République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab Blida

Faculté des sciences

Département informatique

Mémoire de fin d'études

En vue d'obtenir le diplôme de master

Filière : Informatique

Spécialité : Informatique

Option : Ingénier de logiciel

THEME

Optimisation de la planification de l'auto-adaptation des systèmes par l'utilisation des métaheuristiques

Présenté par :

BOUMEDIENE Khalil ARIBI Dahmane

Soutenu le : -- Octobre 2019, devant le jury composé de :

	Président
	Examinateur
M^{me}. GUESSOUM D	Promotrice

Année universitaire : 2018/2019

Résumé :

Les systèmes modernes nécessitent des services d'adaptation dynamiques qui leur permettent d'adapter leur comportement et améliorer leur capacité à réagir de manière dynamique aux changements d'environnement.

Dans le cadre de ce projet, nous nous sommes intéressés à l'autoadaptation des applications mobiles. Nous avons proposé deux méthodes basées sur des métaheuristiques (algorithme génétique et essaimes particulaires). L'approche proposée permet d'adapter le comportement de l'application mobile en variant les fonctionnalités de cette dernière (wifi, GPS, Bluetooth, mode audio …) et en tenant compte de la consommation de batterie du téléphone. Différents tests ont été effectués sur une application mobile que nous avons développé afin de tester l'efficacité de l'approche proposée.

Mots clés : Autoadaptation, algorithme génétique, optimisation par essaims de particulaires.

Abstract:

Modern systems require dynamic adaptation services that allow them to adapt their behavior and improve their ability to respond dynamically to changing environments.

As part of this project, we were interested in the self-adaptation of mobile applications. We proposed two methods based on metaheuristics (genetic algorithm and particle swarm optimization). The proposed approach allows to adapt the behavior of the mobile application by varying their functionality (wifi, GPS, Bluetooth, audio mode ...) and taking into account the battery consumption of the phone. Various tests were performed on a mobile application that we developed to test the effectiveness of the proposed approach.

Key words: Self-adaptation, genetic algorithm, particle swarm optimization.

ملخص:

تتطلب الأنظمة الحديثة خدمات التكيف الديناميكي التي تتيح لها تكييف سلوكها وتحسين قدرتها على الاستجابة ديناميكيًا للبيئات المتغيرة.

كجزء من هذا المشروع، اهتممنا بالتكيف الذاتي للتطبيقات المحمولة. اقترحنا طريقتين مختلفتين تستندان على خوارزمية الأدلة العليا (الخوارزمية الجينية و خوارزمية تحسين سرب الجسيمات). يسمح النهج المقترح بتكييف سلوك تطبيقات الهاتف المحمول من خلال تغيير وظائفهم (واي فاي ، جي بي اس ،بلوتوث ، وضع الصوت ...) ومراعاة استهلاك بطارية الهاتف. تم إجراء اختبارات مختلفة على تطبيق الهاتف المحمول الذي طورناه لاختبار فعالية النهج المقترح.

الكلمات المفتاحية : التكيف الذاتي ، الخوارزمية الجينية ، تحسين سرب الجسيمات.

Dédicaces

Je dédie ce modeste travail

A mes chers parents qui m'ont tout donné durant ma vie, et tous les mots de l'univers ne suffisent pas pour les remercier

A mes frères Ahmed et Kader.

A mes seours .

pour leur Soutien et encouragement.

A mon binôme Khalil

A mes amis Mouafak ,Krimo ,djamel ,Fouad

A mes collegues Imad ,Rafik , Rim

A mes enseignants Mr chikhi et tout les enseignant

A tout qu'ils sont proche de moi

A toute la promotion 2019-2020

Dahmane

Dédicaces

Je dédie ce modeste travail

A mes chers parents qui m'ont tout donné durant ma vie, et tous les mots de l'univers ne suffisent pas pour les remercier

A mes frères Abdelbaki,Mohamed ,Alaa,Issam.

pour leur Soutien et encouragement.

A mon binôme DAHMANE

A mes amis Mohamed,Abdelkrim,Brahim,Oussama,Amine

A mes collegues Rafik,Tarek,Yacin ,Hani,

A tout qu'ils sont proche de moi

A toute la promotion 2019-2020

khalil

Remerciement

Nous remercions Dieu de nous avoir donnée patience et courage pour accomplir ce travail

Nous remercions nos parents pour leurs sacrifices et leur encouragement durant toute notre scolarité

Nous remercions surtout notre promotrice Mme D. Guessoum

pour son aide et soutient durant toute la période de travail,

et nous sommes très reconnaissant pour tout ce qu'il nous a fait

Nous tenons aussi à exprimer notre gratitude à nos profs

qui nous ont formés durant notre cycle universitaire,

ainsi que tout le staff du département

d'informatique

Sans oublier tous nos amis et camarades qui font toujours notre bonheur

Aux membres de jury qui nous ont fait l'honneur d'accepter de juger ce modeste travail

Enfin, nous espérons que ce mémoire servira d'exemple et de support pour les années à venir

Abréviation :

AG	***Algorithme Génétique***
PSO	***Particul Swarm Optimization***
UML	***Unified Modeling Language***
EA	***Evolutionary Algorithm***
GUI	***Graphical User Interface***
SI	***Swarm Intelligence***
SOA	***Architecture orientée services***
Gbest	***Global best***
Pbest	***Personal best***
CVL	***Common Variability Language***
CMS	***Contexte monitoring service***
DRS	***Dynamique reconfiguration service***
MAPE-k	***Monitoring, Analyses, Plans, Execution, knowledge***
AS	***Ante System***
PVC	***Problème du Voyageur de Commerce***

Table des matiers :

Liste des figures :

Liste des tableaux :

Introduction générale :

Le contexte d'exécution dans lequel les systèmes modernes s'exécutent change continuellement. Par conséquent, les applications de ces systèmes nécessitent une prise en charge pour l'auto-adaptation aux changements continus .Souvent des approches conçues pour les systèmes auto-adaptatifs implémentent un service d'adaptation qui reçoit en entrée la liste de toutes les configurations possibles et les plans pour basculer entre elles.

Problématique :

Cependant, Les applications mobiles nécessitent des services d'adaptation dynamiques qui leur permettent d'adapter leur comportement aux changements de contexte continus qui se produisent dans leur environnement. Ces services d'adaptation doivent faire face à la grande variabilité des configurations possibles qui correspondent aux différents contextes dynamiques.

D'autre part, les applications mobiles s'exécutant sur des appareils légers disposant de ressources limitées (batterie, mémoire, processeur, etc.), elles doivent donc adapter leurs fonctionnalités aux variations constantes des ressources et aux besoins de l'utilisateur. Idéalement, cette optimisation devrait être gérée de manière autonome par l'application elle-même, qui devrait pouvoir optimiser elle-même son fonctionnement.

Un défi important que doit relever tout service exécuté dans un environnement mobile consiste à réduire autant que possible les ressources utilisées par le service lui-même. En particulier, pour un service d'adaptation, le temps est critique car, pour être utiles, les applications doivent être adaptées sans que le temps supplémentaire employé pour le processus d'adaptation ne soit noté.

Objectifs

Il est question dans ce PFE d'implémenter une approche alternative pour la génération automatique de plans d'adaptation au moment de l'exécution. L'approche proposée devra générer des plans d'adaptation optimales en ce qui concerne différents critères, tels que la fonctionnalité ou la consommation de ressources (par exemple, le taux de batterie ou la mémoire). Ceci devra être réalisé à l'aide de métaheurstiques (GNA et PSO) qui tentent de trouver des configurations presque optimales à l'exécution. L'approche proposée devra être validée par une étude de cas pour montrer que l'approche peut être efficace et adaptée aux

appareils disposant de ressources limitées. L'étude de cas est une application mobile de gestion des évènements du département d'Informatique.

La suite du mémoire est organisée en quatre chapitres :

Le premier chapitre *: Généralité sur les systèmes auto adaptatifs* : dans ce chapitre nous présenteront les concepts de base de l'auto adaptation des systèmes.

Le deuxième chapitre : *Généralité sur les métaheuristiques*: ce chapitre sera consacré à l'étude de quelques métaheuristiques, leur classification et leurs caractéristiques ainsi que l'application de ces derniers sur l'auto adaptation.

Le troisième chapitre : *Conception et modélisation de l'autoadaptation basé sur deux métaheurstiques :* dans ce chapitre nous présentons la modélisation de notre étude de cas ainsi que la conception de notre solution basée sur l'algorithme génétique et essaimes particulaires.

Le quatrième chapitre : *Implémentation de l'application et tests :* dans ce dernier chapitre nous présentons les outils utilisés pour développer notre approche et les différents tests et résultats obtenus en appliquant notre approche.

Chapitre *I* : Généralité sur les systèmes auto adaptatif

1. Introduction

De nos jours, les systèmes modernes sont caractérisés par des changements fréquents de l'environnement dans lequel ils s'exécutent. Par conséquent, ces systèmes doivent changer leurs fonctionnalités et/ou leur configuration afin de suivre ces changements pour rester toujours fonctionnels, on parle alors des systèmes auto-adaptatifs. Ce chapitre sera consacré pour définir le contexte général des systèmes auto-adaptatifs.la dernière partie de ce chapitre sera consacré à présenter quelques travaux de recherche les plus significatifs qui ont traité l'auto adaptation des systèmes.

2 .Définition d'un système auto adaptative :

Les systèmes « auto-adaptatif » sont devenus un besoin et une réalité inévitable. Cependant, cette nouvelle branche du génie logiciel ne dispose pas encore de normes ou de standard, et il n'y a pas une définition ou vocabulaire commun **[1].**

Les chercheurs proposent donc une définition qui permet de cerner les limites de cette étude :

Un système auto adaptative est un système capable de modifier sa configuration ou sa structure interne pour prendre en compte les changements de son environnement et ainsi d'offrir un service d'une qualité optimale **[1].**

3. L'intérêt des systèmes auto adaptatif :

Les systèmes d'auto-adaptation permettent de suivre les changements fréquents qui se produisent dans leur environnement afin d'assurer une qualité de service optimal pour l'utilisateur final. Ces systèmes sont gérés de manière autonome. L'objectif inhérent aux systèmes adaptatifs est de maintenir la qualité du service rendu quel que soit l'état de l'environnement. Il s'agit de stabiliser la qualité de service **[1].** Trois capacités essentielles caractérisent un système adaptatif : l'observation, la décision et l'intro-action **[1].**

- **Observation :** Un système adaptatif est un système qui observe son environnement et en détecte les changements. D'un point de vue technique, cette capacité se traduit par l'existence de sondes (logicielles ou matérielles) permettant de mesurer les propriétés pertinentes de l'environnement **[1].**

- **Décision :** En fonction de l'état de l'environnement, mais également fonction de sa configuration actuelle (capacité d'introspection), un système adaptatif décide par lui-même de la nouvelle configuration à adopter pour fonctionner de manière optimale **[1].**
- **Intro-action :** Pour modifier sa propre configuration, le système adaptatif manipule et modifie les éléments qui le constituent. Ces « intro-actions » sont des actions de l'intérieur sur l'intérieur. Il s'agit d'une capacité d'introspection active **[1].**

4. Propriétés des systèmes auto-adaptatifs :

La figure ***I.1*** illustre les quatre propriétés des systèmes auto-adaptatifs (l'auto configuration, l'auto optimisation, l'auto protection et l'auto réparation).

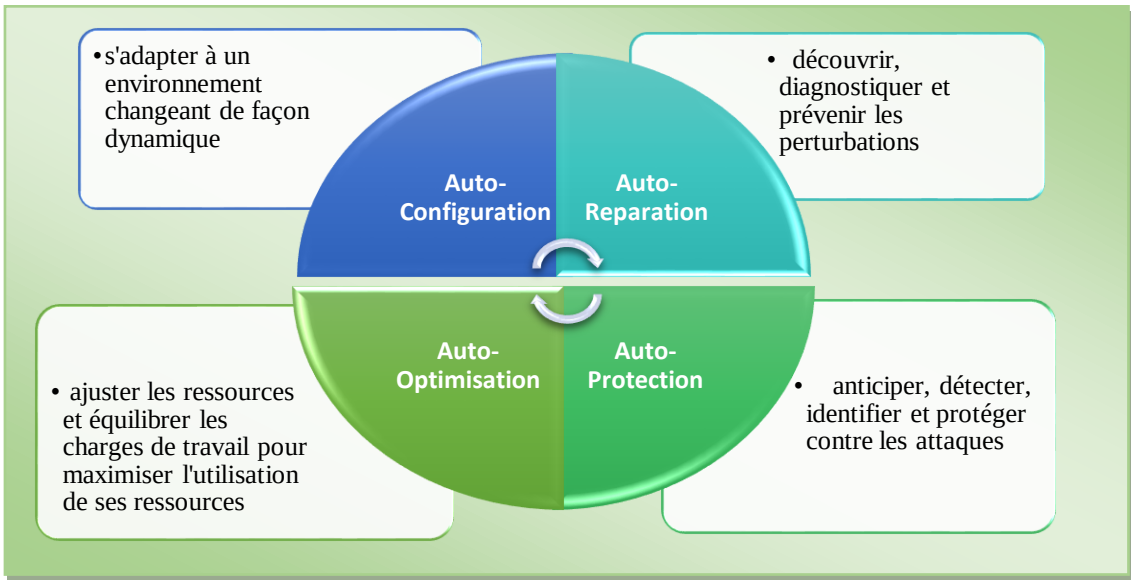

Figure I.1 **:** Propriétés des systèmes auto-adaptatifs. [22]

4.1. Auto-configuration et auto-reconfiguration :

A. Auto configuration :

Une Propriété qui permet l'adaptation transparente d'un nouveau composant ou d'un nouvel environnement d'exécution sans intervention humaine. Il peut s'adapter dynamiquement à des environnements changeants **[4]**. Ses modifications peuvent inclure le déploiement de nouveaux composants ou la suppression de composants existants, ou des modifications considérables des caractéristiques du système.

La configuration automatique est générique dans le sens où elle peut être utilisée pour réaliser toute autre propriété autonome, ou même tout objectif d'auto-adaptation réalisable par la reconfiguration automatique de l'architecture du système (la structure du système géré et le mécanisme d'auto-adaptation) **[5].**

B. Auto reconfiguration :

L'auto-reconfiguration est la capacité d'un système de passer de manière autonome d'une configuration à une autre pour répondre à une défaillance ou à un changement de contexte. Elle est considérée comme une fonctionnalité essentielle pour les futurs systèmes logiciels, car elle leurs permettent d'évoluer et de s'adapter à des environnements ouverts et dynamiques afin qu'ils puissent continuer à remplir leurs objectifs **[6].**

Les mécanismes d'auto-reconfiguration sont intégrés aux applications, leurs analyse et réutilisation est difficile. Une approche alternative est l'adaptation externalisée, dans laquelle les composants système sont utilisés au moment de l'exécution par un composant externe pour détecter et résoudre les problèmes du système.

4.2. Auto-réparation :

Lorsque les systèmes informatiques ne fonctionnent pas correctement, la détection et la résolution du problème nécessitent beaucoup de temps et d'efforts. Par conséquent, les logiciels doivent s'adapter sans intervention humaine pour parvenir à une capacité d'auto-réparation.

L'auto-réparation est un mécanisme permettant la détection et la réparation des erreurs, concerne toute appareil ou système capable de détecter qu'il ne fonctionne pas correctement et sans intervention humain, et de procéder aux ajustements nécessaires pour revenir à un fonctionnement correct.

Le système est capable de se reconfigurer en déployant de manière autonome les composants logiciels affecté par une panne matériel ou logiciel, l'auto réparation est la capacité de découvrir, diagnostiquer et de réagir aux perturbations. Il a un objectif principal de maximisation de la disponibilité, la capacité de survie, la maintenabilité et la fiabilité du système **[9].**

4.2.1. Principe d'autoréparation :

Pour l'autoréparation, l'objectif principal est de maximiser la disponibilité, la capacité de survie, la maintenabilité et la fiabilité du système. Il y a quatre aspects importants dans l'auto-guérison **[10]** :

- **Surveillance**: le système doit toujours être en surveillance pour qu'il puisse observer le comportement anormal de l'application.

- **Interprétation**: il est important de déterminer le moment où le système engendre un problème, ainsi que le mécanisme permettant de détecter si le système a été modifié de manière incorrecte.

- **Résolution**: C'est le mécanisme proposé pour la réparation du système.

- **L'adaptation ou la reconfiguration** : est la phase finale de l'autoréparation, elle consiste à mettre en œuvre le mécanisme de récupération des modules endommagés du système.

4.3. Autoprotection :

Les systèmes logiciels auto protecteurs représentent une classe des systèmes autonomes capables de détecter et d'atténuer les menaces de sécurité lors de l'exécution. Ils gagnent de l'importance, car les méthodes statiques rigides utilisées pour sécuriser les systèmes logiciels sont révélées inadaptées aux défis posés par les systèmes logiciels modernes. Bien que la recherche existante ait fait des progrès significatifs en matière de sécurité autonome et adaptative, des lacunes et des défis subsistent.

4.3.1. Définition de l'autoprotection :

La sécurité est le talon d'Achille de la plupart des systèmes logiciels modernes. En dépit des progrès importants accomplis au cours des dernières décennies, les problèmes de sécurité sont plus présents que jamais. Au fur et à mesure que les modèles de sécurité traditionnels, souvent statiques et rigides, prennent de plus en plus en compte les limites, la recherche se déplace

vers des modèles dynamiques, dans lesquels les menaces à la sécurité sont détectées et atténuées au moment de l'exécution, c'est-à-dire l'autoprotection **[11].**

L'autoprotection est étroitement liée aux autres propriétés, telles que l'auto-configuration et l'auto-optimisation. D'un côté, un système avec des propriétés d'auto-configuration et auto-optimisation s'appuie sur des fonctions d'auto-protection pour assurer son intégrité lors de changements dynamiques. D'autre part, la mise en œuvre des fonctions d'autoprotection peut également utiliser les mêmes techniques que celles utilisées pour la reconfiguration et l'adaptation du système.

4.3.2. Mécanismes de l'autoprotection :

L'autoprotection a été identifiée comme l'un des traits essentiels de l'autogestion pour les systèmes informatiques autonomes. **Kephart** and **Chess** a divisé les systèmes d 'autoprotection en deux catégories:

- **Réactif** : le système se défend automatiquement contre les attaques malveillantes ou les défaillances en cascade **[11]**.
- **Proactif** : le système anticipe les problèmes de sécurité et prendre les mesures nécessaires pour les atténuer [**11].**

4.4. Auto-optimisation :

4.4.1. L'objectif de l'optimisation :

L'objectif principal de l'optimisation est de maximiser ou minimiser une ou plusieurs fonctionnalités par rapport à un ensemble représentant souvent une gamme de choix disponibles dans une situation donnée. La fonction permet de comparer les différents choix pour déterminer lequel pourrait être **« le meilleur** ». (Applications de l'optimisation: coût minimal, profit maximal, erreur minimale, conception optimale…etc.) **[7].**

4.4.2. Principe de l'auto optimisation :

Les applications mobiles fonctionnent sur des dispositifs légers avec peine de ressource (par exemple la batterie, mémoire, CPU, etc.), de sorte qu'ils doivent adapter leur fonctionnalité aux variations des ressources continues (ex : variation de l'état de batterie), et aussi aux besoins de l'utilisateur (ex : envoyer une vidéo, texte ou image), ce principe de fonctionnement construit le principe de l'optimisation. Idéalement, cette optimisation doit être gérée de façon autonome par l'application elle-même sans intervention humain, qui devrait être en mesure d'auto-optimiser son fonctionnement **[8]**.

Lorsque la disponibilité des certaines ressources diminue ou augmente de manière significative, la situation idéale serait de pouvoir décider quelle configuration architecturale offre la meilleure fonctionnalité, tout en ne dépassant pas les ressources disponibles. Ainsi, les algorithmes rapides pour calculer la configuration lors de l'exécution optimum sont souhaitables. Puisque cela peut être simulé comme un problème lucratif d'optimisation, des algorithmes peuvent être utilisées pour optimiser la sélection des points de variation architecturaux qui seront conformes à la nouvelle configuration.

5. Présentation de quelques approches sur les systèmes auto-adaptatifs :

5.1. Approche MUSIC:

MUSIC**:** Prise en charge de middleware pour l'auto-adaptation dans des environnements omniprésents et orientés service [19]

5.1.1. Problématique de ce travail :

Avec l'amélioration continue des capacités des appareils, les applications mobiles deviennent non seulement sensibles au contexte, mais aussi auto-adaptatives. Cette nouvelle tendance des applications est capable d'ajuster leur comportement en fonction des changements observés dans l'environnement. Cependant, ces adaptations sont souvent limitées par les caractéristiques statistiques du middleware de support, qui ne permet pas d'exploiter les opportunités offertes par l'environnement pour améliorer la qualité des adaptations.

5.1.2. Objectif de ce travail :

Ce travail présente les principes de l'adaptation basée sur la planification et leur mise en œuvre dans la plate-forme MUSIC. Qui offre un support modulaire maximisant la satisfaction de l'utilisateur en adaptant de manière dynamique les applications mobiles. Au-delà de cette prise en charge, la plate-forme MUSIC peut elle-même être adaptée à l'environnement en évolution (par exemple, localisation, connectivité réseau) afin de satisfaire à la fois les exigences de l'utilisateur et les propriétés de l'appareil.

5.1.3. Aperçu sur le travail :

Pour soutenir les principes SOA (architecture orientée services) ils ont intégré de nouveaux composants à la plate-forme MUSIC qui est un middleware basé sur OSGi pour le développement d'applications adaptatives sensibles au contexte.

Il s'agit d'une approche basée sur les composants et axée sur les services, composée principalement de deux parties différentes: le contexte et les middlewares d'adaptation. Le middleware d'adaptation est responsable de l'adaptation des applications, en déployant la configuration la mieux adaptée au contexte actuel en évaluant une fonction d'utilitaire spécifiée par l'architecte logiciel.

Dans ce travail il ont étudié comment remplacer des services des applications mobiles automatiquement qu'il interrompe a un moment donnée, avec d'autre service plus performant, plus efficace et plus utile que le premier. et aussi comment observer le changement, replanifié , reconfigurer et exécuté ce nouveau service avec l'utilisation de l'approche MUSIC dans l'architecture orienté service .

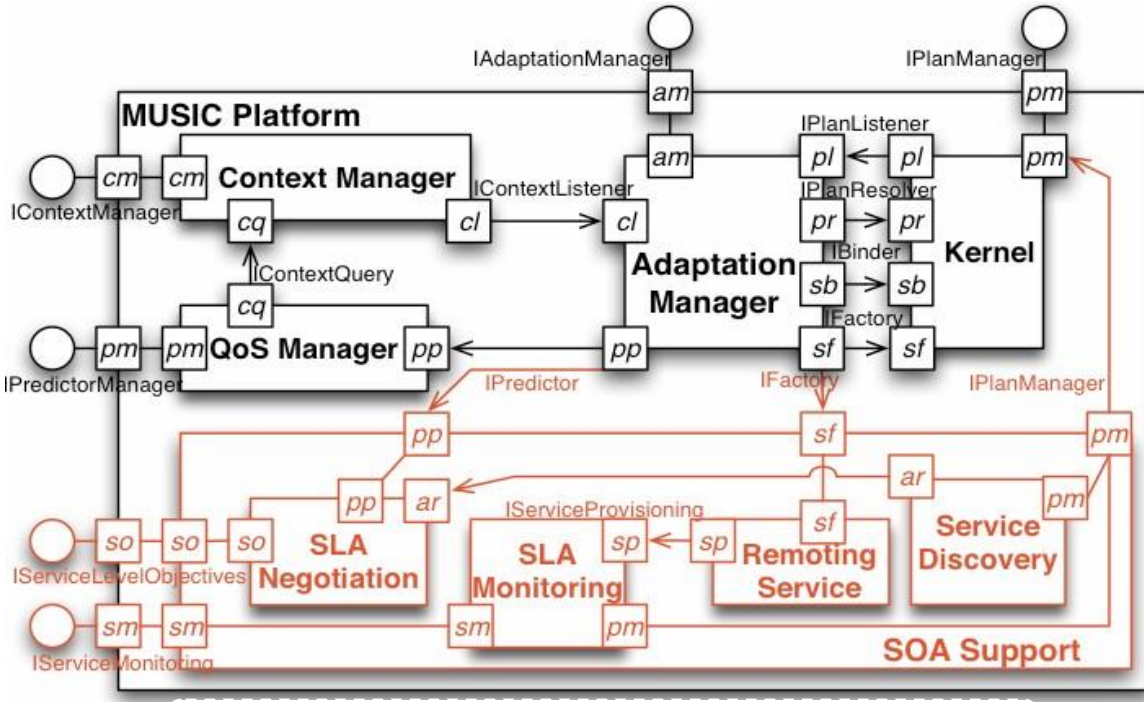

Figure I.2 : Configuration SOA de la plateforme MUSIC [19]

5.1.4. Résultat de ce travail :

Ce travail présente le rôle d'une approche MUSIC qui rend les applications adaptables automatiquement au moment d'un changement dans leur environnement, il remplace les services inutilisables par d'autres services utilisables et plus efficaces.

par exemple le service RATP Map de haute qualité a été remplacer par le service de carte commercial qui désormais l'utilitaire le plus performant

5.2. Approche RAINBOW

Améliorer l'auto-adaptation basée sur l'architecture Grâce à la prévision des ressources. [20]

5.2.1 .Problématique de ce travail :

Une préoccupation de plus en plus importante pour la conception de systèmes modernes est de savoir quelle est la meilleure façon d'intégrer l'auto-adaptation dans les systèmes afin d'améliorer leur capacité à réagir de manière dynamique aux erreurs, aux variations de ressources et aux besoins changeants des utilisateurs. Une approche prometteuse consiste à utiliser des modèles architecturaux comme base pour la surveillance, la détection des

problèmes et la sélection des réparations. Bien que cette approche ait donné des résultats positifs, les systèmes actuels utilisent une approche réactive: ils ne répondent aux problèmes que lorsqu'ils se présentent.

5.2.2. Objectif de ce travail :

L'objectif de Ce travail et de soutenue qu'il est possible d'améliorer l'auto-adaptation en adoptant une approche anticipative dans laquelle les prévisions sont utilisées pour éclairer les stratégies d'adaptation. Démontrer comment une telle approche peut être incorporée dans un cadre d'adaptation basé sur une architecture et leurs avantages.

5.2.3. Vue d'ensemble du cadre RAINBOW:

Cette approche auto-adaptative basée sur l'architecture est intégrée dans un cadre d'ingénierie appelé Rainbow, qui fournit des mécanismes pour surveiller un système cible et son environnement d'exécution, refléter les observations dans un modèle d'architecture, détecter les opportunités d'amélioration, sélectionner un cours d'action et changements d'effet.

La structure Rainbow (*Figure I.3*) utilise un modèle d'architecture composant / connecteur du système cible pour surveiller et raisonner sur les stratégies appropriées d'adaptation du système Les mécanismes de surveillance (sondes et jauges) observent le système cible en fonctionnement. Les observations sont signalées pour mettre à jour les propriétés du modèle d'architecture géré par le gestionnaire de modèles

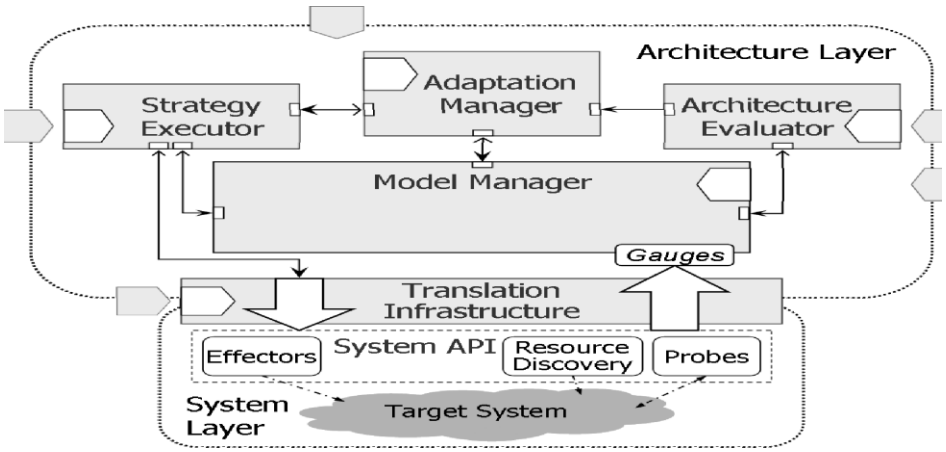

Figure I.3 : Le cadre RANBOW. [20]

L'évaluateur d'architecture évalue le modèle lors de la mise à jour pour garantir que le système fonctionne dans une plage acceptable, telle que déterminée par les contraintes architecturales.

Si l'évaluateur détermine que le système ne fonctionne pas dans la plage acceptée, il déclenche le gestionnaire d'adaptation pour lancer le processus d'adaptation et choisir une stratégie d'adaptation appropriée. Le Strategy Executor exécute ensuite la stratégie sur le système en cours d'exécution via des effecteurs au niveau du système.

5.2.4. Prévision de ressources:

la prédiction est une estimation informée des futures valeurs aléatoires d'une variable système ou d'environnement.par exemple le niveau futur disponible de certaines resources requises par le système.

5.2.5. Intégration des prévisions de ressources dans Rainbow :

Comment les prévisions de ressources peuvent être incorporées dans le cadre d'auto-adaptation Rainbow ? Rainbow doit répondre aux exigences suivantes du cadre de Poladian:

1. **Utilité**: évaluer la qualité des différentes adaptations possibles sur le système.
2. **Pénalité**: pour quantifier les coûts de réalisation des adaptations.
3. **Informations historiques**: pour faciliter la prévision, les valeurs observées dans le passé doivent être intégrées au cadre de prévision.

5.2.6. Points d'intégration pour rendre disponibles les informations prédictives :

Dans Rainbow, les prévisions de ressources peuvent fournir un levier supplémentaire pour évaluer et choisir entre d'autres stratégies d'adaptation. Par exemple, en connaissant la probabilité que le niveau disponible d'une ressource critique, telle que la bande passante, soit inférieur à un certain seuil dans 5 minutes, Rainbow peut choisir une stratégie mettant au repos les sessions client de priorité inférieure, de sorte qu'il continuer à être satisfait avec une latence tolérable. Si, au contraire, la probabilité que la bande passante soit restaurée à des niveaux qui ramèneront naturellement le système dans l'état souhaité, Rainbow peut choisir de

réduire la fidélité de tout ou partie des sessions client. Rainbow peut même choisir de ne rien faire.

5.2.7. Résultat de ce travail :

Ce chapitre, présente une approche visant à améliorer l'auto-adaptation basée sur l'architecture par le biais d'une prédiction anticipée de la disponibilité future des ressources. L'approche utilise un cadre combinant diverses formes de prédiction (statistique, bornée et saisonnière) de manière pratique pouvant être appliquée à diverses circonstances. Il ont soutenu que les systèmes auto-adaptatifs peuvent tirer parti de la prédiction pour améliorer le choix des adaptations et réduire les perturbations du système. Nous avons accordé une attention particulière aux changements nécessaires pour intégrer les prévisions dans un système d'auto-adaptation basé sur une architecture réactive, Rainbow. Nous avons mené plusieurs expériences montrant une amélioration de l'adaptation lorsque la prédiction est utilisée, et discuté de la manière dont nous avons résolu certains problèmes rencontrés lors de l'intégration.

5.3. Approche : L'Auto-adaptation de systèmes mobiles pilotée par le langage de variabilité commune [21].

5.3.1. Problématique de ce travail :

Le contexte d'exécution dans lequel les systèmes omniprésents ou l'informatique mobile s'exécutent change constamment. Par conséquent, les applications de ces systèmes nécessitent une prise en charge de l'auto-adaptation aux changements continus de contexte.

D'autre part, les applications mobiles s'exécutant sur des appareils légers disposant de ressources limitées (batterie, mémoire, processeur, etc.), elles doivent donc adapter leurs fonctionnalités aux variations constantes des ressources, ainsi qu'aux besoins de l'utilisateur.

5.3.2. Objective de ce travail :

Ce travail présente une approche alternative pour la génération automatique de configurations d'application et les plans de reconfiguration au moment de l'exécution. Ceci

est réalisé : en modélisant la variable architecturale au moment de la conception à l'aide du langage de variabilité commune (CVL), et en utilisant un algorithme génétique qui trouve des configurations quasi optimales à l'exécution en utilisant les informations fournies par le modèle de variabilité.

5.3.3. Aperçu sur le travail :

Après avoir modélisé les variabilité de ce système avec le langage de modélisation **CVL** (common variability language) , le **CMS** (context monitoring service) surveiller l'environnement de ressource et fournit les informations de context au **DRS** (dynamique reconfiguration service) qui charger d'adapter les applications au moment de l'exécution . la génération de plan ce fait par la boucle **MAPE-k** (monitoring ,analyses ,plans,execution ,knowledge) qui fait partie de système **DRS** . Lorsque le contexte change et le **DRS** décide qu'il est nécessaire de reconfigurer l'application, l'algorithme génétique est exécuté pour générer un plan de reconfiguration puis de déployer une nouvelle configuration architecturelle adapter au contexte d'exécution actuelle. Cette exécution ce fait par trois étapes :

A. Génération de la population initial :

À cette étape, l'algorithme génétique génère un ensemble de **chromosomes** initiaux, qui représentent des solutions valides et, par conséquent, des modèles de résolution pour l'arborescence dont les éléments (c.-à-d. VSpec) sont similaires aux entités de la modélisation d'entités, représentant les choix liés aux points de variation . De plus, ces modèles de résolution ne dépassent pas les ressources disponibles. Les valeurs de ces gènes étant sélectionnées de manière aléatoire, il est nécessaire d'appliquer certaines transformations à chacun d'eux pour obtenir une solution valide pour chaque solution générée de manière aléatoire .

B. L'Evolution à travers les générations :

Une fois qu'une population initiale de modèles de résolution valides a été générée, l'étape suivante consiste à faire évoluer la population au fil des générations afin de trouver des modèles de résolution conduisant à des configurations architecturales mieux adaptées au contexte actuel. La figure 5 montre un exemple de ce processus d'évolution. A chaque

génération, un nouveau chromosome est généré et introduit dans la population en appliquant plusieurs opérateurs:

Sélection : sélectionner les deux meilleures solutions de la population

croisement : Les gènes des deux chromosomes sont combinés en sélectionnant les gènes des deux, ce qui donne un chromosome descendant. Les gènes de la progéniture sont prélevés chez les deux parents en appliquant un masque généré aléatoirement.

Mutation : Une fois que la progéniture a été générée, une mutation est introduite dans la progéniture en choisissant au hasard un gène du chromosome et en inversant sa valeur. actuel n'est pas valide et doit être modifié pour en obtenir un valide.

Transformation : Le chromosome résultant est invalide et doit être modifié pour en obtenir un valide.

C. Le retour des solutions :

Renvoie le meilleur chromosome. Le processus d'évolution est ensuite répété jusqu'à ce qu'une condition d'arrêt soit atteinte. Par exemple, lorsqu'un nombre maximum de générations est atteint ou lorsque la population n'a pas évolué avec succès après un certain nombre de générations consécutives.

5.3.4. Résultat de ce travail :

Le travail étudié montre que le système DRS est efficace pour être exécuté sur des appareils mobiles (=21,17 ms en temps d'exécution) et qu'il est capable de générer des configurations architecturales presque optimales au moment de l'exécution, adaptées au contexte d'exécution actuel (>87,4% de optimalité par rapport à la solution optimale).

Dans l'étude de cas, la durée de la batterie et l'utilité globale fournie à l'utilisateur par l'application sont améliorés lorsque DRS est utilisé (+ 45,9% de la durée de la batterie et + 10,31% dans l'utilitaire fourni).

5.4. Discutions :

Après avoir étudié les trois travaux de l'adaptation dynamique nous avons constaté que les deux premier travaux ont abordé différentes problématiques et ils n'ont pas abordé la problématique de l'optimalité de l'adaptation au moment de l'exécution. Cette problématique

a été traitée dans troisième travaux mais avec un seul métaheurstique (AG) .par contre dans notre projet de fin d'étude nous avons traité cette problématique a l'aide de deux métaheuristiques. Le tableau ci-dessus résume le résultat d'étude de ces travaux

***Tableau I .1*:** résultat comparatif des approches étudié

Titre	Objectif	Critique	Métaheuristique	Référence
MUSIC: Prise en charge de middleware pour l'auto-adaptation dans des environnements omniprésents et orientés service	offre un support modulaire maximisant la satisfaction de l'utilisateur en adaptant de manière dynamique les applications mobiles	nécessitent de disposer au moment de l'exécution de toutes les configurations valides d'une application	Non	[19]
Améliorer l'auto-adaptation basée sur l'architecture Grâce à la prévision des ressources.	améliorer l'auto-adaptation en adoptant une approche anticipative dans laquelle les prévisions sont utilisées pour éclairer les stratégies d'adaptation.	il n'est pas possible d'évaluer l'adéquation de cette approche dans le cas d'applications mobiles.	Non	[20]

L'Auto-adaptation de systèmes mobiles pilotée par le langage de variabilité commune	génération automatique de configurations d'application et les plans de reconfiguration au moment de l'exécution modélisé par CVL	Il utilise CVL et un seul algorithme optimisation basé sur le niveau actuelle de la batterie .	Oui mais par une seule métaheuristique	[21]
Notre approche	génération automatique de plans d'adaptation optimale au moment de l'exécution modélisé par UML		Notre approche utilise deux métaheuristiques , et la modélisation en UML. Evaluation basé sur une nouvelle fonction de fitness	/

6. Conclusion :

Dans ce chapitre nous avons passé en revue les différentes propriétés des systèmes auto-adaptatifs, nous avons également étudié quelques travaux de recherche les plus significatifs qui ont traité l'auto adaptation des systèmes. Ce qui nous a permis de bien cerner notre problématique afin de définir un système auto-adaptatif. Dans le chapitre suivant nous parlerons de quelques métaheuristiques, leur classification et leurs caractéristiques ainsi que l'application de ces derniers sur l'auto adaptation.

Chapitre *II* : Généralité sur les métaheuristiques

1. Introduction

Pour un service d'adaptation, le temps est critique car, pour être utiles, les applications doivent être adaptées sans que le temps supplémentaire employé pour le processus d'adaptation ne soit noté. De plus, il est difficile de trouver une solution optimale à un problème d'optimisation dans un temps limité. Il est nécessaire disposer de techniques comme les heuristiques et les métaheuristiques qui tente d'offrir une solution proche de l'optimal dans un temps polynomial. Nous avons consacré ce chapitre pour présenter quelques algorithmes de résolution et leur impact qui nous sera utile dans notre projet afin de trouver le plan d'adaptation optimal dans un temps limité.

2. Algorithme de résolution

La majorité des problèmes d'optimisation ne possèdent pas à ce jour un algorithme efficace de complexité polynomiale, valable de trouver la solution optimale en un temps raisonnable.

Ceci a motivé les chercheurs à développer de nombreuses méthodes de résolution en recherche opérationnelle et en intelligence artificielle. Ces méthodes de résolution peuvent être réparties en deux grandes classes:

- **Les Algorithmes exactes**
- **Les Heuristiques**

2.1. Les algorithmes exactes

Ils se basent généralement sur une recherche complète de l'espace des combinaisons afin de trouver une solution optimale.

Le principe des méthodes exactes consiste généralement à énumérer, souvent de manière implicite, l'ensemble des solutions dans le but de trouver la solution optimale, par exemple branche and bound, Programmation dynamique, Programmation linaire [12]

Les méthodes de résolution exactes permettent d'obtenir une solution dont l'optimalité est garantie, et dans certain situation on cherche des solutions sans garantie l'optimalité, mais au profit d'un temps de calcul plus réduit. Pour cela on applique des méthodes appelée heuristique et métaheurstiques .quelle sont ces méthodes ?

2.2. Les Heuristiques :

Une heuristique est une technique de résolution spécialisée a un problème, fournit rapidement une solution réalisable, pas nécessairement optimale ou exacte, pour un problème d'optimisation difficile. [13] .Il existe deux types de l'heuristique :

2.2.1 .Heuristique Spécifique :

Sont des heuristiques qui spécifié a chaque type de problème

2.2.2. Les MétaHeuristiques :

Les méta heuristiques sont des algorithmes généraux d'optimisation applicables à une grande variété de problèmes. Sans pour autant que l'utilisateur ait à modifier leur structure, elles sont apparues pour résoudre au mieux des problèmes d'optimisation et inspirées par des analogies avec la physique (recuit simulé), avec la biologie (algorithmes évolutionnaires) ou encore l'éthologie (colonies de fourmis, essaims particulaires). Cependant, elles ont l'inconvénient d'avoir plusieurs paramètres à régler, les méta heuristiques permettent de trouver des solutions, peut être pas toujours optimales, en tout cas très proches de l'optimum et en un temps raisonnable. [14]

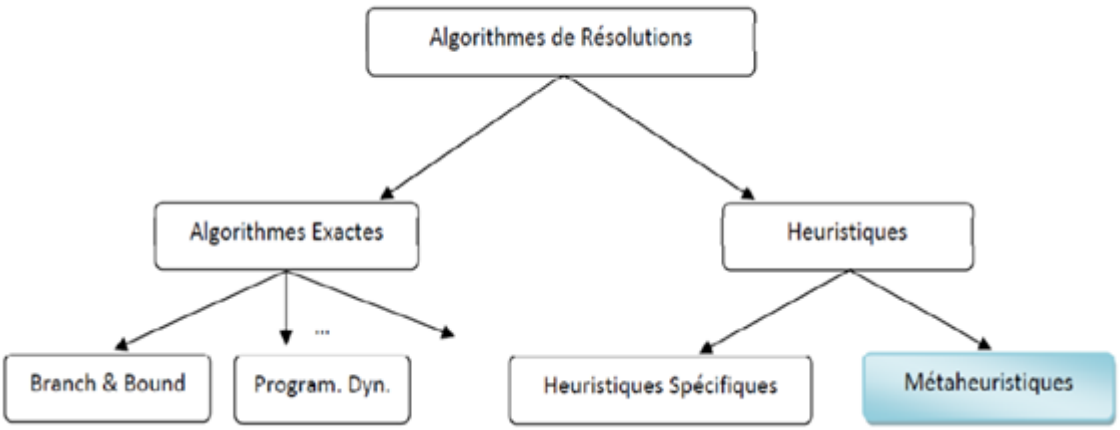

Figure II.1 : classes des méthodes de résolutions. [23]

Il existe plusieurs façons de classer les métas heuristiques. On donne quelques une, et nous adopterons celle faisant la différence entre les méthodes de trajectoire (solution unique) et les méthodes basées sur une population.

2.2.2.1. Méta Heuristique solution unique :

Les méta heuristiques à solution unique commencent avec une seule solution initiale et s'en éloignent progressivement, en construisant une trajectoire dans l'espace de recherche. Les méthodes de trajectoire englobent essentiellement la méthode de descente, la méthode du recuit simulé, la recherche tabou, la recherche à voisinage variable, la recherche locale itérée, et leurs variantes. [13]

Tableau II .1: caractéristiques de cinq métaheuristiques a solution unique

Métaheuristique	Principes générique	Principe spécifique (a un problème)
Méthode par descente	Sélection d'un voisin	Evaluation d'une solution Génération des voisinages
Recherche tabou	Critère d'arrêt Critère d'aspiration Liste taboue	Evaluation d'une solution Génération des voisinages
Recuit simulé	Système de refroidissement Critère d'arrêt	Evaluation d'une solution Génération des voisinages
Recherche locale itérative	Critère d'arrêt Critère d'acceptation Intensification	Evaluation d'une solution Perturbation
Recherche à voisinage variable	Parcours des voisinages Intensification	Evaluation d'une solution Génération des voisinages

2.2.2.2 .Métaheuristique à population :

À l'inverse des méthodes de recherche à solution unique, les métaheuristiques à population de solutions sont des méthodes qui font évoluer simultanément un ensemble d'individus (solutions) dans l'espace de recherche, où chacun profite de l'expérience du groupe, de manière directe ou indirecte. Ces méthodes sont principalement inspirées du vivant **[16].** On peut distinguer deux catégories de métaheuristiques à population :

- **les algorithmes évolutionnaires** : inspirés de la théorie de l'évolution de C. Darwin
- **les algorithmes d'intelligence en essaim** : inspirés de l'éthologie et de la biologie.

A. les algorithmes d'intelligence en essaim :

L'intelligence en essaim (**SI : Swarm Intelligence**) est née de la modélisation mathématique et informatique des phénomènes biologiques rencontrés en éthologie [Bonabeau et al., 1999]. Elle recouvre un ensemble d'algorithmes, à base de population d'agents simples (entités capables d'exécuter certaines opérations), qui interagissent localement les uns avec les autres et avec leur environnement. Deux exemples phares d'algorithmes de l'intelligence en essaim sont les algorithmes de **colonies de fourmis** et les algorithmes d'optimisation par **essaim particulaire**.

A.1. Essaime particulaire :

L'Optimisation par Essaim Particulaire (**OEP**) a été proposée par Kennedy et Eberhart en 1995 [Kennedy et al., 1995]. Cette méthode est inspirée du comportement social des animaux évoluant en essaim. L'exemple le plus souvent utilisé est le comportement des bancs de poissons [Wilson, 1975 ; Reynolds, 1987]. En effet, on peut observer chez ces animaux des dynamiques de déplacement relativement complexes, alors qu'individuellement chaque individu a une intelligence limitée et une connaissance seulement locale de sa situation dans l'essaim. Un individu de l'essaim n'a pour connaissance que la position et la vitesse de ses plus proches voisins. Chaque individu utilise donc, non seulement, sa propre mémoire, mais aussi l'information locale sur ses plus proches voisins pour décider de son propre déplacement. Des règles simples, telles que '' aller à la même vitesse que les autres'', ''se déplacer dans la même direction'' ou encore ''rester proche de ses voisins'' sont des exemples de comportements qui suffisent à maintenir la cohésion de l'essaim, et qui permettent la mise en œuvre de comportements collectifs complexes et adaptatifs. ''L' intelligence globale'' de l'essaim est donc la conséquence directe des interactions locales entre les différentes particules de l'essaim. **[15]**

Le déplacement d'une particule est influencé par les trois composantes suivantes :

• **Une composante physique** : la particule tend à suivre sa direction courante de déplacement.

• **Une composante cognitive** : la particule tend à se diriger vers le meilleur site par lequel elle est déjà passée ;

• **Une composante sociale** : qui incite la particule à se diriger vers le meilleur site trouvé par ses congénères

Dans le cas d'un problème d'optimisation, la qualité d'un site de l'espace de recherche est déterminée par la valeur de la fonction objective en ce point.

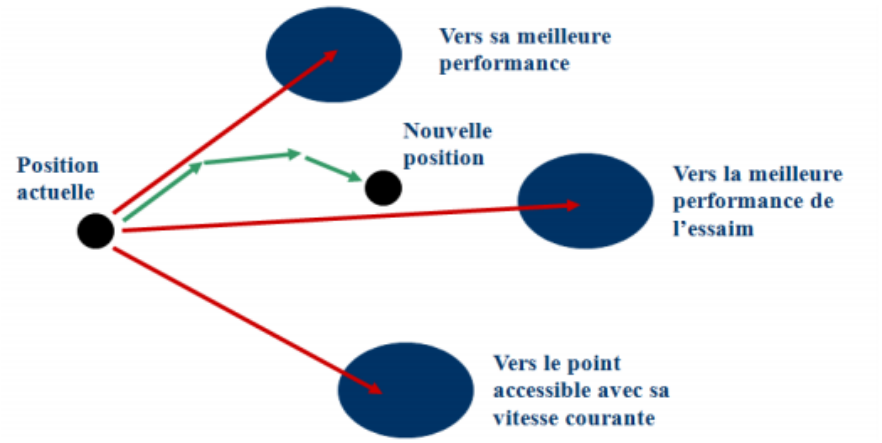

Figure II.2 **:** la stratégie de déplacement d'une particule

A.2. Colonie de fourmis :

Les algorithmes de colonies de fourmis sont des métaheuristiques d'optimisation inspirées du comportement collectif des fourmis dans leur processus de recherche de nourriture et d'optimisation du chemin entre leur nid et la source de nourriture trouvée**.[15]**

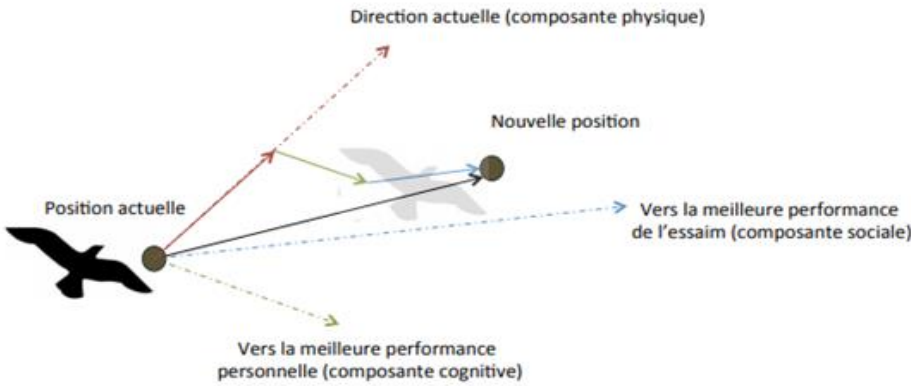

Figure II.3 : algorithme OEP: exemple de déplacement d'une particule dans l'espace de recherche. [15]

Ces algorithmes ont été initiés par Dorigo et al. [Dorigo, 1992; Dorigo et al., 1996] en s'inspirant des travaux de Deneubourg et al. [Deneubourg et al., 1983] qui ont modélisé le comportement aléatoire des fourmis. Le premier algorithme d'optimisation, basé sur les colonies de fourmis, intitulé "**Ant System**" **(AS),** a été proposé pour résoudre le problème du voyageur de commerce [Dorigo et al., 1996]. Depuis, plusieurs améliorations et variantes ont

vu le jour et ont été appliquées dans plusieurs domaines, avec plus ou moins de succès. Il était remarqué [Goss et al., 1989; Beckers et al., 1992] que les fourmis réelles sont capables d'emprunter le chemin le plus court entre leur nid et une source de nourriture grâce à un comportement collaboratif collectif, sachant qu'aucun individu n'a une vision globale du parcours. Le comportement des fourmis est basé sur les principes suivants :

– **L'auto-organisation** : les fourmis sont capables, en effectuant individuellement des tâches simples, de résoudre des problèmes complexes. Le comportement simple et local de chaque individu fait émerger un modèle ("pattern", en anglais) de niveau global grâce aux interactions.

– **La stigmergie** : qui est la technique de communication entre les individus (fourmis) en modifiant, d'une manière dynamique, l'environnement où elles évoluent. En effet, en se déplaçant, les fourmis déposent de la phéromone 1 pour marquer le chemin parcouru. En absence de cette substance, les fourmis se déplacent aléatoirement dans l'environnement. Par contre, en sa présence, une fourmi la détecte et peut suivre sa trace avec une probabilité proportionnelle à son intensité. Ainsi, plus une piste est parcourue, plus elle est attirante.

– **Un contrôle décentralisé** : cela signifie qu'il n'y a pas de décision prise à un niveau donné ou par un seul individu. En effet, chaque individu effectue des actions relativement simples en se basant uniquement sur des informations locales de l'environnement, sans vision du problème dans sa globalité.

– **Une hétérarchie dense** : par opposition à une structure hiérarchique, où la population est dirigée par un individu, l'hétérarchie dense est une structure horizontale, où les individus sont fortement connectés, agissant ainsi sur les propriétés globales du système.

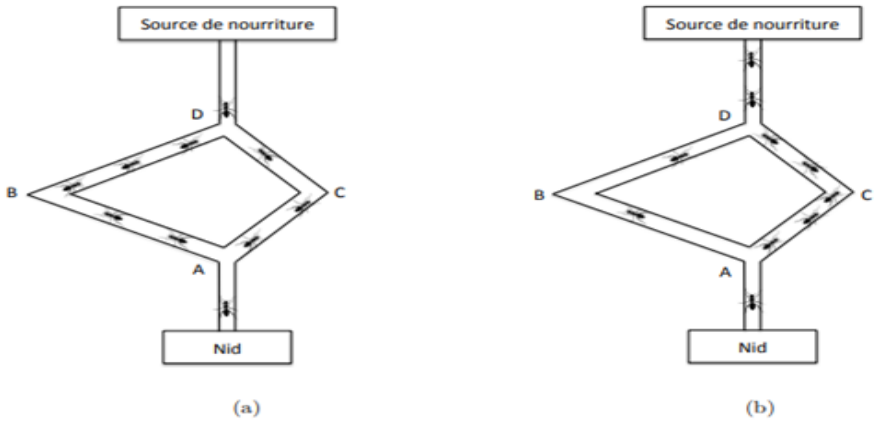

Figure II.4 **:** Expérience de sélection des branches les plus courtes par une colonie de fourmis : (a) au début de l'expérience, (b) a la fin de l'expérience. [24]

En effet, au début de l'expérience **(*Figure II.4* (a)),** les fourmis arrivent au point **(A),** et comme il n'y a pas de traces de phéromone, elles choisiront l'une des deux directions aléatoirement avec la même probabilité. Les fourmis qui ont emprunté le chemin **ACD** arriveront, en toute logique, plus rapidement à la source de nourriture que les autres et feront le chemin inverse plus tôt. Ainsi la quantité de phéromone déposée sur le parcours **ACD** sera plus importante que celle déposée dans **ABD**. Comme un parcours contenant plus de phéromone a plus de chance d'être emprunté, plus de fourmis vont choisir le parcours **ACD** **(*Figure II.4* (b)).** Ainsi, à terme, le chemin le plus court est renforcé et sera emprunté par la majorité des individus ; de plus, si on considère le phénomène d'évaporation du phéromone, après un certain temps, tous les individus parcourront le chemin le plus court.

B. algorithme évolutionnaires :

Les algorithmes évolutionnaires sont des méthodes stochastiques d'optimisation globale basées sur la théorie Darwinienne de l'évolution des espèces biologiques, ils utilisent à la fois les principes de la survie des individus les mieux adaptées et ceux de la propagation du patrimoine génétique qui s'inspirent des mécanismes de sélection naturelle et des phénomènes génétiques tel que des mécanismes d'évolution de la nature : croisements, mutations, sélections, etc. Pour utiliser ces algorithmes, il faut disposer d'une population d'individus. Chaque individu dispose d'une chaîne chromosomique qui dirige son comportement. Cette chaîne s'apparente à l'ADN dans les organismes vivants. Comme dans les systèmes naturels, des croisements sont réalisés périodiquement et permettent à l'algorithme de créer la génération suivante d'individus, ainsi des mutations sont aussi effectuées. Ces mutations évitent à l'ensemble de la population de converger vers une solution qui ne serait pas optimale. Il existe plusieurs types de ces algorithmes mais l'idée essentielle est la même : simuler l'évolution d'une population dans un espace de recherche à l'aide de trois opérateurs: sélection, croisement, mutation. Malgré la simplicité du processus évolutionnaire, fabriquer un algorithme évolutionnaire efficace est une tâche difficile, car les processus évolutionnaires sont très sensibles aux choix algorithmiques et paramétriques. L'expérience a prouvé que les réussites les plus importantes sont fondées sur une très bonne connaissance du problème à traiter, et une compréhension délicate des mécanismes évolutionnaires, La figure suivante décrit le squelette d'un algorithme évolutionnaire type. **[15]**

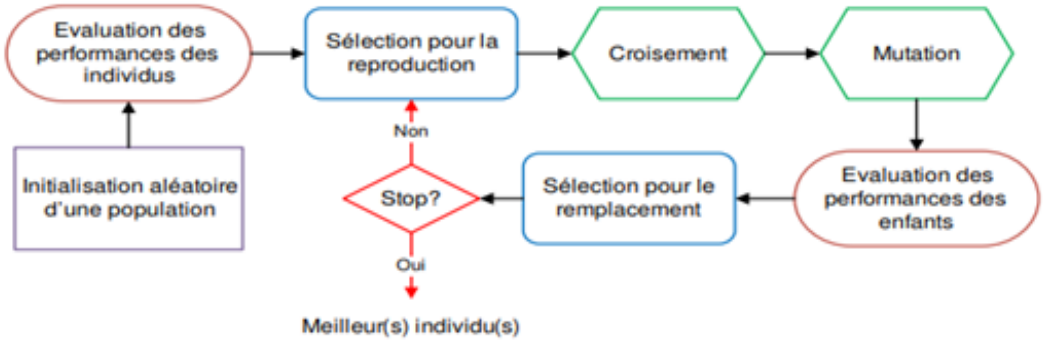

Figure II.5 : principe d'un algorithme évolutionnaire (EA) **[13]**

On distingue quatre grandes familles d'algorithmes évolutionnaires comme indiqué par la figure 8 :

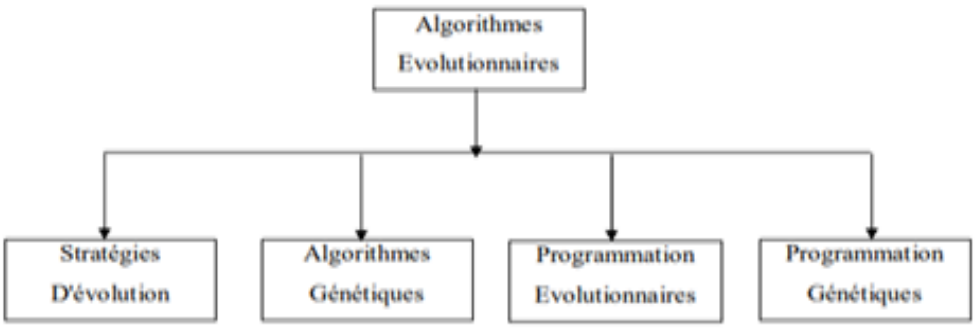

Figure II.6 : différentes classes d'algorithmes évolutionnaires

Les algorithmes génétiques **(GA : Genetic Algorithms)** sont, sans conteste, la technique la plus populaire et la plus largement utilisée des EAs

3. Algorithme Génétique :

3.1. Définition :

Les algorithmes génétiques (AGs) sont des algorithmes d'exploration fondés sur les mécanismes de la sélection naturelle et de la génétique. Un AG modélise le processus d'évolution collectif d'une population d'individus pour s'adapter à un environnement**[16].**sont basés sur les phénomènes de reproduction de la nature qui mettent en œuvre : le croisement, la mutation et la sélection (survie des plus aptes). Ils permettent, par un codage approprié, de

trouver en un temps limité, de bonnes solutions aux problèmes d'optimisation complexes. Ces algorithmes sont jugés efficaces et robustes pour résoudre ces types de problèmes, efficaces parce qu'ils font évoluer une population de solutions potentielles et pas seulement une seule solution, et robustes grâce à leur pouvoir de résolution des problèmes non linéaires et discontinus.

3.2. La mise en œuvre d'un algorithme génétique :

La mise en œuvre d'un algorithme génétique est réalisée suivant les étapes suivantes :

1. Création d'une population initiale.
2. Evaluation des individus de la population.
3. Sélection des meilleurs individus.
4. Reproduction (Croissement et mutation).
5. Formation d'une nouvelle génération.

La figure suivante montre l'organigramme de fonctionnement d'un algorithme génétique.

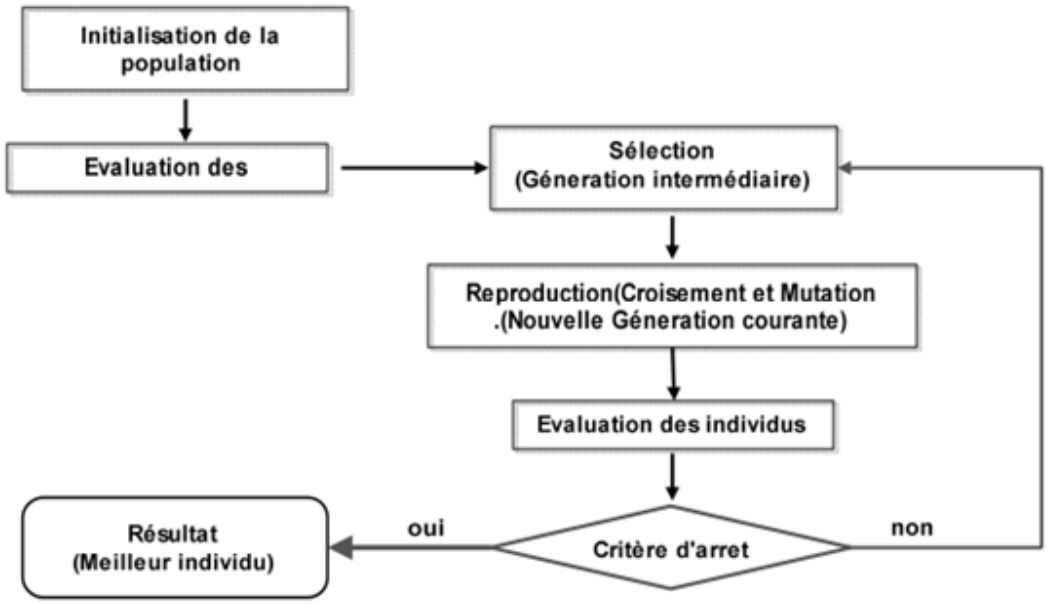

Figure II.7 : organigramme de l'algorithme génétique. [25]

3.3. Principe d'Algorithme Génétique :

Le principe des algorithmes génétiques est de simuler le processus d'évolution des espèces dans leur milieu naturel, en s'inspirant des théories de l'évolution proposées par Charles Darwin. L'évolution des espèces vivantes repose sur les lois de survie ou bien ce qu'on appelle la sélection naturelle. Selon ces lois, seuls les individus les mieux adaptés ont une longévité suffisante pour générer une descendance (se reproduire). Dans ce contexte, un individu est identifié par un ensemble de données ou **gènes** appelés **chromosomes**. La reproduction se fait par une recombinaison de chromosomes de deux individus primaires, donnant ainsi naissance à des individus enfants ayant une empreinte génétique héritée des parents. En fait, le code génétique des enfants peut contenir des gènes inexistants chez les parents. Ce phénomène génétique s'appelle « **la mutation** ». Il permet des changements dans la morphologie des espèces, toujours dans le sens d'une meilleure adaptation au milieu naturelle principe de fonctionnement des AGs est extrêmement simple. On part d'un ensemble initial d'individus nommé « **population initiale** » généré le plus souvent d'une manière aléatoire. Ensuite, on évalue la performance de chaque individu en calculant la valeur de la fonction coût. L'application des trois opérateurs génétiques de « **sélection, croisement et mutation** » permet de créer un nouvel ensemble d'individus appelé « **population enfant** ». Cette population est évaluée à son tour pour indiquer la performance de chacun de ses individus. Cette connaissance permet de décider lesquels des individus enfants méritent de remplacer certains parents. La nouvelle population obtenue, appelée « **population parent** », constitue la population parent de la nouvelle génération. Si les critères d'arrêt sont vérifiés, on considère la ou les solutions obtenues comme satisfaisantes, autrement, on recommence le cycle jusqu'a satisfaction de ces critères.

Le critère d'arrêt d'un AG peut être la convergence de l'ensemble des individus de la génération courante vers un même optimum. Le plus souvent, l'algorithme est arrêté au bout d'un nombre d'itérations (générations) fixé a priori. **[17]**

il est nécessaire de définir quelque termes de base rencontrés dans la littérature **:[18]**

- **Gène** : partie élémentaire (caractère) non divisible d'un chromosome.
- **Chromosome** : solution potentielle du problème sous une forme codée (forme de chaîne de
- caractères)
- **Individu** : solution potentielle du problème

- **Population** : ensemble fini d'individus (de solution)

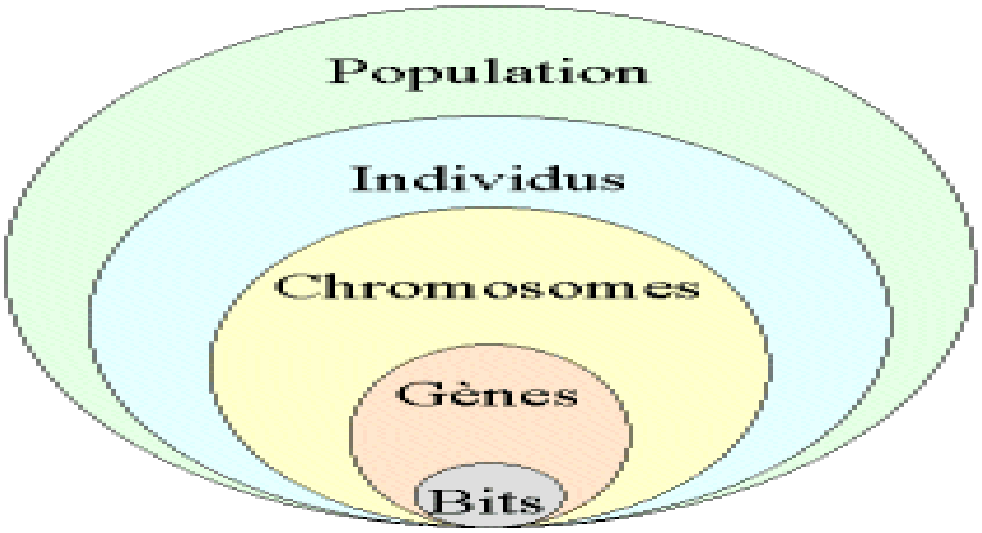

***Figure II.8* :** les cinq niveaux d'organisation d'un algorithme génétique [26]

3.3.1. Codage des Variables :

La littérature définit deux types de codage : **binaire et réel.**

3.3.1.1. Le codage binaire :

Le codage binaire consiste à représenter chaque vecteur des paramètres X par une chaîne de bits dont chaque instance prend la valeur 0 ou 1. Cette chaîne peut être une concaténation des différents variables du problème. Chaque paramètre est transformé en sa valeur binaire avec un nombre de bits fixé a priori. **[17]**

3.3.1.2. Codage réel :

La représentation des solutions dans le cadre des AG n'est pas nécessairement réduite à un alphabet de faible cardinalité (0,1), il existe toute une école pour laquelle la représentation la plus efficace est celle qui s'appuie sur des nombres réels. Cette représentation est à la base de l'approche évolutionnaire « Evolution stratégie ». Ce type de codage présente certains avantages par apport au codage binaire :

- Le codage réel est robuste pour les problèmes considérés comme difficile pour le codage binaire.
- Ce codage nécessite une adaptation des opérateurs de croisement et mutation. **[23]**

3.3 .2. Population Initiale :

Plusieurs mécanismes de génération de la population initiale sont utilisés dans la littérature. Le choix de l'initialisation se fera en fonction des connaissances que l'utilisateur a sur le problème. S'il n'a pas d'informations particulières, alors une initialisation aléatoire, la plus uniforme possible afin de favoriser une exploration de l'espace de recherche maximum, sera la plus adaptée. Par ailleurs, cette étape présente un problème principal qui est celui de choix de la taille de la population. En effet une taille de population trop grande augmente le temps de calcul et nécessite un espace mémoire considérable, alors qu'une taille de population trop petite conduit à l'obtention d'un optimum local. **[23]**

3.3.3. La fonction d'évaluation :

La fonction d'évaluation ou de coût nommée « **fonction de fitness** » en terminologie anglo-saxonne, prend en argument l'individu et lui attribue un coût de performance: une valeur numérique (**fitness**), qui est supposée proportionnelle à la qualité de l'individu. Par exemple, pour le Problème du Voyageur de Commerce (**PVC**), la fonction d'évaluation utilisée calcule la distance parcourue par le commis voyageur pour un chemin donné. Plus la distance est courte, plus le parcours est meilleur car moins coûteux. Cette fonction doit guider l'algorithme vers l'optimum en réalisant implicitement une pression de sélection dans cette direction. En effet, l'efficacité d'un AG s'appuie pour une grande part, sur la qualité de cette fonction. La mise au point d'une bonne fonction d'adaptation doit respecter plusieurs critères qui se rapportent à sa complexité et à la satisfaction des contraintes du problème**. [17]**

3.3.4. Les opérateurs génétiques :

Un algorithme génétique simple utilise les trois opérateurs suivants : **la sélection, le croisement et la mutation**

3.3.4.1. La Sélection :

Une fois la génération et l'évaluation de la population initiale terminée, l'étape suivante consiste en la détermination des individus à se reproduire (les parents de la prochaine génération). Cette sélection est fondée sur la qualité des individus, estimée a l'aide de la fonction d'adaptation (**fonction objectif**). En règle générale, la probabilité de survie d'un individu est directement liée à son efficacité relative au sein de la population. Il existe plusieurs méthodes pour la sélection **[17].** On peut citer à titre d'exemple:

a. La sélection par tournoi (tournement) :

Le tournoi le plus simple consiste à choisir aléatoirement un nombre k d'individus dans la population et à sélectionner celui qui a la meilleure performance. Les individus qui participent à un tournoi sont remis ou sont retirés de la population, selon le choix de l'utilisateur. Avec le tournoi binaire, sur deux individus en compétition, le meilleur gagne avec une probabilité p ∈ [0, 5; 1]

b. la sélection par roulette (Wheel) :

C'est une méthode stochastique qui exploite la métaphore d'une roue de loterie. Elle consiste à associer à chaque individu un secteur (ou case de la roue) dont l'angle est proportionnelle à sa performance. La roue étant lancée, l'individu sélectionné est celui sur lequel la roue s'est arrêtée. On répète cette opération N fois afin de sélectionner les N individus qui vont se reproduire dans la prochaine étape. C'est clair que les individus ayant la plus grande valeur de fitness (plus de surface) auront plus de chance d'être choisis.

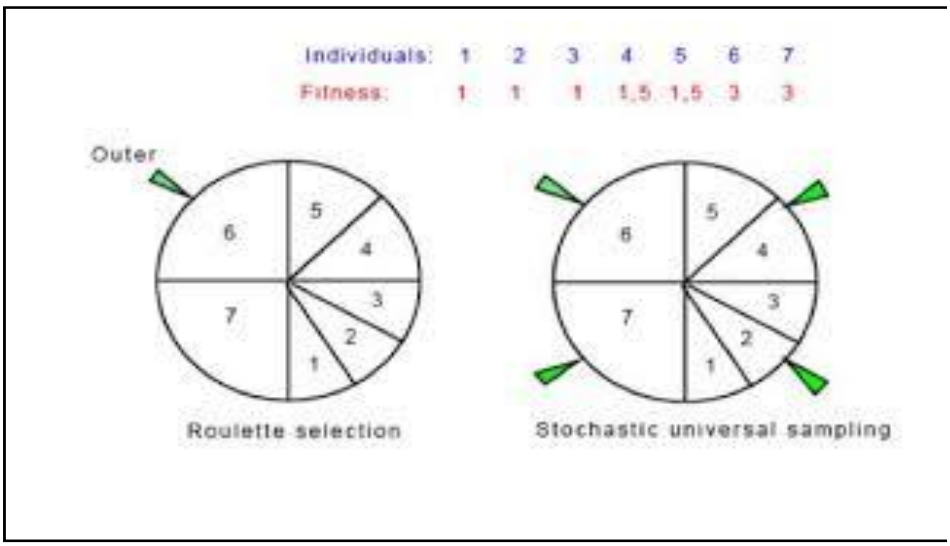

Figure II.9: la sélection par roulette. **[17]**

3.3.4.2. Le croissement :

La naissance d'un nouvel individu, nécessite la prise aléatoire d'une partie des gènes de chacun des deux parents. Ce phénomène, issu de la nature est appelé croisement (crossover). Il s'agit d'un processus essentiel pour explorer l'espace des solutions possibles. Une fois la

sélection terminée, les individus sont aléatoirement répartis en couples. Les chromosomes parents sont alors copiés et recombinés afin de produire chacun deux descendants ayant des caractéristiques issues des deux parents. Dans le but de garder quelques individus parents dans la prochaine population, on associe à l'algorithme génétique une probabilité de croisement, qui permet de décider si les parents seront croisés entre eux ou s'ils seront tout simplement recopiés dans la population suivante. Il existe plusieurs types de croissement parmi lesquels on trouve : le croissement en 1 point, le croissement en deux points et le croissement en N points, ces types sont résumés dans la figure suivante [18]:

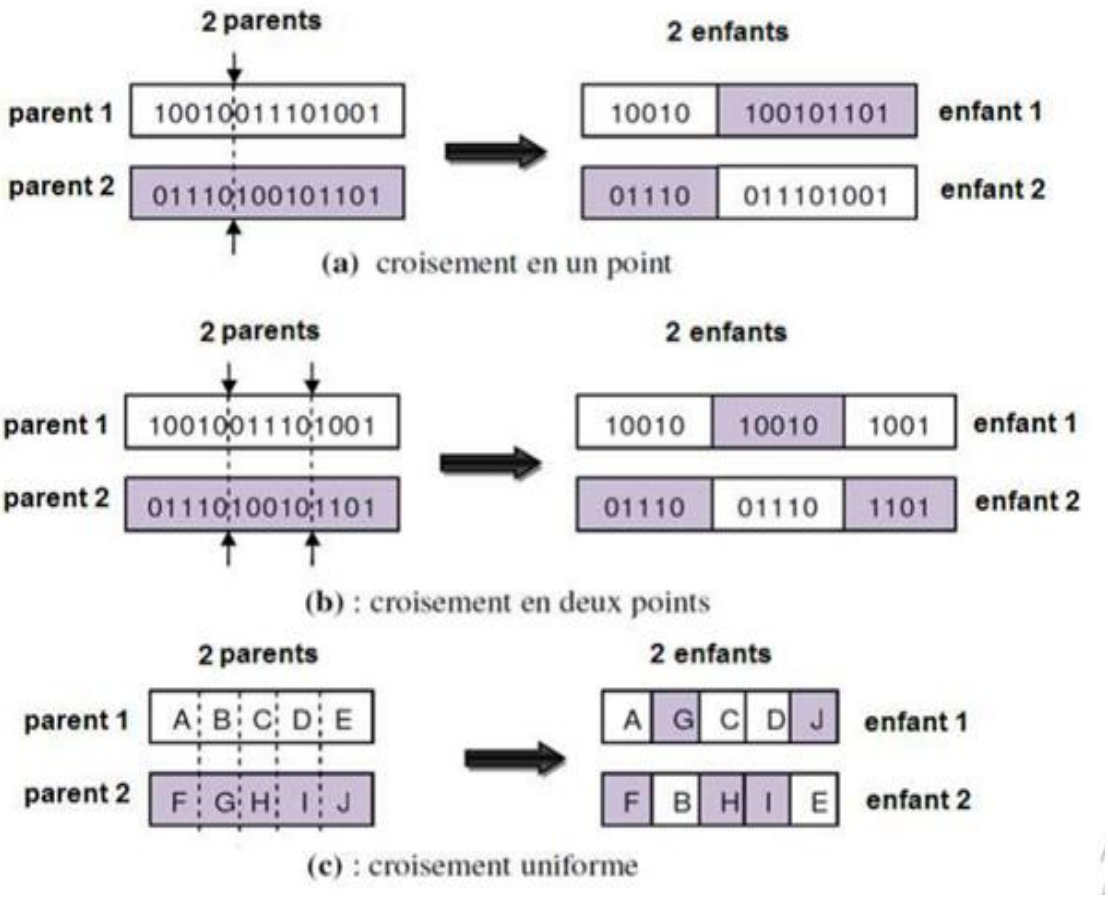

***Figure II.10*:** exemple d'opération de croisement **[23]**

3.3.4.3. La mutation :

L'opérateur de mutation, en générant de nouveaux gènes, a pour rôle de permettre d'explorer la totalité (en théorie) de l'espace d'état, ce qui correspond à la propriété d'ergodicité de parcours d'espace, essentielle aux AG pour leurs propriétés de convergence. En effet les preuves théoriques de convergence des AG peuvent fonctionner sans croisement,

mais pas sans mutation. L'opérateur de mutation fonctionne comme suit. Pour les problèmes discrets, un gène du chromosome est tiré aléatoirement et sa valeur est remplacée par une des autres valeurs possibles (tirée aléatoirement elle aussi). Dans le cas des problèmes continus, le gène est également tiré aléatoirement, et remplacé par une valeur aléatoire du domaine d'extension des gènes (espace d'état). La figure suivante montre un exemple de mutation. [18]

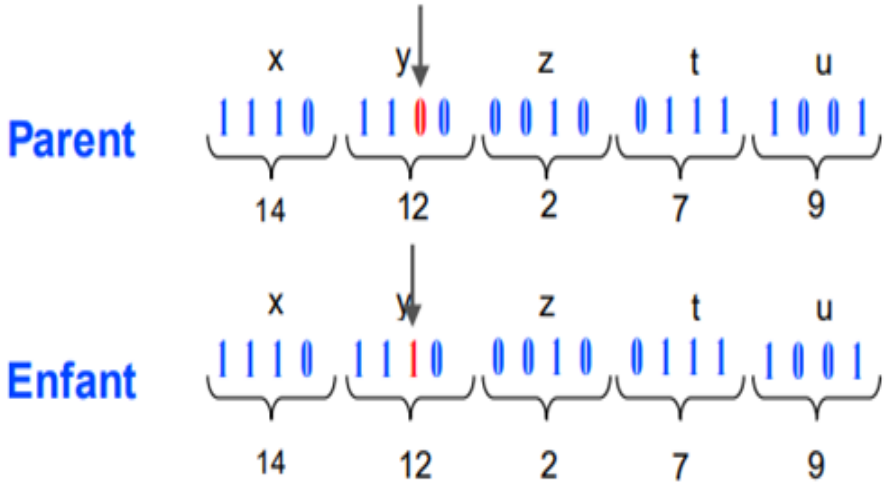

Figure II.11 : exemple de mutation **[23]**

4. Conclusion :

Nous avons présenté dans ce chapitre un aperçu général sur les métaheuristiques en présentant d'abord quelques notions préliminaires sur les principaux algorithmes d'optimisation, en les divisant en deux classes : les méthodes qui font évoluer une seule solution et les méthodes à base de population de solutions ,nous avons étudié les différentes métaheuristiques à population (colonie de fourmis , essaims particulaire et algorithme évolutionnaire) et nous avons focalisé nos recherches sur les algorithmes évolutionnaires (plus précisément l' algorithme génétique) et l'algorithme d'essaime particulaire qui sont destinés à résoudre des problèmes d'optimisation difficiles.

Chapitre III : Conception et modélisation de l'approche proposée

1. Introduction :

Parmi les grands challenges des appareils mobiles c'est d'assurer l'autonomie de la batterie pour une longue durée afin de permettre à l'utilisateur final plus de mobilité et flexibilité dans l'utilisation de différentes applications mobiles.

Afin d'étendre la durée de vie de la batterie, l'utilisateur de l'appareil mobile doit gérer manuellement l'utilisation des applications de son appareil mobile.

L'objectif de notre projet est de trouver une solution alternative qui permet d'adapter l'utilisation de différentes fonctionnalités (wifi, gps, Bluetooth ….) à un instant donné avec le niveau de chargement de la batterie de l'appareil mobile. Cette solution doit être auto adaptatif sans aucune intervention manuelle par l'utilisateur final. Ce chapitre sera consacré à la conception de notre solution qui permet la planification de l'auto adaptation d'une application mobile , en se basant sur deux métaheuristiques précédemment étudiées à savoir : les algorithmes génétique et l'essaim particulaire.

Tout d'abord nous allons décrire notre étude de cas avec deux diagrammes de langage UML (Unfied Modeling Language) : diagramme de cas d'utilisation et le diagramme de class.

2. Cas d'étude :

Nous avons choisis comme étude de cas une application mobile qui consiste à aider les participants (étudiants et enseignants) d'assister aux événements universitaires, les tient au courant des dernières nouvelles et fournit plusieurs fonctionnalités sociales. L'application fournit l'ensemble de services variables suivant:

- Accès aux informations sur les événements et les nouvelles concernant le département informatique (affichage de : consultation, examen, planning et délibération) comme image ou texte selon différentes qualités (haute, moyenne, faible)
- Recevoir des vidéos d'un événement universitaire (soit : une conférence, réunion, soutenance), La qualité de la vidéo reçue est variable (haute, moyenne, basse).
- Trouver la localisation d'un membre utilisant le GPS ou le WlAN
- partage d'information entre les utilisateurs par internet, Bluetooth, NFC
- Echange des messages.

Cette application peut être adaptée en fonction des préférences de l'utilisateur (par exemple, une vidéo de haute qualité est préférable), de la disponibilité des ressources (par exemple, le WLAN est utilisé car le GPS n'est pas disponible) ou de la quantité de ressources consommées (par exemple, utiliser une qualité vidéo faible). Par ce que la batterie du téléphone mobile est faible).

3. Modélisation :

Pour concevoir notre système, nous avons choisi d'utiliser le langage de modélisation unifié UML (Unified Modeling Language). Ce dernier est un langage de modélisation graphique structuré sur un méta modèle définissant les éléments de modélisation (concept manipulé par le langage) et la sémantique de ces éléments (définitions et sens de leurs utilisations). C'est un langage formel organisé autour des diagrammes qui seront développés dans la suite du présent chapitre

3.1. Diagramme des cas d'utilisation

Les cas d'utilisation permettant de structurer les besoins des utilisateurs et les objectifs correspondants d'un système. Dans un diagramme de cas d'utilisation, Un acteur représente un rôle joué par une personne qui interagit avec le système (*Tableau III .1*).

Tableau III .1: L'ensemble des cas d'utilisation

Cas d'utilisation	Acteurs	Description des cas d'utilisation
Partager événement	Utilisateur	• partager vidéo par internet ou Bluetooth ou NFC • partager image par internet ou Bluetooth ou NFC • partager texte par internet ou Bluetooth ou NFC
Recevoir notification	Utilisateur	• Recevoir vidéo • Recevoir image • Recevoir texte
Envoyer message	Utilisateur	• Envoyer un message privé a un autre utilisateur
Trouver localisation	utilisateur	• Trouver la position d'un enseignant par GPS • Trouver position d'un enseignant par W-LAN

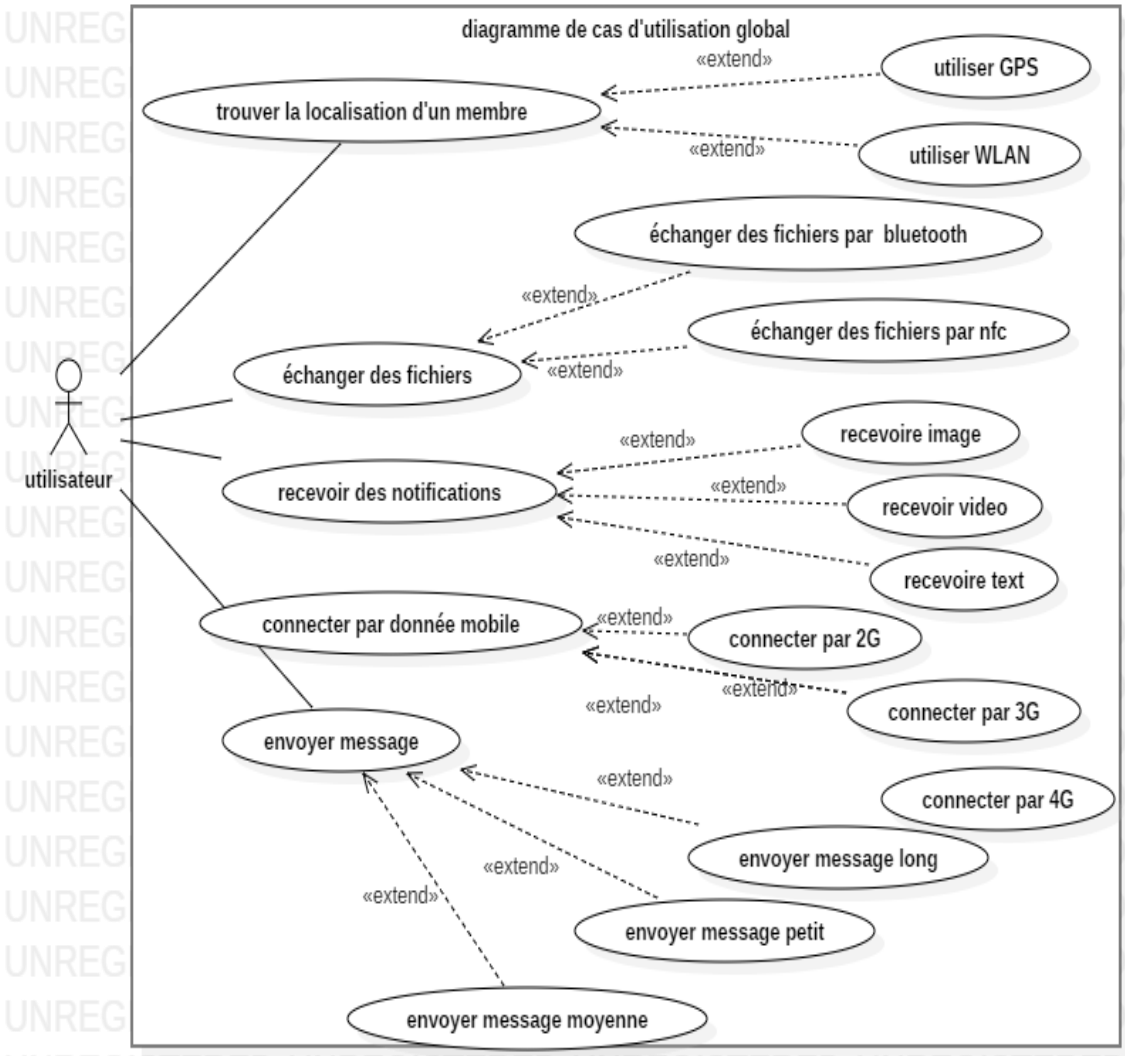

Figure III.1 : Diagramme des cas d'utilisation « général »

4. Planification de notre approche :

Le développement de deux algorithme (AG et PSO) avec le langage java prend beaucoup de temps ce qui nous empêche de faire une application mobil avec androïde studio, donc on a décidé de faire une application bureau pour tester la validité de notre approche.

Initialement, l'utilisateur peut exécuter tous les services qu'il souhaite utiliser, ce qui entraîne une diminution de la durée de vie de la batterie.

Pour surveiller l'environnement de niveau de la batterie et contrôler son statut, nous avons développé une stratégie auto adaptative en utilisant des métaheuristiques.

Cette stratégie Représenté dans la comparaison suivante :

Si {(taille batterie –total consommation) / total consommation} <seuil

Le fonctionnement de l'un de ces métaheuristique est automatiquement lancer afin de reconfigurer ces services par d'autres services ayant le même usage avec moins de consommation de batterie, ce qui augmente la durée de vie de la batterie et c'est ca l'objectif souhaité.

Tel que : Le seuil est Le niveau de batterie qui nécessite l'exécution des algorithmes. Total consommation est la consommation total des services en cours d'exécution qui sont activé par l'utilisateur

L'exemple suivant exprime cette opération :

On prend : niveau de batterie = 4000 mAh, le seuil = 10 et on active GPS ,4G, Luminosité haut qualité

Totale consommation de ces 3 services est égale 140 mAh.

Le niveau de batterie commence a diminué par temps de la manière suivante :

Minute 1 : 4000-140= 3860 ➔ 3860/140 = 27➔ 27 > seuil.

Minute 2 : 3860 – 140 = 3720 ➔3720/140=26 ➔26>seuil.

On continue ces opérations jusqu'on obtient une valeur inferieure au seuil, et donc l'un de ces métaheuristique à exécuter automatiquement et remplacer les services activés par d'autres services ayant le même usage avec moins de consommation de batterie.

5. Conception de l'approche basée sur L'algorithme génétique :

Pour répondre à notre problématique qui peut être présentée comme un problème d'optimisation (maximisation d'utilité des fonctionnalités avec une minimisation de consommation de batterie), nous utilisons comme première solution l'algorithme génétique afin de trouver la meilleure stratégie d'adaptation comme expliquer dans les sections suivantes.

5.1. Codage du chromosome proposé :

L'une des étapes importantes pour l'implémentation de l'AG est le codage .Pour appliquer notre solution basée sur l'AG, la configuration de notre application sera représentée comme

un chromosome constitué de gènes. Chaque gène représente un service ou une fonctionnalité du téléphone (GPS, W-LAN, BLUETOOTH, WIFI...Etc.) Ce service se caractérise par :

1. **un état** : si ce service est activé où désactivé

Etat de gène { 0. s'il est désactivé ; 1. s'il est activé

2. **une utilité** : qui le spécialise, l'utilité c'est-à-dire l'efficacité de l'utilisation de ce service. par exemple l'utilité de GPS est meilleure que l'utilité de W-LAN à cause de son efficacité (GPS efficace que W-LAN)
3. **Une consommation de batterie** : c'est à dire combien le service consomme la batterie.

Le chromosome réel est formé en collectant simplement tous les gènes (tous les services).

***Tableau III .2*: les services avec leurs utilités et consommations de batterie**

num	gène	Consommation batterie (mA)	utilité	num	gène	Consommation batterie (mA)	Utilité
1	Local-gps	75	100	**11**	Limun-haute	65	75
2	Local-wlan	20	60	**12**	Limun-faible	25	32
3	Checkin-bluetooth	40	70	**13**	Son	27	60
4	Checkin-nfc	10	35	**14**	vibreur	15	20
5	Not-video-haute qual	60	90	**15**	2g	70	25
6	Not-video-moyenne qual	45	55	**16**	3g	50	50
7	Not-video-faible qual	30	35	**17**	4g	25	80
8	Not-img-hd	15	23	**18**	msglong	22	20
9	Not-img-brut	10	15	19	Msg moyen	12	13
10	Not-text	5	14	20	Msg petit	4	6

5.2. La population initiale

La population initiale est considérée comme un point de départ pour l'AG. Les générations futures sont au fur et à mesure améliorées afin d'accéder aux meilleures solutions, en appliquant les opérateurs génétiques.

Une bonne initialisation aide d'une manière considérable l'évolution du processus. On trouve plusieurs mécanismes pour générer une population initiale. Parmi ces mécanismes : le tirage aléatoire, les heuristiques ou une combinaison de solution heuristique et l'aléatoire. Dans notre approche, on génère la population initiale de façon aléatoire uniforme (l'état de service est fait par la fonction Random qui donne des services aléatoires parmi les service existe : 000, 001,010…etc.)

Voici un exemple de population de 3 individus sur 20 gènes (7 service)

	1	**2**	**3**	**4**	**5**	**6**	**7**
Chromosome 0	01	10	000100	00	10	010	001
Chromosome 1	00	10	100000	10	01	100	100
Chromosome 2	10	01	000010	01	01	001	010

5.2.1 Description des services existent :

Pour garder la structure correcte d'un individu et éviter de tomber dans une configuration qui n'a aucun sens nous avons effectué un classement des services et un codage d'état des services de la manière suivante :

Les services sont choisis d'une manière aléatoire

Service 1 : Localisation

Etat	**Désactivé**	**GPS**	**WLAN**
Codage	00	10	01

Service 2 : Envoyer fichier

Etat	**Désactivé**	**BLUETOOTH**	**NFC**
Codage	00	10	01

Service3 : Notification

Etat	Désactivé	VIDEO HAUT QUALITE	VIDEO MOYENNE QUALITE	VIDEO BAS QUALITE	IMAGE HD	IMAGE BRUTE	TEXT
Codage	000000	100000	010000	001000	000100	000010	000001

Service 4: Luminosité

Etat	Luminosité haute	Luminosité bas
Codage	10	01

Service 5: Mode audio

Etat	Son	Vibreur
Codage	10	01

service 6 : Donnée Mobile

Etat	Désactivé	2G	3G	4G
Codage	000	100	010	001

Service7 : Message

Etat	Désactivé	Message LONG	Message moyenne	Message petit
Codage	00	10	01	

5.3. La fonction de fitness :

Pour déterminer la meilleur solution dans l'espace de solution (trouver l'individu le plus proche au l'optimum) on a besoin de fonction qui s'appelle fonction de fitness .cette dernière

utilise des critères qui spécialise notre problème d'optimisation pour calculer l'optimalité de chaque individu.

Dans notre système nous sommes concentrés sur les deux critères suivants :

- Critère 1 : la minimisation de la consommation de batterie.
- Critère 2 : la maximisation d'utilité

La fonction de fitness *f(X)* est définit comme étant la somme pondérée de ces deux objectifs (critères), caractérise par :

- **L'utilité U (i)** : l'utilité de service ou l'efficacité d'attendre le but.
- **Consommation C(i)** *:* consommation de la batterie d'un service.
- **les poids Wi :** sont présentés sous forme de nombres de types réels associés à chaque critère, et qui traduisent leurs importances relativement à l'exigence de l'utilisateur. La somme de ces poids égal à 1.

On n'a choisie que critères 1 à un poids (W1) de 75 % et critères 2 à un poids de 25%. Parce que l'importance est de minimiser la consommation de la batterie plus que de maximisé l'utilité

Tableau III .3: Attributs de poids

Dimensions de qualité	Pourcentage
Consommation de la batterie(CM)	75%
Utilité de service (US)	25%

- **Etat E(i) :** si le gène est actif (1) sinon (0)

. La fonction fitness *f (X)* pour le chromosome X de N gènes peut être exprimée comme si :

Si on minimise un critère on multiple par (-1) Sinon on multiple par (1)

$$f(X) = \sum_{i=1}^{N} E(i)\ (-W_1 * Ci + W_2 * Ui)$$

- Dans cette fonction on peut trouver des valeur fitness chromosome négative qui cause des problèmes dans la prochaine état de l'algorithme génétique (sélection) .donc on vas normalisé avec une formule pour éviter les possibilité d'avoir une valeur négative de fitness .

La formule est expliquée en détail dans la section suivante :

5.3.1. Matrice de limites

En utilisant des critères de qualité, nous définirons une matrice de limites ML. ML donne une plage de valeurs acceptable pour chaque critère et son poids. Chaque colonne représente un critère, représentant respectivement la valeur minimale acceptable, puis la valeur maximale acceptée et enfin son poids.

$$ML = \begin{pmatrix} Qumin & Qcmin \\ Qumax & Qcmax \\ w1 & w2 \end{pmatrix}$$

5.3.2. La fonction de fitness normalisée

On a deux critères pour normalisation Qu (critère d'utilité), Qc (critère de consommation de batterie).Puisque Les deux caractéristiques de la qualité ayant des normes différentes, nous devons les normaliser sur la même échelle dans [0.1].

Supposons que Q(*S*) = X(*S*) ∪Y(S) où X(*S*) = $Q_U(S)$ un ensemble de critères de qualité positifs et Y (*S*) = Qc(*S*) ensemble de critères de qualité négatifs. On note une meilleure qualité de X par des valeurs plus élevées calculées par l'équation et une meilleure qualité de Y par des valeurs plus faibles. Les valeurs des critères de qualité de X (respectivement Y) sont normalisées à X' (respectivement Y') comme :

$$X_i'(S) = \left\{ \frac{X_i(S) - Qu_i^{min}}{Qu_i^{max} - Qu_i^{min}} \right. \qquad \text{i=1}$$

$$Y_j'(S) = \left\{ \frac{Qc_j^{max} - Y_j(S)}{Qc_j^{max} - Qc_j^{min}} \right. \qquad \text{j=2}$$

La fonction fitness *f* est calculée par une simple somme pondérée de valeurs des critères normalisées :

$$f(S) = \sum_{\forall i | Qi \in X} W_i * X'_i(S) + \sum_{\forall j | Qj \in y} W_j * Y'_j(S)$$

5.3.3. Impacts sur les dimensions de qualité pour chaque stratégie :

Pour relier les services en cours d'exécutions que nous appelons l'état actuel avec les solutions des métaheuristiques qu'on souhaite obtenus on doit alors Spécifie l'importance de chaque chromosome de la population. il faut d'abord calculer la différance entre les services de ces chromosomes avec les services de chromosome de l'état actuel.

Ce différence est symbolisé par (k), il montre le degré de changement qu'on veux le faire par augmentation de qualité ou bien par Diminution de qualité.

Donc :

Si le k montre que le service de l'état actuel est mieux que l'autre il faut diminuer la qualité de service par l'ajoute a son utilité (-8 % * k) et a son consommation (-20%*k)

Si non on augmente la qualité de service

***Tableau III .4*:** Impacts sur les dimensions de qualité pour chaque stratégie

Stratégie	US	CM
Augmenter qualities service	+8 %*k	+20 %*k
diminué qualities service	-8%*k	-20%*k

5.4. Les opérateurs génétiques

5.4.1. La sélection

Dans notre approche on utilise une stratégie de sélection par roulette (Roue de loterie) cette sélection consiste à dupliquer chaque individu (chromosome) C_i proportionnellement à la valeur de la fonction d'adaptation (fitness). Ainsi même les individus les plus faibles ont une chance de survivre.

Premièrement, on a calculé la somme des fitness des chromosomes de population et on obtient un intervalle de [0, somme fitness], puis on utilise un nombre aléatoire dans cette intervalle.

Et on commence à calculer la somme des fitness des chromosomes et on compare a chaque fois avec le nombre aléatoire jusqu' on obtient une somme supérieur ou égale ce nombre et on garde le dernier chromosome qu'on a calculé.

Exemple :

4 chromosomes de population avec leur fitness :

Chromosome	Fitness
Chromosome 1	0.5
Chromosome 2	0.2
Chromosome 3	0.6
Chromosome 4	0.3

On prend x=1 tel que x est un nombre aléatoire dans l'intervalle [0, 1.5]

Et on commence la comparaison :

X>0.5 puis x>0.5+0.2 puis x<0.5+0.2+0.6

On arrête la comparaison et on choisie le chromosome 3.

5.4.2. Le croisement uniforme

Par la méthode de sélection qu'on a faite précédemment, deux parents ont été choisie pour obtenir un enfant chromosome par l'operateur génétique croissement.
Il existe plusieurs méthodes de croissement, parmi eux on utilise le croisement par pourcentage, c'est-à-dire lorsqu'on a deux parents C_1 et C_2 on va faire le croisement de pourcentage de 75%, le chromosome enfant sera composé de 75% de C_1 et le reste de C_2.

5.4.3. La mutation uniforme

Apres l'étape du croisement, la prochaine génération est construite à partir de la mutation de tous les chromosomes de l'ancienne génération.

La mutation du chromosome se fera par service à l'aide d'un pourcentage de 25% déterminer qui ne pas doit être dépassé.

5.4.4. Les critères d'arrêt

Le critère d'arrêt peut être fixé en imposant un nombre maximal de générations (Max Gen).On estime alors que l'algorithme a convergé et que l'individu de plus forte performance en une des générations correspond à la solution au problème. Dans notre cas nous avons choisis MaxGen = 100 comme un critère d'arrêt.

6. Conception de l'approche basée sur Essaime particulaire :

Comme deuxième solution nous proposons d'implémenter notre approche d'auto adaptation en utilisant l'optimisation par essaimes particulaires.

6.1 .Codage de l'essaime particulaire :

Pour le codage de particule proposé (Chromosome en AG), l'initialisation d'essaim initiale (population en AG), et la fonction de fitness se fait de la même manière que l'algorithme génétique. La seule différence entre ces deux algorithme est que l'algorithme génétique base sur les operateurs génétique (sélection, croisement, mutation) et l'essaime particulaire fonctionne de la manière suivante :

6.2. Principe de Fonctionnement de l'essaime particulaire :

D'une manière générale Dans l'espace de recherche (composé de particules) de dimension **D**, une particule ***i*** de l'essaim est représentée par son vecteur position et par son vecteur vitesse.

Dans notre approche on représente ces deux vecteur par fitness (Xi) et valeur d'avancement de particule (Vi) (numéro aléatoire dans la premier essaime) comme suit :

$$\overrightarrow{Xi} = (X_{iD}, X_{iD}, \dots, X_{iD})$$

$$\overrightarrow{Vi} = (V_{iD}, V_{iD}, \dots, V_{iD})$$

Etape 1 : choix de Gbest :

Choisir le chromosome (particule) qui a la grande valeur de fitness parmi les chromosomes de la population (essaime) et on l'appelle Global Best(Gbest)

Etape 2 : mise à jour de vélocité

Les autres particules doivent suivre cette global best qui représente la meilleure solution temporairement. Pour se faire, il faut que chaque particule change sa position (fitness) pour être proche ou meilleure que Gbest, en comparent chaque position de particule par la position du Gbest. Cette comparaison faite entre les fitness des services de particule et les fitness des services de globale best. Ce changement se base sur la fonction de vélocité (la valeur d'avancement de chaque service) qui est définit comme suit :

$$\boldsymbol{V_{i;j}^{t+1} = wv_{i,j}^{t} + C_1 r_{1\,i,j}^{t}\left[P_{best\,i,j}^{t} - x_{i,j}^{t}\right] + C_2 r_{2\,i;j}^{t}\left[G_{best}^{t} - x_{i,j}^{t}\right], j \in \{1, 2, \dots, D\}}$$

Tel que :

- ***w*** est une constante, appelée coefficient/pourcentage d'inertie.
- ***c1,*** **c*2*** sont deux constantes, appelées coefficients/pourcentage d'accélération.
- ***r1, r2*** sont deux nombres aléatoires tirés uniformément dans **[0,0.5]**, et ce à chaque itération ***t*** et pour chaque dimension ***j***.
- **Pbest** est la meilleure position (fitness) obtenue de chaque chromosome (particule) dans chaque itération.

Etap3 : évaluer la position du service

Après le calcul de vélocité de chaque service, une nouvelle position (nouveau service) a été obtenue et calculer comme suit :

$$\boldsymbol{X_{i;j}^{t+1} = X_{i;j}^{t} + v_{i,j}^{t+1}, j \in \{1, 2, \dots, D\}}$$

Tel que :

$X_{i,j}^{t+1}$: Nouvelle position obtenue ou bien fitness de nouveau service

$X_{i,j}^{t}$: Ancien position de service ou bien fitness de l'ancien service

$V_{i,j}^{t+1}$: Velocité calculé

Etape 4 : mise à jour de Pbest et Gbest

A la fin du déplacement des particules dans une itération donnée, les nouvelles positions sont évaluées et les deux vecteurs ***Pbest*** et ***Gbest*** sont réindexés, conformément aux deux équations suivantes dans le cas d'une maximisation d'une fonction objective

$$\textbf{Si} \quad X_{i,j}^{t+1} < Pbesti(t) \quad : \quad Pbesti(t+1) = Pbesti(t)$$

$$\textbf{Sinon} \quad : Pbesti(t+1) = X_{i,j}^{t+1}$$

Etape 5 : fonction d'arrêt et récupération de solution finale :

L'algorithme répète les étapes 1 à 4 jusqu'à ce que certaines conditions de terminaison soient remplies, telles qu'un nombre prédéfini d'itérations ou l'absence de progression pour un certain nombre d'itérations. Une fois terminé, l'algorithme rapporte les valeurs de Gbest comme solution.

6.3. Exemple de travail :

Pour comparer la fitness de chromosome Gbest avec les autres fitness des chromosomes, il faut comparer les fitness des services de ces deux derniers.

Dans cet exemple nous avons comparé le service notification dans le chromosome Gbest (plus précisément Image brut) avec le service notification dans le chromosome Pbest (plus précisément Image HD).

Initialisation :

Fitness image HD =0.65 et fitness image brut = 0.68.

$\mathbf{P_{best}}$=X (Fitness image HD) au début.

$\mathbf{G_{best}}$ = fitness image brute.

Vi = nombre aléatoire de [0 ; (Fitness image HD + fitness image brut)/2]

$$\boldsymbol{C_1 = 0.5\ ; C_2 = 0.5\ ;\ R_1 = 0.5\ \ ; R_2 = 0.5}$$

Fonction de vélocité calculée comme suit (étape 2) :

$$\boldsymbol{V_{i+1} = V_i + C_1 R_1 (P_{best} - X_i) + C_2 R_2 (G_{best} - X_i)}$$

$$\boldsymbol{V_{i+1} = .015 + 0 + (0.25 * 0.03) = 0.022}$$

Fonction de position ou nouvelle fitness calculée comme suit (étape 3) :

$$\boldsymbol{X_{i+1} = X_i + V_{i+1} \quad => \quad X_{i+1} = 0.65 + 0.022 = 0.672}$$

mise à jour de *Pbesti* et *Gbest* (Etape 4):

$$\mathbf{P_{best}} = X_{i+1} = 0.672 \ \ et \quad \mathbf{G_{best}} = 0.68$$

7. Conclusion :

Dans ce chapitre nous avons présenté le cas d'étude de notre application, ainsi que le détail des approches utilisées (algorithme génétique et essaime particulaire), nous avons expliqué son principe de fonctionnement c a d le codage de chromosome (particule) et la fonction de fitness et les méthodes d'évaluations pour chaque algorithme.

Le chapitre suivant sera donc consacré à exposer les tests ainsi que les résultats obtenus de nos expérimentions.

Chapitre *IV* : Implémentation de l'application et tests

1. Introduction :

Précédemment nous avons présenté notre solution pour l'optimisation de la planification de l'auto adaptation ainsi que les techniques et les mécanismes utilisés pour avoir un bon résultat.

Dans ce chapitre nous allons présenter l'environnement et les outils de travail utilisés pour l'implémentation de notre approche, par la suite décrire les tests et les résultats obtenus.

2. Les outils utilisés :

La technologie offre plusieurs outils pour la réalisation des projets et des approches de fin d'étude, on a utilisé certains parmi eux selon nos objectifs et nos besoins.

2.1. L'environnement de développement NetBeans :

L'outil de développement NetBeans est un environnement de développement (un outil pour les programmeurs) pour écrire, compiler, déboguer des programmes. Il est écrit en Java mais peut supporter d'autres langages de programmation comme Python, C, C++, XML et HTML. placé en open source par Sun en juin 2000 sous licence CDDL (Common Développement and Distribution License), sans aucune restriction sur son utilisation. Il existe plusieurs versions de NetBeans, dans notre cas on a utilisé NetBeans IDE 8.2

2.2 JavaFX :

JavaFX est une bibliothèque Java utilisée permettant de créer et de fournir des applications de bureau, ainsi que des applications Internet riches. Les applications écrites à l'aide de cette bibliothèque peuvent être exécutées de manière cohérente sur plusieurs plates-formes. Les applications développées à l'aide de JavaFX peuvent s'exécuter sur divers périphériques tels que les ordinateurs de bureau, les téléphones mobiles, les téléviseurs, les tablettes, etc.

Pour développer des applications GUI (Graphical User Interface) utilisant le langage de programmation Java, les programmeurs s'appuient sur des bibliothèques telles que Advanced Windowing Toolkit et Swing. Après l'avènement de JavaFX, ces programmeurs Java peuvent désormais développer des applications d'interface graphique avec un contenu riche.

2.3. SceneBuilder

JavaFX fournit une application nommée Scene Builder. En intégrant cette application dans les IDE tels que Eclipse et NetBeans, les utilisateurs peuvent accéder à une interface de conception par glisser-déposer, utilisée pour développer des applications FXML (tout comme les applications Swing Drag & Drop et DreamWeaver).on a utilisé dans notre cas la version SceneBuilder 8.5.0.

2.4. *JFoenix*

JFoenix est une bibliothèque Java open source, qui implémente GOOGLE Matériel Design en utilisent des composant Java. Pour commencer à utiliser JFoenix, il suffit de le télécharger depuis GitHub.on a utilisé dans notre cas la version *JFoenix 8.0.8.*

3. Présentation de l'application :

Pour faire un scenario de teste de notre approche nous avons implémenté une application, que nous allons présenter son fonctionnement dans les sections qui suivent :

3.1. Interface d'accueil :

Lorsque l'application est exécutée, une interface de départ (d'accueil) apparaît à l'écran pour présenter le thème de notre projet, l'interface contient quatre buttons (simulation avec AG, simulation avec PSO, simulation sans Métaheuristiques, Quitter)

Figure IV.1 : Interface d'accueil

3.2. Interface des deux algorithmes :

Afin d'examiner le comportement de notre algorithme, et examiner la convergence de la fonction de fitness, nous devons passer par l'étape de la configuration pour choisir les paramètres de simulation de l'AG et OEP. Notre application permet la configuration à partir de l'interface d'AG et OEP présenté dans la Figure. ***IV.2*** et la Figure ***IV.3***

3.2.1 Les paramètres utilisés dans le test :

Table *IV.1* : paramètres de l'algorithme génétique

Paramètre	Valeur
Population	8
Pourcentage De Mutation	0.25
Pourcentage de Croissement	0.75
Poids d'utilité	0.25
Poids de consommation	0.75
Nombre de générations	100

Figure IV.2 : Interface de paramètre d'Algorithme Génétique

Table *IV.2*: paramètres de l'algorithme d'essaime particulaire

Paramètre	Valeur
Population	8
Paramètre cognitive C1	0.5
Paramètre cognitive C2	0.5
Poids d'utilité	0.25
Poids de consommation	0.75
Nombre d'itérations	100

Figure IV.3 : Interface de paramètre de l'Essaime Particulaire

3.3. Interface simulation

Notre approche doit passer par l'étape de configuration pour choisir les paramètres de simulation comme : activé service, ajouter son durée de vie (combien de temps en travail par ce service) et tableau qui affiche le temps d'ajoute, durée de vie et nom de service.

Remarque : la simulation est un teste qui fait par l'utilisateur pour savoir la validité de l'application par l'obtention des graphes comme résultats.

Figure IV.4 : Interface de simulation

4. Exemple de travail :

4.1. Etat actuel :

Représente les services exécutés par l'utilisateur sur son Smartphone, chaque service a un temps d'arrivé et une durée de vie (Combien l'utilisateur utilisera ce service).

Table *IV.3* : les services ajoutés avec leurs durées de vie

Nom de service	Durée de vie (minute)
GPS	30
Bluetooth	30
Vidéo Haute Qualité	30
Luminosité Haute	30
Son	30

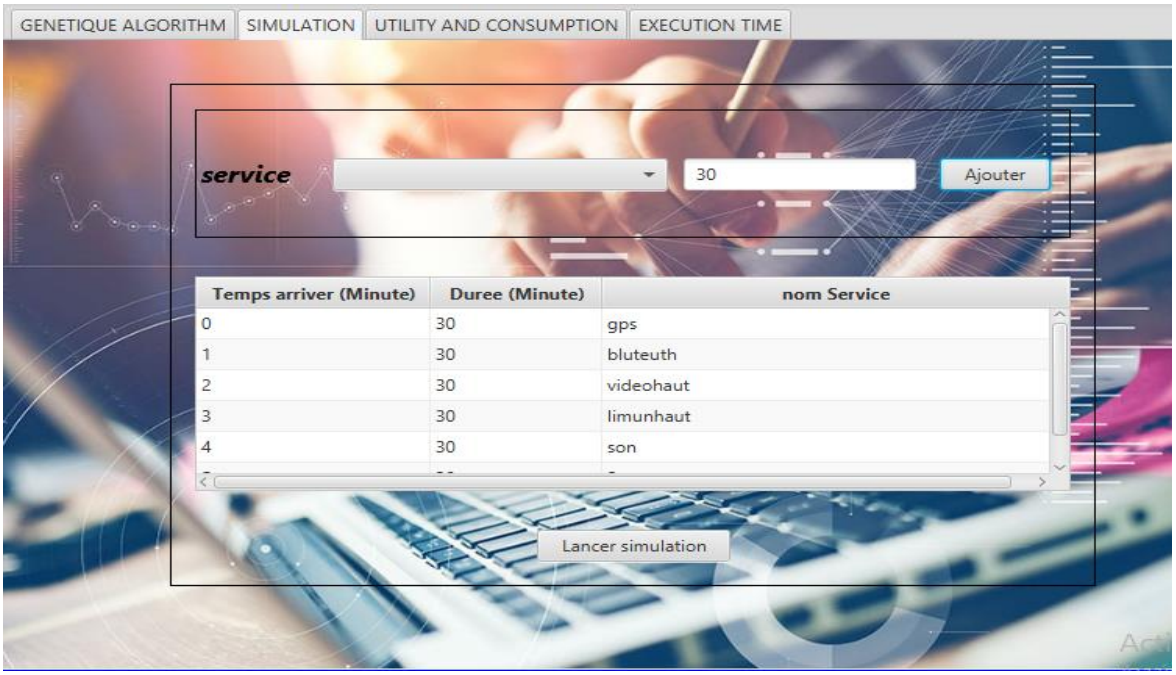

Figure IV.5 : Etat actuel

4.2. Simulation générale par les deux métaheuristique (AG et PSO) :

Au début, nous avons activé tous les services de meilleure qualité de notre Smartphone donc le niveau de la batterie à commencer de diminué jusqu'il arrive au seuil (60 % de niveau de la batterie)

Ca permet d'exécuter les metaheuristiques automatiquement pour trouver une nouvelle configuration (nouveaux services) moine consommation que les premiers services.

4.3. Chargement de batterie :

La charge de la batterie se fait lorsque le niveau de la batterie inferieur a 500 mAh, cette opération permet d'utiliser tout les services existe dans notre système.

Tableau IV.4 : Etat actuel après la simulation de l'algorithme génétique :

Nom de service	Durée de vie (minute)
GPS	**30**
Bluetooth	**30**
Vidéo Haut qualité	**30**
Luminosité haut	**30**

Figure IV.6 : les services activés durant le chargement de batterie

4.4. Deuxième simulation :

Lorsque li niveau de la batterie à augmenter a un niveau donné, on débranche le chargeur et on commence la deuxième simulation avec les mêmes services qu'on a utilisé dans le chargement de la batterie. On obtient le résultat affiché dans ***Figure IV.8***

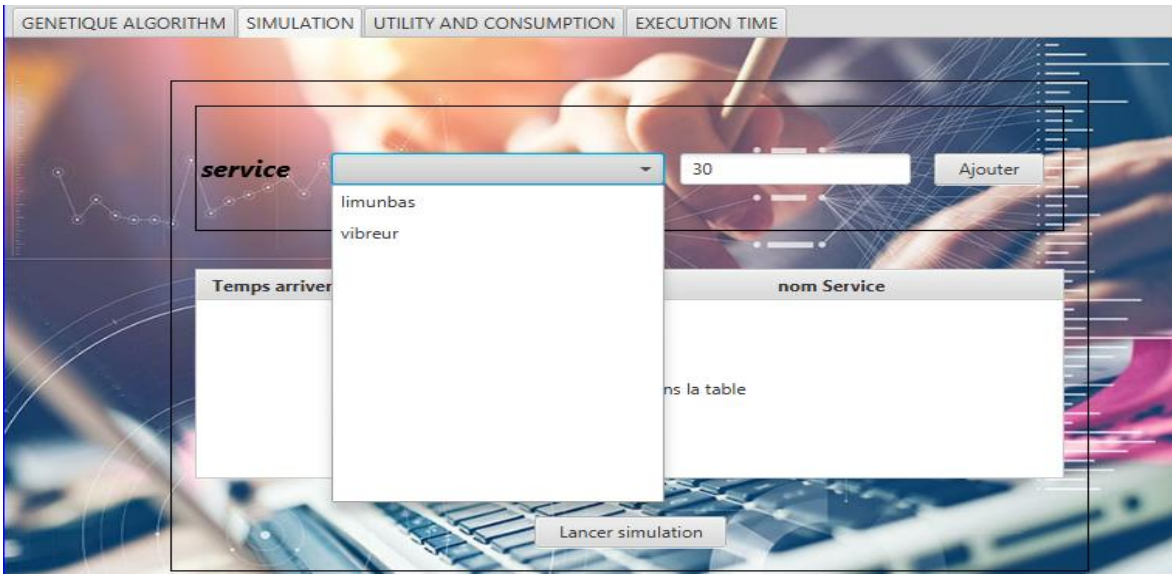

Figure IV.7: résultat obtenue après l'exécution de la deuxième simulation

5. Interface des courbes graphiques :

Après chaque simulation, notre application permet de générer des graphes à l'aide de la bibliothèque JLineChart et Barchart . Ces courbes permettent de voir les caractéristiques et les résultats de chaque simulation.

Les testes ont été effectués sur un ordinateur portable DELL sous Windows 10 professionnel .avec un processeur Intel(R) Core (TM) i5-5200u CPU@2.20GHz et une RAM de 6.00 Go

5.1. Graphe de durée de vie de batterie Sans ME:

Ce graphe représente le niveau de la batterie, la consommation et l'utilité des services par temps sans utiliser les métaheuristiques. Cette simulation a été effectuée par différent étapes.

Etap1 : de 0 à 7 minutes, l'utilité et la consommation des services qu'on a activés sont respectivement 4400 et 3200.

Etap2 : de 7 à 16 minutes, en minute 7 on a branché le chargeur qui nous permet d'activé tout les services de meilleure qualité qu'on souhait utilisé. Ce qui augmente le niveau de la batterie.

Etap3 : de 16 à 27 minutes, le niveau de la batterie commence a diminué depuis la minute 16. On remarque que le niveau de la consommation et l'utilité des services reste stable pendant les trois étapes.

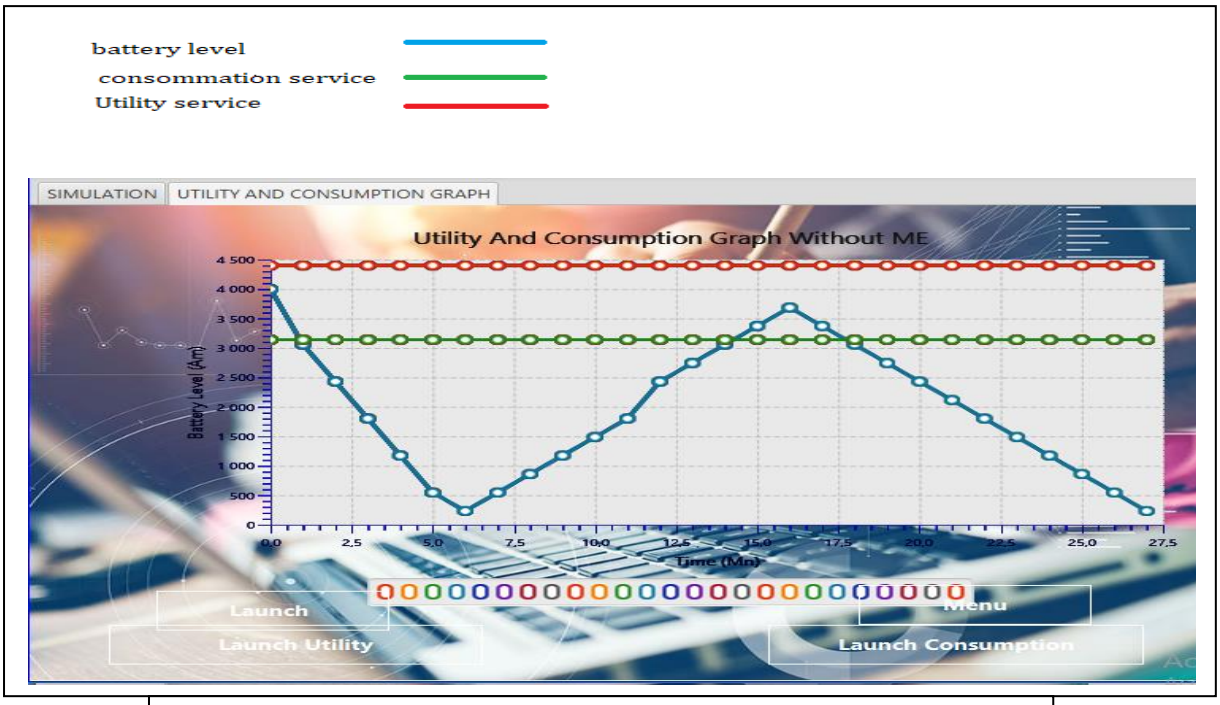

Figure IV.8: Graphe de durée de vie de batterie sans ME

5.2. Graphe de simulation des services de l'étude de cas avec AG :

Ce graphe représente le niveau de batterie ainsi que le niveau de la consommation et l'utilité des services activés. Cette simulation a été effectuée par différent étapes.

Etap1 : de 0 à 17 minutes, l'utilité et la consommation des services qu'on a activés sont respectivement 4400 et 3200.

La reconfiguration a été faite deux fois par l'exécution de l'algorithme génétique. La première dans la minute 2, le niveau de la consommation et l'utilité sont respectivement 2000 et 1100.

La deuxième exécution dans la minute 11, le niveau de la consommation et l'utilité sont respectivement 1700 et 800.

Etap2 : de 17 à 26 minutes, a la minute 17 on a branché le chargeur qui nous permet d'activé tout les services de meilleure qualité qu'on souhait utilisé. Ce qui augmente le niveau de la batterie, la consommation et l'utilité.

Etap3 : de 26 à 51 minutes, La reconfiguration a été faite quatre fois par l'exécution de l'algorithme génétique. Le niveau de la batterie, la consommation et l'utilité sont diminué pendant cette étape.

Figure IV.9: simulation des services de l'étude de cas avec AG

5.3. Graphe de simulation des services de l'étude de cas avec PSO :

Ce graphe représente le niveau de batterie en utilisant ainsi que niveau de la consommation de la batterie et l'utilité .cette simulation a été effectuée par différent étapes.

Etap1 : de 0 à 13 minutes, l'utilité et la consommation des services qu'on a activés sont respectivement 4400 et 3200.

La reconfiguration a été faite deux fois par l'exécution de l'algorithme d'essaime particulaire. La première dans la minute 2, le niveau de la consommation et l'utilité sont respectivement 2700 et 2000.

La deuxième exécution dans la minute 6 le niveau de la consommation et l'utilité sont respectivement 1500 et 1000.

Etap2 : de 13 à 21 minutes, a la minute 13, on a branché le chargeur qui nous permet d'activé tout les services de meilleure qualité qu'on souhaite utilisé. Ce qui augmente le niveau de la batterie, la consommation et l'utilité.

Etap3 : de 21 à 43 minutes, La reconfiguration a été faite quatre fois par l'exécution de l'algorithme d'essaime particulaire. Le niveau de la batterie, la consommation et l'utilité sont diminué pendant cette étape.

Figure IV.10: Graphe de simulation des services de l'étude de cas avec PSO

5.4. Graphe de temps d’exécution PSO et AG :

Le temps d'exécution de l’algorithme génétique est compris entre 3400 et 2900 (ms) par une simulation.

***Figure IV.11**:* Graphe de temps d’exécution AG

Par contre : Le temps d'exécution de l’algorithme d’Essaime particulaire est compris entre e1600 t 1200 (ms) par une simulation.

***Figure IV. 12**:* Graphe de temps d’exécution PSO

6. Discussions :

Les tests réalisés précédemment avec l'algorithme génétique (AG) et l'algorithme d'essaim particulaire (PSO) ont montré une amélioration significative de la durée de vie de la batterie du Smartphone. Nous avons trouvé que la durée de vie augmente de 26% et 46% en utilisant respectivement les algorithmes PSO et AG. L'écart de 20% entre les deux durées de vie indique que la reconfiguration des services par AG est plus efficace et moins gourmande en consommation d'énergie. Cependant l'exécution du PSO est plus rapide, son temps d'exécution pour trouver une solution est presque de moitié celui de AG (a cause de ces opération : Sélection, Croissement et mutation).

.

7 .Conclusion

Ce chapitre présente l'application développer , l'environnement et les outils de travail utilisé ainsi que les graphs de différents tests (utilité , consommation de la batterie et la durée de vie de la batterie) effectué et les résultats obtenue qui montre que l'utilisation de notre approche a augmenter la durée de vie de la batterie plus que 46 % avec AG et 26% avec PSO.

Conclusion générale

Le projet que nous avons présenté dans ce mémoire consiste à la conception et le développement d'une application d'Optimisation de la planification de l'auto-adaptation des systèmes par l'utilisation des métaheuristiques

Il s'articule autour de l'autoadaptation des applications mobiles, tout en représentant l'optimisation de la planification de l'auto adaptation des systèmes mobiles par des métaheuristiques. Cette application utilise des techniques d'optimisation ainsi que les deux méthodes basées sur les métaheuristiques(algorithme génétique et essaimes particulaires).

Lors de la réalisation de ce projet, nous avons commencé par une étude bibliographique sur les systèmes auto adaptatif**.** Par la suite, nous avons mené une étude sur les métaheuristiques en se basant principalement sur AG et PSO que on a utilisé dans notre approche l'auto adaptation des systèmes basée sue deux métaheuristiques qui permettent à choisir la meilleure reconfiguration.

Suite à cela, nous avons passé à représenté la conception et l'implémentation de l'approche adoptée et les solutions proposées de l'auto adaptation basé sur deux métaheurstiques.

Finalement, nous avons présenté les tests et les validations de l'application proposés, les résultats obtenus sont satisfaisante en terme de durée de vie de batterie du téléphone.

En perspective, nous recommandons de traiter les thématiques suivantes :

1. poursuivre le développement de notre approche d'adaptation des systèmes en utilisant d'autres algorithmes méta-heuristiques.
2. généraliser notre application à d'autres problèmes réels.

Bibliographie :

[1] .Franck Chauvel, Méthodes et outils pour la conception de systèmes logiciels auto-adaptatifs,2008

[2] Mohammad Reza Nami, Koen Bertels, Autonomic Computing Systems: Issues and Challenges, 03 June 2014

[3] Giovanna Di Marzo Serugendo,Marie-Pierre Gleizes ,Anthony Karageorgos, Self-organising Software,2011

[4]. Malik Jahan Khan a,b,⇑ , Mian Muhammad Awais a , Shafay Shamail a , Irfan Awan, An empirical study of modeling self-management capabilities in autonomic systems using case-based reasoning ,2011.

[5] Ivan Mistrik Nour Ali Rick Kazman John Grundy Bradley Schmerl, Managing Trade-offs in Adaptable Software Architectures ,

[6].Fabiano Dalpiaz, Paolo Giorgini, and John Mylopoulos, An Architecture for Requirements-Driven Self-reconfiguration ,2009

[7] R. T. Rockafellar, "FUNDAMENTALS OF OPTIMIZATION", University of Washington Seattle ,2007

[8]. Gustavo G. Pascual ∗, Mónica Pinto, Lidia Fuentes, Self-adaptation of mobile systems driven by the Common Variability Language,2013

[9] Catalin Boja, Zamfiroiu Alin, Self-Healing for Mobile Applications ,juin 2012

[10] Stefania Montani , Cosimo Anglano,Achieving self-healing in service delivery software systems by means of casebased reasoning, April 2008

[11] Eric Yuan , Sam Malek ,A Taxonomy and Survey of Self-Protecting Software Systems, 2012

[12] Jin-Kao Hao, Philippe Galinier, Michel Habib, "Méthaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes», Revu d'Intelligence Artificielle, Université d'Angers 1999.

[13] Ilhem Boussaid, Perfectionnement de métaheuristiques pour l'optimisation continue, Université Paris-Est, 2013.

[14] Amir Nakib. "Conception de métaheuristiques d'optimisation pour la segmentation d'images. Application aux images biomédicales. Informatique», Sujet de Thèses, Université Paris XII Val de Marne, 2007.

[15] Ahmed Nasreddine Benaichouche,Conception de métaheuristiques d'optimisation pour la segmentation d'images : application aux images IRM du cerveau et aux images de tomographie par émission de positons , 20 Apr 2015

[16] Ihsen SAAD1 , Fatma TANGOUR2 , Pierre BORNE3, Application des algorithmes génétiques aux problèmes d'optimisation,2002 .

[17] Naziha Ali Saoucha, Paramétrage des algorithmes génétiques pour l'optimisation de la QoS dans les réseaux radios cognitifs ,2010 .

[18] Dr. Mohamed Assaad HAMIDA, Introduction aux Méthodes de Contrôle Intelligent,2014/2015

[19] Romain Rouvoy, Paolo Barone , Yun Ding , Frank Eliassen , Svein Hallsteinsen , Jorge Lorenzo , Alessandro Mamelli , and Ulrich Scholz, MUSIC: Middleware Support for Self-Adaptation in Ubiquitous and Service-Oriented Environments,2009.

[20] Shang-Wen Cheng, Vahe V Poladian, David Garlan, Bradley Schmerl, Improving Architecture-Based Self-Adaptation Through Resource Prediction,2008.

[21]Gustavo G.Pascual ,Monica Pinto ,Lidia Fuentes, Self-adaptation of mobile systems driven by the Common Variability Language ,2014

[22] Bhawani Sankar Biswal, Anjali Mohapatra, Analogy of autonomic computing: An AIS (artificial immune systems),2014

[23]. Dr. Lemouari Ali, Introduction Aux Métaheuristiques, Université de Jijel, 2014.

[24]. Johann Dreo, Adaptation de la methode des colonies de fourmis pour l'optimisation en variables continues. Application en genie biomedical.

[25]. Dr. Mohamed Assaad HAMIDA, Introduction aux Intelligent, 2014/2015.

[26] Souquet Amédée Radet Francois-Gérard, ALGORITHMES GENETIQUES,21/06/2004