The pole balancing problem from the viewpoint of system flexibility

Léo Françoso Dal Piccol Sotto
leo.francoso.dal.piccol.sotto@scai-extern.fraunhofer.de
Fraunhofer SCAI
Sankt Augustin, Germany

Sebastian Mayer
sebastian.mayer@scai.fraunhofer.de
Fraunhofer SCAI
Sankt Augustin, Germany

Jochen Garcke*
garcke@ins.uni-bonn.de
Institut für Numerische Simulation
Universität Bonn
Bonn, Germany

ABSTRACT

Whereas evolutionary computation usually solves problems from scratch, organisms evolve under changing environments and possess flexibility, adapting from being good at one task to being good at a related task. There is abundant evidence that there are general properties that promote flexibility in nature, such as hierarchy, modularity, exploratory behavior, and degeneracys or neutrality.

Our interest is to understand if such properties can also be identified for non-biological systems. We thus study if a controller evolved by a genetic algorithm for one pole balancing task can be adapted to a different pole balancing task, and if this saves training time compared to evolving a new controller from scratch. Moreover, we investigate how diversity and degeneracy in the controllers population affect adaption efficiency by promoting high quality solutions that are both structurally and behaviorally diverse, concluding that it can potentially decrease the adaption cost.

CCS CONCEPTS

• Theory of computation → Bio-inspired optimization; **• Computing methodologies → Genetic algorithms**; *Control methods*; *Modeling and simulation.*

KEYWORDS

Flexibility, Adaptability, Pole Balancing, Exploration, Degeneracy, Neutrality

ACM Reference Format:
Léo Françoso Dal Piccol Sotto, Sebastian Mayer, and Jochen Garcke. 2022. The pole balancing problem from the viewpoint of system flexibility. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3520304.3529040

1 INTRODUCTION

We call a learning system flexible if it easily adapts from being good at one task to being good at a related task and if it can cope with a diversity of related tasks. Approaches that increase flexibility have been intensively studied in machine learning (transfer learning [16], learning to learn [13]), but are less established in evolutionary computation (evolutionary dynamic optimization [17], evolutionary transfer optimization [12]). The theory of facilitated variation [5] shows that high evolvability is the result of certain universal design features of biological systems such as hierarchical organization, modularity, weak regulatory linkage, and exploratory behavior, that allow the organism to render small, random genetic changes into complex, useful adaptions in its phenotpye. Work following this line of reasoning applied in technical systems shows that modularity and hierarchy can arise spontaneously under dynamically-changing tasks and speed up adaption for new tasks [8]. Quality-diversity optimization was used to let robots quickly adapt to a diversity of task variations [3].

Further features contributing to evolvability are degeneracy and neutral mutations [4, 14]. Degeneracy refers to systems that are functionally equivalent in one environment but functionally different in other environments, or system configurations with a similar behavior but a different structure, and relates to the notion of neutrality in evolutionary computation [10]. Degenerate systems are more robust, as mutations in a neutral space do not affect the system's function, and more evolvable, as systems that perform the same function can be located in different regions of a neutral space and will thus respond differently to mutations [14].

The pole balancing problem is a well known control theory and reinforcement learning benchmark where an agent has to learn to balance a cart with a pole attached to it without letting the cart or the poles move out of a pre-specified range [2]. The system is composed of a cart of mass m_c and N poles of masses $m_{p1}, \dots, m_{pN}$, and lengths $l_{p1}, \dots, l_{pN}$. At each time step t, a controller receives as input the position of the cart in a track $x(t)$, the velocity of the cart $\dot{x}(t)$, the angular positions of the poles $\theta_{p1}(t), \dots, \theta_{pN}(t)$, and the angular velocities of the poles $\dot{\theta}_{p1}(t), \dots, \dot{\theta}_{pN}(t)$, and outputs a horizontal force $F(t)$ to be applied to the cart. The simulation is considered successful if the agent can keep the cart and the pole within limits for 100,000 time steps.

Motivated by the line of work from biology on flexible systems, we study the adaption of a controller by a genetic algorithm under varying instances of the pole balancing problem [2], and analyze how adapting a previously evolved solution to solve a new related task decreases the learning cost, how we can measure the flexibility of a system, and, inspired by exploration and degeneracy in biological systems flexibility, if diversity promotion combined with exploration of the neutral space provide a decrease in adaption cost.

*also with Fraunhofer SCAI

2 METHODOLOGY

2.1 Measuring system flexibility

As our interest is to study the flexibility of a learning system and we do not aim for improving on existing results, we consider the basic pole balancing problem with one pole and choose n different values for the pole length, obtaining a *task context* $\mathcal{T} = \{T_1, \dots, T_n\}$ of n related learning tasks. Now we wonder if an agent that has already learned a control law for one task, say T_i, can exploit this to learn another task, say T_j, more quickly. As learning system we consider the tuple $L = (F_w, A)$, where F_w is the controller and A some learning or adaption algorithm. Given a task T and a learning system L, we denote the cost of L to learn the task from scratch by $c_0(L, T)$. Now suppose we are given two tasks T_1 and T_2 and the learning system has already learned the source task T_1. Then, we denote the cost to adapt the solution found for T_1 to the target task T_2 by $c_{\text{ada}}(L, T_1, T_2)$. We measure the flexibility of the learning system in the given context by the cost it takes the system to adapt solutions of source tasks to solutions of target tasks, either by the **worst-case adaption cost**

$$c_{\text{ada}}^{\text{wor}}(L, \mathcal{T}) = \max_{T_1 \neq T_2} c_{\text{ada}}(L, T_1, T_2). \tag{1}$$

or by the **average-case adaption cost** given by

$$c_{\text{ada}}^{\text{avg}}(L, \mathcal{T}) = \frac{1}{n(n-1)} \sum_{T_1 \neq T_2} c_{\text{ada}}(L, T_1, T_2). \tag{2}$$

Note that the above definitions are not specific to evolutionary algorithms but can be used for any learning systems, e.g., artificial neural networks. We only have to provide measures for c_0 and c_{ada} that are suitable for the considered learning system.

2.2 Adaption via Genetic Algorithm

There are many approaches that were successfully used in the literature to solve the pole balancing problem, including, for example, the use of evolutionary algorithms and the evolution of neural networks [1]. It is known, however, see [2], that the problem setup with only one pole can be solved by a linear neuron of the form

$$F_w(t) = F_m * sgn\big(w_1 x(t) + w_2 \dot{x}(t) + w_3 \theta(t) + w_4 \dot{\theta}(t)\big), \tag{3}$$

where $w = (w_1, w_2, w_3, w_4)$ is the weight vector to be optimized, F_m is a constant force, and *sgn* is the sign function that outputs either -1 or $+1$. We use a Genetic Algorithm (GA) [15] to optimize the weights in a population of linear neurons of this form, where a candidate solution is a vector of four real-valued weights, and the fitness value is the number of time steps for which the resulting controller can keep the cartpole system within limits. Single-point crossover can be used as usual, but for the mutation we perform a random perturbation on the weights given a mutation rate.

Suppose we have tasks T_1 and T_2, which correspond to different pole lengths and with respective solutions S_1 and S_2. First, we run the GA as normal on task T_1 from scratch, using a randomly initialized population. However, differently from the standard GA, that would halt after finding the optimal solution, in our adaptive scheme we keep searching after that until we complete the maximum number of generations. This way, we end up with a final population of high quality solutions. We then load this population as the initial population for the GA run for task T_2. We use this simple adaptive scheme as our flexibility baseline. For more complex problems and larger populations, however, one may want to store just a number of high quality solutions and/or initialize the population only with a fraction of previous solutions. Moreover, one can also choose to stop the search before the maximum number of generations for longer runs. As learning and adaption costs we employ the *minimal Computational Effort* (CE), which is an approximation for how many individual evaluations are needed for finding the optimum with 99% probability [6]. The learning cost $c_0(L, T)$ to solve task T from scratch is the CE value over a number of runs, and the adaption cost $c_{\text{ada}}(L, T_1, T_2)$ for adapting a solution from task T_1 to task T_2 is the CE value obtained when employing the adaptive scheme described.

Based on the discussion on exploration and degeneracy conferring flexibility to biological systems in section 1, besides the flexibility baseline described above, we study the element of exploration by promoting diversity in the GA populations, and degeneracy by promoting a population of solutions that are at the same time optimal or near-optimal but diverse in respect to its structure or behavior. Here, **structural diversity** refers to solutions with different weights, and **behavioral diversity** to the actions a controller takes within a simulation - the time series describing, at each time step, if a force is applied to the right or to the left. For structural diversity, we measure the Euclidean distance between the chromosomes and define the **structural diversity score** of an individual by its mean Euclidean distance to all other individuals in the population. In large populations, one may want to sample only a fraction of it. For behavioral diversity, we use the final state of a simulation, which corresponds to the vector $[x, \dot{x}, \theta, \dot{\theta}]$ at the end of a simulation. Then we again define the **behavioral diversity score** by calculating the mean Euclidean distance to all other individuals in the population. As we use a fixed initial state for the simulation, the final state should not vary for the same controller and task.

To promote diverse individuals, we modify the tournament procedure in order to generate a non-parametric selection pressure towards diverse solutions, inspired on a technique called Proportional Tournament, previously used for promoting diversity and fighting bloat [7], that selects individuals during tournament either using fitness or some other metric, here structural or behavioral diversity, according to some probability. Instead of using a probability, we select based on diversity when the fitness of individuals sampled by the tournament is the same. Thus, tournament of size T samples T individuals from the population and returns the most fit or, if all have the same fitness, the most diverse according to our diversity score. The diversity promotion strategy is used when evolving from scratch. When adapting solutions, standard tournament is used.

3 EXPERIMENTS

3.1 Experimental setup

We compare the performances of evolving a population from scratch for each task with adapting a solution from a previous task both with and without diversity promotion. We implemented a GA using

Python and for the pole balancing simulation we adapted the code from the library Gym[1]. Gravity is $-9.8 m/s^2$, cart friction is $5 * 10^{-4}$, and pole friction is $2 * 10^{-6}$ (equations from [1]). The full pole length varies from 0.2 to 2.0 with a step of 0.2, resulting in 10 tasks. The initial state is fixed as $x = 0$, $\dot{x} = 0$, $\theta = 0$, $\dot{\theta} = 0$. Remaining parameters are the standard from Gym.

The GA population is initialized with weights drawn from a uniform distribution in the range $[-0.5, 0.5]$. We use a population size of 10 and run 10 generations, as it was enough to solve the problem, and employ tournament selection with size 2, the smallest possible given the population size. Standard one-point crossover is applied to parents with a probability of 90% (standard value with no influence on preliminary results). Mutation consists of adding to a weight a value drawn from a uniform distribution in the range $[-0.1, 0.1]$, and is applied to each gene of the offspring with a probability of 25%, so that in the average one gene will be mutated. We use elitism of size 1.

3.2 Results and discussion

We show in Table 1-(top) our baseline adaption results. As the CE metric for learning and adaption costs uses the number of generations, the minimum possible value here is 10 for one generation given our population size of 10. Adapting a previous population provides a good advantage that grows with how similar tasks are. Finding solutions for shorter pole lengths using solutions from longer pole lengths is more difficult, and the same is true for finding solutions for longer pole lengths using solutions from shorter pole lengths. As finding solutions for shorter pole lengths is more difficult in general, this probably means that these solutions are in more distant regions in the search space. Thus, moving from the region where solutions for longer pole lengths are to the region where solutions for shorter pole lengths are, and vice versa, is more difficult than starting with a random population. We highlight these pairs of tasks in Table 1 as our area of interest. In the average-case, adapting a solution makes the learning system more flexible in the given task context.

We summarize three points that we do not show here due to space constraints: 1) We plotted all optimal solutions for two representative tasks in the weights range from -5 to 5 by a step of 0.5, and found that there is a very high number of different weight configurations that are optimal (highly degenerate search space), which motivates the study of promotion of structural diversity for improving adaption performance. 2) We plotted the actions of the controllers above and found that all controllers have a very similar behavior pattern. However, each sequence is unique, and behaviors differ on average by 49,000 time steps, which leaves us the possibility to explore the potential of promoting behavioral diversity. 3) When promoting diversity, we report that both structural and behavioral diversity across generations is higher in comparison to the baseline, although the increase is not dramatic.

When promoting structural diversity (Table 1-(middle)), there is considerable improvement in adaption performance in comparison to the baseline in the first area of interest (first two rows). In total, structural diversity promotion improved adaption results for 37 cases against 9 cases where it was worse than the baseline.

[1]https://gym.openai.com/

When promoting behavioral diversity (Table 1-(bottom)), adaption results for both the first and second areas of interest (adaption from shorter to longer poles) are mixed. In general, behavioral diversity promotion led to more worse than better results, with 17 cases in total where it improved adaption results compared to the baseline against 28 where the result was worse. Based on that, making use of degeneracy in optimal solutions to spread individuals in the final population of source tasks in the search and behavior spaces is potentially beneficial for improving the flexibility of a learning system. As different pairs of source and target tasks have different search spaces, it is possible that in some cases the region where solutions were initially concentrated was better for adaption than spreading them. It is possible that a procedure that better covers the search and/or behavior space with high quality solutions is able to overcome this difficulty. One specific concern with behavioral diversity is that it is not guaranteed that two solutions with different behaviors in a source task will also have a comparably different behavior in the target task, as the sequence of actions is first known during simulation. This could potentially be alleviated by quality-diversity exploration using behavioral descriptors as, e.g., in [3].

4 CONCLUSIONS AND FUTURE WORK

After systematically analyzing the pole balancing problem from a system flexibility viewpoint, we found that adapting a previous population helps to reduce training time for related learning tasks in many situations, with an improvement on average proportional to how similar tasks are. Moreover, diversity promotion making use of degeneracy on the structural and behavioral levels is potentially useful for improving system flexibility, but a better approach is necessary for a more uniform improvement. We believe the questions we addressed regarding flexibility, both in biological and learning systems, can be interpreted on a higher level and transferred to a broader range of problems. As future work, one can consider more complex versions of the pole balancing benchmark as well as benchmarks from other domains or real-world applications, further develop the flexibility notions in a mathematical formal way, and explore other elements of flexibility, also in other methods and domains - graph-based Genetic Programming, for example, has innate properties of modularity and neutrality/degeneracy [9], from which Cartesian Genetic Programming has already been used to evolve Neural Networks [11].

ACKNOWLEDGMENTS

This work was carried out during the tenure of an ERCIM 'Alain Bensoussan' Fellowship Programme.

REFERENCES

[1] Timothy Atkinson, Detlef Plump, and Susan Stepney. 2020. Horizontal gene transfer for recombining graphs. *Genetic Programming and Evolvable Machines* 21 (2020), 321–347.
[2] J. Scott Brownlee. 2005. The pole balancing problem: a benchmark control theory problem.
[3] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. 2015. Robots that can adapt like animals. *Nature* 521, 7553 (2015), 503–507.
[4] Gerald M Edelman and Joseph A Gally. 2001. Degeneracy and complexity in biological systems. *Proceedings of the National Academy of Sciences* 98, 24 (2001), 13763–13768.

Table 1: Cost of learning from scratch and adapting from a previous task (CE values) for the three studied approaches and all pairs of tasks. Values are calculated over 100 runs for each task or pair of tasks. The value in (i, j) means adaption from T_j to T_i. We highlight the area of interest for improving adaption performance. Better/worse in the lower tables stand for the number of cases where diversity promotion resulted in better adaption performance compared to the baseline for each line.

Baseline - Without diversity promotion

		Learning cost	**Adaption cost c_{ada} from task $T_i = polelength$**									
Task	**Polelength**	**from scratch c_0**	**0.2**	**0.4**	**0.6**	**0.8**	**1.0**	**1.2**	**1.4**	**1.6**	**1.8**	**2.0**
T_1	**0.2**	440	10	40	100	120	190	**270**	**240**	**400**	**560**	**370**
T_2	**0.4**	240	20	10	20	40	40	**60**	**60**	**80**	**120**	**120**
T_3	**0.6**	190	30	10	10	20	20	40	40	40	50	60
T_4	**0.8**	100	30	10	10	10	20	20	30	40	30	50
T_5	**1.0**	100	60	20	10	10	10	20	20	20	10	30
T_6	**1.2**	120	50	30	20	10	10	10	10	20	10	20
T_7	**1.4**	100	**60**	**30**	20	20	10	20	10	10	10	20
T_8	**1.6**	100	**70**	**30**	20	20	10	10	10	10	10	20
T_9	**1.8**	70	**80**	**40**	30	20	20	20	10	10	10	10
T_{10}	**2.0**	70	**70**	**40**	30	30	20	20	10	10	10	10
Worst-case		$c_0^{\text{wor}}(L, \mathcal{T}) = 440$	$c_{\text{ada}}^{\text{wor}}(L, \mathcal{T}) = 560$									
Average-case		$c_0^{\text{avg}}(L, \mathcal{T}) = 153$	$c_{\text{ada}}^{\text{avg}}(L, \mathcal{T}) = 47.60$									

Structural diversity promotion

		Learning cost	**Adaption cost c_{ada} from task $T_i = polelength$**										
Task	**Polelength**	**from scratch c_0**	**0.2**	**0.4**	**0.6**	**0.8**	**1.0**	**1.2**	**1.4**	**1.6**	**1.8**	**2.0**	**Better/Worse**
T_1	**0.2**	380	10	50	60	70	140	**160**	**150**	**150**	**210**	**180**	8/1
T_2	**0.4**	240	20	10	20	40	40	**60**	**70**	**50**	**80**	**80**	3/1
T_3	**0.6**	120	20	10	10	20	30	20	30	30	40	40	6/1
T_4	**0.8**	120	30	20	10	10	20	10	20	20	30	30	4/1
T_5	**1.0**	110	40	20	10	10	10	10	20	20	20	20	3/1
T_6	**1.2**	80	60	20	20	10	10	10	20	10	10	20	2/2
T_7	**1.4**	70	**50**	**30**	10	20	10	10	10	10	10	10	4/0
T_8	**1.6**	60	**70**	**30**	20	20	10	10	10	10	10	20	0/0
T_9	**1.8**	60	**70**	**30**	20	20	20	10	10	10	10	20	4/1
T_{10}	**2.0**	80	**80**	**40**	20	20	20	10	10	10	10	10	3/1
Total													37/9
Worst-case		$c_0^{\text{wor}}(L, \mathcal{T}) = 380$	$c_{\text{ada}}^{\text{wor}}(L, \mathcal{T}) = 210$										
Average-case		$c_0^{\text{avg}}(L, \mathcal{T}) = 132$	$c_{\text{ada}}^{\text{avg}}(L, \mathcal{T}) = 33$										

Behavioral diversity promotion

		Learning cost	**Adaption cost c_{ada} from task $T_i = polelength$**										
Task	**Polelength**	**from scratch c_0**	**0.2**	**0.4**	**0.6**	**0.8**	**1.0**	**1.2**	**1.4**	**1.6**	**1.8**	**2.0**	**Better/Worse**
T_1	**0.2**	450	10	40	90	150	220	**210**	**220**	**290**	**290**	**900**	5/3
T_2	**0.4**	180	20	10	30	40	50	**90**	**100**	**130**	**100**	**200**	1/6
T_3	**0.6**	150	20	10	10	10	30	40	40	60	60	60	2/3
T_4	**0.8**	90	30	20	10	10	10	20	30	40	30	50	1/1
T_5	**1.0**	100	40	30	20	10	10	20	10	30	20	30	2/4
T_6	**1.2**	90	40	20	20	20	10	10	20	20	20	20	2/3
T_7	**1.4**	90	**60**	**50**	30	20	10	10	10	10	20	20	1/3
T_8	**1.6**	80	**70**	**40**	30	20	10	10	10	10	10	10	1/2
T_9	**1.8**	100	**70**	**50**	30	20	20	10	10	10	10	10	2/1
T_{10}	**2.0**	70	**90**	**50**	30	30	20	20	10	10	10	10	0/2
Total													17/28
Worst-case		$c_0^{\text{wor}}(L, \mathcal{T}) = 450$	$c_{\text{ada}}^{\text{wor}}(L, \mathcal{T}) = 900$										
Average-case		$c_0^{\text{avg}}(L, \mathcal{T}) = 140$	$c_{\text{ada}}^{\text{avg}}(L, \mathcal{T}) = 51.80$										

[5] Marc W Kirschner and John C Gerhart. 2008. *The plausibility of life.* Yale University Press.

[6] John R. Koza. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, USA.

[7] Sean Luke and Liviu Panait. 2002. Fighting Bloat With Nonparametric Parsimony Pressure, Vol. 2439.

[8] Merav Parter, Nadav Kashtan, and Uri Alon. 2008. Facilitated variation: how evolution learns from past environments to generalize to new environments. *PLoS computational biology* 4, 11 (2008), 1–15.

[9] Léo Françoso Dal Piccol Sotto, Paul Kaufmann, Timothy Atkinson, Roman Kalkreuth, and Márcio P. Basgalupp. 2021. Graph representations in genetic programming. *Genetic Programming and Evolvable Machines* 22 (2021), 607–636.

[10] Léo Françoso Dal Piccol Sotto, Franz Rothlauf, Vinícius Veloso de Melo, and Márcio P. Basgalupp. 2022. An Analysis of the Influence of Noneffective Instructions in Linear Genetic Programming. *Evolutionary Computation* 30, 1 (2022), 51–74.

[11] Masanori Suganuma, Masayuki Kobayashi, Shinichi Shirakawa, and Tomoharu Nagao. 2020. Evolution of Deep Convolutional Neural Networks Using Cartesian Genetic Programming. *Evolutionary Computation* 28, 1 (2020), 141–163.

[12] Kay Chen Tan, Liang Feng, and Min Jiang. 2021. Evolutionary Transfer Optimization - A New Frontier in Evolutionary Computation Research. *IEEE Computational Intelligence Magazine* 16, 1 (2021), 22–33.

[13] Sebastian Thrun and Lorien Pratt. 1998. *Learning to learn.* Springer Science & Business Media.

[14] James M Whitacre. 2010. Degeneracy: a link between evolvability, robustness and complexity in biological systems. *Theoretical Biology and Medical Modelling* 7, 1 (2010), 1–17.

[15] Darrell Whitley. 1998. A Genetic Algorithm Tutorial. *Statistics and Computing* 4 (1998).

[16] Qiang Yang, Yu Zhang, Wenyuan Dai, and Sinno Jialin Pan. 2020. *Transfer learning.* Cambridge University Press.

[17] Shengxiang Yang. 2013. Evolutionary computation for dynamic optimization problems. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation.* 667–682.