FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Michal Pospěch

Thesis title is N/A

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the bachelor thesis: Mgr. Roman Neruda, CSc.
Study programme: Computer science
Study branch: General computer science

Prague 2021

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In . . . . date . . . .

Author's signature

Dedication. It is nice to say thanks to supervisors, friends, family, book authors and food providers.

Title: Thesis title is N/A

Author: Michal Pospěch

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Mgr. Roman Neruda, CSc., Department of Theoretical Computer Science and Mathematical Logic

Abstract: Abstracts are an abstract form of art. Use the most precise, shortest sentences that state what problem the thesis addresses, how it is approached, pinpoint the exact result achieved, and describe the applications and significance of the results. Highlight anything novel that was discovered or improved by the thesis. Maximum length is 200 words, but try to fit into 120. Abstracts are often used for deciding if a reviewer will be suitable for the thesis; a well-written abstract thus increases the probability of getting a reviewer who will like the thesis.

Keywords: key words

Contents

Introduction

Chapter 1

Important first chapter

Chapter 2

Theory

2.1 Reinforcement learning

- problem description (state space, action space, reward...)
- various methods
  - value function
  - criterion of optimality
  - direct policy search (and various methods)

2.2 Evolutionary algorithms

Evolutionary algorithms (EA) are a type of optimisation metaheuristics inspired by the process of bilogical evolution. At first a number of possible solutions to the problem at hand is generated (*population*) and each solution (*individual*) is encoded (via a domain-specific encoding) and evaluated giving us the value of its *fitness*. Fitness is a function describing how good that particular individual is and it is everything that is needed for creation of Then a new population is created using a *crossover* (combination) of 1 or more individuals which are selected using the operator of *parental selection*. Each of the newly created individuals has a chance to be mutated via the *mutation* operator. Finally a new population is selected from *offsprings* and possibly the parents based of fitness and enters the next iteration of the EA and the following generation is chosen using *environmental selection* operator. The algorithm repeats until the stop condition is met, usually a set number of iterations or small improvement of fitness between 2 generations.

There are many variands of EAs such as genetic algorithms (most common), genetic programming, evolutionary programming, neuroevolution or evolutionary strategies that are further described in following chapter. [1] [2]

Algorithm 1 Evolutionary algorithm

initialize population P^0 with n individuals
set $t = 0$
repeat
    $Q^t = \{\}$
    for $i \in \{1 \dots m\}$ **do**
        $p_1, \dots, p_\rho = ParentalSelection(P^t)$
        $q = Crossover(p_1, \dots, p_\rho)$
        $q = Mutation(q)$ with chance p
        $Q^t = q \cup Q^t$
    end for
    $P^{t+1} = EnvironmentalSelection(Q^t \cup P^t)$
    increment t
until stop criterion fullfilled

2.3 Evolutionary strategies

Evolutionary strategies (ES) are a type of optimisation metaheuristic which further specialises EA and restricts their level of freedom. The selection for crossover is unbiased, mutation is parametrised and thus controllable, individuals which should be put to next generation are chosen ordinally based on fitness and individuals contain not only the problem solution but also control parameters.

More formally ES $(\mu/\rho, \kappa, \lambda)$ has μ individuals in each generation, which produces λ offsprings, each created by crossover of ρ individuals and each individual is able to survive for up to κ generations as described in algorithm 2. This notation further generalizes the old (μ, λ) and $(\mu + \lambda)$ notations, where the "," notation means $\kappa = 1$ and "+" notation $\kappa = \infty$.

To design an ES one must first select an appropriate representation for an individual and the most natural one is prefered in most cases, if all parameters are of one type (e.g. a real number) a simple vector will suffice, if the types are mixed, a tuple of vectors is required. This however causes an increased complexity of the variation operator.

As for design of the variation operator there are some guidelines that should be followed when designing it.

Algorithm 2 $(\mu/\rho, \kappa, \lambda)$-ES

```
initialize population P^0 with μ individuals
set age for each p ∈ P^0 to 1
set t = 0
repeat
    Q^t = {}
    for i ∈ {1 ... λ} do
        select ρ parents p_1, ..., p_ρ ∈ P^t uniformly at random
        q = variation(p_1, ..., p_ρ) with age 0
        Q^t = q ∪ Q^t
    end for
    P^{t+1} = select μ best (wrt. fitness) individuals from Q^t ∪ {p ∈ P^t : age(p) < κ}
    increment age by 1 for each p ∈ P^{t+1}
    increment t
until stop criterion fullfilled
```

Reachability every solution should be reachable from any other solution in a finite number of applications of the variation operator with probability $p > 0$

Unbiasedness the operator should not favour any particular subset of solution unless provided with information about problem at hand

Control the operator should be parametrised in such way that the size of the distribution can be controlled (practice had shown that decreasing it as the optimal solution is being approached is necessary)

kovariance

[3] [1]

2.3.1 CMA-ES

TODO [4]

2.4 Evolutionary strategies as replacement for reinfocement learning

uvod

2.4.1 OpenAI Evolutionary Strategy

Compared to reinfocement learning using evolutionary strategies have the advantage of not needing a gradient of the policy performance. Also as the state transition function is not known the gradient can't be computed using backpropagation-like algorithm. Thus some noise needs to be added to make the problem smooth and the gradient to be estimable. Here is where reinfocement learning and evolutionary strategies differ, reinfocement learning adds noise in the action space (actions are chosen from a distribution) while evolutionary strategies add noise in the parameter space (parameters perturbed while actions are deterministic).

dimensionality, what and when better

Not requiring backpropagation has several advantages over other RL methods. First the amount of computation necessary for one episode of ES is much lower (about one third, potentially even less for memory usage). Not calculating gradient using analytic methods also protects these methods from suffering from *exploding gradient* which is a common issue with recurrent neural networks. And last, the network can contain elements that are not differentiable such as hard attention.

Due to perturbing the parameters and not the actions ES are invariant to the frequency at which the agent acts in the envirionment. Tradtional MDP-based reinforcement learning methods rely on *frameskip* as one their parameters that is crucial to get right for the optimization to be successful. While this is solvable for problems that do not require long term planning and actions, long term strategic behaviour poses a challenge and reinfocement learning needs hiearchy to be succesful unlike evolutionary strategy.

- Evolution Strategies as a Scalable Alternative to Reinforcement Learning [5]
  - algorithm description
  - comparison with RL
  - paralellization
  - smoothing in action vs. param space
- Improving Exploration in Evolution Strategies for Deep Reinforcement Learning via a Population of Novelty-Seeking Agents [6]
  - novelty search
  - ratio of fitness and novelty and its effects

Chapter 3

Results and discussion

Conclusion

Bibliography

[1] Günter Rudolph. “Evolutionary Strategies”. In: *Handbook of Natural Computing*. Ed. by Grzegorz Rozenberg, Thomas Bäck, and Joost N. Kok. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 673–698. ISBN: 978-3-540-92910-9. DOI: 10.1007/978-3-540-92910-9_22. URL: https://doi.org/10.1007/978-3-540-92910-9_22.

[2] P. A. Vikhar. “Evolutionary algorithms: A critical review and its future prospects”. In: *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*. 2016, pp. 261–265. DOI: 10.1109/ICGTSPICC.2016.7955308.

[3] Hans-Paul Schwefel and Günter Rudolph. “Contemporary evolution strategies”. In: *Advances in Artificial Life*. Ed. by Federico Morán et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 891–907. ISBN: 978-3-540-49286-3.

[4] Nikolaus Hansen. *The CMA Evolution Strategy: A Comparing Review*. 2006.

[5] Tim Salimans et al. *Evolution Strategies as a Scalable Alternative to Reinforcement Learning*. 2017. arXiv: 1703.03864 [stat.ML].

[6] Edoardo Conti et al. *Improving Exploration in Evolution Strategies for Deep Reinforcement Learning via a Population of Novelty-Seeking Agents*. 2018. arXiv: 1712.06560 [cs.AI].

Appendix A

Using CoolThesisSoftware