2017 IEEE International Conference on
Autonomous Robot Systems and Competitions (ICARSC)
April 26-28, Coimbra, Portugal

Improving CGDA execution through Genetic Algorithms incorporating Spatial and Velocity constraints

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

***Abstract*—In the Continuous Goal Directed Actions (CGDA) framework, actions are modelled as time series which contain the variations of object and environment features. As robot joint trajectories are not explicitly encoded in CGDA, Evolutionary Algorithms (EA) are used for the execution of these actions. These computations usually require a large number of evaluations. As a consequence of this, these evaluations are performed in a simulated environment, and the computed trajectory is then transferred to the physical robot. In this paper, constraints are introduced in the CGDA framework, as a way to reduce the number of evaluations needed by the system to converge to the optimal robot joint trajectory. Specifically, spatial and velocity constraints are introduced in the framework. Their effects in two different CGDA commonly studied use cases (the "wax" and "paint" actions) are analyzed and compared. The experimental results obtained using these constraints are compared with those obtained with the Steady State Tournament (SST) algorithm used in the original proposal of CGDA. Conclusions extracted from this study depict a high reduction in the required number of evaluations when incorporating spatial constraints. Velocity constraints provide however less promising results, which will be discussed within the context of previous CGDA works.**

I. Introduction

Robot imitation is the general name given to frameworks where the user teaches new actions to the robot through a set of real world demonstrations. The robot is in charge of generating a generalized trajectory from this set of demonstrated actions. The selection of a model that is used as the internal representation of these generalized actions greatly defines the framework's characteristics and possibilities. In Programming by Demonstration (PbD), Hidden Markov Models [1] and Gaussian Mixture Models [2] have been used as a way to encode the robot joint and Cartesian space trajectories of these actions. Another approach is the use of Dynamic Motion Primitives (DMP), where actions are encoded using control laws that generate Cartesian space trajectories [3]. In Continuous Goal Directed Actions (CGDA), actions are modelled as time series which contain the variations of object and environment scalar features [4], rather than by the robot's or demonstrator's movements. As a consequence of this, actions are independent from robot or demonstrator's kinematics. The correspondence problem, which is one of the major problems presented in programming by demonstration, is not present in CGDA. In CGDA, an action regarding moving an object can be encoded similar to other approaches using the X,Y,Z coordinates of the object's centroid (three scalar features). However, other actions, such as painting a wall, can be represented using a single scalar feature: the percentage of the object or environment that has been painted. Features can be manually chosen, or automatically selected using the demonstration and feature selection algorithm presented in [5].

As the robot joint trajectory is not explicitly encoded in CGDA, Evolutionary Algorithms (EA) are used for the execution of these actions, which usually requires hundreds or even thousand of evaluations. The execution of all of these evaluations on a real physical robot is considered infeasible, as is an extremely time consuming task. These evaluations are, therefore, performed in a simulated environment. The computed trajectory is then transferred to the physical robot. A long-term goal of CGDA research is to be able to perform all the evaluations using the real robot. The advantage of this is to have an independence from the simulated environment, removing the problem of how to simulate unknown environments or scenarios. In this paper, spatial and velocity constraints are introduced in the EA, aiming at reducing the search space and thus the number of evaluations.

The CGDA framework is described in section II. In section III a study of the state of the art of constrained genetic algorithms is presented. The constraints used in this paper are described in section IV. The results of the experiments are presented in V, followed by the conclusions section.

II. The CGDA Framework

CGDA is a way to encode the effects of an action on the environment, based on action demonstrations perceived by a robot [4]. The CGDA framework is used for generalizing, recognizing and executing actions by their effects on the environment. A continuous analysis generates a trajectory in an n-dimensional feature space, where n is the number of tracked environment scalar features. Fig. 1 represents a simplified block diagram of the CGDA framework.

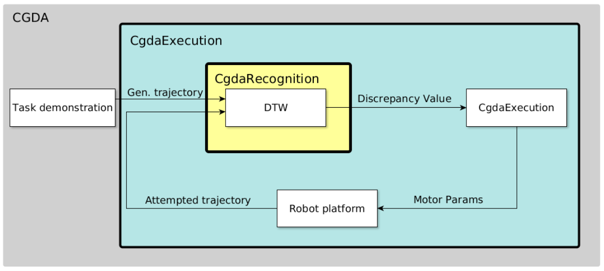

Fig. 1. Continuous Goal-Directed Actions (CGDA) framework diagram.

All of the authors are members of the Robotics Lab research group within the Department of Systems Engineering and Automation, Universidad Carlos III de Madrid (UC3M). raufernа@ing.uc3m.es

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 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].

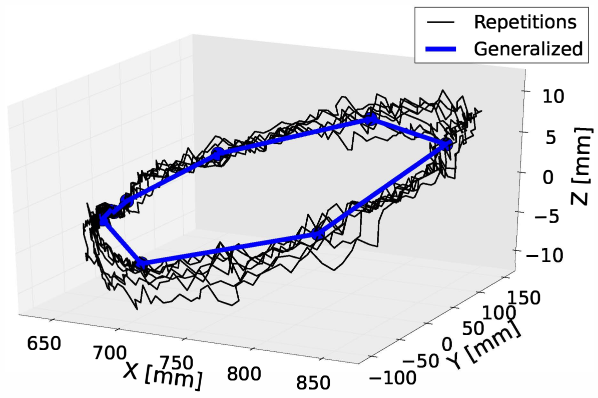

Fig. 2. Plot representing a three feature trajectory. Black lines are user action repetitions. The blue line is the generalization of all the repetitions.

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 most advanced evolutionary strategy within CGDA execution, Incrementally Evolved Trajectories (IET), was presented in [6]. The idea behind IET is to perform an individual evolution over each of the time intervals of the feature trajectory. For the evaluation of the trajectory at the interval time t, the t-1 first evolved trajectory intervals are executed before evaluating t, in order to reuse optimal results and additionally take time dependencies into account. IET is a strategy that is general enough to be applied using any EA, and was initially evaluated using the Steady State Tournament (SST). The following section analyses literature that incorporates constraints to further reduce the number of required evaluations in Genetic Algorithms (GA), the branch of EA which includes SST.

III. Constrained Genetic Algorithms

Genetic Algorithms (GA) [8] have been widely used as an optimization method for complex problems. One of their advantage is that they are simple to use and to implement. However, the initial proposition of GA was designed to work with non-constrained problems, whereas most of the areas where GA are used (engineering, robotics...) present problems that involve constraints. In these problems, GA have to deal with situations where some of the solutions, if not all, are not feasible. This is the reason why different methods to handle constraints in GA have been proposed. These methods can be divided, based on the way they deal with infeasible solutions, into the following groups [9]:

- **Rejecting strategy:** In this approach, infeasible solutions are rejected, and only the feasible ones are taken into account.
- **Repairing strategy:** The objective of this strategy is to preserve the feasibility of the solutions. Infeasible solutions are repaired and converted to feasible ones. The drawback of this strategy is that repairing solutions can sometimes be as complex as the optimization problem itself. Chootinan and Chen used the gradient information obtained from the constraint function for the repair process [10]. The gradient is used to redirect the infeasible solutions to the feasible space.
- **Modifying genetic operator strategy:** The genetic operators are modified in order to only obtain feasible solutions within the GA.
- **Penalty strategy:** In this strategy, the infeasible solutions are penalized with a penalty parameter. The constrained function is transformed into an unconstrained one, changing the infeasible solutions into penalized ones. This transformation can be done in two different ways:

$$eval(x) = \begin{cases} f(x), & if \quad x \in F \\ f(x) + p(x) & otherwise \end{cases} \tag{1}$$

$$eval(x) = \begin{cases} f(x), & if \quad x \in F \\ f(x)p(x) & otherwise \end{cases} \tag{2}$$

Where $f(x)$ represents the objective function, $p(x)$ is the penalty function, and F is the feasible space. From these two options, the additive one (1) is the most used.

Of all of the strategies presented here, the penalty strategy is the one with most related literature. It can be divided in different groups, depending on what strategy is used to obtain the penalty value [11]. In **Death penalties** strategies, the fitness penalty value is fixed to ∞ for all the infeasible solutions. **Static penalties** set this value as a finite constant. In **Dynamic penalties**, this value is a function of the generation number. A different approach is **Adaptive penalties**, where the penalty value changes according to the information obtained from the population in each iteration. An example of this approach is [12], where the penalty value is increased when all of the obtained solutions in the last iteration are infeasible. If they are all feasible, the penalty value is decreased. Finally, **Annealing Penalties** use Annealing Algorithms with the penalty value as a function of the temperature value [13].

Some other works have tried to introduce constraints in their objective functions as a way to reduce problem complexity. In [14], Yong and Sannomiya observed that in a flowshop problem there may be many solutions that are not valid combinations. The goal was to introduce constraints to remove those combinations, reducing the search space of the GA, and therefore the complexity of the problem.

IV. Reducing the search space in CGDA

With the goal of reducing the search space for CGDA execution, two different constraints are introduced in this paper. Since CGDA is a way to generalize, recognize, and execute actions, the objective is to choose a set of constraints that can be as ubiquitous as the rest of the CGDA framework. The constraints used in this paper are spatial and velocity constraints.

The spatial constraint is defined as a set of constraints that define a valid Cartesian space. As an initial approach, the limits imposed by the spatial constraint are given by the minimum bounding box that encloses the solution space, expanded by a dilatation value on all axes.

The velocity constraint is defined inside the GA definition. This constraint locally limits the joint velocity of each individual. This is achieved as follows: after the individual has moved in the search space, the new solution obtained with this individual is compared to the previous solution of the same individual in the joint space. If this difference is bigger than the threshold set by the constraint, then the new solution is set as not valid.

V. Experiments and Results

In the previous literature of CGDA [6], a Steady State Tournament (SST) was the method used for the evolution of trajectories. For coherence reasons, this method was also used in this paper. From previous works with Steady State Tournament (SST), a large number of evaluations was expected for convergence of the algorithms. For this reason, experiments were performed in a simulated environment using OpenRAVE [15]. The robotic platform used for the simulation was TEO[1], the humanoid robot from the Robotics Lab of Universidad Carlos III de Madrid [16]. For both experiments, three of the six Degrees of Freedom of the right arm of the humanoid were used, maintaining all other joints (including torso, legs and head) static.

The experiments consisted on the execution of the "wax" (also known as "clean") and the "paint" actions as proposed in [6], using the IET strategy. Each of the actions were executed using CGDA with the two different constraints proposed in this paper. Due to the amount of literature related, the Death penalty strategy was used to purge the solutions outside the constrained space. The goal of these experiments was to measure the impact of the proposed spatial and velocity constraints. These constraints have been adapted to the CGDA architecture, implemented and open-sourced [2].

A. *Wax*

The "wax" action is defined by the movement of a grasped object's centroid following one revolution of a circumference of 30 cm of diameter. Fig. 2 represents the generalized trajectory of this action. The three scalar features tracked by the CGDA system in this action are the Cartesian coordinates (X,Y,Z) of a grasped object's centroid. While this setting of the "wax" action makes it equivalent to solving the inverse kinematics of the manipulator, 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 of the "wax" experiments, 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. Joint parameters movements were restricted between -15 and 100 degrees. The individual mutation probability was set to 60%.

The spatial constraint was evaluated for the "wax" action for the following dilatation values (distance between the solution space and the valid space in terms of bounding boxes): 0.01 m, 0.05 m, 0.1 m, 0.2 m, 0.3 m and ∞. The number of evaluations required and DTW discrepancy (fitness) of these experiments are represented in Table I. These result are the average of running the "wax" action 50 times with the given parameters.

TABLE I
Experimental results for the "wax" action using a spatial Constraint

Dilatation [m]	0.01	0.05	0.1	0.2	0.3	∞
Evaluations	3212	3163	2993	4960	5722	9679
Discrepancy	465 (2)	503 (1)	471 (3)	312	331	274

The first three cases (0.01 m, 0.05 m, 0.1 m) are marked with a number in parenthesis () describing the number of the experiments that resulted in a DTW Discrepancy = ∞. These

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

solutions, that resulted outside of the valid space, were not computed within the average of the experiments.

In Fig. 3, the cumulative number of evaluations needed at each time interval (set at 1 s) is represented. In these experiments, the best solutions in terms of evaluation reduction were obtained using a dilatation $\leq$ 0.1 m. In the three experiments with a dilatation $\leq$ 0.1 m constrained space, the reduction was quite similar. The number of evaluations was reduced about 60% in these experiments. This means a reduction of over 6000 evaluations. However, the discrepancy of the generated trajectory obtained with dilatation $\leq$ 0.1 m also increased, obtaining the cited invalid solutions. In the 0.2 m case, the reduction was 49%. The discrepancy in this case increased 13%. In the 0.2 m case, the reduction was about 40%, while the discrepancy increased 17%.

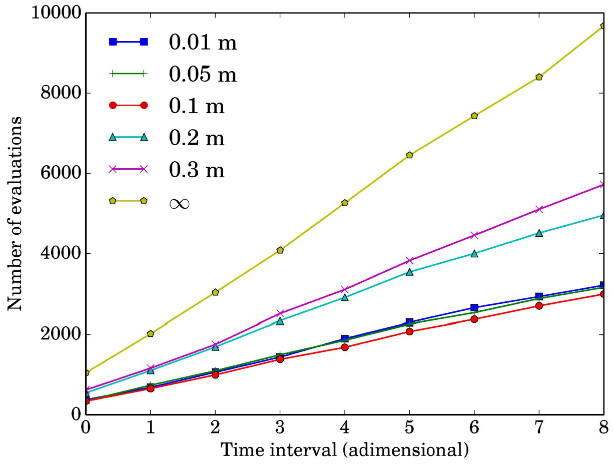

Fig. 3. “Wax” spatial experiment: Cumulative number of evaluations at each time interval (set at 1 s) for different dilatations (0.01 m, 0.05 m, 0.1 m, 0.2 m, 0.3 m and ∞).

The second set of experiments with the “wax” action was performed using the velocity constraint. Table II contains the results of these experiments using different values of the velocity constraint. These result are the average of running the “wax” action 50 times with the given parameters.

TABLE II
EXPERIMENTAL RESULTS FOR THE “WAX” ACTION USING A VELOCITY CONSTRAINT

Max. Velocity	5	10	20	60	80	∞
Evaluations	3591	4058	5723	6876	7349	9679
Discrepancy	540	483	331	346	330	274

Fig. 4 depicts the number of evaluations needed for each of the constraints values, in each time interval. The results of these experiments show how the number of evaluations is reduced when the velocity threshold is reduced. The maximum evaluation reduction is obtained in the 5 degree/iteration case, with a reduction in the number of evaluations of 63% with respect to the standard SST algorithm (which corresponds to ∞ velocity constraint). However, this reduction in the number of evaluation also implies an increment in the discrepancy of the obtained trajectory. The discrepancy obtained with the generated trajectory is a 50% lower using the standard SST algorithm than in the case of a constraint of 5 degree/iteration.

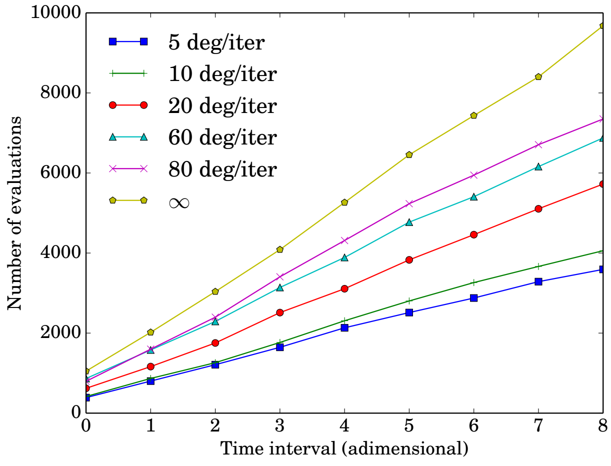

Fig. 4. “Wax” velocity experiment: Cumulative number of evaluations at each time interval (set at 1 s) for different Velocities (5, 10, 20, 60, 80 degree/iteration and ∞).

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. 5 shows an example of this action execution.

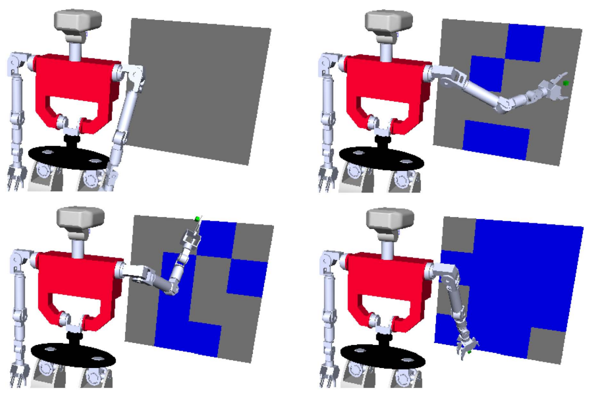

Fig. 5. Execution of the “paint” action using constraints, performed by the humanoid robot TEO in the CGDA framework.

For all of the “paint” experiments, 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. The individual mutation probability was set to 60%.

The spatial constraint was evaluated for the "paint" action for the following dilatation values: 0.01 m, 0.05 m, 0.1 m, 0.2 m, 0.3 m and ∞. The number of evaluations required and the DTW discrepancy (fitness) of these experiments are represented in Table III. These result are the average of running the "paint" action 100 times with the given parameters.

TABLE III
EXPERIMENTAL RESULTS FOR THE "PAINT" ACTION USING A SPATIAL CONSTRAINT

Dilatation [m]	0.01	0.05	0.1	0.2	0.3	∞
Evaluations	319	307	220	334	360	539
Discrepancy	193	17	5.7	7.3	6.3	7.3

Fig. 6 depicts the cumulative number of evaluations needed at each time interval (set at 1 s). In this case, the best performance is obtained with the 0.1 m constraint. With this constraint, the number of evaluations is reduced about 60% compared with the standard SST algorithm (which corresponds to ∞ spatial constraint), while having a lower discrepancy. As the constraint value moves away from the 0.1 m case, the number of required evaluations increases. Compared with the 0.1 m case, the 0.2 and 0.3 m cases need 151% and 164% more evaluations respectively. In the case of 0.01 m and 0.05 m, this increment is about 145% and 139% respectively. The discrepancy value increases in the two cases where the dilatation value is lower than 0.1 m. In the rest of the cases, fitness varies from 5.7 (0.1 m dilatation) to 7.3 (0.2 m dilatation). The case with the greatest discrepancy is 0.01 m, obtaining over a 190 discrepancy value in the generated trajectory.

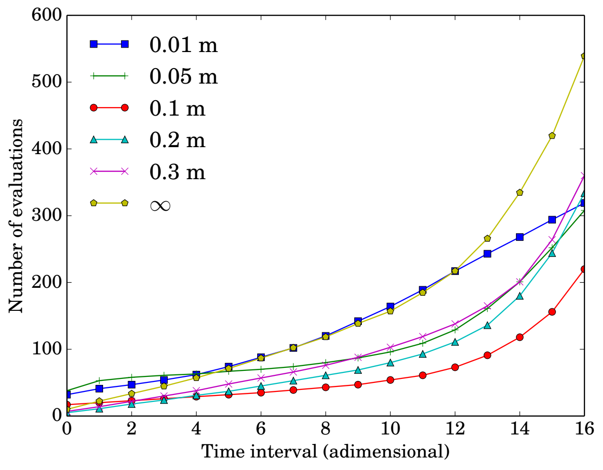

Fig. 6. "Paint" spatial experiment: Cumulative number of evaluations at each time interval (set at 1 s) for different dilatations (0.01 m, 0.05 m, 0.1 m, 0.2 m, 0.3 m and ∞).

Table IV depicts the results obtained using the velocity constraint with the "paint" action. In Fig. 7 the cumulative number of evaluations at each time interval using this constraint are represented.

TABLE IV
EXPERIMENTAL RESULTS FOR THE "PAINT" ACTION USING A VELOCITY CONSTRAINT

Max. Velocity	20	60	80	100	120	∞
Evaluations	543	557	572	527	529	539
Discrepancy	24.8	12.8	10.4	8.1	8	7.3

The greatest reduction in the number of evaluations using this constraint is with the 100 degree/iteration constraint, experiencing a evaluation reduction of about 2.2%. The 120 degree/iteration constraint obtains a reduction of 1.9%. In the rest of the cases, the number of evaluations are larger than in the case of using the standard SST algorithm (which corresponds to ∞ velocity constraint). The maximum number of evaluations is needed with a constraint of 80 degree/iteration, with an increment of 6% with respect to the standard SST algorithm. In terms of discrepancy, the solutions obtained with the constrained scenarios are worse than in the non-constrained one. The more constrained the system is, the larger the resulting discrepancy. The greatest discrepancy is obtained with a 20 degree/iteration constraint.

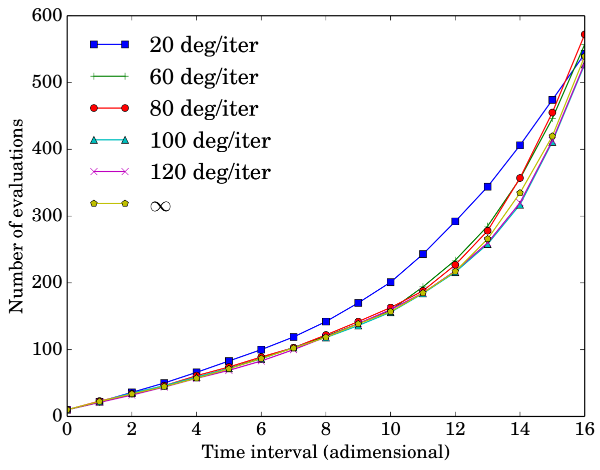

Fig. 7. "Paint" velocity experiment: Cumulative number of evaluations at each time interval (set at 1 s) for different Velocities (20, 60, 80, 100, 120 degree/iteration and ∞).

VI. CONCLUSIONS

In this paper, spatial and velocity constraints were evaluated within two CGDA commonly studied use cases ("wax" and "paint"). The spatial constraint has provided considerable improvements with respect to the standard SST algorithm.

For the "paint" action, the 0.1 m spatial constraint establishes a maximum in terms of performance. The results obtained with this constraint are not only better in evaluations, but also in minimizing the discrepancy with respect to the standard SST algorithm. The number of evaluations is reduced by a 60%, a reduction of over 300 evaluations that would have to be performed by the robot. For the "wax" action, the best case greatly reduces evaluations (69% reduction of evaluations at 0.1 m dilatation) compared with the standard SST algorithm, at the cost of a discrepancy increase which may or not be relevant. It is also interesting to note how the results obtained with the 0.01 m, 0.05 m and 0.1 m cases are similar. This is possibly due to the fact that the constrained region is a box, while the solution space is something similar to a toroid. This means that the relative size of the valid space between the solution space and the constrained space does not suffer any significant reduction once reaching a dilatation reduction value. One of the best results was obtained with the 0.2 m case. In this scenario, the "wax" action experienced a reduction in the number of evaluations over 50%, while having an increment in the discrepancy of about 14%.

Regarding the velocity constraint results, the maximum reduction was obtained with the 5 degree/iteration constraint. In this scenario, the number of evaluations was reduced by a 63% with respect to the standard SST algorithm. However, this came at the cost of also increasing the discrepancy value obtained with respect to the standard SST algorithm. In the case of the "paint" action, introducing this constraint does not significantly reduce the number of evaluations in any of the cases. However, the discrepancy value still increased when lowering the threshold. These not so promising results could have an explanation in past works related to CGDA. In [17], the authors studied the performance of a Sequential Incremental Combinatorial Search of motor primitives, as a way to execute actions in the CGDA framework. For the experiments, the authors used different sets of primitives. Some sets were highly constrained, resulting in primitives that were limited to a maximum length. Other sets allowed more variations between the primitives. The results of the experiments presented a better performance using the sets that allowed more variations, rather than the more constrained sets. The authors concluded this was due to the richer possibilities offered by less constrained sets, such as coarse and fine adjustment, which allowed the system to have a better performance on CGDA execution. With the velocity constraint presented in this paper, we are producing a similar effect to the case of the set of constrained primitives. The constraints introduced in the individual velocity imply a reduction in the variation of solution possibilities, and therefore a decay in performance.

The variations of the effects of both constraints between the two presented actions "wax" and "paint" show the importance of correctly selecting the correct constraints for each action, in order to effectively reduce the search space. Some knowledge about the action and the environment is therefore required. However, future works aim at obtaining the constraints directly from the user demonstrations, through statistical learning techniques. The introduction of constraints in GA and in EA in general opens a new window of possibilities towards performing CGDA execution on real robots.

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. Jardon, 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] J. H. Holland, *Adaptation in natural and artificial systems*, second edition (1992). (first edition, university of michigan press, 1975). ed. Cambridge: MIT Press, 1975.

[9] K. Deep and Dipti, "A self-organizing migrating genetic algorithm for constrained optimization," *Applied Mathematics and Computation*, vol. 198, no. 1, pp. 237–250, Apr. 2008.

[10] P. Chootinan and A. Chen, "Constraint handling in genetic algorithms using a gradient-based repair method," *Computers & Operations Research*, vol. 33, no. 8, pp. 2263–2281, Aug. 2006.

[11] O. Yeniay, "Penalty Function Methods for Constrained Optimization with Genetic Algorithms," *Mathematical and Computational Applications*, vol. 10, no. 1, pp. 45–56, Apr. 2005.

[12] A. Ben Hadj-Alouane and J. C. Bean, "A Genetic Algorithm for the Multiple-Choice Integer Program," *Operations Research*, vol. 45, no. 1, pp. 92–101, 1997.

[13] Z. Michalewicz and N. Attia, "Evolutionary Optimization of Constrained Problems," in *Proceedings of the 3rd Annual Conference on Evolutionary Programming*. River Edge, NJ: World Scientific Publishing, 1994, pp. 98–108.

[14] Z. Yong and N. Sannomiya, "An Improvement of Genetic Algorithms by Search Space Reductions in Solving Large-Scale Flowshop Problems," in *Transactions of The Institute of Electrical Engineers of Japan*, 2001, vol. 121, pp. 1010–1015.

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

[16] 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.

[17] S. Morante, J. G. Victores, A. Jardón, and C. Balaguer, "On using guided motor primitives to execute Continuous Goal-Directed Actions," in *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, Aug. 2014, pp. 613–618.