Common tasks in Evolutionary Robotics, an overview

Corné Sprong

April 25, 2011

1 Introduction

The field of Evolutionary Robotics (ER) studies the use of Evolutionary Computing to generate controllers for robots. The objective of this paper is to summarise and structurise tasks that are and have been used in ER. We will end with a discussion of trends in ER. This paper is also intended for students that are new to ER, so I will start with some basic notions.

2 Basic Notions

2.1 Terminology and abbreviations

- a-priori information - Information that is given before the evolutionary run, that is information about the solution that is incorporated in the fitness function.
- Distance metric - A measure of similarity
- DC - Direct Current
- DOF - Degrees of Freedom
- ER - Evolutionary Robotics
- Heuristic - An algorithm that can generate a reasonably good solution in a decent amount of time, but for which no formal proof of correctness exists.
- GA - Genetic Algorithm
- GP - Genetic Programming / Genetic Program
- IR - Infra-red
- Morphology - 'The shape of the robot'
- NEAT - Neuro Evolution of Augmented Topologies, a genetic algorithm for evolving neural networks.
- Phototaxis - Movement in response to light
- Proprioception - Perception of the relative position of parts of the body.
- Tactile - Relating to touch.

2.2 Robots

Many researchers use custom made robots for their experiments. These are usually small, wheeled robots with a differential drive. For gait evolution legged robots are used. The following are commonly used robots or robot kits.

The Khepera is a circular robot with a diameter of 55 mm, a height of 30 mm and weighing 70 g. Its two wheels are powered by two DC motors. It comes with 8 IR sensors, though it can also be customized through the use of extension turrets. The original Khepera was a very popular robot, having been used for some ten years by hundreds of universities and aiding in the development of Evolutionary robotics.

The Sony AIBO is a quadruped robot outfitted with light sensors, touch sensors and sound sensors. It comes in various models.

LEGO Mindstorm is a kit for building robots.

The s-bot has a cylindrical turret (diameter 11.6 cm) which can be rotated w.r.t the chassis. The motor base is outfitted with 2 tracks and 2 wheels, this combination is labelled Differential Treels Drive where a motor on each side powers both the track and the connected wheel. S-bots are capable of physically connecting to one another, forming a swarm-bot. Each s-bot has 8 leds, omnidirectional sensors, a microphone and loudspeakers. [Mondada et al., 2004]

The Cyber Rodent is a two-wheeled robot, 250 mm long and weighs 1.7 kg. It has omnidirectional vision, infra-red proximity sensors, three coloured LEDs and IR ports.

The e-puck is a small (75 mm in diameter) two-wheeled robot. It has 8 IR-sensors, 3 microphones, a colour camera, and a 3D accelerometer. Its actuators include a speaker and red and green coloured LEDs. The capabilities can be further increased using extension modules. [Mondada et al., 2009]

Elvis is a 60 cm humanoid robot with 42 DOF powered by servos. It has two cameras mounted on its head, microphones and touch sensors in its fingers. It uses a three layered software architecture that is designed for use with evolutionary algorithms and genetic programming in particular. [Nordin and Nordahl, 1999]

Research in evolutionary robotics is not limited to small, mobile robots. For example, in [Furey and Harvey, 2008], the robot is a traction kite on a spool and in [Bianco and Nolfi, 2004] a robotic arm is used.

2.3 Controllers

A controller is the computational part of a robot, its "brain". Many different controller types are used. Neural networks are based on the naturally occuring networks of biological neurons. They are a popular controller representation because of robustness and expressivity. A downside to neural networks is a lack of insight in how the controller works. Neural networks can have recurrent and self-connections, which can serve as memory and allow time-dependent behaviour. Other controller types include directed graphs, sets of parameters, fuzzy logic and collections of rules.

2.4 Fitness functions

For complex tasks, hand-coding controllers becomes impractical or impossible. Evolutionary robotics seeks to generate controllers automatically through a process of artificial evolution. This requires the task to be performed to be captured in a fitness function, a measure of how well a robot is doing. This fitness function acts as a bridge between task and evolutionary algorithm. The choice of fitness function is a - or even *the* - key element in the quality of the resulting controllers. Fitness functions can be classified according to the amount of apriori information they incorporate [Nelson et al., 2009]. Training sets specify the desired output for given inputs. This introduces a large amount of apriori information. It might not be possible to generate an appropriate training set, for example when an effective heuristic is not known. Behavioral fitness function terms reward sensor-reaction mappings, this adds a medium amount of a-priori information. Aggregrate functions 'summarize' the result after the evaluation period (How well it did, not what it did). This adds a low amount of a-priori information, but see bootstrap problem below.

In its current state, ER is focused on designing evolutionary systems capable of performing new tasks with greater complexity, instead of improving learning efficiency. A big problem in ER is that it currently does not scale well for more complex tasks. The initial population is likely to perform below the minimal functionality limit. When all fitnesses are equal, no selection is possible. This is known as the bootstrap problem. To avoid this problem the training can be done in phases, where the controller is first trained on simpler subproblems. This is called staged evolution. A drawback of this approach is that it forces the evolution to occur along a certain path, based on the assumptions of the user. These assumptions might be unjustified and lead to a suboptimal solution. For certain complex tasks it might be difficult or impossible to identify subproblems. Other solutions to the bootstrap problem are environmental complexification, fitness shaping and behavioral decomposition. [Mouret and Doncieux, 2009] suggests using multi-objective optimisation algorithms to increase the diversity in a population. A downside is that it requires a distance metric between behaviors, which for some problems might not be easily defined, though several general suggestions are made.

2.5 Embodied versus Simulated evolution

Embodied evolution happens in physical robots. Embodied evolution is not practical for all problems, for example when there is a risk that the robots get damaged. This would quickly halt progress until the robots are repaired or replaced. Embodied evolution also needs to occur in real-time, whereas simulation can take place at faster than real-time. A drawback with using simulation is that the evolved behaviour might not be as effective in the real world, due to inaccuracies in the used models. This is known as the "correspondence problem"'. Most researchers that perform the evolution in a simulated environment transfer the behaviour to real robots for validation. [Jakobi, 1998a] discusses the conditions that aid in a successful transfer.

2.6 Environment

The environment does not need to remain the same during a run. It can gradually be made more complex as the population improves. The authors of [Pasemann et al., 2001] first evolved robot controllers for wall avoidance. They then used the resulting population as a basis for a phototaxis task. The environment was a maze with light sources. During the run, they added more obstacles and reduced the number of light sources. The increasing

difficulty led to robust controllers that were able to backtrack and explore dead ends. The controllers obtained in simulation successfully transferred to real Khepera robots.

2.7 Fitness function notation

To simplify comparisons between fitness functions, common symbols were standardised as in [Nelson et al., 2009]. A lower case f denotes an integrand (over time), while a capital F will be used for aggregate fitness function. If a function is minimised instead of maximised, this is written with a subscript, e.g. $f_{(-)}$. v_l and v_r will be used for left and right speeds, where appropriate. Distance travelled as d, sensor activation levels as s. A boolean function as B and coordinates as x,y.

3 Tasks

Tasks can be classified by properties such as the degree of interaction with the environment, or whether they naturally involve groups of robots or just one. First I consider tasks where the goal is to acquire locomotion, this includes wheeled, legged and winged robots. Next are tasks where there is a given destination or orientation, such as object homing or phototaxis. Then tasks that require the robot to manipulate its environment, for example box pushing or gathering. This ordering is roughly in ascending complexity, because tasks in one category are subtasks of tasks in the next, for example a task like box pushing also requires learning locomotion.

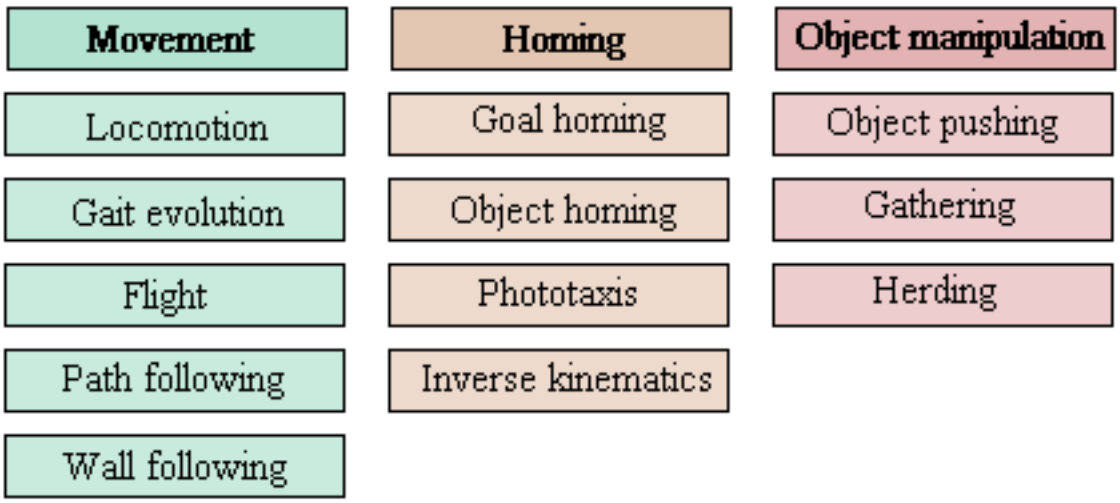

Figure 1: Classification of tasks into categories

Sometimes tasks involve multiple robots cooperating and/or competing. Robots are always implicitly competing for fitness, but tasks are only considered cooperative/competitive when the robots operate at the same time, potentially influencing each other. For such tasks robots might work in a team towards a common goal, or multiple groups might be active at the same time, potentially with distinct (sub)tasks. An example of a competitive task is a predator-prey system, where one group of robots chases and the other avoids.

3.1 Locomotion

In [Thompson, 1995], a controller made of evolvable hardware (FPGA) was evolved for locomotion and wall avoidance. This was done in "virtual reality", meaning that simulated input was sent to the real hardware, which output was fed back into the simulation.

$$f = e^{-c_1 x(t)^2} + e^{-c_2 y(t)^2} - B$$

where c_1 and c_2 are normalisation constants, $x(t)$, $y(t)$ are the horizontal and vertical distances to the center of the arena and B equals 1 if the robot is not moving. The resulting controller successfully transfered to the real robot.

[Floreano and Mondada, 1996] trained neural networks in real Khepera robots for a locomotion and obstacle avoidance task.

$$f = mean(v_l, v_r)(1 - \sqrt{(|v_l - v_r|)})(1 - s_{ir})$$

where v_l and v_r are speeds of the left and right drive motor and s_{ir} is the highest current activation level of the IR sensors. As another task in the same paper, a robot was trained to travel around the environment. It could 'recharge' at a location marked with a light. The robots were additionally outfitted with light sensors.

$$f = mean(v_l, v_r)(1 - s_i r)$$

Note that returning to the recharge station is not explicitly rewarded in the fitness function. However, since failure to return to the recharge station in time resulted in immobility for the remainder of the evaluation period, robots that did recharge could achieve a higher average speed.

[Lund and Miglino, 1996] used Khepera robots controlled by simple neural networks without hidden layers for a locomotion and object avoidance task. The initial population was evolved in a simulation for 200 generations, then transferred to real robots and run for 20 more generations.

$$f = mean(v_l, v_r)(1 - (v_l - v_r)^2)(1 - s_{ir})$$

where v_l and v_r are the left and right drive motor speeds and s_{ir} is the highest current activation level of the IR sensors. The best controllers reliably performed the task in the real environment.

[Banzhaf et al., 1997] evolved GP controllers for several tasks. Embodied evolution with 8 IR-sensors. Forward motion with object avoidance:

$$f_{(-)} = s_{ir} - (v_l + v_r - |v_l - v_r|)$$

where s_{ir} is the total of the IR-sensor activation levels. Wall following:

$$f_{(-)} = (s_{ir1} - c_1)^2 + (s_{ir2} - c_2)^2 + (s_{ir3})^2 - (v_l + v_r)^2$$

where s_{ir1}, s_{ir2} are activation levels for sensors on the wall side of the robot, s_{ir3} is the activation of the outward side sensor and c_1 and c_2 are constants. Controllers were successfully evolved for those tasks.

[Jakobi, 1998b] evolved 8-legged robots for movement and obstacle avoidance. Evolved for 3500 generations in simulation, then verified in real robots. The robots were outfitted with IR sensors on the left and right sides and a touch-sensitive bumper on the front. Each leg has its own neural net and is connected with the networks of adjacent legs.

$$f = B_1(v_l + v_r) + B_2(v_l - v_r) + B_3(-v_l + v_r) + B_4(-v_l - v_r)$$

where B_1 through B_4 are booleans and v_l and v_r are the velocities for the left and right side of the octopod. The booleans were set depending on the sensor readings. B_1 is true when no obstacles are detected by the IR sensors and the bumper is not pressed. B_2/B_3 are true when there is an object in range of the right/left sensor respectively. B_4 is true when the bumper is pressed. This effectively encodes the desired behaviour in the fitness function.

[Gomi and Ide, 1998] evolved a controller of an OCT-Ib robot (octopod) for movement. Its behaviour is determined by gait parameters for each leg. This robot comes with a large number of sensors, but only the motor current sensors and two belly contact sensors were used in the experiment.

$$F = (strides) * (1 - overcurrents) * (differences) * (1 - hits) * 1000$$

where,

$$
\begin{aligned}
strides &= \sum_{i=1}^{lifespan} \frac{liftswing_i}{lifespan} \\
liftswing_i &= \begin{cases} 1 & \text{when the leg lifts up and swings back} \\ & \text{or moves down and swings forward} \\ -1 & \text{otherwise} \end{cases} \\
overcurrents &= \sum_{i=1}^{lifespan} \sum_{j=1}^{legs} (lift\_current_{ij} + \frac{swing\_current_{ij})}{lifespan} \\
differences &= \sum_{i=1}^{lifespan} (up\_lift_i) - down\_lift_i) + (forward\_swing_i - backward\_swing_i) \\
hits &= \sum_{i=1}^{lifespan} \frac{(belly\_hit_i)}{lifespan}
\end{aligned}
$$

All runs ended in successful behaviour, the longest took 110 generations of embodied evolution.

[Matellán et al., 1998] evolved Fuzzy Logic Controllers for locomotion and object avoidance in a Khepera robot. The fitness function used was:

$$F = \frac{V(1-\sqrt{D})(1-I)}{\#rules}$$

with $I = \frac{sensor}{1023}$, $V = \frac{average}{10}$, $D = \frac{|v1-v2|}{20}$, where I is the normalised sensor value, V is the rotation average speed of the two wheels and D is the normalized absolute value of the difference between the speeds of the two wheels. This fitness function was the result of experimentation with other fitness functions that resulted in unwanted behaviours. A controller that avoids obstacles was typically found after 60 generations.

In [Nordin et al., 1998] controllers were evolved for locomotion and object avoidance of Khepera robots. Wall following was also investigated. The controllers consist of two processes, a planning process and a learning process. The learning process tries to build a model of the world, which the planning process uses to perform the task. This world model is a linear genetic program, which predicts the value of an action based on current sensor input and proposed motor actions. Fitness for a LGP is given by

$$F = \sum_{i=1}^{n} \left(f(e_i) - p(e_i)\right)^2$$

where n is the number of events, e_i denotes the i-th event, p(e_i) is the predicted value of that event by the LGP and f(e_i) is its actual value. For locomotion and object avoidance, $f(e)$ was

$$f(e) = (|m_l - m_r| + |m_l| + |m_r| - (m_l + m_r)) + \sum_{i=1}^{4} s_i$$

where m_l and m_r were the left and right motor activations and s_i the activation of the i-th sensor.

[Andersson et al., 2000] investigated embodied gait evolution in a four-legged robot dog with 8 DOF. This robot did not have input sensors, and used linear genetic programming as controller. F = Net forward travel distance. The system was reported to quickly adapt to mechanical failures.

Embodied evolution of locomotion for a Kafka robot (hexapod) was performed in [Earon et al., 2000]. Each leg was controlled by a cellular automaton, which changed state based on its own state and those of the surrounding legs. Fitness was measured by distance travelled.

In [Ziegler and Banzhaf, 2001], a controller for locomotion and object avoidance was evolved in simulation. The controller is a directed graph, a model of chemical processes. The evolved controllers were transferred to a Khepera and tested in a maze. $f_{(-)} = |v_l - v_r|$

[Wolff and Nordin, 2002] gait evolution for a humanoid robot (28 cm, 1.49 kg). This is a difficult task compared to gaits for robots with more than 2 legs. The researchers chose to perform the evolution in a real robot with visual sensors and an IR-sensor. The test environment consists of a white wall with a vertical black stripe in the center. A horizontal beam, 65 cm above the robot, supports the security cable and power cables. The robot starts about 40 cm away from the wall, facing the black stripe. Evaluation is based on distance

travelled (average speed) and straightness of path. If a robot falls over during evaluation it receives a score of 0. Speed for the best controller was 10 cm/minute.

Flapping patterns for an ornithopter were evolved in [Augustsson et al., 2002]. The ornithopter was custom built with 90 cm wings, each wing having three degrees of freedom. The controller was a list of instructions specifying target angles (one DOF per instruction), running at 20 instructions per second and repeating three times. In the first experiment, controllers were evolved for vertical movement. An individual's fitness is its average height, with some penalty for 'cheating'. The best generation did not manage to generate enough lift to carry its own weight. In the experiment for horizontal movement, the ornithopter was suspended from a rail, and fitness was measured by average forward speed.

The robot in [Marocco and Floreano, 2002] has a camera that can be tilted and panned independently. A neural network controller with recurrent connections was used to control a Koala robot (6-wheeled). The task is to move as fast as possible without colliding. Although the robot was capable of moving its camera independently, this behaviour disappeared in later generations. This might have been caused by the robot learning the layout of its (static) environment.

$$f = (v_r + v_l) - |v_r - v_l|$$

A controller for an M-Tran robot was evolved in [Yoshida et al., 2003]. The M-Tran robot is made of modules, allowing it to reconfigure itself and move in various ways. Each module has two degrees of freedom. The task was to move in a given direction. Two approaches were tested. In the first, the genotype encoded "segments", each segment consisting of motor orientations and connection information. Physically infeasible solutions were rejected. In the second approach, each module's motor had its own a neural oscillator, with input from the other modules.

$$F = c_1 * d - c_2 * w - c_3 * E/N$$

where d is the distance travelled in the given direction, w is the deviation from the straight line, E is the total energy spent and N is the number of modules. Stable movement behaviours were obtained for all of the tested configurations.

[Okura et al., 2003] Embodied evolution in a Khepera robot using FPGA controllers for movement and obstacle avoidance. The robot had 4 forward sensors. The arena was a 40 cm square with one 20 cm square object at the center.

$$F = D * (1 - S) + R * c$$

Where D is the total distance travelled, S is the total sensor input, R is the number of changes in the motor rotation and c is a constant.

In [Lund, 2003], a 2-wheeled robot had to follow a black line on the floor. To detect this line, the LEGO Mindstorms robot had two downward sensors. Notably, both the linear perceptron controller and body parameters (type of wheels, location of wheels and location of sensors) were evolved.

$$f = 4 * B_1 * max(v_l + v_r, 0) * (2 - |v_l - v_r|) + 100 * B_2 - s$$

where B_1 is a boolean that equals 1 when the robot is on the line and B_2 equals 1 when the robot entered a new segment of the line (which is made of 20 segments) in this timestep. s is a term that punishes standing still, moving backwards or rotating at maximum speed by subtracting 10 points.

An artificial neural network controller was evolved in [Nelson et al., 2004] for exploration of a maze. Each EvBOT only had 5 forward facing binary tactile sensors. The minimal information provided by the sensors means that a purely reactive controller cannot perform this task effectively, it requires some temporal behaviour. The controllers were evolved in simulation and successfully transferred to a colony of eight real robots.

$$F = c_1 * d + c_2 * dist_{net} + c_3 * dist_{max} + c_4 * stuck$$

where d is the total distance travelled (curve length), $dist_{net}$ is the distance between starting position and ending position, $dist_{max}$ is the greatest distance between start and endpoint during evaluation and $stuck$ holds whether or not the robot got stuck in the environment.

In [A. Boeing, 2004], a gait for a biped robot was evolved in simulation. The controller took the form of splines describing the servo activations over time. Only points at fixed time intervals were subject to evolution, and the derivative at those points was set at 0. Control was kept broad to help cross the reality gap. The evolved controllers successfully transferred to the real robot, but performed worse than a good handdesigned gait.

$$F = 5 * d - 50 * ave\_vel\_lowering$$

and a run stopped if the torso hit the floor (the robot toppled).

[Hornby et al., 2005] performed embodied evolution in 2 models of quadruped robots, Sony's OPEN-R and ERS-110. The task was gait evolution, using a set of gait parameters as controller. Fitness was a function of average speed and straightness of movement.

In [Van Breugel and Lipson, 2005], controllers for simulated ornithopter were evolved. The ornithopter was 2 meters long with 4 rectangular wings. The goal was to find a wing movement pattern which generated enough lift for the ornithopter to fly by its own power. Evolutionary runs were performed with sinusoidal and bezier control patterns per wing. The best sinusoidal controller was not very stable and barely able to remain airborne. The bezier controllers were more successful, though the authors note that a physical implementation of these movement patterns would probably strain existing servo motors.

$$F = z$$

where z is the final height of the ornithopter.

The authors of [Sellers and Manning, 2007] attempted to reconstruct the top running speeds of bipedal dinosaurs using ER techniques. This was done by evolving gaits for models of the dinosaurs' bodies in simulation. To validate the results, gaits were also evolved for models of extant bipeds with more or less known top speeds. The controllers were sets of gait parameters, with muscle activations for 5 points during half a gait cycle, with the gait cycle duration also subject to evolution. Fitness was measured by distance travelled over 3 or 5 seconds.

In [Furey and Harvey, 2008], a controller for a traction kite was evolved in simulation. The lines of the kite were connected to a spool coupled to a dynamo. When properly controlled, the energy gained by letting the kite reel out is greater than that spent to reel it back in, giving a net gain in energy. The spool was controlled by a discrete time recurrent neural network, fed with information about traction and line angles. The task was to maximise aerodynamic forces along the line. Fitness was estimated by simulating wind with variable amplitude and frequency and measuring the average aerodynamic forces along the lines.

In [Hutter et al., 2009], gaits were evolved for a computer model "Puppy" based on the physical "Puppy II". The controller used Central Pattern Generators (CPG) for some of the experiments, and sine-wave generators for the others. Both morphology and controller parameters were evolved in several phases. Each phase unlocked more parameters for optimisation. Simulated only, using the Webots simulation software. Evolution took place on flat terrain, with distance travelled as fitness.

3.2 Homing

In [Harvey et al., 1994], the robot is suspended from a gantry, and signals are translated to X and Y changes. The robot has to home in on a triangle, while avoiding a rectangle. The controller is a neural network. The visual inputs are subject to evolution as well, the genome encodes up to 256 visual patches that are sampled from the camera input.

[Banzhaf et al., 1997] evolved GP controllers for several tasks. Embodied evolution in a Khepera robot with 8 IR-sensors. Object homing/following:

$$f_{(-)} = (s_{ir1} + s_{ir2} + s_{ir3} + s_{ir4} - c)^2$$

where c is the ideal distance to the target and the s_{ir}'s are the activation levels of the four forward IR-sensors. Light avoiding:

$$f_{(-)} = s_{photo} - (v_l + v_r - |v_l - v_r|)$$

where s_{photo} is the activation level of a light-sensitive sensor. Controllers were successfully evolved for those tasks.

A genetic program was evolved in [sik Seok et al., 2000] for phototaxis and obstacle avoidance. Evolution took place in a wheeled robot with light sensors and ultra-sonic sensors. Fitness was updated each time step:

$$F(t+1) = [F(t) + (s_{light,max} - s_{light,fw}) * c_L + (\frac{2 * s_{ultra,fw} + s_{ultra,l} + s_{ultra,r}}{2 * s_{ultra,max}}) * k * c_U + penalty]/2$$

where c_L and c_U are the weights for the phototaxis and obstacle avoidance terms, $s_{l,max}$ and $s_{ultra,max}$ are the maximum activations of the light and ultra-sonic sensors.

In [Floreano and Urzelai, 2000], update rules for Hebbian networks are evolved, instead of the weights themselves. The network weights are randomly initialised at the beginning of an individual's life, so each individual has to learn the task during his lifetime. This favors the networks that learn their task more quickly. The task requires the robot to move to a designated area to turn on a light, then move to the light and stop there. The

initial training was done using Khepera robots, outfitted with visual sensors (for detecting the light area) and IR-sensors (for avoiding the walls). The arena was a simple rectangular area surrounded by walls, no obstacles were present. The performance of controllers with dynamic synapses was tested against controllers with the synapse weights encoded in their genome. The evolved controllers were transferred to Koala robots and proved to be robust with respect to changed sensor and motor characteristics.

In [Langdon and Nordin, 2001], inverse kinematics was learned for the right arm of the *Elvis* robot. First, a dataset was generated by moving the arm and recording the coordinates of the finger tip as seen by the two cameras, as well as the joint angles. This data was used to evolve GP controllers for each servo, with the (x,y) pairs as input and a target angle as output. Fitness was the sum of squared differences between the calculated target angles and the ones stored in the datapoints. To demonstrate the learned behaviour, the best controllers were then used to make the finger tip move towards a various locations, designed by a laserdot. In most cases, the distance between the laser and the fingertip was less than 6 cm.

[Watson et al., 2002] Phototaxis in a 130x200 arena with a light in the center. An IR beacon sends a signal that can be detected by robots that are at the light, triggering behaviour that randomises their position and restarts the task. Robots get powered by conductive strips on the floor.

[Nehmzow, 2002] Three tasks: Phototaxis, Obstacle avoidance and Robot Seeking. Small mobile robots with sonar, infrared, tactile and photosensors. There are two robots running at the same time. At the end of each run they locate each other and exchange strings and fitness information using an error-correcting protocol over the IR sensors. This requires the robots to be less than 1 meter apart. Each robot holds two strings, the current string and the best one found so far. After the transfer, crossover and mutation were applied.

In [Marocco et al., 2003], a robotic arm with 6 DOF was used. Each epoch, the arena contained one of two objects, either a sphere and a cube, placed in a random location. The arm had to touch the sphere and avoid the cube. To distinguish between the objects, the neural network controller received tactile and proprioceptive input. As part of the experiment, agents would also receive the last values of two output neurons from agents of the previous generation. This enabled the evolution of a lexicon, which allowed the agents to determine whether to avoid or home in on the object, without first having to touch the object to determine its shape.

$$f = c * -B$$

where B is the event of not touching the sphere if it exists, or touching the cube if it exists.

A controller for a Sony Aibo was evolved in [Gu et al., 2003] for ball chasing and goal homing. An Aibo uses a layered controller, with a cognition layer, a behaviour layer and a walking layer. The cognition layer extracts high level information from the sensors such as ball position (if one is detected). This is passed to a fuzzy logic controller in the behaviour layer, which then selects a predefined gait in the walking layer. The fitness function for both tasks was given by $F = (1 - distance/3000) * (1 - angle/180) * (1 - time/maximum\_time)$, where *distance* is the distance to the target, *angle* is the angle between a straight line to the target and the robots current heading, and *time* is the time spent in the behaviour that was being evolved for. All three terms were normalised in $[0, 1]$,

[Barlow et al., 2004] evolved controllers for an Unmanned Aerial Vehicle in simulation and transferred it to an EvBOT II with acoustic array (a wheeled robot). The task was to home in on a radar signal. The controllers were GP's with the amplitude and direction of the signal as input (accurate within 45 degrees). Once the robot had successfully moved within a certain range, the task shifted to circling around the goal. The environment had no obstacles, only the homing task was important.

In [Parker and Georgescu, 2005] a Cyclic Genetic Algorithm (CGA) is used to evolve a multi-loop controller for phototaxis. The simulation took place in a square arena with 5 obstacles. The light source is in the lower left corner, the robot starts in the upper right corner, facing the light. The location of the obstacles is fixed throughout a test. Fitness was given by

$$F = 2 * c_1^2 - (x^2 + y^2).$$

where (x,y) is the final location of the robot and c_1 is the length of the sides of the arena. After 350 generations, the best controller was transferred into a lego robot dubbed "Amsterdam". This robot was equipped with 2 photosensors and a forward tactile sensor. The real arena was an 8 feet square with wooden walls and 1 foot obstacles. 5 tests were performed for each of 3 configurations of obstacles. In all cases the robot successfully reached the light source.

[Capi and Doya, 2005] uses the Cyber Rodent robot. The task is to visit sites for food, water and nests in sequence. The locations of food and water are swapped twice during the run, at 1/3 and 2/3 of the total evaluation time. This requires the robot to have some memory. The controller is a locally connected neural

network with memory nodes. The memory nodes have self-excitation and lateral inhibition connections. The fitness is only updated when the robot arrives at a landmark.

$$\Delta F = \begin{cases} +1 & \text{if the landmark is the next in the sequence} \\ -1 & \text{otherwise} \end{cases}$$

The controllers evolved in simulation were transferred to a real Cyber Rodent.

In [Togelius and Lucas, 2006], standard RC cars were evolved to race in eight simulated tracks. The tracks were delimited by walls, and the cars were outfitted with distance sensors. The angle to the next waypoint was also sent to the neural controllers. In one experimental setup, controllers were trained on all tracks at once and in another, they were trained incrementally from easy to harder tracks. The effect of evolvable sensor angle and position was also investigated. The resulting controllers could competently navigate the tracks they were evolved on, although they performed poorly on the other tracks.

In [Peniak et al., 2009] a simulated approximation of the Mars Rover (a 6 wheeled robot, approximately 290x270x220 cm, with 18 IR sensors) had to move and avoid obstacles in an unknown rough terrain. It was controlled by a fully connected feed forward neural network with no hidden layers. Training took place on three terrains. The first terrain was 60x60 meters, with inclining and declining slopes, holes, flat and rough terrain and small and large obstacles. The second terrain had the same obstacles as the first, but with all rough terrain. The third terrain had more obstacles than the first.

$$f = \frac{S_t * S_p}{S * T}$$

where T is the number of trials and S is the number of timesteps in a trial. S_p maps speed to [0,1], where 1 is maximum speed and 0 is no movement or backwards movement. S_t maps steering angle to [0,1], where 1 is straight forward and 0 is an absolute angle of 30 degrees or more. To evaluate the degree of exploration, the terrains were subdivided into 3x3m squares, with the measure being the fraction of passable terrain visited by the robot. Exploration test function:

$$E = \frac{S_{visited}}{S_{total} - S_{obstacles}}$$

Where S is the number of squares. Each controller was evaluated for 10,000 timesteps for 100 random starting positions on all three terrains.

Neural network controllers for Micro Unmanned Aerial Vehicles (MAV) were evolved in [Ruini and Cangelosi, 2009]. The authors used a combination of Multi-Agent Systems and Evolutionary Robotics. The MAVs are airplane-like, this means that course adjustments have strict time requirements compared to hovering helicopters. The task is goal homing and obstacle avoidance in simulated 2d and 3d environments. The 2d arena is a map of canary wharf, where the tallest buildings act as obstacles. The robots have to locate a target and detonate while nearby. A robot is 'destroyed' when it detonates, collides with a building or another UAV, runs out of energy or leaves a predefined area. A MAV moves at a fixed speed of 3 Graphical units (GU) per timestep, in the direction it is facing. The first task was to search and destroy a stationary target. It is assumed that the MAVs know the location of the target (for example by satelite tracking). MAVs work in teams of 4. Four trials are performed. The corresponding fitness function was:

$$F = -\alpha + (\beta/50) + (\sigma * 50) + (\epsilon * 10)$$

Where α is the average distance between the target and the MAV that detoned closest to it. β is the average amount of energy for the MAV that detonated closest to the target. σ is the number of tests that ended in success (target was destroyed) ϵ is the total number of MAVs still alive after the four tests. A harder task involves a target that actively moves away from the MAVs. The authors experimented with various escape speeds, expressed as a fraction of the MAVs speed. Speeds above 1/3 led to decreased fitness, compared to the stationary target. The next setup made the target require several hits. The MAVs now got additional information about the status of the target (damaged/undamaged) and whether or not another MAV was within 30 pixels.

$$F = (\gamma * \frac{c}{4}) + (\eta * \frac{c}{2}) + (\lambda * c) + (\epsilon * 10) + (\frac{\beta}{50})$$

Where γ is the number of tests where a MAV detonated within 64 px from the target. η is the number of tests where at least one MAV damaged the target. λ is the number of tests completed successfully (target destroyed). ϵ is the number of MAVs still alive after the four tests, β is the average amount of energy retained by the MAVs that eventually detonated near the target and c is a constant. Finally, the model was extended to 3d simulation,

which adds two more degrees of freedom to the MAV's. The environment is now a cuboid, 1000x750x650 GU (graphical units) in size. The obstacles were removed, but a MAV that leaves the area is considered 'lost'.

$$F = -\alpha + \beta$$

α is the average distance of the MAV that detonated closest to the target and β is the average amount of energy left for the MAV that detonated closest.

[Hartland et al., 2009] compares the performance of Multi Layer Perceptrons and Echo State Networks. Simulated evolution using Khepera II robots with 8 proximity IR sensors. The task is known as the Tolman maze, the robot needs to reach the endpoint of the third branch. In absence of rich sensors, efficiently performing this task requires memory. One fitness function with a large amount of apriori information:

$$F_{(-)} = (min_{t=1..T}||x(t) - z^*||^2 + 1) * (min_{t=1..T}||x(t) - x^*||^2)$$

where $x(t)$ is the location of the robot at time t, x^* is the goal location and z^* is the location of the entry point of the third branch. Another fitness function with less apriori information:

$$F_{(-)} = min_{t=1..T}||x(t) - x^*||^2$$

This function has a local minimum at the end of the second branch.

In [Moioli et al., 2010], the robot is located at the bottom of the arena facing upward and can only move horizontally. In their first experiment, objects are created at the top, moving downwards, the robot needs to move towards circles and avoid squares. A run consists of 20 trials. The task of the second experiment is the same as the first, with the addition of 20 trials with inverted input. The controller, a network of oscillators, gets input from 7 ray sensors, which are distributed with equal angles to give a 60 degree upward view angle. Performed in simulation only.

3.3 Object Manipulation

A Khepera robot with neural network controller was evolved for peg collection and goal homing in [Nolfi and Parisi, 1995]. The arena was a 60x35 cm rectangle, surrounded by walls. The object to be retrieved was a 3 cm high cardboard cylinder, covered in white paper. The robot had to avoid the walls until it had picked up the peg, then deposit the peg outside the arena.

[Schultz et al., 1996] A controller for a shepherd robot is evolved. The shepherd robot has to lead a sheep robot (with fixed controller) to a goal location without colliding with any obstacles. The shepherd controller is a collection of stimulus-response rules, using the SAMUEL rule learning system.

In [po Lee et al., 1997] a Khepera robot with infra red and ambient light sensors was trained to push a box towards a goal, indicated by a light source. The arena consisted of the box and the lightsource, with no further obstacles. The controllers were Genetic Programs, the island model was used to maintain diversity.

In [Pok and Keung, 1999] a GP controller was evolved for a small wheeled robot with two arms and two forward eyes. The robot had to push a stationary object out of an otherwise empty area, 42x65cm in size. The experiments were run for objects of various shapes and weights.

In [Sprinkhuizen-kuyper et al., 2000], a neural network controller for a Khepera robot with IR sensors was evolved in simulation. The neural network was a fully connected single layer perceptron. The task involved pushing a circular box to a light area, in a square arena with two walls as extra obstacles. The authors investigated two different viewpoints on fitness assessment. Global versus local is the difference between calculating the fitness after the run and integrating at each timestep. Internal versus external relates to the source of information, which can be internal (only using information available to the robot through its sensors), or external (for example from a bird-eye camera). The combinations gave rise to four fitness functions:

Global external (GE)

$$F = d(B_T, B_0) - \frac{1}{2}d(B_T, R_T)$$

Local external (LE)

$$f = d(B_t, B_{t-1}) + \frac{1}{2}[d(B_{t-1}, R_{t-1}) - d(B_t, R_t)]$$

Global internal (GI)

$$F = \frac{1}{2 \times 1024}(S_{d2} + S_{d3}) + (1 - \frac{1}{4 \times 500}\sum_{i=1}^{4} s_{photo_i})$$

Local internal (LI)

$$f = \frac{1}{2 \times 1024}(s_{d2} + s_{d3}) + \frac{1}{2 \times 20}(M[L] + M[R]) - \frac{1}{3 \times 20}\,|M[L] - M[R]|$$

Out of those four fitness functions, GE functioned the best. The evolved controllers from that fitness function successfully transferred to a real Khepera.

The authors of [Kamio and Iba, 2005] integrated Genetic Programming and Reinforcement Learning. First the Genetic Programs were evolved in simulation for a box-pushing task. The GP used terms like move-forward from an action set. After simulation, the best controller was transferred to a real robot. The effects of these actions were tweaked in the real robots using Q-learning. The experiments were performed using an Aibo and a HAOP-1 as physical robots.

A neural network controller for a quadruped robot with a front gripper is evolved in simulation in [Bongard, 2008]. The robot needs to move towards an object and place that object on its back.

$$F = \begin{cases} max_{k=1}^{t} s_{leftclawtip}(k) * s_{rightclawtip}(k) & \text{if both claws never touched the object simultaneously} \\ 1 + max_{k=1}^{t} s_{backsensor}(k) & \text{otherwise} \end{cases}$$

where $s_{component}(k)$ is the activation of the distance sensor (closer is higher) at time k.

While the controller is successful, the distance between object and starting position is increased.

[Mouret and Doncieux, 2009] used an electric circuit with lights and buttons that turn additional lights on, training a robot to turn a specific light on.

$$F(x) = min_{n=1,2,3}(-\frac{\phi(n,i)}{T})$$

where $\phi(n,i)$ is the time needed to switch on the target light in experiment n and T is the duration of an experiment. [Mouret and Doncieux, 2008] explores the same task, but using this fitness function for each light, using multi-objective optimisation.

3.4 Cooperative and competitive tasks

[Ciesielski et al., 2002] used strongly typed genetic programming to evolve controllers for robot soccer. Each team consists of 11 players sharing the same controller. The first experiment only used low-level functions (kick, turn, dash). The teams of each generation competed in an elimination tournament, where fitness depended on the round in which the team was eliminated. Most individuals from experiment 1 failed to score any goals or even kick a ball. The second experiment replaced the set of non-terminals by higher-level functions (kickTo, turnTo, moveTo). Fitness was a weighted sum of goals scored and kicks made. The best controllers of experiment 2 were able to play a basic game of soccer, though they tended to all move towards the ball, resulting in 'ball smothering'. Experiment 3 added more low-level functions and terminals to the set from experiment 1. This commonly resulted in behaviour where robots would circle the ball and kick it in a random direction, allowing it to score goals but scoring own goals as well.

[Quinn et al., 2002] A group of 3 robots had to move some distance away from the starting point, avoid collisions and stay in sensor range of each other (move in formation). Wheeled robots, outfitted with 2 front and 2 back IR sensor/emitters. All three robots used the same artificial network controller.

$$F = P * \sum_{t=1}^{T} f(d_t, D_{t-1}) * (1 + tanh(s_t/20))$$

Where P is a term in $[0.5, 1]$ penalising collisions, $P = 1 - c/2c_{max}$ with c the number of collisions during this trial and c_{max} the maximum (20). A function to reward distance, $f(d_t, D_{t-1}) = max(min(d_t, D_{max}) - D_{t-1}, 0)$, with D_{max} the desired distance from the starting point, d_t the distance at time t and D_{t-1} the greatest distance obtained before time t.

Evolved in simulation, transferred to real robots.

[Trianni and Dorigo, 2006] used a group of s-bots for movement in a straight line, avoiding holes in the environment. The robots were outfitted with ground sensors and traction sensors. For the setups with signalling, a microphone and loudspeakers were added. Because the ground sensors were positioned below the robot, an individual robot could not detect a hole without falling in, so this task required cooperation. A single neural network controller was evolved that was used in all robots. Three setups were tested, one without signalling,

one with handwritten signalling and one with evolved signalling. The setup with evolved signalling performed significantly better than the other two.

For each s-bot s belong to the swarm-bot S the fitness at cycle t is computed.

$$f_s(t) = \omega_s(t) * \Delta\omega_s(t) * \gamma_s(t)$$

where $\omega_s(t)$ is a term evaluating fast motion:

$$\omega_s(t) = (|\omega_s, l(t)| + |\omega_s, r(t)|)/(2 * \omega_M)$$

where $\omega_{s,l}(t)$ and $\omega_{s,r}(t)$ are the angular velocities of the left and right wheel of s-bot s at cycle t, and ω_M is the maximum attainable angular velocity. where $\Delta\omega_s(t)$ is a term evaluating straightness of movement:

$$\Delta\omega_s(t) = \begin{cases} 1 - \sqrt{(}|\omega_s, l(t) - \omega_s, r(t)|/\omega_M) & \text{if } \omega_{s,l}(t) * \omega_{s,r}(t) >= 0 \\ 0 & otherwise \end{cases}$$

and where $\gamma_s(t)$ accounts for coordination and hole avoidance: $\gamma_s(t) = 1 - max(F_s(t), G_s(t), S_s(t))$ where $F_s(t)$ is the intensity of the traction force perceived by bot s at time t, $G_s(t)$ is the maximum activation of the ground sensors and $S_s(t)$ is a binary value denoting whether or not bot s was emitting a tone at time t. The fitness of the trial is given by: $\mathrm{F} = 1/T \sum_{t=1}^{T} \min_s f_s(t)$, or 0 if fallen.

[Nelson and Grant, 2007] made two robots play games of Capture the Flag against each other. This task requires them to be able to efficiently navigate in an environment. The fitness function had two modes: the bootstrap mode to overcome the problem of subminimality, and the aggregate mode that was based on the outcome of the matches.

$$F = F_{mode\_1} XOR\ F_{mode\_2},$$

where XOR is an operator that returns F_{mode_2} if F_{mode_2} is above zero and F_{mode_1} otherwise.

$$F_{mode\_1} = F_{dist} - stuck - m$$

where F_{dist} is proportional to the difference of the distance d travelled by the robot and half the size of the biggest dimension of the training environment D. *stuck* and m are constants that penalise getting stuck and producing output that exceeds the range of the actuators respectively. Each robot of the current generation played 2 matches against a robot from the previous generation.

$$F_{mode\_2} = 1.5 * wins - 0.5 * draws - 1 * losses.$$

The authors report that the evolved controllers performed comparable to a knowledge based handcoded controller.

In [Baldassarre et al., 2007], a group of 4 s-bots with identical neural controllers was evolved for coordinated movement. The s-bots started connected to eachother in a linear shape, but with a randomised direction of their chassis. To perform well at the task they needed to align their individual direction and move their center of mass away from the starting point. Evolved in simulation and tested extensively both in simulation and in real robots. These tests included moving on rough terrain, increasing the number of robots, changing shape of the swarm-bot and linking the s-bots to a passive object. The controller performed successfully in all setups, even though it was not evolved explicitely for these situations. The authors also note the complexity of the group behaviour, despite the simplicity of the single-layered neural controller.

In [Groβ and Dorigo, 2008], controllers for s-bots were evolved in simulation to push an object weighing 250 or 500 grams. Each individual in the group used the same neural network controller. Fitness was calculated by the distance between the final position of the object and its starting position. The 500 gram object was too heavy to be moved by a single robot. While the robots did not have sensors to detect each other directly, they still managed to coordinate by measuring the forces applied on the object. This resulted in behaviour where robots glided along the edge of the object until the pushing directions were more or less aligned.

[Froese and Di Paolo, 2008] simulated two robots that can only move horizontally. They are 40 units wide with an on/off sensor in the middle, which triggers when their centers are less than 20 units apart. Both robots use the same fully connected neural network controller with self connected links. At the beginning of each run, random relative starting positions are given to the robots. The task is to move as far as possible while keeping sight of the other. Because the robots face each other and use the same controller this task is non-trivial, requiring coordination.

[Tuci et al., 2008] trained two s-bots to connect to each other using a gripper, while minimising collisions between the two.

$$F_e = A_e * C_e * S_e$$

with

$$A_e = \begin{cases} 1.0/(1.0 + atan(\frac{d_{rr}-16}{16})) & \text{if } d_{rr} > 16 \text{ cm} \\ 1.0 & \text{otherwise} \end{cases}$$

where d_{rr} is the distance between bots.

$$C_e = \begin{cases} 1.0 & \text{if } n_c = 0 \\ 0.0 & \text{if } n_c > 20 \\ 1.0/(0.5 + \sqrt{n_c}) & \text{otherwise} \end{cases}$$

n_c = the number of collisions recorded during training.

$$S_e = \begin{cases} 100.0 & \text{if } GG(T) = 1.0, \text{ for any robot} \\ 1.0 + 29.0 * sum_{t=0}^{T} K(t)/T & \text{otherwise} \end{cases}$$

$K(t) = 1$ if the sensor GS of any bot is active at time t, 0 otherwise. The fitness is the average over 40 runs with random initial orientations.

In [Bell et al., 2009], robots were coevolved in simulation to communicate with each other. A seeing robot leads a blind robot to a light. The robot gains a point if it moves nearer to the light, and loses one if it moves away. If the robot reaches the light, the light is moved to a random location and the robot gains 1000 points. A trial lasted for 200 time steps. Both robots used neural network controllers, with NEAT as evolutionary algorithm.

3.5 Implicit

In recent years, much research has been devoted to implicit fitness functions. One of the projects that focus on autonomous, localized evolution and implicit pressure in robot swarms is SYMBRION [Baele et al., 2009]. With implicit fitness, the inherent task could be considered to be survival and reproduction. Despite the absence of an explicit fitness function, it is still possible to evolve controllers for specific tasks. An example is [Bird et al., 2008], where a robot gains energy by drawing lines over food circles on the floor (invisible to the robot). Here energy essentially corresponds to accumulated fitness.

4 Conclusions

I have given an overview of the common tasks in ER. For added structure, I have proposed to divide them into fundamental categories: movement, homing and object manipulation. I have still only touched a fraction of all published works in ER, but for this section I will assume that my sample is at least representative of the complete body of work.

In most of the literature, controllers were evolved successfully. This typically happened in 20 - 300 generations, although some runs took thousands of generations. Sometimes performance of the evolved controllers was compared to that of a hand-crafted controller. When evolving in a physical robot, population sizes were kept small (below 100), due to time constraints. Many researchers use a single physical robot, evaluating one controller at a time. The speed benefits of simulation allow for larger population sizes. The cost of computing power has continued to decrease in the past decades. Recently, the focus has moved away from increasing clockspeeds, and towards increasing the number of processors. Since simulation lends itself to parallelisation quite well, it is expected that the speed benefits of simulation will continue to increase. Embodied evolution also benefits, with the increasing speed allowing richer sensor input and more complex controllers.

4.1 Tasks

Research in locomotion tasks was very common. This may be because it is the most basic task for a mobile robot to perform and a requirement for the more complex tasks. Locomotion is almost always combined with obstacle avoidance. The rarity of locomotion for winged robots can be explained by a lack of winged robotics platforms, as well as the computational cost of simulating flight. To minimise the risk of damage from crashes, embodied flight evolution has always taken place in a carefully constructed setup.

Category	Task	Popularity
Movement	Locomotion	Very common
	Gait Evolution	Common
	Flight	Rare
	Path following	Rare
	Wall following	Common
Homing	Goal Homing	Common
	Object Homing	Common
	Phototaxis	Common
	Inverse Kinematics	Rare
Object Manipulation	Object Pushing	Common
	Gathering	Rare
	Herding	Rare

Table 1: Relative popularity of tasks

Homing is another fundamental aspect of robotics. In order to perform useful work, robots need to be able to find and navigate towards objects and areas of interest. In most literature, a target area is indicated using light or sound. Sometimes homing is towards an object instead. The arena can contain multiple objects, some of which need to be avoided and others homed in on. This requires the robot to evolve the ability to discriminate between objects. Objects may differ in shape [Moioli et al., 2010], though the number of shapes is typically restricted to two.

Manipulating objects is an essential part of many real-world tasks. In the literature, the mode of interaction is typically restricted to pushing an object, while a few robots are equipped with a gripper that can grab and/or lift the object. Destinations for the object have included 'away from the start location', 'outside the arena', or 'into a lighted area'. Herding can be seen as object manipulation with an active object, normally another robot with a fixed controller.

For evolutionary robotics to advance beyond 'toy tasks', researchers should seek new challenges to overcome. Current experiments often take place in a simplified environment. While this does not prevent the emergence of robust controllers [Baldassarre et al., 2007], it could be worthwhile to position experiments in more realistic settings. I think ER is at a stage where more practical applications can be investigated. For example, I think that controllers can be successfully evolved for a task like litter collection, which is an extension of foraging with objects of varying shapes and sizes.

4.2 Robots

One thing to note is that in most of the research the morphology of the robot is fixed. There is evidence that having the morphology change can improve the quality of the controllers, as well as the speed of evolution. [Bongard, 2011]. A wide variety of sensors are used. Some robots are equipped with cameras, but the rich input from the camera is usually preprocessed before it is passed to the controller. This preprocessing step reduces the dimensionality of the input and thus the search space. The parameters for the method of preprocessing can be encoded in the genome. The choice of sensors also influences the difficulty of a task. With low bandwidth sensors, controllers may need to rely on memory and integrating information over time. The s-bot is a popular platform for multi-robot research. It has been used in several experiments involving coordinated movement and using its ability to physically connect to other s-bots.

4.3 Controllers

More than half of the discussed literature from the last decade used some form of neural network as controller. The popularity of neural controllers can be ascribed to their robustness to noise and expressiveness. Neural nets, unlike for example gait parameters, also have the benefit of being applicable to many different tasks. With the use of algorithms like NEAT, controllers can be evolved with the right amount of complexity for a given task, leading to a succesful controller for capture the flag [Nelson and Grant, 2007] with 100 neurons and some 5000 connections. Most successful neural controllers are much simpler though, using less than 15 hidden nodes.

References

[A. Boeing, 2004] A. Boeing, S. Hanham, T. B. (2004). Evolving autonomous biped control from simulation to reality. In *Proceedings of the 2nd International Conference on Autonomous Robots and Agents*, pages 440–445.

[Andersson et al., 2000] Andersson, B., Svensson, P., Nordahl, M., and Nordin, P. (2000). On-line evolution of control for a four-legged robot using genetic programming. In Cagnoni, S., editor, *Real-World Applications of Evolutionary Computing*, volume 1803 of *Lecture Notes in Computer Science*, pages 322–329. Springer Berlin / Heidelberg.

[Augustsson et al., 2002] Augustsson, P., Wolff, K., and Nordin, P. (2002). Creation of a learning, flying robot by means of evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002), volume*, pages 1279–1285. Citeseer.

[Baele et al., 2009] Baele, G., Bredeche, N., Haasdijk, E., Maere, S., Michiels, N., de Peer, Y. V., Schmickl, T., Schwarzer, C., and Thenius, R. (2009). Open-ended on-board evolutionary robotics for robot swarms. In *IEEE Congress on Evolutionary Computation*, pages 1123–1130. IEEE.

[Baldassarre et al., 2007] Baldassarre, G., Trianni, V., Bonani, M., Mondada, F., Dorigo, M., and Nolfi, S. (2007). Self-organised coordinated motion in groups of physically connected robots. *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, 37(1):224–239.

[Banzhaf et al., 1997] Banzhaf, W., Nordin, P., and Olmer, M. (1997). Generating adaptive behavior using function regression within genetic programming and a real robot. In *In 2nd International Conference on Genetic Programming*, pages 35–43. Morgan Kaufmann.

[Barlow et al., 2004] Barlow, G. J., Mattos, L. S., and Grant, E. (2004). Transference of evolved unmanned aerial vehicle controllers to a wheeled mobile robot. In *In Proceedings of the IEEE International Conference on Robotics*, pages 704–720. MIT Press.

[Bell et al., 2009] Bell, A., Pace, A., and Santos, R. (2009). Evolutions of communication.

[Bianco and Nolfi, 2004] Bianco, R. and Nolfi, S. (2004). Evolving the neural controller for a robotic arm able to grasp objects on the basis of tactile sensors. *Adaptive Behavior*, 12(1):37.

[Bird et al., 2008] Bird, J., Husbands, P., Perris, M., Bigge, B., and Brown, P. (2008). Implicit fitness functions for evolving a drawing robot. In *Evo'08: Proceedings of the 2008 conference on Applications of evolutionary computing*, pages 473–478, Berlin, Heidelberg. Springer-Verlag.

[Bongard, 2008] Bongard, J. (2008). Behavior chaining: Incremental behavior integration for evolutionary robotics. *Artificial Life*, 11:64.

[Bongard, 2011] Bongard, J. (2011). Morphological change in machines accelerates the evolution of robust behavior. *Proceedings of the National Academy of Sciences*, 108(4):1234–1239.

[Capi and Doya, 2005] Capi, G. and Doya, K. (2005). Evolution of recurrent neural controllers using an extended parallel genetic algorithm. *Robotics and Autonomous Systems*, 52(2-3):148 – 159.

[Ciesielski et al., 2002] Ciesielski, V., Mawhinney, D., and Wilson, P. (2002). Genetic programming for robot soccer. In *RoboCup 2001: Robot Soccer World Cup V*, pages 319–324. Springer-Verlag.

[Earon et al., 2000] Earon, E., Barfoot, T., and Deleuterio, G. (2000). From the sea to the sidewalk: The evolution of hexapod walking gaits by a genetic algorithm. In Miller, J., Thompson, A., Thomson, P., and Fogarty, T., editors, *Evolvable Systems: From Biology to Hardware*, volume 1801 of *Lecture Notes in Computer Science*, pages 51–60. Springer Berlin / Heidelberg.

[Floreano and Mondada, 1996] Floreano, D. and Mondada, F. (1996). Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics*, pages 396–407.

[Floreano and Urzelai, 2000] Floreano, D. and Urzelai, J. (2000). Evolutionary robots with on-line self-organization and behavioral fitness. *Neural Networks*, 13:431–443.

[Froese and Di Paolo, 2008] Froese, T. and Di Paolo, E. (2008). Stability of coordination requires mutuality of interaction in a model of embodied agents. *From Animals to Animats 10*, pages 52–61.

[Furey and Harvey, 2008] Furey, A. and Harvey, I. (2008). Robust adaptive control for kite wind energy using evolutionary robotics. *Proc. Biological Approaches for Engineering*.

[Gomi and Ide, 1998] Gomi, T. and Ide, K. (1998). Evolution of gaits of a legged robot. In *The 1998 IEEE International Conference on Fuzzy Systems, Proceedings of the 1998 IEEE World Congress on Computational Intelligence*, volume 1, pages 159–164.

[Groβ and Dorigo, 2008] Groβ, R. and Dorigo, M. (2008). Evolution of solitary and group transport behaviors for autonomous robots capable of self-assembling. *Adaptive Behavior*, 16(5):285.

[Gu et al., 2003] Gu, D., Hu, H., Reynolds, J., and Tsang, E. (2003). Ga-based learning in behaviour based robotics. In *in Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 16–20.

[Hartland et al., 2009] Hartland, C., Bredeche, N., and Sebag, M. (2009). Memory-enhanced evolutionary robotics: the echo state network approach. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 2788–2795. IEEE.

[Harvey et al., 1994] Harvey, I., Husbands, P., and Cliff, D. (1994). Seeing the light: artificial evolution, real vision. In *Proceedings of the third international conference on Simulation of adaptive behavior : from animals to animats 3: from animals to animats 3*, pages 392–401, Cambridge, MA, USA. MIT Press.

[Hornby et al., 2005] Hornby, G. S., Takamura, S., Yamamoto, T., and Fujita, M. (2005). Autonomous evolution of dynamic gaits with two quadruped robots. *IEEE Transactions on Robotics*, 21:402–410.

[Hutter et al., 2009] Hutter, S., Hoffmann, M., and Pfeifer, R. (2009). Co-evolution of Morphology and Controller of a Simulated Underactuated Quadruped Robot using Evolutionary Algorithms.

[Jakobi, 1998a] Jakobi, N. (1998a). Minimal simulations for evolutionary robotics. Technical report, University of Sussex.

[Jakobi, 1998b] Jakobi, N. (1998b). Running across the reality gap: Octopod locomotion evolved in a minimal simulation. In *Proceedings of the First European Workshop on Evolutionary Robotics: EvoRobot98*, pages 39–58. Springer-Verlag.

[Kamio and Iba, 2005] Kamio, S. and Iba, H. (2005). Adaptation technique for integrating genetic programming and reinforcement learning for real robots. *IEEE Transactions on Evolutionary Computation*, 9(3):318–333.

[Langdon and Nordin, 2001] Langdon, W. and Nordin, P. (2001). Evolving hand-eye coordination for a humanoid robot with machine code genetic programming. *Genetic Programming*, pages 313–324.

[Lund, 2003] Lund, H. (2003). Co-evolving control and morphology with LEGO Robots. *Morpho-functional machines: the new species: designing embodied intelligence*, page 59.

[Lund and Miglino, 1996] Lund, H. H. and Miglino, O. (1996). From simulated to real robots. In *International Conference on Evolutionary Computation*, pages 362–365.

[Marocco et al., 2003] Marocco, D., Cangelosi, A., and Nolfi, S. (2003). The emergence of communication in evolutionary robots. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 361(1811):2397.

[Marocco and Floreano, 2002] Marocco, D. and Floreano, D. (2002). Active vision and feature selection in evolutionary behavioral systems. In *From Animals to Animats 7: Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior*, pages 247–255. MIT Press.

[Matellán et al., 1998] Matellán, V., Fernandez, C., and Molina, J. (1998). Genetic learning of fuzzy reactive controllers. *Robotics and Autonomous Systems*, 25(1-2):33–41.

[Moioli et al., 2010] Moioli, R., Vargas, P., and Husbands, P. (2010). Exploring the Kuramoto model of coupled oscillators in minimally cognitive evolutionary robotics tasks. *IEEE Cong. Evol. Comput.*

[Mondada et al., 2009] Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J., Floreano, D., and Martinoli, A. (2009). The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th conference on autonomous robot systems and competitions*, volume 1, pages 59–65.

[Mondada et al., 2004] Mondada, F., Pettinaro, G. C., Guignard, A., Kwee, I., Floreano, D., Deneubourg, J.-L., Nolfi, S., Gambardella, L., and Dorigo, M. (2004). SWARM-BOT: a New Distributed Robotic Concept. *Autonomous Robots, special Issue on Swarm Robotics*, 17(2-3):193–221.

[Mouret and Doncieux, 2008] Mouret, J.-B. and Doncieux, S. (2008). Incremental evolution of animats' behaviors as a multi-objective optimization. In *SAB '08: Proceedings of the 10th international conference on Simulation of Adaptive Behavior*, pages 210–219, Berlin, Heidelberg. Springer-Verlag.

[Mouret and Doncieux, 2009] Mouret, J.-B. and Doncieux, S. (2009). Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity. In *Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, CEC'09, pages 1161–1168, Piscataway, NJ, USA. IEEE Press.

[Nehmzow, 2002] Nehmzow, U. (2002). Physically embedded genetic algorithm learning in multi-robotscenarios: The pega algorithm. In *International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*.

[Nelson et al., 2009] Nelson, A. L., Barlow, G. J., and Doitsidis, L. (2009). Fitness functions in evolutionary robotics: A survey and analysis. *Robot. Auton. Syst.*, 57(4):345–370.

[Nelson and Grant, 2007] Nelson, A. L. and Grant, E. (2007). Aggregate selection in evolutionary robotics. In *Mourelle (Eds.), Mobile Robots: The Evolutionary Approach, in: Studies in Computational Intelligence*, pages 63–88. Springer.

[Nelson et al., 2004] Nelson, A. L., Grant, E., Galeotti, J., and Rhody, S. (2004). Maze exploration behaviors using an integrated evolutionary robotics environment. In *Journal of Robotics and Autonomous Systems, 2003*, pages 159–173.

[Nolfi and Parisi, 1995] Nolfi, S. and Parisi, D. (1995). Evolving non-trivial behaviors on real robots: an autonomous robot that picks up objects. In *ROBOTICS AND AUTONOMOUS SYSTEMS*, pages 187–198. Springer Verlag.

[Nordin and Nordahl, 1999] Nordin, J. and Nordahl, M. (1999). An evolutionary architecture for a humanoid robot. In *Proceeding of: The Fourth International Symposium on Artificial Life and Robotics (AROB 4th 99) Oita Japan*. Citeseer.

[Nordin et al., 1998] Nordin, P., Banzhaf, W., and Brameier, M. (1998). Evolution of a world model for a miniature robot using genetic programming. *Robotics and Autonomous Systems*, 25(1-2):105–116.

[Okura et al., 2003] Okura, M., Matsumoto, A., Ikeda, H., and Murase, K. (2003). Artificial evolution of fpga that controls a miniature mobile robot khepera. In *Proceedings of the Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE 2003)*, volume 3, pages 2858–2863.

[Parker and Georgescu, 2005] Parker, G. and Georgescu, R. (2005). Using cyclic genetic algorithms to evolve multi-loop control programs.

[Pasemann et al., 2001] Pasemann, F., Pasemann, F., Steinmetz, U., Steinmetz, U., Hlse, M., Hulse, M., Lara, B., and Lara, B. (2001). Evolving brain structures for robot control. In *IWANN'01 Proceedings LNCS, 2085*, pages 13–15. Springer-Verlag.

[Peniak et al., 2009] Peniak, M., Marocco, D., and Cangelosi, A. (2009). Co-evolving controller and sensing abilities in a simulated Mars Rover explorer. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 2772–2779. IEEE.

[po Lee et al., 1997] po Lee, W., Hallam, J., Lund, H. H., and K, E. U. (1997). Applying genetic programming to evolve behavior primitives and arbitrators for mobile robots. In *In Proceedings of IEEE 4th International Conference on Evolutionary Computation*, pages 495–499. IEEE Press.

[Pok and Keung, 1999] Pok, J. and Keung, C. (1999). Learning coordinated maneuvers in complex environments: a sumo experiment. In *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 99*, volume 1, pages 343–349.

[Quinn et al., 2002] Quinn, M., Smith, L., Mayley, G., and Husbands, P. (2002). Evolving team behaviour for real robots. In *North Carolina State University*, pages 14–16.

[Ruini and Cangelosi, 2009] Ruini, F. and Cangelosi, A. (2009). Extending the Evolutionary Robotics approach to flying machines: An application to MAV teams. *Neural Networks*, 22(5-6):812–821.

[Schultz et al., 1996] Schultz, A. C., Grefenstette, J. J., and Adams, W. (1996). Roboshepherd: Learning a complex behavior. In *Proceedings of the FLAIRS'96*, volume 6, pages 763–768.

[Sellers and Manning, 2007] Sellers, W. and Manning, P. (2007). Estimating dinosaur maximum running speeds using evolutionary robotics. *Proceedings of the Royal Society B*, 274(1626):2711.

[sik Seok et al., 2000] sik Seok, H., ju Lee, K., gun Joung, J., and tak Zhang, B. (2000). An on-line learning method for object-locating robots using genetic programming on evolvable hardware. In *Proceedings of the Fifth International Symposium on Artificial Life and Robotics, AROB'00*, volume 1, pages 321–324.

[Sprinkhuizen-kuyper et al., 2000] Sprinkhuizen-kuyper, I. G., Kortmann, R., and Postma, E. O. (2000). Fitness functions for evolving box-pushing behaviour. In *Proceedings of the Twelfth Belgium-Netherlands Artificial Intelligence Conference*, pages 275–282.

[Thompson, 1995] Thompson, A. (1995). Evolving electronic robot controllers that exploit hardware resources. *Advances in Artificial Life*, pages 640–656.

[Togelius and Lucas, 2006] Togelius, J. and Lucas, S. (2006). Evolving robust and specialized car racing skills. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 1187–1194. IEEE.

[Trianni and Dorigo, 2006] Trianni, V. and Dorigo, M. (2006). Self-organisation and communication in groups of simulated and physical robots. *Biological Cybernetics*, 95(3):213–231.

[Tuci et al., 2008] Tuci, E., Ampatzis, C., Trianni, V., Christensen, L., Dorigo, M., Bruxelles, U. L. D., Tuci, E., Ampatzis, C., Trianni, V., Christensen, A. L., and Dorigo, M. (2008). Self assembly in physical autonomous robots: the evolutionary robotics approach.

[Van Breugel and Lipson, 2005] Van Breugel, F. and Lipson, H. (2005). Evolving buildable flapping ornithopters. *GECCO 2005*.

[Watson et al., 2002] Watson, R. A., Ficici, S. G., and Pollack, J. B. (2002). Embodied evolution: Distributing an evolutionary algorithm in a population of robots. *Robotics and Autonomous Systems*, 39:1–18.

[Wolff and Nordin, 2002] Wolff, K. and Nordin, P. (2002). Evolution of efficient gait with an autonomous biped robot using visual feedback. In *Proceedings of the Mechatronics Conference. University of Twente*, pages 482–489. Morgan Kaufmann Publishers, Inc.

[Yoshida et al., 2003] Yoshida, E., Murata, S., Kamimura, A., Tomita, K., Kurokawa, H., and Kokaji, S. (2003). Evolutionary synthesis of dynamic motion and reconfiguration process for a modular robot M-TRAN. In *Computational Intelligence in Robotics and Automation, 2003. Proceedings. 2003 IEEE International Symposium on*, volume 2, pages 1004–1010. IEEE.

[Ziegler and Banzhaf, 2001] Ziegler, J. and Banzhaf, W. (2001). Evolving control metabolisms for a robot. *Artificial Life*, 7(2):171–190.