



République Algérienne Démocratique et Populaire

Université Abou Bakr Belkaïd – Tlemcen

Faculté des Sciences

Département Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Master en Informatique

Option : Réseau et Système Distribué (R.S.D)

Thème

**Optimisation des Algorithmes de Routage dans les Architectures
Manycores par des Méthodes Bio-inspirées**

Réalisé par :

- MR. BELABBES NABIL

Présenté le 30 SEPTEMBRE 2026 devant le jury composé de :

-MR BENAMMAR ABDELKARIM

PRESIDENT

-MR MANA MOHAMMED

ENCADREUR

-MR BENAÏSSA MOHAMMED

JURY

Année universitaire : 2025 - 2026

Remerciements

Je tiens tout d'abord à remercier Dieu Tout-Puissant de m'avoir donné la force, la patience et la volonté pour mener à terme ce travail.

Je souhaite exprimer ma profonde gratitude à mon encadrant, Mr Mana Mohammed, pour sa disponibilité, ses conseils avisés, sa patience et le suivi rigoureux qu'il a assuré tout au long de la réalisation de ce mémoire.

Mes remerciements s'adressent également aux membres du Jury pour avoir accepté d'évaluer ce travail, ainsi qu'à l'ensemble des enseignants du Département d'Informatique de l'Université Abou Bakr Belkaïd de Tlemcen pour la qualité de la formation dispensée durant ce cursus.

Enfin, je remercie ma famille et mes amis pour leur soutien moral indéfectible durant toute la période de préparation de ce mémoire.

Dédicace

Je dédie ce modeste travail à mes chers parents, pour leur soutien constant, leurs sacrifices et leurs encouragements tout au long de mon parcours universitaire. Aucune dédicace ne saurait exprimer ma reconnaissance envers eux.

À mes frères et sœurs, pour leur présence et leurs encouragements.

À mes amis et camarades de promotion, avec qui j'ai partagé les moments difficiles et les moments de réussite de ces années d'études.

À toute personne ayant contribué de près ou de loin à la réalisation de ce mémoire.

Table des matières

Remerciements.....	
Dédicace.....	
Table des matières	1
Liste des figures	3
Liste des tableaux.....	4
Liste des abréviations.....	5
Introduction Générale	6
Problématique	7
Objectifs du mémoire.....	7
Organisation du mémoire.....	7
Chapitre I : Les Architectures Manycores et les Réseaux sur Puce.....	8
I.1 Introduction	9
I.2 Des architectures multicœurs aux architectures manycores	9
I.3 De l'interconnexion par bus au réseau sur puce.....	9
I.4 Concepts fondamentaux d'un NoC	10
I.4.1 Le routeur	10
I.4.2 Le lien.....	10
I.4.3 La topologie.....	10
I.4.4 La commutation.....	10
I.5 La topologie Mesh.....	11
I.6 Algorithmes de routage	11
I.6.1 Routage déterministe : le routage XY	11
I.6.2 Routage adaptatif.....	12
I.6.3 Vers une optimisation globale du routage	12
I.7 La problématique de la congestion et des hotspots	13
I.8 Conclusion.....	13
Chapitre II : Les Méthodes Bio-inspirées	14
II.1 Introduction	15
II.2 Métaheuristiques et optimisation combinatoire	15
II.3 Optimisation par Colonie de Fourmis (ACO).....	15
II.4 Optimisation par Essaim Particulaire (PSO).....	16
II.5 Algorithme Génétique (GA)	17
II.6 Application des méthodes bio-inspirées au routage dans les NoC	18
II.7 Synthèse comparative des trois méthodes.....	19
II.8 Conclusion.....	19

Chapitre III : Implémentation	20
III.1 Introduction.....	21
III.2 Choix technologiques et environnement de développement	21
III.3 Architecture logicielle.....	21
III.4 Modélisation du problème de routage	22
III.5 Le routage XY comme référence.....	23
III.6 Implémentation d'ACO	23
III.7 Implémentation de PSO	23
III.8 Implémentation de GA.....	24
III.9 Interface utilisateur	24
III.10 Conclusion	26
Chapitre IV : Résultats Expérimentaux et Analyse Comparative.....	27
IV.1 Introduction	28
IV.2 Protocole expérimental	28
IV.3 Résultats obtenus	28
IV.4 Analyse et discussion.....	29
IV.4.1 Latence et énergie : un résultat attendu	29
IV.4.2 Un gain modeste mais réel sur la congestion	29
IV.4.3 Un compromis intéressant sur la congestion de pointe	30
IV.4.4 Convergence des trois métaheuristiques vers une solution équivalente.....	30
IV.5 Limites de l'étude.....	31
IV.6 Perspectives d'amélioration	31
IV.7 Conclusion	32
Conclusion Générale.....	33
Références Bibliographiques	34
Résumé – Abstract – ملخص	35

Liste des figures

Figure I.1 — Exemple de topologie Mesh 2D pour un réseau sur puce

Figure I.2 — Principe du routage XY (Dimension-Order Routing)

Figure II.1 — Principe de la métaheuristique Colonie de Fourmis (ACO)

Figure II.2 — Principe de la métaheuristique Essaim Particulaire (PSO)

Figure II.3 — Principe de l'Algorithme Génétique (GA)

Figure III.1 — Architecture logicielle de l'application (packages model / algo / ui)

Figure III.2 — Écran d'accueil de l'application

Figure III.3 — Écran de configuration de la topologie (Mesh)

Figure III.4 — Écran de configuration du trafic

Figure III.5 — Écran de configuration des hyper-paramètres des algorithmes

Figure IV.1 — Résultats comparatifs XY / ACO / PSO / GA (trafic uniforme)

Figure IV.2 — Scénario A : écran de résultats de la simulation

Figure IV.3 — Scénario B : écran de résultats de la simulation

Liste des tableaux

Tableau II.1 — Synthèse comparative des trois métaheuristiques (ACO, PSO, GA)

Tableau III.1 — Paramètres de configuration de la topologie Mesh

Tableau III.2 — Paramètres de configuration du trafic

Tableau III.3 — Hyper-paramètres des algorithmes bio-inspirés

Tableau IV.1 — Paramètres du scénario expérimental

Tableau IV.2 — Résultats comparatifs détaillés (coût, latence, énergie, congestion)

Tableau IV.3 — Paramètres du scénario A

Tableau IV.4 — Résultats comparatifs du scénario A (maillage 16×8 , 500 paquets)

Tableau IV.5 — Paramètres du scénario B

Tableau IV.6 — Résultats comparatifs du scénario B (maillage 16×16 , 400 paquets)

Tableau IV.7 — Gains sur le coût total par rapport au routage XY pour les trois scénarios

Liste des abréviations

Abréviation	Signification
NoC	Network-on-Chip (Réseau sur Puce)
IP	Intellectual Property (bloc / cœur de calcul réutilisable)
ACO	Ant Colony Optimization (Optimisation par Colonie de Fourmis)
PSO	Particle Swarm Optimization (Optimisation par Essaim Particulaire)
GA	Genetic Algorithm (Algorithme Génétique)
XY	Routage déterministe par ordre de dimension (dimension-order routing)
MPSoC	Multi-Processor System-on-Chip
IHM	Interface Homme-Machine
QoS	Quality of Service (Qualité de Service)
2D	Bidimensionnel

Introduction Générale

La course à la performance des systèmes informatiques s'est heurtée, au cours des deux dernières décennies, aux limites physiques de la montée en fréquence des processeurs. Face à ce mur, les architectes de circuits se sont tournés vers le parallélisme massif : plutôt que de faire fonctionner un unique cœur toujours plus rapidement, on intègre désormais des dizaines, voire des centaines de cœurs de calcul sur une même puce. Ces architectures, dites Manycores, sont aujourd'hui au cœur des systèmes embarqués, des serveurs de calcul haute performance et des accélérateurs spécialisés.

Cette multiplication des unités de calcul déplace le problème : la performance globale ne dépend plus seulement de la puissance de chaque cœur, mais également de la capacité du système à faire communiquer efficacement ces cœurs entre eux. Les bus partagés traditionnels, conçus pour un petit nombre d'unités, atteignent rapidement leurs limites en termes de bande passante et de scalabilité. C'est dans ce contexte qu'est né le concept de réseau sur puce (Network-on-Chip, NoC) : un réseau d'interconnexion structuré, comparable à un réseau informatique miniaturisé, qui achemine les données entre les cœurs par l'intermédiaire de routeurs et de liens.

Au sein d'un NoC, l'algorithme de routage c'est-à-dire la stratégie qui détermine le chemin emprunté par chaque paquet de données entre sa source et sa destination joue un rôle déterminant. Un routage mal adapté peut entraîner une congestion localisée, en particulier lorsque le trafic n'est pas uniformément réparti (trafic dit hotspot, où un nœud reçoit une part disproportionnée des communications), dégradant la latence et augmentant la consommation énergétique du système.

Les algorithmes de routage déterministes classiques, tels que le routage XY, offrent une implémentation matérielle simple et un comportement prévisible, mais ne tiennent pas compte de l'état réel du réseau au moment de l'acheminement. Face à cette limite, la recherche s'est orientée vers des approches adaptatives capables d'exploiter la connaissance de la charge du réseau pour mieux répartir le trafic. Parmi les pistes explorées, les métaheuristiques bio-inspirées inspirées de phénomènes naturels tels que le comportement collectif des colonies de fourmis, la dynamique des essaims d'animaux, ou les mécanismes de la sélection naturelle se sont révélées être des outils pertinents pour aborder des problèmes d'optimisation combinatoire complexes, dont le routage fait partie.

Problématique

La problématique centrale de ce mémoire peut être formulée ainsi : dans quelle mesure des métaheuristiques bio-inspirées (Colonie de Fourmis, Essaim Particulaire, Algorithme Génétique) permettent-elles d'améliorer, par rapport à un routage déterministe de référence, les performances d'un réseau sur puce en topologie Mesh, en particulier en présence d'un trafic irrégulier de type hotspot ?

Objectifs du mémoire

- Étudier les fondements des architectures Manycores et des réseaux sur puce, ainsi que les problématiques de routage qui leur sont associées.
- Présenter les principes des trois métaheuristiques bio-inspirées retenues (ACO, PSO, GA) et leur adéquation avec le problème du routage.
- Concevoir et implémenter une application logicielle permettant de modéliser un NoC en topologie Mesh, de générer différents profils de trafic, et d'appliquer les trois métaheuristiques à la sélection des chemins de routage.
- Comparer expérimentalement les performances des trois approches bio-inspirées à celles du routage XY de référence, selon les critères de latence, de coût énergétique et de congestion.
- Analyser de façon rigoureuse et honnête les résultats obtenus, y compris leurs limites, afin d'en tirer des conclusions scientifiquement fondées.

Organisation du mémoire

Ce mémoire est structuré en quatre chapitres. Le premier chapitre présente les architectures Manycores et les réseaux sur puce, en détaillant les concepts fondamentaux nécessaires à la compréhension de la problématique du routage. Le deuxième chapitre est consacré aux méthodes bio-inspirées retenues dans ce travail, à savoir la Colonie de Fourmis, l'Essaim Particulaire et l'Algorithme Génétique. Le troisième chapitre décrit la conception et l'implémentation de l'application logicielle développée, incluant l'architecture du code, la modélisation du problème et l'interface utilisateur. Enfin, le quatrième chapitre présente le protocole expérimental mis en place, les résultats obtenus, ainsi qu'une analyse comparative et critique de ces résultats. Le mémoire se termine par une conclusion générale synthétisant les apports du travail et ouvrant des perspectives pour des travaux futurs.

Chapitre I :

Les Architectures

Manycores et les Réseaux

sur Puce

I.1 Introduction

Ce chapitre pose les bases nécessaires à la compréhension du sujet traité dans ce mémoire. Il retrace le passage des architectures monoprocesseur et multicœurs vers les architectures Manycores, présente le concept de réseau sur puce (NoC) qui en assure l'interconnexion, puis détaille les notions de topologie et de routage indispensables à l'étude menée dans les chapitres suivants.

I.2 Des architectures multicœurs aux architectures Manycores

Pendant plusieurs décennies, l'amélioration des performances des processeurs reposait principalement sur l'augmentation de la fréquence d'horloge et la complexification des unités d'exécution. Cette approche a atteint ses limites au milieu des années 2000, en raison de contraintes physiques liées à la dissipation thermique et à la consommation énergétique. L'industrie s'est alors tournée vers le parallélisme au niveau matériel, donnant naissance aux architectures multicœurs, qui intègrent plusieurs unités de calcul (cœurs) sur une même puce.

Lorsque le nombre de cœurs dépasse quelques dizaines, on parle d'architectures Manycores. Ces architectures se distinguent des architectures multicœurs classiques non seulement par le nombre d'unités de calcul, mais également par la nature de leur réseau d'interconnexion : au-delà d'une dizaine de cœurs, un bus partagé unique ne peut plus offrir une bande passante suffisante ni une scalabilité acceptable, ce qui impose le recours à un réseau d'interconnexion structuré.

I.3 De l'interconnexion par bus au réseau sur puce

Les premières architectures multicœurs utilisaient un bus partagé pour relier les différents cœurs à la mémoire et aux périphériques. Cette solution, simple à mettre en œuvre, présente cependant un inconvénient majeur : un seul transfert de données peut avoir lieu à un instant donné sur le bus, ce qui crée un goulot d'étranglement lorsque le nombre de cœurs augmente. De plus, la longueur des interconnexions physiques dans un bus partagé pose des problèmes de synchronisation à mesure que la taille de la puce augmente.

Le concept de réseau sur puce (Network-on-Chip, NoC), introduit au début des années 2000, propose de remplacer ce bus unique par un véritable réseau de communication, structurellement proche des réseaux informatiques classiques. Un NoC est composé de routeurs interconnectés par des liens, chaque cœur de calcul (ou bloc IP) étant relié à un routeur local. Les données circulent sous forme de paquets, découpés en unités élémentaires appelées flits, qui sont acheminés de routeur en routeur jusqu'à leur destination.

I.4 Concepts fondamentaux d'un NoC

I.4.1 Le routeur

Le routeur est l'élément central d'un NoC. Il reçoit les paquets entrants sur ses différents ports (généralement un port par direction du réseau, plus un port vers le cœur de calcul local), détermine, selon l'algorithme de routage employé, le port de sortie approprié, puis retransmet le paquet vers le routeur voisin correspondant. Chaque routeur dispose de tampons (buffers) permettant de stocker temporairement les paquets en cas de contention sur un lien de sortie.

I.4.2 Le lien

Un lien relie deux routeurs voisins et possède une bande passante limitée. Lorsque plusieurs paquets doivent emprunter simultanément le même lien, une contention se produit : certains paquets doivent patienter dans les tampons du routeur, ce qui accroît la latence de bout en bout.

I.4.3 La topologie

La topologie décrit la manière dont les routeurs sont interconnectés. Elle conditionne la distance moyenne entre deux nœuds (exprimée en nombre de sauts), le nombre de chemins possibles entre une source et une destination, ainsi que la complexité matérielle du réseau. Parmi les topologies les plus étudiées figurent la topologie en anneau, la topologie Mesh (grille bidimensionnelle), la topologie Torus (grille avec bouclage des extrémités), et des topologies hiérarchiques ou irrégulières.

I.4.4 La commutation

La commutation détermine la manière dont un paquet progresse à travers les routeurs : commutation de circuits (un chemin complet est réservé avant l'envoi),

commutation de paquets avec stockage intégral (store-and-Forward), ou commutation de type Wormhole, où le paquet est découpé en flits qui progressent en pipeline à travers les routeurs, réduisant la latence et les besoins en mémoire tampon.

I.5 La topologie Mesh

La topologie Mesh, retenue dans ce mémoire, organise les routeurs selon une grille bidimensionnelle régulière. Chaque routeur, hormis ceux situés en bordure, est connecté à quatre voisins directs (Nord, Sud, Est, Ouest) ainsi qu'au cœur de calcul local. Cette régularité facilite la conception physique du réseau (adressage simple par coordonnées (x, y), câblage court et régulier) et en fait l'une des topologies les plus utilisées dans l'industrie et dans la recherche académique. Sa principale limite réside dans le nombre de sauts qui peut devenir important entre deux nœuds éloignés, ainsi que dans la sensibilité à la congestion localisée en cas de trafic déséquilibré.

Topologie Mesh 2D (4x4) — Routeur (R) + cœur de calcul (PE) par nœud

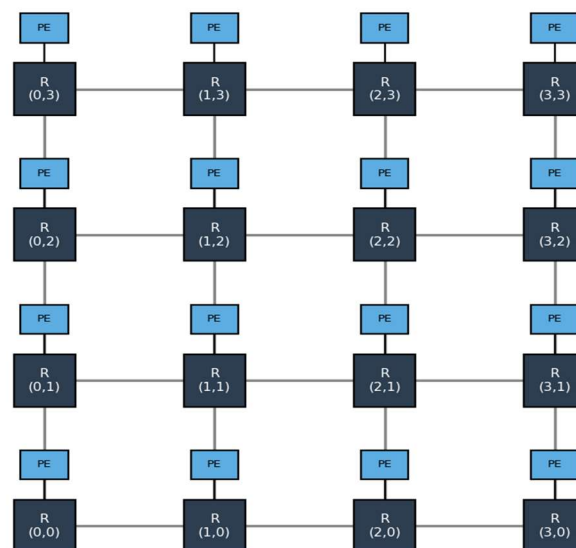


Figure I.1 — Exemple de topologie Mesh 2D pour un réseau sur puce

I.6 Algorithmes de routage

I.6.1 Routage déterministe : le routage XY

Le routage XY, aussi appelé routage par ordre de dimension (dimension-order routing), est l'algorithme de routage déterministe le plus répandu dans les topologies Mesh. Un paquet se déplace d'abord entièrement selon l'axe X jusqu'à atteindre la

coordonnée X de sa destination, puis entièrement selon l'axe Y. Cette règle simple garantit l'absence d'interblocage (DeadLock) et permet une implémentation matérielle minimale, mais présente l'inconvénient de toujours emprunter le même chemin entre deux nœuds donnés, sans tenir compte de l'état de charge du réseau.

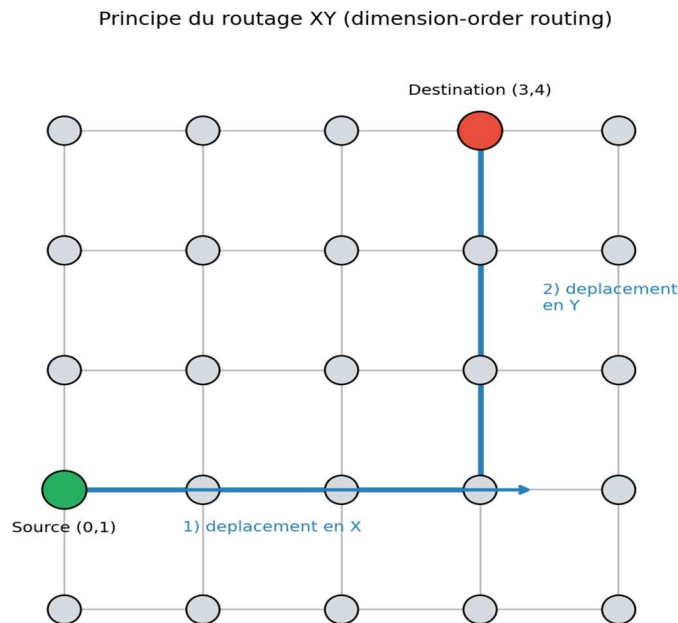


Figure I.2 — Principe du routage XY (dimension-order routing)

I.6.2 Routage adaptatif

Contrairement au routage déterministe, le routage adaptatif choisit dynamiquement le chemin emprunté par un paquet en fonction de l'état courant du réseau (occupation des tampons, congestion des liens voisins). Il permet en théorie de mieux répartir la charge et de contourner les zones congestionnées, au prix d'une complexité matérielle et logicielle accrue, et de la nécessité de mécanismes garantissant l'absence d'interblocage.

I.6.3 Vers une optimisation globale du routage

Entre le routage purement déterministe et le routage adaptatif décidé localement à chaque routeur, il est possible d'envisager une optimisation plus globale : déterminer, pour un ensemble de communications connu ou statistiquement caractérisé, l'affectation de chemins qui minimise un critère de performance donné (latence, énergie, congestion). Ce problème d'affectation, de nature combinatoire, devient rapidement intraitable par une recherche exhaustive lorsque le nombre de communications augmente, ce qui justifie le

recours à des méthodes d'optimisation approchées telles que les métaheuristiques bio-inspirées, présentées au chapitre suivant.

I.7 La problématique de la congestion et des hotspots

La congestion survient lorsque la demande de trafic sur un lien ou un ensemble de liens dépasse la capacité disponible, entraînant une accumulation de paquets dans les tampons des routeurs et une augmentation de la latence. Ce phénomène est particulièrement marqué dans le cas d'un trafic dit hotspot, où un ou plusieurs nœuds (par exemple un contrôleur mémoire partagé, ou un cache de dernier niveau) reçoivent une proportion disproportionnée des communications du réseau. Dans ce cas de figure, la congestion tend à se concentrer sur les liens d'entrée du nœud fortement sollicité, un phénomène qui sera observé expérimentalement au chapitre IV de ce mémoire.

I.8 Conclusion

Ce chapitre a présenté le contexte technologique des architectures Manycores et des réseaux sur puce, ainsi que les concepts fondamentaux nécessaires à l'étude du routage : topologie, commutation, et algorithmes de routage déterministes et adaptatifs. La problématique de la congestion, en particulier sous trafic hotspot, motive l'exploration de méthodes d'optimisation plus sophistiquées que le simple routage XY. Le chapitre suivant présente les méthodes bio-inspirées retenues dans ce mémoire pour répondre à cette problématique.

Chapitre II :

Les Méthodes Bio-inspirées

II.1 Introduction

De nombreux problèmes d'ingénierie, dont le routage optimal dans un réseau sur puce, se formulent comme des problèmes d'optimisation combinatoire pour lesquels une résolution exacte est impraticable en temps raisonnable dès que la taille du problème augmente. Les métaheuristiques bio-inspirées constituent une famille de méthodes d'optimisation approchées, qui s'inspirent de mécanismes observés dans la nature pour explorer efficacement un espace de solutions et converger vers des solutions de bonne qualité, sans garantie d'optimalité mais avec un coût de calcul raisonnable. Ce chapitre présente les trois métaheuristiques retenues dans ce mémoire : l'Optimisation par Colonie de Fourmis, l'Optimisation par Essaim Particulaire et l'Algorithme Génétique.

II.2 Métaheuristiques et optimisation combinatoire

Une métaheuristique est une méthode générique d'optimisation, indépendante du problème traité, qui guide une recherche dans l'espace des solutions au moyen de mécanismes d'exploration (diversification) et d'exploitation (intensification autour des solutions prometteuses). Contrairement aux méthodes exactes, une métaheuristique ne garantit pas de trouver l'optimum global, mais permet généralement d'obtenir une solution de bonne qualité en un temps de calcul maîtrisé, ce qui la rend particulièrement adaptée aux problèmes combinatoires de grande taille tels que l'affectation de chemins de routage à un grand nombre de communications simultanées.

II.3 Optimisation par Colonie de Fourmis (ACO)

L'Optimisation par Colonie de Fourmis (Ant Colony Optimization, ACO), introduite par Marco Dorigo au début des années 1990, s'inspire du comportement collectif des colonies de fourmis lors de la recherche de nourriture. Dans la nature, chaque fourmi dépose sur son chemin une substance chimique, la phéromone, dont la concentration influence le choix de trajet des fourmis suivantes. Les chemins les plus courts étant parcourus plus rapidement, ils bénéficient d'un dépôt de phéromone plus fréquent, ce qui crée un phénomène d'Auto-renforcement conduisant progressivement la colonie à converger vers le chemin le plus court entre le nid et la source de nourriture.

Transposé à un problème d'optimisation combinatoire, ce principe donne naissance à un algorithme itératif dans lequel une population d'agents artificiels (les fourmis) construit chacune une solution complète au problème, en choisissant probabilistiquement les composants de la solution en fonction d'une piste de phéromone (mémoire collective des choix passés) et d'une heuristique propre au problème (par exemple, favoriser les chemins courts). À la fin de chaque itération, la phéromone s'évapore partiellement (pour éviter une convergence prématurée) et est renforcée sur les chemins empruntés par les meilleures solutions trouvées.

Dans le contexte du routage NoC étudié dans ce mémoire, chaque « fourmi » construit une affectation complète de chemins pour l'ensemble des paquets à acheminer, le dépôt de phéromone favorisant, au fil des itérations, les chemins qui contribuent à des solutions globalement moins coûteuses (en termes de latence, d'énergie et de congestion).

Principe de la Colonie de Fourmis (ACO) : renforcement du chemin le plus court

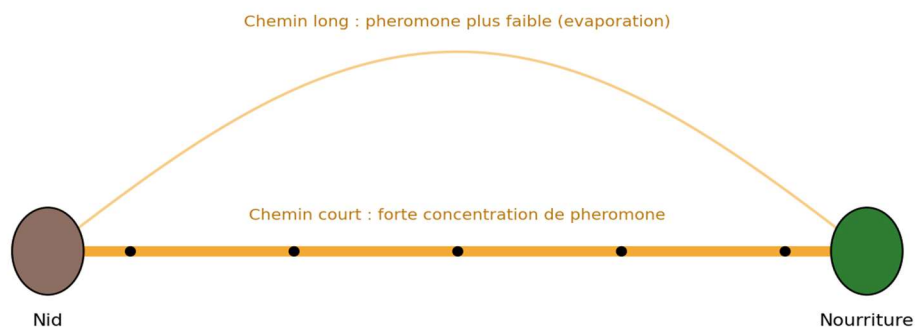


Figure II.1 — Principe de la métaheuristique Colonie de Fourmis (ACO)

II.3.1 Formalisation, paramètres et pseudo-code

Dans la formulation retenue, chaque fourmi construit une solution en choisissant, pour chaque paquet i , un chemin parmi les chemins candidats disponibles. La probabilité de choisir le chemin j pour le paquet i est donnée par :

$$P_{ij} = (\tau_{ij}^\alpha \cdot \eta_{ij}^\beta) / \sum_k (\tau_{ik}^\alpha \cdot \eta_{ik}^\beta) \quad (1)$$

où τ_{ij} désigne la quantité de phéromone associée au chemin j du paquet i , η_{ij} l'information heuristique locale (favorisant les chemins courts), α l'importance accordée à la phéromone et β celle accordée à l'heuristique. Après évaluation des solutions, la

phéromone est mise à jour en deux temps : une évaporation globale, puis un renforcement par les meilleures fourmis de l'itération, proportionnel à l'inverse du coût de leur solution :

$$\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij} \quad (2)$$

$$\tau_{ij} \leftarrow \tau_{ij} + Q / C(S) \quad \text{pour chaque choix } (i, j) \text{ d'une meilleure solution } S \quad (3)$$

où $\rho \in]0, 1]$ est le taux d'évaporation, Q une constante de dépôt et $C(S)$ le coût de la solution S (somme de la latence, de l'énergie et de la pénalité de congestion, définie à la section III.4). Un taux d'évaporation élevé favorise l'exploration, car la mémoire des choix passés s'efface plus vite ; un taux faible favorise au contraire l'exploitation des chemins déjà renforcés.

```
Initialiser la phéromone  $\tau(i, j)$  à une valeur uniforme
Pour  $t = 1$  à  $\text{nb\_iterations}$  :
    Pour chaque fourmi  $a = 1$  à  $\text{taille\_population}$  :
        Pour chaque paquet  $i$  : choisir un chemin  $j$  avec la probabilité  $P(i, j)$ 
        Évaluer le coût  $C$  de la solution construite
        Évaporer la phéromone :  $\tau(i, j) \leftarrow (1 - \rho) \cdot \tau(i, j)$ 
        Renforcer les choix des meilleures fourmis :  $\tau(i, j) \leftarrow \tau(i, j) + Q / C$ 
    Mémoriser la meilleure solution trouvée
Retourner la meilleure solution
```

Algorithme II.1 — Pseudo-code de l'optimisation par colonie de fourmis (ACO)

Dans l'application, ces paramètres sont saisis dans l'écran « Algorithmes » (Figure III.5) sous les noms `aco_evaporation` (ρ), `aco_alpha` (α) et `aco_beta` (β) ; le nombre de fourmis correspond à `taille_population` et le nombre de cycles à `nb_iterations`. Les plages usuelles sont $\rho \in]0, 1]$, avec α et β positifs.

II.4 Optimisation par Essaim Particulaire (PSO)

L'Optimisation par Essaim Particulaire (Particle Swarm Optimization, PSO), proposée par James Kennedy et Russell Eberhart en 1995, s'inspire du comportement collectif observé chez certaines espèces animales évoluant en groupe, telles que les bancs de poissons ou les vols groupés d'oiseaux. Chaque individu (particule) de l'essaim ajuste sa trajectoire en fonction de sa propre expérience (la meilleure position qu'il a lui-même atteinte) et de l'expérience collective du groupe (la meilleure position atteinte par l'ensemble de l'essaim), réalisant ainsi un compromis entre exploration individuelle et coordination sociale.

Formellement, chaque particule est représentée par un vecteur de position dans l'espace des solutions, associé à un vecteur de vitesse. À chaque itération, la vitesse est

mise à jour selon une combinaison pondérée de trois termes : l'inertie (tendance à poursuivre le mouvement précédent), la composante cognitive (attraction vers la meilleure position personnelle) et la composante sociale (attraction vers la meilleure position globale de l'essaim). La position de la particule est ensuite mise à jour en ajoutant sa nouvelle vitesse.

Dans le problème traité ici, chaque particule représente une affectation complète de chemins pour l'ensemble des paquets, chaque coordonnée du vecteur de position étant associée, après arrondi, au choix du chemin candidat pour un paquet donné.

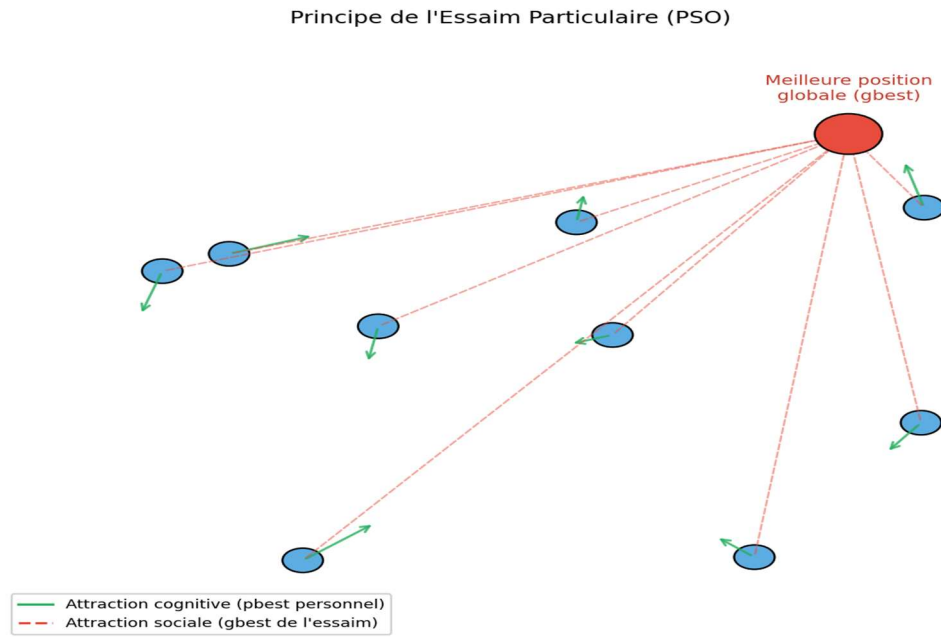


Figure II.2 — Principe de la métaheuristique Essaim Particulaire (PSO)

II.4.1 Formalisation, paramètres et pseudo-code

Chaque particule possède une position x et une vitesse v , de dimension égale au nombre de paquets à router, ainsi que la mémoire de sa meilleure position personnelle (pbest). L'essaim partage en outre la meilleure position globale trouvée (gbest). À chaque itération, la vitesse puis la position sont mises à jour, pour chaque coordonnée d , selon :

$$v_d \leftarrow w \cdot v_d + c_1 \cdot r_1 \cdot (pbest_d - x_d) + c_2 \cdot r_2 \cdot (gbest_d - x_d) \quad (4)$$

$$x_d \leftarrow x_d + v_d \quad (5)$$

où w est le coefficient d'inertie, c_1 et c_2 les coefficients cognitif et social, et r_1 , r_2 deux nombres aléatoires tirés uniformément dans $[0, 1]$ à chaque mise à jour. La position étant un vecteur de nombres réels, elle est ensuite arrondie et bornée afin de désigner un indice de chemin candidat valide pour chaque paquet :

$$indice_d = \min(\max(\text{arrondi}(x_d), 0), m_d - 1) \quad (6)$$

où m_d est le nombre de chemins candidats du paquet d . Une inertie élevée favorise l'exploration de l'espace de recherche, tandis qu'une inertie faible renforce l'exploitation autour des meilleures positions connues.

```
Initialiser aléatoirement la position  $x$  et la vitesse  $v$  de chaque particule
Évaluer chaque particule ;  $pbest \leftarrow x$  ;  $gbest \leftarrow$  meilleure position de l'essaim
Pour  $t = 1$  à  $nb\_iterations$  :
    Pour chaque particule :
        Mettre à jour  $v$  puis  $x$  selon les équations (4) et (5)
        Arrondir et borner  $x$  pour obtenir l'affectation des chemins (équation (6))
        Évaluer le coût de la solution obtenue
        Si le coût est meilleur que celui de  $pbest$  : mettre à jour  $pbest$ 
        Si le coût est meilleur que celui de  $gbest$  : mettre à jour  $gbest$ 
Retourner  $gbest$ 
```

Algorithme II.2 — Pseudo-code de l'optimisation par essaim particulaire (PSO)

Dans l'application, ces paramètres correspondent aux champs $pso_inertie$ (w), pso_c1 (c_1) et pso_c2 (c_2) ; la taille de l'essaim correspond à $taille_population$. Les plages usuelles retenues dans la littérature sont une inertie comprise entre 0,4 et 0,9 et des coefficients c_1 et c_2 compris entre 1,5 et 2 environ.

II.5 Algorithme Génétique (GA)

L'Algorithme Génétique (Genetic Algorithm, GA), dont les bases théoriques ont été posées par John Holland dans les années 1970, s'inspire des mécanismes de l'évolution biologique par sélection naturelle. Une population de solutions candidates, appelées chromosomes ou individus, évolue au fil des générations au moyen de trois opérateurs principaux : la sélection, qui privilégie la reproduction des individus les mieux adaptés (les moins coûteux, dans un problème de minimisation) ; le croisement (crossover), qui combine le matériel génétique de deux parents pour produire un ou plusieurs descendants et la mutation, qui introduit aléatoirement de petites modifications afin de maintenir la diversité de la population et d'éviter une convergence prématurée vers un optimum local.

Dans ce mémoire, chaque chromosome encode une affectation complète de chemins pour l'ensemble des paquets à router, chaque gène représentant l'indice du chemin candidat choisi pour un paquet donné. Une stratégie d'élitisme est employée pour garantir que la meilleure solution trouvée à chaque génération n'est jamais perdue.

Principe de l'Algorithme Genetique (GA)

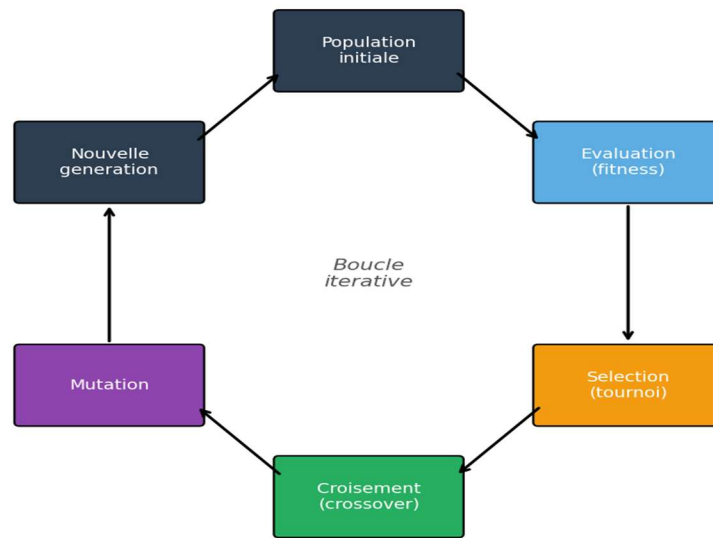


Figure II.3 — Principe de l'Algorithme Génétique (GA)

II.5.1 Formalisation, paramètres et pseudo-code

Un chromosome est un vecteur d'entiers dont chaque gène i donne l'indice du chemin candidat choisi pour le paquet i . À chaque génération, une nouvelle population est construite à l'aide des opérateurs suivants :

La sélection par tournoi tire au hasard quelques individus de la population et retient le meilleur, c'est-à-dire celui dont le coût est le plus faible. Le croisement uniforme est appliqué à deux parents avec une probabilité p_c : chaque gène de l'enfant est alors hérité aléatoirement de l'un des deux parents ; dans le cas contraire, l'enfant est une copie d'un parent. La mutation remplace chaque gène, avec une probabilité p_m , par un indice de chemin candidat tiré au hasard. Enfin, l'élitisme conserve systématiquement le meilleur individu d'une génération à la suivante.

```

Initialiser aléatoirement une population de chromosomes ; évaluer leur coût
Pour g = 1 à nb_iterations :
    Conserver le meilleur individu (élitisme)
    Tant que la nouvelle population n'est pas complète :
        Sélectionner deux parents par tournoi
        Avec la probabilité p_c : croisement uniforme ; sinon : copie d'un
        parent
        Pour chaque gène : avec la probabilité p_m, tirer un nouvel indice de
        chemin
        Ajouter l'enfant à la nouvelle population
    Évaluer la nouvelle population
Retourner le meilleur individu trouvé

```

Algorithme II.3 — Pseudo-code de l'algorithme génétique (GA)

Dans l'application, p_c et p_m correspondent aux champs `ga_croisement` et `ga_mutation`, la taille de la population à `taille_population` et le nombre de générations à `nb_iterations`. Ces deux probabilités appartiennent à l'intervalle $[0, 1]$: le taux de croisement est généralement élevé (de l'ordre de 0,6 à 0,9) et le taux de mutation faible (typiquement inférieur ou égal à 0,1), afin de préserver les bonnes solutions tout en maintenant la diversité.

II.6 Application des méthodes bio-inspirées au routage dans les NoC

De nombreux travaux de la littérature ont exploré l'application des métaheuristiques bio-inspirées à des problèmes d'optimisation liés aux réseaux sur puce, tels que le placement de tâches sur les nœuds du réseau, la configuration des tables de routage, ou encore l'ordonnancement des communications. Ces méthodes offrent un compromis intéressant entre la qualité des solutions obtenues et le temps de calcul nécessaire, en particulier lorsque le problème doit être résolu hors ligne (à la conception) plutôt qu'en temps réel à l'exécution.

II.7 Synthèse comparative des trois méthodes

Critère	ACO	PSO	GA
Inspiration	Comportement collectif des fourmis	Comportement collectif des essaims	Sélection naturelle / génétique
Représentation de la solution	Construction probabiliste guidée par phéromone	Vecteur de position réel	Chromosome (vecteur discret)
Mémoire collective	Phéromone (implicite, partagée)	Meilleure position globale (explicite)	Population (implicite via sélection)

Opérateurs principaux	Construction + évaporation/dépôt	Mise à jour vitesse/position	Sélection, croisement, mutation
Points forts	Bonne exploitation de solutions structurées	Convergence rapide, peu de paramètres	Bonne diversité, robustesse

II.8 Conclusion

Ce chapitre a présenté les principes fondamentaux des trois métaheuristiques bio-inspirées retenues dans ce mémoire. Bien que reposant sur des mécanismes d'inspiration différents (phéromones, dynamique d'essaim, évolution génétique), ces trois approches partagent une structure commune : une population de solutions candidates qui évolue itérativement vers des solutions de meilleure qualité, guidée par une fonction objective à minimiser. Le chapitre suivant détaille la manière dont ces trois méthodes ont été implémentées et appliquées au problème du routage dans un réseau sur puce en topologie Mesh.

Chapitre III : Implémentation

III.1 Introduction

Ce chapitre décrit la conception et l'implémentation de l'application logicielle développée dans le cadre de ce mémoire. Cette application permet de modéliser un réseau sur puce en topologie Mesh, de générer différents profils de trafic, et de comparer les performances du routage déterministe XY avec celles des trois métaheuristiques bio-inspirées présentées au chapitre précédent (ACO, PSO, GA).

III.2 Choix technologiques et environnement de développement

L'application a été développée en langage Java, avec la bibliothèque graphique Swing pour l'interface utilisateur, dans l'environnement de développement NetBeans. Ce choix technologique offre plusieurs avantages : portabilité multiplateforme du code Java, absence de dépendance à des bibliothèques externes complexes à installer, et une bonne adéquation avec les compétences acquises durant le cursus universitaire. Aucun simulateur cycle-accurate externe (tel que Noxim) n'a été utilisé pour cette application autonome ; le moteur de simulation a été entièrement développé en Java, sur la base d'un modèle de coût simplifié mais représentatif des phénomènes étudiés (latence, énergie, congestion).

Pour la réalisation des différentes simulations, nous avons utilisé une machine sous Windows 10 Entreprise (version 22H2, build 19045.6456) de 64 bits, avec un processeur Intel(R) Celeron(R) N2830 cadencé à 2,16 GHz, une mémoire RAM de 8 Go (7,90 Go utilisables), une carte graphique Intel(R) HD Graphics et un disque dur HDD de 466 Go. Il s'agit d'une configuration matérielle modeste ; les temps d'exécution présentés dans le chapitre IV sont donc donnés à titre comparatif entre les algorithmes, dans des conditions matérielles identiques.

III.3 Architecture logicielle

Le code de l'application est organisé selon une architecture en couches, répartie en trois paquetages (packages) aux responsabilités clairement séparées, ce qui facilite la lisibilité, la maintenance et l'évolution du code :

- Le paquetage model regroupe les classes représentant les données du problème : configuration de la topologie (MeshConfig), configuration du trafic

(TrafficConfig), hyper-paramètres des algorithmes (AlgoConfig), ainsi que les entités de simulation (Paquet, Chemin).

- Le paquetage algo regroupe la logique de calcul : génération du trafic (TrafficGenerator), génération des chemins candidats (CandidatePathGenerator), évaluation du coût d'une affectation (RoutingCostEvaluator), et l'implémentation des quatre stratégies de routage comparées (XYRoutingBaseline, AntColonyRoutingOptimizer, ParticleSwarmRoutingOptimizer, GeneticRoutingOptimizer).
- Le paquetage ui regroupe l'ensemble des composants de l'interface graphique : fenêtre principale à navigation par onglets (MainFrame), écrans de configuration (MeshPanel, TrafficPanel, AlgoPanel), et écran de résultats avec tableau et graphiques comparatifs (ResultsPanel).

Cette séparation entre données, logique métier et interface graphique correspond à une pratique reconnue en génie logiciel : elle permet notamment de tester et de faire évoluer le moteur de simulation (paquetage algo) indépendamment de l'interface utilisateur.

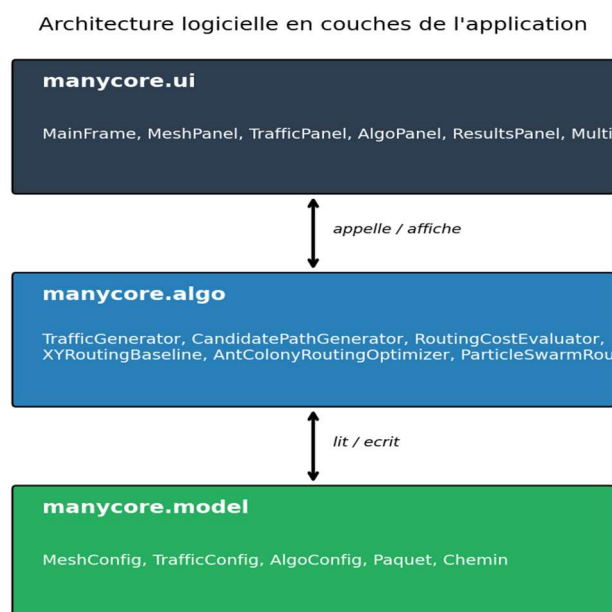


Figure III.1 — Architecture logicielle de l'application (packages model / algo / ui)

III.4 Modélisation du problème de routage

Le réseau est modélisé comme une grille bidimensionnelle de dimensions configurables (largeur × hauteur), chaque nœud étant identifié par des coordonnées

entières (x, y) . Le trafic est représenté par un ensemble de paquets, chacun caractérisé par un nœud source et un nœud destination, générés selon un motif uniforme (destinations réparties aléatoirement sur l'ensemble du réseau) ou hotspot (une majorité des paquets convergent vers un nœud central, simulant par exemple un contrôleur mémoire partagé fortement sollicité).

Pour chaque paquet, un ensemble de chemins minimaux candidats est généré : le chemin XY (déplacement complet selon l'axe X puis selon l'axe Y), le chemin YX (ordre inverse), ainsi que des chemins « en escalier » alternant aléatoirement les déplacements en X et en Y. Tous ces chemins candidats ont, par construction, la même longueur (le nombre de sauts correspondant à la distance de Manhattan entre source et destination) ; ils ne diffèrent donc que par l'ensemble des liens physiques empruntés, ce qui permet d'isoler l'effet de la répartition de charge (congestion) de celui de la longueur du chemin.

Le problème d'optimisation consiste alors à choisir, pour chaque paquet, lequel de ses chemins candidats emprunter, de manière à minimiser une fonction de coût globale définie comme la somme de trois termes : la latence totale (proportionnelle au nombre total de sauts), l'énergie totale consommée (également proportionnelle au nombre de sauts), et une pénalité de congestion appliquée à chaque lien emprunté par plus d'un paquet simultanément. C'est cette fonction de coût qui sert de fonction objective aux trois métaheuristiques comparées.

III.5 Le routage XY comme référence

Le routage XY, utilisé comme référence de comparaison, correspond simplement au choix systématique du premier chemin candidat (indice 0, qui est toujours le chemin XY par construction du générateur de chemins) pour l'ensemble des paquets, sans aucune optimisation ni prise en compte de la congestion.

III.6 Implémentation d'ACO

L'algorithme ACO maintient une matrice de phéromone associant à chaque couple (paquet, chemin candidat) une valeur de désirabilité. À chaque itération, un ensemble de fourmis construit une solution complète en choisissant, pour chaque paquet, un chemin par tirage aléatoire pondéré par la phéromone et par une heuristique locale favorisant les chemins courts. Après évaluation du coût de chaque solution, la phéromone est évaporée

globalement puis renforcée par les meilleures fourmis de l'itération (stratégie élitiste), proportionnellement à l'inverse de leur coût.

III.7 Implémentation de PSO

L'algorithme PSO représente chaque particule par un vecteur de nombres réels, de dimension égale au nombre de paquets à router, chaque coordonnée étant ensuite arrondie et bornée pour désigner l'indice du chemin candidat choisi. La mise à jour de la vitesse de chaque particule suit la formule standard combinant inertie, attraction vers la meilleure position personnelle et attraction vers la meilleure position globale de l'essaim.

III.8 Implémentation de GA

L'algorithme génétique fait évoluer une population de chromosomes, chacun étant un vecteur d'entiers représentant l'affectation de chemins pour l'ensemble des paquets. La sélection des parents s'effectue par tournoi, le croisement est de type uniforme (chaque gène de l'enfant est hérité aléatoirement de l'un des deux parents), et la mutation réaffecte aléatoirement un gène avec une probabilité paramétrable. Une stratégie d'élitisme garantit que le meilleur individu trouvé est systématiquement conservé d'une génération à l'autre.

III.9 Interface utilisateur

L'interface graphique de l'application reprend une navigation par onglets successifs, guidant l'utilisateur à travers les différentes étapes de configuration avant le lancement de la simulation.

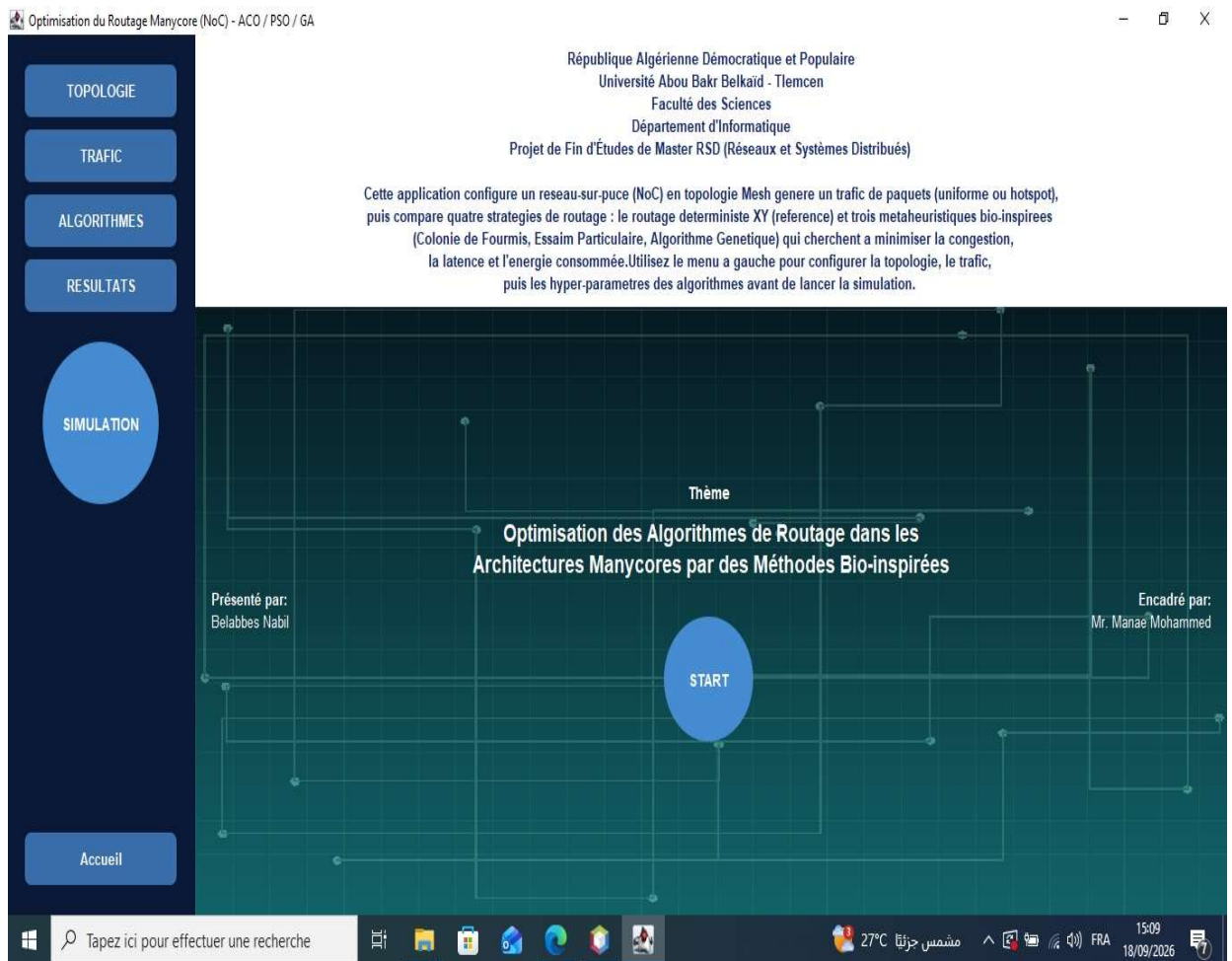


Figure III.2 — Écran d'accueil de l'application

L'écran de topologie permet de configurer les dimensions du maillage ainsi que les coûts unitaires de latence, d'énergie et de congestion par saut.

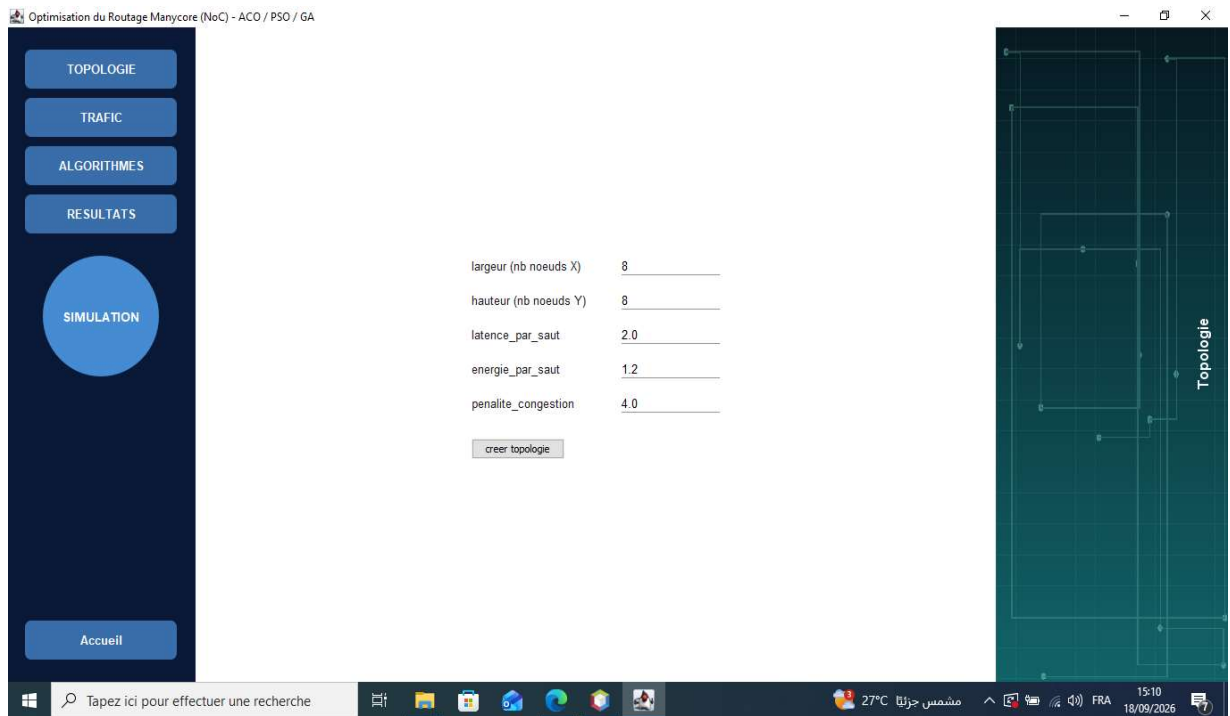


Figure III.3 — Écran de configuration de la topologie (Mesh)

L'écran de trafic permet de définir le nombre de paquets à générer, le motif de trafic (uniforme ou hotspot), ainsi que le nombre de chemins candidats générés pour chaque paquet.

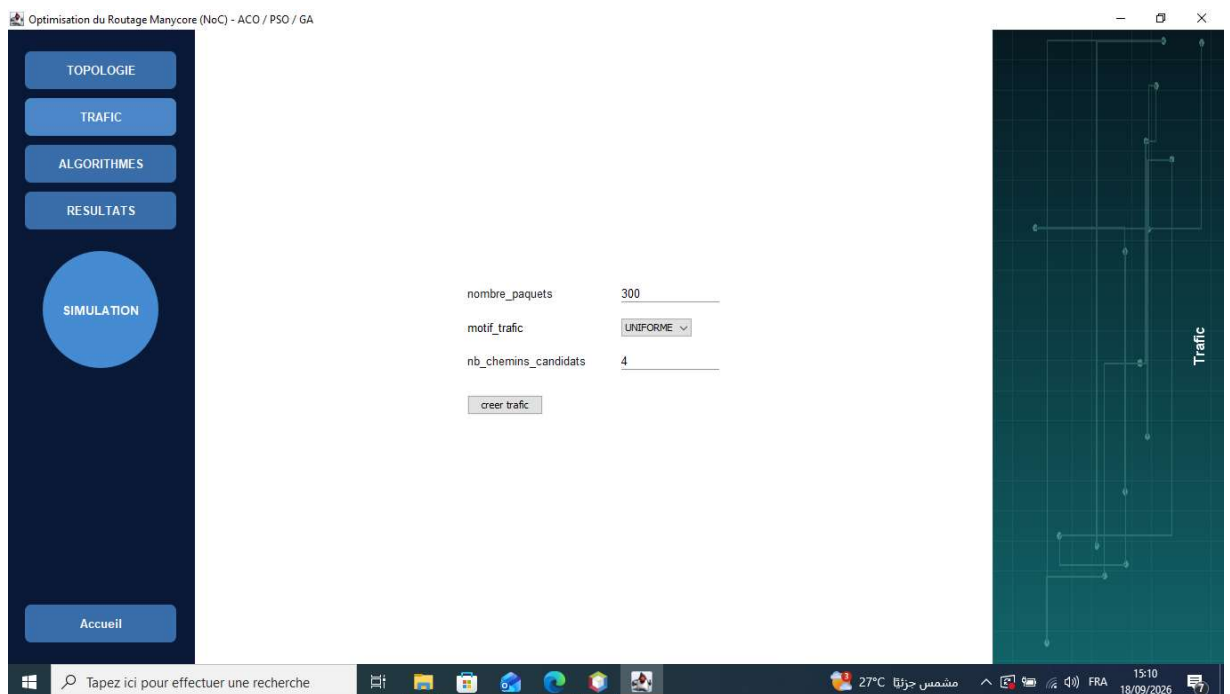


Figure III.4 — Écran de configuration du trafic

L'écran des algorithmes regroupe les hyper-paramètres communs (nombre d'itérations, taille de la population) ainsi que les paramètres spécifiques à chaque métaheuristique. C'est depuis cet écran que la simulation comparative est lancée.

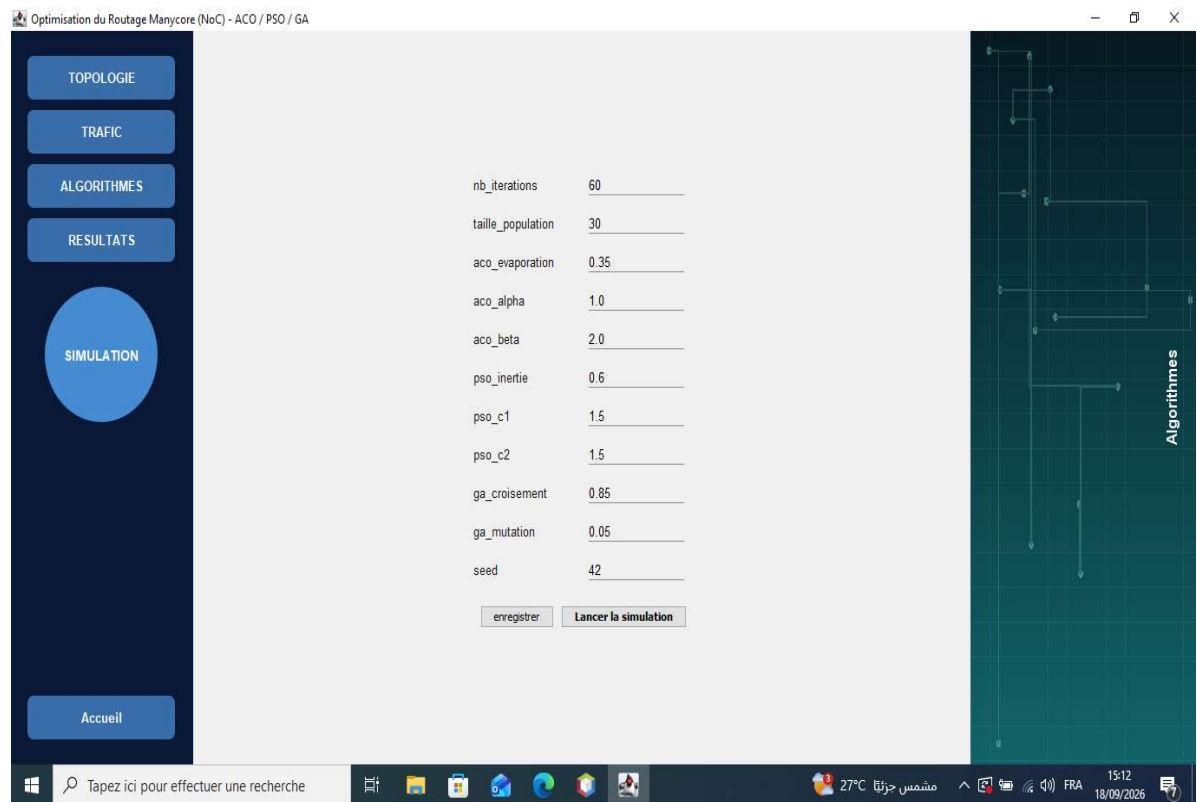


Figure III.5 — Écran de configuration des hyper-paramètres des algorithmes

Le champ seed permet de fixer la graine du générateur de nombres aléatoires, et les boutons « enregistrer » et « Lancer la simulation » valident les paramètres puis déclenchent l'exécution comparative.

À l'issue de la simulation, l'écran de résultats affiche un tableau comparatif des quatre approches (coût total, latence totale, énergie totale, congestion totale, nombre total de sauts et congestion maximale par lien), accompagné de graphiques en barres pour le coût total, la latence, l'énergie et la congestion. Des exemples de cet écran sont présentés au chapitre IV (Figures IV.2 et IV.3).

III.10 Conclusion

Ce chapitre a présenté l'architecture logicielle et les choix d'implémentation de l'application développée pour ce mémoire. La séparation claire entre modélisation, logique d'optimisation et interface utilisateur, ainsi que la définition d'une fonction de coût commune aux quatre stratégies comparées, permettent une évaluation équitable des

trois métaheuristiques bio-inspirées face au routage XY de référence. Le chapitre suivant présente le protocole expérimental mis en œuvre ainsi que les résultats obtenus.

Chapitre IV : Résultats Expérimentaux et Analyse Comparative

IV.1 Introduction

Ce chapitre présente le protocole expérimental mis en place pour comparer le routage XY de référence aux trois métaheuristiques bio-inspirées (ACO, PSO, GA), ainsi que les résultats obtenus et leur analyse. Une attention particulière est portée à l'interprétation honnête des résultats, y compris lorsque ceux-ci ne confirment que partiellement l'hypothèse d'une amélioration significative apportée par les métaheuristiques. Trois scénarios en trafic uniforme sont présentés : un scénario de référence sur un maillage 8×8 , puis deux scénarios complémentaires sur des maillages plus grands (16×8 et 16×16).

IV.2 Protocole expérimental

Le scénario de référence présenté dans cette section correspond à la configuration par défaut de l'application, avec un trafic uniforme (destinations réparties aléatoirement sur l'ensemble du réseau) :

Paramètre	Valeur
Dimensions du maillage	8×8 (64 nœuds)
Motif de trafic	Uniforme
Nombre de paquets	300
Nombre de chemins candidats par paquet	4
Nombre d'itérations (ACO / PSO / GA)	60
Taille de population / essaim / colonie	30

IV.3 Résultats obtenus

Le tableau suivant présente les résultats bruts obtenus pour les quatre approches comparées, sur une exécution du scénario décrit ci-dessus. Le coût total constitue la fonction objective minimisée par les trois métaheuristiques ; il correspond à la somme de la latence totale, de l'énergie totale et de la pénalité de congestion totale.

Approche	Coût total	Latence totale	Énergie totale	Congestion totale	Sauts totaux	Congestion max/lien
XY (référence)	10528,0	3170,0	1902,0	5456,0	1585	17
ACO	10516,0	3170,0	1902,0	5444,0	1585	15
PSO	10516,0	3170,0	1902,0	5444,0	1585	17
GA	10516,0	3170,0	1902,0	5444,0	1585	19



Figure IV.1 — Résultats comparatifs XY / ACO / PSO / GA (trafic uniforme)

Afin d'observer le comportement de l'application sur des réseaux de plus grande taille, deux scénarios complémentaires ont été exécutés en trafic uniforme. Les paramètres de chacun d'eux sont regroupés dans un tableau, et l'écran de résultats de l'application est présenté en capture d'écran.

IV.3.1 Scénario complémentaire A : maillage 16×8

Le premier scénario porte sur un maillage de 16×8 nœuds (128 nœuds), avec 500 paquets en trafic uniforme. Ses paramètres sont regroupés dans le tableau IV.3 ; les résultats obtenus figurent dans le tableau IV.4 et sont illustrés par la capture de la figure IV.2.

Tableau IV.3 — Paramètres du scénario A

Paramètre	Valeur
Dimensions du maillage	16 × 8 (128 nœuds)
Latence par saut	2,0
Énergie par saut	1,2
Pénalité de congestion	4,0
Motif de trafic	Uniforme
Nombre de paquets	500
Nombre de chemins candidats par paquet	20
Nombre d'itérations (ACO / PSO / GA)	30
Taille de population / essaim / colonie	100
ACO : évaporation ; α ; β	0,35 ; 1,0 ; 2,0
PSO : inertie ; c_1 ; c_2	0,6 ; 1,5 ; 1,5
GA : croisement ; mutation	0,85 ; 0,05
Graine aléatoire (seed)	30

Tableau IV.4 — Résultats comparatifs du scénario A (maillage 16 × 8, 500 paquets)

Approche	Coût total	Latence totale	Énergie totale	Congestion totale	Sauts totaux	Congestion max/liens
XY (référence)	25895,2	7702,0	4621,2	13572,0	3851	21
ACO	25871,2	7702,0	4621,2	13548,0	3851	29
PSO	25871,2	7702,0	4621,2	13548,0	3851	26
GA	25871,2	7702,0	4621,2	13548,0	3851	24



Figure IV.2 — Scénario A : écran de résultats de la simulation

Comme dans le scénario de référence, la latence totale (7702,0), l'énergie totale (4621,2) et le nombre total de sauts (3851) sont identiques pour les quatre approches (voir section IV.4.1). Les trois métaheuristiques obtiennent le même coût total, 25871,2, contre 25895,2 pour le routage XY, soit un gain de 24,0 unités (environ 0,09 %), qui provient uniquement de la baisse de la congestion totale (13548,0 contre 13572,0). En revanche, la congestion maximale sur un lien unique est plus élevée avec les métaheuristiques (29 pour ACO, 26 pour PSO et 24 pour GA) qu'avec le routage XY (21), ce qui est cohérent

avec le compromis entre congestion totale et congestion de pointe analysé à la section IV.4.3.

IV.3.2 Scénario complémentaire B : maillage 16×16

Le second scénario porte sur un maillage plus grand de 16×16 nœuds (256 nœuds), avec 400 paquets en trafic uniforme. Ses paramètres figurent dans le tableau IV.5. Il convient de noter que plusieurs de ces valeurs (évaporation, inertie, taux de croisement et de mutation) dépassent les plages habituellement retenues pour ces paramètres, notamment l'intervalle $[0, 1]$ pour les taux. Ce scénario illustre donc le comportement de l'application avec des paramètres non standards, et non une configuration optimisée ; ses résultats doivent être interprétés avec prudence.

Tableau IV.5 — Paramètres du scénario B

Paramètre	Valeur
Dimensions du maillage	16×16 (256 nœuds)
Latence par saut	4
Énergie par saut	2
Pénalité de congestion	8,0
Motif de trafic	Uniforme
Nombre de paquets	400
Nombre de chemins candidats par paquet	8
Nombre d'itérations (ACO / PSO / GA)	50
Taille de population / essaim / colonie	50
ACO : évaporation ; α ; β	2 ; 2 ; 4
PSO : inertie ; c_1 ; c_2	2 ; 3 ; 3
GA : croisement ; mutation	2 ; 3
Graine aléatoire (seed)	20

Les résultats sont présentés dans le tableau IV.6 et la figure IV.3.

Tableau IV.6 — Résultats comparatifs du scénario B (maillage 16×16 , 400 paquets)

Approche	Coût total	Latence totale	Énergie totale	Congestion totale	Sauts totaux	Congestion max/lien
XY (référence)	52430,0	17076,0	8538,0	26816,0	4269	14
ACO	52294,0	17076,0	8538,0	26680,0	4269	15
PSO	52318,0	17076,0	8538,0	26704,0	4269	15
GA	52286,0	17076,0	8538,0	26672,0	4269	16



Figure IV.3 — Scénario B : écran de résultats de la simulation

La latence totale (17076,0), l'énergie totale (8538,0) et le nombre de sauts (4269) restent identiques pour les quatre approches. Les trois métaheuristiques améliorent le coût total par rapport au routage XY (52430,0) : GA obtient la meilleure valeur (52286,0, soit un gain de 144 unités, environ 0,27 %), suivi d'ACO (52294,0, environ 0,26 %) puis de PSO (52318,0, environ 0,21 %). Contrairement au scénario de référence et au scénario A, les trois méthodes n'aboutissent donc pas ici à la même valeur du coût total. La congestion maximale sur un lien passe de 14 (XY) à 15 pour ACO et PSO et à 16 pour GA, ce qui confirme que la réduction de la congestion totale ne s'accompagne pas d'une amélioration du pire cas.

IV.4 Analyse et discussion

IV.4.1 Latence et énergie : un résultat attendu

Comme le montre le tableau IV.2, la latence totale (3170,0 cycles) et l'énergie totale (1902,0 unités) sont rigoureusement identiques pour les quatre approches. Ce résultat n'est pas fortuit : il découle directement de la construction des chemins candidats, tous de longueur minimale égale à la distance de Manhattan entre source et destination. Le choix du chemin parmi les candidats ne modifie donc jamais le nombre de sauts total, et n'a d'effet que sur la répartition de la charge entre les liens, c'est-à-dire sur la congestion. Ce constat confirme que le modèle isole correctement l'effet recherché : l'apport des métaheuristiques ne peut se manifester, dans ce protocole, que sur le critère de congestion.

IV.4.2 Un gain modeste mais réel sur la congestion

Sur le critère de congestion totale, les trois métaheuristiques obtiennent un résultat identique entre elles (5444,0) et inférieur à celui du routage XY (5456,0), soit un gain de 12 unités. Rapporté au coût total, ce gain représente une amélioration d'environ 0,11 % par rapport au routage XY (0,22 % sur la seule congestion). Ce résultat, bien que réel, reste modeste : en trafic uniforme, la charge est déjà naturellement bien répartie par le routage XY sur une topologie Mesh symétrique, ce qui laisse peu de marge de progression à une optimisation fondée uniquement sur le choix du chemin.

Ce constat se confirme sur les scénarios complémentaires présentés à la section IV.3 : le gain sur le coût total est d'environ 0,09 % dans le scénario A (de 25895,2 à 25871,2) et compris entre 0,21 % (PSO) et 0,27 % (GA) dans le scénario B (de 52430,0 à 52318,0 pour PSO et à 52286,0 pour GA). Le tableau IV.7 récapitule les gains obtenus sur les trois scénarios.

Tableau IV.7 — Gains sur le coût total par rapport au routage XY pour les trois scénarios

Scénario	Coût XY	Coût ACO (gain)	Coût PSO (gain)	Coût GA (gain)
Référence (8 × 8, 300 paquets)	10528,0	10516,0 (0,11 %)	10516,0 (0,11 %)	10516,0 (0,11 %)
A (16 × 8, 500 paquets)	25895,2	25871,2 (0,09 %)	25871,2 (0,09 %)	25871,2 (0,09 %)
B (16 × 16, 400 paquets)	52430,0	52294,0 (0,26 %)	52318,0 (0,21 %)	52286,0 (0,27 %)

IV.4.3 Un compromis intéressant sur la congestion de pointe

Un résultat plus nuancé apparaît sur la congestion maximale observée sur un lien unique : le routage XY affiche ici une valeur de référence de 17, et les trois métaheuristiques se comportent chacune différemment vis-à-vis de ce repère. ACO l'améliore nettement (15, soit deux unités de moins que XY), PSO l'égale exactement (17), tandis que GA la dégrade (19, soit deux unités de plus que XY). Ce résultat est particulièrement instructif : alors que les trois métaheuristiques obtiennent un coût total et une congestion totale rigoureusement identiques entre elles (voir section IV.4.4), elles se distinguent nettement sur la congestion de pointe. Autrement dit, réduire la congestion totale du réseau (somme sur tous les liens) ne garantit ni une réduction ni une dégradation systématique du pire cas sur un lien unique, le comportement dépend ici de l'algorithme utilisé, et non uniquement de l'objectif optimisé. Ce compromis illustre une nuance importante : minimiser une métrique agrégée (la congestion totale) n'implique pas mécaniquement une amélioration du pire cas (la congestion de pointe), qui peut pourtant être le facteur limitant en pratique (un lien très sollicité pouvant devenir un point de latence critique, indépendamment de la moyenne du réseau). Une fonction de coût

intégrant explicitement une pénalité sur la congestion maximale, plutôt que sur la seule congestion totale, permettrait de traiter ce compromis directement et d'orienter la recherche vers des solutions comme celle trouvée par ACO dans cette exécution.

La même tendance s'observe dans les scénarios complémentaires : la congestion maximale passe de 21 (XY) à 29, 26 et 24 pour ACO, PSO et GA dans le scénario A, et de 14 à 15, 15 et 16 dans le scénario B. Le classement des trois algorithmes sur ce critère varie donc d'un scénario à l'autre, et aucun ne se dégage systématiquement comme le meilleur sur la congestion de pointe.

IV.4.4 Convergence des trois métaheuristiques vers une solution équivalente

Un résultat notable est que les trois métaheuristiques, bien que reposant sur des mécanismes de recherche très différents (pistes de phéromone pour ACO, dynamique d'essaim pour PSO, sélection et croisement pour GA), convergent ici vers un coût total strictement identique (10516,0) et une congestion totale identique (5444,0). Ce résultat suggère que, pour ce scénario, les trois approches parviennent à identifier des solutions de même qualité agrégée, proche de l'optimum atteignable compte tenu de la faible marge de progression disponible en trafic uniforme, un indicateur positif de leur capacité à converger de façon cohérente entre elles sur le critère qu'elles optimisent explicitement. Il ne s'agit toutefois pas nécessairement de la même solution au sens strict : la congestion de pointe, qui n'entre dans la fonction de coût qu'indirectement, diffère selon l'algorithme (15 pour ACO, 17 pour PSO, 19 pour GA, voir section IV.4.3). Trois affectations de chemins distinctes peuvent donc produire un coût total et une congestion totale identiques tout en répartissant différemment la charge de pointe sur le réseau signe que l'espace de recherche comporte, pour ce scénario, plusieurs optima équivalents du point de vue de la fonction de coût agrégée, mais non du point de vue du pire cas.

Cette équivalence n'est toutefois pas observée dans tous les scénarios. Dans le scénario A, les trois méthodes obtiennent de nouveau le même coût total (25871,2), alors que dans le scénario B les valeurs diffèrent légèrement (52286,0 pour GA, 52294,0 pour ACO et 52318,0 pour PSO). Ce dernier scénario ayant été exécuté avec des paramètres situés hors des plages usuelles (section IV.3.2), cette différence ne permet pas de classer les algorithmes de façon définitive.

IV.5 Limites de l'étude

- Les résultats présentés dans ce chapitre proviennent d'une seule exécution de chacun des trois scénarios (référence, A et B). Les trois métaheuristiques étant stochastiques (elles reposent sur des tirages aléatoires), une conclusion statistiquement robuste nécessiterait de répéter l'expérience un nombre suffisant de fois (par exemple 20 à 30 exécutions) et de comparer les moyennes et écarts-types obtenus, plutôt qu'une exécution unique.
- Le modèle de coût utilisé est une simplification pédagogique : il ne simule pas de manière cycle-accurate le comportement d'un routeur réel (occupation des tampons, arbitrage, contrôle de flux), contrairement à des simulateurs spécialisés tels que Noxim.
- Les hyper-paramètres retenus pour le scénario de référence (60 itérations, population de 30, valeurs par défaut de l'application) restent modestes au regard de la taille de l'espace de recherche (4 choix possibles par paquet, soit 4^N combinaisons pour N paquets) une étude de sensibilité plus poussée permettrait d'évaluer l'effet d'un budget de calcul plus important.
- Le scénario B a été exécuté avec des paramètres (évaporation, inertie, taux de croisement et de mutation) situés hors des plages usuelles ; ses résultats ont une valeur illustrative et devraient être confirmés avec des paramètres standards.
- Seul le motif de trafic uniforme a été détaillé dans ce chapitre ; un trafic hotspot (destinations concentrées sur un nœud), exploré au cours du développement de ce travail, fait apparaître une dynamique différente, la congestion s'y concentre près des liens d'entrée du nœud ciblé et mériterait d'être documenté de façon comparative dans une étude complémentaire.

IV.6 Perspectives d'amélioration

- Répéter chaque configuration expérimentale sur un nombre suffisant d'exécutions indépendantes et présenter les résultats sous forme de moyennes avec intervalles de confiance.
- Comparer systématiquement le trafic uniforme et le trafic hotspot afin de caractériser dans quelles conditions les métaheuristiques bio-inspirées apportent le gain le plus significatif.

- Intégrer explicitement la congestion de pointe (et pas seulement la congestion totale) dans la fonction de coût, pour traiter directement le compromis identifié à la section IV.4.3.
- Introduire la notion de canaux virtuels, qui permettent à un même lien physique de transporter simultanément plusieurs flux logiques, réduisant ainsi la contention sur les liens les plus sollicités.
- Comparer les résultats obtenus avec ce modèle simplifié à ceux d'une simulation cycle-accurate réalisée avec Noxim, afin de valider la pertinence du modèle de coût utilisé.

IV.7 Conclusion

Ce chapitre a présenté et analysé les résultats expérimentaux obtenus en comparant le routage XY de référence aux trois métaheuristiques bio-inspirées étudiées dans ce mémoire, sous un trafic uniforme. Si le gain global observé reste modeste (de l'ordre de 0,09 % à 0,27 % sur le coût total selon le scénario), l'analyse a mis en évidence un compromis intéressant entre congestion totale et congestion de pointe, ainsi qu'une convergence des trois métaheuristiques vers des coûts identiques ou très proches. Ce chapitre illustre l'importance d'une analyse critique et rigoureuse des résultats expérimentaux, y compris lorsqu'ils ne confirment que partiellement les hypothèses de départ, ce qui constitue une démarche scientifique à part entière.

Conclusion Générale

Ce mémoire s'est intéressé à l'optimisation du routage dans les architectures Manycores au moyen de méthodes bio-inspirées. Après avoir présenté le contexte technologique des réseaux sur puce et les limites du routage déterministe classique, trois métaheuristiques : Colonie de Fourmis, Essaim Particulaire et Algorithme Génétique ont été étudiées, implémentées et appliquées au problème du choix de chemins de routage dans une topologie Mesh.

La conception et le développement d'une application logicielle dédiée ont permis de modéliser le réseau, de générer différents profils de trafic, et de comparer expérimentalement les performances des trois métaheuristiques à celles du routage XY de référence. Les résultats obtenus, analysés de façon rigoureuse au chapitre IV, montrent que les trois métaheuristiques améliorent légèrement le coût total par rapport au routage XY sous trafic uniforme (de 0,09 % à 0,27 % selon les trois scénarios étudiés), avec des coûts identiques ou très proches entre elles, avec un compromis notable entre la réduction de la congestion totale et une congestion de pointe légèrement supérieure sur certains liens.

Ce travail met ainsi en évidence à la fois le potentiel des méthodes bio-inspirées pour l'optimisation du routage en particulier leur capacité à converger vers des solutions cohérentes malgré des mécanismes de recherche très différents et les limites intrinsèques de toute approche fondée uniquement sur la sélection de chemins, notamment le compromis entre réduction de la congestion totale et maîtrise de la congestion de pointe. Plusieurs pistes ont été identifiées pour prolonger ce travail, notamment l'introduction du Multipath et des canaux virtuels, une validation sur un simulateur cycle-accurate tel que Noxim, une comparaison systématique entre trafic uniforme et trafic hotspot, ainsi qu'une évaluation statistique plus robuste reposant sur plusieurs exécutions indépendantes.

Au-delà des résultats chiffrés, ce mémoire aura permis de mettre en pratique une démarche complète d'ingénierie logicielle et de recherche expérimentale : depuis l'étude bibliographique et la modélisation d'un problème d'optimisation, jusqu'à l'implémentation, l'expérimentation et l'analyse critique des résultats obtenus.

Références Bibliographiques

- [1] W. J. Dally, B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann, 2004.
- [2] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, 2004.
- [3] J. Kennedy, R. Eberhart, « Particle Swarm Optimization », Proceedings of IEEE International Conference on Neural Networks, 1995.
- [4] J. H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.
- [5] L. Benini, G. De Micheli, « Networks on Chips: A New SoC Paradigm », IEEE Computer, 2002.
- [6] Documentation et code source de l'application développée dans le cadre de ce mémoire (voir annexes).

ملخص – Abstract – Résumé

Résumé

Les architectures Manycores reposent sur des réseaux sur puce (NoC) dont les performances dépendent fortement de l'algorithme de routage. Ce mémoire étudie l'apport de trois métaheuristiques bio-inspirées (ACO, PSO, GA) à l'optimisation du routage dans une topologie Mesh 2D, par rapport au routage XY de référence. Une application Java a été développée pour comparer la latence, le coût énergétique et la congestion. Sur trois scénarios en trafic uniforme (maillages 8×8 , 16×8 et 16×16), les trois méthodes améliorent légèrement le coût total par rapport à XY, avec des gains de 0,11 %, 0,09 % et de 0,21 % à 0,27 % respectivement, et un compromis notable entre congestion totale et congestion de pointe.

Mots-clés : *Manycore, Réseau sur Puce (NoC), routage adaptatif, métaheuristiques, ACO, PSO, GA, congestion.*

Abstract

Manycore architectures rely on Networks-on-Chip (NoC), whose performance strongly depends on the routing algorithm. This thesis studies the contribution of three bio-inspired metaheuristics (ACO, PSO, GA) to routing optimization in a 2D Mesh topology, compared with the baseline XY routing. A Java application was developed to compare latency, energy cost and congestion. On three uniform-traffic scenarios (8×8 , 16×8 and 16×16 meshes), the three methods slightly improve the total cost compared with XY, with gains of 0.11%, 0.09% and 0.21% to 0.27% respectively, and a notable trade-off between total congestion and peak congestion.

Keywords: *Manycore, Network-on-Chip (NoC), adaptive routing, metaheuristics, ACO, PSO, GA, congestion.*

ملخص

تعتمد البنى متعددة الأنوية (Manycores) على الشبكات داخل الشريحة (NoC)، ويؤثر اختيار خوارزمية التوجيه بشكل كبير على أدائها. تدرس هذه المذكرة إسهام ثلاث خوارزميات مستوحاة من الطبيعة: مستعمرة النمل (ACO) وحشد الجسيمات (PSO) والخوارزمية الجينية (GA) في تحسين التوجيه ضمن شبكة مشبكية ثنائية الأبعاد، مقارنةً بالتوجيه المرجعي (XY). تم تطوير تطبيق بلغة جافا لمقارنة زمن الانتقال وتكلفة الطاقة والازدحام. أظهرت النتائج، في ثلاثة سيناريوهات بحركة مرور منتظمة (شبكات 8×8 و 16×8 و 16×16)، أن الخوارزميات الثلاث تحسن التكلفة الإجمالية قليلاً مقارنةً بالتوجيه (XY)، بنسب 0.11% و 0.09% و 0.21% ثم من 0.21% إلى 0.27% على التوالي، مع مقايضة ملحوظة بين الازدحام الإجمالي وازدحام الذروة.

الكلمات المفتاحية: متعدد الأنوية، الشبكة داخل الشريحة، التوجيه المتكيف، الخوارزميات الحدية، مستعمرة النمل، حشد الجسيمات، الخوارزمية الجينية، الازدحام.