Projet semestriel du Master 1 ANDROIDE :

Evaluation de systèmes de sélection et de systèmes de mutation dans le but d'optimiser le déplacement d'un groupe de robots.

Leroy Cassandre et Lachiheb Sarah

Encadrant :
Nicolas Bredèche

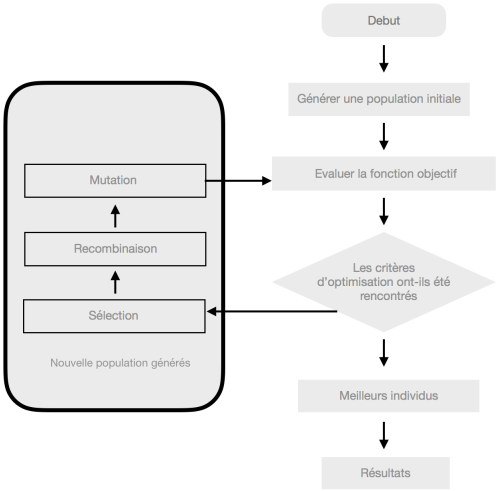
Remerciements :
Parham Shams

Table des matières

Première partie

Introduction

Dans les années 1960, John Holland étudie les systèmes évolutifs et, en 1975, il introduit le premier modèle formel des algorithmes génétiques (The canonical genetic algorithm AGC) dans son livre « Adaptation in Natural and Artificial Systems ». Les algorithmes génétiques appartiennent à la famille des algorithmes évolutionnistes. Leur but est d'obtenir une solution approchée à un problème d'optimisation. Les évolutionnistes sont une famille d'algorithmes dont la méthode s'inspire de la théorie de l'évolution pour résoudre des problèmes. Ils se basent sur le principe d'évolution d'une boucle générationnelle.



Le principe d'un algorithme génétique est de partir d'une population d'individu ayant une solution initiale à un problème. Chacun est évalué, pour lui attribuer une « note », appelée fitness. Plus la fitness d'une solution est haute, dans le cas d'une maximisation, et plus celle-ci est prometteuse. Les meilleurs individus sont ensuite sélectionnés, et se reproduisent. Deux facteurs de variation sont utilisés, le croisement entre deux individus et des mutations aléatoires sur le génome. La prochaine étape consiste à créer la nouvelle génération d'individus, jusqu'à une condition d'arrêt de temps ou de qualité de solution atteinte.

Le principe de codage de l'élément de population. Cette étape associe une structure de données et se place généralement après une phase de modélisation mathématique du problème traité. Sa qualité conditionne le succès de nos algorithmes. Les codages binaires ont souvent été utilisé mais nous utilisons ici un codages réels sont désormais largement utilisés.

Le mécanisme de génération de la population initiale. Ce mécanisme produit une population d'individus non homogène qui servira de base pour les

générations futures, ce qui est très important car cela conditionne la future valeur optimale trouvée. Nous avons donc choisie de répartir la population initiale sur tout le domaine de recherche.

Il nous faut aussi la fonction à optimiser. Celle-ci retourne une valeur appelée fitness. Ici, la fonction à optimiser est le parcours d'une carte, sans heurter d'obstacle. Maintenant que nous avons les bases de notre algorithme génétique, il nous faut les opérateurs permettant de diversifier la population au cours des générations et d'explorer. L'opérateur de croisement recompose les gènes d'individus existant dans la population, l'opérateur de mutation a pour but de garantir l'exploration de solution.

Enfin, les algorithmes génétique possèdent plusieurs paramètres qui jouent un rôle capital dans leur développement : taille de la population, nombre total de générations ou critère d'arrêt, probabilités d'application des opérateurs de croisement et de mutation.

Ce qui sera développé dans ce rapport, est dans une première partie, la recherche d'opérateurs de sélection, de mutation, et de taille de population, optimaux sur une tâche simple d'optimisation de la navigation par un robot autonome. Si dans la première partie, deux types de mutation sont mises en concurrence, dans la deuxième nous développons l'une des deux pour en extraire un maximum d'information. Ensuite nous tenterons de mettre ne oeuvre cette algorithme appliqué sur simulateur, à des robots réels.

Deuxième partie

Algorithme génétique et méthodes de sélection

1 Le principe de l'algorithme

L'algorithme de cette partie, qui correspond à un apprentissage en ligne non distribué, est exécuté sur chaque robot et évalue le comportement du robot lui même. Notre génome est constitué des poids d'un réseau de neurones monocouche. Les entrées sont : — 8 entrées continues correspondant aux capteurs translation et rotation. — 2 entrée continue servant de biais. Les sorties quant à elles sont au nombre de 2, une pour chaque moteur du robot, une pour la translation et l'autre pour la rotation.

On obtient 18 poids et donc un génome de 18 gènes. Toutes les valeurs sont normalisées entre -10 et 10. Représentation d'un génome, avec les 18 gènes

[-10;10]	[-10;10]	[-10;10]	[-10;10]	[-10;10]	[-10;10]	[-10;10]	[-10;10]	[-10;10]	[-10;10]	[-10;10]	[-10;10]	[-10;10]	[-10;10]	[-10;10]	[-10;10]	[-10;10]	[-10;10]

Notre fitness est :

$$f = v_{translation} * (1 - v_{rotation}) * (1 - mean(senseurs))$$

Elle permet de maximiser la vitesse de translation, minimiser la vitesse de rotation et enfin éviter les obstacles.

1.1 Nos méthodes de sélection

L'étape de la sélection sert à déterminer quels individus vont produire la futur génération. Elle peut se faire de deux manières, soit stochastique soit déterministe. La manière stochastique introduit une part de hasard, par inspiration du monde réel. Celle déterministe choisit les individus par des règles ou des formules mathématiques faites en amont. Tout au long de cette section, certains termes sont utilisés pour comparer les différents schémas de sélection : **Pression sélective** : les chances de sélection des plus performants par rapport aux plus faibles. **Biais** : Différence (absolue) entre la forme normalisée d'un individu et sa probabilité de reproduction. **Propager** : Gamme de valeurs possibles pour le nombre de progéniture d'un individu. **Perte de diversité** : Proportion d'individus d'une population non sélectionnée lors de la phase de sélection. **Intensité de sélection** : Valeur d'aptitude moyenne attendue de la population après l'application d'une méthode de sélection. **Variance de sélection** : Mesure de la dispersion des échantillons, après l'application d'une méthode de sélection.

1.1.1 Roulette Wheel (Roulette Wheel Selection ou RWS) (Goldberg, 1989)

Dans un premier temps, avec cette méthode, la plus populaire, stochastique, chaque individu à une chance proportionnelle à sa fitness d'être pris. Ainsi plus une fitness est grande, plus l'individu aura une forte probabilité d'être pris. L'esprit de roulette viens du fait que le choix se fait comme une sorte de roulette, avec un curseur et là où le curseur de stop, l'individu est choisis. Bien que cette méthode soit la plus répandu, elle possède plusieurs d'inconvenants :

- Il y a une forte variance. En effet, il y a une probabilité, s'il y a beaucoup de mauvais individu, que les fitness choisis soient toutes mauvaises. Si tout ou une grande partie des fitness choisis sont mauvaise, cela va fortement impacter la génération suivante. Dans la nature, ce sont les meilleurs individus, les plus forts et plus robustes, qui sont amenés à se reproduire. Par conséquent, ici on voit que le but de l'algorithme de type génétique n'est pas respecter.
- Il peut aussi arriver qu'un individu ait une fitness largement supérieur à toutes les autres réunis. Si l'individu est si majoritaire qu'il a une chance d'être tout le temps choisis, alors on perd en diversité. En effet, tous les enfants viendront alors du même parent. Aussi, sa fitness n'est peut être pas nécessairement bonne, mais peut être simplement bien par rapport aux autres. On risque de stagner vite vers un optimum local par pression sélective. Ainsi on passerait à côté d'un optimum global peut être bien

supérieur. Il y a des technique pour tenter de parer à ce problème, comme le scaling. Cela consiste à remettre à l'échelle si on remarque une trop grande disparité.

1.1.2 Sélection par rang (Ranking selection) (Whitley, 1989)

Avec les problèmes développés dans la partie précédente, on essaie de palier à ce problème avec la méthode de sélection par rang, qui dérive de cette dernière. La sélection par rang trie d'abord la population par fitness. Ensuite, chaque individu se voit associé un rang en fonction de sa position. Ainsi le plus mauvais aura le rang 1, le suivant prendra le rang 2, et ainsi de suite jusqu'au meilleur, qui aura le rang N (où N est la taille de la population). La sélection par rang d'un individu est la même que par roulette, mais les proportions sont en relation avec le rang plutôt qu'avec la valeur de l'évaluation.

Avec cette méthode de sélection, tous les individu ont une chance d'être sélectionnés. Cependant, elle conduit à une convergence plus lente vers la bonne solution car les meilleurs ne diffèrent pas énormément des plus mauvais.

1.1.3 Elitiste (elitism or elitist selection)

Cette méthode est quand à elle, déterministe, et assez simple. Elle consiste a prendre les n individus dont on a besoin pour créer la nouvelle génération, ici nous prenons que le meilleur. Cette méthode à le désavantage de mener de manière quasi certaine à une convergence prématurée vers un optimum local. Il n'y a pas de diversité, ni de variance, et donc le peu de diversité que l'on peut introduire est due à la mutation ou au croisement.

On constate qu'aucune des trois méthodes proposées n'est convaincante, on va donc essayer de proposer des méthodes qui permettent d'être plus diversifiée et plus ressemblante à la génétique.

1.1.4 Les tournois (Tournament selection ou TS)(Miller et Goldberg, 1995)

Dans la population, on sélectionne deux individus, on les fait se rencontrer et celui avec la meilleur fitness l'emporte. On répète ce processus N fois de manière à avoir les i individus nécessaire à la conception de la nouvelle génération.

Plus on diminue i et plus la pression de la sélection est faible, et à l'inverse, plus on l'augmente, plus elle est forte, ainsi l'intensité de la sélection est elle aussi plus forte. La variance de cette méthode reste assez élevée, et on ne peu que remarquer sa ressemblance avec la nature, ce qui reste le but d'un algorithme génétique.

1.1.5 Uniforme

La sélection uniforme a une variance forte mais garde un bon niveau de diversité. La sélection se fait aléatoirement, uniformément et sans intervention de la valeur de la fitness. Chaque individu a donc une probabilité 1/N d'être sélectionné, où N est le nombre total d'individus dans la population.

1.1.6 Sélection universelle stochastique(Stochastic Universal Sampling ou SUS) (Baker, 1987)

Dans son article "Reducing Bias and Inefficiency in the Selection Algorithm" James E.Baker explique l'échantillonnage universel stochastique, avec un biais nul et une variance minimum.

Il utilise aussi une roulette partagée en autant de d'individus, proportionnellement aux valeurs de fitness. Ici, des pointeurs espacés, de manière équidistante, sont placés sur la ligne autant qu'il y a d'individus à sélectionner, disons i le nombre d'individu à sélectionner. Alors la distance entre les pointeurs est 1/i, et la position du premier pointeur est donné par un nombre généré aléatoirement entre [0, 1/i].

Il n'y a aucun moyen de connaitre l'opérateur de sélection le plus adapté à notre problème, par conséquent, il nous faudra faire des test empirique dans le but de sectionner la meilleur méthode de sélection.

1.2 Nos méthodes de mutation

L'opérateur de mutation à pour but d'introduire de la nouveauté au sein de la population, pour permettre d'émergence de nouvelles solutions potentielles, de nouveaux axes de découverte, éventuellement plus pertinents.

Cette opération consiste à choisir certain gène pour le faire muter, et la probabilité qu'un gène soit touché par une mutation s'appelle le taux de mutation. S'il est trop élevé, les bonnes solutions risquent d'être modifiées, et de devenir moins bonnes, s'il est trop bas, on se prive de potentielles meilleures solutions.

1.2.1 La mutation Gaussienne

Avec cette mutation nous devons ajouter une valeur aléatoire gaussienne à tous les gènes du génome de l'individu choisi. Si elle tombe en dehors des limites inférieures ou supérieures spécifiées pour nos gènes, la nouvelle valeur du gène est coupée.

1.2.2 La mutation uniforme limitée

Ici nous devons remplacer la valeur du gène choisi par une valeur aléatoire uniforme choisie entre les limites supérieures et inférieures spécifiées, elles sont celles de nos gènes. Aussi, le taux de mutation dans ce cas a été élevé à 1/N

jusqu'à 1, avec N le nombre d'individus.

Après la sélection, notre algorithme choisis d'instaurer de la diversité par un croisement en un point.

1.2.3 Les méthodes de croisement

Le croisement permet de créer un nouvel individu à partir de ses deux parents, en mixant les informations génétiques. Une position de croisement est choisie au hasard puis tous les gènes situés avant ce point viennent du premier parent, et ceux situés après, du deuxième. Il est discret car il garde les valeurs telles quelles.

Le croisement sera testé à part, pour pouvoir tester son influence. On ne sait que beaucoup moins ce que l'on fait lorsque l'on utilise les croisements. Avec les mutations, on sait ce que l'on touche : soit en dimension une pour la mutation uniforme limitée, où on ne touche que très peu, à toutes les dimensions, avec la gaussienne. Les croisements peuvent parfois casser plus de 10 dimensions en un seul coup. Cela complexifie le système.

1.3 Nos méthodes de remplacement

Le remplacement consiste à réintroduire les descendants obtenus après l'application successive des opérateurs de sélections, de croisements et de mutations, dans la population de leurs parents. Le remplacement ne se fait que sur une certaine proportion de ceux-ci. Nous avons codé deux méthodes de remplacement présentées ici.

1.3.1 Le remplacement stationnaire générationnel

Dans ce cas, les enfants remplacent automatiquement les parents, et ce sans tenir compte de leurs performances. La taille de la population ne varie pas tout au long du cycle d'évolution simulé. Nous remplaçons la totalité de la population de la génération d'avant par la nouvelle génération, cette méthode est connue sous le nom de remplacement générationnel. Ceci engendre une population ayant une grande variance et de ce fait favorise la diversité.

1.3.2 Le remplacement élitiste

Dans ce cas, on garde au moins l'individu possédant les meilleures performances d'une génération à la suivante. En général, on peut partir du principe qu'un nouvel individu (enfant) prend place au sein de la population que s'il remplit le critère d'être plus performant que le moins performant des individus de la population précédente. Donc les enfants d'une génération ne remplaceront pas nécessairement leurs parents comme dans le remplacement stationnaire et par la même la taille de la population n'est pas figée au cours du temps. Ce type

de stratégie améliore les performances des algorithmes évolutionnaires dans certains cas, mais présente aussi un désavantage en augmentant le taux de convergence prématuré.

C'est ce que l'on appel un algorithme mu + lambda. On génère une population de lambda individus, mettons 20, au hasard. Si mu = 5, alors on ne garde que les 5 meilleurs, on génère 15 enfants sur la base de ces 5 meilleurs. On teste ces 15 enfants et on sait alors comment ils se placent par rapport aux 5 parents, donc on garde les 5 meilleurs, on re-genère les 15 enfants et ainsi de suite.

1.4 Gestion du bruit

Il est possible qu'il y ait du bruit sur les évaluations de la fitness d'un individu. C'est à dire que l'on peut évaluer un individu avec les mêmes paramètres et il peut être moins bon, ou bien meilleur. Nous avons donc évalué plusieurs fois les individus pour être sûr que la fitness soit exacte. Les robots disposent de 700 itérations pour calculer leur fitness, cela signifie que les 700 itérations seront refaite plusieurs fois dans le but d'avoir un chiffre proche de la fitness réelle, sans chance ou malchance. On doit regarder au bout de quel nombre de fois la valeur moyenne de la fitness se stabilise pour un individu. Nous avons tester cela avec deux jeux de paramètres différents(J1 et J2), pour être plus certaine que la convergence, par exemple au bout de 20 évaluations de la même fitness, ne soit pas liée à un type de paramètre, et pareillement pour la population(P1 = 10 et P2 = 50).

Les 8 graphes suivants nous montrent l'évolution de la fitness moyenne d'un individu avec les mêmes paramètres pour plusieurs évaluations.

Les Figures 1 et 2 correspondent aux paramètres J1, pour une population de 10, de haute en bas, le premier teste 100 évaluations, le deuxième 50 :

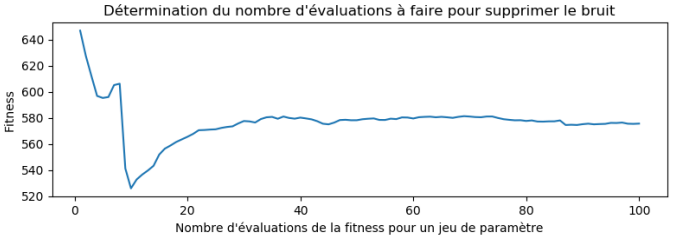


FIGURE 1 – 100 évaluations pour une population de 10 génome J1

On remarque que les 20 premières évaluations ne permettent pas d'établir la véritable fitness d'un individu, il n'y a convergence qu'après la 20ème. Lorsque l'on relance l'individu sur 50 évaluations on remarque que la valeur, pour avoir convergence à moins de 4% d'erreur, est de 25 évaluations.

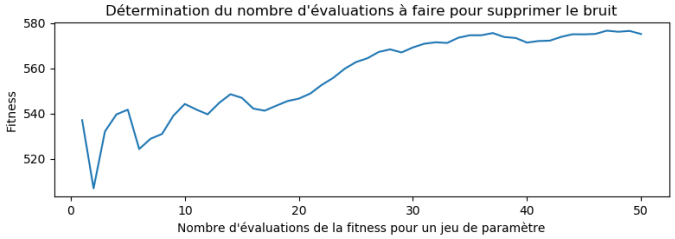


FIGURE 2 – 50 évaluations pour une population de 10 génome J1

Les Figures 3 et 4 correspondent aux paramètres J2, pour une population de 10, de haute en bas, le premier teste 100 évaluations, le deuxième 50 :

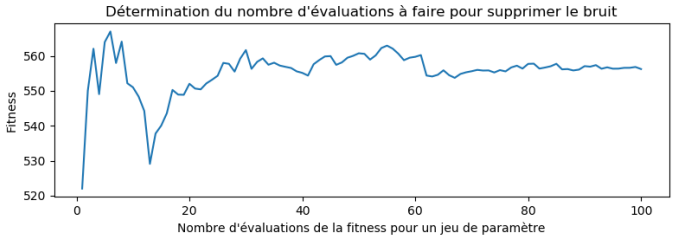


FIGURE 3 – 100 évaluations pour une population de 10 génome J2

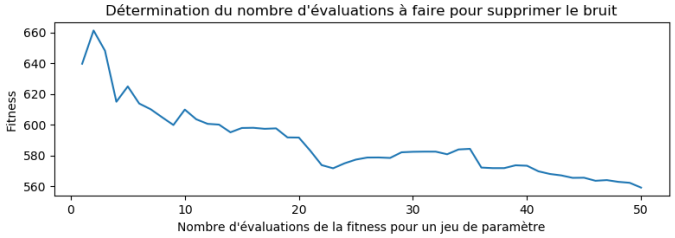


FIGURE 4 – 50 évaluations pour une population de 10 génome J2

Lorsque l'on utilise d'autres paramètres, J2, on voit que le nombre d'évaluations nécessaire est beaucoup plus important, largement supérieur à 20 et plutôt égale à 60. Cependant, si nous considérons 20 évaluations, le taux d'erreur est inférieur à 4%.

Les Figures 5 et 6 correspondent aux paramètres J1, pour une population de 50, de gauche à droite, le premier teste 100 évaluations, le deuxième 50 :

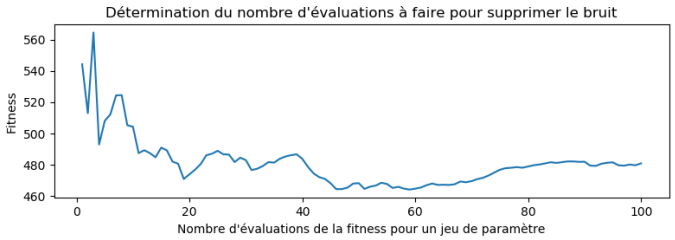


FIGURE 5 – 100 évaluations pour une population de 50 génome J1

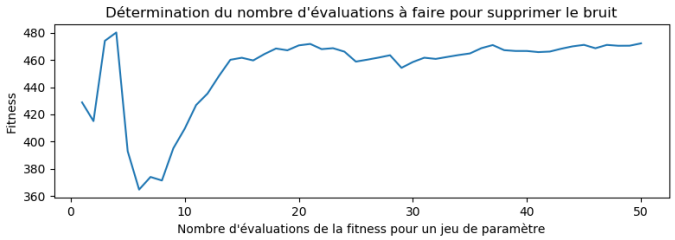


FIGURE 6 – 50 évaluations pour une population de 50 génome J1

Pour une population plus élevée, 50, on remarque que le nombre d'évaluations à 20 permet aussi de limer les incohérences, réduisant l'erreur à moins de 4%, bien que la convergence totale soit plus longue.

Les Figures 7 et 8 correspondent aux paramètres J2, pour une population de 50, de gauche à droite, le premier teste 100 évaluations, le deuxième 50 :

Dans ce cas, comme on peut voir ci dessous sur les Figures 7 et 8, le nombre d'évaluations à 20 ne permet pas de limer les incohérences, et serait plutôt estimé à 40. Nous pouvons en déduire que les interactions avec les robots per-

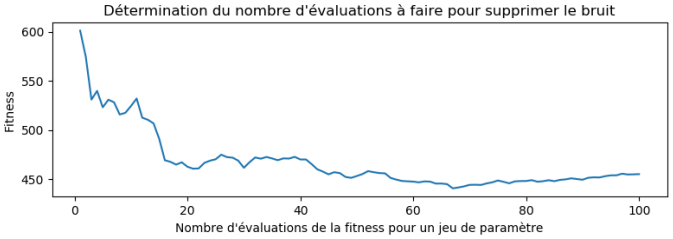


FIGURE 7 – 100 évaluations pour une population de 50 génome J2

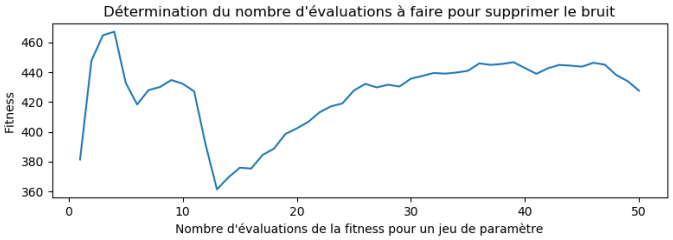


FIGURE 8 – 50 évaluations pour une population de 50 génome J2

turbent grandement les évaluations de la fitness. En effet, pour 50 robots, le nombre d'évaluations nécessaires pour la convergence de la valeur de la fitness est beaucoup plus long, environ 40, tandis que pour 10 robots la convergence est beaucoup plus rapide, environ 20.

Idéalement, il faudra effectuer 700 * 40 évaluations, soit 28 000 évaluations par génération. Compte tenu de la puissance de calcul de l'ordinateur dont nous disposons, nous avons choisis d'effectuer seulement 20 calculs de fitness dans le but de gommer les irrégularités d'avoir peu de mesure. Ainsi nous aurons une idée moyenne de la fitness véritable d'un individu, sans que celui ci bénéficie de chance ou de malchance.

Maintenant que nous avons notre algorithme, ses différents moyens de sélection, de mutation et de remplacement, nous allons le tester sur plusieurs instances, avec l'objectif d'en tirer des paramètres optimaux.

2 Mise en comparaison des systèmes de sélection

Nous avons choisis d'effectuer les tests sur une population, d'abord de 10, puis de 50 individus. Le taux de mutation (sigma) a été fixé à 0,01.

Ce qui est comparé ici est le nombre d'évaluations, il doit être le même. Nous devons montrer, par exemple, ce qu'est capable de produire un algorithme, pour tels paramètres, sur 4000 évaluations. Nous avons choisi 500 évaluations, ce qui

prend un temps de 15 heure, cela correspond à 50 générations d'une population de 10, mais seulement 10 générations pour une population de 50. Aussi, avant chaque évaluation, nous faisons 700 itérations, 20 fois, pour éliminer le bruit.

2.1 Outils de comparaison

Des tests préliminaires ont été établi, dans un premier temps, sur 500 évaluations aussi, dans le but de pré-déterminer les meilleures méthodes de sélections parmi celles décrites dans la partie précédente, les tests présents ici, ayant une durée de plus de 15 heures, sont ceux qui ont été les plus prometteurs. Les tests préliminaires sont disponibles dans le Rapport de Tests, dans le but de ne pas sur-charger ce rapport. Pour l'étude des courbes, ce sont les box plot, visible en Figure 9, qui ont été choisi. Ils nous permettent d'étudier les caractéristiques distributionnelles d'un groupe de scores ainsi que leur niveau.

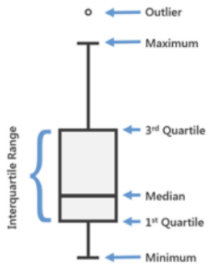


FIGURE 9 – Exemple de box pour un box plot

Les scores sont triés et quatre groupes de taille égale sont fabriqués à partir des partitions, donc 25% de tous les scores sont placés dans chaque groupe. Les lignes qui divisent les groupes sont appelées quartiles.

La médiane (quartile moyen) marque le point médian des données et est indiquée par la ligne qui divise la boîte en deux parties. La moitié des scores sont supérieurs ou égaux à cette valeur et la moitié sont inférieurs.

La "box" du milieu représente 50% des fitness pour le groupe.

Les moustaches supérieures et inférieures représentent des scores en dehors des 50% du milieu. Les moustaches peuvent s'étirer sur un éventail de scores plus large que les groupes du quartile moyen. Des points, dits aberrants, peuvent être présents au dessus ou en dessous des moustaches. Ils nous permettent de visualiser les tendances globales d'un groupe.

Dans cette partie analyse, deux sortes de graphes seront présenter. Dans un premier temps, celui de gauche, présente avec une moyenne sur 11 instances, les

évaluations de toutes les fitness. Son échelle sera de 0 à 800 pour les fitness et de 0 à 500 pour les évaluations. De plus, le graphique de droite nous montre pour chaque exécution, la meilleure génération. L'échelle sera différente, comme les meilleurs sont représentés, la fitness ira de 0 à 1000, les évaluations ne changerons pas.

2.2 Comparaison du meilleur système de sélection pour les deux mutations sur une population de 10 individus

Pour une population de 10 individus, les deux meilleurs moyens de sélection sont les mêmes pour les deux mutations, soit la mutation élitiste.

Figure 10 représentant l'évolution pour un système sélection élitiste et mutation gaussienne :

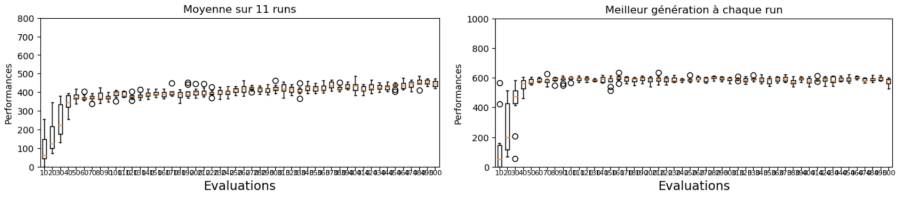


FIGURE 10 – Sélection élitiste, mutation gaussienne

La convergence est très rapidement atteinte pour les Meilleures générations (graphe de droite), 30ème évaluation, tandis qu'elle est un peu plus lente pour la Moyenne de toutes, 60ème. On remarque les caractéristiques d'une mutation gaussienne, les premiers et troisième quartiles sont très proches, petite box, et il n'y a peu ou pas de valeurs éloignées de la normal. Ceci est normal pour la mutation gaussienne qui n'explore que localement, et ne permet donc pas une grande variabilité. En effet la sélection élitiste combinée à une mutation qui ne change que peu les paramètres aboutit à une bonne population globale, aux paramètres quasi-identiques.

Figure 11 représentant l'évolution pour un système sélection élitiste et mutation uniforme limités :

Contrairement à la mutation gaussienne, la mutation uniforme limitée nous retourne des box beaucoup plus grandes, ce qui montre une grande variabilité de la population, avec des génomes bien différents. Le premier quartile reste souvent plus éloigné de la médiane que le troisième, cela signifie que 25% de la population haute possèdent des fitness proches, tandis que les 25% moins bon possèdent des fitness plus éparses. La convergence est beaucoup plus lente, mais elle abouti à une population global de très bonne fitness, légèrement en dessous de la meilleure génération pour la moyenne totale, soit environ 500, tandis que

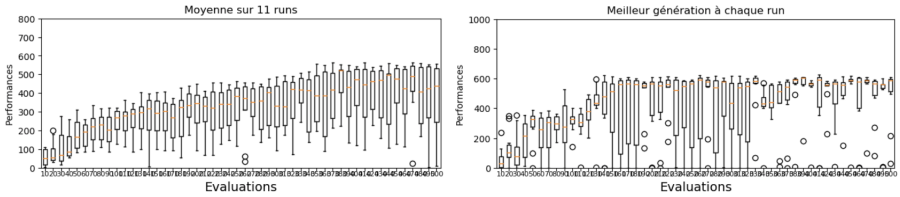


FIGURE 11 – Sélection élitiste, mutation uniforme

la meilleure génération, elle, atteint les 600 de fitness. La différence est habituellement plus marquée.

En comparaison avec les résultats de la mutation gaussienne, on se rend compte que pour une population de 10, l'exploration locale n'est pas la meilleure option. Introduire de la diversité permet d'obtenir des génomes de meilleure qualité. En effet, la fitness moyenne sur les 11 exécutions pour une mutation gaussienne est inférieure à 500.

2.3 Comparaison du meilleur système de sélection pour les deux mutations sur une population de 50 individus

Pour une population de 50 individus, contrairement aux observations faite pour une population 10, les deux meilleurs moyens de sélection ne sont pas les mêmes pour les deux mutations. La mutation gaussienne retrouve une meilleure valeur de fitness pour la sélection élitiste, tandis que la mutation uniforme limitée renvoie ses meilleurs résultats sur un tournoi à 8 places.

Figure 12 représentant l'évolution pour un système sélection élitiste et mutation gaussienne :

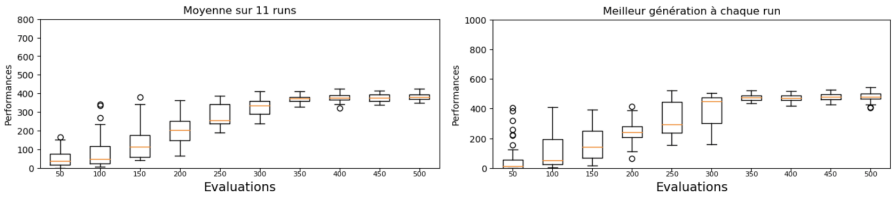


FIGURE 12 – Sélection élitiste et mutation gaussienne

Avec ce jeu de paramètres, on observe les mêmes caractéristiques que pour une population de 10. En effet, les box deviennent de plus en plus plates, pareillement aux moustaches, ce qui permet d'en faire les mêmes conclusions quant à la non diversification des individus. Ici on constate plutôt une perte de diversité des génomes de départ, aléatoire, à partir de la 200ème évaluation. Les générations

suivantes ont bénéficié du meilleur individu évalué à la 100ème évaluation pour calquer son génome et ne plus évoluer.

Cependant, on observe que le nombre d'individus joue sur la valeur de la fitness. Elle est beaucoup moins bonne ici. Si une fitness de 500 était atteinte avec une population de 10, ici, 400 est à peine atteint.

Figure 13 représentant l'évolution pour un système sélection par tournoi, i=8, et mutation uniforme limitée :

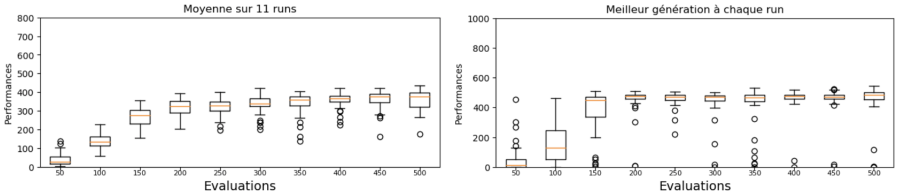


FIGURE 13 – Sélection par tournoi, i=8, et mutation uniforme limitée

Pour le tournoi à 8 individus, on remarque que la vitesse de convergence est très rapide comparée au système retenue pour la mutation Gaussienne, 150ème évaluation contre 350ème. Les box sont aussi très plates, ce qui suggère une cohésion de groupe, bien qu'il y ait encore un bon nombre de point abérants, qu'il n'y a pas pour le système précédent. Cependant les valeurs de fitness médiane sont identiques d'un système à l'autre.

2.4 Conclusion sur les système de sélection

Les conclusions sur les méthodes de section dépendent de ce que l'on recherche. Si nous souhaitons avoir un système qui nous donne le meilleur individu, pour 500 évaluations, alors la meilleure option est d'introduire de la diversité avec la mutation uniforme limitée. Les fortes mutations permettent de créer des solutions originales qui optimisent le parcours des robots.

En revanche, si l'on recherche une cohésion de groupe, c'est à dire des individus avec des performances similaires, alors la mutation gaussienne reste la plus adaptée, pour l'instant, car, avec son exploration locale, elle créé des individus à une distance mutationnelle très proche, et donc souvent à la fitness très proche. La part de hasard introduite au début suffit souvent à faire émerger un bon individu, qui est ensuite dupliqué et légèrement amélioré à chaque pas. Convergence rapide mais stagnation des valeures de fitness. Avec la mutation uniforme limitée, la convergence est moins rapide mais permet de dépasser les valeurs de la mutation gaussienne, par sa diversité.

La notion de voisinage de la mutation gaussienne n'est peut être pas une si bonne option, est ce que cela est réellement moins bon, pour la fitness global du groupe, quand on fait de l'uniforme ? Peut être que le fait de faire varier le paramètre de mutation dans la population permettra une amélioration. La prochaine partie est consacrée à l'exploration de cette méthode de mutation qui semble être intéressante.

Troisième partie

Une évaluation plus poussée de la mutation uniforme limitée

Au delà d'augmenter le nombre de mutation dans la population, nous avons changé toute la structure de notre algorithme pour une stratégie d'évolution ((mu + lambda)-ES) (I.Rechenberg et H.P. Schwefel 1965 Berlin). Il n'y a pas de sélection au sens darwinien : à partir d'une population de taille mu, lambda enfants sont générés par application des opérateurs de variation. L'étape de remplacement est alors totalement déterministe, ce sont les mu meilleurs des mu + lambda parents plus enfants qui survivent.

1 Changements de l'algorithme

Nous avons choisis de tester cette mutation avec du ((1+N)-ES) avec N appartenant à 9, 19, 29, 39, 49, 59. Basé sur le (1+1)-ES classique. Un individu est évalué, possède une valeur de fitness, ensuite un descendant de cet individu est créé, c'est une version mutée de cet individu de référence. La mutation peut être celle que l'on souhaite, soit une mutation gaussienne sur l'ensemble des paramètres qui définissent le parent, pour estimer l'enfant, soit au hasard sur un gène parmi tous les gênes de l'enfant (donc du parent car recopie). Il y a ensuite une évaluation de la fitness du descendant, et là il y a deux cas :

- Soit l'enfant est moins bon que son parent, il est éliminé et on recommence.
- Soit l'enfant est meilleur que le parent. L'enfant devient le parent.

On a remplacé l'ancien jeu de paramètre par ce nouveau jeu de paramètre qui n'était pas très éloigné, qui était à une distance mutationnelle assez réduite du précédent. Lui maintenant joue le rôle de parent à partir duquel on va essayer de trouver un enfant qui sera meilleur.

Le seul changement opéré au (1+1)-ES est que nous allons créer N descendants (avec N appartenant à 9, 19, 29, 39, 49, 59) à partir du meilleur individu. Il est intéressant d'avoir un point et de générer plusieurs mutants autour. Il y a une raison pour laquelle cela pourrait fonctionner. En fait le gaussien est très conservateur, il fait des hypothèses très fortes sur ce qu'est la localité, et

parfois il vaut mieux s'éloigner complètement pour s'améliorer. L'espace est multi-dimensionnel et il y a sûrement énormément de chemins qui mènent vers de meilleures solutions, nous allons bénéficier de cela. La mutation uniforme prends son sens. Une mutation gaussienne qui exploite une localité n'a finalement que peu de sens car localité c'est toujours connu et muter plus loin serait plus bénéfique.

Nous nous sommes heurtés à un problème, si le descendant, le muté, est toujours le plus mauvais, alors nous n'avons aucune évolution. Dans ce cas nous gardons toujours le parent, cependant il se trouve que dans certains schémas, la mutation prend en compte le fait que l'on ait réussit à s'améliorer ou non. On peut trouver des méthodes comme augmenter le taux de mutation au fur et à mesure que l'on n'arrive pas à remplacer le parent par un nouvel individu. Dans notre cas, comme nous testons la mutation, le taux est déjà à 1. Dans un autre schéma, l'algorithme (1+1)-ES permet de modifier le sigma, taux de mutation, au fur et à mesure. En troisième année de licence nous avions vu le (1+1)-ES avec la règle des 1/5ème. Dans ce cas, lorsque l'on n'arrive pas à remplacer le parent, on va appliquer des mutations de moins en moins forte, pour essayer d'explorer un peu plus localement autour du parent.

2 Nouveaux tests sur l'algorithme

Dans le but de pouvoir comparer et vérifier nos résultats, nous avons extrait le meilleur jeu de paramètre de la partie 1, et implémenter un (1+1)-ES règle des 1/5ème.

2.1 Mesures étalon

2.1.1 L'algorithme (1+1)-ES

L'algorithme (1+1)-ES nous retourne de très bons résultats, la fitness du meilleur individu est de 600,05.

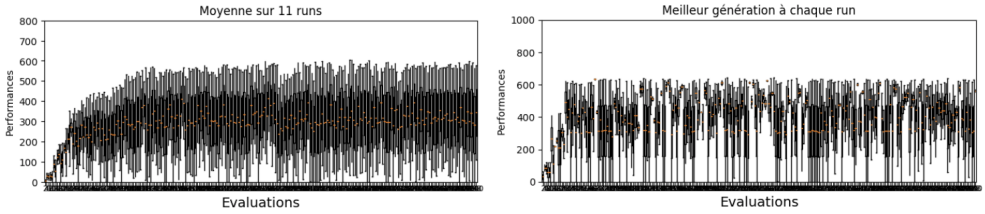


FIGURE 14 – Algorithme (1+1)-ES

2.1.2 Meilleur jeu de paramètres de l'algorithme précédent

Nous disposons, dans cette section, le graphe nous donnant les meilleurs résultats, évalués dans la partie précédente. La meilleure fitness d'un individu étant à 557,25.

Graphes représentant l'évolution pour un système sélection élitiste et mutation uniforme limitée, pour une population de 10 individus :

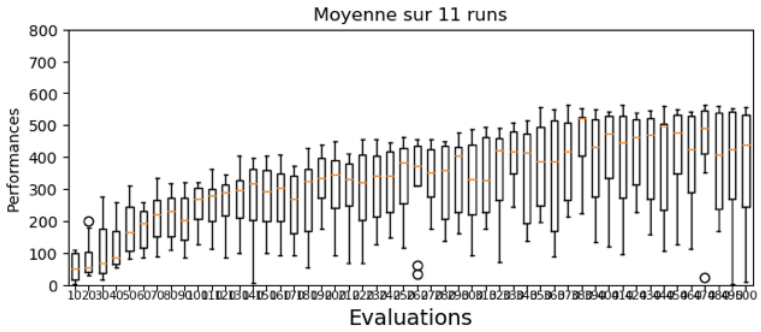


FIGURE 15 – Sélection élitiste et mutation uniforme limités, pour une population de 10 individus

2.2 Résultats des tests, comparaisons et analyses

Dans cette partie analyse, les deux mêmes sortes de graphes seront présentés. Dans un premier temps, celui de gauche, présente avec une moyenne sur 11 instances, les évaluations de toutes les fitness. Son échelle sera de 0 à 800 pour les fitness et de 0 à 500 pour les évaluations. De plus, le graphique de droite nous montre pour chaque exécution, la meilleure génération. L'échelle sera différente, comme les meilleurs sont représentées, la fitness ira de 0 à 1000, les évaluations ne changeront pas.

Figure 16 représentant l'évolution pour un système (1+9)-ES :

Sur la Moyenne des 11 runs, on constate que les box sont relativement longues, avec les premiers et troisièmes quartiles éloignés. Cependant, la médiane reste souvent au milieu de la box, ce qui montre une dispersion égale des données autour de la médiane pour 50% des individus. Les valeurs minimum et maximum sont elles aussi très éloignées, avec de nombreux points au dessus ou en dessous de la normale. Sur la Meilleure génération à chaque run, on constate qu'il y a

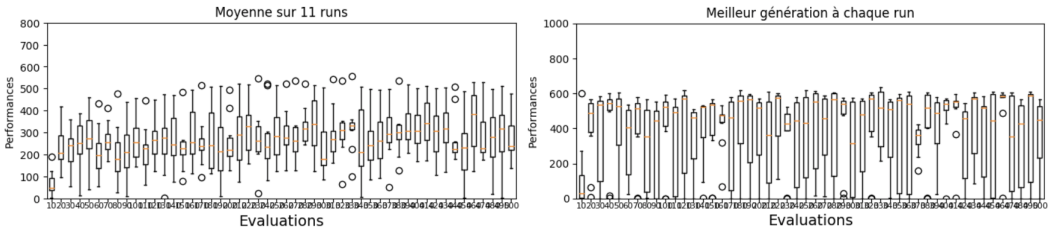


FIGURE 16 – Sélection élitiste et mutation uniforme limités, pour une population de 10 individus

beaucoup de très bons individus, maximum, troisième quartile et médiane étant très proches. Aussi, la moitié de la population de mauvaise fitness possède des fitness très basses, en effet les moustaches basses des box descendent jusqu'à 0 de fitness. Les 50% des bons individus sont très proches, tandis que les moins bons sont très dispersés. La convergence de la population en une bonne fitness n'est pas remarquable, il n'y a pas de stabilité, bien que l'on remarque une large progression entre l'évaluation 0 et la 30ème.

Figure 17 représentant l'évolution pour un système (1+19)-ES :

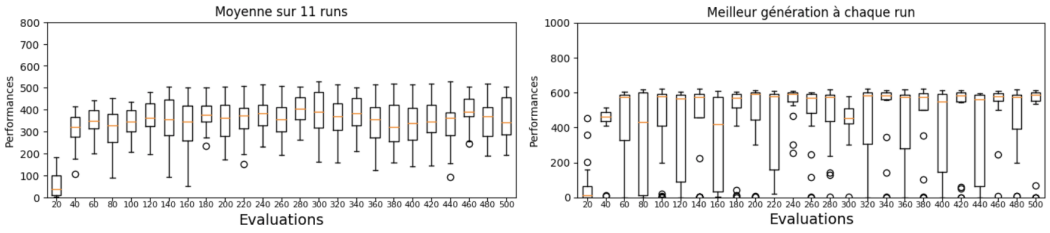


FIGURE 17 – Algorithme (1+19)-ES

Avec 19 descendants on observe que les box sont beaucoup plus courtes, c'est à dire que les individus ont des fitness beaucoup plus proches. La médiane des fitness moyenne est au moins de 100 supérieure à celle de l'algorithme (1+9)-ES. Suivant cette tendance, les Meilleures générations à chaque run sont aussi plus proches même s'il existe encore de nombreuses disparités en comparaison avec la moyenne globale. La convergence est plus perceptible, à partir de la 60ème évaluation, pour les deux graphes. Nous pouvons dès lors conclure que l'algorithme (1+19)-ES est plus efficace car donne de meilleur résultat en moyenne, et en individuel. Continuons d'augmenter le nombre de descendant mutant pour tenter de faire émerger, ou non, un lien entre nombre de mutant et solution de qualité.

Figure 18 représentant l'évolution pour un système (1+29)-ES :

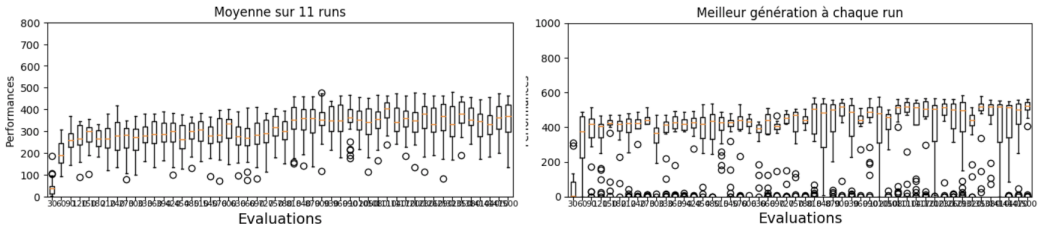


FIGURE 18 – Algorithme (1+29)-ES

Les box sont ici beaucoup plus régulières, la répartition des valeurs est équivalente de part et d'autre de la médiane. Il en est de même pour le graphe des Meilleurs générations. Malgré la présence de nombreux points en aberrants ayant une fitness nulle, les box restent peu étendues. Plus le nombre de descendant augmente, plus il semblerait que les individus soient proches, néanmoins les fitness moyennes, sur toutes les instances ou seulement les meilleures, ne semble pas s'être améliorées.

Figure 19 représentant l'évolution pour un système (1+39)-ES :

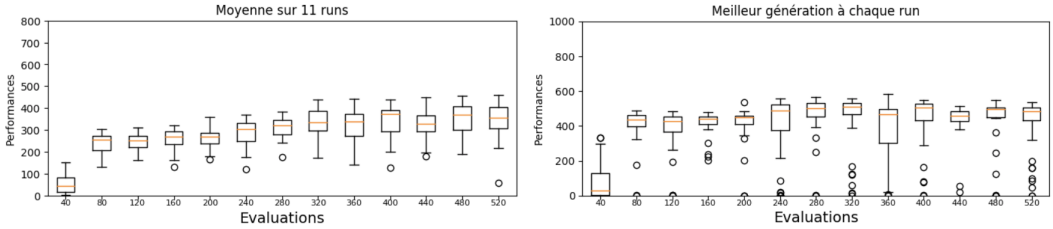


FIGURE 19 – Algorithme (1+39)-ES

La tendance se confirme ici. Les box sont encore plus minces et les moustaches moins étendues. Les individus sont proches, et les fitness, pas significativement meilleures.

Figure 20 représentant l'évolution pour un système (1+49)-ES :

Pour 50 individus, nous ne confirmons les résultats précédent, la fitness apparait ici comme meilleure que les évaluations (1+29)-ES et (1+30)-ES.

Souhaitant pousser l'expérience, nous avons choisi de pousser le nombre d'individus à (1+59)-ES, bien que l'on ne possède pas d'équivalent comparable dans la première partie.

Figure 21 représentant l'évolution pour un système (1+59)-ES :

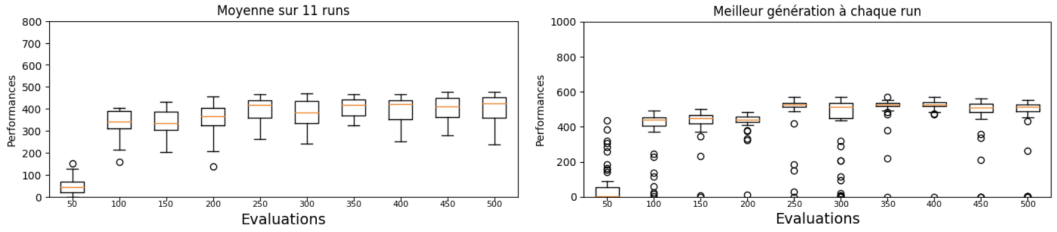


FIGURE 20 – Algorithme (1+49)-ES

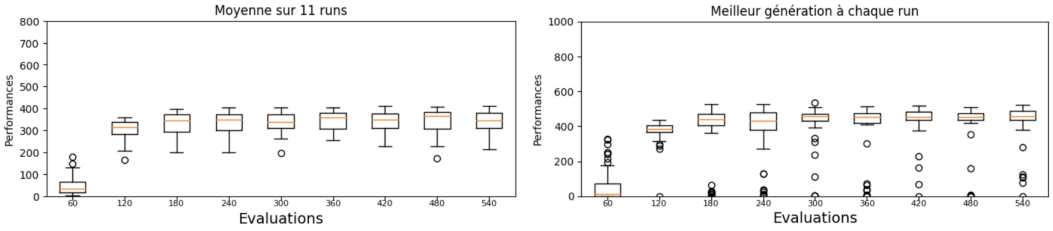


FIGURE 21 – Algorithme (1+59)-ES

Il semble y avoir un lien entre le nombre de descendants et la qualité de la fitness produite. En effet, moins il y a de descendants plus les fitness sont hautes. Lorsque que le lambda est faible, on évite d'utiliser le nombre d'évaluations que l'on possède pour explorer un parent à la fitness moyenne, cela nous permet donc de nous améliorer plus rapidement. Le fait de s'améliorer plus rapidement permet aussi une convergence plus lente, et donc plus haute en bénéficiant du nombre d'évaluations restant comparer à des lambda plus élevé.

Si on souhaite une bonne fitness de groupe, comme la configuration Sélection Elitiste, Mutation Gaussienne, population 10, la mutation uniforme limitée sous cette forme est une très bonne option car les fitness médianes sont tout de même hautes et proches pour des lambda élevés, (1+49)-ES spécialement.

Dans le but de comparer le meilleur individu produit, la dynamique de groupe produite, et le point de convergence, par cette méthode à ceux produits par les méthodes développées précédemment nous avons regroupé les résultats dans un tableau.

Si le (1+19)-ES s'avère nous retourner le meilleur individu de tous, en 500 évaluations, si l'objectif est d'avoir une bonne population globale, alors il faudra plutôt choisir la combinaison sélection élitiste, mutation uniforme limitée, population de 10, bien que la convergence soit tardive. De plus la perte sur le meilleur individu est si légère que l'on peut dire que c'est la meilleure option dont nous disposons dans ce rapport, pour notre problème de parcours.

Ainsi l'implémentation choisis pour les robots Thymio est une sélection

	(1+1)-ES pop 2	Sélection Elitiste Mutation Gaussienne pop 10	Sélection Elitiste Mutation Uniforme Limitée pop 10	Sélection Par Tournoi Mutation Uniforme Limitée pop 50	(1+19)-ES pop 20
Fitness du meilleur individu	600	604	607	554	608
Fitness médiane pour toutes les générations	300	450	470	450	400
point de convergence en nombre d'évaluation	60	60	350	350	30

FIGURE 22 – Comparaison des différents modes

élitiste combiné à une mutation uniforme limitée.

Quatrième partie
La robotique réelle

Dans un premier temps il aurait été intéressant de modifier le système de sélection/mutation codé sur les Thymio, dans le but de comparer les résultats que nous aurions eu à ceux de l'année dernière, c'est à dire une sélection par tournoi avec une mutation gaussienne.

1 Les robots et leur plateforme

Nous disposons de 30 robots Thymio, petite base roulante de 110x110x50 mm à poids réduit, possédant de nombreux capteurs (microphone, récepteur infrarouge, température, proximité, accéléromètre 3 axes, capteurs au sol pour le suivi de lignes), actionneurs (moteurs, haut-parleurs, LEDs) et connecteurs (USB, carte mémoire). Dans le cadre d'une collaboration avec un étudiant en M1 IMA, les robots ont été équipés d'une caméra. Aseba Studio permet de coder simplement ces robots via une syntaxe inspirée de Matlab, mais largement simplifié. Cependant nous utiliseront une Raspberry PI 3, alimentée par une batterie portable, pour leur implémenter un comportement. Cette dernière sera branchée sur le port USB, ce que l'on peut voir sur la Figure 2. Pour coder les comportements sur la Raspberry, nous utiliserons OctoPY, développée par Arthur Bernard, ancien doctorat à l'ISIR. Elle nous permet de communiquer en python avec les robots. Le développement des robots sera évalué dans un arène de 2,5m sur 2,5m mises à disposition à l'ISIR, Institut des Systèmes Intelligents et de Robotique.

2 Les problèmes rencontrés

La documentation utilisée pour la démonstration d'OctoPy est celle utilisée par le groupe du projet robotique Thymios de l'année 2017. Nous avons eu plusieurs problèmes avec cette plateforme : Une mise à jour d'Ubuntu nous a demandé des importations pour pouvoir utiliser à nouveau les fonctionnalités d'OctoPy. Cependant, certaines ont été changées, par exemple le LOOK. Cette commande est censée nous donner la liste sous forme d'host-name des robots connectés au routeur. Cependant, il nous était impossible d'en voir un, la liste ne s'affichait pas et il nous était donc aussi impossible de rentrer en communication avec les robots. Il a fallu, avec l'aide de Parham Sham (qui a eu le projet Thymios de l'année 2017 avec Tanguy Soto), re-coder une partie du script de la fonction LOOK, mais en retour nous avons réussit à n'afficher que les adresse IP des robots. Ceci reste une solution provisoire car beaucoup moins pratique pour identifier et appeler les robots dans les codes. Dans les anciens codes qu'il était demandé d'implémenter, les robots était appelés par leur host-name et les commandes sont faites pour recevoir des host-name. Il fallait donc faire un changement pour que les adresse IP soit recevables.

Aussi, dans le fichier net_conf.sh, qui permet que les raspberries reçoivent une adresse IP par le DHCP, la fonction start ne fonctionnait plus (le texte était grisé sans raison apparente). Comme nous n'avons pas assez de connaissance dans ce qui touche au réseau, nous avons demandé à Parham Shams de nous aider. Après qu'il ait recopié la fonction start en la nommant restart, c'est à dire que c'est la même fonctionnalité dans les deux cas, donc au lieu de faire :

sudo /etc/init.d/isc-dhcp-server start
on fait :
sudo /etc/init.d/isc-dhcp-server restart
et cela fonctionne.

L'envoie de code aux robots ne fonctionnaient pas. La commande propre à cela était SET, cependant, seul le Braitenberg se lançait. Nous n'avons pas réussit à comprendre pourquoi. Et comme il nous était impossible d'envoyer le code aux Thymios, nous n'avons donc pas pu tester nos algorithmes en conditions réeelles.

Cinquième partie

Conclusion

Durant notre projet nous avons appris que l'introduction de diversité était très efficace pour trouver une solution un problème donné. Les systèmes (1+N)-ES avec N descendants pour N appartenant à {9, 19, 29, 39, 49, 59}, nous ont permis de trouver la meilleure fitness pour optimiser le déplacement souhaité. Cependant, la configuration Sélection élitiste, mutation uniforme limitée, population 10 permet un meilleur rapport meilleur individu/capacité du groupe.
Nous avons donc choisis de pousser l'expérience avec 1500 évaluations de cette dernière configuration pouvoir être certaine du point de convergence, et observer s'il était possible d'obtenir des fitness encore meilleures ou si le point de convergence avait été atteint :

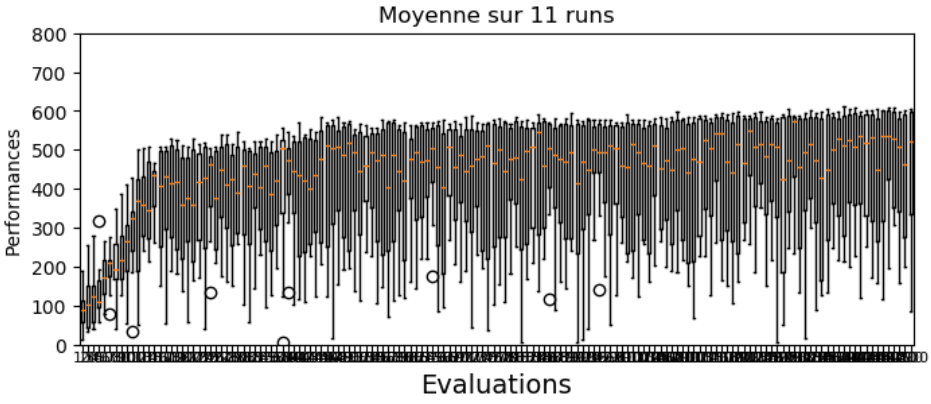


FIGURE 23 – Sélection élitiste, mutation uniforme limitée, population 10

En augmentant le nombre d'évaluations, on permet encore à cette configuration de progresser. La fitness médiane du groupe dépasse les 500 et certains individus dépassent les 600.

Notre projet nous a aussi permis de nous confronter aux problèmes de la robotique réelles, tels que les problèmes de réseaux, de communications et des multiples compétences que nous devons maitriser pour que tout puisse fonctionner. Aussi, nous nous sommes rendu compte que le test des algorithmes sur simulateur est d'une grande facilité comparé à la robotique réelle. Il y a des exigences de temps qui n'auraient pas pu être rempli, comme le fait de laisser s'exécuter un algorithme plus de 15 heures, or la génétique demande beaucoup de temps.

Une évolution du projet, maintenant que nous possédons le meilleur système de sélection et de mutation pour notre problème, aurait pu être de transformer

cet algorithme génétique en un algorithme distribué, c'est à dire que chaque robot communiquera son évolution aux robots qu'il croise.

Nous pourrions ainsi observer l'évolution du passage d'informations dans un groupe de robot pour qu'ils puissent s'améliorer.
C'est un axe que nous souhaitons développer.