montré les meilleurs résultats lors de la phase d'allocation, ont été utilisées dans les tests. Ces résultats ont été publiés dans [202].

Pour les graphes de tâches utilisés lors des tests, les Figures 7.20, 7.21 et 7.22 présentent les résultats obtenus par l'algorithme DEMO avec la stratégie Rand_1, tandis que les Figures 7.23, 7.24 et 7.25 présentent les résultats obtenus par la stratégie BEST. De plus, chaque figure indique les valeurs minimales atteintes pour la consommation d'énergie, la surface et le temps d'exécution pour chaque graphe généré par TGFF selon les deux stratégies. Ces résultats montrent l'efficacité des stratégies de mutation Rand_1 et Best_1 dans l'algorithme DEMO pour trouver des solutions de placement de haute qualité.

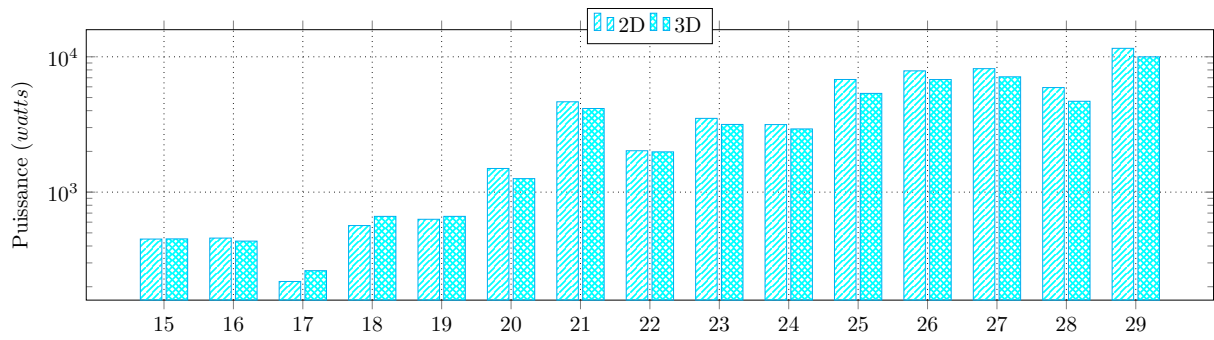


FIGURE 7.20 – Comparaison des placement obtenues par DEMO Rand_1 concernant l'énergie

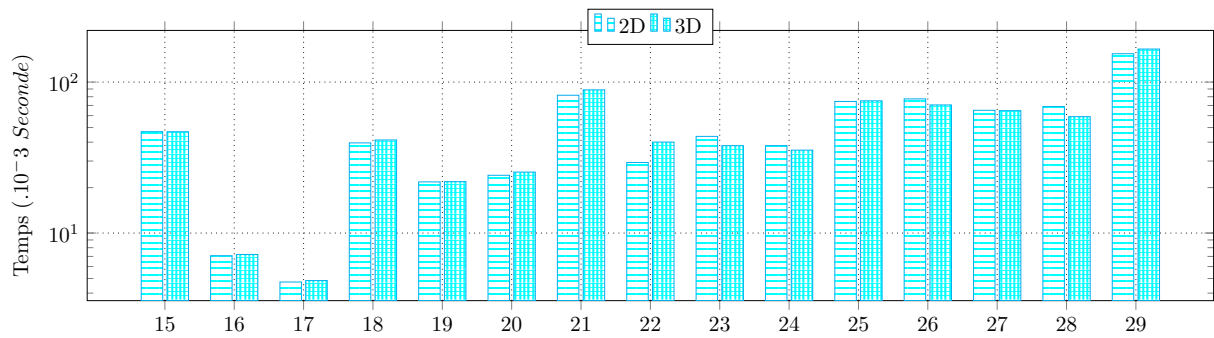


FIGURE 7.21 – Comparaison des placement obtenues par DEMO Rand_1 concernant le temps

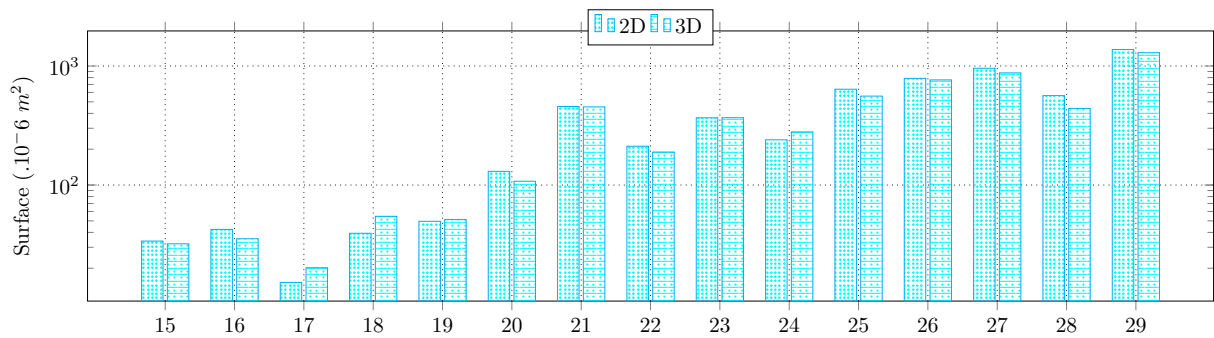


FIGURE 7.22 – Comparaison des placement obtenues par DEMO Rand_1 concernant la surface

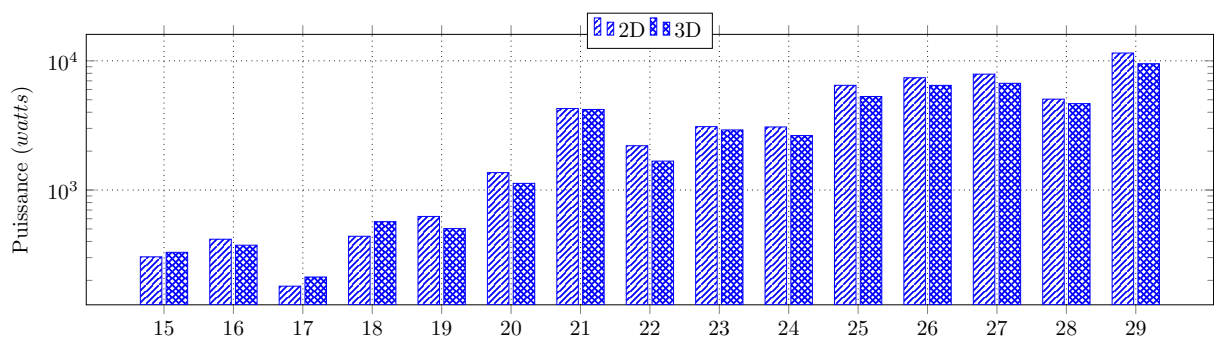


FIGURE 7.23 – Comparaison des placement obtenues par DEMO Best_1 concernant l'énergie

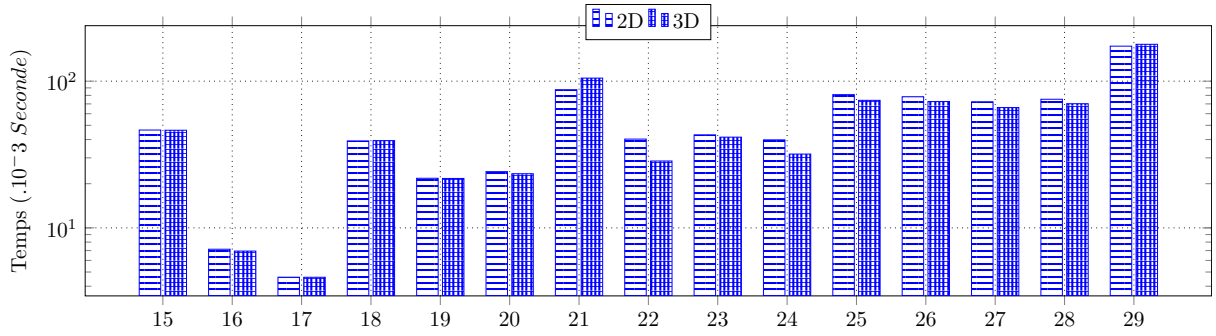


FIGURE 7.24 – Comparaison des placement obtenues par DEMO Best_1 concernant le temps

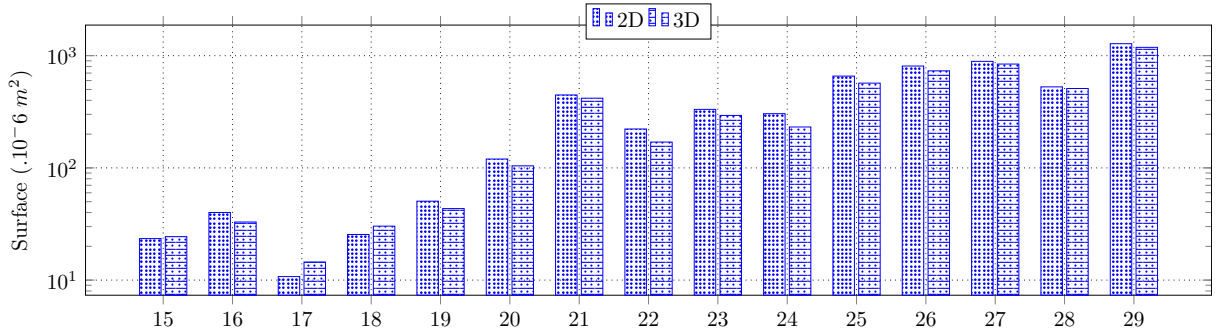


FIGURE 7.25 – Comparaison des placement obtenues par DEMO Best_1 concernant la surface

De même que MOPSO, les deux stratégies ont démontré que les résultats obtenus par un placement sur un NoC 3D sont meilleurs que ceux sur un NoC 2D.

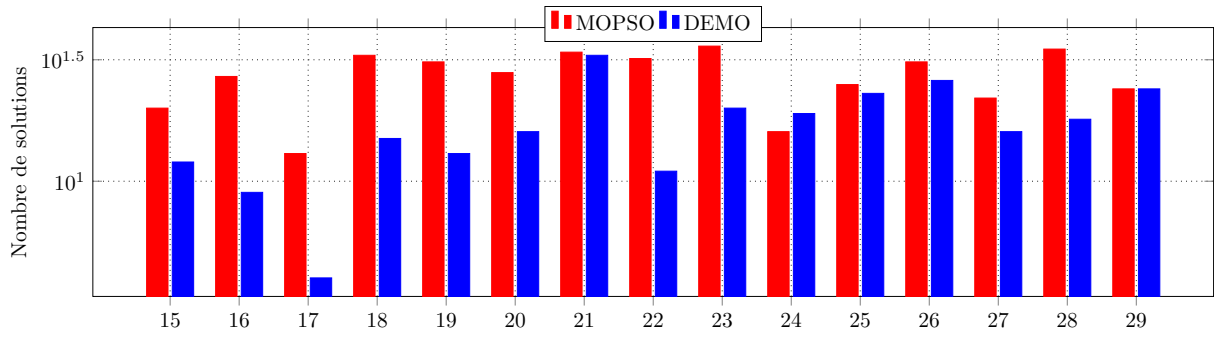
Les Tables C.21 et C.22, présentées dans l'Annexe C, exposent quelques solutions de placement non dominées trouvées par DEMO pour les cinq graphes de tâches utilisés dans le test. Ces tables montrent la solution d'affectation ainsi que son emplacement dans le NoC 2D et 3D respectivement.

7.4.3 Comparaison

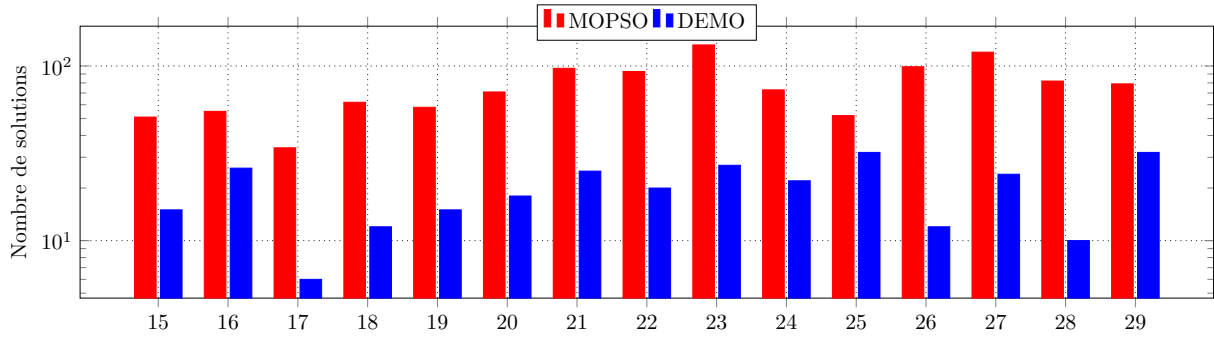
Une comparaison des résultats obtenus a été prise. Les solutions non dominées produites par les algorithmes MOPSO et DEMO ont été utilisées pour cette évaluation. Cette démarche permet d'évaluer la performance de chaque algorithme.

La Figure 7.26 illustre le nombre total de solutions non dominées obtenues pour chacune des méthodes, en ce qui concerne le placement en 2D et 3D. En ce qui concerne la topologie 2D, les Figures 7.29, 7.27 et 7.28 présentent les résultats obtenus selon les stratégies utilisées ainsi que MOPSO. Elles comparent les meilleures valeurs obtenues par chaque méthode en fonction des objectifs de puissance, temps et surface, respectivement

Pour la topologie 3D, la Figure 7.30 montre et permet une meilleure visualisation et comparaison de la consommation d'énergie pour le rendement du placement par rapport aux différentes stratégies comparées, en ce qui concerne les meilleurs résultats.



(a) 2D



(b) 3D

FIGURE 7.26 – Solutions non dominées obtenues par DEMO et MOPSO

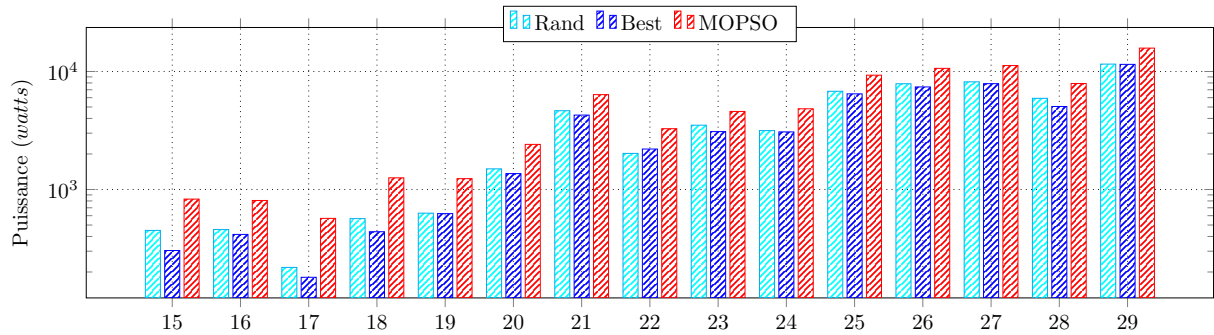


FIGURE 7.27 – Comparaison du placement 2D obtenu en fonction des exigences de puissance

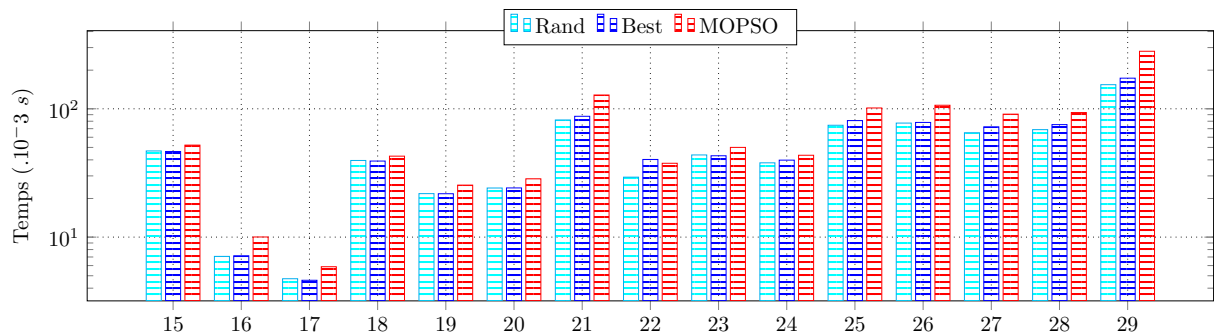


FIGURE 7.28 – Comparaison du placement 2D obtenu en fonction des exigences du temps

De plus, le graphique de la Figure 7.31 permet une meilleure comparaison visuelle des résultats en ce qui concerne la caractéristique temporelle en considérant le meilleur

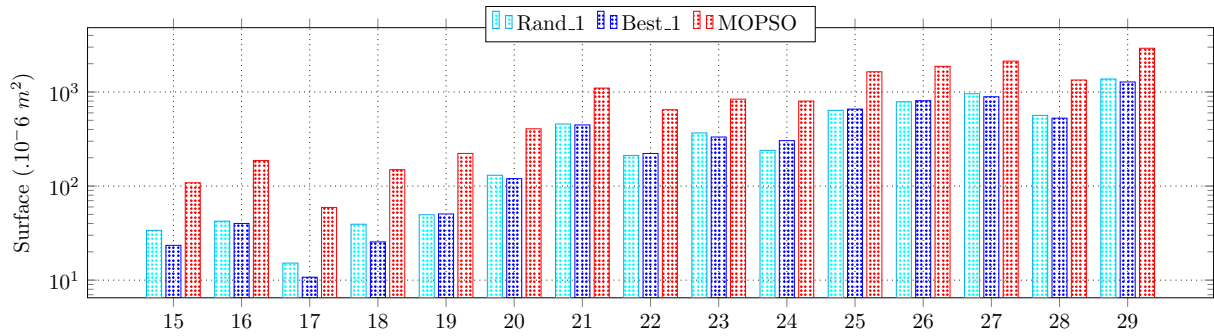


FIGURE 7.29 – Comparaison du placement 2D obtenu en fonction des exigences de surface

placement de qualité, en plus d'afficher ces résultats. Enfin, la Figure 7.32 montre et facilite la comparaison de la surface requise par le placement obtenu par les stratégies comparées.

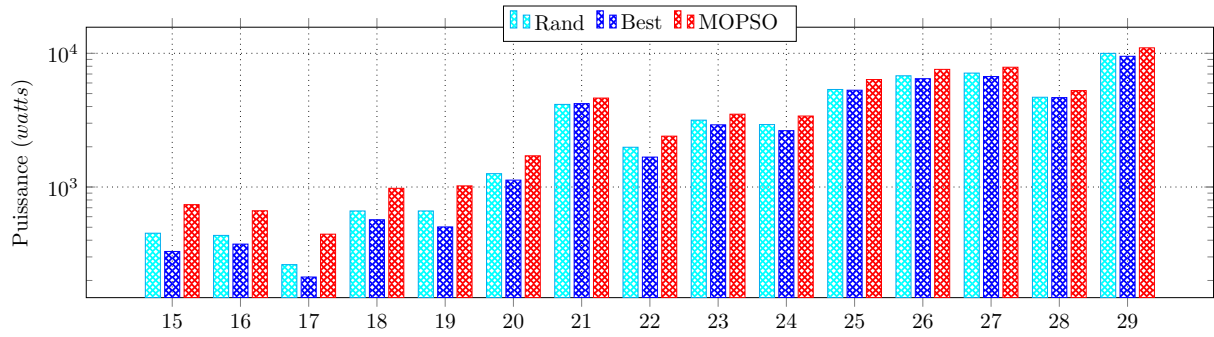


FIGURE 7.30 – Comparaison du placement 3D obtenu en fonction des exigences de puissance

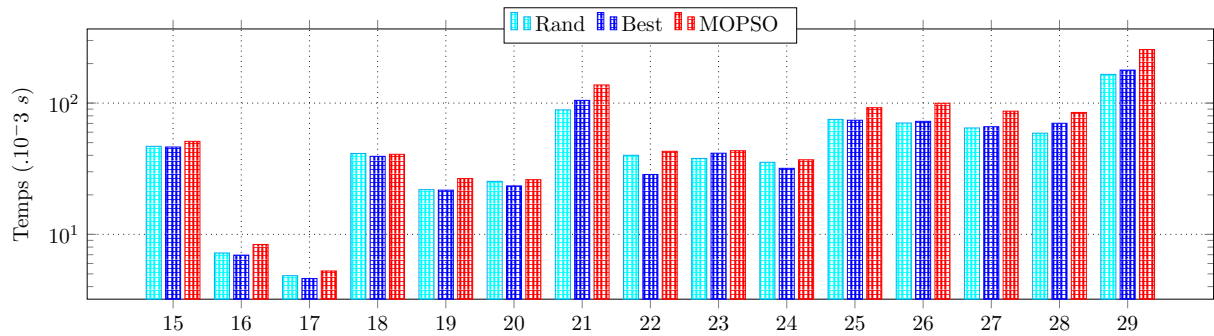


FIGURE 7.31 – Comparaison du placement 3D obtenu en fonction des exigences du temps

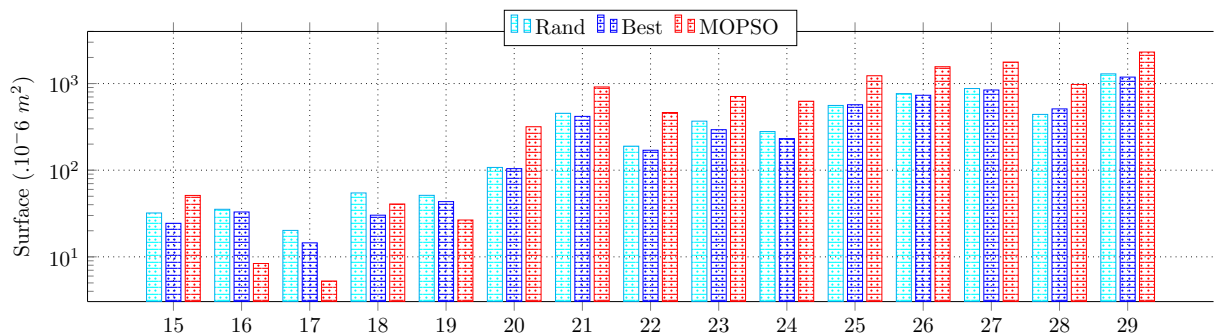


FIGURE 7.32 – Comparaison du mappage 3D obtenu en fonction des exigences de surface

L'algorithme DEMO a démontré sa supériorité par rapport à MOPSO dans le contexte des topologies de maillage 2D et 3D. De plus, la stratégie basée sur Best_1 et Rand_1 ont produit les meilleurs résultats parmi toutes les benchmarks testés. Cependant, des détails supplémentaires sur ces méthodes spécifiques seraient nécessaires pour une compréhension plus approfondie. Les trois métriques utilisées dans la phase d'affectation sont également employées dans cette phase afin de comparer les deux méthodes, MOPSO et DEMO, en ce qui concerne la convergence et la diversité des solutions obtenues. Les deux Tables 7.9 et 7.10 montrent les résultats obtenus concernant les métriques de convergence et de diversité pour le placement en 2D et 3D.

TABLE 7.9 – Métriques de performances obtenues pour le placement en 2D

Benchmark	solutions	MOPSO			solutions	DEMO		
		Contribution	Espacement	Hyper Volume		Contribution	Espacement	Hyper Volume
15	20	0.00	20386.187	0.933	12	100.00	3826.420	0.000266
16	27	0.00	12561.911	1.155	9	100.00	1460.818	0.000003
17	13	0.00	1416.518	0.000178	4	100.00	611.706	0.000000
18	33	0.00	26585.726	1.004	15	100.00	3115.027	0.000108
19	31	0.00	19168.652	0.313	13	100.00	2614.579	0.000022
20	28	0.00	29638.201	0.310	16	100.00	20508.527	0.0028
21	34	0.00	195812.640	22.242	33	100.00	217374.578	1.777
22	32	26.67	195349.640	1.433	11	73.33	10141.642	0.0038
23	36	0.00	171162.765	14.054	20	100.00	110157.789	0.230
24	16	0.00	159150.484	0.893	19	100.00	89404.773	0.138
25	25	0.00	439846.500	11.187	23	100.00	218860.000	3.455
26	31	0.00	675459.000	123.617	26	100.00	289820.875	0.809
27	22	0.00	776885.687	30.044	16	100.00	624807.375	0.061
28	35	0.00	536223.750	7.777	18	100.00	74309.539	0.0034
29	24	0.00	507386.687	32.630	24	100.00	256103.093	1.343

TABLE 7.10 – Métriques de performances obtenues pour le placement en 3D

Benchmark	solutions	MOPSO			solutions	DEMO		
		Contribution	Espacement	Hyper Volume		Contribution	Espacement	Hyper Volume
15	51	0.00	9833.519	1.022	15	100.00	1782.895	0.0305
16	55	0.00	6593.561	4.984	26	100.00	842.611	0.0340
17	34	0.00	3216.507	2.169	6	100.00	188.373	0.000000
18	62	0.00	12470.965	7.909	12	100.00	3142.073	0.000086
19	58	0.00	12736.180	3.338	15	100.00	5431.225	0.000014
20	71	0.00	48392.707	19.363	18	100.00	7928.116	0.048
21	97	0.00	100758.789	92.319	25	100.00	51307.183	0.050
22	93	0.00	86019.320	18.323	20	100.00	18601.365	0.0016
23	132	0.00	88128.164	134.207	27	100.00	39482.914	0.523
24	73	0.00	103511.656	24.854	22	100.00	24272.558	0.393
25	52	0.00	264446.875	28.888	32	100.00	35588.214	0.301
26	99	0.00	236744.328	86.254	12	100.00	51322.226	0.0561
27	120	0.00	297079.031	304.231	24	100.00	173803.156	0.356
28	82	0.00	171272.359	38.615	10	100.00	46229.761	0.117
29	79	0.00	235429.109	368.291	32	100.00	200945.734	5.149

Les résultats obtenus mettent en lumière une distinction notable entre les deux algorithmes évalués. En analysant les résultats de DEMO, nous constatons qu'il excelle à produire des résultats significatifs par rapport à la métrique de contribution, et sa capacité à converger rapidement vers le front pareto optimal est indéniable.

Cependant, lorsqu'on évalue la diversité des solutions générées, des lacunes apparaissent. Les résultats de DEMO ne présentent pas une répartition équilibrée dans la diversité, et les métriques de diversité associées à cet algorithme sont révélatrices d'une faible variabilité dans les solutions obtenues.

En revanche, malgré des performances moins impressionnantes en termes de convergence, MOPSO se démarque par sa capacité à obtenir un front pareto plus diversifié. Les solutions générées par cet algorithme présentent une plus grande variété, couvrant un spectre plus large d'alternatives potentielles. Bien que les résultats en termes de convergence puissent être moins favorables par rapport à DEMO. En termes de convergence, DEMO offre de meilleurs résultats comparativement à MOPSO. En revanche, en termes de diversité, MOPSO est supérieur à DEMO et démontre une capacité à obtenir une variété de solutions diverses dans l'espace de recherche. Cette observation se confirme en comparant le nombre de solutions dans le front obtenu par MOPSO avec celui obtenu par DEMO.

7.5 Résultats de la phase de routage

Dans cette étude, nous avons utilisé le simulateur Access-Noxim [9], une version améliorée de simulateur Noxim. Il s'agit d'un outil de simulation pour les réseaux sur puce [203] écrit en langage C++ basé sur la bibliothèque SystemC. Ce simulateur offre une interface en ligne de commande. Il permet de définir plusieurs paramètres du NoC. L'utilisateur peut ainsi personnaliser: la taille du réseau, la taille des tampons, la distribution des tailles de paquets, l'algorithme de routage, la stratégie de sélection, le taux d'injection de paquets et le modèle de trafic et la répartition du trafic des points chauds.

Le simulateur permet d'évaluer le NoC en termes de débit, de latence et de consommation d'énergie. Ces informations sont fournies à l'utilisateur sous forme de résultats moyens et individuels pour chaque communication. La stratégie de routage mise en œuvre peut être facilement modifiée en écrivant et testant le code en langage C++. Notre principal objectif est de mesurer la latence en tant qu'indicateur de performance. Pour utiliser le simulateur selon nos benchmarks de test et obtenir des résultats, nous adoptons la configuration listée dans la Table 7.11.

Après plusieurs tests, nous avons optimisé les paramètres des deux algorithmes pour obtenir les meilleurs résultats en routage. Pour les paramètres utilisés dans l'algorithme PSO, la taille de la population est fixée à 500, avec w égal à 0.8, $c1$ et $c2$ sont à 1.49. L'algorithme a été exécuté pendant 1000 itérations. Pour l'algorithme

TABLE 7.11 – Paramètre de configuration du simulateur Access-Noxim

Paramètre	Configuration
Network	2D Mesh vs 3D Mesh
Network dimensions	5×5 vs 3×3×3
Packet size	8
Virtual channels	4
Warm-up time	1000
Simulation time	20000 cycles
Routing algorithms	DE et PSO, (XYZ ou XY) et Odd-Even

DE, la taille de la population est de 200, avec F réglé sur 0.5 et CR réglé sur 0.9. L'algorithme a été exécuté pendant 1000 itérations. L'opération de mutation choisi est Rand_1.

Notre méthode de routage appartient à la classe des routages adaptatifs et statiques à la fois. Elle est considérée comme statique, car elle se base sur les résultats du placement des nœuds. Malgré l'utilisation d'un NoC de petite taille, qu'il soit en 2D ou en 3D, cela n'empêche pas de prendre en charge des applications complexes. En effet, il est possible d'allouer plusieurs nœuds à une même ressource du NoC, comme nous l'avons démontré lors des phases d'affectation et de placement. Cette approche permet une utilisation optimale des ressources et garantit une adaptabilité même pour des applications de grande envergure.

Les deux classes de schémas de routage présentées dans la Section 7.1.3 ont été utilisées pour les tests. Pour le schéma aléatoire, nous avons testé: Rand, point chaud et local. En revanche, pour les schémas de routage déterministes, les modèles sont listés dans la Table 7.12.

TABLE 7.12 – Nœuds de destination des schémas de routage déterministes

Network	Schéma de routage	source node	Destination node
2D Mesh	Complement	(x, y)	(size - x - 1, size - y - 1)
	Transpose 1	(x, y)	(size - y - 1, size - x - 1)
	Transpose 2	(x, y)	(y, x)
3D Mesh	Complement	(x, y, z)	(size - x - 1, size - y - 1, size - z - 1)
	Transpose 1	(x, y, z)	(size - y - 1, size - z - 1, size - x - 1)
	Transpose 2	(x, y, z)	(y, z, x)

7.5.1 Résultats de PSO et DE

En comparant les résultats obtenus avec les deux algorithmes de routage, nous avons évalué leur efficacité en termes de latence, de débit et de congestion. Cette comparaison nous a permis de déterminer quel algorithme offre de meilleures performances pour router les communications dans le réseau sur puce étudié.

La Figure 7.33 présente les résultats de congestion obtenus par les deux algorithmes, PSO et DE, pour les modèles de trafic aléatoire et déterministe, respectivement, pour les deux topologies 2D et 3D.

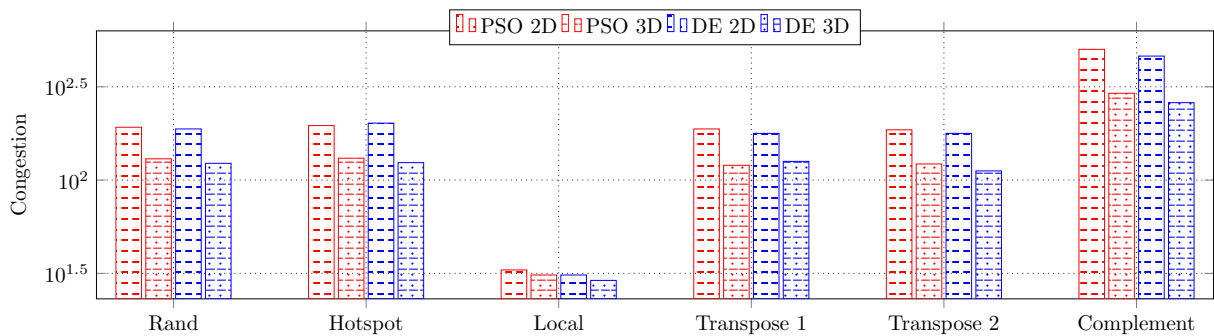


FIGURE 7.33 – Valeurs obtenue de Congestion par DE et PSO

La valeur de congestion indique le niveau de congestion atteint lors du routage des paquets. Une valeur de congestion élevée indiquer une saturation ou une utilisation excessive des ressources, ce qui peut entraîner une augmentation de la latence.

De même, la Figure 7.34 présente les résultats de latence obtenus par les deux algorithmes, PSO et DE, pour les modèles de trafic aléatoire et déterministe, respectivement, pour les deux topologies 2D et 3D.

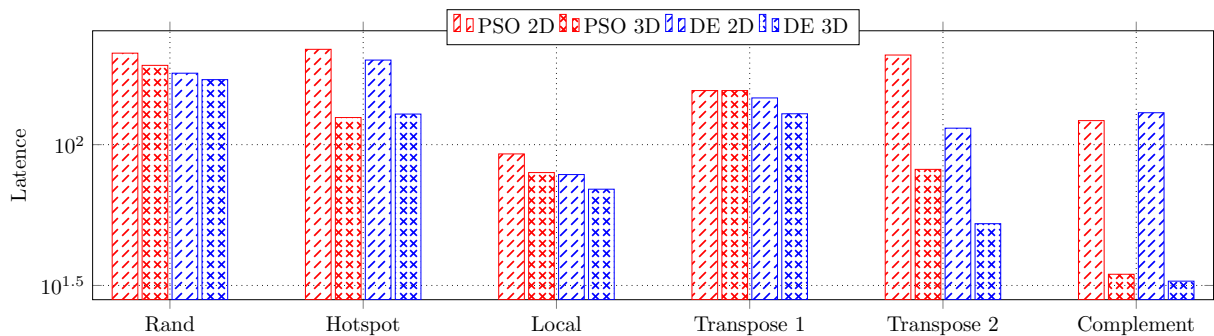


FIGURE 7.34 – Valeurs obtenue de latence par DE et PSO

La valeur globale de la latence représente la latence moyenne observée lors de l'utilisation d'un algorithme de routage. Une latence plus faible est généralement souhaitable car elle indique un temps de communication plus court entre les nœuds.

La Figure 7.35 présente les résultats de débit obtenus par les deux algorithmes, PSO et DE, pour les modèles de trafic aléatoire et déterministe, respectivement, pour les deux topologies 2D et 3D.

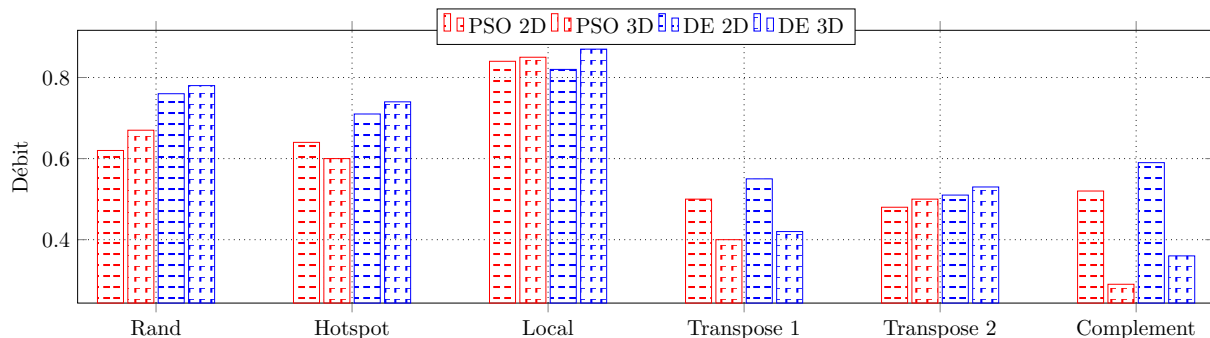


FIGURE 7.35 – Valeurs obtenue de bande passante par DE et PSO

Le débit maximum représente la capacité maximale du système à transmettre des paquets selon l'algorithme de routage appliqué. Un débit plus élevé indique une meilleure utilisation des ressources et une capacité accrue à traiter les paquets. Notons que les résultats du modèle de pattern déterministe ont été publiés dans [204] pour l'algorithme de routage PSO.

Les résultats obtenus ont indiqué que l'algorithme DE a généralement montré de meilleures performances que l'algorithme PSO dans les différentes simulations de schéma de routage. Cependant, il est important de poursuivre les recherches et les expérimentations afin de mieux comprendre les facteurs qui influencent les performances de chaque algorithme dans des conditions spécifiques et d'explorer d'autres métriques et objectifs pertinents dans le cadre de l'optimisation du routage en réseau sur puce.

7.5.2 Comparaison

Dans cette section, nous avons réalisé une comparaison des résultats obtenus en utilisant deux algorithmes de routage déterministes : XY [205] et Odd Even [98] pour la topologie 2D, et XYZ [96] et Odd Even [98] pour la topologie 3D. L'objectif de cette comparaison est d'évaluer les performances de ces deux algorithmes par rapport à nos algorithmes DE et PSO.

Nous avons collecté des données expérimentales en simulant des réseaux sur puce avec différentes configurations, puis en exécutant les algorithmes de routage PSO et DE. Les mesures de performance prises en compte étaient principalement la latence, c'est-à-dire le temps nécessaire pour qu'un paquet atteigne sa destination dans le réseau. De plus, nous présentons les résultats en appliquant les algorithmes de routage sur des cas réels, plus précisément sur les graphes de tâches générés avec l'outil TGFF. Nous avons utilisé seulement 5 graphes dans cette étude.

Pour les schémas de routage, les résultats de cette analyse représente la latence obtenue en fonction des différents pourcentages d'injection de paquets. Ces résultats permet d'observer comment la latence varie lorsque le taux d'injection augmente ou diminue. Elle fournit une vision globale des performances de routage. La Figure 7.36 correspond au schéma de routage Rand, où les paquets sont générés de manière aléatoire dans le réseau. Nous présentons ici les résultats obtenus par les quatre algorithmes pour un routage en 2D et en 3D. Dans l'analyse du routage en 2D, les algorithmes PSO et DE ont obtenu de

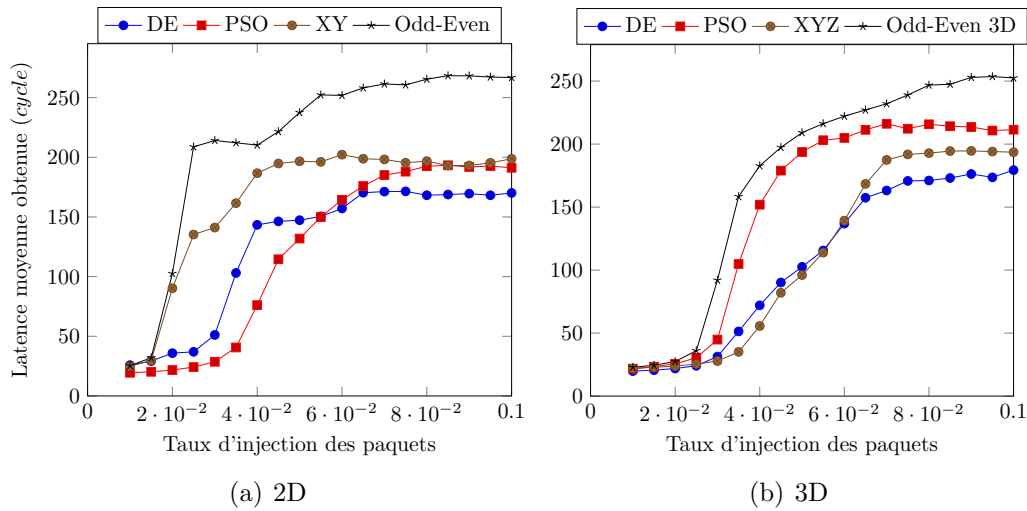


FIGURE 7.36 – Résultat obtenu de latence pour un schéma de routage Rand

meilleurs résultats que les algorithmes XY et Odd-Even. Cependant, PSO est meilleur que DE. Néanmoins, lorsque le taux d'injection de paquets (PIR) atteint 0.06, DE surpasse PSO. En revanche, dans le cas du routage en 3D, les résultats mettent en évidence que l'algorithme DE offre de meilleures performances en termes de latence par rapport à PSO, XYZ et Odd-Even dans les patterns aléatoires. On remarque également que PSO n'obtient pas de meilleurs résultats que DE, tandis que DE est supérieur à XYZ, et XYZ est supérieur à PSO.

La Figure 7.37 représente les résultats de latence obtenus en 2D et 3D pour le schéma de routage Hotspot, où certains nœuds du réseau génèrent un trafic plus intense que les autres. Dans le cas du Hotspot en NoC 2D, les algorithmes PSO et DE surpassent les algorithmes XY et Odd-Even, démontrant leur efficacité dans la gestion de ce type de trafic intense. Dans le cas du NoC 3D, PSO présente initialement de bonnes performances en termes de latence jusqu'à ce que le PIR atteigne 0.065. À partir de là, DE surpasse à la fois PSO et XYZ, tandis que PSO reste meilleur que XYZ.

Enfin, la Figure 7.38 montre les courbes de latence pour le schéma de routage Local, dans lequel les paquets sont générés localement par chaque nœud pour un routage en 2D et 3D.

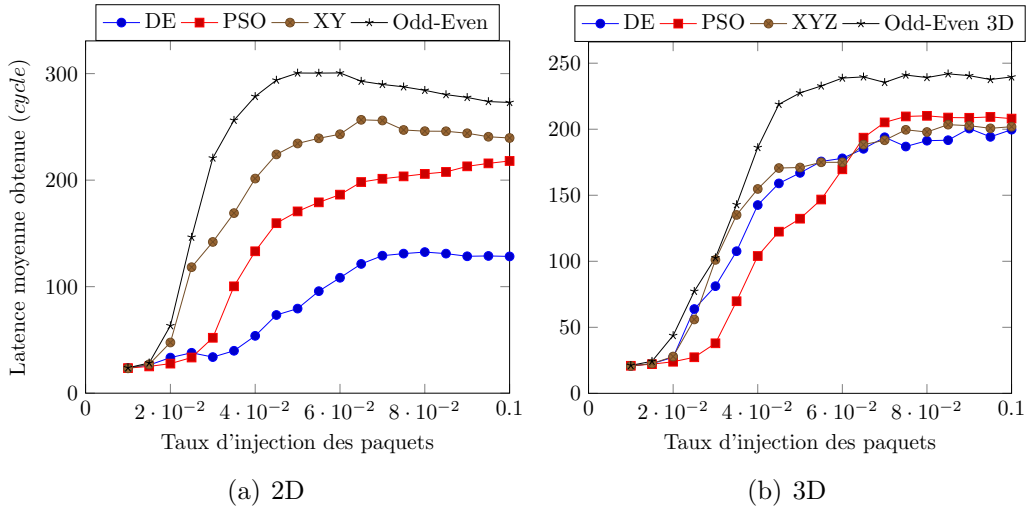


FIGURE 7.37 – Résultat obtenu de latence pour un schéma de routage Hotspot

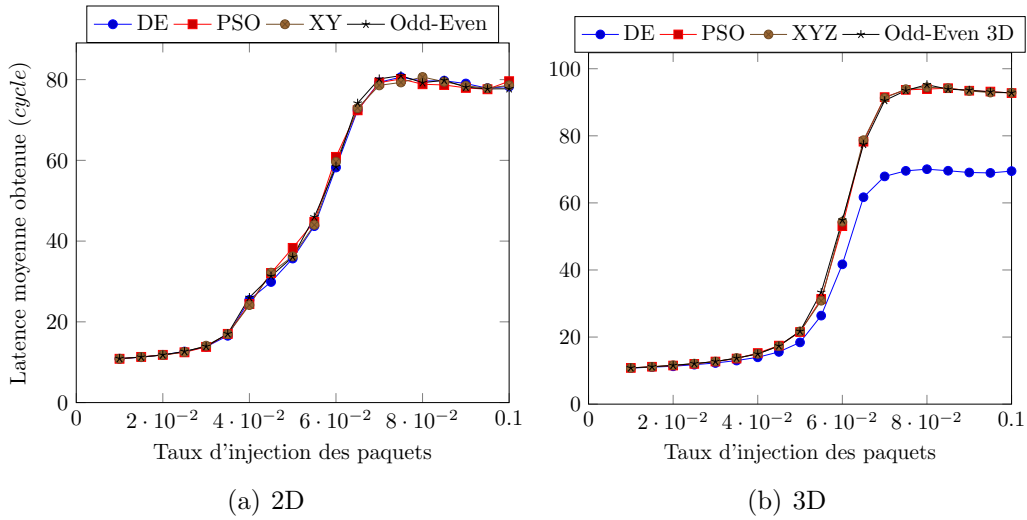


FIGURE 7.38 – Résultat obtenu de latence pour un schéma de routage Local

Dans un routage en 2D, tous les algorithmes de routage produisent des courbes de latence similaires. Cela indique que, pour ce schéma spécifique, les performances des différents algorithmes de routage sont comparables et ne montrent pas de différences significatives en termes de latence. En ce qui concerne le routage en 3D, XYZ, Odd-Even et PSO présentent pratiquement la même courbe de latence, tandis que DE se révèle être le meilleur des trois algorithmes.

Dans le routage avec des schémas de routage déterministes, la Figure 7.39 présente visuellement les courbes de latence spécifiques au schéma de routage Transpose_1, tant en 2D qu'en 3D.

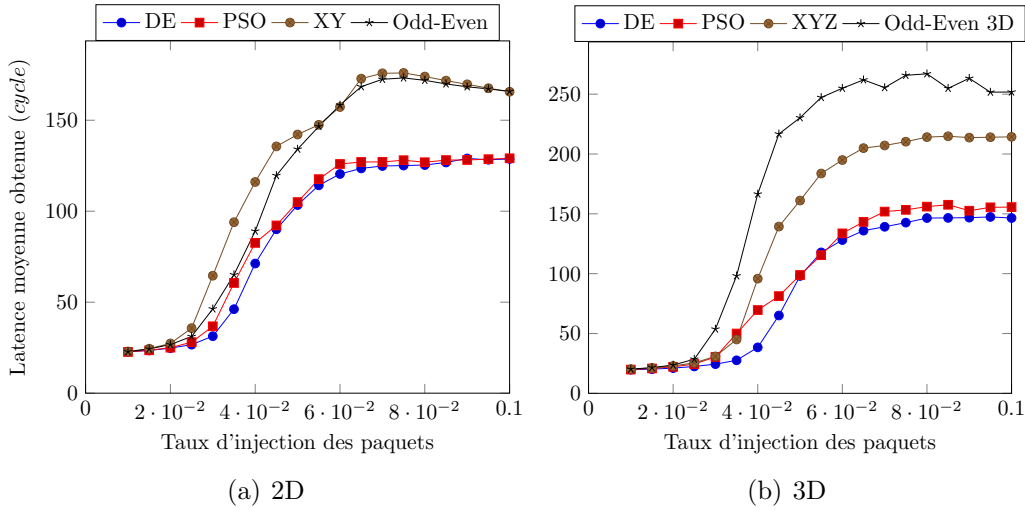


FIGURE 7.39 – Résultat obtenu de latence pour le schéma de routage Transpose_1

Dans l'analyse du routage en 2D, nous constatons que les algorithmes PSO et DE présentent des valeurs assez similaires, et surpassent les algorithmes XY et Odd Even. En ce qui concerne les résultats du routage en 3D, les courbes de latence de PSO et DE sont presque identiques et surpassent celles de XYZ et Odd-Even. Cela signifie que PSO et DE sont plus performants dans ce schéma par rapport à XYZ et Odd-Even.

La Figure 7.40 illustre graphiquement les courbes de latence pour le schéma de routage Transpose_2, à la fois en 2D et en 3D.

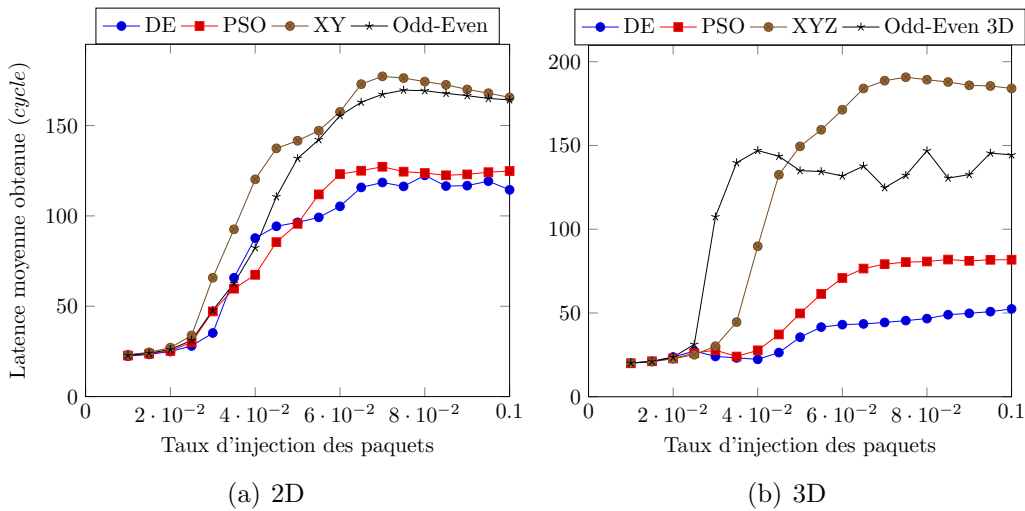


FIGURE 7.40 – Résultat obtenu de latence pour le schéma de routage Transpose_2

Pour le schéma Transpose_2 dans un NoC 2D, DE montre des performances améliorées en termes de latence lorsque le PIR dépasse 0.045. Cela suggère que DE est plus adapté pour gérer des taux d'injection de paquets plus élevés dans ce motif spécifique. En revanche,

dans un NoC 3D, à partir d'un PIR de 0.04, PSO et DE commencent à fournir de meilleurs résultats en termes de latence, mais DE surpasse PSO. Cela indique que DE est plus efficace que PSO dans la réduction de la latence dans ce pattern particulier.

Finalement, la Figure 7.41 présente visuellement les courbes de latence spécifiques pour le schéma de routage Complément, aussi bien en 2D qu'en 3D.

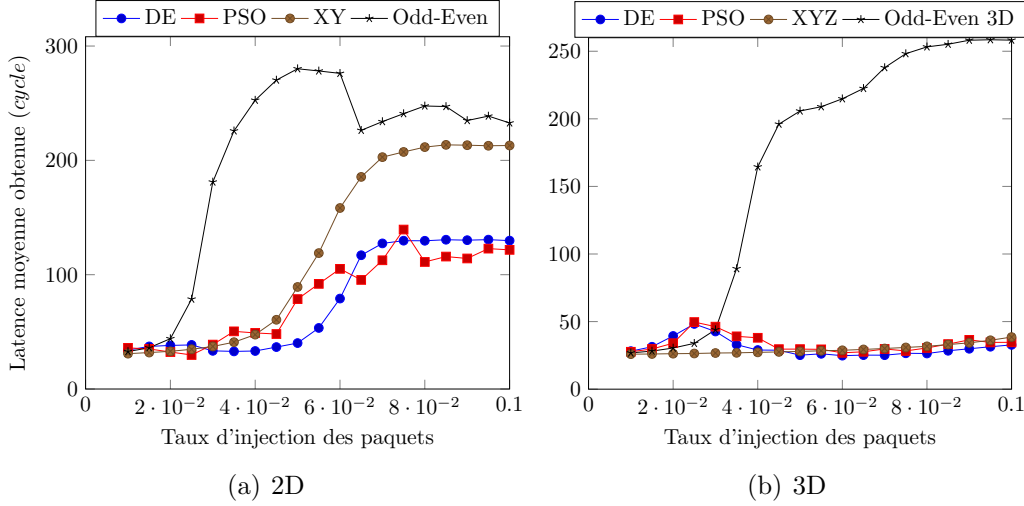


FIGURE 7.41 – Résultat obtenu de latence pour le schéma de routage Complément

Dans le schéma Complément en 2D, PSO est meilleur que DE lorsque le taux d'injection de paquets (PIR) est supérieur à 0.065. PSO semble mieux s'adapter lorsque le PIR est élevé. En ce qui concerne le routage en 3D, XYZ présente une courbe de latence plate, suggérant qu'il ne parvient pas à améliorer significativement la latence. PSO, quant à lui, présente une courbe similaire à XYZ, mais DE offre de meilleurs résultats à partir d'un PIR de 0.045. Cela signifie que DE est plus performant que PSO et XYZ dans la réduction de la latence dans le schéma Complément en 3D.

En effet, un algorithme de routage doit être capable d'obtenir de bons résultats, quelle que soit la nature des benchmarks utilisés et quelle que soit la disposition des tâches dans le NoC. Afin de valider les deux algorithmes de routage proposés dans les tests citer avant, des simulations impliquant l'utilisation d'applications sont nécessaires. Dans cette thèse, les benchmarks créés par TGFF sont utilisés pour le routage. À chaque benchmark on associe un placement aléatoire des tâches. Nous utilisons pour chaque benchmark cinq placements aléatoires. Les Figures 7.42, 7.43, 7.44, 7.45, et 7.46 présentent les valeurs de la latence obtenues par les algorithmes de routage DE, PSO, XY, XYZ, OddEven pour le routage en NoC 2D et 3D.

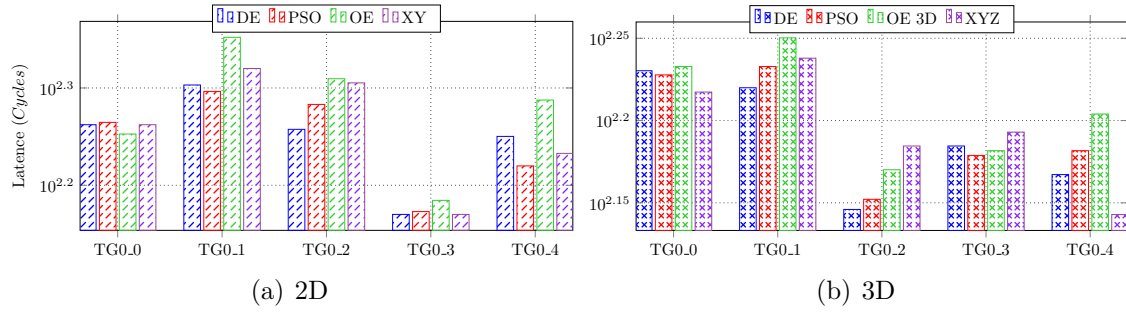


FIGURE 7.42 – Résultat de latence obtenue pour pour le graphe de tâches TG0

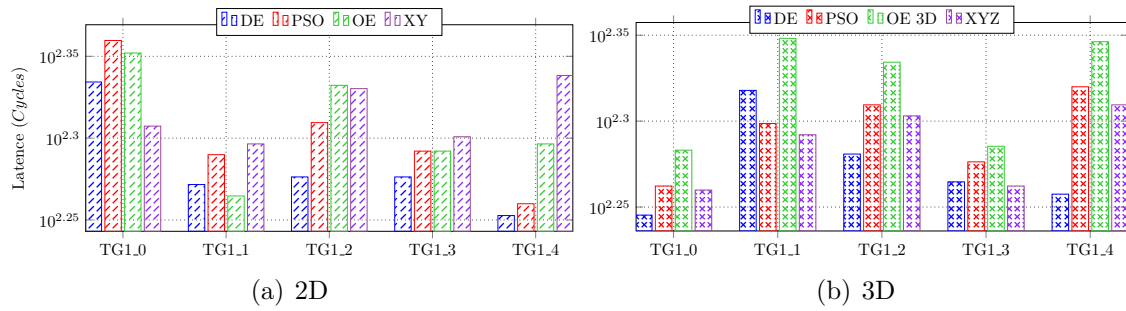


FIGURE 7.43 – Résultat de Latance obtenue pour pour le graphe de tâches TG1

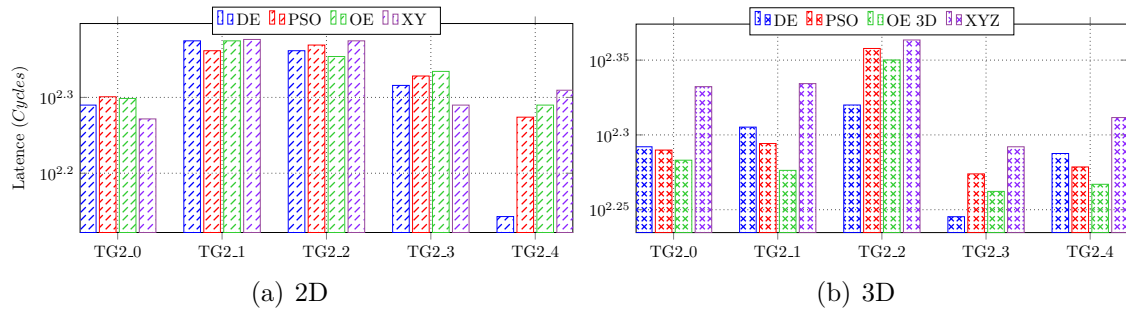


FIGURE 7.44 – Résultat de latence obtenue pour pour le graphe de tâches TG2

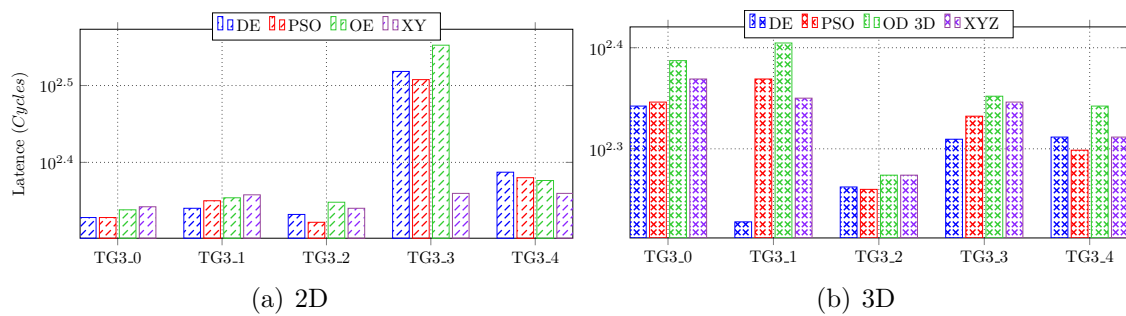


FIGURE 7.45 – Résultat de latence obtenue pour pour le graphe de tâches TG3

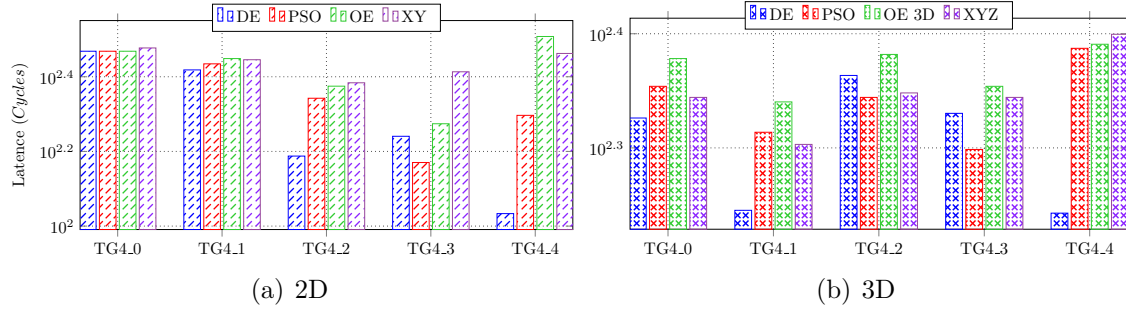


FIGURE 7.46 – Résultat de latence obtenue pour pour le graphe de tâches TG4

Suite à une analyse approfondie des résultats, nous notons avec satisfaction que nos algorithmes de routage, notamment PSO et DE, ont démontré une amélioration significative de la latence par rapport à d'autres, tels que OE, XY et XYZ, dans la plupart des 30 benchmarks sur les 50 examinés. Cependant, il est crucial de souligner la diversité des performances observées, puisque dans certains benchmarks spécifiques, le DE a surpassé le PSO dans 32 cas, tandis que le PSO a amélioré la latence par rapport au DE dans 16 benchmarks.

XY et XYZ ont toutefois conservé leur supériorité dans certains cas spécifiques parmi les 16 benchmarks analysés. Cette complexité dans les observations peut être attribuée à la nature du processus de placement, où un positionnement aléatoire des tâches peut occasionnellement les regrouper à proximité, conférant ainsi des avantages de performance à XY et XYZ. Il est essentiel de noter que DE et PSO se distinguent également dans certains benchmarks, surpassant généralement OE dans 42 cas, indiquant une capacité d'adaptation et d'optimisation supérieure dans ces contextes spécifiques. De plus, une observation approfondie révèle généralement de meilleures performances de DE par rapport à PSO dans la plupart des cas, bien que ce dernier puisse surpasser DE dans certains benchmarks spécifiques.

7.6 Considération finale du chapitre

Dans ce chapitre, nous avons présenté les résultats des tests sur l'optimisation des réseaux sur puce, en couvrant les trois principales phases: l'allocation des tâches d'une application, le placement des IPs de l'application dans le réseau sur puce et le routage des paquets. L'implémentation, repose sur la modélisation des trois phases précédemment décrites, comme exposé dans le chapitre précédent. Les tests ont été réalisés sur des benchmarks issus des applications E3S et l'outil TGFF. Les objectifs principaux utilisés pour l'évaluation étaient la surface occupée, la consommation d'énergie et le temps d'exécution. Ensuite, nous avons comparé deux algorithmes de routage basés sur PSO et DE, en utilisant des benchmarks tirés de modèles tels que Transpose et Uniforme.

Les résultats obtenus ont démontré l'efficacité de nos approches d'optimisation multi-objectifs pour l'allocation des tâches, le placement des IPs et le routage des paquets dans les réseaux sur puce. Ces résultats ont été comparés à ceux obtenus avec d'autres benchmarks et ont confirmé la performance de nos algorithmes. Ces tests ont permis de mettre en évidence les avantages des approches PSO et DE dans le domaine de l'optimisation des réseaux sur puce, en termes de surface occupée, de consommation d'énergie et de temps d'exécution. Ces résultats ouvrent la voie à de futures recherches et améliorations dans le domaine de l'optimisation des réseaux sur puce.

Chapitre 8

Conclusion générale et perspectives

8.1 Conclusions

Dans le cadre de cette thèse, nous avons exploré différents aspects liés à l'intégration des composants et à l'optimisation de leur fonctionnement dans la conception des systèmes sur puce. Dans un premier temps, notre étude s'est penchée sur les réseaux sur puce (NoCs), une approche novatrice dans la conception des systèmes sur puce (SoC). Nous avons approfondi notre compréhension de leur structure et de leur fonctionnement, mettant en lumière leur rôle crucial dans la facilitation d'une communication efficace entre les divers composants du SoC.

En outre, une attention particulière a été accordée aux réseaux sur puce en trois dimensions (3D-NoC). Nous avons exploré en détail leur architecture et leur mode de fonctionnement, mettant en évidence leur capacité à exploiter la dimension verticale pour augmenter significativement la capacité de communication. Cette approche tridimensionnelle offre des avantages remarquables, notamment une réduction des latences de communication et une optimisation de l'utilisation de l'espace physique disponible sur la puce.

Cette thèse éclaire les trois phases essentielles pour la conception d'une application embarquée sur un réseau sur puce, à savoir l'affectation des tâches aux processeurs, leur placement sur la plateforme NoC, et enfin, le routage des données entre ces processeurs.

Dans ce contexte, une section est dédiée à la présentation des travaux de recherche antérieure sur ces trois phases. Plus précisément, nous avons étudié les différentes approches utilisées dans la conception des réseaux sur puce, en les associant à chaque phase respective. Nous avons examiné en détail les avantages et les limitations de chaque méthode, ainsi que leur pertinence dans les différentes étapes du processus de conception. De plus, une classification a été présentée en fonction des algorithmes choisis.

Selon cette étude, l'utilisation des algorithmes d'optimisation est une nécessité compte tenu de la nature du problème de chaque phase. Pour cela, une section spécifique a été dédiée à l'optimisation, où nous avons examiné en détail les différentes approches et concepts clés. Nous avons commencé par définir le concept d'optimisation, mettant en évidence son importance dans la conception des réseaux sur puce. Ensuite, nous avons exploré les spécifications des méthodes d'optimisation, en analysant leurs caractéristiques et leurs applications potentielles dans ce contexte.

Vu que la nature des problèmes traités est généralement multi-objectifs, nous avons mis l'accent sur l'optimisation multi-objectifs, une approche sophistiquée qui permet de prendre en compte simultanément plusieurs objectifs contradictoires dans le processus d'optimisation. Nous avons examiné en quoi cette approche diffère des méthodes d'optimisation traditionnelles, en soulignant ses avantages et ses applications dans la conception des réseaux sur puce.

Deux algorithmes évolutifs, PSO et DE, ont été utilisés pour comparer à la fois les résultats obtenus et les performances intrinsèques de ces algorithmes. Notre analyse approfondie s'est concentrée sur ces deux méthodes d'optimisation spécifiques, ce qui nous a permis de mieux comprendre leurs fondements théoriques et leurs mécanismes de recherche, tant pour les problèmes à objectif unique que pour les problèmes à objectifs multiples. Le choix de ces méthodes a été motivé par leurs performances prometteuses dans les problèmes d'optimisation multi-objectifs. De plus, ces algorithmes présentent des approches évolutives distinctes, justifiant ainsi une comparaison approfondie des résultats.

Une fois les algorithmes d'optimisation sélectionnés, plusieurs étapes ont été nécessaires pour mettre en œuvre les algorithmes choisis. Cela comprenait la définition d'une source de données de référence pour les IP, le codage des individus représentant les solutions aux problèmes, la détermination des fonctions objectives d'évaluation des individus et l'analyse des résultats obtenus.

Pour les IPs, la source de référence était E3S, qui contient les données des processeurs, des tâches, des applications de référence et des graphes de tâches fournis par l'outil TGFF. Le codage des individus a été adapté au problème d'affectation, tandis que les fonctions d'évaluation pour chaque objectif ont été développées en se basant sur l'étude du problème d'affectation, incluant la consommation d'énergie, le temps d'exécution et la surface. L'optimisation a été réalisée en utilisant les données des IPs et les graphes de tâches. L'analyse des résultats obtenus a montré que les deux algorithmes ont réussi à trouver un grand nombre de solutions communes. Cependant, dans la plupart des cas, DEMO a trouvé davantage de solutions en moins de temps et avec des valeurs minimales plus petites.

Pour le problème de placement des IP, un placement d'IP multi-objectifs a été implémenté. Les algorithmes choisis ont été mis en œuvre de manière à ce que les solutions puissent représenter les IPs choisies selon le problème d'affectation et leur placement sur la

plateforme NoC choisie. Dans cette thèse, nous avons utilisé deux plateformes NoC, 2D et 3D. Le codage a été mis à jour pour représenter la plateforme NoC ainsi que les objectifs choisis. Dans les tests, nous avons ajouté des objectifs liés à la communication, tels que l'énergie de communication, le temps nécessaire pour la consommation et la surface. L'analyse des résultats obtenus a révélé que, dans la plupart des problèmes, DEMO a trouvé de bonnes solutions en moins de temps, tandis que MOPSO a trouvé plus de solutions que DEMO. Nous avons observé que DEMO est plutôt fort dans l'optimisation, convergent rapidement vers des solutions optimales, tandis que MOPSO est plus diversifiant, offrant des solutions plus diversifiées et plus équilibrées dans l'espace de recherche.

Pour le problème de routage, consistant à déterminer les chemins de communication optimaux entre les différents composants du système, nous avons proposé des approches d'optimisation pour résoudre ce problème. Ces approches visent à garantir une communication efficace tout en minimisant les retards et en évitant les congestions. Par ailleurs, nous avons réalisé une comparaison approfondie des performances des algorithmes d'évolution différentielle (DE) et de l'optimisation par essaim de particules (PSO) dans chacune de ces phases d'optimisation. Nos résultats ont mis en évidence une efficacité supérieure de l'algorithme DE par rapport à l'algorithme PSO dans la plupart des tests effectués. Ces conclusions suggèrent que l'algorithme DE est mieux adapté pour résoudre les problèmes d'optimisation spécifiques à notre domaine, offrant ainsi des perspectives prometteuses pour l'amélioration des performances de notre système.

8.2 Recherches futures

Au terme de notre travail, plusieurs perspectives de recherche émergent, ouvrant la voie à de nouvelles opportunités d'exploration et de développement. Voici quelques-unes de ces perspectives :

1. L'utilisation d'un NoC 3D partiellement connecté est un nouvel axe de recherche. L'idée est de personnaliser un NoC 3D en plaçant non pas la totalité des liens verticaux, mais seulement quelques-uns.
2. L'utilisation d'autres méta-heuristiques pour exploiter différentes stratégies de recherche. L'idée est d'exploiter différentes stratégies de recherche et d'exploration pour améliorer la diversité des solutions explorées. Chaque méta-heuristique a ses propres principes et heuristiques, ce qui leur permet d'explorer l'espace de recherche de manière unique.
3. La collaboration entre plusieurs méthodes. Cette collaboration peut être une approche puissante pour améliorer les performances globales du processus d'optimisation. L'idée derrière cette collaboration est de combiner les avantages des différentes méthodes pour obtenir des résultats plus performants.

Il existe différentes façons de collaborer entre les méthodes d'optimisation. Voici quelques exemples :

- Approche en cascade: Dans cette approche, les méthodes d'optimisation sont exécutées de manière séquentielle, où les résultats d'une méthode sont utilisés comme point de départ pour la méthode suivante.
 - Approche en parallèle: Cette approche permet d'explorer différentes régions de l'espace de recherche en parallèle et peut être particulièrement utile lorsque le temps de calcul est critique. Les méthodes d'optimisation sont exécutées simultanément et indépendamment les unes des autres.
 - Approche d'hybridation: Dans cette approche, les méthodes d'optimisation sont combinées en utilisant des opérateurs ou des stratégies spécifiques. Par exemple, les mécanismes d'évolution de l'algorithme génétique peuvent être combinés avec des mécanismes de recherche dans d'autres méthodes d'optimisations
4. L'extension du routage en routage multi-objectifs: C' est une approche qui vise à prendre en compte plusieurs objectifs simultanément lors du processus de routage. Contrairement au routage traditionnel, qui se concentre généralement sur un seul objectif

Ces perspectives ouvrent de nouvelles possibilités pour améliorer les performances des méthodes d'optimisation dans la conception de réseaux sur puce, et pour explorer de nouvelles approches qui permettent de résoudre efficacement les problèmes complexes liés à l'intégration des composants et à l'optimisation de leur fonctionnement.

Bibliographie

- [1] G. E. MOORE, « Cramming more components onto integrated circuits », *Proceedings of the IEEE*, vol. 86, no. 1, p. 82–85, 1998.
- [2] W. M. ARDEN, « The international technology roadmap for semiconductors—perspectives and challenges for the next 15 years », *Current Opinion in Solid State and Materials Science*, vol. 6, no. 5, p. 371–377, 2002.
- [3] L. BENINI et G. DE MICHELI, « Networks on chip: a new paradigm for systems on chip design », in *Proceedings 2002 Design, Automation and Test in Europe Conference and Exhibition*, p. 418–419, IEEE, 2002.
- [4] S. KUMAR, A. JANTSCH, J.-P. SOININEN, M. FORSELL, M. MILLBERG, J. OBERG, K. TIENSYRJA et A. HEMANI, « A network on chip architecture and design methodology », in *Proceedings IEEE Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design. ISVLSI 2002*, p. 117–124, IEEE, 2002.
- [5] B. S. FEERO et P. P. PANDE, « Networks-on-chip in a three-dimensional environment: a performance evaluation », *IEEE Transactions on computers*, vol. 58, no. 1, p. 32–45, 2008.
- [6] V. F. PAVLIDIS et E. G. FRIEDMAN, « Interconnect-based design methodologies for three-dimensional integrated circuits », *Proceedings of the IEEE*, vol. 97, no. 1, p. 123–140, 2009.
- [7] S. TYAGI, P. MAHESHWARI, A. AGARWAL et V. AVASTHI, « Exploring 3D network-on-chip architectures and challenges », in *2017 International Conference on Computer and Applications (ICCA)*, p. 97–101, IEEE, 2017.
- [8] D. ZHANG, Z. SONG, L. WANG et C. HUANG, « Survey on topologies of three-dimensional network on chip », *Journal of Frontiers of Computer Science & Technology*, vol. 9, no. 2, p. 129–164, 2015.
- [9] A. NOROLLAH, D. DERAFSHI, H. BEITOLLAHI et A. PATOOGHY, « PAT-NOXIM: a precise power & thermal cycle-accurate NoC simulator », in *2018 31st IEEE International System-on-Chip Conference (SOCC)*, p. 163–168, IEEE, 2018.
- [10] Y. ATAT, *Conception de haut niveau des MPSoCs à partir d’une spécification Simulink: passerelle entre la conception au niveau système et la génération d’architecture*. Thèse doctorat, Institut National Polytechnique de Grenoble-INPG, France, 2007.

- [11] A. S. ABDALLAH, *Conception de SoC à base d'horloges abstraites: vers l'exploration d'architectures en MARTE*. Thèse doctorat, Université des Sciences et Technologie de Lille-Lille I, France, 2011.
- [12] M. WOLF, *Multiprocessor systems-on-chips*. Morgan Kaufmann, 2005.
- [13] A. BOUCHHIMA, *Modélisation du logiciel embarqué à différents niveaux d'abstraction en vue de la validation et la synthèse des systèmes monopuces*. Thèse doctorat, Institut National Polytechnique de Grenoble-INPG, France, 2006.
- [14] R. LEMAIRE, *Conception et modélisation d'un système de contrôle d'applications de télécommunication avec une architecture de réseau sur puce (NoC)*. Thèse doctorat, Institut National Polytechnique de Grenoble-INPG, France, 2006.
- [15] J. DELORME, *Méthodologie de modélisation et d'exploration d'architecture de réseaux sur puce appliquée aux télécommunications*. Thèse doctorat, Institut National des Sciences Appliquées de Rennes, France, 2007.
- [16] A. KOUADRI-MOSTEFAOUI, *Architectures Flexibles pour la Validation et L'exploration de Réseaux-sur-Puce*. Thèse doctorat, Institut National Polytechnique de Grenoble-INPG, France, 2009.
- [17] S. JOVANOVIĆ, *Architecture reconfigurable de système embarqué auto-organisé*. Thèse doctorat, Université Henri Poincaré-Nancy 1, France, 2009.
- [18] J. D. OWENS, W. J. DALLY, R. HO, D. JAYASIMHA, S. W. KECKLER et L.-S. PEH, « Research challenges for on-chip interconnection networks », *IEEE micro*, vol. 27, no. 5, p. 96–108, 2007.
- [19] C. NICOPOULOS, V. NARAYANAN et C. R. DAS, *Network-on-chip architectures: A holistic design exploration*, vol. 45. Springer Science & Business Media, 2009.
- [20] F. DUBOIS, *Une méthodologie de conception de modèles analytiques de surface et de puissance de réseaux sur puce hautement paramétriques basée sur une méthode d'apprentissage automatique*. Thèse doctorat, Université de Grenoble, France, 2013.
- [21] S. EVAIN, *μ Spider Environnement de Conception de Réseau sur Puce*. Thèse doctorat, Institut National des Sciences Appliquées de Rennes, France, 2006.
- [22] P. GUERRIER et A. GREINER, « A generic architecture for on-chip packet-switched interconnections », in *Design, Automation, and Test in Europe*, p. 111–123, Springer, 2008.
- [23] K. TATAS, K. SIOZIOS, D. SOUDRIS et A. JANTSCH, *Designing 2D and 3D network-on-chip architectures*. Springer, 2014.
- [24] T. BJERREGAARD et S. MAHADEVAN, « A survey of research and practices of network-on-chip », *ACM Computing Surveys (CSUR)*, vol. 38, no. 1, p. 1–es, 2006.
- [25] G. DE MICHELI, C. SEICULESCU, S. MURALI, L. BENINI, F. ANGIOLINI et A. PULINI, « Networks on chips: from research to products », in *Proceedings of the 47th Design Automation Conference*, p. 300–305, 2010.

- [26] É. COTA, A. de MORAIS AMORY et M. S. LUBASZEWSKI, *Reliability, Availability and Serviceability of Networks-on-chip*. Springer Science & Business Media, 2011.
- [27] S. MA, L. HUANG, M. LAI et W. SHI, *Networks-on-Chip: from implementations to programming paradigms*. Morgan Kaufmann, 2014.
- [28] X. LI, *Méthodologie de conception automatique pour multiprocesseur sur puce hétérogène*. Thèse doctorat, Université Paris Sud-Paris XI, 2009.
- [29] A. CHARIETE, *Approches d'optimisation et de personnalisation des réseaux sur puce (NoC: Networks on Chip)*. Thèse doctorat, Université de Technologie de Belfort-Montbéliard (UTBM), Belfort, France, 2014.
- [30] E. MORÉAC, *Optimisation de la consommation d'énergie et de la latence dans les réseaux sur puces*. Thèse doctorat, Université de Bretagne Sud, France, 2017.
- [31] A. B. ABDALLAH, *Advanced multicore Systems-On-Chip*. Springer, 2017.
- [32] S. PASRICHA et N. DUTT, *On-chip communication architectures: system on chip interconnect*. Morgan Kaufmann, 2010.
- [33] A. AGARWAL, C. ISKANDER et R. SHANKAR, « Survey of network on chip (NoC) architectures & contributions », *Journal of engineering, Computing and Architecture*, vol. 3, no. 1, p. 21–27, 2009.
- [34] C. RADU, *Optimized algorithms for network-on-chip application mapping*. Thèse doctorat, University of Sibiu, Lucian Blaga, Sibiu, Romania, 2011.
- [35] F. GEBALI, H. ELMILIGI et M. W. EL-KHARASHI, *Networks-on-chips: theory and practice*. CRC press, 2011.
- [36] S. KUNDU et S. CHATTOPADHYAY, *Network-on-Chip: the next generation of system-on-chip integration*. Taylor & Francis, 2014.
- [37] W. J. DALLY et B. P. TOWLES, *Principles and practices of interconnection networks*. Elsevier, 2004.
- [38] A. B. ACHBALLAH et S. B. SAOUD, « A survey of network-on-chip tools », *arXiv preprint arXiv:1312.2976*, 2013.
- [39] M. PALESI et M. DANESHTALAB, *Routing algorithms in networks-on-chip*. Springer, 2014.
- [40] C. C. LIU, J.-H. CHEN, R. MANOHAR et S. TIWARI, « Mapping system-on-chip designs from 2-D to 3-D ICs », in *2005 IEEE International Symposium on Circuits and Systems*, p. 2939–2942, IEEE, 2005.
- [41] L. ZERIOUL, *Modélisation comportementale d'un réseau sur puce basé sur des interconnexions RF*. Thèse doctorat, Université de Cergy Pontoise, France, 2015.
- [42] B. VAIDYANATHAN, W.-L. HUNG, F. WANG, Y. XIE, V. NARAYANAN et M. J. IRWIN, « Architecting microprocessor components in 3D design space », in *20th International Conference on VLSI Design held jointly with 6th International Conference on Embedded Systems (VLSID'07)*, p. 103–108, IEEE, 2007.

- [43] N. JAIN et M. PATEL, « A review of the design challenges for the 3-d on chips network paradigms », *International Journal of Computer Application*, p. 11–15, 2017.
- [44] J. KIM, C. NICOPOULOS, D. PARK, R. DAS, Y. XIE, V. NARAYANAN, M. S. YOUSIF et C. R. DAS, « A novel dimensionally-decomposed router for on-chip communication in 3D architectures », in *Proceedings of the 34th annual international symposium on Computer architecture*, p. 138–149, 2007.
- [45] W. R. DAVIS, J. WILSON, S. MICK, J. XU, H. HUA, C. MINEO, A. M. SULE, M. STEER et P. D. FRANZON, « Demystifying 3D ics: the pros and cons of going vertical », *IEEE Design & Test of Computers*, vol. 22, no. 6, p. 498–510, 2005.
- [46] R. YAREMA, « The via revolution », in *19th International Workshop on Vertex Dectectors-VERTEX 2010*, p. 1–11, 2010.
- [47] J. DUATO, S. YALAMANCHILI et L. NI, *Interconnection networks*. Morgan Kaufmann, 2003.
- [48] G. ASCIA, V. CATANIA et M. PALESI, « Mapping cores on network-on-chip », *International Journal of Computational Intelligence Research*, vol. 1, no. 1, p. 109–126, 2005.
- [49] M. V. C. da SILVA, N. NEDJAH et L. de MACEDO MOURELLE, « Optimal IP assignment for efficient NoC-based system implementation using NSGA-II and microga », *Int. J. Comput. Intell. Syst.*, vol. 2, no. 2, p. 115–123, 2009.
- [50] N. NEDJAH, M. V. C. da SILVA et L. de MACEDO MOURELLE, « Customized computer-aided application mapping on NoC infrastructure using multi-objective optimization », *J. Syst. Archit.*, vol. 57, no. 1, p. 79–94, 2011.
- [51] L. D. R. de Souza e SILVA JUNIOR, N. NEDJAH et L. de MACEDO MOURELLE, « Static routing for applications mapped on NoC platform using ant colony algorithms », *Int. J. High Perform. Syst. Archit.*, vol. 4, no. 1, p. 57–64, 2012.
- [52] L. S. JUNIOR, N. NEDJAH et L. de MACEDO MOURELLE, « Aco-based algorithms for search and optimization of routes in NoC platform », *J. Univers. Comput. Sci.*, vol. 18, no. 7, p. 917–936, 2012.
- [53] N. NEDJAH, L. S. JUNIOR et L. de MACEDO MOURELLE, « Congestion-aware ant colony based routing algorithms for efficient application execution on network-on-chip platform », *Expert Syst. Appl.*, vol. 40, no. 16, p. 6661–6673, 2013.
- [54] R. K. JENA et G. K. SHARMA, « A multi-objective evolutionary algorithm based optimization model for network-on-chip synthesis », in *Fourth International Conference on Information Technology (ITNG'07)*, p. 977–982, IEEE, 2007.
- [55] M. VINÍCIUS, C. da SILVA, N. NEDJAH et L. de MACEDO MOURELLE, « Optimal ip assignment for efficient NoC-based system implementation using NSGA-II and MicroGA », *International Journal of Computational Intelligence Systems*, vol. 2, no. 2, p. 115–123, 2009.

- [56] C. RADU, M. S. MAHBUB et L. VINTAN, « Developing domain-knowledge evolutionary algorithms for network-on-chip application mapping », *Microprocessors and Microsystems*, vol. 37, no. 1, p. 65–78, 2013.
- [57] Q.-q. LE, G.-w. YANG, W. N. HUNG, X.-y. SONG et F.-y. FAN, « Performance-driven assignment and mapping for reliable networks-on-chips », *Journal of Zhejiang University SCIENCE C*, vol. 15, no. 11, p. 1009–1020, 2014.
- [58] R. POP et S. KUMAR, « A survey of techniques for mapping and scheduling applications to network on chip systems », *School of Engineering, Jonkoping University, Research Report*, vol. 4, no. 4, 2004.
- [59] M. JANIDARMIAN et A. R. FEKR, « A survey of meta-heuristic solution methods for mapping problem in network-on-chips », in *Advanced Materials Research*, vol. 403, p. 3994–4008, Trans Tech Publ, 2012.
- [60] P. K. SAHU et S. CHATTOPADHYAY, « A survey on application mapping strategies for network-on-chip design », *Journal of systems architecture*, vol. 59, no. 1, p. 60–76, 2013.
- [61] M. SACANAMBOY, F. BOLANOS et R. NIETO, « A primer for mapping techniques on NoC systems », in *ESA (2014) Proceedings 12th International Conference on Embedded Systems and Applications*, p. 70–75, 2014.
- [62] T. NESRINE, B. DJAMEL et M. ALI, « A classification and evaluation framework for NoC mapping strategies », *Journal of Circuits, Systems and Computers*, vol. 26, no. 02, p. 1730001, 2017.
- [63] P. K. HAMEDANI, S. HESSABI, H. SARBAZI-AZAD et N. E. JERGER, « Exploration of temperature constraints for thermal aware mapping of 3D networks on chip », in *2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, p. 499–506, IEEE, 2012.
- [64] K. MANNA, S. CHATTOPADHYAY et I. SENGUPTA, « Through silicon via placement and mapping strategy for 3D mesh based network-on-chip », in *2014 22nd International Conference on Very Large Scale Integration (VLSI-SoC)*, p. 1–6, IEEE, 2014.
- [65] P. K. SAHU, K. MANNA, T. SHAH et S. CHATTOPADHYAY, « A constructive heuristic for application mapping onto mesh based network-on-chip », *Journal of Circuits, Systems and Computers*, vol. 24, no. 08, p. 1550126, 2015.
- [66] P. WADHWANI, N. CHOUDHARY et D. SINGH, « Energy efficient mapping in 3D mesh communication architecture for NoC », *Global Journal of Computer Science and Technology*, 2013.
- [67] Y. NALCI, P. KULLU, S. TOSUN et O. OZTURK, « Ilp formulation and heuristic method for energy-aware application mapping on 3D-NoCs », *The Journal of Supercomputing*, vol. 77, no. 3, p. 2667–2680, 2021.

- [68] G. FENG, F. GE, S. YU et N. WU, « A thermal-aware mapping algorithm for 3D mesh network-on-chip architecture », in *2013 IEEE 10th International Conference on ASIC*, p. 1–4, IEEE, 2013.
- [69] H. ELMILIGI, F. GEBALI et M. W. EL-KHARASHI, « Power-aware mapping for 3D-NoC designs using genetic algorithms », *Procedia Computer Science*, vol. 34, p. 538–543, 2014.
- [70] L. SHEN, N. WU, G. YAN et F. GE, « Thermal-aware task mapping for communication energy minimization on 3D NoC », *IEICE Electronics Express*, p. 14–20170900, 2017.
- [71] H. HE, F. FANG et W. WANG, « Improved simulated annealing genetic algorithm based low power mapping for 3D NoC », in *MATEC web of conferences*, vol. 232, p. 02022, EDP Sciences, 2018.
- [72] Y. GAN, H. GUO et Z. ZHOU, « 3D NoC low-power mapping optimization based on improved genetic algorithm », *Micromachines*, vol. 12, no. 10, p. 1217, 2021.
- [73] P. K. SAHU, T. SHAH, K. MANNA et S. CHATTOPADHYAY, « Application mapping onto mesh-based network-on-chip using discrete particle swarm optimization », *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 2, p. 300–312, 2013.
- [74] V. JHA, S. DEOL, M. JHA et G. SHARMA, « Energy and latency aware application mapping algorithm & optimization for homogeneous 3D network on chip », *arXiv preprint arXiv:1404.2512*, 2014.
- [75] C. HUANG, D. ZHANG et G. SONG, « Low-power mapping algorithm for three-dimensional network-on-chip based on diversity-controlled quantum-behaved particle swarm optimization », *Journal of Algorithms & Computational Technology*, vol. 10, no. 3, p. 176–186, 2016.
- [76] C. HUANG, D. ZHANG et G. SONG, « A novel mapping algorithm for three-dimensional network on chip based on quantum-behaved particle swarm optimization », *Frontiers of Computer Science*, vol. 11, no. 4, p. 622–631, 2017.
- [77] A. MOSAYYEBZADEH, M. M. AMIRASKI et S. HESSABI, « Thermal and power aware task mapping on 3D network on chip », *Computers & Electrical Engineering*, vol. 51, p. 157–167, 2016.
- [78] J. LI, G. SONG, Y. MA, C. WANG, B. ZHU, Y. CHAI et J. RONG, « Bat algorithm based low power mapping methods for 3D network-on-chips », in *National Conference of Theoretical Computer Science*, p. 277–295, Springer, 2017.
- [79] M. O. AGYEMAN, A. AHMADINIA et N. BAGHERZADEH, « Energy and performance-aware application mapping for inhomogeneous 3D networks-on-chip », *Journal of Systems Architecture*, vol. 89, p. 103–117, 2018.

- [80] W. GAO, Z. QIAN et P. ZHOU, « Reliability-and performance-driven mapping for regular 3D NoCs using a novel latency model and simulated allocation », *Integration*, vol. 65, p. 351–361, 2019.
- [81] J. WANG, L. LI, H. PAN, S. HE et R. ZHANG, « Latency-aware mapping for 3D NoC using rank-based multi-objective genetic algorithm », in *2011 9th IEEE International Conference on ASIC*, p. 413–416, IEEE, 2011.
- [82] H. DING, H. GU, Y. YANG et D. FAN, « 3D networks-on-chip mapping targeting minimum signal TSVs », *IEICE Electronics Express*, vol. 10, no. 18, p. 20130518–20130518, 2013.
- [83] M. O. AGYEMAN et A. AHMADINIA, « Optimised application specific architecture generation and mapping approach for heterogeneous 3D networks-on-chip », in *2013 IEEE 16th International Conference on Computational Science and Engineering*, p. 794–801, IEEE, 2013.
- [84] J. SEPÚLVEDA, G. GOGNIAT, R. PIRES, W. CHAU et M. STRUM, « An evolutive approach for designing thermal and performance-aware heterogeneous 3D-NoCs », in *2013 26th Symposium on Integrated Circuits and Systems Design (SBCCI)*, p. 1–6, IEEE, 2013.
- [85] M. J. SEPÚLVEDA, G. GOGNIAT, D. M. SEPÚLVEDA, R. PIRES, W. J. CHAU et M. STRUM, « 3DMIA: a multi-objective artificial immune algorithm for 3D-MPSoC multi-application 3D-NoC mapping », in *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, p. 167–168, 2013.
- [86] K. BHARDWAJ et P. S. MANE, « C3map and arpso based mapping algorithms for energy-efficient regular 3D NoC architectures », in *Technical papers of 2014 international symposium on VLSI design, automation and test*, p. 1–4, IEEE, 2014.
- [87] M. JANIDARMIAN, A. KHADEMZADEH et M. TAVANPOUR, « Onyx: a new heuristic bandwidth-constrained mapping of cores onto tile-based network on chip », *IEICE Electronics Express*, vol. 6, no. 1, p. 1–7, 2009.
- [88] S. SAEIDI, A. KHADEMZADEH et A. MEHRAN, « SMAP: an intelligent mapping tool for network on chip », in *2007 International Symposium on Signals, Circuits and Systems*, vol. 1, p. 1–4, IEEE, 2007.
- [89] S. SAEIDI, A. KHADEMZADEH et F. VARDI, « Crinkle: a heuristic mapping algorithm for network on chip », *IEICE Electronics Express*, vol. 6, no. 24, p. 1737–1744, 2009.
- [90] H. ZIAEEZIABARI et A. PATOOGHY, « 3D-AMAP: a latency-aware task mapping onto 3D mesh-based NoCs with partially-filled TSVs », in *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, p. 593–597, IEEE, 2017.
- [91] M. Z. DAGELEH et M. A. J. JAMALI, « V-CastNet3D: a novel clustering-based mapping in 3D network on chip », *Nano communication networks*, vol. 18, p. 51–61, 2018.

- [92] S. TOSUN, « New heuristic algorithms for energy aware application mapping and routing on mesh-based NoCs », *Journal of Systems Architecture*, vol. 57, no. 1, p. 69–78, 2011.
- [93] A. ALAGARSAMY, L. GOPALAKRISHNAN, S. MAHILMARAN et S.-B. KO, « A self-adaptive mapping approach for network on chip with low power consumption », *IEEE Access*, vol. 7, p. 84066–84081, 2019.
- [94] A. ALAGARSAMY, L. GOPALAKRISHNAN et S.-B. KO, « KBMA: a knowledge-based multi-objective application mapping approach for 3D NoC », *IET Computers & Digital Techniques*, vol. 13, no. 4, p. 324–334, 2019.
- [95] J. M. JOSEPH, D. ERMEL, L. BAMBERG, A. GARCÍA-ORITZ et T. PIONTECK, « Application-specific soc design using core mapping to 3D mesh NoCs with nonlinear area optimization and simulated annealing », *Technologies*, vol. 8, no. 1, p. 10, 2020.
- [96] A. B. AHMED et A. B. ABDALLAH, « LA-XYZ: low latency, high throughput look-ahead routing algorithm for 3D network-on-chip (3D-NoC) architecture », in *2012 IEEE 6th International Symposium on Embedded Multicore SoCs*, p. 167–174, IEEE, 2012.
- [97] S. AKBARI, A. SHAFIEE, M. FATHY et R. BERANGI, « Afra : A low cost high performance reliable routing for 3D mesh NoCs », in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, p. 332–337, IEEE, 2012.
- [98] G.-M. CHIU, « The odd-even turn model for adaptive routing », *IEEE Transactions on parallel and distributed systems*, vol. 11, no. 7, p. 729–738, 2000.
- [99] S. PASRICHA et Y. ZOU, « A low overhead fault tolerant routing scheme for 3D networks-on-chip », in *2011 12th International Symposium on Quality Electronic Design*, p. 1–8, IEEE, 2011.
- [100] C. FENG, M. ZHANG, J. LI, J. JIANG, Z. LU et A. JANTSCH, « A low-overhead fault-aware deflection routing algorithm for 3D network-on-chip », in *2011 IEEE Computer Society Annual Symposium on VLSI*, p. 19–24, IEEE, 2011.
- [101] M. ZHU, J. LEE et K. CHOI, « An adaptive routing algorithm for 3D mesh NoC with limited vertical bandwidth », in *2012 IEEE/IFIP 20th International Conference on VLSI and System-on-Chip (VLSI-SoC)*, p. 18–23, IEEE, 2012.
- [102] M. EBRAHIMI, X. CHANG, M. DANESHTALAB, J. PLOSIŁA, P. LILJEBERG et H. TENHUNEN, « Dyxyz : Fully adaptive routing algorithm for 3D NoCs », in *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, p. 499–503, IEEE, 2013.
- [103] M. EBRAHIMI, M. DANESHTALAB, P. LILJEBERG et H. TENHUNEN, « Fault-tolerant method with distributed monitoring and management technique for 3D stacked meshes », in *The 17th CSI International Symposium on Computer Architecture & Digital Systems (CADSD 2013)*, p. 93–98, IEEE, 2013.

- [104] M. EBRAHIMI, M. DANESHTALAB et J. PLOSILA, « Fault-tolerant routing approach for 3D stacked meshes », in *Proceedings of the 2014 Workshop on Manufacturable and Dependable Multicore Architectures at Nanoscale*, p. 5–10, 2014.
- [105] X. LEI, X. JIANG, L. ZENG et T. WATANABE, « Vertical-mesh-conscious-dynamic routing algorithm for 3D NoCs », in *TENCON 2015-2015 IEEE Region 10 Conference*, p. 1–6, IEEE, 2015.
- [106] L. ZENG, T. PAN, X. JIANG et T. WATANABE, « A performance enhanced adaptive routing algorithm for 3D network-on-chips », in *TENCON 2015-2015 IEEE Region 10 Conference*, p. 1–4, IEEE, 2015.
- [107] J. ZHOU, H. LI, T. WANG et X. LI, « LOFT: a low-overhead fault-tolerant routing scheme for 3D NoCs », *Integration*, vol. 52, p. 41–50, 2016.
- [108] R. SALAMAT, M. KHAYAMBASHI, M. EBRAHIMI et N. BAGHERZADEH, « A resilient routing algorithm with formal reliability analysis for partially connected 3D-NoCs », *IEEE Transactions on Computers*, vol. 65, no. 11, p. 3265–3279, 2016.
- [109] N. NOSRATI et H. S. SHAHHOSEINI, « G-CARA: a global congestion-aware routing algorithm for traffic management in 3D networks-on-chip », in *2017 Iranian Conference on Electrical Engineering (ICEE)*, p. 2188–2193, IEEE, 2017.
- [110] M. DANESHTALAB, A. SOBHANI, A. AFZALI-KUSHA, O. FATEMI et Z. NAVABI, « NoC hot spot minimization using antnet dynamic routing algorithm », in *IEEE 17th International Conference on Application-specific Systems, Architectures and Processors (ASAP'06)*, p. 33–38, IEEE, 2006.
- [111] L. SILVA JUNIOR, N. NEDJAH et L. DE MACEDO MOURELLE, « Efficient routing in network-on-chip for 3D topologies », *International Journal of Electronics*, vol. 102, no. 10, p. 1695–1712, 2015.
- [112] M. DOURI, S. ELBERNOUSSI et H. LAKHBAB, « Cours des méthodes de résolution exactes heuristiques et métaheuristiques », *Université Mohamed V, Faculté des sciences de Rabat*, p. 5–87, 2009.
- [113] H. CHRISTOS, « Papadimitriou: computational complexity », *Addison-Wesley*, vol. 2, no. 3, p. 4, 1994.
- [114] R. M. KARP, « Reducibility among combinatorial problems », in *Complexity of computer computations*, p. 85–103, Springer, 1972.
- [115] S. A. COOK, « The complexity of theorem-proving procedures », in *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA* (M. A. HARRISON, R. B. BANERJI et J. D. ULLMAN, édés), p. 151–158, ACM, 1971.
- [116] E.-G. TALBI, *Metaheuristics: from design to implementation*, vol. 74. John Wiley & Sons, 2009.

- [117] J. PUCHINGER et G. R. RAIDL, « Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification », in *International work-conference on the interplay between natural and artificial computation*, p. 41–53, Springer, 2005.
- [118] R. H. BARTELS et G. H. GOLUB, « The simplex method of linear programming using lu decomposition », *Communications of the ACM*, vol. 12, no. 5, p. 266–268, 1969.
- [119] D. P. DE FARIAS et B. VAN ROY, « The linear programming approach to approximate dynamic programming », *Operations research*, vol. 51, no. 6, p. 850–865, 2003.
- [120] NARENDRA et FUKUNAGA, « A branch and bound algorithm for feature subset selection », *IEEE Transactions on computers*, vol. 100, no. 9, p. 917–922, 1977.
- [121] S. KIRKPATRICK, C. D. GELATT et M. P. VECCHI, « Optimization by simulated annealing », *science*, vol. 220, no. 4598, p. 671–680, 1983.
- [122] F. GLOVER et M. LAGUNA, « Tabu search », in *Handbook of combinatorial optimization*, p. 2093–2229, Springer, 1998.
- [123] J. H. HOLLAND *et al.*, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.
- [124] J. R. KOZA et R. POLI, « Genetic programming », in *Search methodologies*, p. 127–164, Springer, 2005.
- [125] R. STORN et K. PRICE, « Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces », *Journal of global optimization*, vol. 11, no. 4, p. 341–359, 1997.
- [126] N. NEDJAH, L. de MACEDO MOURELLE et R. G. MORAIS, « Inspiration-wise swarm intelligence meta-heuristics for continuous optimisation : a survey - part I », *Int. J. Bio Inspired Comput.*, vol. 15, no. 4, p. 207–223, 2020.
- [127] N. NEDJAH, L. de MACEDO MOURELLE et R. G. MORAIS, « Inspiration-wise swarm intelligence meta-heuristics for continuous optimisation : a survey - part II », *Int. J. Bio Inspired Comput.*, vol. 16, no. 4, p. 195–212, 2020.
- [128] N. NEDJAH, L. de MACEDO MOURELLE et R. G. MORAIS, « Inspiration-wise swarm intelligence meta-heuristics for continuous optimisation : a survey - part III », *Int. J. Bio Inspired Comput.*, vol. 17, no. 4, p. 199–214, 2021.
- [129] J. KENNEDY et R. EBERHART, « Particle swarm optimization », in *Proceedings of ICNN’95-international conference on neural networks*, vol. 4, p. 1942–1948, IEEE, 1995.
- [130] M. DORIGO, M. BIRATTARI et T. STUTZLE, « Ant colony optimization », *IEEE computational intelligence magazine*, vol. 1, no. 4, p. 28–39, 2006.

- [131] D. KARABOGA et B. BASTURK, « A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm », *Journal of global optimization*, vol. 39, no. 3, p. 459–471, 2007.
- [132] X.-S. YANG et X. HE, « Bat algorithm: literature review and applications », *International Journal of Bio-inspired computation*, vol. 5, no. 3, p. 141–149, 2013.
- [133] S. MIRJALILI, S. M. MIRJALILI et A. LEWIS, « Grey wolf optimizer », *Advances in engineering software*, vol. 69, p. 46–61, 2014.
- [134] S. MIRJALILI et A. LEWIS, « The whale optimization algorithm », *Advances in engineering software*, vol. 95, p. 51–67, 2016.
- [135] S. Z. MIRJALILI, S. MIRJALILI, S. SAREMI, H. FARIS et I. ALJARAH, « Grasshopper optimization algorithm for multi-objective optimization problems », *Applied Intelligence*, vol. 48, no. 4, p. 805–820, 2018.
- [136] E. K. BURKE, M. R. HYDE, G. KENDALL, G. OCHOA, E. ÖZCAN et J. R. WOODWARD, « A classification of hyper-heuristic approaches : revisited », *Handbook of metaheuristics*, p. 453–477, 2019.
- [137] E. K. BURKE, M. GENDREAU, M. HYDE, G. KENDALL, G. OCHOA, E. ÖZCAN et R. QU, « Hyper-heuristics : A survey of the state of the art », *Journal of the Operational Research Society*, vol. 64, p. 1695–1724, 2013.
- [138] K. CHAKHLEVITCH et P. COWLING, « Hyperheuristics : recent developments », *Adaptive and multilevel metaheuristics*, p. 3–29, 2008.
- [139] E.-G. TALBI, M. BASSEUR, A. J. NEBRO et E. ALBA, « Multi-objective optimization using metaheuristics: non-standard algorithms », *International Transactions in Operational Research*, vol. 19, no. 1-2, p. 283–305, 2012.
- [140] V. PARETO, *Cours d'économie politique*, vol. 1. Librairie Droz, 1964.
- [141] S. CONSOLI, *The development and application of metaheuristics for problems in graph theory: a computational study*. Thèse doctorat, Brunel University, School of Information Systems, Computing and Mathematics, Royaume-Uni, 2008.
- [142] J. BROWNLEE, *Clever algorithms: nature-inspired programming recipes*. Jason Brownlee, 2011.
- [143] K. DEB, « Multi-objective optimization. search methodologies », *Search Methodol*, vol. 2014, p. 403–449, 2014.
- [144] D. BEASLEY, D. R. BULL et R. R. MARTIN, « An overview of genetic algorithms: part 1, fundamentals », *University computing*, vol. 15, no. 2, p. 56–69, 1993.
- [145] D. E. GOLDBERG, J. RICHARDSON *et al.*, « Genetic algorithms with sharing for multimodal function optimization », in *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, p. 41–49, Hillsdale, NJ : Lawrence Erlbaum, 1987.

- [146] K. FUJITA, N. HIROKAWA, S. AKAGI, S. KITAMURA et H. YOKOHATA, « Multi-objective optimal design of automotive engine using genetic algorithm », in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 80326, p. V002T02A081, American Society of Mechanical Engineers, 1998.
- [147] C.-L. HWANG, S. R. PAIDY, K. YOON et A. S. M. MASUD, « Mathematical programming with multiple objectives: a tutorial », *Computers & Operations Research*, vol. 7, no. 1-2, p. 5–31, 1980.
- [148] H. MEUNIER, *Algorithmes évolutionnaires parallèles pour l'optimisation multi-objectif de réseaux de télécommunications mobiles*. Thèse doctorat, Université de Lille 1, France, 2002.
- [149] R. T. MARLER et J. S. ARORA, « Survey of multi-objective optimization methods for engineering », *Structural and multidisciplinary optimization*, vol. 26, no. 6, p. 369–395, 2004.
- [150] S. SAYIN et S. KARABATI, « Theory and methodology a bicriteria approach to the two-machine flow shop scheduling problem », *European journal of operational research*, vol. 113, no. 2, p. 435–449, 1999.
- [151] S. ROSTAMI, F. NERI et M. EPITROPAKIS, « Progressive preference articulation for decision making in multi-objective optimisation problems », *Integrated Computer-Aided Engineering*, vol. 24, no. 4, p. 315–335, 2017.
- [152] A. CHARNES et W. W. COOPER, « Goal programming and multiple objective optimizations: part 1 », *European journal of operational research*, vol. 1, no. 1, p. 39–54, 1977.
- [153] Y. HAIMES, « On a bicriterion formulation of the problems of integrated system identification and system optimization », *IEEE transactions on systems, man, and cybernetics*, vol. 1, no. 3, p. 296–297, 1971.
- [154] M. A. IQUEBAL, « Genetic algorithms and their applications: an overview », *White Paper*, 2009.
- [155] C. M. FONSECA et P. J. FLEMING, « Multiobjective genetic algorithms made easy: selection sharing and mating restriction », in *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, p. 45–52, IET, 1995.
- [156] N. SRINIVAS et K. DEB, « Multiobjective optimization using nondominated sorting in genetic algorithms », *Evolutionary computation*, vol. 2, no. 3, p. 221–248, 1994.
- [157] J. HORN, N. NAFPLIOTIS et D. E. GOLDBERG, « A niched pareto genetic algorithm for multiobjective optimization », in *Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence*, p. 82–87, Ieee, 1994.

- [158] E.-g. TALBI, « Métaheuristiques pour l'optimisation combinatoire multi-objectif: état de l'art », *Rapport CNET (France Telecom) Octobre*, 1999.
- [159] D. A. VAN VELDHUIZEN et G. B. LAMONT, « On measuring multiobjective evolutionary algorithm performance », in *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512)*, vol. 1, p. 204–211, IEEE, 2000.
- [160] J. R. SCHOTT, *Fault tolerant design using single and multicriteria genetic algorithm optimization*. Thèse doctorat, Massachusetts Institute of Technology, Etat unis, 1995.
- [161] H. MEUNIER, E.-G. TALBI et P. REININGER, « A multiobjective genetic algorithm for radio network optimization », in *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512)*, vol. 1, p. 317–324, IEEE, 2000.
- [162] E. ZITZLER et L. THIELE, « Multiobjective optimization using evolutionary algorithms – a comparative case study », in *International conference on parallel problem solving from nature*, p. 292–301, Springer, 1998.
- [163] L. WHILE, L. BRADSTREET et L. BARONE, « A fast way of calculating exact hypervolumes », *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 1, p. 86–95, 2011.
- [164] T. AMEUR et M. ASSAS, « Modified pso algorithm for multi-objective optimization of the cutting parameters », *Production Engineering*, vol. 6, no. 6, p. 569–576, 2012.
- [165] R. EBERHART et J. KENNEDY, « A new optimizer using particle swarm theory », in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, p. 39–43, Ieee, 1995.
- [166] A. P. ENGELBRECHT, *Computational intelligence: an introduction*. John Wiley & Sons, 2007.
- [167] S. S. MADEIRO, C. J. BASTOS-FILHO, F. B. L. NETO et E. M. FIGUEIREDO, « Adaptative clustering particle swarm optimization », in *2009 IEEE International Symposium on Parallel & Distributed Processing*, p. 1–8, IEEE, 2009.
- [168] X.-L. LI, R. SERRA et J. OLIVIER, « An investigation of particle swarm optimization topologies in structural damage detection », *Applied Sciences*, vol. 11, no. 11, p. 5144, 2021.
- [169] S. P. GHOSHAL, « Optimizations of pid gains by particle swarm optimizations in fuzzy based automatic generation control », *Electric Power Systems Research*, vol. 72, no. 3, p. 203–212, 2004.
- [170] N. NEDJAH et L. d. M. MOURELLE, « Evolutionary multi-objective optimisation: a survey », *International Journal of Bio-Inspired Computation*, vol. 7, no. 1, p. 1–25, 2015.
- [171] O. ROUDENKO, *Application des algorithmes évolutionnaires aux problèmes d'optimisation multi-objectif avec contraintes*. Thèse doctorat, Ecole Polytechnique de paris, France, 2004.

- [172] S. LALWANI, S. SINGHAL, R. KUMAR et N. GUPTA, « A comprehensive survey: applications of multi-objective particle swarm optimization (mopso) algorithm », *Transactions on combinatorics*, vol. 2, no. 1, p. 39–101, 2013.
- [173] Y. LIU, « A fast and elitist multi-objective particle swarm algorithm: NSPSO », in *2008 IEEE International Conference on Granular Computing*, p. 470–475, IEEE, 2008.
- [174] K. E. PARSOPOULOS, D. K. TASOULIS, M. N. VRAHATIS *et al.*, « Multiobjective optimization using parallel vector evaluated particle swarm optimization », in *Proceedings of the IASTED international conference on artificial intelligence and applications (AIA 2004)*, vol. 2, p. 823–828, Citeseer, 2004.
- [175] M. R. SIERRA et C. A. C. COELLO, « Improving pso-based multi-objective optimization using crowding, mutation and ϵ - dominance », in *International conference on evolutionary multi-criterion optimization*, p. 505–519, Springer, 2005.
- [176] A. J. NEBRO, J. J. DURILLO, J. GARCIA-NIETO, C. C. COELLO, F. LUNA et E. ALBA, « SMPSO: a new pso-based metaheuristic for multi-objective optimization », in *2009 IEEE Symposium on computational intelligence in multi-criteria decision-making (MCDM)*, p. 66–73, IEEE, 2009.
- [177] K. FLEETWOOD, « An introduction to differential evolution », in *Proceedings of Mathematics and Statistics of Complex Systems (MASCOS) One Day Symposium, 26th November, Brisbane, Australia*, p. 785–791, 2004.
- [178] S. DAS et P. N. SUGANTHAN, « Differential evolution: a survey of the state-of-the-art », *IEEE transactions on evolutionary computation*, vol. 15, no. 1, p. 4–31, 2010.
- [179] V. FEOKTISTOV, *Differential evolution*. Springer, 2006.
- [180] M. F. AHMAD, N. A. M. ISA, W. H. LIM et K. M. ANG, « Differential evolution: a recent review based on state-of-the-art works », *Alexandria Engineering Journal*, 2021.
- [181] T. ROBIČ et B. FILIPIČ, « Differential evolution for multiobjective optimization », in *International conference on evolutionary multi-criterion optimization*, p. 520–533, Springer, 2005.
- [182] H. A. ABBASS et R. SARKER, « The pareto differential evolution algorithm », *International Journal on Artificial Intelligence Tools*, vol. 11, no. 04, p. 531–552, 2002.
- [183] N. K. MADAVAN, « Multiobjective optimization using a pareto differential evolution approach », in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, vol. 2, p. 1145–1150, IEEE, 2002.

- [184] K. E. PARSOPOULOS, D. K. TASOULIS, N. G. PAVLIDIS, V. P. PLAGIANAKOS et M. N. VRAHATIS, « Vector evaluated differential evolution for multiobjective optimization », in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*, vol. 1, p. 204–211, IEEE, 2004.
- [185] K. DEB, A. PRATAP, S. AGARWAL et T. MEYARIVAN, « A fast and elitist multiobjective genetic algorithm: NSGA-II », *IEEE transactions on evolutionary computation*, vol. 6, no. 2, p. 182–197, 2002.
- [186] R. ANGIRA et B. BABU, « Non-dominated sorting differential evolution (NSDE): an extension of differential evolution for multi-objective optimization. », in *IICAI*, p. 1428–1443, 2005.
- [187] F. XUE, A. C. SANDERSON et R. J. GRAVES, « Pareto-based multi-objective differential evolution », in *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, vol. 2, p. 862–869, IEEE, 2003.
- [188] F. GRUIAN et K. KUCHCINSKI, « LEneS: task scheduling for low-energy systems using variable supply voltage processors », in *Proceedings of the 2001 Asia and South Pacific design automation conference*, p. 449–455, 2001.
- [189] B. M. PANGRLE et D. D. GAJSKI, « Design tools for intelligent silicon compilation », *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 6, no. 6, p. 1098–1112, 1987.
- [190] H. G. LEE, N. CHANG, U. Y. OGRAS et R. MARCULESCU, « On-chip communication architecture exploration: a quantitative evaluation of point-to-point, bus, and network-on-chip approaches », *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 12, no. 3, p. 1–20, 2008.
- [191] N. A. KAPADIA et S. PASRICHA, « Vision : a framework for voltage island aware synthesis of interconnection networks-on-chip », in *Proceedings of the 21st edition of the great lakes symposium on Great lakes symposium on VLSI*, p. 31–36, 2011.
- [192] N. ALFARAJ, J. ZHANG, Y. XU et H. J. CHAO, « Hope : Hotspot congestion control for clos network on chip », in *Proceedings of the Fifth ACM/IEEE International Symposium on Networks-on-Chip*, p. 17–24, 2011.
- [193] G. P. NYCHIS, C. FALLIN, T. MOSCIBRODA, O. MUTLU et S. SESHAN, « On-chip networks from a networking perspective : Congestion and scalability in many-core interconnects », *ACM SIGCOMM computer communication review*, vol. 42, no. 4, p. 407–418, 2012.
- [194] R. DICK, D. RHODES et W. WOLF, « Tgff : task graphs for free », in *Proceedings of the Sixth International Workshop on Hardware/Software Codesign. (CODES/CA-SHE'98)*, p. 97–101, 1998.
- [195] J. POOVEY *et al.*, « Characterization of the eembc benchmark suite », *North Carolina State University*, vol. 32, p. 37–50, 2007.
- [196] D. RHODES, R. DICK et K. VALLERIO, « Task graph for free », 2012.

- [197] M. BOUGHERARA, N. NEDJAH, D. BENNOUAR, R. RAHMOUN, A. SADOK et L. de macedo MOURELLE, « Core/task associations for efficient application implementation on network-on-chip », in *2018 International Conference on Computer and Applications (ICCA)*, p. 18–22, IEEE, 2018.
- [198] M. BOUGHERARA, N. NEDJAH, L. D. M. MOURELLE, R. RAHMOUN, A. SADOK et D. BENNOUAR, « Ip assignment for efficient NoC-based system design using multi-objective particle swarm optimisation », *International Journal of Bio-Inspired Computation*, vol. 12, no. 4, p. 203–213, 2018.
- [199] M. BOUGHERARA, R. KEMCHA, N. NEDJAH, D. BENNOUAR et L. de MACEDO MOURELLE, « Ip assignment optimization for an efficient NoC-based system using multi-objective differential evolution », in *International Conference on Metaheuristics and Nature Inspired computing (META'2018)*, p. 435–444, Springer, 2018.
- [200] M. V. C. DA SILVA, N. NEDJAH et L. de MACEDO MOURELLE, « Evolutionary ip assignment for efficient NoC-based system design using multi-objective optimization », in *2009 IEEE Congress on Evolutionary Computation*, p. 2257–2264, IEEE, 2009.
- [201] M. BOUGHERARA, N. NEDJAH, D. BENNOUAR, R. KEMCHA et L. d. MACEDO MOURELLE, « Efficient application mapping onto three-dimensional network-on-chips using multi-objective particle swarm optimization », in *International Conference on Computational Science and Its Applications*, p. 654–670, Springer, 2019.
- [202] M. BOUGHERARA, N. NEDJAH, D. BENNOUAR, R. KEMCHA et L. d. MACEDO MOURELLE, « Application mapping onto 3D NoCs using differential evolution », in *International Conference on Computational Science and Its Applications*, p. 89–102, Springer, 2020.
- [203] V. CATANIA, A. MINEO, S. MONTELEONE, M. PALESI et D. PATTI, « Noxim: an open, extensible and cycle-accurate network on chip simulator », in *2015 IEEE 26th international conference on application-specific systems, architectures and processors (ASAP)*, p. 162–163, IEEE, 2015.
- [204] M. BOUGHERARA, N. NEDJAH, D. BENNOUAR et L. d. M. MOURELLE, « Routing in 3D NoCs using genetic algorithm and particle swarm optimization », in *International Conference on Computational Science and Its Applications*, p. 601–613, Springer, 2023.
- [205] E. ALIKHAH-ASL et M. RESHADI, « XY-axis and distance based NoC mapping (XY-ADB) », in *2016 8th International Symposium on Telecommunications (IST)*, p. 678–683, IEEE, 2016.

ANNEXE A - Processeurs / Tâches de référence

Cette annexe présente des détails sur le contenu du référentiel E3S, disponible sur le site électronique <http://ziyang.eecs.northwestern.edu/~dickrp/e3s/>. Les données sont structurées en cinq sections, répertoriant les types de processeurs et de tâches disponibles, les relations processeur/tâche et tâche/processeur/IP, ainsi que les caractéristiques d'intérêt de 6 processeurs utilisés à titre d'exemple dans les explications fournies au chapitre 6 et au chapitre 7.

Liste des processeurs utilisé dans les testes

La Table A.1 présente les 17 types de processeurs disponibles dans la bibliothèque de référence utilisé dans les testes, ainsi que leurs fréquences de fonctionnement respectives et le nombre total de tâches différentes que chaque processeur permet de mettre en œuvre.

TABLE A.1 – Liste de processeurs de la bibliothèque

ID Processeur	Nom Processeur	Fréquence (Mhz)	Nombre de tâche
0	AMD ElanSC520	133	46
1	AMD K6-2E	450	6
2	AMD K6-2E	400	46
3	AMD K6-2E+	500	46
4	AMD K6-III E+	550	41
5	Analog Devices 21065L	60	17
6	IBM PowerPC 405GP	266	46
7	IBM PowerPC 750CX	500	36
8	IDT32334	100	19
9	IDT79RC32364	100	22
10	IDT79RC32V334	150	19
11	IDT79RC64575	250	22
12	Imsys Cjip	40	4
13	Motorola MPC555	40	17
14	NEC VR5432	167	46
15	ST20C2	50	30
16	TI TMS320C6203	300	17

Liste de tâches

La Table A.2 représente les 46 types de tâches différents utilisés dans la bibliothèque de référence. Les tâches sont décrites par la fonctionnalité principale qu'elles implémentent, ainsi que par la taille des données d'entrée dans des cas spécifiques.

TABLE A.2 – Liste de tâches utilisées dans la bibliothèque

Type de la tâche	Description de la tâche
0, 1	Angle to Time Conversion, Basic floating point,
2, 3	Bit Manipulation, Cache Buster
4, 5	CAN Remote Data Request, Fast Fourier Transform (Auto/Indust. Version)
6	Finite Impulse Response Filter (Auto/Indust. Vers)
7, 8	Infinite Impulse Response Filter, Inverse discrete cosine transform
9	Inverse Fast Fourier Transform (Auto/Indust. Vers)
10, 11	Matrix arithmetic, Pointer Chasing
12, 13	Pulse Width Modulation, Road Speed Calculation
14, 15	Table Lookup and Interpolation, Tooth To Spark
16, 17	OSPF/Dijkstra, Route Lookup/Patricia
18, 19, 20	Packet Flow – 512 kbytes, Packet Flow – 1 Mbyte, Packet Flow – 2 Mbytes
21, 22	Autocorrelation – Data1 (pulse), Autocorrelation – Data2 (sine)
23, 24	Autocorrelation – Data3 (speech), Convolutional Encoder – Data1 (xk5r2dt)
25	Convolutional Encoder – Data2 (xk4r2dt)
26	Convolutional Encoder – Data3 (xk3r2dt)
27	Fixed-point Bit Allocation – Data2 (typ)
28	Fixed-point Bit Allocation – Data3 (step)
29	Fixed Point Bit Allocation – Data6 (pent)
30	Fixed Point Complex FFT – Data1 (pulse)
31	Fixed point Complex FFT – Data2 (spn)
32	Fixed Point Complex FFT – Data3 (sine)
33, 34	Viterbi GSM Decoder – Data1 (get), Viterbi GSM Decoder – Data2 (toggle)
35, 36	Viterbi GSM Decoder – Data3 (ones), Viterbi GSM Decoder – Data4 (zeros)
37, 38, 39	Compress JPEG, Decompress JPEG, High Pass Grey-scale filter
40, 41, 42	RGB to CYMK Conversion, RGB to YIQ Conversion, Dithering
43, 44, 45	Image Rotation, Text Processing, src-sink

Liste de tâches par processeur

La Table A.3 présente, pour chacun des 17 processeurs, l'ensemble des différentes tâches que chacun est capable d'exécuter. Les processeurs et les tâches sont identifiés comme indiqué dans la Table A.2.

TABLE A.3 – Liste de tâches par processeur

ID processeur	Type de tâche
0	0, 1, ..., 45
1	16, 17, ..., 20, 45
2	0, 1, ..., 45
3	0, 1, ..., 45
4	0, 1, ..., 36, 42, 43, 44, 45
5	21, 22, ..., 36, 45
6	0, 1, ..., 45
7	0, 1, ..., 15, 21, 22, ..., 36, 42, 43, 44, 45
8	16, 17, ..., 26, 30, 31, ..., 36, 45
9	16, 17, ..., 36, 45
10	16, 17, ..., 26, 30, 31, ..., 36, 45
11	16, 17, ..., 36, 45
12	42, 43, 44, 45
13	0, 1, ..., 15, 45
14	0, 1, ..., 45
15	16, 17, ..., 45
16	21, 22, ..., 36, 45

Listes de processeurs par tâche

La Table A.4 montre, pour chacun des 46 différents types de tâches utilisées, le nombre de processeurs distincts pouvant être utilisés pour les exécuter, ainsi que l'ensemble des identifiants de ces processeurs. Le couple (processeur, tâche) identifie une IP, dont le numéro est également montré.

TABLE A.4 – Listes de processeurs par tâche utilisées dans la bibliothèque

Type de la tâche	Nombre de processeur	ID processeur
0/1/ .../15	8	0, 2, 3, 4, 6, 7, 13, 14
16 /17/ .../20	12	0, 1, 2, 3, 4, 6, 8, 9, 10, 11, 14, 15
21/22/ .../26	14	0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 14, 15, 16
27/28/29	12	0, 2, 3, 4, 5, 6, 7, 9, 11, 14, 15, 16
30 /31/ .../36	14	0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 14, 15, 16
37/38/ .../41	6	0, 2, 3, 6, 14, 15
42/43/44	9	0, 2, 3, 4, 6, 7, 12, 14, 15
45	17	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16

ANNEXE B - Graphes de tâches de référence

Cette annexe classe les graphes de tests utilisés comme références pour évaluer le mérite des travaux décrits dans cette thèse. Ces graphes sont répertoriés en trois classes: Des applications réelles, des graphes générés aléatoirement et des patterns de test.

Graphes de tâches de E3S

Il existe 5 ensembles d'applications de différents domaines. Chaque application est accompagnée d'une description générale, suivie du graphique des tâches correspondant. Les données ont été extraites de la page située à l'adresse <http://www.eembc.org/>.

Applications de l'industrie automobile

Il existe cinq applications utilisées dans l'industrie automobile, généralement mises en œuvre dans les systèmes embarqués. Ces applications font usage de microprocesseurs et de microcontrôleurs pour exécuter les fonctionnalités requises.

Auto-indust-tg0

Cette application permet d'effectuer des tests de charge, incluant la manipulation de bits, la manipulation de matrices, les opérations en virgule flottante, le débordement de cache, et la modulation d'impulsions en largeur (PWM).

Auto-industrie-tg1

Cette application spécifie les algorithmes automobiles de base, tels que l'algorithme de synchronisation du moteur. moteur au démarrage (dent à étincelle), conversion angle en heure, calcul de vitesse et interpolation.



FIGURE B.1 – Graphe de tâches du benchmark Auto-industriel 0

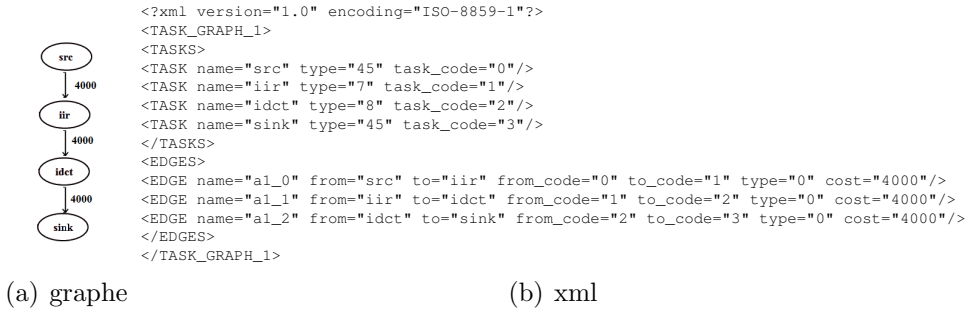


FIGURE B.2 – Graphe de tâches du benchmark Auto-industriel 1

Auto-indust-tg2 et auto-indust-tg3

Ces deux applications spécifient des algorithmes de traitement du signal et des algorithmes utilisés pour calculer stabilité du véhicule basée sur les signaux des capteurs. Inclure FFT (Fast Fourier Transform), IFFT (Transformation de Fourier rapide inverse), FIR (Finite Impulse Response Filter) et IDCT (Transformation inverse en cosinus discret).



FIGURE B.3 – Graphe de tâches du benchmark Auto-industriel 2

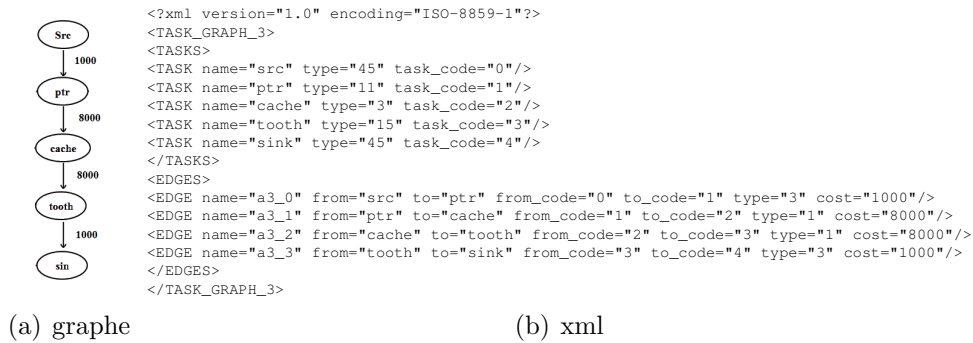


FIGURE B.4 – Graphe de tâches du benchmark Auto-industriel 3

Applications grand public

Il existe deux applications utilisées dans les appareils photo numériques, les imprimantes et autres systèmes embarqués qui manipulent les images numériques.

Consommateur-tg0

Cette application est dédiée aux algorithmes utilisés pour la compression et la décompression d'images, tels que comme algorithme de compression et de décompression d'image JPEG.



FIGURE B.5 – Graphe de tâches du benchmark Consommateur 0

Consommateur-tg1

Cette application est dédiée aux algorithmes de filtrage et de conversion des couleurs, tels que le filtrage passe-haut dans Conversion des niveaux de gris et RVB vers CMJN et conversion RVB vers YIQ.

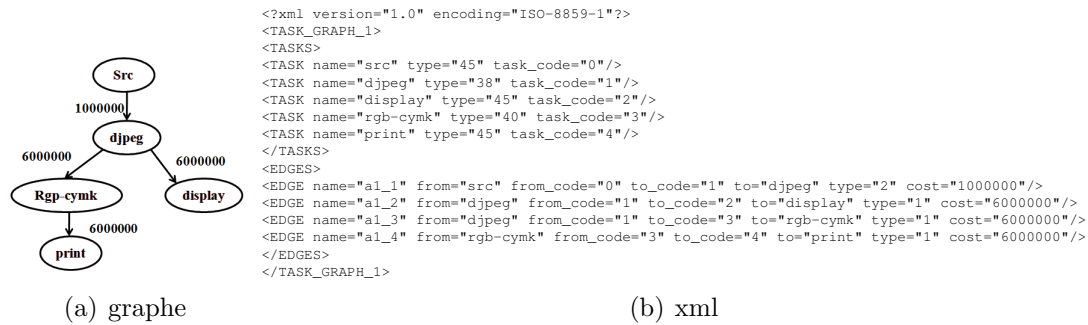


FIGURE B.6 – Graphe de tâches du benchmark Consommateur 1

Applications réseau

Il existe trois applications qui permettent de simuler le transport de paquets dans un réseau de communication.

Networking-tg1

Cette application est utilisée pour vérifier les paquets IP en traitant l'en-tête du paquet. Paquets IP lors de l'envoi et de la réception par les routeurs.

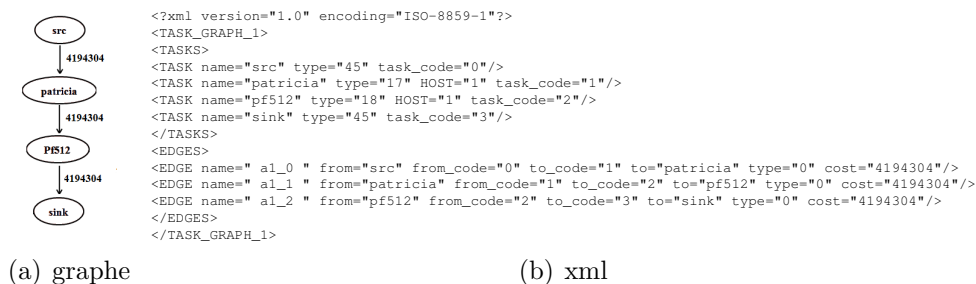


FIGURE B.7 – Graphe de tâches du benchmark Networking 1

Networking-tg2

Cette application modélise la traduction d'adresses réseau IP (NAT-Network Address Translator), qui consiste à réécrire l'adresse IP et le port configurés dans les paquets selon la configuration à partir de la table NAT du routeur.

Networking-tg3

Utilisée pour rechercher des itinéraires. Effectue le traitement d'une structure de données connu sous le nom d'arbre Patrícia, un arbre binaire compact qui permet une analyse rapide et efficace sur les longues chaînes de bits.

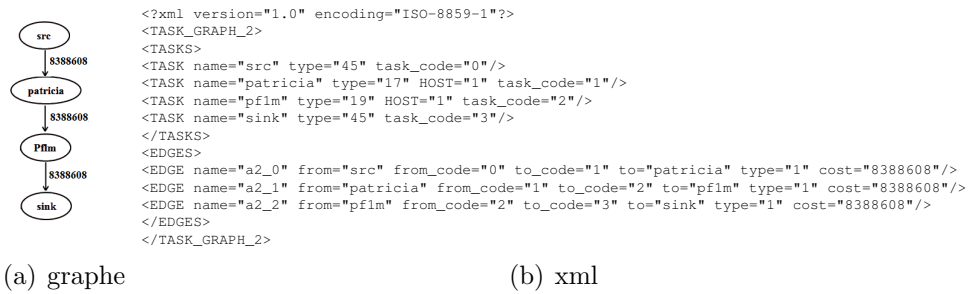


FIGURE B.8 – Graphe de tâches du benchmark Networking 2

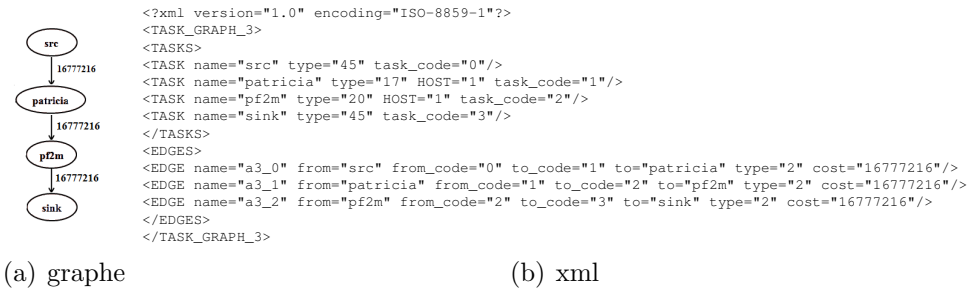


FIGURE B.9 – Graphe de tâches du benchmark Networking 3

Application d'automatisation

Applications exécutées sur des imprimantes, des traceurs et d'autres systèmes bureautiques qui inclure des tâches de traitement d'images et de texte.

Bureautique-tg0

Cette application implémente le traitement de la courbe de Bézier grâce à l'interpolation d'un ensemble de points.

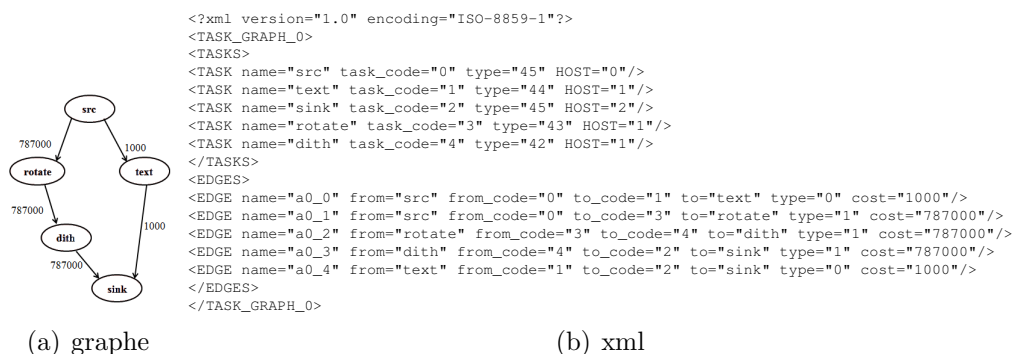


FIGURE B.10 – Graphe de tâches du benchmark Bureautique 0

Applications de télécommunications

Il existe six applications exécutées par des modems et autres appareils embarqués utilisés dans le domaine de télécommunication. Nous utilisons que quatre applications dans notre tests.

Télécom-tg0

Cette application spécifie l'autocorrélation et établit des relations croisées entre un signal et lui-même. Ce mécanisme est utilisé dans les télécommunications pour analyser les signaux.

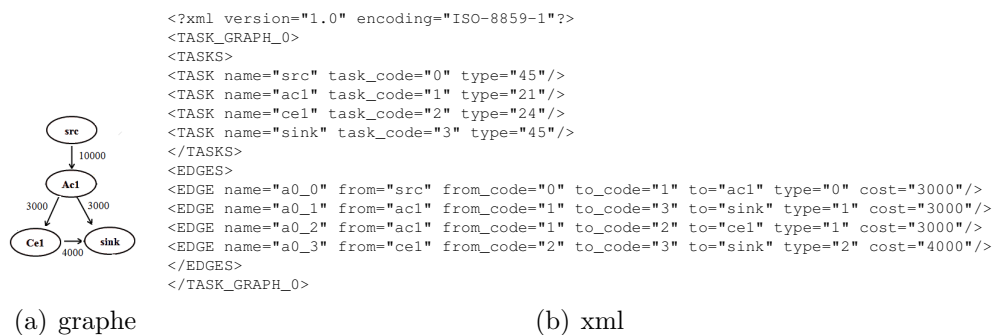


FIGURE B.11 – Graphe de tâches du benchmark Télécom 0

Télécom-tg1

Cette application implémente un codeur à convolution et permet la correction d'erreurs. c'est utilisé pour améliorer la qualité des radios numériques, des téléphones portables, des communications et des connexions par satellite Bluetooth.



FIGURE B.12 – Graphe de tâches du benchmark Télécom 1

Télécom-tg2

Permet de vérifier la capacité de transmission de données sur les lignes ADSL.



FIGURE B.13 – Graphe de tâches du benchmark Télécom 2

Télécom-tg3

Permet la conversion des données du domaine temporel vers le domaine fréquentiel, en utilisant la FFT (Transformation Rapide de Fourier).

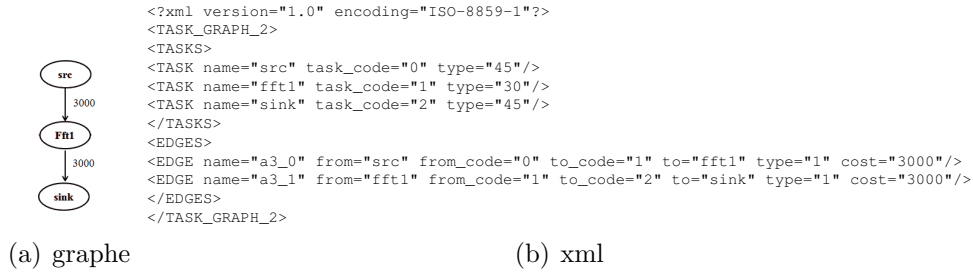


FIGURE B.14 – Graphe de tâches du benchmark Télécom 3

Graphes de tâches par TGFF

Cinq graphes de tâches aléatoires sont générés par TGFF pour des expérimentations. Cela permet une évaluation plus robuste des performances, en tenant compte de la variabilité inhérente aux tâches générées aléatoirement. Les caractéristiques de données présentées dans l'Annexe A sont utilisé pour former ces graphes.

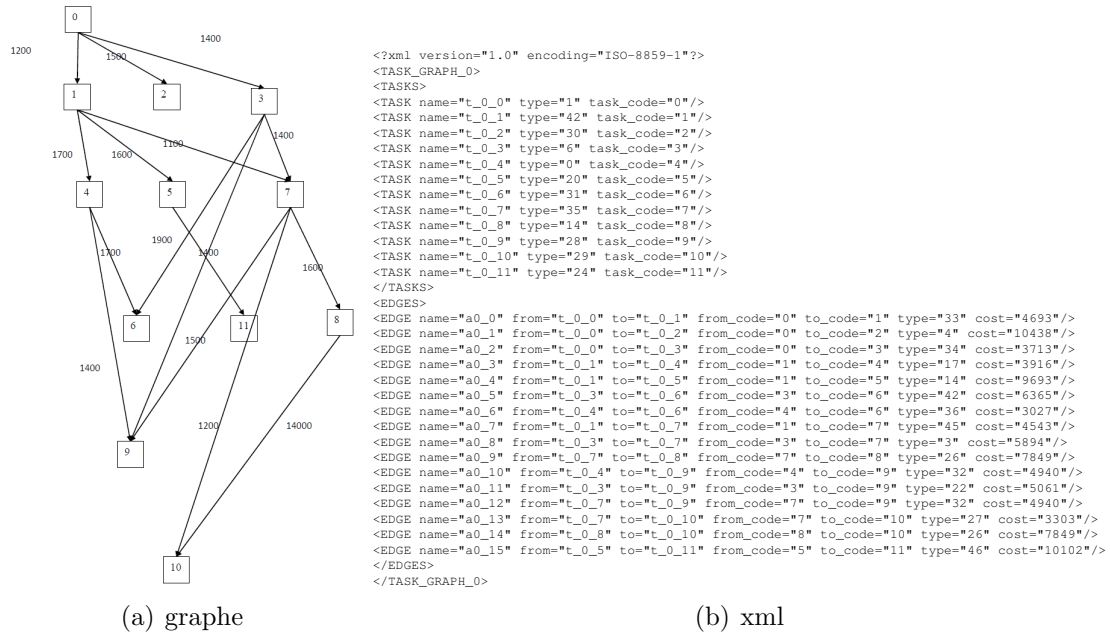


FIGURE B.15 – Graphe de tâches du TG0

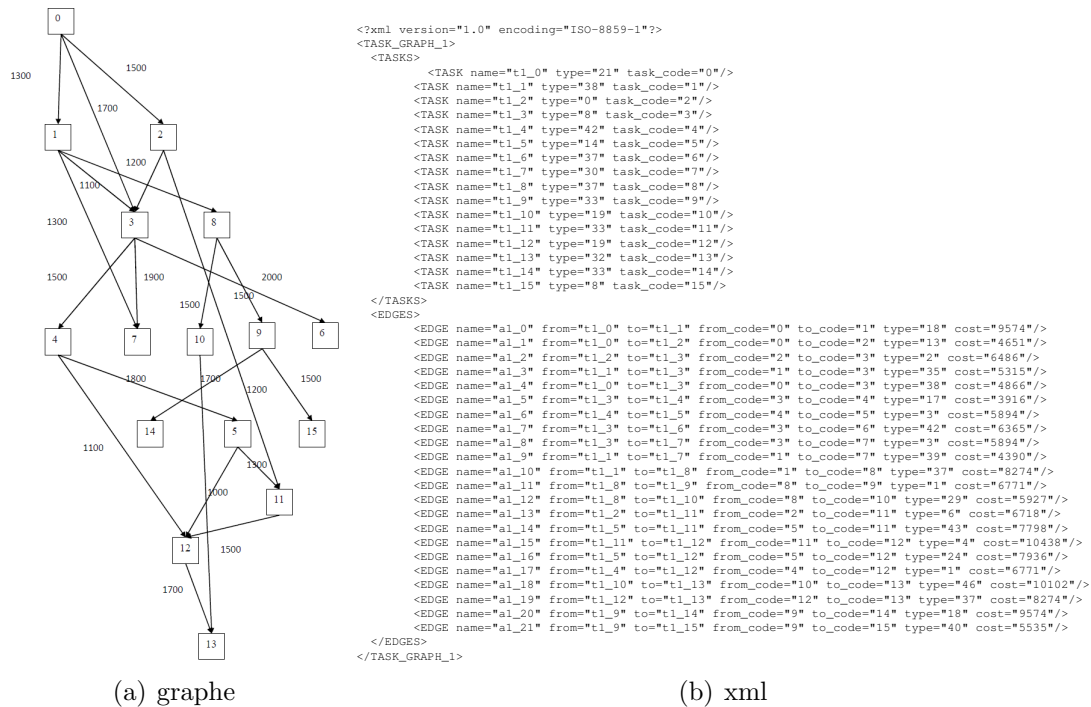


FIGURE B.16 – Graphe de tâches du TG1

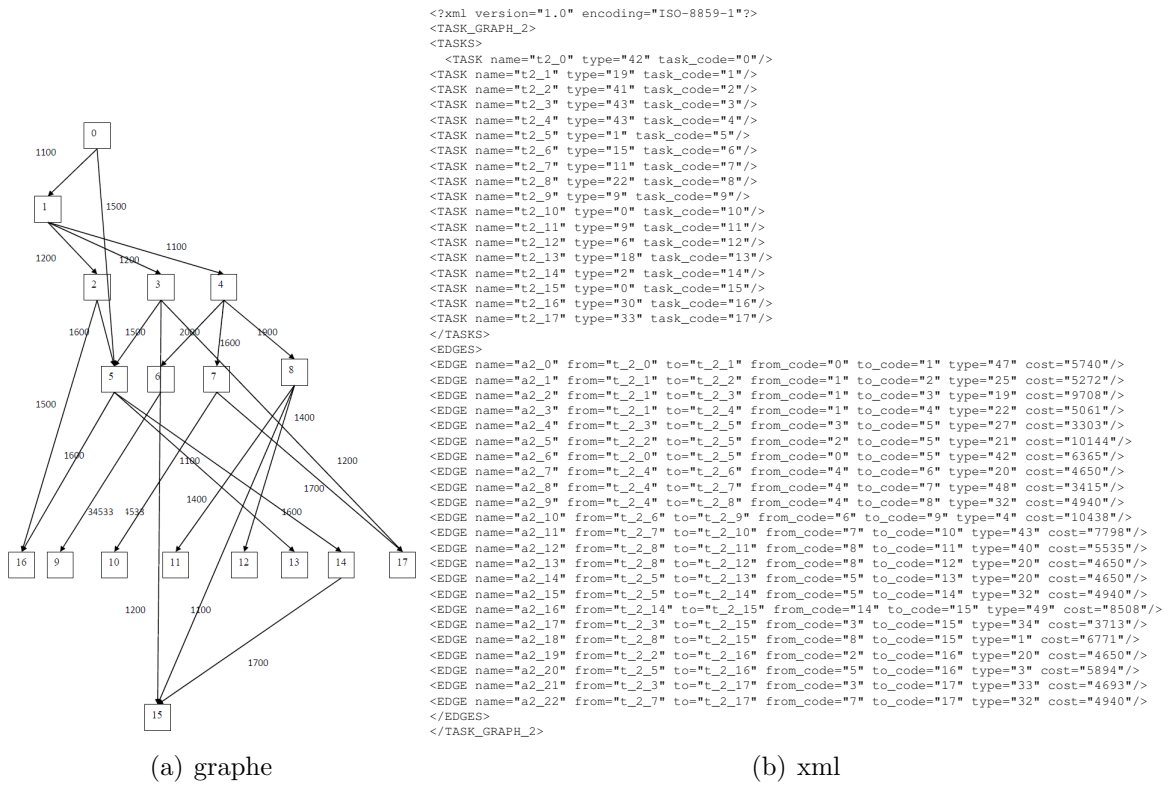


FIGURE B.17 – Graphe de tâches du TG2

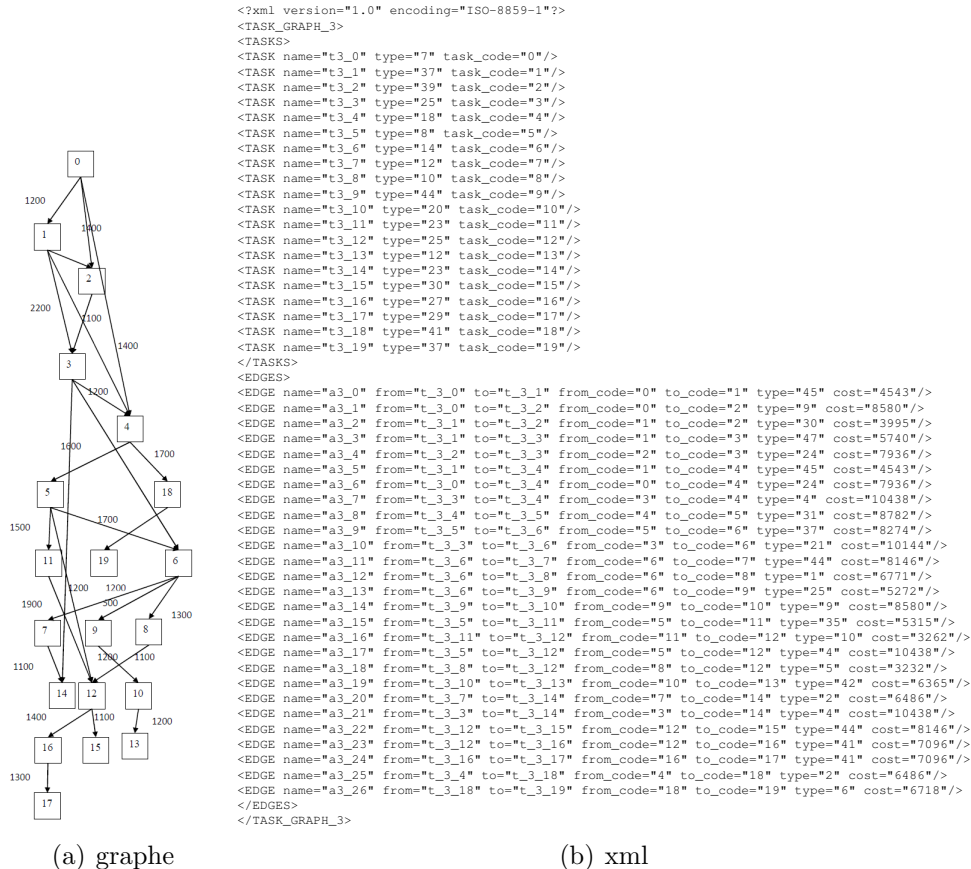


FIGURE B.18 – Graphe de tâches du TG3

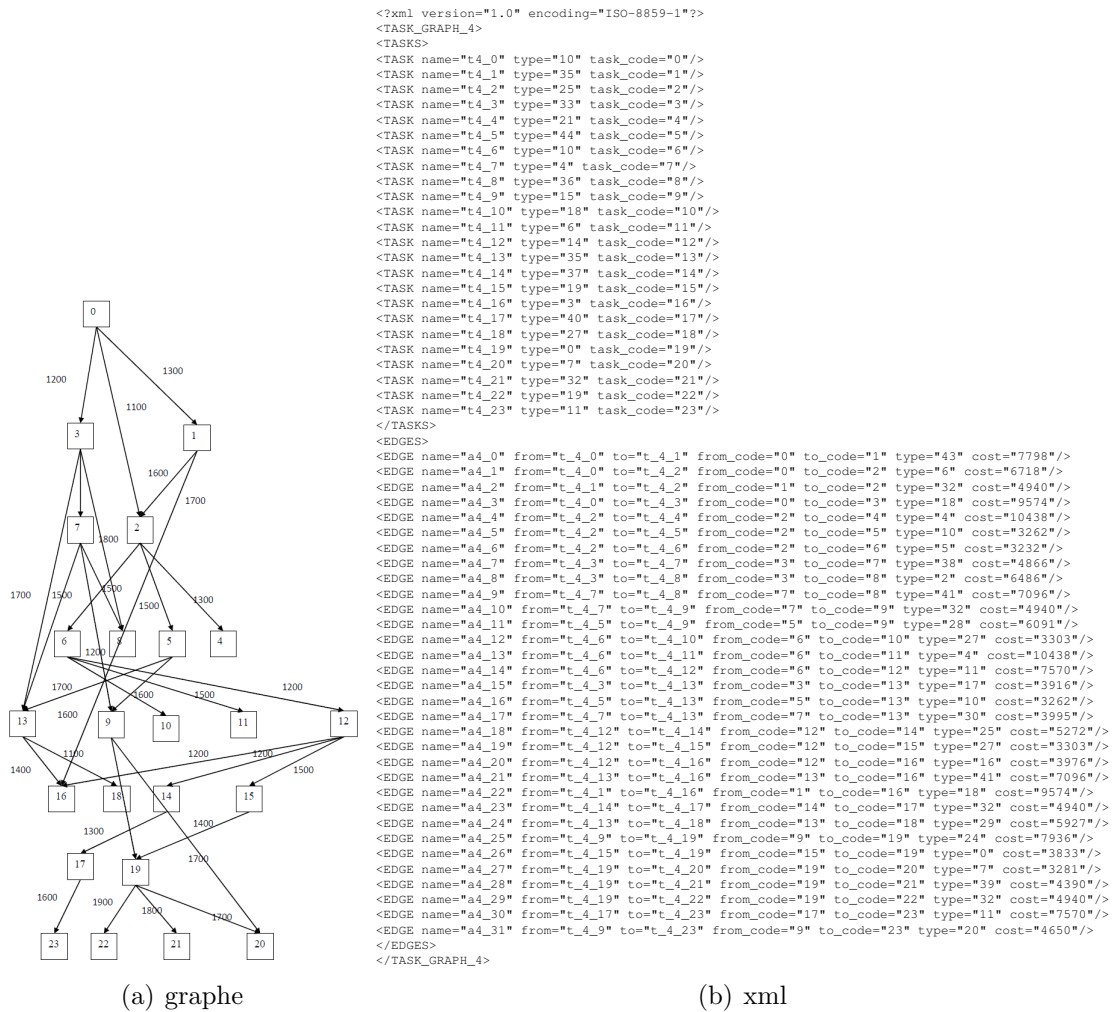


FIGURE B.19 – Graphe de tâches du TG4

Patterne déterministes

Trois modèles de distribution spécifiques sont utilisés qui sont: Le complément, la matrice transposée 1 et la matrice transposée 2. Explorons chaque modèle en détail pour un réseau sur puce en 2D dans la Table B.1 et 3D dans la Table B.2.

TABLE B.1 – Schéma de routage déterministes en 2D

Source		Destination					
		Transpose 1		Transpose 2		Complément	
0	(0, 0)	24	(4, 4)	0	(0, 0)	24	(4, 4)
1	(0, 1)	19	(3, 4)	5	(1, 0)	23	(4, 3)
2	(0, 2)	14	(2, 4)	10	(2, 0)	22	(4, 2)
3	(0, 3)	9	(1, 4)	15	(3, 0)	21	(4, 1)
4	(0, 4)	4	(0, 4)	20	(4, 0)	20	(4, 0)
5	(1, 0)	23	(4, 3)	1	(0, 1)	19	(3, 4)
6	(1, 1)	18	(3, 3)	6	(1, 1)	18	(3, 3)
7	(1, 2)	13	(2, 3)	11	(2, 1)	17	(3, 2)
8	(1, 3)	8	(1, 3)	16	(3, 1)	16	(3, 1)
9	(1, 4)	3	(0, 3)	21	(4, 1)	15	(3, 0)
10	(2, 0)	22	(4, 2)	2	(0, 2)	14	(2, 4)
11	(2, 1)	17	(3, 2)	7	(1, 2)	13	(2, 3)
12	(2, 2)	12	(2, 2)	12	(2, 2)	12	(2, 2)
13	(2, 3)	7	(1, 2)	17	(3, 2)	11	(2, 1)
14	(2, 4)	2	(0, 2)	22	(4, 2)	10	(2, 0)
15	(3, 0)	21	(4, 1)	3	(0, 3)	9	(1, 4)
16	(3, 1)	16	(3, 1)	8	(1, 3)	8	(1, 3)
17	(3, 2)	11	(2, 1)	13	(2, 3)	7	(1, 2)
18	(3, 3)	6	(1, 1)	18	(3, 3)	6	(1, 1)
19	(3, 4)	1	(0, 1)	23	(4, 3)	5	(1, 0)
20	(4, 0)	20	(4, 0)	4	(0, 4)	4	(0, 4)
21	(4, 1)	15	(3, 0)	9	(1, 4)	3	(0, 3)
22	(4, 2)	10	(2, 0)	14	(2, 4)	2	(0, 2)
23	(4, 3)	5	(1, 0)	19	(3, 4)	1	(0, 1)
24	(4, 4)	0	(0, 0)	24	(4, 4)	0	(0, 0)

TABLE B.2 – Schéma de routage déterministes en 3D

Source		Destination					
		Transpose 1		Transpose 2		Complément	
0	(0, 0, 0)	26	(2, 2, 2)	0	(0, 0, 0)	26	(2, 2, 2)
1	(0, 1, 0)	23	(1, 2, 2)	3	(1, 0, 0)	25	(2, 1, 2)
2	(0, 2, 0)	20	(0, 2, 2)	6	(2, 0, 0)	24	(2, 0, 2)
3	(1, 0, 0)	17	(2, 2, 1)	9	(0, 0, 1)	23	(1, 2, 2)
4	(1, 1, 0)	14	(1, 2, 1)	12	(1, 0, 1)	22	(1, 1, 2)
5	(1, 2, 0)	11	(0, 2, 1)	15	(2, 0, 1)	21	(1, 0, 2)
6	(2, 0, 0)	8	(2, 2, 0)	18	(0, 0, 2)	20	(0, 2, 2)
7	(2, 1, 0)	5	(1, 2, 0)	21	(1, 0, 2)	19	(0, 1, 2)
8	(2, 2, 0)	2	(0, 2, 0)	24	(2, 0, 2)	18	(0, 0, 2)
9	(0, 0, 1)	25	(2, 1, 2)	1	(0, 1, 0)	17	(2, 2, 1)
10	(0, 1, 1)	22	(1, 1, 2)	4	(1, 1, 0)	16	(2, 1, 1)
11	(0, 2, 1)	19	(0, 1, 2)	7	(2, 1, 0)	15	(2, 0, 1)
12	(1, 0, 1)	16	(2, 1, 1)	10	(0, 1, 1)	14	(1, 2, 1)
13	(1, 1, 1)	13	(1, 1, 1)	13	(1, 1, 1)	13	(1, 1, 1)
14	(1, 2, 1)	10	(0, 1, 1)	16	(2, 1, 1)	12	(1, 0, 1)
15	(2, 0, 1)	7	(2, 1, 0)	19	(0, 1, 2)	11	(0, 2, 1)
16	(2, 1, 1)	4	(1, 1, 0)	22	(1, 1, 2)	10	(0, 1, 1)
17	(2, 2, 1)	1	(0, 1, 0)	25	(2, 1, 2)	9	(0, 0, 1)
18	(0, 0, 2)	24	(2, 0, 2)	2	(0, 2, 0)	8	(2, 2, 0)
19	(0, 1, 2)	21	(1, 0, 2)	5	(1, 2, 0)	7	(2, 1, 0)
20	(0, 2, 2)	18	(0, 0, 2)	8	(2, 2, 0)	6	(2, 0, 0)
21	(1, 0, 2)	15	(2, 0, 1)	11	(0, 2, 1)	5	(1, 2, 0)
22	(1, 1, 2)	12	(1, 0, 1)	14	(1, 2, 1)	4	(1, 1, 0)
23	(1, 2, 2)	9	(0, 0, 1)	17	(2, 2, 1)	3	(1, 0, 0)
24	(2, 0, 2)	6	(2, 0, 0)	20	(0, 2, 2)	2	(0, 2, 0)
25	(2, 1, 2)	3	(1, 0, 0)	23	(1, 2, 2)	1	(0, 1, 0)
26	(2, 2, 2)	0	(0, 0, 0)	26	(2, 2, 2)	0	(0, 0, 0)

ANNEXE C - Résultats de Testes

Cette annexe offre un aperçu détaillé des résultats issus des simulations effectuées au cours du chapitre 7 pour chacune des phases réalisées.

Résultat de la phase d'affectation

Les Tables C.1, C.2 et C.3 illustrent les résultats de la phase d'affectation concernant la méthode MOPSP pour les graphes de tâches non linéaires. En outre, la Table C.4 présente les résultats obtenus pour les benchmarks de test dans cette phase.

TABLE C.1 – Affectation des IPs avec un ordonnancement ASAP utilisant MOPSO

Benchmark ID	solutions	IPs			Meilleur Affectation			Affectation Moyenne		
		Minimum	Maximum	Moyenne	Puissance	Temps	Surface	Puissance	Temps	Surface
3	7	3	5	3.28	8.95	0.832	8.18	11.95	2.49	23.17
6	7	2	4	2.38	0.375	14.52	2.44	3.321	39.148	7.6
10	27	2	4	2.88	0.375	4.92	1	8.461	303.097	25.2
12	17	3	5	4.1	0.45	0.2097	1	4.54	1.958	7.07
13	10	3	5	3.9	0.45	0.213	1	3.955	1.994	10.9
15	48	6	10	7.958	14.900	45.487	25.225	39.693	218.668	90.719
16	66	6	11	8.030	21.635	6.293	37.227	76.847	75.659	161.786
17	46	5	9	6.695	9.945	3.851	17.608	21.371	307.680	67.526
18	35	5	12	8.514	20.834	39.519	42.409	57.255	114.651	131.094
19	56	7	12	9.642	33.270	20.987	47.402	89.150	35.263	160.879
20	61	8	13	11.131	52.950	21.700	79.217	119.204	126.586	205.847
21	66	11	16	13.530	165.154	102.573	168.823	250.260	426.509	293.645
22	73	9	14	11.863	76.434	37.567	127.385	153.425	101.906	228.126
23	62	11	15	13.500	112.680	32.608	133.376	211.301	237.645	262.288
24	81	10	16	13.061	109.915	31.476	130.383	184.090	182.502	255.610
25	72	11	17	14.458	195.425	64.120	200.440	301.793	599.501	307.694
26	87	11	16	14.080	233.800	75.172	202.440	383.553	524.112	297.489
27	105	12	17	14.971	264.220	65.089	205.433	385.751	523.919	326.065
28	63	12	16	13.650	170.375	86.407	172.816	283.915	205.927	286.154
29	88	13	17	14.659	367.540	203.249	231.416	522.732	992.675	322.950

TABLE C.2 – Affectation des IPs avec un ordonnancement ALAP utilisant MOPSO

Benchmark ID	solutions	IPs			Meilleur Affectation			Affectation Moyenne		
		Minimum	Maximum	Moyenne	Puissance	Temps	Surface	Puissance	Temps	Surface
3	7	3	5	3.85	8.95	0.832	8.18	11.95	2.49	23.17
6	6	2	4	2.83	0.375	14.52	2.44	4.17	28.595	8.40
10	31	2	4	3.03	0.375	4.22	1	4.72	401.60	14.49
12	13	3	5	3.92	0.45	0.213	1	4.279	1.778	9.83
13	10	3	5	3.9	0.45	0.213	1	4.325	1.81	9.83
15	58	5	10	7.603	13.395	49.328	24.778	38.511	272.505	98.464
16	43	4	10	7.906	19.805	7.111	26.225	58.863	32.675	135.003
17	60	4	9	6.833	9.560	3.850	14.615	29.883	114.792	82.384
18	48	6	12	8.979	23.850	38.296	31.211	65.588	123.211	160.048
19	60	7	13	9.433	32.480	20.480	47.402	76.408	90.305	154.729
20	54	9	14	11.407	47.830	20.771	119.452	116.731	51.555	227.714
21	90	10	16	13.211	168.125	108.225	168.823	272.773	579.919	263.743
22	92	9	15	11.793	77.910	40.446	125.397	142.692	157.721	232.906
23	45	11	16	13.511	99.839	34.895	163.442	184.094	267.171	257.295
24	81	10	15	12.913	95.394	27.483	130.383	181.478	145.844	257.686
25	63	11	16	13.968	201.414	69.903	198.447	319.903	562.617	293.007
26	81	12	16	14.259	229.490	73.343	202.440	370.410	409.856	305.751
27	95	12	17	14.694	270.334	55.358	201.440	415.590	602.240	322.112
28	97	11	16	13.587	160.054	65.121	171.816	271.611	229.562	286.744
29	96	13	17	14.625	371.829	251.852	240.880	537.986	1225.420	321.340

TABLE C.3 – Affectation des IPs avec un ordonnancement LS utilisant MOPSO

Benchmark ID	solutions	IPs			Meilleur Affectation			Affectation Moyenne		
		Minimum	Maximum	Moyenne	Puissance	Temps	Surface	Puissance	Temps	Surface
3	8	2	4	3.25	7.15	0.8325	1	8.208	0.8332	7.3
6	10	1	3	2.3	0.375	14.52	2.44	3.321	39.148	7.6
10	33	1	4	2.36	0.375	4.92	1	8.62	294.36	25.6
12	32	2	4	2.62	0.45	0.2097	1	3.937	1.73	7.70
13	25	2	4	2.96	0.45	0.2097	1	3.955	1.509	6.84
15	45	5	10	7.800	13.295	48.817	23.225	34.898	227.324	102.244
16	65	5	12	7.969	18.700	7.158	41.416	66.468	169.594	142.083
17	40	4	9	6.750	9.960	3.864	15.615	26.299	262.966	81.556
18	23	5	11	8.304	20.660	37.511	26.218	53.533	234.381	141.498
19	74	6	13	9.756	33.849	20.452	46.402	73.081	85.487	154.050
20	61	9	15	11.409	53.490	17.406	88.392	117.098	129.282	197.227
21	147	11	16	13.510	203.770	169.213	203.433	277.329	468.424	291.404
22	95	8	15	11.694	79.074	35.457	124.397	155.747	295.997	217.671
23	53	11	15	13.396	98.470	31.650	171.816	204.490	127.280	273.331
24	57	10	16	12.947	102.904	27.768	130.383	194.268	232.321	267.034
25	81	12	17	14.419	198.625	61.669	201.440	294.142	414.082	305.074
26	73	13	17	14.493	234.369	50.772	230.416	336.325	305.614	317.377
27	82	12	17	14.597	266.934	54.453	203.433	423.629	608.325	307.541
28	90	10	16	13.511	158.634	62.396	171.816	287.529	188.539	280.535
29	100	12	17	14.740	356.285	151.580	242.873	515.390	653.247	328.452

TABLE C.4 – Resultats d’affectation des IPs avec MOPSO pour les benchmarks E3S

Benchmark ID	solutions	IPs			Meilleur Affectation			Affectation Moyenne		
		Minimum	Maximum	Moyenne	Puissance	Temps	Surface	Puissance	Temps	Surface
1	3	1	2	1.6	4.15	0.0221	1.0	4.83	0.0221	2.14
2	3	1	2	1.6	2.15	0.0475	1.0	2.83	0.0475	2.14
3	8	2	4	3.25	8.34	0.832	8.18	17.08	0.852	42.84
4	3	1	2	1.6	3.15	0.0395	1.0	3.83	0.0395	2.14
5	12	1	3	2.3	0.525	22.12	2.44	5.34	54.63	9.48
6	10	1	3	2.3	0.375	14.52	2.44	3.395	44.973	8.39
7	14	1	3	2	0.3	0.282	2.44	2.48	1.459	17.90
8	14	1	3	2	0.3	0.311	2.44	2.48	1.88	17.9
9	15	1	3	2	0.3	0.39	2.44	2.38	2.794	19.43
10	33	1	4	2.36	0.375	4.22	1.0	4.72	401.6	14.49
11	8	1	3	1.75	0.3	0.0245	1.0	3.008	0.347	3.008
12	32	2	4	2.625	1.145	0.213	2	4.164	2.16	10.65
13	25	2	4	2.96	1.049	0.209	2	3.88	1.974	10.97
14	6	1	2	1.5	0.224	0.029	1.0	2.11	0.277	1.98

Dans cette partie, les Tables C.5, C.6, C.7, C.8 et C.9 illustrent en détail les résultats de la phase d’affectation selon les cinq stratégies de mutation utilisées.

TABLE C.5 – Résultats d’affectation des IPs avec DEMO rand_1 pour les benchmarks E3S

Benchmark ID	solutions	IPs			Meilleur Affectation			Affectation Moyenne		
		Minimum	Maximum	Moyenne	Puissance	Temps	Surface	Puissance	Temps	Surface
1	3	2	2	2	4.15	0.0221	1.0	4.83	0.0222	2.14
2	3	2	2	2	2.15	0.0475	1.0	2.83	0.0474	2.14
3	8	2	3	2.62	7.15	0.832	1.0	8.208	0.833	7.38
4	3	2	2	2	3.15	0.0395	1.0	3.83	0.0395	2.14
5	16	2	3	2.87	1.745	22.12	6.432	6.147	38.657	9.33
6	12	2	4	3	3.44	0.47	14.52	4.009	21.44	9.0072
7	13	2	3	2.46	0.3	0.282	2.44	2.4	1.44	2.408
8	14	2	3	2.53	0.3	0.311	2.44	2.15	1.908	19.049
9	14	2	3	2.57	0.3	0.39	2.44	2.067	2.835	20.6
10	24	2	4	2.87	0.565	4.22	1.0	5.664	236.90	16.309
11	5	2	3	2.4	0.3	0.0245	2.0	2.373	0.0593	2.864
12	7	3	4	3.42	0.45	0.209	2.0	3.554	0.646	8.377
13	6	3	5	3.83	0.45	0.209	2.44	3.54	0.515	3.2
14	5	2	2	2	0.224	0.029	1.0	1.993	0.1894	2.176

TABLE C.6 – Résultats d’affectation des IPs avec DEMO best_1 pour les benchmarks E3S

Benchmark ID	solutions	IPs			Meilleur Affectation			Affectation Moyenne		
		Minimum	Maximum	Moyenne	Puissance	Temps	Surface	Puissance	Temps	Surface
1	3	2	2	2	4.15	0.0221	1.0	4.83	0.022	2.146
2	3	2	2	2	2.15	0.0475	1.0	2.83	0.0475	2.146
3	8	2	3	2.625	7.15	0.0832	1.0	8.203	0.0833	7.386
4	3	2	2	2	3.15	0.0395	1.0	3.83	0.0395	2.146
5	14	2	3	2.85	2.45	22.12	8.18	6.58	32.53	9.51
6	6	2	4	2.83	0.470	14.2	3.44	2.7	36.77	8.518
7	13	2	3	2.46	0.3	0.282	2.44	2.408	1.442	18.82
8	13	2	3	2.53	0.3	0.311	2.44	2.150	1.908	19.049
9	13	2	3	2.53	0.3	0.39	2.44	1.96	2.78	21.73
10	27	2	4	3.03	0.565	4.92	2	9.562	145.67	29.79
11	6	2	3	2.33	0.3	0.0245	1.0	3.044	0.053	2.553
12	9	3	4	3.55	0.45	0.213	2	4.18	0.67	3.071
13	7	3	4	3.57	0.45	0.209	2.44	2.364	0.86	8.93
14	5	2	2	2	0.224	0.029	1.0	1.993	0.1894	2.176

TABLE C.7 – Résultats d'affectation des IPs avec DEMO current to best pour les benchmarks E3S

Benchmark ID	solutions	IPs			Meilleur Affectation			Affectation Moyenne		
		Minimum	Maximum	Moyenne	Puissance	Temps	Surface	Puissance	Temps	Surface
1	3	2	2	2	4.15	0.022	1.0	4.83	0.022	2.146
2	3	2	2	2	2.15	0.0475	1.0	2.83	0.0475	2.146
3	7	3	5	3.28	8.76	0.832	8.18	10.66	1.035	12.022
4	3	2	2	2	3.15	0.0395	1.0	3.83	0.0395	2.146
5	12	2	4	3.083	1.455	22.12	4.439	6.839	44.361	9.449
6	7	2	4	2.85	0.47	14.52	3.44	2.38	42.66	7.79
7	13	2	3	2.53	0.3	0.282	2.44	2.15	1.467	19.04
8	14	2	4	2.64	0.3	0.311	2.44	2.074	1.96	20.68
9	12	2	3	2.5	0.3	0.39	2.44	2.085	2.785	19.68
10	27	2	4	2.88	0.565	4.22	1.0	4.934	313.84	15.77
11	6	2	3	2.33	0.3	0.0245	1.0	2.584	0.254	2.55
12	17	3	5	4.11	0.545	0.209	2	4.869	0.714	9.052
13	10	3	5	3.9	0.545	0.219	2	4.981	0.662	4.98
14	6	2	2	3	0.225	0.029	1.0	2.115	0.277	1.98

TABLE C.8 – Résultats d'affectation des IPs avec DEMO rand.2 pour les benchmarks E3S

Benchmark ID	solutions	IPs			Meilleur Affectation			Affectation Moyenne		
		Minimum	Maximum	Moyenne	Puissance	Temps	Surface	Puissance	Temps	Surface
1	3	2	2	2	4.15	0.0221	1.0	4.83	0.0221	2.146
2	3	2	2	2	2.15	0.0475	1.0	2.83	0.0475	2.146
3	7	3	5	3.85	9.045	0.832	8.18	9.969	0.835	13.3
4	3	2	2	2	3.15	0.0395	1.0	3.83	0.0395	2.146
5	13	2	4	2.92	2.45	22.12	8.18	7.185	37.18	9.52
6	6	2	4	2.83	0.47	14.52	3.44	2.70	36.77	8.518
7	13	2	3	2.53	0.3	0.282	2.44	2.15	1.467	19.05
8	13	2	3	2.57	0.3	0.311	2.44	2.15	1.908	19.05
9	14	2	3	3.03	0.3	0.39	2.44	2.067	2.635	20.6
10	31	2	4	2.33	0.565	4.92	1.0	9.861	184.9	28.55
11	6	2	3	3.92	0.3	0.024	1.0	3.04	0.0535	2.55
12	13	3	5	3.92	1.84	0.021	3	4.90	0.413	10.06
13	10	3	5	3.9	1.049	0.209	2	3.6	0.938	3.7
14	4	2	2	2	0.224	0.029	2	1.29	0.229	2.47

TABLE C.9 – Résultats d'affectation des IPs avec DEMO best.2 les benchmarks E3S

Benchmark ID	solutions	IPs			Meilleur Affectation			Affectation Moyenne		
		Minimum	Maximum	Moyenne	Puissance	Temps	Surface	Puissance	Temps	Surface
1	3	2	2	2	4.15	0.0221	1.0	4.83	0.0221	2.14
2	3	2	2	2	2.15	0.0475	1.0	2.83	0.0475	2.14
3	8	2	3	2.625	7.15	0.832	1.0	8.20	0.833	7.38
4	3	2	2	2	3.15	0.0395	1.0	3.83	0.0395	2.14
5	16	2	3	2.81	1.03	22.12	3.44	5.79	43.95	8.88
6	14	2	4	2.92	0.375	14.52	2.44	3.62	29.06	8.58
7	13	2	3	2.53	0.3	0.282	2.44	2.15	1.467	19.05
8	13	2	3	2.53	0.3	0.311	2.44	2.15	1.908	19.05
9	14	2	3	2.57	0.3	0.39	2.44	2.067	2.835	20.6
10	28	2	5	3.03	0.565	4.22	1.0	4.42	287.78	14.13
11	6	2	3	2.33	0.3	0.0245	1.0	3.04	0.053	2.55
12	8	3	4	3.5	0.45	0.213	2	3.61	0.659	7.93
13	8	3	4	3.5	0.45	0.213	2	4.12	0.481	7.88
14	6	2	2	2	0.225	0.029	1.0	2.11	0.277	1.98

TABLE C.10 – Résultats d’affectation des IPs avec DEMO best_1 pour les benchmarks TGFF

Benchmark ID	solutions	IPs			Meilleur Affectation			Affectation Moyenne		
		Minimum	Maximum	Moyenne	Puissance	Temps	Surface	Puissance	Temps	Surface
15	80	5	8	6.662	6.385	45.197	5.440	17.876	102.135	41.726
16	95	4	8	6.263	10.655	6.171	5.440	20.304	59.363	41.301
17	83	4	8	5.904	5.200	3.764	5.440	10.088	97.743	29.416
18	66	5	9	7.030	10.460	37.210	10.622	18.829	53.617	33.333
19	65	7	9	7.969	13.275	19.514	10.426	19.950	30.903	30.809
20	24	7	12	9.917	25.440	21.057	18.161	32.440	32.985	44.997
21	32	11	14	12.594	104.285	90.753	89.393	144.436	138.463	203.637
22	19	9	12	10.526	38.505	32.737	27.218	50.256	50.134	75.720
23	9	11	14	12.111	63.150	31.923	51.396	85.568	68.151	142.681
24	17	11	13	11.706	57.390	28.134	46.403	75.246	88.899	127.070
25	15	12	14	13.133	112.725	60.611	159.450	178.601	132.559	240.285
26	12	12	14	13.167	169.350	58.487	225.866	215.098	112.970	273.344
27	21	12	14	13.238	198.875	48.028	172.816	262.596	243.334	288.626
28	16	11	14	12.813	84.205	57.704	92.386	136.599	65.277	214.558
29	17	13	14	13.706	278.280	161.442	230.416	373.929	349.585	295.775

TABLE C.11 – Exemple de Solutions non dominées trouvée par MOPSO pour les benchmarks E3S

Benchmark ID	Affectation non-dominée	P	T	S
auto-indust-tg2	[12, 13, 0, 13, 13, 13, 13, 5]	x		
	[8, 13, 13, 13, 0, 13, 13, 16]		x	
	[10, 13, 13, 13, 13, 13, 13, 6, 10]			x
consumer-tg1	[15, 15, 15, 15, 15]	x		
	[15, 15, 15, 6, 15]		x	
	[15, 6, 15, 6, 12]			x
Office-tg0	[15, 15, 15, 15, 12]	x		
	[15, 6, 15, 6, 6]		x	
	[12, 0, 12, 0, 0]			x
Telecom-tg1	[15, 15, 9, 15, 15, 15]	x		
	[13, 15, 16, 6, 16, 12]		x	
	[5, 5, 5, 6, 6, 6]			x
Telecom-tg2	[15, 15, 15, 15, 5, 12]	x		
	[16, 15, 16, 16, 16, 5]		x	
	[16, 16, 5, 16, 5, 16]			x

TABLE C.12 – Exemple de Solutions non dominées trouvée par DEMO best_1 pour les benchmarks E3S

Benchmark ID	Non-dominated assignment	P	T	A
3	[12, 13, 6, 0, 13, 13, 13, 13, 15]	x		
	[13, 13, 13, 13, 0, 13, 0, 13, 12]		x	
	[6, 13, 13, 6, 13, 13, 13, 13, 13]			x
6	[11, 15, 5, 15, 12]	x		
	[9, 6, 9, 15, 6]		x	
	[15, 6, 8, 6, 6]			x
10	[15, 15, 15, 15, 12]	x		
	[12, 0, 12, 12, 0]		x	
	[12, 12, 12, 12, 12]			x
12	[12, 15, 15, 15, 11, 12]	x		
	[13, 16, 5, 16, 16, 5]		x	
	[12, 5, 5, 5, 5, 5]			x
13	[12, 15, 15, 15, 15, 15]	x		
	[15, 11, 15, 11, 16, 12]		x	
	[5, 16, 16, 16, 16, 16]			x

Résultat de la phase de placement

Dans cette section, nous présentons les résultats de placement concernant les deux approches multiobjectif, MOPSO et DEMO, pour les benchmarks présentés précédemment. Pour l'approche DEMO, nous détaillons les deux méthodes de mutation utilisées: la mutation aléatoire (Rand) et la mutation basée sur le meilleur individu (Best).

TABLE C.13 – Résultats de placement des IPs avec MOPSO en 2D pour les benchmarks de TGFF

Benchmark ID	solutions	Ressources			Meilleur Placement			Placement Moyenne		
		Minimum	Maximum	Moyenne	Puissance	Temps	Surface	Puissance	Temps	Surface
15	20	12	16	14.100	829.461	52.110	108.389	1032.604	790.831	148.511
16	27	13	18	16.037	807.170	10.024	187.319	974.786	40.687	341.345
17	13	9	12	10.769	569.260	5.879	59.291	624.271	7.187	108.791
18	33	14	19	16.879	1255.819	42.748	149.440	1459.135	125.187	304.010
19	31	16	23	20.419	1235.975	25.427	222.436	1497.185	33.792	409.397
20	28	27	32	30.286	2411.104	28.462	405.995	2701.136	43.990	693.397
21	34	48	59	54.265	6366.338	128.077	1100.769	6941.743	421.738	1572.723
22	32	35	40	37.219	3275.829	37.624	647.845	3714.169	139.118	847.137
23	36	44	52	49.111	4585.412	50.026	839.366	5313.901	233.282	1128.485
24	16	38	46	42.563	4829.009	43.441	800.413	5229.450	173.418	933.708
25	25	67	73	70.760	9305.879	101.478	1641.220	10369.975	288.670	2016.511
26	31	75	85	79.484	10628.776	106.568	1875.442	11815.687	380.788	2370.546
27	22	79	91	84.318	11228.160	90.840	2129.237	12463.272	773.550	2506.934
28	35	60	66	63.914	7910.052	93.480	1341.592	8969.018	160.930	1692.286
29	24	96	113	102.417	15779.565	280.757	2917.607	17271.156	783.267	3209.812

TABLE C.14 – Résultats de placement des IPs avec DEMO rand_1 en 2D pour les benchmarks de TGFF

Benchmark ID	solutions	Ressources			Meilleur Placement			Placement Moyenne		
		Minimum	Maximum	Moyenne	Puissance	Temps	Surface	Puissance	Temps	Surface
15	19	7	10	8.263	450.087	46.935	33.820	494.721	71.428	42.075
16	12	7	13	9.833	457.524	7.077	42.380	511.413	350.530	50.948
17	3	6	6	6.000	218.741	4.738	15.190	242.802	4.811	19.187
18	11	7	11	9.091	566.280	39.572	39.320	660.741	48.264	57.050
19	8	9	12	10.250	630.178	21.822	49.570	678.019	34.250	58.772
20	19	19	26	21.737	1495.662	24.136	130.294	1603.750	39.989	167.555
21	12	39	47	42.667	4650.603	81.821	456.812	5092.773	118.660	602.652
22	11	24	32	29.091	2021.964	29.323	211.791	2331.791	33.645	274.652
23	15	32	46	39.533	3507.534	43.688	367.238	3829.262	120.431	462.897
24	13	30	40	35.385	3155.855	37.929	239.833	3579.894	41.045	376.443
25	21	44	66	57.762	6797.375	74.422	638.378	7271.439	245.521	864.235
26	24	43	69	59.083	7872.955	77.395	787.891	8604.454	122.936	1072.127
27	21	56	79	64.476	8171.287	65.027	957.475	8898.047	96.885	1155.800
28	20	51	62	56.200	5923.770	68.797	563.900	6245.054	132.769	739.256
29	19	60	80	71.789	11548.097	154.134	1374.777	12408.670	369.125	1649.775

TABLE C.15 – Résultats de placement des IPs avec DEMO best_1 en 2D pour les benchmarks de TGFF

Benchmark ID	solutions	Ressources			Meilleur placement			placement Moyenne		
		Minimum	Maximum	Moyenne	Puissance	Temps	Surface	Puissance	Temps	Surface
15	12	5	8	7.167	304.114	46.447	23.440	389.544	53.457	29.239
16	9	8	12	9.889	416.361	7.152	40.062	460.101	10.359	47.361
17	4	4	5	4.500	180.360	4.610	10.750	194.708	5.021	12.033
18	15	6	9	7.067	438.851	39.150	25.500	501.733	40.937	37.332
19	13	8	13	10.846	624.729	21.782	50.510	672.177	24.607	55.439
20	16	19	25	22.563	1363.917	24.171	119.966	1516.351	26.965	201.056
21	33	33	50	40.485	4271.200	87.412	446.730	4843.104	141.743	560.426
22	11	29	35	30.727	2202.502	40.289	222.186	2406.202	45.942	249.120
23	20	31	42	38.000	3094.972	43.053	332.343	3461.042	89.165	433.582
24	19	30	40	33.684	3077.387	39.805	304.107	3454.499	75.889	354.179
25	23	45	61	54.435	6463.903	80.938	657.874	7006.591	206.272	818.788
26	26	56	66	61.846	7398.623	78.295	808.499	8342.050	131.911	1054.975
27	16	58	69	64.063	7887.603	72.344	890.123	8471.765	103.167	1079.608
28	18	47	57	52.389	5051.230	75.294	527.187	5603.305	79.524	646.351
29	24	64	81	74.083	11490.228	173.254	1278.710	12213.003	283.687	1599.517

TABLE C.16 – Résultats de placement des IPs avec MOPSO en 3D pour les benchmarks de TGFF

Benchmark ID	solutions	Ressources			Meilleur Placement			Placement Moyenne		
		Minimum	Maximum	Moyenne	Puissance	Temps	Surface	Puissance	Temps	Surface
15	51	11	16	12.941	738.316	51.132	82.525	854.337	98.269	154.914
16	55	12	18	14.327	664.573	8.375	88.288	792.048	516.003	196.612
17	34	9	12	10.294	444.057	5.262	47.175	494.918	200.673	122.562
18	62	13	19	16.242	978.255	40.747	110.325	1140.635	223.581	222.732
19	58	16	23	19.448	1019.796	26.611	230.073	1171.935	297.458	405.590
20	71	24	32	28.803	1712.014	26.100	317.283	1982.433	372.416	551.759
21	97	44	57	51.557	4627.245	137.484	913.661	5076.645	465.971	1270.826
22	93	31	39	35.183	2401.523	42.931	463.207	2757.045	94.280	797.283
23	132	41	53	46.583	3505.427	43.379	709.082	3875.688	245.673	1045.644
24	73	37	44	40.699	3395.762	37.010	625.049	3764.982	75.155	960.310
25	52	63	73	68.038	6364.672	92.265	1229.920	6951.438	288.215	1570.616
26	99	73	84	78.455	7585.351	99.726	1563.869	8261.618	235.131	2049.792
27	120	77	89	82.808	7869.803	86.805	1769.401	8566.195	729.789	2049.688
28	82	52	65	60.598	5265.562	84.877	976.463	5831.163	174.681	1540.806
29	79	87	106	98.797	10988.529	255.784	2309.044	11695.416	694.992	2891.481

TABLE C.17 – Resultats de placement des IPs avec DEMO rand_1 en 3D pour les benchmarks de TGFF

Benchmark ID	solutions	Ressources			Meilleur Placement			Placement Moyenne		
		Minimum	Maximum	Moyenne	Puissance	Temps	Surface	Puissance	Temps	Surface
15	20	6	11	8.600	451.408	46.875	32.070	510.093	105.490	43.209
16	15	8	11	9.400	434.615	7.215	35.380	503.290	138.684	50.480
17	9	7	11	8.000	262.747	4.844	20.190	304.583	70.837	24.396
18	28	10	17	13.321	662.100	41.351	54.616	810.037	82.374	94.142
19	4	8	11	9.500	662.292	21.906	51.260	687.527	23.808	55.188
20	17	19	28	23.588	1258.325	25.341	107.610	1515.538	26.846	203.052
21	20	41	52	45.400	4148.799	88.794	453.474	4405.583	118.129	550.069
22	23	23	34	29.652	1984.349	40.040	189.374	2090.420	44.958	251.263
23	18	36	47	43.389	3163.427	37.913	368.042	3385.305	73.659	507.561
24	22	32	43	36.727	2930.779	35.411	279.310	3167.083	44.207	363.697
25	8	52	63	57.750	5364.094	75.138	558.714	5662.082	86.845	687.119
26	28	63	79	71.393	6791.555	70.576	764.852	7316.530	91.645	1006.179
27	4	61	76	68.000	7113.403	64.659	875.570	7280.517	70.242	1149.799
28	14	53	61	56.357	4692.121	58.996	440.369	5015.846	63.860	543.655
29	7	70	80	76.571	9990.729	165.506	1293.738	10186.693	278.463	1470.463

TABLE C.18 – Résultats de placement des IPs avec DEMO best_1 en 3D pour les benchmarks de TGFF

Benchmark ID	solutions	Ressources			Meilleur Placement			Placement Moyenne		
		Minimum	Maximum	Moyenne	Puissance	Temps	Surface	Puissance	Temps	Surface
15	15	4	8	6.333	329.622	46.293	24.380	371.177	129.977	32.214
16	26	6	9	7.654	374.271	6.947	32.940	407.164	198.940	40.636
17	6	6	7	6.333	212.252	4.606	14.500	235.425	4.754	18.832
18	12	8	12	9.750	569.253	39.320	30.250	636.837	40.070	49.673
19	15	7	10	9.133	503.110	21.722	43.380	635.702	22.169	52.334
20	18	19	23	21.222	1126.295	23.390	104.231	1292.030	57.829	135.793
21	25	37	51	46.160	4205.705	104.932	417.993	4476.336	125.744	548.880
22	20	24	31	27.100	1674.770	28.563	169.928	1952.865	32.796	215.935
23	27	36	49	42.519	2918.296	41.533	293.460	3181.613	60.518	445.375
24	22	28	37	31.318	2639.820	31.850	231.512	2814.102	139.356	291.808
25	32	52	64	57.094	5296.903	74.037	569.424	5606.127	97.453	737.971
26	12	59	74	66.417	6458.840	72.623	732.509	6870.811	85.862	893.747
27	24	63	76	69.792	6700.114	66.082	842.046	7134.066	85.157	1068.042
28	10	53	61	57.100	4664.708	70.211	509.950	4909.966	104.429	706.960
29	32	62	86	76.063	9500.075	178.380	1186.738	10055.689	279.961	1464.504

TABLE C.19 – Exemple de solutions de placement non dominées trouvée par MOPSO en NoC 2D

Graphe de Tâche	Affectation non-dominée	Placement non-dominée	P	T	S
TG2	[6, 6, 6, 6, 6, 10, 16, 6, 6, 5, 11, 6]	[2, 2, 2, 1, 0, 3, 4, 1, 2, 5, 6, 1]	x		
	[2, 4, 6, 6, 14, 11, 3, 16, 13, 11, 6, 11]	[3, 1, 7, 2, 6, 4, 8, 5, 9, 4, 2, 4]		x	
	[13, 6, 5, 0, 6, 10, 5, 6, 6, 5, 9, 10]	[2, 3, 7, 4, 3, 5, 6, 3, 3, 6, 8, 5]			x
TG0	[5, 6, 13, 0, 0, 0, 15, 14, 15, 0, 11, 15, 14, 14, 16, 13]	[1, 2, 3, 4, 4, 4, 5, 7, 6, 0, 8, 5, 7, 7, 15, 9]	x		
	[10, 6, 2, 6, 3, 0, 6, 10, 6, 15, 0, 0, 3, 3, 3, 6]	[8, 3, 1, 11, 2, 0, 10, 12, 3, 4, 5, 5, 2, 2, 13, 6]		x	
	[5, 6, 13, 0, 0, 0, 15, 14, 15, 0, 11, 15, 14, 14, 16, 13]	[3, 4, 2, 5, 5, 5, 13, 6, 7, 8, 9, 7, 6, 6, 10, 1]			x
TG1	[12, 10, 15, 15, 0, 14, 13, 0, 16, 0, 13, 6, 0, 8, 0, 13, 7, 6]	[6, 3, 8, 7, 4, 9, 5, 13, 10, 4, 12, 11, 13, 14, 1, 12, 15, 16]	x		
	[6, 10, 6, 6, 6, 4, 3, 2, 15, 13, 6, 6, 0, 11, 0, 6, 2, 4]	[1, 6, 11, 13, 7, 12, 8, 9, 10, 14, 13, 20, 15, 16, 15, 17, 0, 1]		x	
	[12, 10, 15, 15, 0, 14, 13, 0, 16, 0, 13, 6, 0, 8, 0, 13, 7, 6]	[6, 3, 8, 7, 4, 9, 5, 13, 10, 4, 12, 11, 13, 14, 1, 12, 15, 16]			x
TG3	[6, 6, 2, 5, 0, 13, 0, 6, 0, 6, 6, 8, 6, 6, 5, 3, 0, 6, 6, 6]	[3, 3, 4, 5, 6, 1, 2, 7, 6, 7, 3, 8, 3, 3, 5, 9, 10, 7, 3, 3]	x		
	[14, 6, 6, 6, 11, 4, 14, 3, 2, 4, 1, 3, 9, 2, 5, 6, 16, 16, 6, 6]	[7, 4, 8, 9, 10, 11, 12, 13, 14, 15, 5, 19, 18, 14, 1, 20, 16, 16, 8, 8]		x	
	[6, 6, 2, 5, 0, 13, 0, 6, 0, 6, 6, 8, 6, 6, 5, 3, 0, 6, 6, 6]	[3, 3, 4, 5, 6, 1, 2, 7, 6, 7, 3, 8, 3, 3, 5, 9, 10, 7, 3, 3]			x
TG4	[13, 11, 8, 6, 15, 2, 6, 6, 11, 0, 6, 2, 13, 9, 6, 2, 6, 3, 2, 6, 6, 5, 10, 6]	[11, 12, 13, 6, 4, 14, 15, 15, 2, 16, 15, 14, 7, 17, 15, 14, 15, 9, 14, 18, 15, 24, 19, 18]	x		
	[2, 10, 10, 16, 8, 6, 4, 4, 15, 13, 9, 0, 2, 2, 6, 8, 0, 6, 6, 6, 4, 4, 2, 6]	[7, 3, 2, 16, 1, 4, 8, 5, 12, 6, 0, 9, 10, 7, 14, 11, 9, 4, 13, 13, 15, 5, 17, 4]		x	
	[0, 14, 5, 9, 8, 6, 13, 3, 16, 13, 0, 13, 14, 16, 6, 10, 14, 6, 6, 6, 14, 16, 15, 6]	[0, 1, 5, 6, 10, 9, 11, 7, 17, 8, 14, 8, 12, 13, 15, 16, 12, 18, 15, 18, 19, 20, 21, 18]			x

TABLE C.20 – Exemple de solutions de placement non dominées trouvée par MOPSO en NoC 3D

Graphe de Tâche	Affectation non-dominée	Placement non-dominée	P	T	S
TG2	[6, 12, 0, 6, 13, 8, 10, 15, 0, 15, 15, 5]	[5, 13, 8, 3, 4, 15, 6, 7, 16, 7, 7, 9]	x		
	[6, 6, 4, 2, 7, 11, 4, 16, 13, 11, 16, 6]	[13, 4, 14, 12, 6, 16, 21, 3, 5, 15, 2, 13]		x	
	[13, 12, 16, 13, 13, 6, 16, 5, 13, 16, 6, 9]	[17, 14, 16, 17, 9, 10, 15, 13, 1, 7, 10, 20]			x
TG0	[5, 6, 6, 6, 12, 6, 6, 15, 6, 5, 15, 15, 8, 6, 15, 0]	[9, 12, 12, 12, 13, 1, 22, 23, 14, 24, 8, 3, 4, 5, 15, 0]	x		
	[9, 6, 6, 4, 6, 6, 6, 6, 0, 11, 6, 9, 15, 7, 13]	[24, 25, 25, 23, 22, 25, 11, 20, 16, 1, 12, 25, 19, 21, 4, 13]		x	
	[15, 6, 13, 13, 0, 13, 6, 15, 6, 6, 0, 5, 9, 9, 16, 13]	[6, 7, 24, 15, 8, 24, 7, 21, 7, 7, 16, 9, 17, 17, 13, 10]			x
TG1	[6, 15, 6, 6, 6, 6, 6, 6, 0, 13, 13, 14, 0, 6, 15, 9]	[17, 13, 14, 14, 22, 14, 22, 24, 22, 26, 15, 16, 25, 4, 15, 17, 23, 0]	x		
	[6, 11, 6, 3, 6, 2, 2, 14, 3, 6, 0, 13, 2, 9, 2, 7, 5, 16]	[7, 4, 5, 13, 9, 14, 18, 24, 21, 22, 6, 15, 3, 23, 2, 19, 10, 12]		x	
	[15, 15, 15, 12, 15, 6, 6, 0, 16, 13, 0, 13, 0, 9, 13, 13, 9, 9]	[26, 20, 18, 10, 18, 5, 25, 0, 3, 12, 0, 15, 0, 1, 9, 12, 1, 1]			x
TG3	[13, 6, 6, 0, 0, 0, 13, 14, 6, 6, 15, 16, 2, 13, 15, 9, 5, 2, 6, 6]	[23, 26, 13, 14, 14, 14, 12, 21, 15, 0, 3, 22, 16, 6, 17, 1, 10, 16, 11, 18]	x		
	[6, 6, 6, 14, 4, 4, 3, 13, 3, 4, 2, 6, 9, 4, 6, 9, 16, 4, 6, 6]	[13, 20, 10, 21, 22, 22, 23, 11, 23, 12, 15, 13, 14, 6, 9, 14, 16, 17, 13, 18]		x	
	[0, 0, 15, 5, 6, 0, 13, 6, 13, 0, 8, 15, 16, 13, 16, 5, 15, 15, 0, 6]	[6, 4, 7, 5, 8, 9, 10, 8, 12, 9, 22, 21, 24, 17, 11, 16, 7, 13, 6, 8]			x
TG4	[0, 9, 11, 15, 5, 0, 3, 14, 8, 14, 10, 14, 6, 7, 6, 6, 7, 3, 2, 6, 2, 0, 3, 0]	[4, 3, 12, 25, 15, 13, 1, 22, 26, 14, 2, 0, 5, 16, 5, 10, 6, 7, 8, 17, 8, 23, 1, 13]	x		
	[13, 16, 8, 14, 16, 15, 6, 14, 11, 2, 2, 13, 13, 6, 15, 10, 0, 15, 2, 0, 6, 9, 14, 14]	[11, 12, 1, 13, 12, 23, 5, 13, 8, 14, 14, 19, 4, 17, 15, 16, 25, 18, 14, 25, 17, 24, 22, 13]			x

TABLE C.21 – Exemple de solutions de placement non dominées trouvée par DEMO best_1 en NoC 2D

Graphe de Tâche	Affectation non-dominée	Placement non-dominée	P	T	S
TG2	[13, 6, 15, 13, 13, 11, 15, 15, 13, 15, 11, 15]	[1, 2, 3, 1, 1, 4, 3, 3, 1, 3, 4, 3]	x		
	[13, 6, 15, 13, 13, 11, 16, 16, 13, 11, 16, 16]	[0, 1, 2, 0, 0, 3, 4, 4, 0, 3, 4, 4]		x	
	[6, 6, 11, 13, 13, 11, 15, 16, 13, 11, 16, 16]	[0, 1, 2, 0, 0, 2, 3, 4, 0, 2, 4, 4]			x
TG0	[15, 6, 13, 13, 6, 13, 6, 15, 6, 15, 15, 15, 15, 15, 16, 13]	[0, 1, 2, 2, 1, 2, 5, 0, 1, 3, 0, 0, 0, 0, 4, 6]	x		
	[16, 6, 13, 13, 2, 13, 6, 15, 6, 15, 15, 16, 11, 16, 16, 13]	[0, 1, 2, 2, 3, 2, 6, 4, 1, 4, 4, 5, 7, 5, 5, 2]		x	
	[15, 15, 13, 13, 0, 13, 6, 15, 6, 15, 15, 15, 15, 15, 13]	[0, 0, 1, 1, 2, 1, 3, 4, 3, 4, 0, 4, 4, 4, 1]			x
TG1	[6, 15, 15, 6, 0, 13, 13, 13, 15, 13, 13, 13, 13, 15, 15]	[1, 2, 6, 1, 3, 7, 4, 4, 5, 4, 0, 0, 7, 11, 12, 7, 6, 5]	x		
	[6, 11, 6, 6, 13, 13, 13, 16, 13, 13, 13, 11, 13, 13, 15, 16]	[3, 4, 1, 3, 3, 5, 5, 7, 8, 5, 9, 9, 2, 10, 5, 5, 11, 6]		x	
	[6, 11, 15, 6, 6, 13, 13, 13, 15, 13, 13, 13, 13, 9, 13, 15, 15]	[3, 4, 5, 3, 3, 6, 6, 6, 7, 6, 6, 8, 11, 9, 8, 6, 5, 5]			x
TG3	[13, 6, 6, 15, 9, 13, 13, 13, 13, 6, 9, 15, 15, 13, 15, 9, 15, 15, 6, 6]	[0, 1, 1, 2, 3, 0, 0, 0, 6, 1, 3, 2, 2, 4, 2, 3, 2, 2, 1, 1]	x		
	[13, 6, 6, 15, 11, 13, 13, 13, 13, 6, 11, 15, 15, 13, 15, 15, 16, 16, 6, 6]	[0, 1, 1, 2, 3, 4, 4, 4, 1, 3, 2, 2, 4, 2, 6, 5, 5, 1, 1]		x	
	[13, 6, 6, 15, 9, 13, 13, 13, 13, 6, 9, 15, 15, 13, 15, 15, 16, 15, 6, 6]	[0, 1, 1, 2, 3, 4, 4, 4, 5, 1, 3, 2, 2, 0, 2, 2, 6, 2, 1, 1]			x
TG4	[13, 16, 15, 15, 15, 6, 13, 13, 15, 13, 9, 13, 15, 6, 15, 13, 6, 15, 13, 15, 15, 13]	[1, 2, 3, 3, 3, 4, 7, 1, 8, 1, 9, 7, 7, 3, 10, 5, 6, 10, 5, 1, 6, 3, 12, 6]	x		
	[13, 16, 16, 16, 15, 6, 13, 13, 15, 13, 11, 13, 13, 16, 6, 15, 13, 6, 15, 13, 13, 15, 11, 13]	[1, 2, 2, 5, 3, 4, 7, 6, 8, 7, 9, 7, 7, 2, 10, 8, 7, 10, 3, 7, 7, 8, 9, 15]			x
	[13, 15, 15, 16, 15, 6, 13, 13, 15, 13, 15, 13, 13, 16, 6, 15, 13, 15, 15, 13, 13, 15, 15, 13]	[1, 2, 3, 4, 3, 5, 7, 0, 6, 1, 6, 7, 7, 4, 5, 2, 1, 6, 2, 1, 7, 2, 2, 1]			x

TABLE C.22 – Exemple de solutions de placement non dominées trouvée par DEMO best_1 en NoC 3D

Graphe de Tâche	Affectation non-dominée	Placement non-dominée	P	T	S
TG2	[13, 6, 15, 13, 13, 11, 15, 16, 13, 15, 16, 15]	[0, 1, 2, 0, 3, 4, 5, 6, 0, 2, 6, 2]	x		
	[13, 6, 15, 13, 13, 11, 15, 16, 13, 11, 16, 16]	[1, 2, 3, 4, 4, 5, 6, 7, 4, 5, 7, 7]		x	
	[13, 6, 11, 13, 13, 11, 11, 16, 13, 11, 16, 16]	[0, 1, 2, 0, 4, 2, 5, 3, 0, 2, 3, 3]			x
TG0	[15, 6, 13, 13, 6, 13, 6, 15, 6, 15, 15, 6, 15, 15, 15, 13]	[1, 2, 3, 3, 2, 3, 2, 1, 2, 1, 0, 2, 1, 1, 0, 3]	x		
	[11, 6, 13, 13, 2, 13, 6, 11, 6, 11, 11, 16, 11, 16, 16, 13]	[0, 1, 2, 2, 3, 2, 9, 4, 1, 4, 4, 5, 0, 5, 5, 2]		x	
	[15, 15, 13, 13, 12, 13, 15, 15, 15, 15, 15, 15, 15, 16, 13]	[0, 0, 1, 1, 2, 1, 3, 3, 0, 0, 0, 3, 3, 4, 1]			x
TG1	[6, 11, 15, 6, 6, 13, 13, 13, 15, 13, 13, 13, 13, 9, 13, 13, 15, 15]	[0, 1, 4, 0, 2, 5, 5, 5, 4, 5, 5, 5, 5, 7, 5, 5, 4, 3]	x		
	[6, 11, 6, 6, 13, 13, 13, 16, 13, 13, 13, 13, 11, 13, 13, 11, 16]	[1, 2, 4, 3, 1, 5, 5, 5, 6, 5, 5, 5, 5, 7, 5, 5, 2, 6]		x	
	[12, 11, 15, 15, 12, 13, 13, 13, 15, 13, 13, 13, 11, 13, 13, 15, 15]	[0, 1, 4, 2, 3, 5, 5, 5, 6, 5, 5, 7, 5, 8, 5, 5, 4, 2]			x
TG3	[13, 6, 6, 11, 11, 13, 13, 13, 6, 11, 15, 15, 13, 15, 15, 11, 6, 6]	[0, 1, 1, 2, 2, 4, 4, 3, 7, 5, 6, 8, 8, 3, 8, 8, 8, 2, 1, 11]	x		
	[13, 6, 6, 16, 11, 13, 13, 13, 6, 11, 15, 15, 13, 15, 16, 16, 6, 6]	[1, 0, 2, 3, 4, 5, 1, 1, 5, 0, 4, 6, 6, 5, 6, 6, 7, 10, 0, 2]		x	
	[13, 6, 6, 15, 11, 13, 13, 13, 6, 11, 15, 15, 13, 15, 16, 15, 6, 6]	[0, 1, 1, 2, 3, 4, 4, 4, 5, 1, 3, 2, 2, 4, 6, 6, 7, 6, 8, 8]			x
TG4	[13, 15, 15, 15, 15, 6, 13, 13, 15, 13, 11, 13, 13, 15, 6, 15, 13, 6, 15, 13, 13, 15, 11, 13]	[1, 2, 2, 2, 2, 4, 1, 1, 2, 1, 5, 1, 1, 2, 3, 6, 1, 3, 6, 7, 1, 6, 5, 7]	x		
	[13, 16, 16, 16, 15, 6, 13, 13, 15, 13, 11, 13, 13, 16, 6, 15, 13, 6, 15, 13, 13, 15, 11, 13]	[1, 3, 3, 2, 4, 5, 1, 1, 6, 1, 7, 1, 1, 8, 5, 6, 9, 5, 10, 9, 9, 10, 7, 9]		x	
	[13, 16, 15, 16, 15, 6, 13, 13, 15, 13, 10, 13, 13, 16, 6, 15, 13, 6, 15, 13, 13, 15, 10, 13]	[1, 2, 4, 3, 4, 5, 6, 6, 7, 6, 8, 6, 6, 9, 5, 7, 1, 5, 7, 6, 10, 4, 8, 6]			x

Resultat de la phase de routage

Pour la phase de routage, les tests effectués avec chacun des six modèles de génération de trafic synthétique ont abouti à des résultats organisés de la manière suivante: pour chaque modèle de trafic utilisé, nous exposons la valeur de congestion obtenue, la latence par paquet, et le débit par l'utilisation de l'algorithme PSO en topologie 2D ainsi que 3D. Ces résultats sont exposés dans les Tables C.23 et C.24. De même, les résultats des Tables C.25 et C.26 illustrent les résultats obtenus par l'application du routage avec l'algorithme DE en 2D ainsi que 3D.

TABLE C.23 – Résultat de routage utilisant PSO en NoC 2D

Type	Pattern	#Congestion	Latence	Débit
Aléatoire	Rand	192	211.26	0.62
	HotSpot	196	217.99	0.64
	Local	33	92.7	0.84
Déterministe	Transpose 1	188	155.6	0.50
	Transpose 2	186	208.05	0.48
	Complement	502	121.7	0.52

TABLE C.24 – Résultat de routage utilisant PSO en NoC 3D

Type	Pattern	#Congestion	Latence	Débit
Aléatoire	Rand	130	191.2	0.67
	HotSpot	131	124.8	0.60
	Local	31	79.6	0.85
Déterministe	Transpose 1	120	155.6	0.40
	Transpose 2	122	81.7	0.50
	Complement	292	34.66	0.29

TABLE C.25 – Résultat de routage utilisant DE en NoC 2D

Type	Pattern	#Congestion	Latence	Débit
Aléatoire	Rand	188	179.34	0.76
	HotSpot	202	199.67	0.71
	Local	31	78.31	0.82
Déterministe	Transpose 1	178	146.5	0.55
	Transpose 2	178	114.42	0.51
	Complement	462	129.8	0.59

TABLE C.26 – Résultat de routage utilisant DE en NoC 3D

Type	Pattern	#Congestion	Latence	Débit
Aléatoire	Rand	123	170.126	0.78
	HotSpot	124	128.391	0.74
	Local	29	69.48	0.87
Déterministe	Transpose 1	126	128.7	0.42
	Transpose 2	112	52.4	0.53
	Complement	260	32.76	0.36

Les valeurs numériques issues des simulations de routage effectuées sur des benchmarks sont exposées dans cette partie. Les Tables C.29, C.27, C.28, C.30, et C.31 illustrent les résultats de la latence obtenue lors de la simulation pour chacun des placements effectués pour chaque benchmark.

TABLE C.27 – Latence obtenue pour le graphe de tâches TG0

Benchmarks	Placement 2D				Placement 3D			
	OE	XY	PSO	DE	OE 3D	XYZ	PSO	DE
TG0_0	225	203	229	216	192	182	183	176
TG0_1	184	198	195	187	223	196	199	208
TG0_2	215	214	204	189	216	201	204	191
TG0_3	196	200	196	189	193	183	189	184
TG0_4	198	218	182	179	222	204	209	181

TABLE C.28 – Latence obtenue pour le graphe de tâches TG1

Benchmarks	Placement 2D				Placement 3D			
	OE	XY	PSO	DE	OE 3D	XYZ	PSO	DE
TG1_0	199	187	200	195	215	192	195	196
TG1_1	237	238	230	237	216	189	197	202
TG1_2	226	237	234	230	231	224	228	209
TG1_3	216	195	213	207	196	183	188	176
TG1_4	195	204	188	139	205	185	190	194

TABLE C.29 – Latence obtenue pour le graphe de tâches TG2

Benchmarks	Placement 2D				Placement 3D			
	OE	XY	PSO	DE	OE 3D	XYZ	PSO	DE
TG2_0	179	183	184	183	171	165	169	170
TG2_1	225	209	198	201	178	173	171	166
TG2_2	204	202	192	181	148	153	142	140
TG2_3	153	148	149	148	152	156	151	153
TG2_4	194	171	166	178	160	139	152	147

TABLE C.30 – Latence obtenue pour le graphe de tâches TG3

Benchmarks	Placement 2D				Placement 3D			
	OE	XY	PSO	DE	OE 3D	XYZ	PSO	DE
TG3_0	218	220	213	213	244	234	222	220
TG3_1	226	228	224	219	254	224	234	169
TG3_2	223	219	210	215	188	188	182	183
TG3_3	357	229	322	330	225	222	215	204
TG3_4	238	229	240	244	220	205	199	205

TABLE C.31 – Latence obtenue pour le graphe de tâches TG4

Benchmarks	Placement 2D				Placement 3D			
	OE	XY	PSO	DE	OE 3D	XYZ	PSO	DE
TG4_0	294	300	294	294	239	221	226	212
TG4_1	281	279	272	262	219	201	206	214
TG4_2	237	242	220	154	241	223	221	231
TG4_3	188	259	148	174	226	221	199	214
TG4_4	322	290	198	108	246	251	244	175