uc3m | Universidad Carlos III de Madrid

e-Archivo

This is a postprint version of the following published document:

Fernandez-Fernandez, R., Estevez, D., Victores, J. G. & Balaguer, C. (26-28 April 2017). *Reducing the number of evaluations required for CGDA execution through Particle Swarm Optimization methods* [proceedings]. 2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Coimbra, Portugal.

DOI: 10.1109/ICARSC.2017.7964089

© 2017, IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Reducing the Number of Evaluations Required for CGDA Execution through Particle Swarm Optimization Methods

Raul Fernandez-Fernandez, David Estevez, Juan G. Victores and Carlos Balaguer

***Abstract*—Continuous Goal Directed Actions (CGDA) is a robot learning framework that encodes actions as time series of object and environment scalar features. As the execution of actions is not encoded explicitly, robot joint trajectories are computed through Evolutionary Algorithms (EA), which require a large number of evaluations. The consequence is that evaluations are performed in a simulated environment, and the optimal robot trajectory computed is then transferred to the actual robot. This paper focuses on reducing the number of evaluations required for computing an optimal robot joint trajectory. Particle Swarm Optimization (PSO) methods have been adapted to the CGDA framework to be studied and compared: naïve PSO, Adaptive Fuzzy Fitness Granulation PSO (AFFG-PSO), and Fitness Inheritance PSO (FI-PSO). Experiments have been performed for two representative use cases within CGDA: the "wax" and the "painting" action. The experimental results of PSO methods are compared with those obtained with the Steady State Tournament used in the original proposal of CGDA. Conclusions extracted from these results depict a reduction of the number of required evaluations, with simultaneous tradeoff regarding the degree of fulfillment of the objective given by the optimization cost function.**

I. Introduction

In robot imitation, robots learn actions from a set of user real world demonstrations. The selection of a model for the robot to internally represent a generalized action is a core decision, that greatly defines a framework's characteristics and possibilities. Different robot imitation frameworks have used different internal generalized models to represent real world actions. Programming by Demonstration (PbD) has used both Hidden Markov Models [1] and Gaussian Mixture Models [2] to encode actions as robot joint and Cartesian space trajectories. Dynamic Motion Primitives (DMP) encode actions as control laws that lead to Cartesian space trajectories [3]. Finally, Continuous Goal Directed Actions (CGDA) encodes actions as time series of object and environment scalar features, modelling the continuous changes upon the environment [4]. This way, in CGDA, an action is defined by the effects that it produces in the environment, rather than the demonstrator action that made them occur during the demonstrations. One of the main advantages of CGDA is that actions are independent from robot or demonstrator kinematics, which means that their definition is independent from the differences between the structure and morphology of the demonstrator and the robot. Therefore, the correspondence problem, a common problem in PbD systems, is not present in CGDA. An action regarding moving an object can be encoded using X,Y,Z coordinates of the object's centroid (action described by three scalar features). A wall painting action, however, can be represented using only the percentage of object or environment that has been painted (action described by a single scalar feature). The actual scalar features that are relevant for a specific action can be handcrafted, or may be automatically extracted using the CGDA demonstration and feature selection algorithm [5].

Since the execution of actions is not encoded explicitly in CGDA, robot joint trajectories are unknown. They are normally computed through Evolutionary Algorithms (EA), which require a large number of evaluations. These evaluations are extremely time consuming, and it becomes infeasible to perform them on an actual physical robotic platform. They are therefore performed in a simulated environment, and the final optimal computed robot joint trajectory is then transferred to the actual physical robot. The long-term goal within CGDA execution is to be able to perform all the evaluations directly on the physical robotic platform, which will be feasible when the number of evaluations required is sufficiently reduced.

An overview of the CGDA framework will be given in section II. Different approaches and methods that deal with reducing the number of required evaluations are studied in sections III and IV. Experiments and results will be presented in section V, leading to the final section of conclusions.

II. The CGDA Framework

As presented in [4], CGDA is a way to encode the effects of an action, when the action is demonstrated to a robot. The CGDA framework is used for generalizing, recognizing and executing actions by their effects on the environment. A continuous analysis generates a trajectory in a n-dimensional feature space, where n equals the number of tracked object scalar features. A simplified block diagram of CGDA framework can be found on Fig. 1.

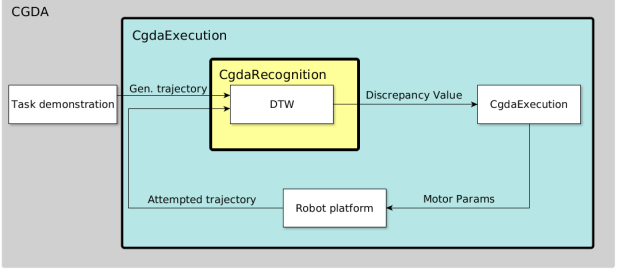

Fig. 1. Continuous Goal-Directed Actions (CGDA) framework diagram.

A. Generalization

The process of generalizing consists on extracting a representative n-dimensional feature trajectory of the task from several repetitions. First, the demonstrated actions are discretized

©2017 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. DOI: 10.1109/ICARSC.2017.7964089

and normalized in time. Then, the generalized trajectory is obtained using a Radial Basis Function interpolation between time intervals of a fixed duration with the population of demonstrated actions. A generalization example can be seen in Fig. 2, extracted from [6].

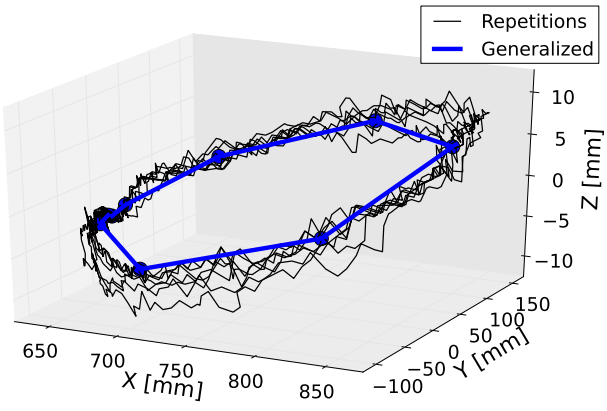

Fig. 2. Plot representing a three feature trajectory. Black lines are user action repetitions. The blue line is the generalization of all the repetitions.

B. Recognition

The generalized trajectory can be used as a tool to compare the action it represents with another action. The goal of the recognition step is to obtain a metric of the discrepancy between these actions.

Recognition is performed using an intermediate result of the Dynamic Time Warping (DTW) algorithm. The DTW algorithm is usually used to optimally align two temporal sequences [7]. This is done by evaluating all pairs of points between the sequences. A cost matrix is obtained between the sequences using a norm such as L^2. The path in the matrix with the lowest cost is the alignment that minimizes the discrepancy between the two sequences.

In CGDA, this cost of alignment is the used DTW intermediate result. It describes the discrepancy between two actions if only a single feature was taken into account. The total recognized discrepancy between two actions is the sum of the costs of each individual feature.

C. Execution

CGDA is considered as both a way to encode actions to be recognized, and to be executed by robots. However, since CGDA does not encode joint motor parameters, conventional methods can not be used for execution. Due to this fact, several strategies have been studied by the authors based on Evolutionary Algorithms (EA), where the recognition discrepancy was used as the fitness.

The currently most successful strategy is the Incrementally Evolved Trajectories (IET) [6] strategy. The idea behind IET is to perform an individual evolution over each of the time intervals of the feature trajectory. Then, the fitness of the new point is the fitness of evaluating the execution of the previous points (already evolved) and the new one (the one that is being evolved). The main advantage of using this method is that it reduces the search space of the problem by a coefficient of T, where T is the number of time intervals. This directly translates to an important reduction in the number of evaluations and therefore a speed-up of the system, which is one of the main concerns when working with EA.

In the following sections, different methods that aim to reduce the number of evaluations needed for EA will be discussed.

III. Reducing the Number of Evaluations in EA

Reducing the number of evaluations is a challenge not only present in CGDA. It is an intrinsic problem related to working with Evolutionary Algorithms (EA). This kind of algorithms shines in situations where the problem to optimize is very complex or not well defined. EA require performing a large number of iterations and evaluations, which has become possible with the emerge of more and more powerful computers. However, the problem of reducing the number of evaluations is still present in time sensitive scenarios, and in real world applications where performing hundreds or thousands of evaluations becomes unfeasible. Different approaches have been proposed for the reduction of the number of evaluations required for EA to converge. These approaches can be divided in three different groups [8]:

- **Problem Approximation:** These methods try to replace the original definition of the problem for a simplified version of the same problem.
- **Functional Approximation:** Here the cost function is approximated with a mathematical function that is simpler to solve. An example is [9], where the objective of the authors was to increase the speed of a Differential evolution (DE) algorithm. With this in mind, a second order function approximation of the cost function was used, simplifying the problem function, and reducing the number of evaluations required.
- **Evolutionary Approximation:** In this case the EA is the one which is simplified in order to reduce the number of evaluations. There are two kinds of methods that can be included in this group: Fitness Inheritance (FI) [10] and Fitness Approximation (FA) [8].
  In FI, for a proportion of random particles in each iteration, the fitness value is calculated using an approximate fitness formula. This way, the number of evaluations needed is reduced by that proportion. FI is used in [11] for the optimization of chemotherapy dose schedules. This is a critical scenario where time is a key factor. Due to the complexity of the model (multiple drugs, schedules, effects...), and the large number of evaluations, this is usually a computationally expensive problem.
  On the other hand, in FA, fitness clusters are generated in the cost function. Then, new particles placed in the cluster take the fitness value of the cluster, without a need of evaluation. FA is used in [12] for bot evolution in the computer game Unreal Tournament 2004™. The goal

here is to reduce the number of evaluations. The reason is because each evaluation requires simulation at playtime, so it is very costly. Fitness Fuzzy Approximation is used in [13], for reducing the number of evaluations required for optimizing a start-up phase of a combined cycle power plant.

IV. Particle Swarm Optimization in CGDA

In the original proposal of the CGDA framework, a Steady State Tournament (SST) method was used for the execution of actions. In this paper, the implementation of the most promising state of the art Particle Swarm Optimization (PSO) methods for reducing the number of evaluations is presented and studied within the CGDA framework.

A. Particle Swarm Optimization (PSO)

The idea of PSO was first introduced by Kennedy [14]. The idea behind PSO was to create an optimization method based on social interactions, rather than individual behaviors. In this method, a particle population is placed in the search space of some function, and each individual particle is evaluated. Then, for every iteration, each particle moves to a different position, and then, is evaluated again. The movement of each particle is a function of combining the particle movement, and the best particle personal position, with the position of one or more members of the swarm [15].

B. Adaptive Fuzzy Fitness Granulation PSO (AFFG-PSO)

The idea of AFFG was initially proposed in [16]. This method is part of the Fitness Approximation (FA) group. It is based on the idea of clustering the different individuals of the EA in granules. In this method, these granules correspond to Gaussian distributions. If a new individual is similar enough to an already known granule, it is assigned with the fitness of that granule and not evaluated. If this does not happen, the fitness of the new individual is normally evaluated, and a new granule is created. The experiments in [17], in certain problems, show that the number of evaluations was reduced by almost the 90%, while reaching statistically similar performance in terms of fitness optimization. The integration of AFFG with PSO (AFFG-PSO) is not part of previous literature, but is an original contribution of this paper.

C. Fitness Inheritance PSO (FI-PSO)

Fitness Inheritance (FI) was initially proposed in [10] as a solution to the high computational cost of evaluating each individual of a population. The idea was to propose a method where a portion of the population is normally evaluated, while the fitness of the rest of the population is obtained as an approximation of their parent's fitness.

In [18], the authors run a study about the feasibility and performance of FI in a real world scenario. In these experiments they concluded that FI strategies only had good performance when evaluated in convex functions, while not being able to reach the optimal solution in non-convex ones.

Later, in [19] the authors study the implementation of FI in a PSO algorithm (FI-PSO). However, the results here were quite different than the obtained in [18]. The FI-PSO algorithm was able to reach the optimal solution even in non-convex functions. In [20] they also proposed different modifications of the FI-PSO algorithm proposed in [19]. Here, a method based on the flight formula of PSO was the one which performed best.

V. Experiments and Results

Four different methods have been used for the experiments of this paper: Steady State Tournament (SST) [21] (used in the original proposal of CGDA), naïve PSO [14], a proposal of a modified version of PSO with Adaptive Fuzzy Fitness Granulation [17] (AFFG-PSO), and the Fitness Inheritance PSO (FI-PSO) algorithm as proposed in [20]. The algorithms have been adapted to the CGDA architecture, implemented and open-sourced[1].

From previous works with SST, a large number of evaluations was expected for convergence of the algorithms. Experiments were performed in a simulated environment using OpenRAVE [22]. The robotic platform used for the simulation was TEO[2], the humanoid robot from the Robotics Lab of Universidad Carlos III de Madrid [23]. For both experiments, 3 of the 6 joints of the right arm were used, maintaining all other joints (including torso, legs and head) static. The experiments consisted on the executions of the "wax" (also known as "clean") and the "paint" actions using the IET strategy as proposed in [6].

A. WAX

The goal of the "wax" action is the movement of the object's centroid following a circumference of 30 cm of diameter for one revolution. The three scalar features tracked by the CGDA system in this action are the Cartesian's coordinates (X,Y,Z) of the object's centroid. While this setting of the "wax" action makes it equivalent to solving the inverse kinematics, its purpose is to demonstrate how the CGDA framework returns results within the expected ranges, despite it is agnostic with respect to the nature of the given scalar features.

For all methods, the population of individuals (collections of 3 joint parameters) was set to 50. The termination condition for the system to converge was set to 3 consecutive generations without improvement of the found solution. Joint parameters movements were restricted between -15 and 100 degrees. For SST, the individual mutation probability was set to 60%. The PSO inertia weight was 1.2, and the maximum particle velocity was 5. The inheritance proportion of FI-PSO was set to 55%.

The results of the experiments for the "wax" action are represented in Table I. The results presented are the average of running the "wax" action 50 times.

In Fig. 3, the cumulative number of evaluations needed at each time interval (set at 1 s) is represented. In Fig. 4 the generated trajectory for each of the methods is plotted and compared with the generalized one.

[1]https://github.com/roboticslab-uc3m/xgnitive

[2]Model available at https://github.com/roboticslab-uc3m/teo-main

TABLE I
EXPERIMENTAL RESULTS FOR THE "WAX" ACTION

Method	Evaluations	Fitness (DTW discrepancy)
SST	9679	274
PSO	8470	213
AFFG-PSO	5314	434
FI-PSO	3432	362

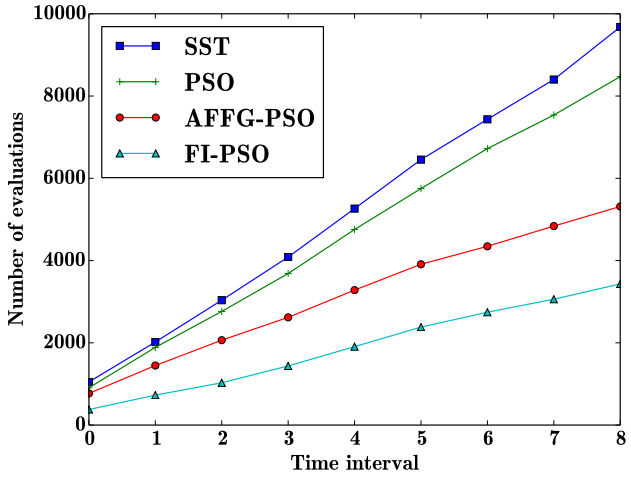

Fig. 3. "Wax" experiment: Cumulative number of evaluations at each time interval (set at 1 s).

The results from these experiments show how FI-PSO is the method that is capable of reducing the number of evaluations the most. Compared to the results obtained with the original SST approach, the number of evaluations is reduced by a 65%, while the error is increased by a 25%.

B. PAINT

The objective of the "paint" action is to have the robot to paint a wall. The only scalar feature tracked in this case is the percentage of the wall painted. Fig. 7 depicts an example of this action execution.

For all methods, the population of individuals (collections of 3 joint parameters) was set to 10. The termination conditions for the evolution process was to reach a zero error in the obtained trajectory, or to experience 10 followed generations without any improvement in the fitness value. This number of generations for the termination condition was increased with respect to the "wax" action, due to the expected faster convergence of the "paint" action. Joint parameters movements were restricted between -15 and 100 degrees. For SST, the individual mutation probability was set to 60%. The PSO inertia weight was 1.2, and the maximum particle velocity was 5. The inheritance proportion of FI-PSO was set to 55%.

The results obtained in the experiments are the ones represented in Table II. The results presented are the average of running the "paint" action 100 times.

In Fig. 5, the cumulative number of evaluations needed at each time interval (set at 1 s) is represented. Fig. 6 depicts a

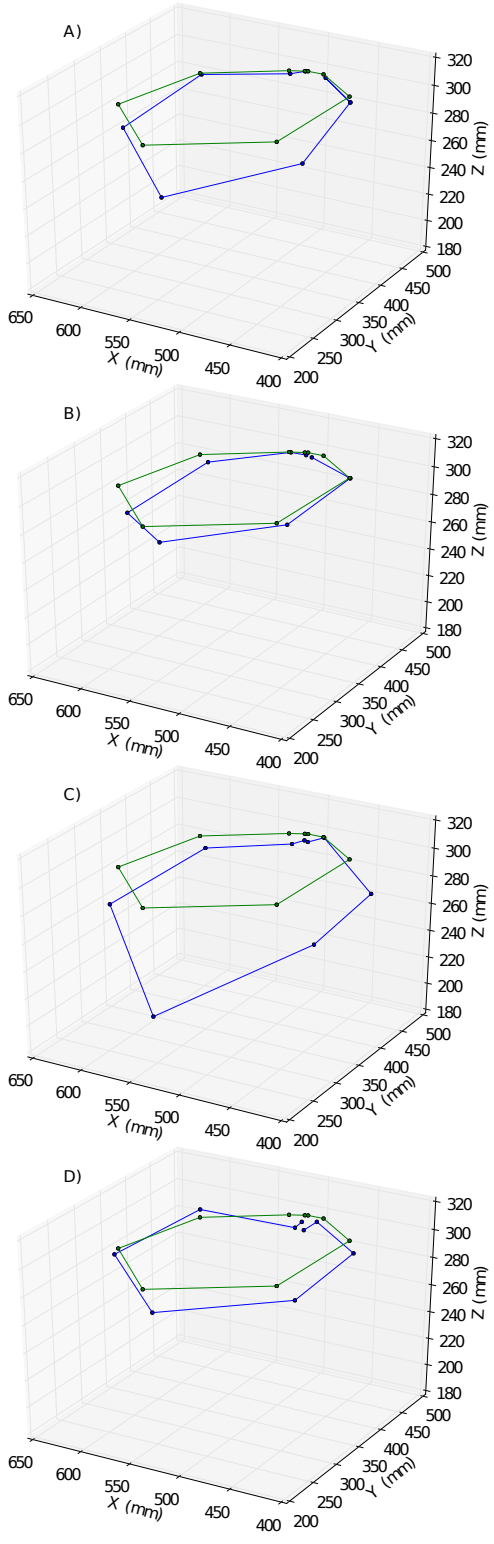

Fig. 4. Obtained trajectories for the "wax" action studied in this paper. The green line corresponds to the generalized trajectory. The methods used for each case are the following: A)SST, B)PSO, C)AFFG-PSO and D)FI-PSO.

TABLE II
EXPERIMENTAL RESULTS FOR THE "PAINT" ACTION

Method	Evaluations	Wall Painted (%)
SST	539	94.4
PSO	583	91.44
AFFG-PSO	537	89.75
FI-PSO	441	87.88

comparison between the feature trajectories obtained for each of the methods and the generalized one.

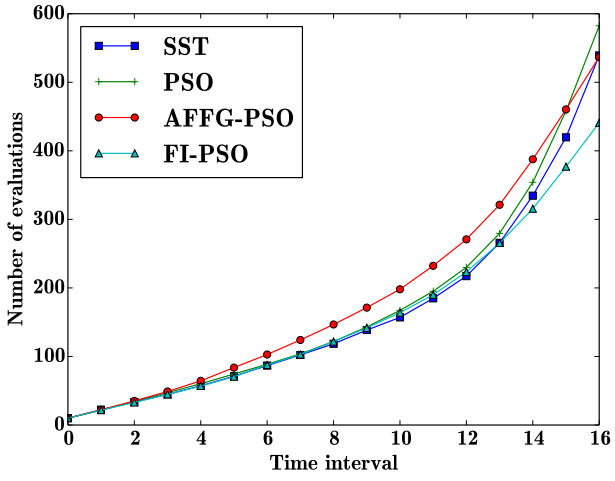

Fig. 5. "Paint" experiment: Cumulative number of evaluations at each time interval (set at 1 s).

In this experiment, FI-PSO again required the least number of evaluations for the execution of the task. This method needed 17% less evaluations than in the case of using SST, with a tradeoff of an average of 7% less painted wall.

VI. CONCLUSIONS

The results obtained during the experiments show that FI-PSO was the method that was able to reduce the most the number of evaluations for both of the actions, at the cost of introducing an error in the action performance. In the case of the "wax" action, this method was able to reduce by 65% the evaluations needed for the system, with a 24% error increment. In the "paint" scenario, a 17% reduction in the number of evaluations was achieved, with a tradeoff of a 7% less painted wall.

In the case of AFFG-PSO, however, the results did not meet the expectations arisen from the related literature. In both experiments scenarios, the results were worse compared with the ones obtained with FI-PSO. In the authors' opinion, the reason of this worse performance was due to the high dependence between its performance and the correct tuning of the internal parameters of the method. Some research should be done in this direction, since there is a lack of literature regarding the correct tuning of the AFFG methods. In this case, this is aggrieved by the fact that in IET, each time interval

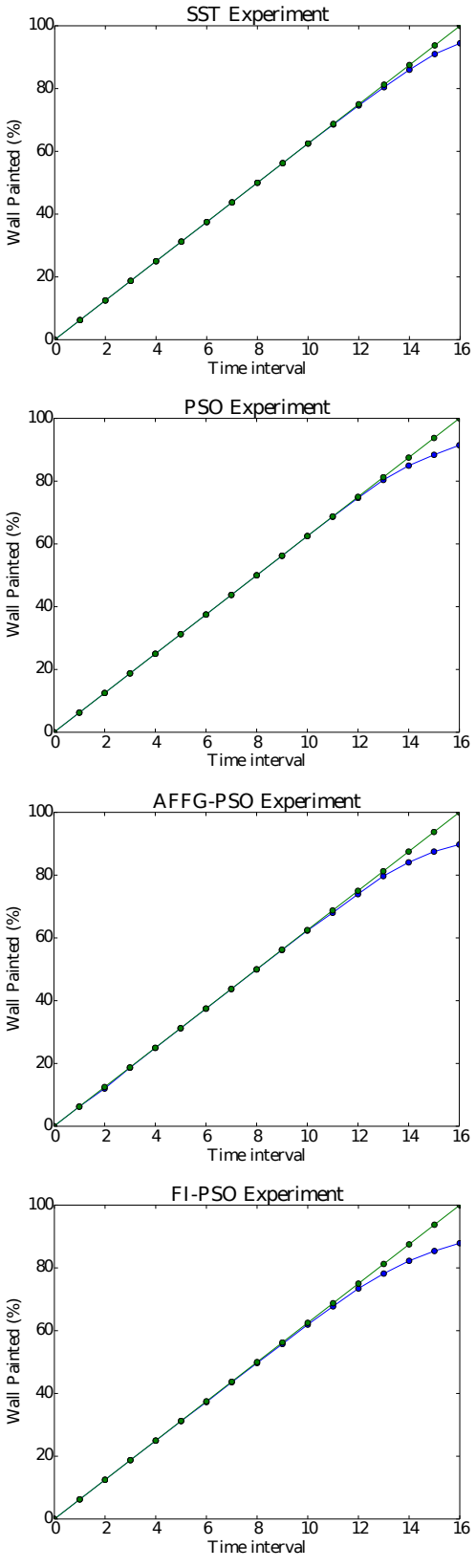

Fig. 6. Obtained trajectories for the "paint" action studied in this paper. The straight line corresponds to the generalized trajectory. The methods used for each case are the following: SST, PSO, AFFG-PSO and FI-PSO.

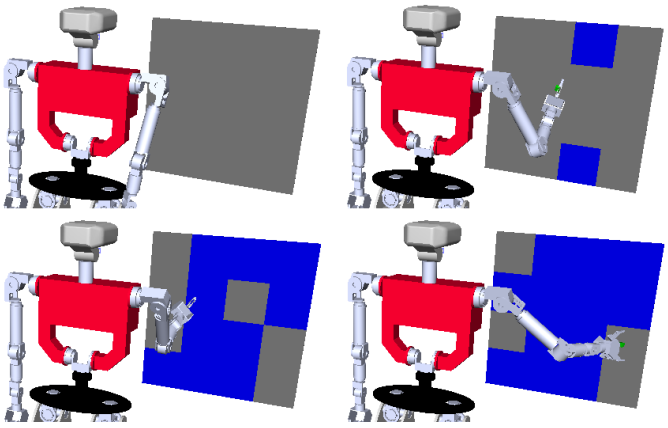

Fig. 7. Execution of the "paint" action using Particle Swarm Optimization methods, performed by the humanoid robot TEO in the CGDA framework.

represents a different evolutionary problem, although the cost function remains the same.

As a general conclusion extracted from this paper, the overall performance of both of these methods (AFFG-PSO and FI-PSO) is determined by the performance of the original PSO method. This is clearly shown in the experiments performed in this paper, comparing the results obtained with the two actions proposed. In the "wax" action, the performance of the three PSO methods with respect to the SST approach were better than in the "paint" action. However, it can be said that the modified PSO methods studied in this paper both achieve an important reduction of the number of evaluations for both actions. A new range of possibilities are now open, and choosing one or another of the methods will depend on the particular circumstances of the problem, and the needed tradeoff between error and number of evaluations.

VII. ACKNOWLEDGMENT

The research leading to these results has received funding from the RoboCity2030-III-CM project (Robtica aplicada a la mejora de la calidad de vida de los ciudadanos. fase III; S2013/MIT-2748), funded by Programas de Actividades I+D en la Comunidad de Madrid and cofunded by Structural Funds of the EU, and by a FPU grant funded by Miniesterio de Educación, Cultura y deporte.

REFERENCES

[1] S. Calinon and A. Billard, "Recognition and Reproduction of Gestures Using a Probabilistic Framework Combining PCA, ICA and HMM," in *Proceedings of the 22Nd International Conference on Machine Learning*, ser. ICML '05. New York, NY, USA: ACM, 2005, pp. 105–112.

[2] S. Calinon, F. Guenter, and A. Billard, "On Learning, Representing, and Generalizing a Task in a Humanoid Robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 286–298, Apr. 2007.

[3] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, Feb. 2013.

[4] S. Morante, J. G. Victores, A. Jardón, and C. Balaguer, "Action effect generalization, recognition and execution through continuous goal-directed actions," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1822–1827.

[5] S. Morante, J. G. Victores, and C. Balaguer, "Automatic demonstration and feature selection for robot learning," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, Nov. 2015, pp. 428–433.

[6] S. Morante, J. G. Victores, A. Jardn, and C. Balaguer, "Humanoid robot imitation through continuous goal-directed actions: an evolutionary approach," *Advanced Robotics*, vol. 29, no. 5, pp. 303–314, 2015.

[7] M. Müller, *Dynamic Time Warping*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 69–84.

[8] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing*, vol. 9, no. 1, pp. 3–12, 2003.

[9] L. Vincenzi and M. Savoia, "Improving the speed performance of an Evolutionary Algorithm by a second-order cost function approximation," *Proceedings of 2nd International Conference on Engineering Optimization*, Sep. 2010.

[10] R. E. Smith, B. A. Dike, and S. A. Stegmann, "Fitness inheritance in genetic algorithms," in *Proceedings of the 1995 ACM Symposium on Applied Computing*, ser. SAC '95. ACM, 1995, pp. 345–350.

[11] R. Barbour, D. Corne, and J. McCall, "Accelerated optimisation of chemotherapy dose schedules using fitness inheritance," in *IEEE Congress on Evolutionary Computation*, 2010, pp. 1–8.

[12] A. I. Esparcia-Alczar and J. Moravec, "Fitness approximation for bot evolution in genetic programming," *Soft Computing*, vol. 17, no. 8, pp. 1479–1487, Dec. 2012.

[13] I. Bertini, M. De Felice, A. Pannicelli, and S. Pizzuti, "Soft computing based optimization of combined cycled power plant start-up operation with fitness approximation methods," *Applied Soft Computing*, vol. 11, no. 6, pp. 4110–4116, Sep. 2011.

[14] J. Kennedy and R. Eberhart, "Particle swarm optimization," in , *IEEE International Conference on Neural Networks, 1995. Proceedings*, vol. 4, 1995, pp. 1942–1948 vol.4.

[15] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[16] M. Davarynejad, M. R. Akbarzadeh-T, and N. Pariz, "A novel general framework for evolutionary optimization: Adaptive fuzzy fitness granulation," in *2007 IEEE Congress on Evolutionary Computation*, 2007, pp. 951–956.

[17] M. R. Akbarzadeh-T, M. Davarynejad, and N. Pariz, "Adaptive fuzzy fitness granulation for evolutionary optimization," *International Journal of Approximate Reasoning*, vol. 49, no. 3, pp. 523–538, 2008.

[18] E. Ducheyne, B. D. Baets, and R. D. Wulf, "Is fitness inheritance useful for real-world applications?" in *Evolutionary Multi-Criterion Optimization*, ser. Lecture Notes in Computer Science, C. M. Fonseca, P. J. Fleming, E. Zitzler, L. Thiele, and K. Deb, Eds. Springer Berlin Heidelberg, 2003, no. 2632, pp. 31–42.

[19] M. Reyes-Sierra and C. A. C. Coello, "Fitness inheritance in multi-objective particle swarm optimization," in *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.*, 2005, pp. 116–123.

[20] ——, "A study of fitness inheritance and approximation techniques for multi-objective particle swarm optimization," in *2005 IEEE Congress on Evolutionary Computation*, vol. 1, 2005, pp. 65–72 Vol.1.

[21] G. Syswerda, "A study of reproduction in generational and steady-state genetic algorithms," in *Foundations of Genetic Algorithms*. Elsevier, 1991, vol. 1, pp. 94–101.

[22] R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, Carnegie Mellon University, Robotics Institute, August 2010.

[23] S. Martínez, C. A. Monje, A. Jardón, P. Pierro, C. Balaguer, and D. Munoz, "Teo: Full-size humanoid robot design powered by a fuel cell system," *Cybernetics and Systems*, vol. 43, no. 3, pp. 163–180, 2012.