

ROMERO: UN PÈLERINAGE ROBOTIQUE À SANTA FE

Christof Teuscher, Eduardo Sanchez, Moshe Sipper

Laboratoire de Systèmes Logiques, Ecole Polytechnique Fédérale de Lausanne

CH-1015 Lausanne, Suisse

Web: lslwww.epfl.ch

E-mail: {prénom.nom}@di.epfl.ch

Résumé - *Beaucoup de chercheurs en robotique évolutive aimeraient avoir à disposition une population de vrais robots. Afin, de rendre réalité ce rêve, notre travail poursuit deux buts: (1) construire un robot bon marché, modulaire et extensible et (2) utiliser une population de ces robots pour faire des expériences dans le domaine de la robotique évolutive. Ce papier présente les premiers pas dans cette direction: nous décrivons la conception et l'implémentation d'un nouveau contrôleur utilisé dans un robot à coût bas, et son utilisation dans une version réelle de la piste de Santa Fe*

Mots clé - robots mobiles, programmation génétique, robotique évolutive.

1 Introduction

Beaucoup de chercheurs en robotique évolutive aimeraient avoir à disposition une population de vrais robots. Hélas, ce rêve se transforme trop souvent en un cauchemar sitôt que le prix est mis en considération. Comme dit Floreano [3], si l'on veut que des robots fassent partie de notre vie quotidienne, il faut les faire moins chers, plus modulaires et plus facilement adaptables à différentes utilisations.

Jusqu'à maintenant, des expériences avec une dizaine de robots étaient faites surtout, voire uniquement, en simulation. Le but ultime de notre travail est de disposer d'une grande population de vrais robots, afin de les utiliser dans des expériences du monde réel.

L'utilisation d'une telle population de robots diminue considérablement le temps d'évolution par rapport à la méthode classique, où un seul robot est utilisé séquentiellement pour évaluer tous les individus d'une génération. De plus, l'interaction entre les robots devient possible, ce qui ouvre des domaines plein de promesses, tels que la co-évolution, la coopération et la compétition entre les individus de la population.

Ce papier présente les premiers pas dans cette direction : on a construit un robot très bon marché et on l'a utilisé pour résoudre une version réelle de la piste de Santa Fe. Dans le sens du proverbe "in the two's company, three's a crowd", nous ne disposons pour l'instant que de deux exemplaires, mais nous nous attendons à une foule pour un futur proche.

Dans la première section, le matériel du robot est présenté, tandis que le problème de la piste de Santa Fe fait l'objet de la deuxième partie. La section 4 décrit l'expérimentation et présente les premiers résultats ; fi-

nalement, la section 5 présente les conclusions et une vision du travail futur.

2 Un nouveau matériel pour des robots évolutifs

2.1 LEGO MindStorms

L'introduction, l'année passée, de LEGO MindStorms ¹ a été le point de départ de ce travail. Les composantes de ce nouveau produit permettent de construire un grand nombre de robots en quelques minutes, de les modifier et de les adapter à un environnement. Il était cité récemment, dans *IEEE Spectrum* [9], comme l'un des produits d'électronique de divertissement les plus intéressants.

Plusieurs groupes de recherche, tels que LegoLab en Danemark [8] ou l'Université d'Edinburg [14] travaillent déjà avec des pièces LEGO.

L'innovation principale introduite par LEGO MindStorms est le contrôleur RCX (Robot Command System), le coeur (cerveau?) du système, chargé de contrôler les moteurs et de lire les capteurs. Toutefois, le RCX nous a paru trop limité pour faire de la recherche dans le domaine de la vie artificielle et de la robotique, raison pour laquelle nous avons décidé de construire notre propre contrôleur, tout en gardant les éléments d'interface de LEGO (capteurs et moteurs).

¹LEGO, the LEGO logo, the LEGO brick, and LEGO MindStorms are all trademarks of the LEGO Group.

2.2 EvoMaster : Un nouveau contrôleur de robot

Notre contrôleur – l’*EvoMaster* – possède les avantages suivants par rapport au RCX : (1) une connexion à un nombre plus élevé de capteurs et de moteurs, (2) une meilleure communication entre les robots, (3) une programmation en C et (4) une architecture plus ouverte pour des extensions futures.

Le coeur de l’*EvoMaster* est un microprocesseur HC11F1, avec 32kB de mémoire EPROM externe et 32kB de mémoire vive pour les données. Il est possible de connecter jusqu’à sept capteurs et quatre moteurs au contrôleur.

Pour faciliter de futures extensions, une mémoire porte SRAM à double port est accessible depuis le processeur et depuis le bus d’extension. Cette mémoire contient des registres permettant le contrôle de toutes les ressources du robot. Il est possible de brancher deux cartes d’extension sur la carte mère. La figure 1 montre l’architecture de *EvoMaster*.

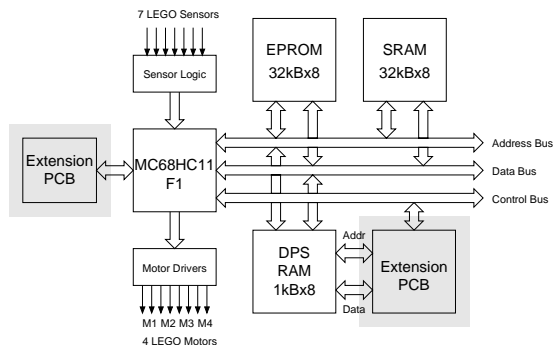


FIG. 1: L’architecture du contrôleur *EvoMaster*.

Nous avons également développé un émetteur-récepteur d’ultrason, pour l’échange de messages entre deux robots, ainsi qu’un module d’aide au déverminage, comprenant une interface RS-232 et une unité d’affichage. Le module d’ultrason peut être également utilisé pour mesurer des distances.

2.3 Romero, le robot pèlerin

Sur la base du contrôleur *EvoMaster*, nous avons construit un robot nommé Romero (pèlerin en espagnol), satisfaisant les contraintes exposées dans la section 1 : il est bon marché (coût inférieur à 150 dollars), il utilise des pièces LEGO et il s’adapte facilement à de nouvelles tâches et de nouveaux environnements. Romero est équipé de deux roues, deux moteurs et deux capteurs de lumière orientés vers le sol. Il contient une pile rechargeable (1400mA/h) lui donnant une autonomie d’à peu près cinq heures.

La figure 2 montre les jumeaux Romero, chacun équipé avec un module ultrason. En comparaison avec le robot bien connu Khepera [10], Romero n’offre pas la même qualité, mais est nettement moins cher et donne la possibilité de construire une grande population, à un coût raisonnable.

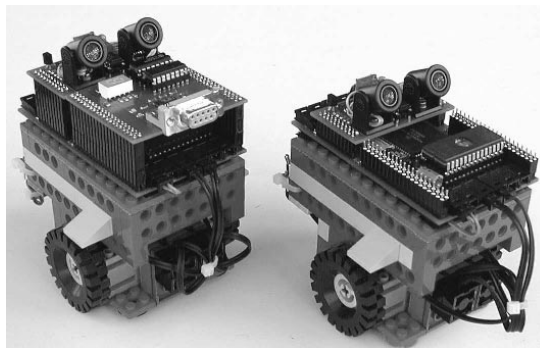


FIG. 2: Les jumeaux Romero.

3 Le pèlerinage de Romero à Santa Fe

Pour tester Romero dans un environnement réel, on a choisi de faire évoluer une solution pour le problème bien connu de la piste de Santa Fe. Cette section décrit le problème, tandis l’expérience est présentée à la section suivante.

3.1 La piste de Santa Fe

Une fourmi artificielle est placée sur la piste de Santa Fe [7] – un environnement de 32×32 cellules. Son but est de manger le maximum de pièces de nourriture placées tout au long d’une piste, dans un temps donné. La fourmi est presque aveugle : elle ne peut voir que la cellule qui lui fait face. Le déplacement commence dans la cellule en haut à droite de l’environnement, en direction sud. La difficulté réside dans le fait que les pièces de nourriture ne sont pas toutes contiguës : la piste présente donc des trous. Au total, il y a 89 pièces de nourriture. La difficulté de suivre la piste augmente vers le fin – les trous dans la piste deviennent plus grands et plus nombreux.

Plusieurs chercheurs ont utilisé ce problème comme expérience d’évaluation pour différents algorithmes évolutionnistes. Koza [7] a fait évoluer des machines d’états contrôlant les fourmis : à la fin une fourmi était capable de manger toutes les 89 pièces. Collins *et al.* [2] et Jefferson *et al.* [6] ont fait des expériences similaires avec des machines d’états et des réseaux neuronaux. Tous ces travaux étaient réalisés en simulation

et, à notre connaissance, personne n'a jamais étudié le problème de la piste de Santa Fe dans un environnement réel.

La piste de Santa Fe du LSL (figure 4) est une version réelle de la piste originale utilisée en simulation, destinée à être parcourue par de vrais robots. Nous avons ajouté une piste spéciale autour de l'environnement, guidant le retour du robot au point de départ. De plus, l'environnement de 32×32 n'est plus sous forme d'un toroïde. Par contre, les proportions de la trace et des trous sont bien respectés.

La piste de Santa Fe du LSL est imprimée sur un papier A0 (90cm x 120cm). La figure 3 montre les deux pistes pour nos deux robots.

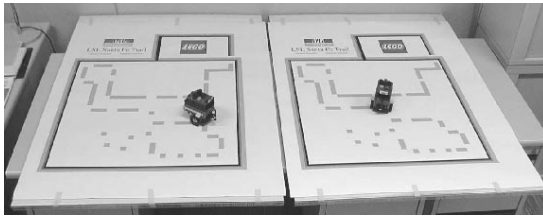


FIG. 3: L'environnement pour nos expériences.

3.2 Réalité contre simulation

Il existe un grand nombre de différences entre notre environnement réel et des environnement de simulation des expériences précédentes. Les fourmis dans la nature sont capables de trouver le chemin le plus court entre une source de nourriture et le nid sans disposer d'outils de vision sophistiqués [4]. Dans cette philosophie, Romero dispose uniquement de ses deux capteurs de lumière comme système de navigation ou de détection. De plus, les cases de notre environnement ne sont pas délimitées et, en conséquence, Romero ne peut pas avancer par des pas d'une case. L'environnement est donc en quelque sorte plus proche de la situation à laquelle est confrontée une fourmi réelle.

Dans la simulation de Koza [7], les pièces de nourriture étaient enlevés de l'environnement au fur et à mesure que la fourmi les mangeaient. Dans notre environnement, ceci n'est plus possible car les pièces sont imprimées sur le papier. La piste est donc plus proche d'une piste de phéromones que d'une piste "mangeable". Cette situation peut générer des solutions bien notées (avec un grand nombre de pièces ramassées), où le robot tourne tout simplement autour d'un axe, en passant plusieurs fois par le même bout de piste. Nous verrons plus tard que la probabilité d'une telle solution a été baissée par le paramétrage de notre expérience.

La piste de Santa Fe simulée est nettement plus simple à résoudre que le problème réel. L'environne-

ment simulé se comporte de manière complètement déterministe – c'est-à-dire, sans bruit. Par contre, la nature est bruitée, non prédictible et non déterministe. De nombreux papiers [5, 11, 13] proposent des solutions pour rendre un environnement réel plus déterministe, à l'aide des systèmes de navigation et de localisation. Mais la mise en place de ces outils n'est pas souvent évidente, ils sont imprécis et demandent un surplus de matériel.

Notre approche est plus simple : Romero ne connaît pas sa position à l'intérieur de l'environnement. La source la plus importante de bruit vient des différences de vitesse des moteurs, produites par les variations de l'état de chargement des piles. Comme une conséquence de ces variations, deux exécutions du même programme ne produisent jamais exactement le même résultat. Ceci rend le problème nettement plus difficile que celui de la piste simulée.

4 L'expérience évolutive

4.1 La définition du génome

Koza [7] a défini quatre mots de base – LEFT, RIGHT, MOVE, NOP – qui représentent les mouvements élémentaires. Une possibilité serait de coder une séquence de ces mouvements dans un génome linéaire. Toutefois, ce génome serait sûrement très long – trop long en tout cas pour faire évoluer une solution dans un temps raisonnable avec des robots réels. Ces mouvements sont donc d'un niveau trop bas pour notre problème.

Notre approche consiste à utiliser les mêmes mouvements élémentaires, mais groupés afin de réduire l'espace des solutions. Par contre, on ne peut pas garantir que les mouvements de base sont exécutés avec 100% de précision. En plus, Romero possède deux comportements innés : (1) il traverse tout seul un bloc contigu de cellules et (2) il traverse tout seul un trou simple dans un coin.

Il reste donc à évoluer les comportements pour traverser les obstacles restants : un trou simple, un double trou, un triple trou, un double trou dans un coin, un triple trou dans un coin. Une analyse de la piste de Santa Fe a montré qu'il n'existe en fait que six comportements possibles :

- Forward : Avancer (l'équivalent d') une cellule dans la direction courante.
- Forward--Forward : Avancer (l'équivalent de) deux cellules dans la direction courante.
- Forward--Left : Avancer (l'équivalent d') une cellule dans la direction courante et tourner à gauche.

- Forward--Right : Avancer (l'équivalent d') une cellule dans la direction courante et tourner à droite.
- Forward--Forward--Left : Avancer (l'équivalent de) deux cellules dans la direction courante et tourner à gauche.
- Forward--Forward--Right : Avancer (l'équivalent de) deux cellules dans la direction courante et tourner à droite.

Basé sur ces six comportements, nous avons défini des "codons" génétiques (Tableau 1). La commande Nop a été rajoutée pour avoir des codons de même longueur ; dans notre cas, tous les codons sont donc de longueur trois.

| Codon | Description |
|-------|---------------------------|
| NNN | Nop – Nop – Nop |
| NNF | Nop – Nop – Forward |
| NFF | Nop – Forward – Forward |
| NFL | Nop – Forward – Left |
| NFR | Nop – Forward – Right |
| FFL | Forward – Forward – Left |
| FFR | Forward – Forward – Right |

TAB. 1: Le sept "codons" génétiques de Romero.

A partir de ces codons, nous avons défini le génome, qui est en fait une séquence de codons : --NNF--NFL--NFR--FFL--NNN--FFR--. Les sept codons du tableau 1 forment donc bien les blocs de base de l'évolution. Il est à noter que le groupement de trois codons a considérablement réduit l'espace des solutions. Le codon NNN ne fait rien du tout et sert uniquement au remplissage, afin de garantir une même longueur pour tous les génomes. Le génome est traité comme un programme exécuté de façon linéaire. Comme il y a 24 trous (obstacles) dans la piste, une séquence de 24 codons devrait en théorie être suffisante pour résoudre le problème. Dans le monde réel, par contre, une séquence plus longue peut être nécessaire. Pour ne pas augmenter l'espace des solutions (7^{24}), on a décidé de ne pas rallonger le génome et de fixer la longueur à 24 codons.

4.2 Les opérations génétiques (sélection, croisement, mutation).

Comme mentionné dans la section 1 (voir aussi figure 3), nous ne disposons pour l'instant que de deux robots, tout en espérant en avoir bientôt une cinquantaine. Dans la situation à venir, les 50 pistes seront alignées, avec un robot par piste (une extension de la situation de la figure 3). Actuellement, cet alignement

est fait virtuellement : les 50 robots sont testés sur les deux robots réels. L'alignement permet d'appliquer la sélection par tournoi local [1, 12]. A la base, chaque robot voit uniquement les génomes de ses deux voisins, à la portée des émetteurs-récepteurs. Dans le tournoi virtuel, chaque robot communique donc uniquement avec ses voisins de gauche et de droite. On pourrait bien entendu choisir d'autres topologies.

Les tournois locaux ont lieu de façon asynchrone (proche de l'algorithme steady-state) : chaque robot exécute son génome, évalue la fitness et transmet le génome aux voisins, qui, à leur tour, réalisent la même opération. Ensuite, on applique l'opérateur de croisement entre le meilleur génome reçu du voisinage et son propre génome. Le position de croisement est sélectionnée aléatoirement sur le génome, et les parties des génomes sont échangées. La seule contrainte est que le croisement peut avoir lieu uniquement entre deux codons—en d'autres mots, les codons restent toujours inchangés. On n'a pas défini explicitement une opération de mutation, mais comme les génomes sont transmis par ultrason, il y a de temps en temps des erreurs de transmission qui correspondent à des mutations bien plus réalistes.

4.3 La fitness.

Dans le monde simulé, la fitness est définie comme le nombre de pièces mangées dans le temps à disposition : la valeur a donc une borne supérieure de 89 (le nombre de pièces sur la piste). Pour notre expérience, cette définition n'est pas possible puisque Romero ne peut pas distinguer deux cases dans un bloc contigu de pièces. Le problème est contourné de la façon suivante : la fitness est le temps (en secondes) passé sur des pièces de nourriture (à noter que ce temps est proportionnel au nombre de pièces mangées).

$$fitness = \sum_{i=1}^{blocks} (t_{out} - t_{in}),$$

blocks est le nombre de blocs contigus, $t_{out} - t_{in}$ est le temps passé dans ce bloc. A noter que les codons ne permettent pas que le robot tourne sur place ; par contre, il est possible de faire une boucle plus grande et de passer plusieurs fois par la même bloc.

4.4 L'expérience.

L'histoire des jumeaux Romero n'a pas encore de *happy end*, malgré le fait qu'ils ont déjà travaillé plusieurs jours dans leurs environnements réels. La figure 4 montre les 2 meilleures solutions, qui, en fait, ramassent à peu près 75% des pièces de nourriture.

Nous avons l'intention de faire des expériences dans un futur proche avec deux robots qui collaborent sur une même piste pour ramasser les pièces de nourriture.

Des sujets très intéressants, tels que l'auto-organisation, la co-évolution, le comportement d'une population et la compétition entre individus, pourront être étudiés sur une grande population de vrais robots.

Plusieurs extensions matérielles sont planifiées, comme par exemple l'évolution d'un circuit programmable qui contrôle le matériel directement, ou des modules supplémentaires pour la vision et la navigation.

En résumé, nous sommes convaincus que notre contrôleur EvoMaster avec ses extensions constitue un pas important dans la direction des robots bon marché, offrant un outil de choix pour les chercheurs actifs dans le domaine de la vie artificielle.

Remerciements.

Nous remercions André Badertscher pour les photos et l'assemblage des circuits imprimés.

Références

- [1] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic Programming – An Introduction : On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann Publishers, San Francisco, 1997.
- [2] R. J. Collins and D. R. Jefferson. AntFarm : Towards simulated evolution. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, volume X of *SFI Studies in the Sciences of Complexity*, pages 579–601, Redwood City, CA, 1992. Addison-Wesley.
- [3] D. Floreano. Evolutionary robotics in artificial life and behavior engineering. In T. Gomi, editor, *Evolutionary Robotics*, AAI Books. Ontario (Canada), 1998.
- [4] B. Holldobler and E. O. Wilson. *The Ants*. Harvard University Press, Cambridge, Massachusetts, 1990.
- [5] S.-G. Hong and J.-J. Lee. A local motion planner for car-like robots in a cluttered environment. *Artificial Life and Robotics*, 1 :39–42, 1997.
- [6] D. Jefferson, R. Collins, C. Cooper, M. Dyer, M. Flowers, R. Korf, C. Taylor, and A. Wang. Evolution as a theme in Artificial Life : The Genesis/Tracker system. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, volume X of *SFI Studies in the Sciences of Complexity*, pages 549–578, Redwood City, CA, 1992. Addison-Wesley.
- [7] J. R. Koza. *Genetic Programming : On the Programming of Computers by Means of Natural Selection*. The MIT Press, Cambridge, Massachusetts, 1992.
- [8] LEGO Lab. <http://legolab.daimi.aau.dk>.
- [9] P. Miller. Consumer electronics. *IEEE Spectrum*, 36(1) :41–45, January 1999.
- [10] F. Mondada, E. Franzi, and P. Jenne. Mobile robot miniaturization : A tool for investigation in control algorithms. In T. Yoshikawa and F. Miyazaki, editors, *Proceedings of the Third International Symposium on Experimental Robotics*, pages 501–513. Springer Verlag, 1993.
- [11] N. Roy and S. Thrun. Online self-calibration for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 1999.
- [12] A. Tettamanzi and M. Tomassini. Evolutionary algorithms and their applications. In D. Mange and M. Tomassini, editors, *Bio-Inspired Computing Machines : Toward Novel Computational Architectures*, pages 59–98. Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland, 1998.
- [13] S. Thrun. Bayesian landmark learning for mobile robot localization. *Machine Learning*, 33(1), 1998.
- [14] B. Webb. A cricket robot. *Scientific American*, 275(6) :62–67, December 1996.