Autonomous and Adaptive Systems

Multi-agent Learning

Mirco Musolesi

mircomusolesi@acm.org

Definition of Multiagent Systems

- Several possible definitions:
  - *Multiagent systems are distributed systems of independent actors called agents that are each independently controlled but that interact with one another in the same environment.* (see: Wooldridge, “Introduction to Multiagent Systems”, 2002 and Tulys and Stone, “Multiagent Learning Paradigms”, 2018).
  - *Multiagent systems are systems that include multiple autonomous entities with (possibly) diverging information* (see Shoham and Leyton-Brown, “Multiagent systems”, 2009).

Definition of Multiagent Learning

- We will use the following definition of multiagent learning:
  - *"The study of multiagent systems in which one or more of the autonomous entities improves automatically through experience*".

Characteristics of Multiagent Learning

- Different scale:
  - A city or an ant colony or a football team.
- Different degree of complexity:
  - A human, a machine, a mammal or an insect.
- Different types of interaction:
  - Frequent interactions (or not), interactions with a limited number of individuals, etc.

Presence of Regularity

- It is fundamental that there is a certain degree of regularity in the system otherwise prediction of behaviour is not possible.

- Assumption: past experience is somehow predictive of future expectations.

- Dealing with non-stationarity is a key problem.

  - It is the usual problem of reinforcement learning at the end.

Potential Paradigms

- We will consider 5 paradigms:
  - Online Multi-agent Reinforcement Learning towards individual utility
  - Online Multi-agent Reinforcement Learning towards social welfare
  - Co-evolutionary learning
  - Swarm intelligence
  - Adaptive mechanism design

Online Reinforcement Learning towards Individual Utility

- One of the most-studied scenarios in multiagent learning is that in which multiple independent agents take actions in the same environment and learn online to maximise their own individual utility functions (i.e., expected returns).

- From a formal point view (game-theory point of view), this can be considered a repeated normal form game.

  - A *repeated game* is a game that is based of a certain number of repetitions.

  - *Normal form games* are games that are presented using a matrix.

    - As aside, an *extensive form game* is a game for which an explicit representation of the sequence of the players' possible moves, their choices at every decision point and the information about other player's move and relative payoffs.

Example: Prisoner's Dilemma

- The Prisoner's Dilemma is a classic 2-player game.
- Description of the "game": two prisoners committed a crime together and are being interrogated separately.
- If neither of them confesses to the crime (they both "cooperate"), then they will both get a small punishment (corresponding to a payoff of 5).
- If one of them confesses (or "defects"), but the other does not, then the one that confesses gets off for free (payoff of 10), but the other gets the worst punishment possible (payoff of 0).
- If they both defect, they get a worst punishment (payoff of 1)

Prisoners' Dilemma

	Defect	Cooperate
Defect	(1, 1)	(10, 0)
Cooperate	(0, 10)	(5, 5)

Example: Prisoner's Dilemma

- Normal games were initially introduced as one-shot game.
- The players know each other's full utility (reward) functions and play the game only once.
- In this setting, the concept of Nash equilibrium was introduced: a set of actions such that no player would be better off deviating given that the other player's actions are fixed.
- Games can have one or multiple Nash equilibria.
- In the Prisoner's Dilemma, the only Nash Equilibrium is for both agents to defect.

[13] Whitehead, J. H. C., "Simple Homotopy Types." If $W = 1$, Theorem 5 follows from (17:3) on p. 155 of S. Lefschetz, *Algebraic Topology*, (New York, 1942) and arguments in §6 of J. H. C. Whitehead, "On Simply Connected 4-Dimensional Polyhedra" (*Comm. Math. Helv.*, 22, 48–92 (1949)). However this proof cannot be generalized to the case $W \neq 1$.

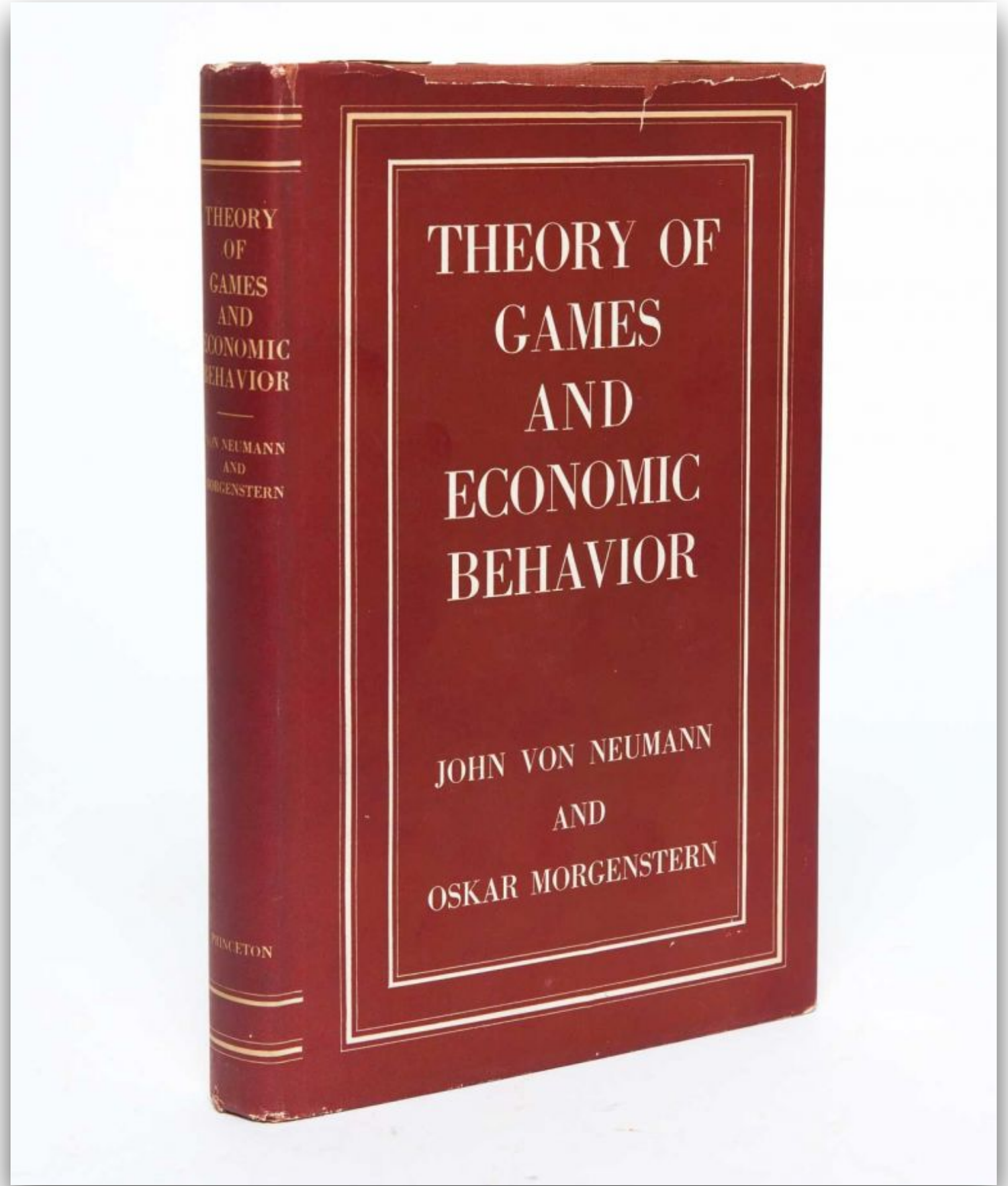
EQUILIBRIUM POINTS IN N-PERSON GAMES

By John F. Nash, Jr.*

Princeton University

Communicated by S. Lefschetz, November 16, 1949

One may define a concept of an n-person game in which each player has a finite set of pure strategies and in which a definite set of payments to the n players corresponds to each n-tuple of pure strategies, one strategy being taken for each player. For mixed strategies, which are probability


THEORY OF GAMES AND ECONOMIC BEHAVIOR
JOHN VON NEUMANN
AND
OSKAR MORGENSTERN
PRINCETON

Repeated Normal Form Games

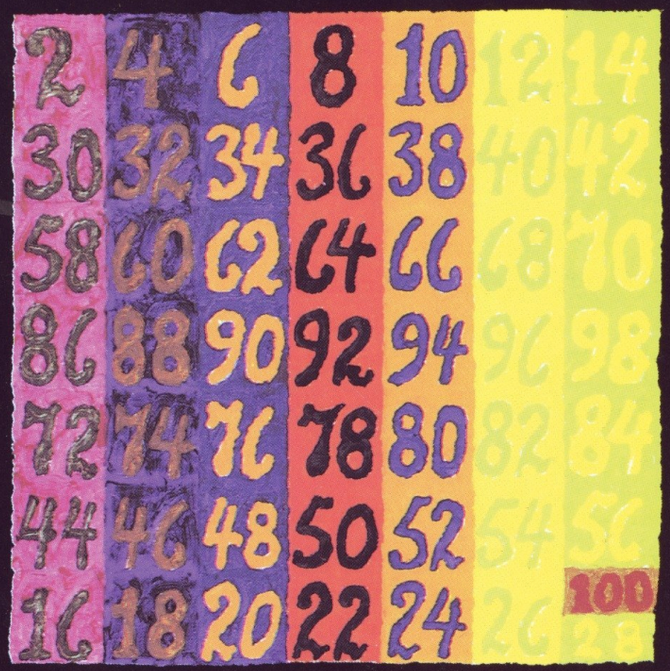
- In repeated normal form games, players interact with one another multiple times with the objectives of maximising their sum utilities (i.e., expected returns) over time.

- As you can imagine, Reinforcement Learning and possibly Deep Reinforcement Learning is well suited for this type of problems.

- Reinforcement Learning can also be used to understand the problem of the “evolution of cooperation” and the presence of altruism: why do we humans cooperate even if in presence of maximisation of personal reward function?

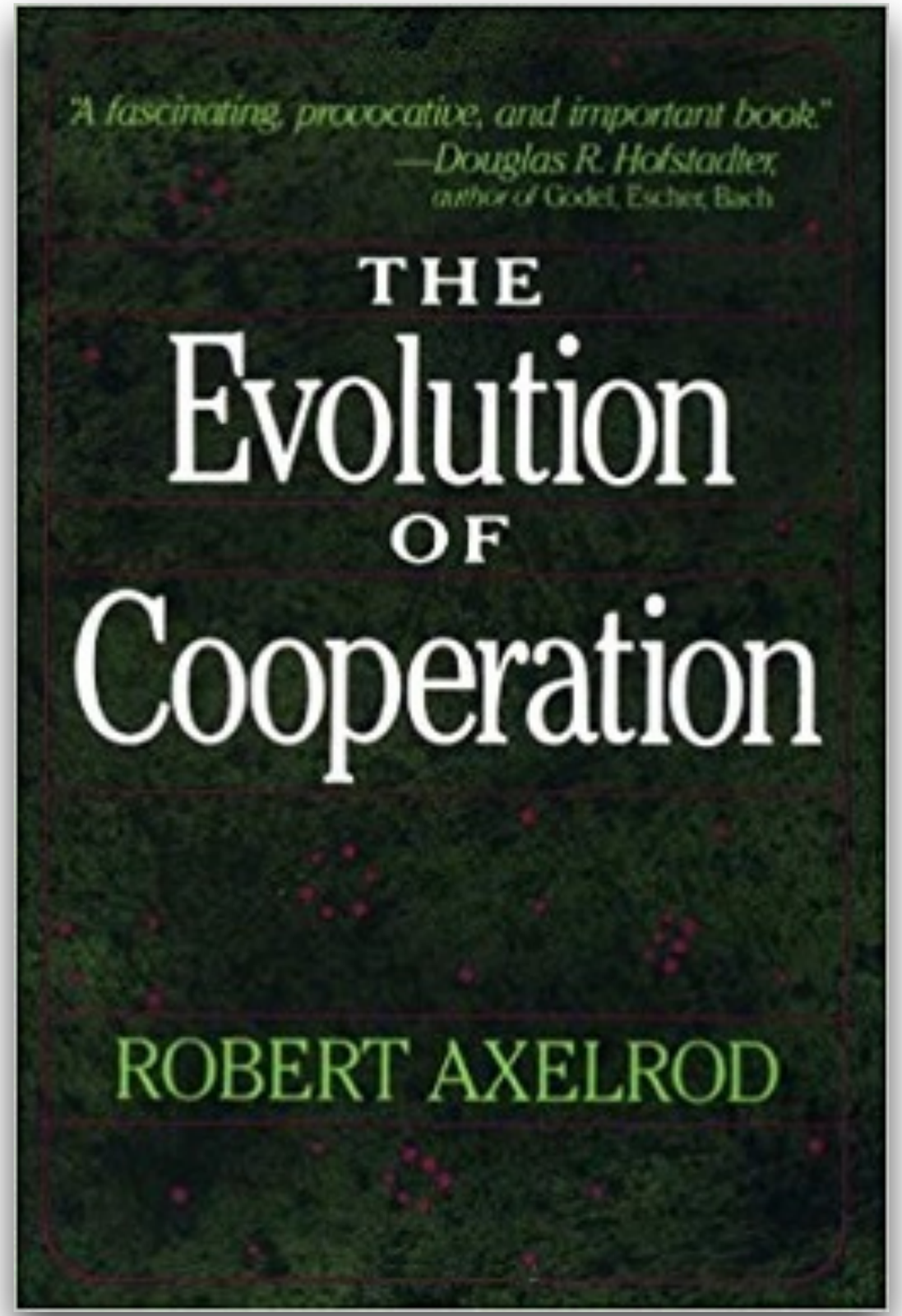
PRISONER'S DILEMMA

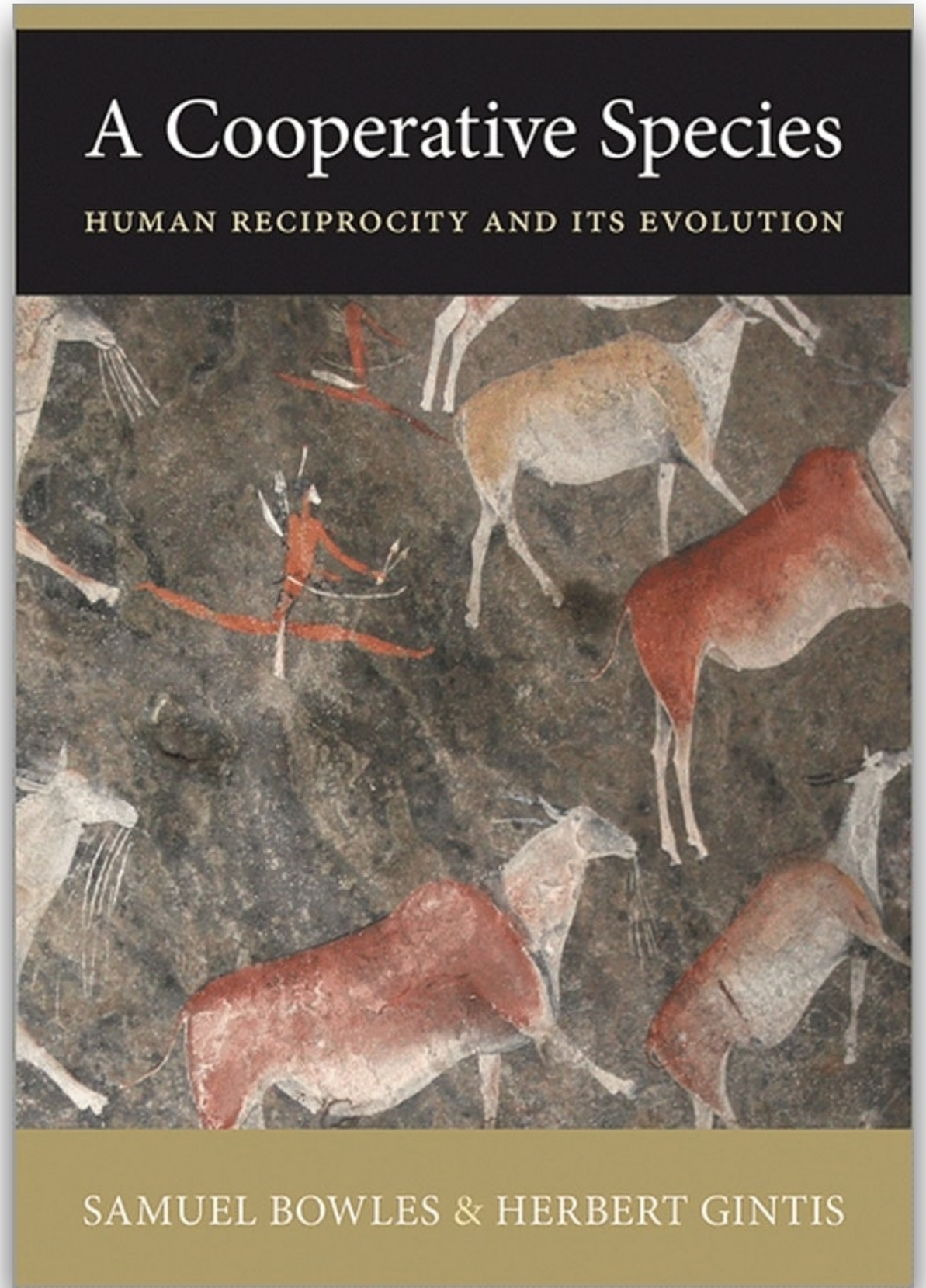
"Both a fascinating biography of von Neumann... and a brilliant social history of game theory and its role in the Cold War and nuclear arms race."
—San Francisco Chronicle



JOHN VON NEUMANN, GAME THEORY, AND THE PUZZLE OF THE BOMB

WILLIAM POUNDSTONE


"A fascinating, prooocative, and important book."
—Douglas R. Hofstadter,
author of Godel, Escher, Bach
THE
Evolution
OF
Cooperation
ROBERT AXELROD


A Cooperative Species
HUMAN RECIPROCITY AND ITS EVOLUTION
SAMUEL BOWLES & HERBERT GINTIS

Online Reinforcement Learning towards Social Welfare

- An alternative paradigm is the one in which multiple independent agents take actions in the same environment and learn online to maximise a global utility function.

- These are also called coordination games, where different players coordinate to achieve a givenobjective (i.e., global expected return).

Multiagent Cooperation and Competition with Deep Reinforcement Learning

Ardi Tampuu* **Tambet Matiisen***

Dorian Kodelja **Ilya Kuzovkin** **Kristjan Korjus**

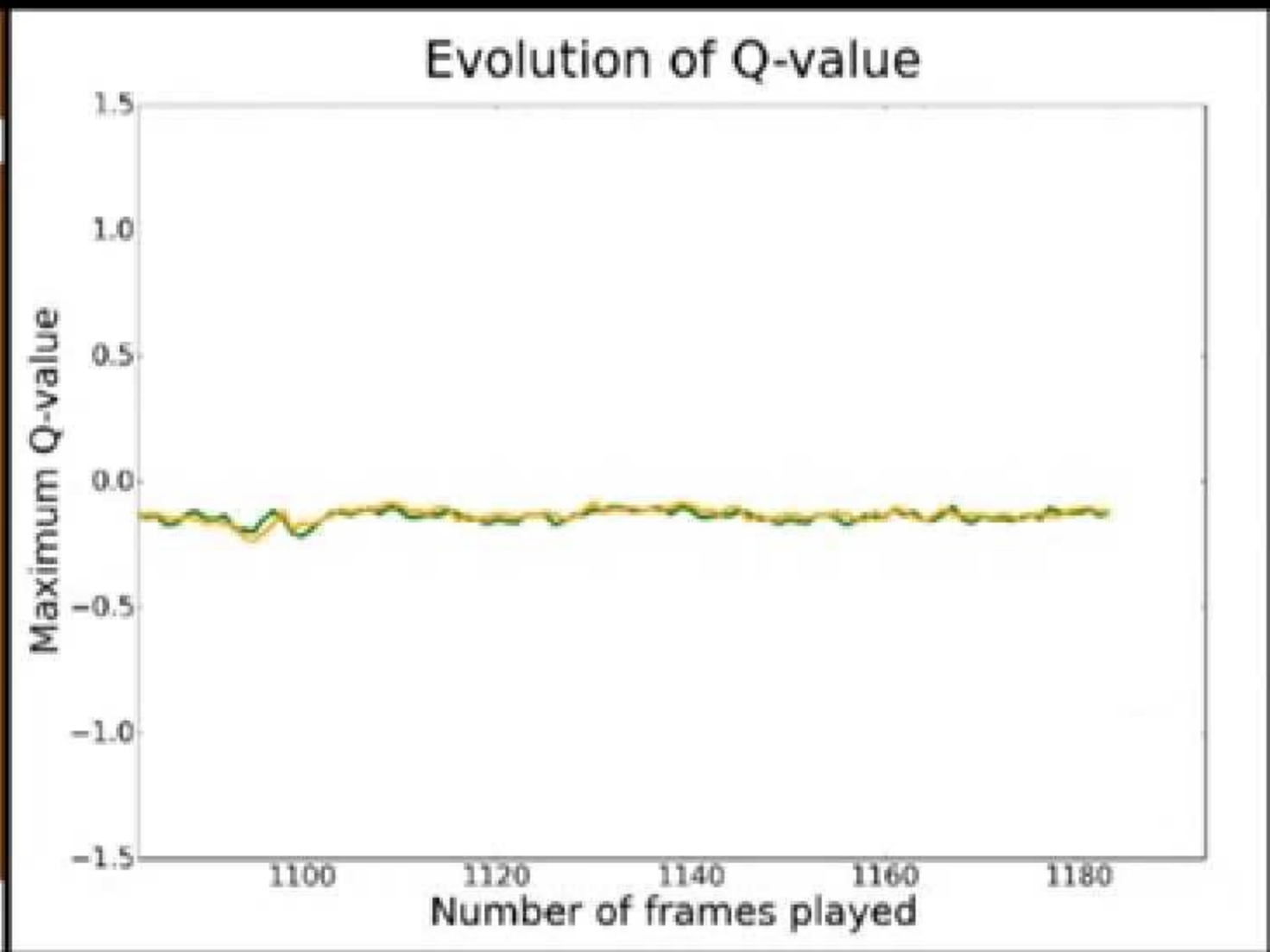
Juhan Aru[†] **Jaan Aru** **Raul Vicente**[✉]

Computational Neuroscience Lab, Institute of Computer Science, University of Tartu
[†] Department of Mathematics, ETH Zurich
ardi.tampuu@ut.ee, tambet.matiisen@ut.ee, raul.vicente.zafra@ut.ee

** these authors contributed equally to this work*

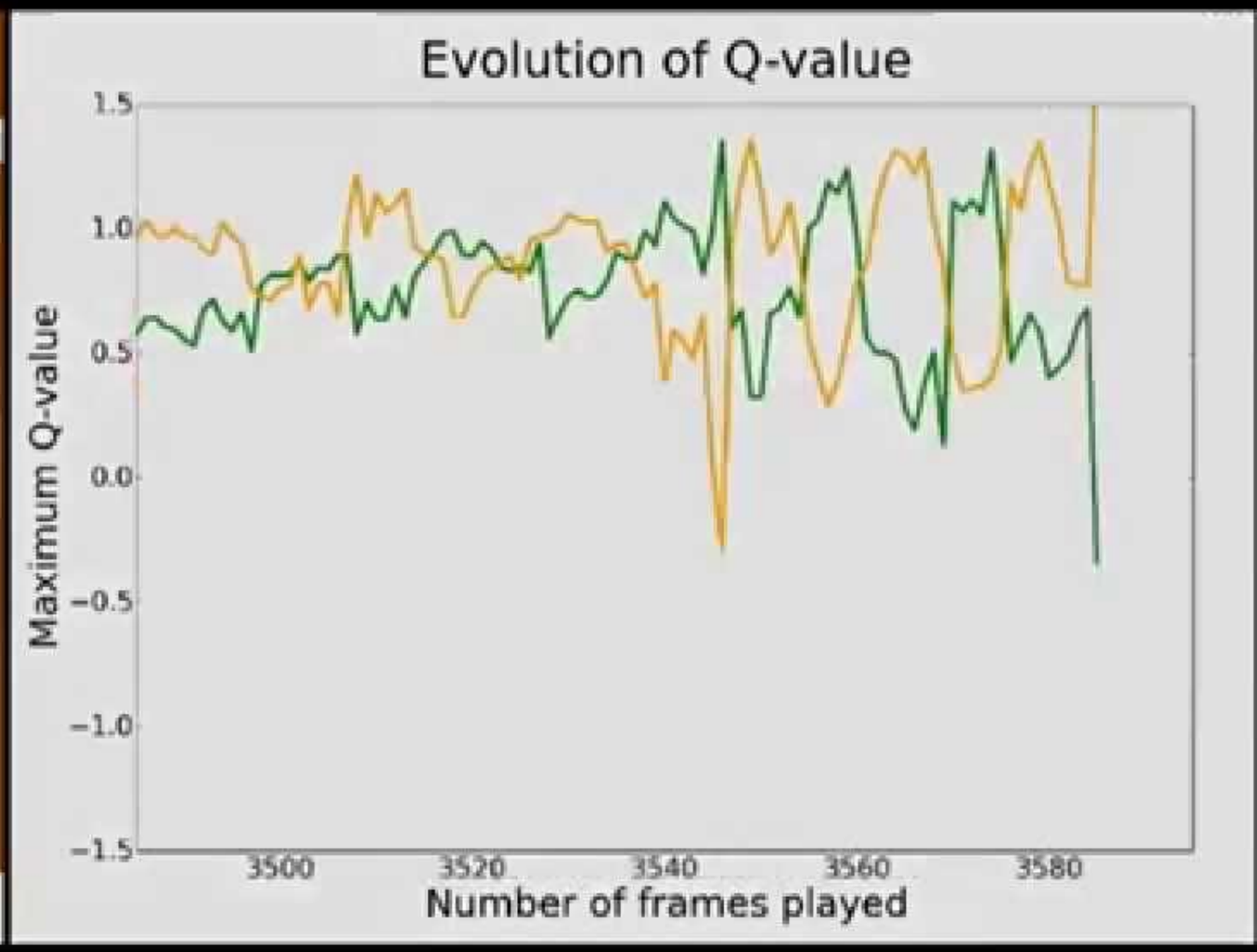
Abstract

Multiagent systems appear in most social, economical, and political situations. In the present work we extend the Deep Q-Learning Network architecture proposed by Google DeepMind to multiagent environments and investigate how two agents controlled by independent Deep Q-Networks interact in the classic videogame *Pong*. By manipulating the classical rewarding scheme of Pong we demonstrate how competitive and collaborative behaviors emerge. Competitive agents learn to play and score efficiently. Agents trained under collaborative rewarding schemes find an optimal strategy to keep the ball in the game as long as possible. We also describe the progression from competitive to collaborative behavior. The present work demonstrates that Deep Q-Networks can become a practical tool for studying the decentralized learning of multiagent systems living in highly complex environments.



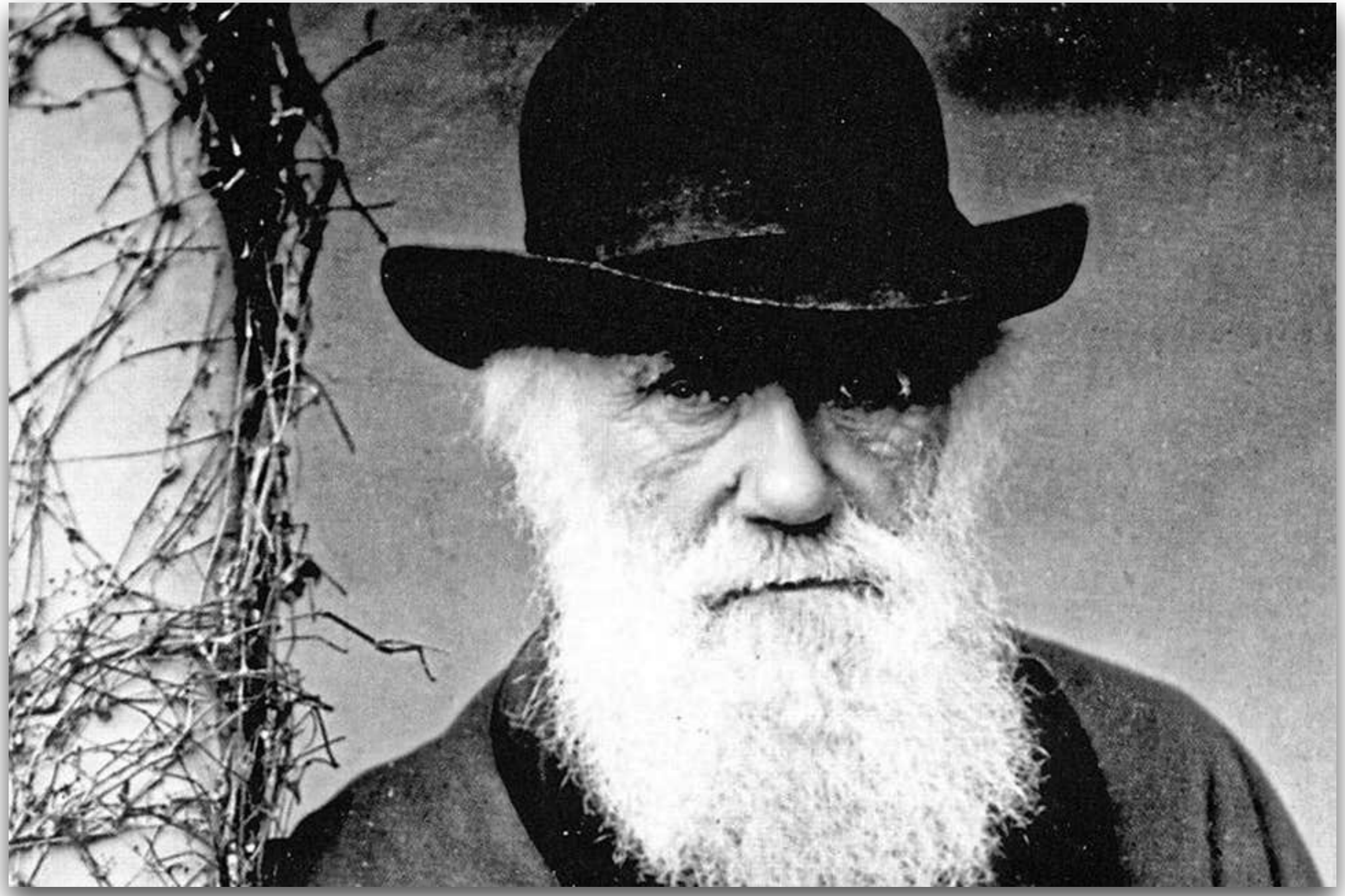
... but are sometimes capable of reaching a state where they never loose the ball.

https://www.youtube.com/watch?v=Gb9DprIgdGw



The agents have learned to deal better with fast and bouncing balls.

https://www.youtube.com/watch?v=nn6_GUVDnVw



Co-evolutionary Approaches

- Evolution can be also used to model and learn agent behaviour as well. According to this paradigm, abstract Darwinian models of evolution are applied to to refine populations of agents (knows as *individuals*).

- These represent candidate solutions to a given problem.

- This process, usually called a *genetic algorithm*, consists of five steps: *representation*, *selection*, *generation of new individuals* (crossover and mutation), *evaluation* and *replacement*.

Evolutionary Algorithms

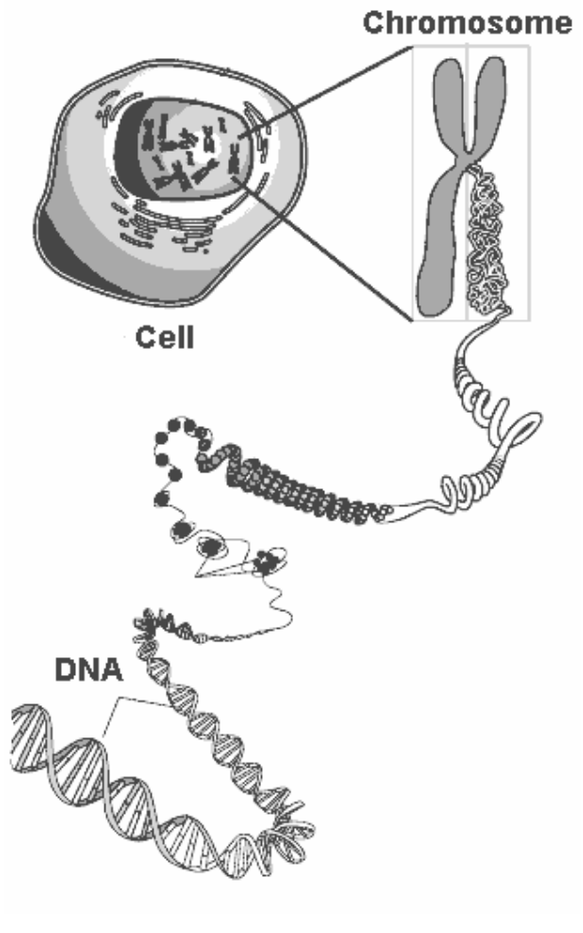
- An evolutionary algorithm begins with an initial population of randomly-generated agents. Each member of this population is then evaluated and assigned a fitness value.

- The evolutionary algorithm then uses a fitness oriented procedure to select agents, breeds and mutates them to produce child agents, which are then added to the population, replacing older agents.

- One evaluation, selection and breeding cycle is knows as a *generation.*

- Successive generations refine a population.

  - You have a given set goal and you might have a time budget.

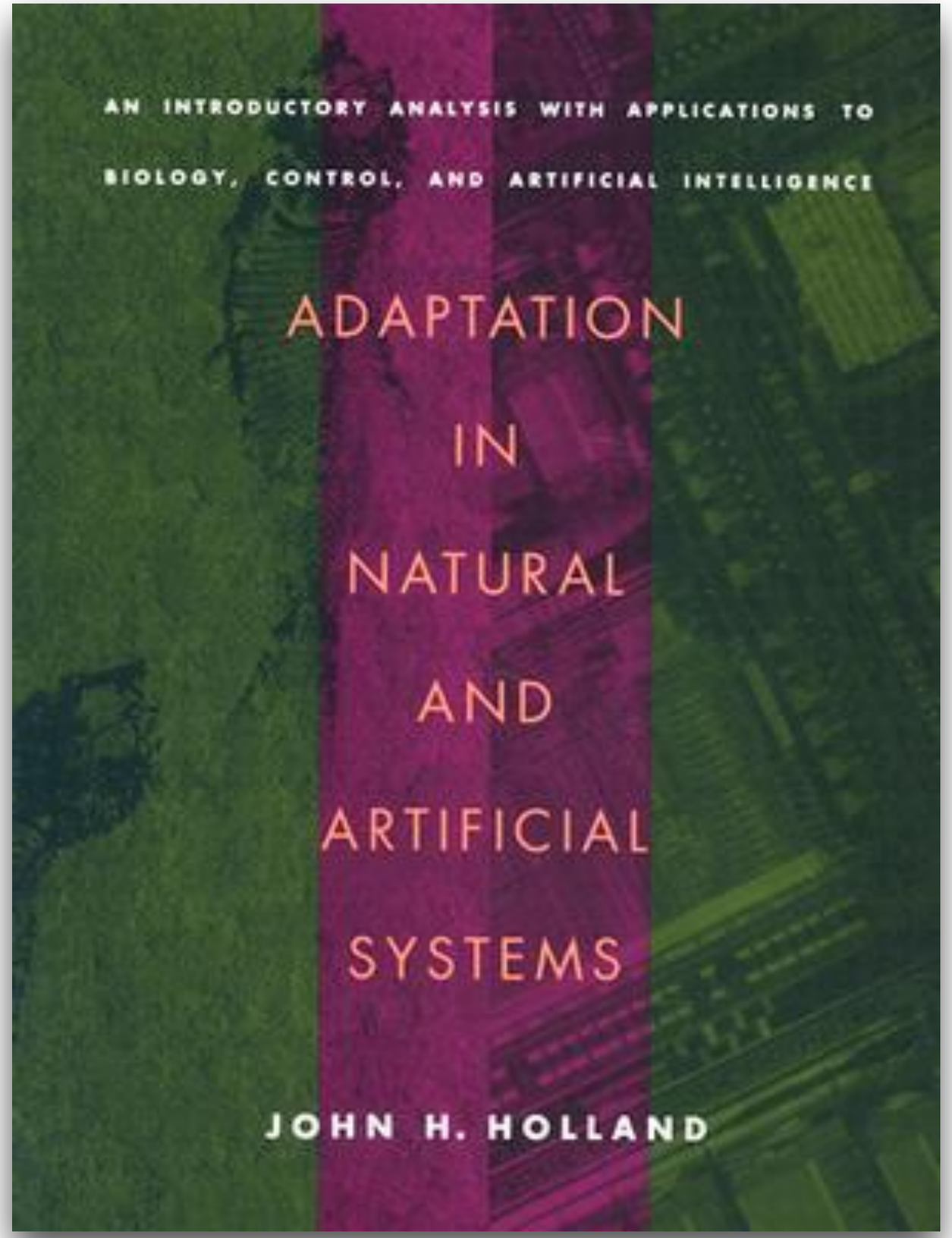


Selection Cycle

Images from Dario Floreano and Claudio Mattiussi. Bio-Inspired Artificial Intelligence. MIT Press 2011.

Problem of Representation

- Typically you encode a “vector” of information using binary coding (for example 4 bits for element of the vector).
  - This was the original version proposed by John Holland in 1970.
- Floating point representations are possible.
- Extensions include the use of real numbers.


AN INTRODUCTORY ANALYSIS WITH APPLICATIONS TO
BIOLOGY, CONTROL, AND ARTIFICIAL INTELLIGENCE
ADAPTATION
IN
NATURAL
AND
ARTIFICIAL
SYSTEMS
JOHN H. HOLLAND

Fitness Function

- You evaluate the performance of the phenotype (i.e., the actual performance of the behavior of your agent encoded through this genotype).
- There is a clear mapping with the biological analogy.
  - Darwin's "survival of the fittest".
- From a practical point of view, you can think about a performance measure as we did in deep learning.

Selection

- At each generation, only some of the individuals reproduce.
- The probability that an individual will make offsprings will be proportional to their fitness.
- One possibility is to have proportionate selection, i.e., the probability that an individual makes an offspring is proportional to how good its fitness is with respect to the population fitness.
- The probability of reproduction will be:

$$p_i = \frac{fitness_i}{\sum_j fitness_j}$$

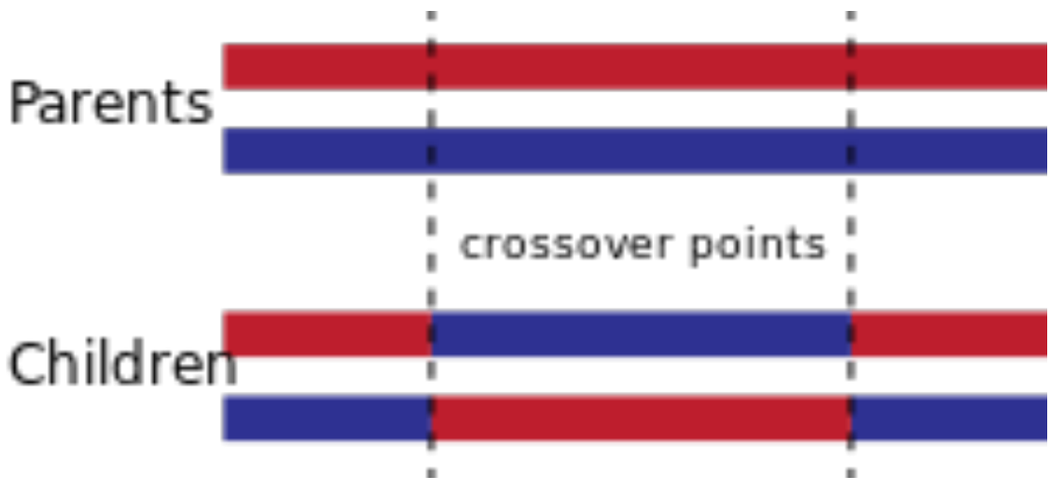
with the sum at the denominator over the entire population.

- You might have a system where only the top K individuals will reproduce (i.e., you sub-select first a set of individuals with high fitness and then you apply the form above).

Cross-over

- In genetic algorithms, crossover (recombination) is an operator that combine the genetic information of two parents to generate offsprings.

- It is one way to stochastically generate new solutions from an existing population and it is analogous to the crossover that happens during biological sexual reproduction.



Source: Wikimedia

Mutation

- Mutation is an operator used to maintain genetic diversity from one generation to the next (see biological mutation).
- Mutation alters one or more bits in the representation (chromosome).
- Example: bit string mutation (one or more bits)
  - 1010**1** -> 1010**0**
- A variety of mutation types have been explored (with different distribution, for groups of bits, flipping bits, etc.)

Evaluation and Replacement

- At each generation, the fitness of each individual is evaluated and using the mechanisms described above, all the entire population is usually entirely replaced by offspring (like in a real biological situation).

- Alternative solutions include an “elitist” solution where we maintain the n best individuals from the previous generation to prevent loss of the best individuals from the population (for example because of the effects of mutations or sub-optimal fitness evaluation).

Coevolution

- Coevoution is an extension of evolutionary algorithms for domains with multiple agents.

- Using evolutionary algorithms, we can train a policy to perform a state to action mapping. In this approach, rather than update the parameters of a single agent interacting with the environment as is done in reinforcement learning, one searches through a *population of policies* that have the highest fitness for the task at hand.

- For example we can use a probability vector as a representation of the policy.

- Alternative include the use of evolutionary algorithms for estimating the hyper-parameters of the networks.

Aside: DNA Computing

- The evolutionary algorithms described above are based on "simulated" DNA.
- Recently, "biological" DNA has been used for computation.
  - This is usually called DNA computing.
- It is characterised by slow processing: response time in minutes/hours, but highly parallel computations (possibly millions/billions).
- Problem: how are you going to read the output?

Aside: DNA Computing

- Initial experiment was done by Leonard Adleman in 1994.
- He address the problem of the directed Hamiltonian Path Problem (“travelling salesman problem”).
  - Different DNA fragment, one per city.
  - DNA mixed in test tube, small fragments form bigger ones presenting the different travel routes.
  - Through a chemical reaction, the DNA fragments representing the longer were eliminated.
  - But it took a while to get an answer…
    - A week!
    - …but this was a proof of concept.

Aside: DNA Computing

- In 2002, Macdonald et al. created a DNA computer able to play tic-tac-toe against a human computer.

- In 2002, the 3-SAT problem with 20 elements was solved (it's a NP-complete).

- Current efforts on reversibility, i.e., reuse of the DNA (DNA gates).

ences therein.
2. M. Orrit, J. Bernard, R. I. Personov, *J. Phys. Chem.* **97**, 10256 (1993).
3. F. Güttler, T. Irngartinger, T. Plakhotnik, A. Renn, U. P. Wild, *Chem. Phys. Lett.* **217**, 393 (1994).
4. M. Ishikawa, K. Hirano, T. Hayakawa, S. Hosoi, S. Brenner, *Jpn. J. Appl. Phys.* **33**, 1571 (1994).
5. E. Betzig and R. J. Chichester, *Science* **262**, 1422 (1993).
6. J. K. Trautman, J. J. Macklin, L. E. Brus, E. Betzig, *Nature* **369**, 40 (1994).
7. D. C. Nguyen, R. A. Keller, J. H. Jett, J. C. Martin, *Anal. Chem.* **59**, 2158 (1987).
8. K. Peck, L. Stryer, A. N. Glazer, R. A. Mathies, *Proc. Natl. Acad. Sci. U.S.A.* **86**, 4087 (1989).
9. W. B. Whitten, L. M. Ramsey, S. A. Arnold, B. V. Bronk, *Anal. Chem.* **63**, 1027 (1991); K. C. Ng, W. B. Whitten, S. A. Arnold, L. M. Ramsey, *ibid.* **64**, 2914 (1992).
10. M. Eigen and R. Rigler, *Proc. Natl. Acad. Sci. U.S.A.* **91**, 5740 (1994).
11. In fluorescence correlation spectroscopy, the intensity recorded at time t is multiplied by that recorded at $t + \Delta t$, and the product is integrated over a finite period of time; see D. E. Koppel, *Phys. Rev. A* **10**, 1938 (1974).
12. T. T. Perkins, D. E. Smith, S. Chu, *Science* **264**, 819 (1994); T. T. Perkins, S. R. Quake, D. E. Smith, S. Chu, *ibid.*, p. 822.
13. S. B. Smith, L. Finzi, C. Bustamante, *ibid.* **258**, 1122 (1992); C. Bustamante, *Annu. Rev. Biophys. Biophys. Chem.* **20**, 415 (1991).
14. M. Washizu and O. Kurosawa, *IEEE Trans. Ind. Appl.* **26**, 1165 (1990).
15. S. B. Smith, P. K. Aldridge, J. B. Callis, *Science* **243**, 203 (1989).
16. N. J. Rampino and A. Chrambach, *Anal. Biochem.* **194**, 278 (1991).
17. K. Morikawa and M. Yanagida, *J. Biochem.* **89**, 693 (1981).
18. I. Auzanneau, C. Barreau, L. Salome, *C. R. Acad. Sci. Paris* **316**, 459 (1993).
19. H. Kabata *et al.*, *Science* **262**, 1561 (1993).
20. D. A. Schafer, J. Gelles, M. P. Sheetz, R. Landick, *Nature* **352**, 444 (1991).
21. Laser excitation at 488.0 and 514.5 nm was provided by an argon ion laser (Lexel Lasers, Fremont, CA). The laser beam entered the microscope through a back port and was directed to an oil-immersion objective (×100, NA = 1.3, Nikon Instrument Group, Melville, NY) by a dichroic beamsplitter (505DRLP02 or

Molecular Computation of Solutions to Combinatorial Problems

Leonard M. Adleman

The tools of molecular biology were used to solve an instance of the directed Hamiltonian path problem. A small graph was encoded in molecules of DNA, and the "operations" of the computation were performed with standard protocols and enzymes. This experiment demonstrates the feasibility of carrying out computations at the molecular level.

In 1959, Richard Feynman gave a visionary talk describing the possibility of building computers that were "sub-microscopic" (*1*). Despite remarkable progress in computer miniaturization, this goal has yet to be achieved. Here, the possibility of computing directly with molecules is explored.

A directed graph G with designated vertices v_{in} and v_{out} is said to have a Hamiltonian path (*2*) if and only if there exists a sequence of compatible "one-way" edges e_1, e_2, . . ., e_z (that is, a path) that begins at v_{in}, ends at v_{out}, and enters every other vertex exactly once. Figure 1 shows a graph that for $v_{in} = 0$ and $v_{out} = 6$ has a Hamiltonian path, given by the edges 0→1, 1→2, 2→3, 3→4, 4→5, 5→6. If the edge 2→3 were removed from the graph, then the resulting graph with the same designated vertices would not have a Hamiltonian path. Similarly, if the designated vertices were changed to $v_{in} = 3$ and $v_{out} = 5$ there would be no Hamiltonian path (because, for example, there are no edges entering vertex 0).

There are well-known algorithms for deciding whether an arbitrary directed graph with designated vertices has a Hamiltonian path or not. However, all known algorithms for this problem have exponential worst-case complexity, and hence there are instances of modest size for which these algorithms require an impractical amount of computer time to render a decision. Because the directed Hamiltonian path problem has been proven to be NP-complete, it seems likely that no efficient (that is, polynomial time) algorithm exists for solving it (*2*, *3*).

The following (nondeterministic) algorithm solves the directed Hamiltonian path problem:

Step 1: Generate random paths through the graph.

Step 2: Keep only those paths that begin with v_{in} and end with v_{out}.

Step 3: If the graph has n vertices, then keep only those paths that enter exactly n vertices.

Step 4: Keep only those paths that enter all of

Department of Computer Science and Institute for Molecular Medicine and Technology, University of Southern California, 941 West 37th Place, Los Angeles, CA 90089, USA.

The Future: Quantum Machine Learning

- Quantum computing is based on properties of quantum mechanics to compute problems that are not feasible for classical computers.

- The basic idea is to use *qubits*, which are like the classic bits, with the two quantum properties of super-position and entanglement.

- In classical computers, probabilistic methods are simulated. In quantum computers, quantum properties can be exploited for speeding up computation considerably.

- The current interest is on developing *quantum neural networks (QNN)* that used to describe parametrised quantum computational model to be executed on quantum computer.

- Quantum neural networks (QNN) are also called parametrised quantum circuit (PQC).

REVIEW

doi:10.1038/nature23474

Quantum machine learning

Jacob Biamonte[1,2], Peter Wittek[3], Nicola Pancotti[4], Patrick Rebentrost[5], Nathan Wiebe[6] & Seth Lloyd[7]

Fuelled by increasing computer power and algorithmic advances, machine learning techniques have become powerful tools for finding patterns in data. Quantum systems produce atypical patterns that classical systems are thought not to produce efficiently, so it is reasonable to postulate that quantum computers may outperform classical computers on machine learning tasks. The field of quantum machine learning explores how to devise and implement quantum software that could enable machine learning that is faster than that of classical computers. Recent work has produced quantum algorithms that could act as the building blocks of machine learning programs, but the hardware and software challenges are still considerable.

Long before we possessed computers, human beings strove to find patterns in data. Ptolemy fitted observations of the motions of the stars to a geocentric model of the cosmos, with complex epicycles to explain the retrograde motions of the planets. In the sixteenth century, Kepler analysed the data of Copernicus and Brahe to reveal a previously hidden pattern: planets move in ellipses with the Sun at one focus of the ellipse. The analysis of astronomical data to reveal such patterns gave rise to mathematical techniques such as methods for solving linear equations (Newton–Gauss), learning optima via gradient descent (Newton), polynomial interpolation (Lagrange), and least-squares fitting (Laplace). The nineteenth and early twentieth centuries gave rise to a broad range of mathematical methods for analysing data to reveal the patterns that it contained.

The construction of digital computers in the mid-twentieth century allowed the automation of data analysis techniques. Over the past half-century, the rapid progression of computer power has allowed the implementation of linear algebraic data analysis techniques such

a set of instructions solving a problem, such as determining whether two graphs are isomorphic, that can be performed on a quantum computer. Quantum machine learning software makes use of quantum algorithms as part of a larger implementation. By analysing the steps that quantum algorithms prescribe, it becomes clear that they have the potential to outperform classical algorithms for specific problems (that is, reduce the number of steps required). This potential is known as quantum speedup.

The notion of a quantum speedup depends on whether one takes a formal computer science perspective—which demands mathematical proofs—or a perspective based on what can be done with realistic, finite-size devices—which requires solid statistical evidence of a scaling advantage over some finite range of problem sizes. For the case of quantum machine learning, the best possible performance of classical algorithms is not always known. This is similar to the case of Shor's polynomial-time quantum algorithm for integer factorization: no sub-exponential-time classical algorithm has been found, but the possibility is not provably

TensorFlow Install Learn API Resources More Search Language GitHub Sign in

Quantum

Overview Guide & Tutorials API

TensorFlow Quantum is a library for hybrid quantum-classical machine learning.

TensorFlow Quantum (TFQ) is a quantum machine learning library for rapid prototyping of hybrid quantum-classical ML models. Research in quantum algorithms and applications can leverage Google's quantum computing frameworks, all from within TensorFlow.

TensorFlow Quantum focuses on *quantum data* and building *hybrid quantum-classical models*. It integrates quantum computing algorithms and logic designed in Cirq, and provides quantum computing primitives compatible with existing TensorFlow APIs, along with high-performance quantum circuit simulators. Read more in the TensorFlow Quantum white paper.

Start with the overview, then run the notebook tutorials.

```
# A hybrid quantum-classical model.
model = tf.keras.Sequential([
    # Quantum circuit data comes in inside of tensors.
    tf.keras.Input(shape=(), dtype=tf.dtypes.string),

    # Parametrized Quantum Circuit (PQC) provides output
    # data from the input circuits run on a quantum computer.
    tfq.layers.PQC(my_circuit, [cirq.Z(q1), cirq.X(q0)]),

    # Output data from quantum computer passed through model.
    tf.keras.layers.Dense(50)
])
```

Announcing TensorFlow Quantum

Read on the Google AI blog

TensorFlow Quantum (TF Dev Summit '20)

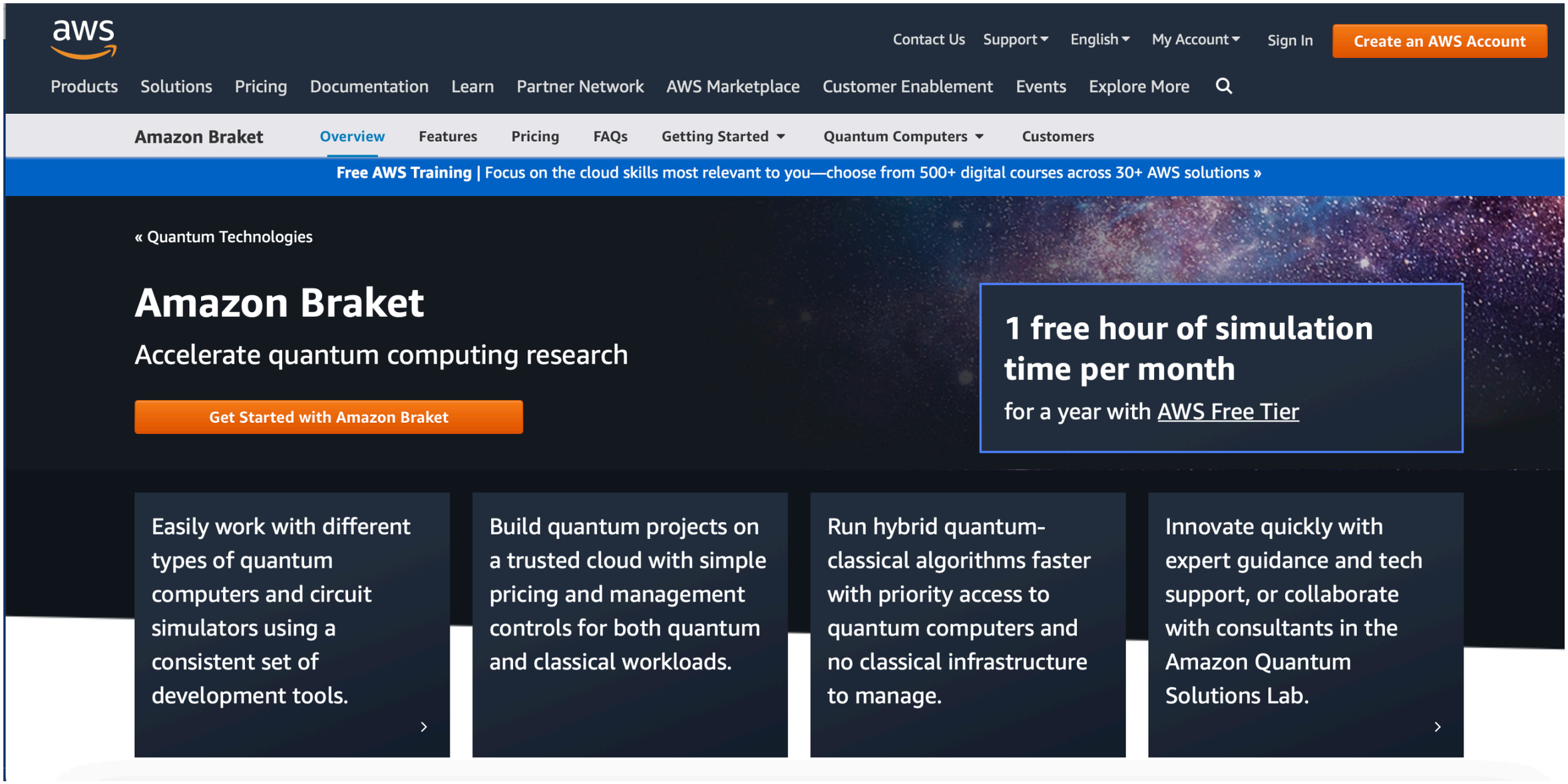
Watch the video

Programming a quantum computer with Cirq

Watch the video

TensorFlow Quantum on GitHub

View on GitHub


aws
Contact Us
Support
English
My Account
Sign In
Create an AWS Account
Products
Solutions
Pricing
Documentation
Learn
Partner Network
AWS Marketplace
Customer Enablement
Events
Explore More
Amazon Braket
Overview
Features
Pricing
FAQs
Getting Started
Quantum Computers
Customers
Free AWS Training | Focus on the cloud skills most relevant to you—choose from 500+ digital courses across 30+ AWS solutions »
« Quantum Technologies
Amazon Braket
Accelerate quantum computing research
Get Started with Amazon Braket
1 free hour of simulation time per month
for a year with AWS Free Tier
Easily work with different types of quantum computers and circuit simulators using a consistent set of development tools.
Build quantum projects on a trusted cloud with simple pricing and management controls for both quantum and classical workloads.
Run hybrid quantum-classical algorithms faster with priority access to quantum computers and no classical infrastructure to manage.
Innovate quickly with expert guidance and tech support, or collaborate with consultants in the Amazon Quantum Solutions Lab.

Swarm Intelligence

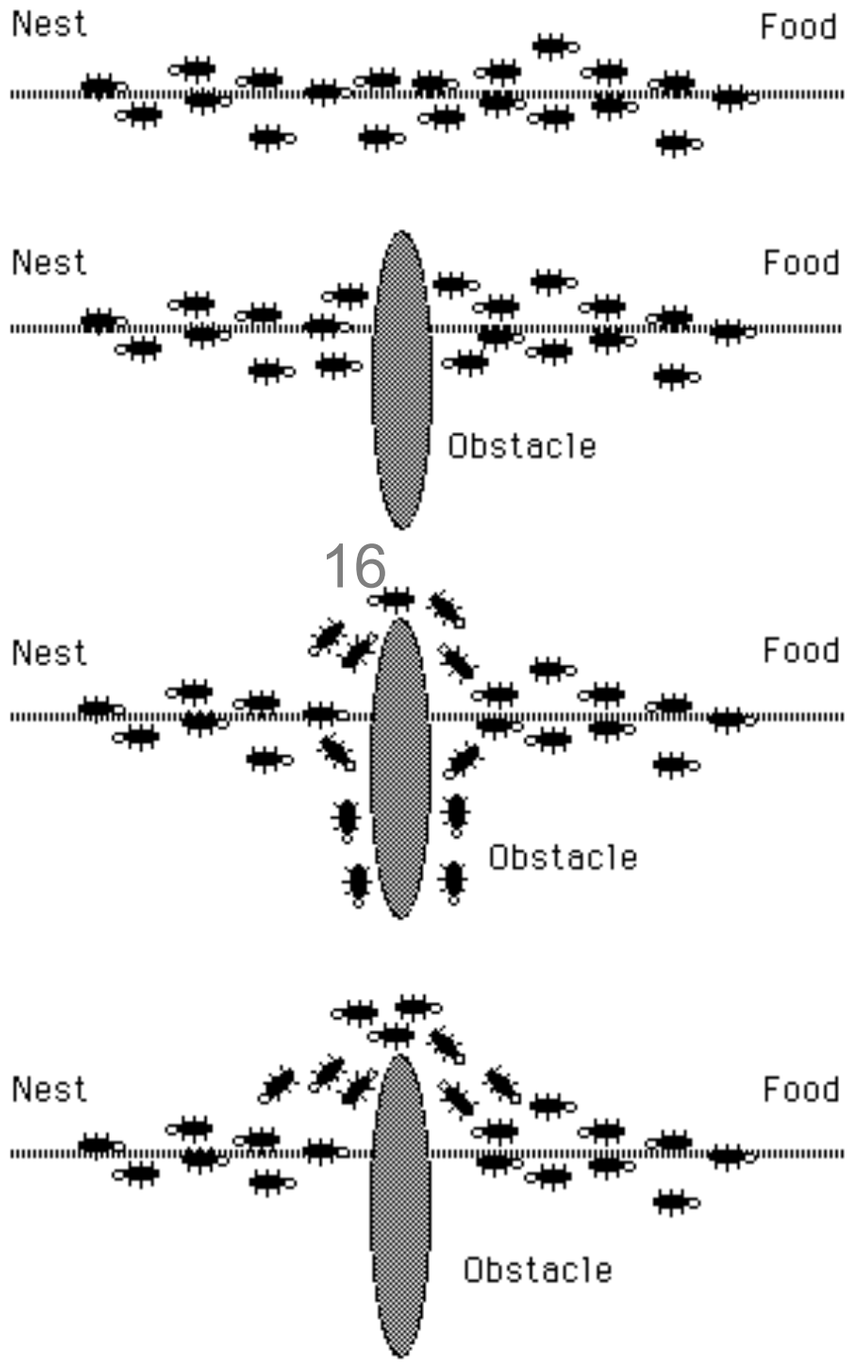
- Swarm intelligence is a bio-inspired machine learning technique based on the behaviour of social insects (e.g., ants and honeybees).
- The goal is to develop self-organised and decentralised adaptive algorithms.
- The learning is based on a large number of agents (usually with limited “computation” capabilities) that locally interact.
- The idea is to develop algorithms that lead to the emergence of cooperative behaviour in the population.
- Complex behaviour from simple local rules.

Swarm Intelligence

Video from Dario Floreano and Claudio Mattiussi. Bio-Inspired Artificial Intelligence. MIT Press 2011.

Stigmergy

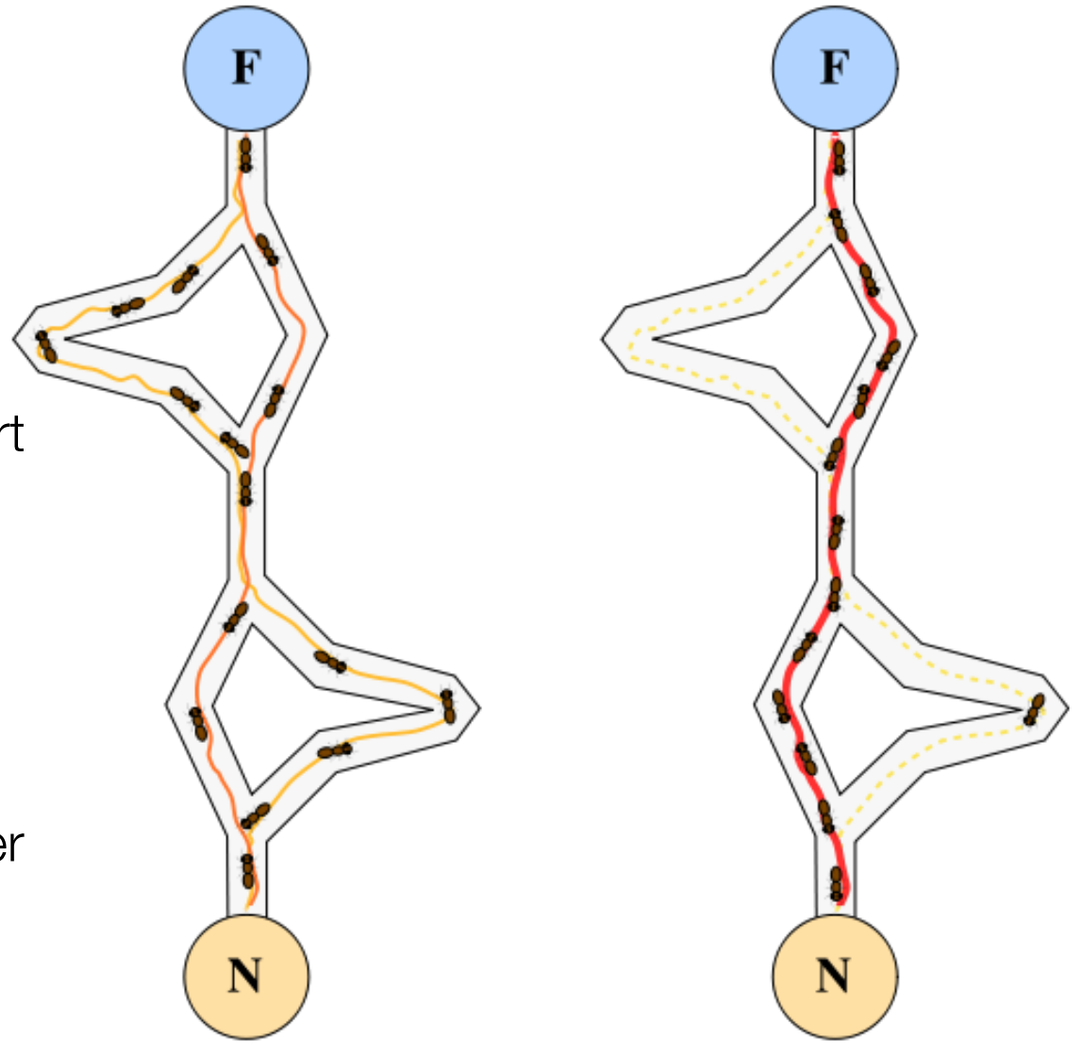
- The term *stimergy* indicates communication among individuals through modification of the environment.
- For example, some ants leave a chemical (pheromone) trail behind to trace the path.
- The chemical decays over time.
- This allows other ants to find the paths between the food and the nest.
- It also allows ants to find the *shortest path* among alternatives.



Source: Dario Floreano and Claudio Mattiussi. Bio-Inspired Artificial Intelligence. MIT Press 2011.

Stigmergy and Shortest Paths

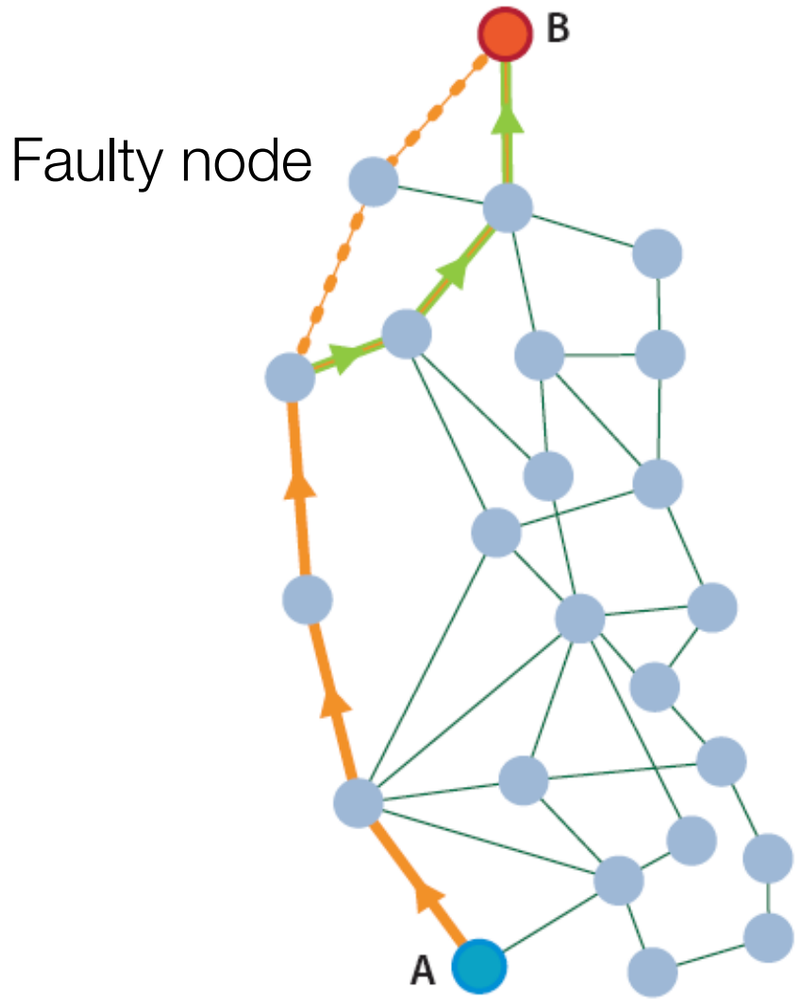
- As they move, ants deposit pheromone.
- Pheromone decays in time.
- Ants follow the paths with the highest pheromone concentration.
- Without pheromone, equal probability of choosing short or long path.
- Shorter paths allow for higher number of passages (it takes less time to go back and forth!)
- Therefore, pheromone level will be higher on the shorter path.
- Ants will increasingly tend to choose the shorter path.



Source: Dario Floreano and Claudio Mattiussi. Bio-Inspired Artificial Intelligence. MIT Press 2011.

Ant Colony Optimization

- Ant Colony Optimization is an algorithm developed by Dorigo et al. inspired upon stigmergic communication to find the shortest path in a network.



- Typical examples are Internet/computer networks problems and other problems that can be described by the Travel Salesman Problem.

- Other problems include scheduling of robots and coverage of areas (represented as networks).

Source: Dario Floreano and Claudio Mattiussi. Bio-Inspired Artificial Intelligence. MIT Press 2011.

Ant Colony Optimization: A New Meta-Heuristic

Marco Dorigo
IRIDIA
Université Libre de Bruxelles
mdorigo@ulb.ac.be

Gianni Di Caro
IRIDIA
Université Libre de Bruxelles
gdicaro@iridia.ulb.ac.be

Abstract- Recently, a number of algorithms inspired by the foraging behavior of ant colonies have been applied to the solution of difficult discrete optimization problems. In this paper we put these algorithms in a common framework by defining the Ant Colony Optimization (ACO) meta-heuristic. A couple of paradigmatic examples of applications of these novel meta-heuristic are given, as well as a brief overview of existing applications.

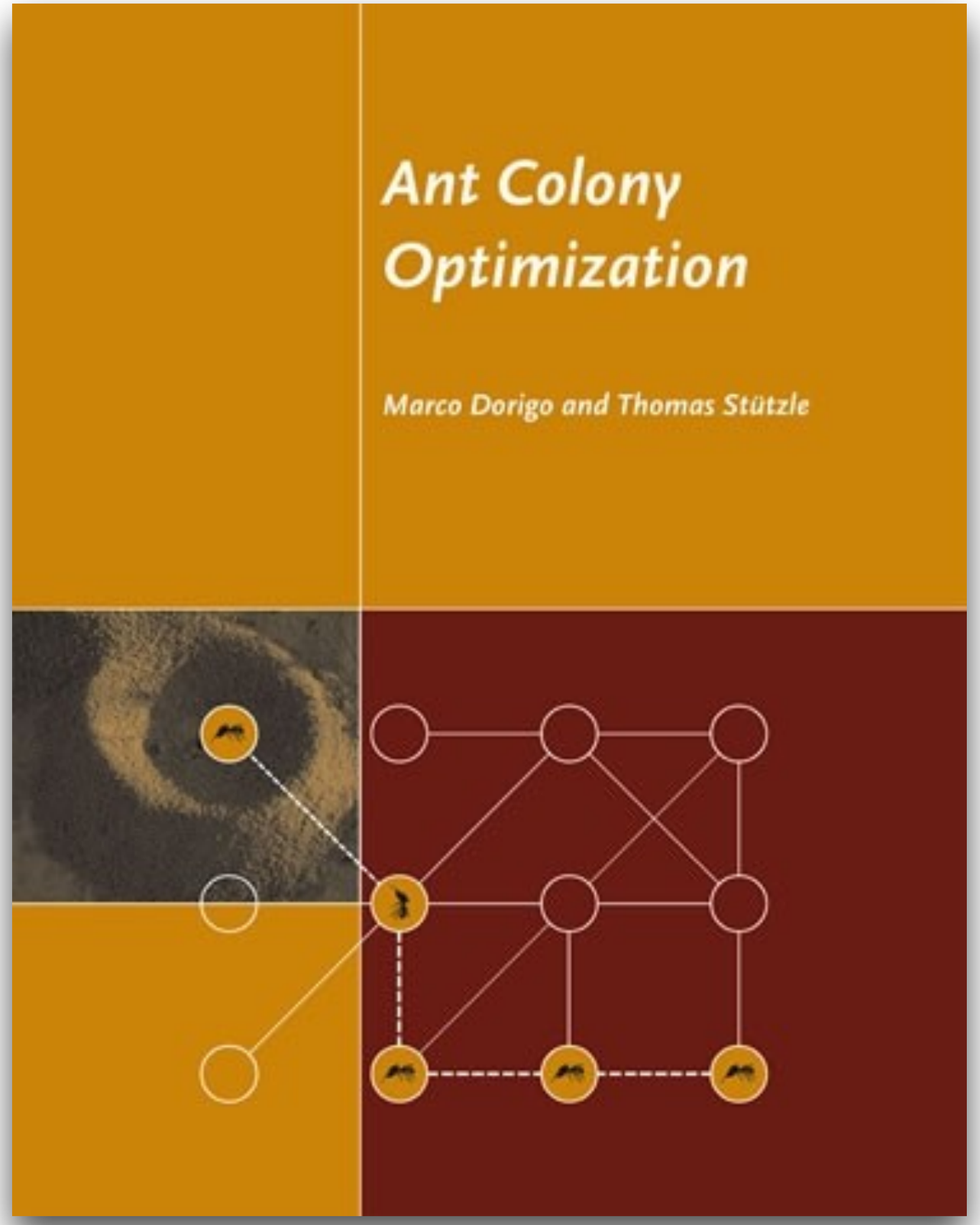
1 Introduction

In the early nineties an algorithm called *ant system* was proposed as a novel heuristic approach for the solution of combinatorial optimization problems (Dorigo *et al.*, 1991; Dorigo, 1992; Dorigo *et al.*, 1996). Ant system (AS), which was first applied to the traveling salesman problem, was recently extended and/or modified both to improve its performance and to apply it to other optimization problems. Improved versions of AS include, among others, ACS (Dorigo & Gambardella, 1997), $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System (Stützle & Hoos, 1998b), and AS_{rank} (Bullnheimer *et al.*, 1997b). All these algorithms have been applied to the TSP with varying degrees of success, but always improving over AS perfor-

2 The ACO meta-heuristic

The ACO meta-heuristic can be applied to discrete optimization problems characterized as follows.

- $C = \{c_1, c_2, \ldots, c_{N_C}\}$ is a finite set of *components*.
- $L = \{l_{c_i c_j} \mid (c_i, c_j) \in \tilde{C}\}$, $|L| \leq N_C^2$ is a finite set of possible *connections/transitions* among the elements of $\tilde{C}$, where $\tilde{C}$ is a subset of the Cartesian product $C \times C$.
- $J_{c_i c_j} \equiv J(l_{c_i c_j}, t)$ is a *connection cost* function associated to each $l_{c_i c_j} \in L$, possibly parameterized by some time measure t.
- $\Omega \equiv \Omega(C, L, t)$ is a finite set of *constraints* assigned over the elements of C and L.
- $s = \langle c_i, c_j, \ldots, c_k, \ldots \rangle$ is a sequence over the elements of C (or, equivalently, of L). A sequence s is also called a *state* of the problem. If S is the set of all possible sequences, the set $\tilde{S}$ of all the (sub)sequences that are feasible with respect to the constraints $\Omega(C, L, t)$, is a subset of S. The elements in $\tilde{S}$ define the problem's *feasible states*. The length of a sequence s, that is, the number of components in the sequence, is expressed by $|s|$.


Ant Colony
Optimization
Marco Dorigo and Thomas Stützle

Adaptive Mechanism Design

- It is also possible to think of a multi-agent learning setting in which the agents are fixed (or not controllable by us), but the interaction mechanism is to be learned.
- Typical example is an auction with a population of bidders.
  - Several parameters can be controlled:
    - Minimum price, simultaneous or non-simultaneous actions, mechanism (English auction, Vickrey auction, Dutch auction, etc.).
  - In this case, the auction house is not able to control the bidders (interacting agents), but the rules of interaction.
  - The parameters can be learned and refined over time (mechanism design).
- Other applications: frequency bidding, design of competition markets, etc.

DeepMind

2020-12-17

Open Problems in Cooperative AI

Allan Dafoe[1], Edward Hughes[2], Yoram Bachrach[2], Tantum Collins[2], Kevin R. McKee[2], Joel Z. Leibo[2], Kate Larson[2, 3] and Thore Graepel[2]

[1]Centre for the Governance of AI, Future of Humanity Institute, University of Oxford, [2]DeepMind, [3]University of Waterloo

Problems of cooperation—in which agents seek ways to jointly improve their welfare—are ubiquitous and important. They can be found at scales ranging from our daily routines—such as driving on highways, scheduling meetings, and working collaboratively—to our global challenges—such as peace, commerce, and pandemic preparedness. Arguably, the success of the human species is rooted in our ability to cooperate. Since machines powered by artificial intelligence are playing an ever greater role in our lives, it will be important to equip them with the capabilities necessary to cooperate and to foster cooperation.

References

- Stefano V. Albrecht and Peter Stone. Autonomous Agents Modelling Other Agents: A Comprehensive Survey and Open Problems. Artificial Intelligence. Volume 258. 2018.

- Lucian Busoniu, Robert Babuska, Bart De Schutter. A Comprehensive Survey of Multiagent Reinforcement Learning. In IEEE Transactions on Systems, Man and Cybernetics. Volume 38. Issue 2. March 2008.

- Dario Floreano and Claudio Mattiussi. Bio-Inspired Artificial Intelligence. Theories, Methods and Technologies. MIT Press. 2011.

References

- Kevin Leyton-Brown and Yoav Shoham. Multiagent Systems, Game-theoretic and Logical Foundations. Cambridge University Press. 2009.

- John H. Holland. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. MIT Press. 1992 (original edition 1975).

- Karl Tuyls and Gerhard Weiss. Multiagent Learning: Basics, Challenges and Prospects. AI Magazine. Volume 33. Issue 3. 2012.

References

- Karl Tulys and Peter Stone. Multiagent Learning Paradigms. In Francesco Belardinelli and Estefania Argente, editors, Multi-agent Systems and Agreement Technologies. Lecture Notes in Artificial Intelligence. Pages 3-21. Springer 2018.

- Micheal Wooldridge. An Introduction to MultiAgent Systems. Second Edition. Wiley. 2009.

References

- Kaiqing Zhang, Zhuoran Yang and Tamer Basar. Multi-agent Reinforcement Learning: A Selective Overview of Theories and Algorithms. arXiv:1911.10635v2.

- Sven Gronauer and Klaus Diepold. Multi-agent Reinforcement Learning: A Survey. Artificial Intelligence Review. 55:895-943. Springer. 2022.