Robotic Swarm Cooperation by Co-adaptation

François-Michel De Rainville
Laboratoire de vision et systèmes numériques
Département de génie électrique et de génie informatique
Université Laval, Québec (Québec), Canada G1V 0A6
francois-michel.de-rainville.1@ulaval.ca

ABSTRACT

This paper presents a framework for co-adapting mobile sensors in hostile environments to allow telepresence of a distant user. The presented technique relies on cooperative co-evolution for sensor placement. It is shown that cooperative co-evolution is able to find simultaneously the required number of sensors to observe a given environment and a configuration that is consistently better than other well know optimization algorithms. Moreover, it is presented that co-evolution is also able to quickly reach a new configuration when the environment changes.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics—*sensors, autonomous vehicules*; G.1.6 [**Numerical Analysis**]: Optimization—*global optimization*

General Terms

Algorithms, Performance

Keywords

Evolutionary and swarm robotics, mobile sensor network, co-adaptation, visibility

1. INTRODUCTION

Recent democratization of mobile computing and miniaturized robotics paved the way for using swarms of small and relatively simple robots to accomplish tasks that a sophisticated and expensive one cannot achieve alone. In fact, the distributed fashion in swarm robotics grants three valuable characteristics that are hard to fulfill for a single robot: robustness, flexibility, and scalability [13]. The application of swarm robotics to inspection and surveillance is appealing as the general objective is to gather information rapidly, safely, and reliably, especially when exploring hazardous or inaccessible environments. For example, a swarm of autonomous robots would be of great utility in investigating environments such as defective nuclear power plants (e.g. Fukushima) and distant locations (e.g. Mars).

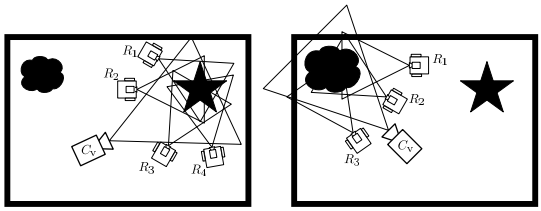


Figure 1: Mobile sensor placement in an environment according to the desired virtual points of view.

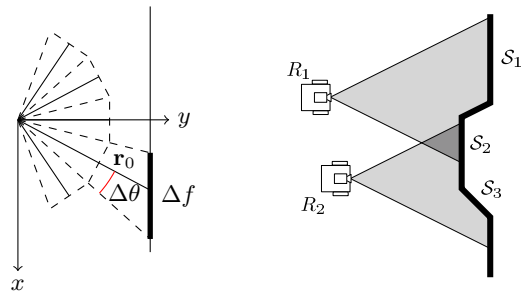
The current research project investigates coordination of mobile sensors in unfriendly dynamic environments. In particular, we are interested in placing a limited number of mobile sensors in such manner that an external user can observe a scene, inaccessible for different reasons, from a desired point of view (even if no physical sensor is or can be placed at this location). To achieve their goal of reproducing the *virtual* point of view of the environment, the robots must coordinate and optimize their sensing actions according to the geometry of the environment. As shown in Figure 1, we want a swarm of sensors (R_1 to R_4) to configure itself automatically so that the necessary information about the environment is gathered for the virtual point of view (C_v) be reprojected. Although C_v is represented as a normal camera, it can be of any imaginable type of sensor since it is virtual, e.g. three-dimensional, wide angle, omniscient, etc. It is only limited by the data collected by the real sensors.

This paper proposes a framework for co-adapting the configuration of a swarm of mobile sensors to render desired virtual points of view of environments. Section 2 of this work describes the world representation used internally by the robots and the way we estimate their visibility. Then, Section 3 puts forward a co-evolutionary technique for co-adapting the configuration of the robots given the environment. Section 4 presents preliminary results on co-adapting a swarm of robots to sense a dynamic environment. Finally, Section 5 concludes on the impact of the proposed method and propounds several avenues for the project future.

2. ESTIMATING VISIBILITY

Estimating the visibility of a sensor at a certain position is primordial in order to optimize its position. Visibility can be defined by how much a sensor covers the environment

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'12 Companion, July 7–11, 2012, Philadelphia, PA, USA.
Copyright 2012 ACM 978-1-4503-1178-6/12/07 ...$10.00.



(a) Simulation of a sensor footprint.

(b) Combining multiple sensors footprint.

Figure 2: Estimation of the sensor coverage.

surface. If seen as projectors, the sensors project triangular beams that produce a footprint on the surfaces of the environment as seen on Figure 2(a). This section presents the technique used to compute the area observed by one or multiple sensors based on their footprint.

2.1 Environment Representation

The environment must first be represented internally so that the visibility of any given sensor can be estimated. The occupancy grid [4, 11], partitioned in an *octree* [10], offers an ideal geometrical model for its capability to represent complex three-dimensional spaces. An occupancy grid is a volumetric model in which the environment is subdivided in cells of equal volume. The octree partitioning of the occupancy grid allows memory efficient representation of large uniform space and fast ray-tracing operations [2, 14].

2.2 Sensor Simulation

As shown in Figure 2, sensors can be simulated by using a ray casting technique. In fact, by tracing rays $\mathbf{r}_i$ in certain directions from the sensor position we can retrieve the first occupied cells hit by those rays. If we know the normal of the surface hit and the distance between the sensor and this surface we can easily compute the triangle bases Δf. Once all bases are calculated summing them up gives us the footprint of the sensor. Again, our representation allows us to efficiently reconstruct the normals of the environment through the marching cubes algorithm [9].

In a multi-sensor configuration, the sensors footprint can overlap. This leads to a total effective footprint that is smaller than the sum of each sensor footprint taken separately. As shown in Figure 2(b), sections $\mathcal{S}_1$ and $\mathcal{S}_3$ are direct results of each sensors footprint, but section $\mathcal{S}_2$ is seen by both sensors. The estimation of the footprints combination is obtained by using a single sensor footprint in the overlapping area.

3. CO-ADAPTING ROBOTS

Evolutionary optimization of sensor networks has been addressed in the last decade [1, 7, 15]. Most of these works considered the sensor network has a whole and implemented the optimization of all sensors of the network at the same time. Here we propose to break down the optimization into several sub-problems that are resolved cooperatively by multiple sub-solutions [12].

3.1 Cooperative Co-evolution

Additionally to breaking down the problem, cooperative co-evolution offers a simple mechanism to adapt the number of species to the problem at hand. In fact, when the evolution stagnates for a certain number of generations, namely when the fitness of the individuals stops improving, a new species is added to the population and species not contributing enough to the evolution are removed. The right number of sensors is thus emerging naturally from the evolutionary process.

```
1  initialize 𝔓 ← {𝔖_i, i = 1,...,n}
2  choose ℜ ← {select_random(𝔖_i), i = 1,...,n}
3  while ¬stop do
4  |  ℜ' ← {}
5  |  foreach species 𝔖_i ∈ 𝔓 do
6  |  |  𝔖_i ← mate(𝔖_i)
7  |  |  𝔖_i ← mutate(𝔖_i)
8  |  |  evaluate(𝔖_i, ℜ\r_i)
9  |  |  ℜ' ← ℜ' ∪ {select_best(𝔖_i)}
10 |  |  𝔖_i ← select(𝔖_i)
11 |  end
12 |  ℜ ← ℜ'
13 |  if improvement < T_i then
14 |  |  remove species with contribution < T_c
15 |  |  ℜ ← ℜ\ℜ⁻
16 |  |  add a new species 𝔓 ← 𝔓 ∪ {𝔖'}
17 |  |  ℜ ← ℜ ∪ {select_random(𝔖')}
18 |  end
19 end
```

Algorithm 1: Cooperative co-evolution.

Algorithm 1 presents the pseudo-code for co-evolution as introduced by Potter and De Jong [12]. It shows how each species $\mathfrak{S}_i$ of the population $\mathfrak{P}$ is evolved independently of the others and how interaction through evaluations is acheived. More specifically, on line 8, the evaluation of the species individuals is made in collaboration with the species representatives excluding the representative of the current species $\mathfrak{R}\backslash\mathbf{r}_i$. The inner loop, beginning at line 5, is a classical genetic algorithm with crossover, mutation, and selection steps. At each ecosystem generation, i.e. the outer loop beginning at line 3, the representatives are updated with the best individual of each species for the next generation. Then, the improvement of the evolution is verified at line 13. If the fitness of the solution given by all representatives has not improved over the last generations by more than a given threshold T_i, then the system is stagnating. In this situation, unproductive species that contribute less than a threshold T_c are removed from the population[1] – their representative ($\mathfrak{R}^-$) are also removed from $\mathfrak{R}$ – and one new species is added to the population. The evolution continues until a termination criterion is reached.

3.2 Evolving Sensors

In our framework, we evolve homogeneous sensors represented by a position and an orientation in a float vector. We use a simple float vector genetic algorithm as the inner loop of our co-evolution with α-blend crossover [5], gaussian

[1]Newer species are removed first and the contribution is recalculated after each removal favouring older species.

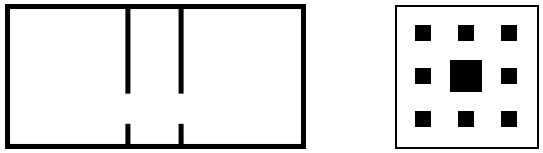


Figure 3: Static (left) and dynamic (right) worlds.

mutation, and tournament selection, respectively on lines 6, 7, and 10 of Algorithm 1.

The evaluation of the individuals is made by combining the evaluated individual with the representatives of the other species and simulate them in the occupancy grid as mentioned above. The fitness is assigned only to the individual being evaluated. Each species contribution to the cooperative fitness is evaluated by subtracting the fitness of its representative alone to the fitness of the group of sensors. This allows the identification of species that are unproductive (contribution smaller than the threshold T_c) and shall be removed. Finally, the improvement is computed over a generation window. If the cooperation fitness of the representatives has not increased enough during that window (cooperation smaller than the threshold T_i), the stagnation mechanism to add a new species and remove the unfit ones is executed.

4. PRELIMINARY EXPERIMENTS

Our preliminary experiments concern the placement of sensors in complex static and dynamic environments using the presented co-evolutionary technique. Both environments are presented in Figure 3. In the dynamic world, only two small blocks are present at a time. They start from the north-west and south-east positions and rotate counter clockwise to the next positions.

We want to reproduce an omniscient sensor that is seeing all static parts of the environment, i.e. sensors are rewarded only for there footprint area that is on static parts of the environment. With those experiments, we want to see if a cooperative co-evolutionary algorithm is able to

- find a good enough configuration of sensors so that most of the environment is sensed,
- find the number of sensors required to accomplish the task, and
- adapt the found configuration when the environment changes.

Figures 4 and 5 present respectively the evolution of the sensor coverage in the static environment and a solution found by the co-evolutionary algorithm. The jumps in the fitness function are produced by the addition of sensors while slower increases are due to optimization of the network. As we see, after 60 000 evaluations, sensors are also deleted (decrease of the fitness function) keeping only sensors that have a significant contribution to the sensors network. Additionally to find the required number of sensors, the co-evolutionary technique discovers better solutions than a canonical Particle Swarm Optimization (PSO) [8] and Covariance Matrix Adaptation Evolution-Strategy (CMA-ES) [6], that are set to evolve a fixed number of sensors.

Figures 6 and 7 show the optimization of the coverage in a dynamic environment, changing at three different paces. As

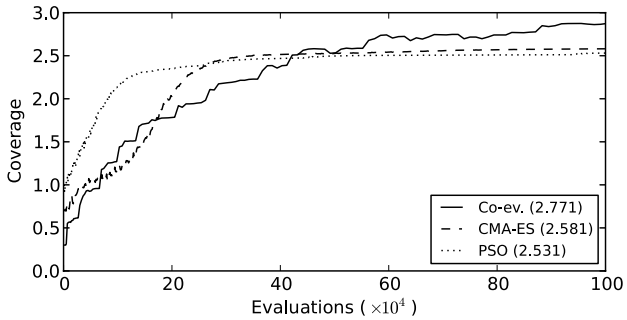


Figure 4: Optimization of the visibility in a static environment.

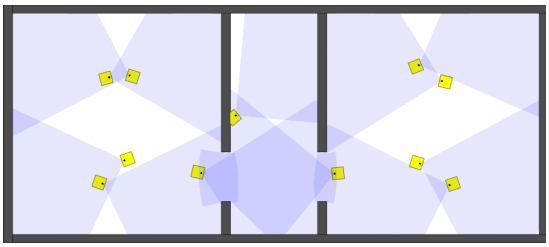


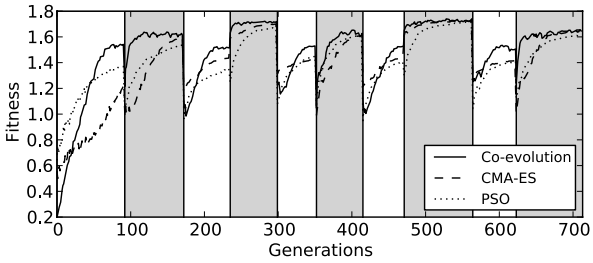
Figure 5: Co-adapted configuration of 11 sensors.

we can see in all three cases, the co-evolution is able to reconfigure the sensors conveniently and rapidly after a change, while CMA-ES and PSO seems to have more difficulties to adjust the sensors configuration. In fact, once the right number of sensors is found the co-evolutionary methods shows faster convergence in all three situations. We also note that the most important mechanism of the co-evolutionary technique in dynamic environment is the stagnation detection that acts like a restart for sensors that are the most affected by a change. Indeed, shortly after a change the sudden deterioration of the fitness forces the algorithm to enter its stagnation phase, thus deleting unfit sensors and adding a new sensor. This mechanism helps the algorithm to quickly reach a new configuration that suits better the current environment geometry.

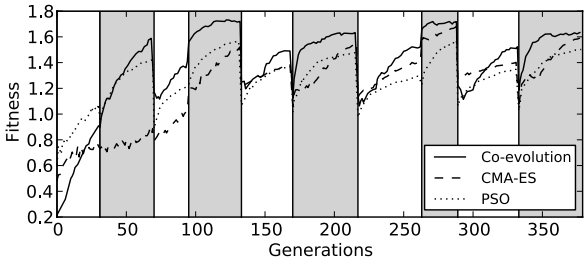
5. CONCLUSION

This paper presented an efficient cooperative co-evolutionary technique to optimize the configuration of a swarm of robots in static and dynamic environments. Moreover, it can find simultaneously a good configuration and the number of sensors to maximize the coverage of the surfaces in a given environment. The presented technique has also been shown to produce better results than other evolutionary techniques with equivalent computational effort. When applied to swarm robotics, this technique provides a way to place robots in an environment so that an area of interest can be efficiently observed using the right number of robots.

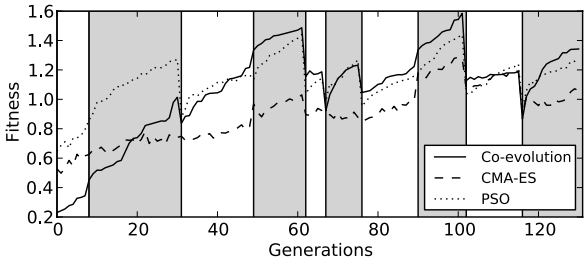
The current work can be extended in many ways. In fact, the currently studied avenues for the framework span in two major directions. First, we are working on specializing the internal representation of the world so that information on moving objects is included directly in the occupancy grid. This would allow the cost function of our optimization algorithm to treat moving cells differently than unknown, free,



(a) slow



(b) medium



(c) fast

Figure 6: Optimization of the visibility in a dynamic environment.

and occupied cells. Furthermore, knowing that some cells of the occupancy grid are produced by moving objects, we can build a host/parasite type of competitive co-evolutionary algorithm between different moving objects configurations and different sensors placements. This would help to find a sensors arrangement that is resilient to movements. The second direction concerns directly the optimization algorithm. In fact, our problem is part the dynamic objective function problem class and we wish to investigate how well evolutionary algorithms can tackle such kind of problems, following what has been proposed by Branke [3].

Acknowledgements

This work is funded by the FQRNT (Québec) and NSERC (Canada).

6. REFERENCES

[1] V. Akbarzadeh, C. Gagné, M. Parizeau, and M. A. Mostafavi. Black-box optimization of sensor placement with elevation maps and probabilistic sensing models. In *Proc. of the International Symposium on Robotics and Sensor Environments*, pages 89–94, 2011.

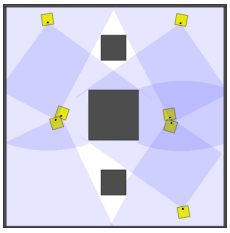


Figure 7: Co-adapted configuration after the fourth movement occurred in the medium pace environment.

[2] J. Amanatides and A. Woo. A fast voxel traversal algorithm for ray tracing. In *Proc. of Eurographics*, 1987.

[3] J. Branke. *Evolutionary optimization in dynamic environments*. Kluwer Academic Publishers, 2001.

[4] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[5] L. J. Eshelman and J. D. Schaffer. Real-coded genetic algorithms and interval-schemata. In *Foundations of Genetic Algorithms*, pages 187–202, 1993.

[6] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[7] D. B. Jourdan and O. L. de Weck. Layout optimization for a wireless sensor network using a multi-objective genetic algorithm. In *Proc. of the Vehicular Technology Conference*, pages 2466–2470, 2004.

[8] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, 2001.

[9] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *SIGGRAPH Computer Graphics*, 21(4):163–169, 1987.

[10] D. Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129–147, 1982.

[11] H. P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2):61–74, 1988.

[12] M. A. Potter and K. A. De Jong. Cooperative coevolution : An architecture for evolving co-adapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2001.

[13] E. Şahin. Swarm robotics: From sources of inspiration to domains of application. In E. Sahin and W. Spears, editors, *Swarm Robotics*, volume 3342 of *Lecture Notes in Computer Science*, pages 10–20. 2005.

[14] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Proc. of the ICRA Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, 2010.

[15] Y. Xu and X. Yao. A GA approach to the optimal placement of sensors in wireless sensor networks with obstacles and preferences. In *Proc. of the Consumer Communications and Networking Conference*, pages 127–131, 2006.