Chapter 21
Evolutionary Machine Learning for Space

Moritz von Looz, Alexander Hadjiivanov, and Emmanuel Blazquez

Abstract The Venn diagram of evolutionary computation, machine learning and space applications shows some intriguing overlaps. As evolutionary algorithms are often resource-intensive, they have not yet been applied *in* space. Nevertheless, it has been decisively demonstrated that evolutionary machine learning (EML) is a valuable tool *for* space, specifically in fields such as trajectory optimisation, optimal control and neuroevolution for robot control, where high-dimensional, discontinuous, sparse and/or non-linear problems abound. In the following chapter, we introduce common problems faced by the space research and application community, together with EML techniques used for generating robust, performant and, sometimes indeed, state-of-the-art solutions. The often complex mathematics behind some problems (especially in trajectory optimisation and optimal control) has been simplified to the minimum necessary to convey the essence of the challenge without encumbering the overview of the relevant EML algorithms. We hope that this chapter provides useful information to both the EML and the space communities in the form of algorithms, benchmarks and standing challenges.

21.1 Introduction

The intersection between evolutionary methods and classical machine learning is extensive and varied [55]. Both approaches have found many applications in the space domain individually: evolutionary methods for interplanetary trajectory optimisation, recurrent neural network architectures for guidance, navigation and control (GNC) tasks, neurocontrollers for in-space robotics or convolutional neural networks for image recognition in earth observation (EO) and astronomy. One of the earliest demonstrations of a real-world application of evolution was an antenna designed by a genetic programming algorithm [34] for NASA's Space Technology 5 mission. The performance of the final design was found to be comparable to hand-crafted

M. von Looz (✉) · A. Hadjiivanov · E. Blazquez
European Space Agency ESA, Keplerlaan 1, 2201 AZ Noordwijk, Netherlands
e-mail: moritz@vlooz.de

W. Banzhaf et al. (eds.), *Handbook of Evolutionary Machine Learning*,
Genetic and Evolutionary Computation,
https://doi.org/10.1007/978-981-99-3814-8_21

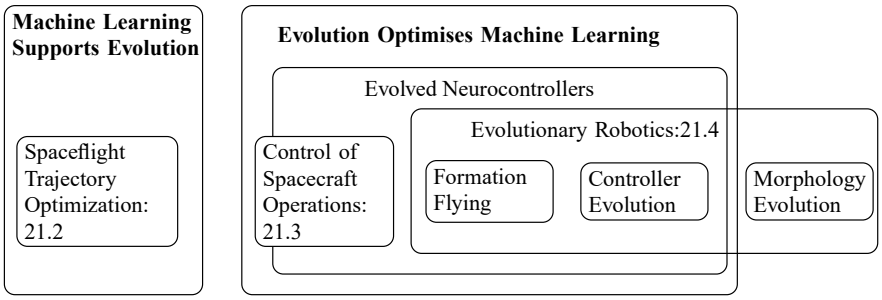

Fig. 21.1 EML for space applications considered in this chapter

antennae. It was evaluated on an actual satellite, and a subsequent study successfully re-evolved the design [3] to comply with changes in the requirements.

An important distinction is that the space domain is not restricted to applications *physically* in space (generally known as *on-board* applications). Traditionally, requirements in terms of robustness and reliability for any physical device deployed in space have been, and remain, notoriously high. This is due to the stringent safety qualification and performance requirements, which in turn stem from the harsh environment and high cost of failures. Furthermore, due to the limited power budget, computational capabilities and support for parallelisation of spaceflight on-board hardware and software, the use of evolutionary algorithms is currently limited to on-ground pre- and post-processing.

Still, just like life on Earth, which flourishes even in the most inhospitable environments thanks to biological evolution, evolutionary machine learning (EML) has found its niche applications in space-related research. Specifically, EML *for* space applications is an optimisation paradigm that is growing in popularity and can benefit from everything that evolution has to offer, such as tackling large and discontinuous fitness landscapes, as well as robustness to local minima.

The applications of EML methods in the space sector can be broadly grouped into two categories, see Fig. 21.1. The first one is the use of machine learning methods for assisting evolution: Evolutionary methods have been highly successful in interplanetary trajectory design [29], and we will discuss potential applications of EML for trajectories in Sect. 21.2.

The second area of intersection is classical machine learning (most commonly artificial neural networks) assisted by evolutionary methods. This takes the form of neural architecture search, genetic programming or direct optimisation of network weights. Many guidance and path planning tasks in space can be mathematically framed as control problems and addressed with neurocontrollers, as we will discuss in Sect. 21.3. This applies specifically to robotic planetary exploration, see Sect. 21.4.

It is important to note that vision-based object classification is a major task in astronomy, and the field has embraced evolutionary machine learning methods for this purpose [16, 24]. One particular challenge is the identification of objects in the presence of stray light sources and diffusion. For instance, Jones et al. [30] use an

evolutionary approach to neural architecture search, in which they evolve parameters specifying the topology of a convolutional neural network tasked with identifying galaxies in the Zone of Avoidance (the part of the universe that is obscured by the galactic disk of the Milky Way). However, we choose to direct our focus to spaceflight applications in the remainder of this chapter as this matches our area of expertise. Below, we present several categories of such applications, together with examples and future directions in Sect. 21.5.

21.2 Spaceflight Trajectory Optimisation

Among the applications of evolutionary algorithms for spaceflight, the optimization of trajectories is by far the most successful. Numerous winning entries in the series of the Global Trajectory Optimisation Competition (GTOC) have relied primarily on evolutionary algorithms, occasionally augmented with machine learning techniques.

In the absence of other forces, a spacecraft will follow a ballistic, predictable arc. A trajectory is thus characterised by the timing and nature of its *manoeuvres*, either using some propulsion system or gravitational interactions. Propulsion systems can be classified by their *exhaust velocity* and *thrust*. A higher exhaust velocity makes the system more efficient, and larger manoeuvres can be done with the same amount of fuel. A higher thrust allows manoeuvres to be done in shorter time, making their planning and modelling much easier. For classical, *chemical* engines, the thrust is high enough that on the time scales for interplanetary trajectories, manoeuvres can be modelled as instantaneous. In contrast, *low-thrust* engines, for example, ion drives and other sorts of electrical propulsion, are using their fuel much more efficiently, but manoeuvres take weeks or months [59]. Spacecraft with low-thrust engines effectively accelerate continuously on a large part of their trajectory, travelling on long spirals instead of ballistic arcs. Figure 21.2 shows an example of a low-thrust trajectory from an Earth parking orbit to the moon.

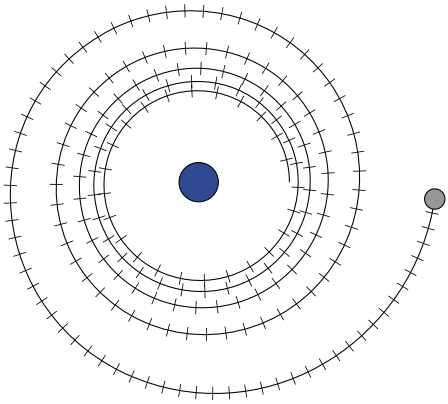

Fig. 21.2 An illustration of a low-thrust trajectory, inspired by the SMART-1 technology demonstrator mission to the moon. The continuous thrust results in trajectories with a large number of free variables, illustrated by the ticks. (Not to scale)

The key figure of merit of a trajectory is the total *change of velocity* required to be fulfilled by its propulsive manoeuvres.[1] It is denoted as ΔV, and the fuel requirements of a mission scale exponentially with it

$$m_{\text{fuel}} = \left(e^{\Delta V / v_{\text{exhaust}}} - 1\right) \cdot m_{\text{dry}}. \tag{21.1}$$

Equation 21.1 stems from rearranging the famous Tsiolkovsky's rocket equation, with m_{fuel} denoting the fuel mass, v_{exhaust} the exhaust velocity of the propulsion system and m_{dry} the mass of everything else: the payload, but also the weight of the propulsion system and empty tanks. Due to the exponential growth, fuel requirements quickly become prohibitive for higher values of ΔV. Designing efficient trajectories is thus imperative not only to lower the cost of a mission but also to make it possible at all.

A basic building block of efficient trajectory design is the use of *gravity assist* manoeuvres (GAs). In these, a spacecraft flies past a moving massive body, and exchanges momentum due to the gravitational interaction. Multiple flybys can be chained together to increase the saved ΔV, but require precise timing. This technique, denoted as *Multiple Gravity Assists* (MGA), enabled the Grand Tour of Voyager 2 [31] and has become nearly indispensable in modern interplanetary missions. Figure 21.3 shows an example trajectory with four gravity assists to reach Jupiter.

The parameterisation of a trajectory can be conceptually reduced to the epochs, magnitudes and directions of its manoeuvres, including gravity assists. For *impulsive* trajectory designs making use of chemical engines, the arcs between manoeuvres are ballistic, see also Fig. 21.3. For low-thrust trajectories with continuous acceleration, the continuous-time optimisation problem is first discretised and free parameters, such as thrust angles and epochs, are specified for each discretisation node, as shown in Fig. 21.2.

The resulting optimisation problems are high-dimensional, even more so for low-thrust trajectories. As the effect of a gravity assist depends on precise timing, the cost functions have steep gradients and many local optima which makes them challenging to optimise. Analytical gradients are not always available and local optimisation relies heavily on the quality of an initial estimate. Evolutionary algorithms have thus been used extensively and with considerable success to initialise global trajectory optimisation searches [29, 59]. In these, the genotype specifies the trajectory parameters, while the fitness is set to its cumulative ΔV (and thus the logarithm of fuel use) over the entire trajectory.

Multiple approaches have been developed to accelerate the evolutionary trajectory search. Izzo et al. [27] introduce the *archipelago* model to make use of parallel computing resources. In this model, multiple populations of solutions are evolved simultaneously with regular migration. In addition to allowing a higher degree of parallelisation, this approach prevents the evolution from getting stuck in local minima. Some trajectory problems have combinatorial aspects in addition to the continuous

[1] Other considerations include time of flight, radiation load, timing of manoeuvres and targets of opportunity.

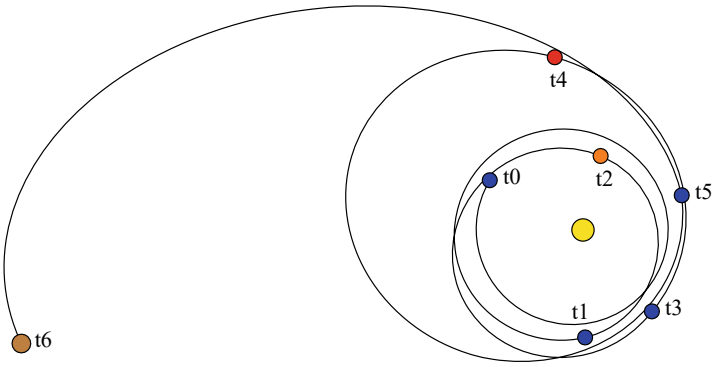

Fig. 21.3 A trajectory using multiple gravity assists on Earth, Mars and Venus on the way to Jupiter. Each coloured dot marks the location of a planet during a gravity assist manoeuvre. The Earth is visited multiple times. The trajectory is inspired by ESA's JUICE mission, launched in 2023 and currently en route to the Jupiter system. In this parametrisation, the free parameters are only the times t0–t6: the timings of launch, the flybys and the arrival

ones. Examples include the right order of planets to visit for flybys or the creation of tours covering multiple moons or asteroids. Well-known instances include the *Global Trajectory Optimisation Challenge* (GTOC), a regular competition of complex trajectory problems. The seventh GTOC, for example, challenged participants to explore the asteroid belt with multiple craft, while the ninth GTOC posed the problem of selecting, visiting and removing pieces of space debris with a limited fuel budget [25]. More applied mission proposals also face the problem of target selection and visitation orders [47].

21.2.1 Use of Machine Learning Methods

Due to the inherent randomness of evolutionary optimisation and the many local minima prevalent in trajectory problems, it is beneficial to perform multiple restarts of the optimisation process, starting from different seeds to have a higher chance of reaching a global optimum. Due to this process, but also during a single evolutionary run, large amounts of data about the fitness landscape are gathered.

Multiple machine learning approaches have been developed to take advantage of this data to learn the fitness landscape. Cassioli et al. [8] use a Support Vector Machine (SVM) to classify starting points by whether Monotonic Basin Hopping (MBH) will find a good enough solution if starting from that point. They test this approach on a set of trajectory problems from ESA's Advanced Concepts Team (ACT) [56] and report a reduction of total fitness evaluations by about a third. More recently, *Estimation of Distribution Algorithms* (EDAs) [35] have seen some applications to trajectory design, in particular to low-thrust scenarios. EDAs refer to a

class of evolutionary algorithms that makes use of probabilistic models to detect the most promising chromosomes in a population and evolve it. Shirazi et al. [50] applied an enhanced EDA algorithm based on Gaussian distribution learning to solve time-varying Lyapunov control problems for low-thrust transfers in the Earth environment, showing the promise of these techniques in obtaining feasible near-optimal solutions with reasonable performance while offering insight into the probabilistic fitness landscape of the problem at hand. In regards to target selection, Choi et al. [9] target six multiple gravity assist (MGA) trajectory optimisation problems (including the 1st GTOC) as benchmarks for a novel adaptive differential evolution (DE) method, which relies on selecting the most performant strategy from a pool of EAs (DE/rand/1, DE/rand/2, DE/best/2, DE/current-to-best/1, DE/current-to-pbest/1 and DE/current-to-opposition/1) at runtime. Specifically, the adaptive DE method involves recording two extra pieces of information for each individual in the population: the current strategy being used and the time since the last fitness update. If the individual is stagnant, its strategy is updated with a random one from the pool of algorithms, and the best-performing strategies become increasingly likely to be selected as a result of applying the Cauchy distribution method in [10]. Choi et al. [9] report that their DE method matched the currently known best results for two problems and improved on the state of the art for the remaining four problems.

21.2.1.1 Surrogate Models

An alternative approach is to train a model to approximate the fitness function directly, commonly known as *surrogate model*. For instance, Ampatzis et al. [1] describe a surrogate model for Multiple Gravity Assist problems. They use a feed-forward network with two hidden layers of 25 neurons each, with the logistic and tangent activation functions used in the first and second hidden layers, respectively. With this, they find that apart from the strict performance benefit, using a surrogate function also improves the evolutionary process by smoothing out the rugged fitness landscape. The training data for the surrogate model comes from the evolved population itself. In order to address the trade-off between the actual and the surrogate models in terms of the number of fitness evaluations, Ampatzis et al. [1] use a fixed number of generations and alternate evolving the population with the true model and the surrogate model. They use Differential Evolution (DE) [45] for this optimisation and find that on problems such as recreating the Cassini mission trajectory [44], this hybrid approach needs in expectation as many generations as the non-surrogate version, but with much faster function evaluations in generations using the surrogate model.

Stubbig et al. [54] train a three-layer ReLU network as surrogate function for low-thrust trajectories in the hodographic shaping method. They find that feature engineering is relevant, as including more, even redundant information in the state vectors increases the model accuracy.

Hennes et al. [20] develop fast approximators for low-thrust transfers in the main asteroid belt. Low-thrust transfers are expensive to compute and the number of pos-

sible transfers is large in a dense asteroid population, it thus becomes necessary to quickly decide which of them to compute in detail. Approximating low-thrust transfers as ballistic (Lambert) transfers with impulsive manoeuvres is fast but lacks accuracy. The authors find that most common machine learning methods outperform the ballistic approximation, with gradient-boosted decision trees performing best, but all need the right input features. These include the phasing, defining how far along each asteroid is in its orbit. Crucially, the machine learning methods perform best if the ballistic approximation itself is also part of the input features.

Merata et al. [38] extend this approach to Near-Earth Asteroids (NEA), which are closer to the sun than the main belt asteroids and thus have shorter orbital periods. The transfers may thus span multiple orbital periods and the phasing indicators become meaningless. The authors generate a database of 60000 Optimal Control Problem (OCP) descriptions of near-earth asteroid transfers and train an ensemble of machine learning methods. They find that feature selection of the right astrodynamical variables (time of transfer and differences in orbital radius and inclination) is critical for good performance.

21.2.1.2 Parameter Learning

The behaviour of evolutionary algorithms is commonly governed by parameters, whose tuning can have a marked effect on the algorithm's performance. Within evolutionary approaches, Omran et al. [40] use *self-adaptation* to great advantage in differential evolution and Zuo et al. [61] build on that success with a technique they name *case learning*: Keeping a reservoir of mutation and crossover parameters that resulted in improved offspring within differential evolution.

21.3 Optimal Control of Spacecraft Operations

Optimal control refers to the methodology of finding, given a dynamical system and an associated figure of merit defined as the *cost*, a series of control inputs over a period of time that results in the minimisation or maximisation of said cost. A challenge in many control problems, especially for spacecraft applications, is that designing robust and highly performant near-optimal controllers comes at the expense of thorough system identification and computationally expensive open- and closed-loop simulations. Rephrased more generally in the reinforcement learning literature, optimal control is akin to an agent perceiving an environment and selecting actions in order to maximise a reward. Actions are selected according to a *policy*, commonly implemented as a neural network. This can be done by training estimators for the expected value of actions and states, as is done with Q-learning or actor–critic networks, or by optimising the network weights of a policy network directly. The latter strategy is called *direct policy search*. Most neuroevolution methods used for optimal control correspond to gradient-free direct policy search. In recent years, *neurocon-*

trollers have appeared as potential solutions for real-time on-board control of space systems. Neuro-controllers are typically defined as neural network mapping a series of system state inputs to control action outputs with the goal to replace traditional controller design with neural architectures.

For example, Leitner et al. [32] design a neurocontroller for autonomous rendezvous and docking between two spacecraft. It is assumed that only one spacecraft is controllable in translation and attitude with thruster actuators controlled via feed-forward neural networks. In a two-dimensional dynamical system, the controlling network receives position, speed, attitude and angular velocity, for a total of six inputs, and produces two outputs that directly control two thrusters. The overall network is relatively small, consisting of 30 neurons and 103 weights, which are optimised using a simple genetic algorithm from randomly selected starting points. Leitner et al. [32] find that the neurocontroller uses more fuel and time than a numerical optimal control strategy, but can deal better with unexpected changes to the dynamics. They refer to this trade-off as *the price of robustness*.

Dachwald [11] consider very-low-thrust trajectories suitable for solar sails or nuclear electric propulsion, with thrusts in the range of mN/kg. They discretise the trajectory, with the thrust angle and magnitude used as free parameters at each time step. Then, instead of optimising the trajectory directly, they pose the control problem as a reinforcement learning problem and design a neural network (neurocontroller) to take the current state as input and return the thrust as output. They then optimise the network weights via direct policy search.

Willis et al. [57] design a neurocontroller for hovering over a non-spherical asteroid. A particular challenge in navigation around small bodies is their weak gravitational field, leading to a relatively stronger effect of inhomogeneities, solar radiation pressure and other perturbative forces. They investigate a sensor setup without direct distance measurements, instead using optical flow to estimate the ratio of relative velocity to distance from the surface. They then propose a fixed two-layer feed-forward neural network to minimise the offset in each direction independently and optimise the weights with direct policy search using generational particle swarm optimisation (PSO) on an archipelago setup. Multiple populations are evolved independently with regular migration, for a total population of 504 individuals over 1000 generations. They find that, although the lack of absolute distance measurements has a negative effect on the control performance, the evolutionary approach yields significantly better hovering controllers than previous work.

Yang et al. [33] apply a neural guidance scheme using an artificial neural network to control thrust vector steering during low-thrust transfers in the Earth environment, using a Lyapunov control scheme and an improved cooperative evolutionary algorithm to evolve the parameters of the network. The proposed solution shows reasonable accuracy in the satisfaction of the constraints of the corresponding optimal control problem, with the potential of offering an on-board autonomous guidance solution. The robustness of the proposed architecture and its integration within a complete GNC solution still remain open questions.

Marchetti et al. [36] design a hybrid control scheme in which they first find a control law using genetic programming, then optimise it (possibly online) with

artificial neural networks. The coefficients in the evolved control law and the weights of control variables are first optimised with classical approaches (Broyden–Fletcher–Goldfarb–Shanno and Nelder–Mead) with a set of random disturbances. The reference trajectories together with the optimised scalars are used as training data for neural networks. They test this approach on a control task of a single-stage-to-orbit transfer vehicle with random disturbances. Evaluating the neural networks trained on the optimised trajectories yields a success ratio (being within 1% of the reference trajectory) of roughly 65-85%.

Zhang et al. [60] survey different combinations of genetic programming and machine learning approaches in the context of job shop scheduling. Some of them, for example, surrogate functions, are also widely used within evolutionary machine learning for space. In their case, the surrogate models consist of equivalent, but smaller scheduling problems. While genetic programming approaches for space applications are currently a niche application, many of the assistive techniques mentioned by Zhang et al. [60] (for example, feature selection) are likely to be beneficial as well.

21.4 Evolutionary Robotics

In the specific context of space, there are a number of confounding factors that robotics design needs to take into account. Deep space is an exceedingly harsh environment characterised by high vacuum, strong ionising radiation and fluctuating extreme temperatures. Similarly, when considering planetary exploration, there can be a number of complicating factors, such as low or high gravitational pull, broken terrain characteristics and a corrosive, abrasive or pressurised environment. None of these factors are specific to evolutionary robotics (ER)—indeed, they must be dealt with by *any* algorithm used for optimising a robot's morphology or behaviour. In this section, rather than a thorough overview of the fundamentals of ER, we present approaches that take into consideration at least some of the constraints associated with robotic space exploration. We therefore introduce some insight into morphology evolution (ME) and controller evolution (CE) within a space context, followed by an overview of self-assembly and reconfiguration, which are more niche but prospective, realistic future space applications subject to harsh environmental constraints.

21.4.1 Morphology Evolution

Morphology evolution (ME) focuses on evolving the shape, propulsion mode, materials and other physical aspects of the robot. In ME, the controller is usually fixed, whereas controller evolution (CE) focuses on evolving the robot's behaviour while keeping its morphology fixed (or at least that the *type* of control is fixed, in other words, if the mode of propulsion is bipedal motion, it is guaranteed that the controller

is not going to have to deal with jet-based propulsion). Morphology/controller co-evolution is a combination of both: it is a manifestation of the philosophy of *embodied intelligence* [21, 22]—the idea that true intelligence can only be achieved with a physical embodiment. CE is conceptually and practically very different from ME—while robot morphology can be evolved in simulation, there is significant overhead associated with the production and testing of the final design, as well as a limitation on the number of components and their types. In contrast, evolved controllers can be readily transferred to actual hardware and tested immediately, leading to a much faster prototyping cycle.

Studies on ME that take into account the fact that the robot will be operating in space are few and far between. In one example, Rommerman et al. [46] apply Covariance Matrix Adaptation—Evolutionary Strategy (CMA-ES) [19] to the morphology evolution of a crawling robot. However, CMA-ES itself does not consider any space-related constraints; instead, it is the base robotic platform (ARAMIES [53]) that makes the results relevant to space applications. Indeed, the ARAMIES robot was designed specifically for moving over rough terrain at steep inclinations, with high robustness, ease of maintenance and long mean time between failures. Other studies have focused on evolving specific subsystems, such as active vision [7, 42, 43] in a simulated rover, or behaviour, such as trajectory optimisation [11, 12] or path planning behaviour [51] for a swarm of rovers. In general, ME for robots designed specifically for space exploration suffers from the issue of *unknown unknowns*—sets of physical parameters that are too poorly understood to be used as constraints or objectives. This is not a limitation of ER *per se*—rather, it is the result of our limited understanding of and inability to reproduce the environments in which the evolved robots might operate, leading to a degree of uncertainty that is currently too high for ME to be applied to real-world optimisation tasks.

21.4.2 Controller Evolution

A relevant example of CE for space applications deals with the feasibility of bio-inspired design specifically targeting deployment on Mars as presented in [14]. The study considers realistic constraints associated with the particular environment of its intended operation, such as communication, temperature range, batteries, landing site and even soil composition. Consequently, Ellery et al. [14] consider the versatility of different propulsion techniques and adopt an insect-like hexapodal structure as well as other insect-like qualities such as perceptual, behavioural and functional. A noteworthy feature of the Mars Walker includes the implementation of *reflex motions*, namely the *searching* reflex and the *elevator* reflex [15], which counteract external perturbations by activating when obstacles or gaps are encountered during regular locomotion. Critically, lessons learned from previous missions to Mars (such as opportunity becoming stuck on relatively flat terrain) as well as the particular operating conditions (e.g., a limited power budget) are translated into requirements for the *controller*, namely, energy efficiency, ability to cover large gaps, robustness

and stability. Therefore, Ellery et al. [14] opt for a model of the stick insect as a natural match for the insect-like body design of the Mars Walker. The weights of the neural networks designed as controllers for each of the hexapod's legs are optimised using the island model [23], which has emerged as a useful tool for evolutionary optimisation in cases that involve multiple constraints, and in particular in studies on CE for neurocontrollers in the context of designing robots for space exploration. In the case of the Mars Walker, the controller is modelled as a continuous-time recurrent neural network (CTRNN) with weights evolved with coevolutionary distributed GA. Ellery et al. [14] use the Open Dynamics Engine (ODE) simulator [52] to evaluate the robot's performance. The same approach is also taken by Peniak et al. [42], where the objective is developing an active vision system for obstacle avoidance in unknown environments, targeting a simulated version of a Mars rover. A dedicated study on the island model for multi-agent CE scenarios is given in [7].

Ampatzis et al. [2] consider a robot self-assembly task and investigate the role of isolated populations and migrations for the evolution of neurocontrollers. They use Continuous-Time Recurrent Neural Networks (CTRNNs) of 24 neurons in total and optimise their weights with direct policy search using differential evolution on an archipelago with 10 islands, both without migration and a ring topology. They find that migration has a strong effect on the final fitness: Without migration, only 80% of the maximum fitness is reached, even with a higher number of generations. Peniak et al. [41] use evolutionary search to design neurocontrollers for autonomous planetary rovers. The controllers are implemented as neural networks and receive as input a simple sensory system that can serve as a backup in case 3D vision becomes unavailable. The evolutionary search is a simple genetic algorithm implemented on the archipelago framework, with 9 islands of 10 individuals each. A virtual version of the Mars Science Laboratory (MSL) rover is used in a virtual environment as the base for the simulation. The authors find the island model to be successful for evolving neurocontrollers in this framework.

In a work of de Croon et al. [13], a robot controller is evolved with the objective of odour-based localisation of a simulated methane plume on Mars in the presence of wind with low or high turbulence. They consider a robot equipped with a single chemical sensor and a single wind sensor, and train a CTRNN architecture consisting of 4 input neurons (whose inputs are computed from the two simulated sensors), 10 hidden neurons and 4 output neurons. A population of 30 individuals is used, with 1 elite individual and 6 parents selected on a roulette wheel principle. Crossover is carried out with a probability of 0.1, Gaussian mutation with a probability of 0.03 per gene. They use the simple genetic algorithm implementation from the PyGMO/PAGMO platform [4] with direct policy search, where the objective is to locate and approach the methane source. An interesting approach taken in the study is to interpret the evolved policies as finite-state machines, thus distilling an algorithm from the network behaviour. Arguably, the ability to translate a more or less 'opaque' neural model into a highly explainable and convenient symbolic or algorithmic representation for human mission analysts is in fact one crucial step towards the more widespread adoption of evolutionary methods in space-related research involving any form of control.

A key takeaway is that the main differentiator in the design of ME and CE for robots deployed in space is whether the design process considers specific factors of the mission or the environment as either constraints or objectives. The following is a non-exhaustive list of factors to consider:

- **Payload restrictions**: What is the payload mass that can be launched, determined by launch capacity.
- **Extreme conditions**: If the robot is meant to operate in highly inhospitable environments involving as high pressure, radiation, corrosive substances or extreme temperatures (high, low or alternating between the two extremes).
- **Communication delay**: Even at the speed of light, a round-trip to Mars takes over 30 min, and the delay only becomes longer for more distant missions. This translates into a requirement for semi- or even fully autonomous robots.
- **Microgravity**: If the robot is expected to operate in space rather than on (or beneath) the surface of a celestial body, the non-trivial effect of microgravity must be taken into account for everything from propulsion to collision avoidance.

Microgravity is also one of the aspects of space in terms of *environmental conditions* that has no analogue on Earth. For instance, friction is effectively absent in space, and therefore one cannot assume that the robot would simply stop moving when the propulsion is cut off. Microgravity can serve either as an obstacle or as an opportunity depending on the application. Thus, it provides a segue into the next section, which looks at the specific application of *formation flying*, where ER has been applied in a microgravity environment.

21.4.3 Formation Flying and In-Orbit Assembly

In-orbit assembly [6, 58] refers to the process of putting individual components together in-orbit to create a larger structure, and reconfiguration is the process of reshaping or otherwise altering an already assembled structure. Arguably the most successful case of in-orbit assembly is the International Space Station (ISS) [6], which is composed of several modules designed, manufactured and launched by different international actors and space agencies. A unique feature of in-orbit assembly is that it happens in a microgravity environment, which allows certain manoeuvres that are impossible or impractical on the ground.

A noteworthy type of in-orbit assembly is *self-assembly*, where the components self-organise into a larger target structure. An interesting example is given by Shen et al. [49], who study the feasibility of achieving a stable predefined structure from a random starting configuration of *tethered* units (individual robots in a swarm). The tethers feature universal connectors that allow any two connectors to dock together, enabling the robots to form any possible configuration. The study models a system of Intelligent Reconfigurable Components (IRCs) composed of FIMER and CONRO robots that provide tethering connections between pairs of IRCs, systems for position, orientation and wireless communication, and an onboard controller for

topology discovery, planning and communication. As the tethers are flexible, when a connection is established between two IRCs, the tether is reeled in to eliminate slack. The taut link, that is established, then behaves as a rigid beam due to the synchronised coupled orbital motion of the robots at either end. System optimisation relies on a hormone-inspired distributed control algorithm developed specifically for the CONRO robots [48]. Notably, the mechanics of the docking procedure produces non-trivial effects, such as dampened oscillatory spinning that results in alternating clockwise/counterclockwise twisting of the tether until an equilibrium point is reached. Nevertheless, Shen et al. [49] show that the method can grow stable tethered structures by allowing the robots to dynamically reconfigure their communication and control strategy by following the simple preferential attachment instructions encoded in the form of artificial hormones.

There are several key motivations for pursuing in-orbit (self-)assembly:

- **Payload management**: Smaller components are easier and cheaper to launch than a single large structure.
- **Maintainability**: Modular structures are easier to repair and upgrade *in situ* (i.e. in orbit) without decommissioning the entire structure.
- **Repurposing**: Reconfigurable structures can be repurposed into different configurations.

In most cases, in-space assembly relies on the fundamental concept of *formation flying* [37], where the objective is to control a fleet of independent satellites in such a way that the respective *relative* distance between each pair of satellites remains fixed. This enables servicing procedures such as rendezvous and docking to take place, while satellites flying in formation can effectively emulate a rigid structure.

From the point of view of ER, formation flying poses an interesting challenge. Orbital dynamics is well-studied and tractable with various optimisation algorithms, while the environment poses some unique constraints (such as working in microgravity) and opportunities (allowing for the creativity of evolution to shine in design and optimisation settings). In the following, we briefly expand on an interesting application of evolutionary algorithms in the context of an inverse-dynamic approach to formation flying named *equilibrium shaping* [26].

21.4.4 Case Study: Equilibrium Shaping of Spacecraft Swarm

Equilibrium shaping (ES) has been developed for the purpose of achieving a stable spacecraft formation from arbitrary initial conditions (i.e. with satellites distributed randomly within a certain volume). It is a behaviour-based path planning algorithm based on a swarm control technique that introduces an artificial potential field [17], where the agents follow the negative gradient of the potential towards unique attractors identified as minima in the global potential landscape. In ES, each agent in the spacecraft swarm follows distinct behaviours that define the overall velocity field

for each agent. The net effect of all behaviours ensures that the agent ends up in one of the equilibrium points designed to coincide with the desired formation. Key properties of ES are the minimisation of inter-agent communication and its limited sensory information requirements.

Formally, the total velocity v_i of each satellite i is distributed as the sum of the velocities defined by three distinct behavioural patterns: *gather*, *dock* and *avoid*:

$$v_i = v_i^{gather} + v_i^{dock} + v_i^{avoid} \tag{21.2}$$

where each of the individual behavioural velocity components can be decomposed as a discrete sum of non-linear functions over state decision vectors, leading to a straightforward *nonlinear programming* formulation. This makes ES quite amenable to evolutionary optimisation techniques. To that end, Izzo et al. [28] apply ES to the problem of formation flying as posed in terms of swarm control [18], where neural controllers are tasked with maintaining the formation in a decentralised manner, without a dedicated control unit. Specifically, the set-up involves the SPHERES robotic platform, consisting of three spherical devices which can manoeuvre independently in six degrees of freedom. The authors design two distinct multi-layer perceptrons (MLPs) networks, one of which translates the relative positions of the satellites into velocities and another one for translating the target formation into angular velocities. They find that the controllers encounter difficulties achieving rotational invariance, i.e. having the controller perform the same action for two states that only differ by a rotated reference frame. For this reason, they use an additional sensor input representing an absolute reference frame. Note that the controllers are identical for each satellite, so every agent is controlled in the same way by the controller and can be swapped freely. Using particle swarm optimisation, the controllers are evolved to achieve sub-micrometre positional accuracy in 99.93% of 25000 simulation runs with the help of the SPHERES simulator [39].

21.4.5 Future Challenge: Fault Recovery

Robots deployed in space would be operating in unknown environments that are often harsh and can cause damage to the structural elements of the robot, its hardware controller (including processor, memory and other components), or both. However, unlike a terrestrial robot, which would be repaired or replaced in the event of damage, a robot in space would need to recover from the damage as fully as possible in order to complete the mission. Such failure recovery should take place at the controller level while still maintaining the strong guarantees for robustness and interpretability normally required for a space-grade controller.

In this regard, the translation of a neural network behaviour into a series of logical steps in de Croon et al. [13] highlights a potential pathway for neuroevolution for robot control in space. Specifically, as evolution is well suited for dealing with

changes in the topology of the search space and the solution itself, it can serve as a valuable tool for controller recovery in the case of irreparable hardware damage. For instance, reconfigurable hardware platforms, such as FPGAs, are becoming increasingly popular and energy efficient, and could ultimately find their way onboard space probes and robots. A controller implemented in FPGA can be reconfigured dynamically in case of a failure; however, since there is no way to estimate with certainty the type of failure, the reconfiguration procedure should be as generic as possible. The recovered controller might need to work with altered inputs (for instance, a blocked wheel or undeployed solar panel) or more subtle changes such as permanently corrupted memory or processor registers. Currently, the place-and-route step is a very computationally intensive part of the FPGA update cycle, which needs to be done with ground support. Further algorithmic improvement would be necessary to limit the recomputations to a small environment around a fault. While to our knowledge there are no dedicated studies on neuroevolution for controller recovery for robots deployed in space, a system such as that in [5] could in principle be used as a fallback system for controller recovery, even if it is initially designed to be used *only* in the case of such irreparable damage. An additional advantage is that robot systems and failures can be simulated with high accuracy and reproducibility, paving the way to fast prototyping cycle. We anticipate that advancing the state of the art in that direction would greatly expand the potential for the application of evolution to robot control for space applications.

21.5 Conclusions

Evolutionary computation and machine learning have been conjointly used for a wide number of space applications, including trajectory and path optimisation, robotics guidance and control and morphology evolution.

In some applications, combining evolutionary and machine learning methods helps in further reducing the need for human design choices. For instance, the design of neural networks, including the number and width of layers or the choice of activation functions, can be partially automated using evolutionary neural architecture search (ENAS). Evolutionary algorithms are used extensively in the prospect of automating offline trajectory design, and machine learning methods is used in shaping the search space, reducing expensive function evaluations and complex system modeling.

So far, the combined use of machine learning and evolutionary optimisation remains exploratory in nature. No evolutionary machine learning technique has yet become the standard used to solve a particular challenge in space engineering, despite the promise shown by the prospective studies showcased in this review. A dynamic subfield within EML for space applications is optimal control problems solved with artificial neural networks, which are in turn optimised with evolutionary methods. This, together with learning the search space in evolutionary trajectory design seems to be the key promising areas for future developments.

References

1. Ampatzis, C., Izzo, D.: Machine learning techniques for approximation of objective functions in trajectory optimisation. In: Proceedings of the IJCAI-09 Workshop on Artificial Intelligence in Space, pp. 1–6 (2009)
2. Ampatzis, C., Izzo, D., Ruciński, M., Biscani, F.: Alife in the galapagos: migration effects on neuro-controller design. In: Advances in Artificial Life. Darwin Meets von Neumann: 10th European Conference, ECAL 2009, Budapest, Hungary, 13–16 Sept. 2009, Revised Selected Papers, Part I 10, pp. 197–204. Springer (2011)
3. Basak, A., Lohn, J.D.: A comparison of evolutionary algorithms on a set of antenna design benchmarks. In: 2013 IEEE Congress on Evolutionary Computation, pp. 598–604. IEEE (2013)
4. Biscani, F., Izzo, D.: A parallel global multiobjective framework for optimization: pagmo. J. Open Sour. Softw. **5**(53), 2338 (2020)
5. Borrett, F., Beckerleg, M.: A comparison of an evolvable hardware controller with an artificial neural network used for evolving the gait of a hexapod robot. Gen. Programm. Evol. Mach. **24**(1), 5 (2023)
6. Boyd, I.D., Buenconsejo, R.S., Piskorz, D., Lal, B., Crane, K.W., De La Rosa, Elena, B.: On-Orbit Manufacturing and Assembly of Spacecraft. Technical report, Institute for Defense Analyses (2017)
7. Cangelosi, A., Marocco, D., Peniak, M., Bentley, B., Ampatzis, C., Izzo, D.: Evolution in Robotic Islands. Technical Report Ariadna ID: 09-8301, ESA (2010)
8. Cassioli, A., Di Lorenzo, D., Locatelli, M., Schoen, F., Sciandrone, M.: Machine learning for global optimization. Comput. Optim. Appl. **51**, 279–303 (2012)
9. Choi, J.H., Lee, J., Park, C.: Deep-space trajectory optimizations using differential evolution with self-learning. Acta Astronautica **191**, 258–269 (2022)
10. Choi, T.J., Togelius, J., Cheong, Y.-G.: Advanced cauchy mutation for differential evolution in numerical optimization. IEEE Access **8**, 8720–8734 (2020)
11. Dachwald, B.: Optimal solar sail trajectories for missions to the outer solar system. J. Guid. Control Dyn. **28**(6), 1187–1193 (2005)
12. Dachwald, B.: Optimization of very-low-thrust trajectories using evolutionary neurocontrol. Acta Astronautica **57**(2–8), 175–185 (2005)
13. de Croon, G., O'connor, L.M., Nicol, C., Izzo, D.: Evolutionary robotics approach to odor source localization. Neurocomputing **121**, 481–497 (2013)
14. Ellery, A., Scott, G.P., Gao, Y., Husbands, P., Vaughan, E., Eckersley, S.: Mars Walker. Technical Report AO/1-4469/03/NL/SFe, ESA (2005)
15. Espenschied, K.S., Quinn, R.D., Beer, R.D., Chiel, H.J.: Biologically based distributed control and local reflexes improve rough terrain locomotion in a hexapod robot. Robot. Auton. Syst. **18**(1–2), 59–64 (1996)
16. Fluke, C.J., Jacobs, C.: Surveying the reach and maturity of machine learning and artificial intelligence in astronomy. Wiley Interdiscip. Rev.: Data Mining Knowl. Disc. **10**(2), e1349 (2020)
17. Gazi, V.: Swarm aggregations using artificial potentials and sliding-mode control. IEEE Trans. Robot. **21**(6), 1208–1214 (2005)
18. Gazi, V., Fidan, B., Marques, L., Ordonez, R.: Robot swarms: dynamics and control. In: Kececi, E.F., Ceccarelli, M. (eds.), Mobile Robots for Dynamic Environments, pp. 79–126. ASME Press (2015)
19. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evol. Comput. **9**(2), 159–195 (2001)
20. Hennes, D., Izzo, D., Landau, D.: Fast approximators for optimal low-thrust hops between main belt asteroids. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–7. IEEE (2016)
21. Howard, D., Eiben, A.E., Kennedy, D.F., Mouret, J.-B., Valencia, P., Winkler, D.: Evolving embodied intelligence from materials to machines. Nat. Mach. Intell. **1**(1), 12–19 (2019)

22. Howard, D., Glette, K., Cheney, N.: Editorial: evolving robotic morphologies. Front. Robot. AI **9**, 874853 (2022)
23. Husbands, P.: Distributed coevolutionary genetic algorithms for multi-criteria and multi-constraint optimisation. In: Fogarty, T.C. (ed.) Evolutionary Computing. Lecture Notes in Computer Science, vol. 865, pp. 150–165. Springer, Berlin, Heidelberg (1994)
24. Ivezić, Ž., Connolly, A.J., VanderPlas, J.T., Gray, A.: Statistics, data mining, and machine learning in astronomy. In: Statistics, Data Mining, and Machine Learning in Astronomy. Princeton University Press (2014)
25. Izzo, D.: Problem description for the 9th global trajectory optimisation competition. Acta Futura **11**, 49–55 (2017)
26. Izzo, D., Pettazzi, L.: Autonomous and distributed motion planning for satellite swarm. J. Guid. Control Dyn. **30**(2), 449–459 (2007)
27. Izzo, D., Ruciński, M., Biscani, F.: The generalized Island model. Parallel Arch. Bioinspired Algorim. 151–169 (2012)
28. Izzo, D., Simões, L.F., Croon, G.C.H.E.: An evolutionary robotics approach for the distributed control of satellite formations. Evol. Intell. **7**(2), 107–118 (2014)
29. Izzo, D., Sprague, C.I., Tailor, D.V.: Machine learning and evolutionary techniques in interplanetary trajectory design. In: Modeling and Optimization in Space Engineering: State of the Art and New Challenges, pp. 191–210 (2019)
30. Jones, D., Schroeder, A., Nitschke, G.: Evolutionary deep learning to identify galaxies in the zone of avoidance (2019). arXiv:1903.07461
31. Kohlhase, C.E., Penzo, P.A.: Voyager mission description. Space Sci. Rev. **21**(2), 77–101 (1977)
32. Leitner, J., Ampatzis, C., Izzo, D.: Evolving anns for spacecraft rendezvous and docking. In: Proceedings of the 10th International Symposium on Artificial Intelligence, Robotics and Automation in Space, i-SAIRAS 2010, pp. 386–393. European Space Agency (ESA) (2010)
33. Yang, D.l., Xu, B., Zhang, L.: Optimal low-thrust spiral trajectories using lyapunov-based guidance. Acta Astronautica **126**, 275–285 (2016)
34. Lohn, J.D., Hornby, G.S., Linden, D.S.: An evolved antenna for deployment on Nasa's space technology 5 mission. In: O'Reilly, U.-M., Yu, T., Riolo, R., Worzel, B. (eds.), Genetic Programming Theory and Practice II, volume 8 of Genetic Programming, pp. 301–315. Springer (2005)
35. Lozano, J.A., Larrañaga, P., Inza, I., Bengoetxea, E. (eds.): Towards a New Evolutionary Computation. Springer, Berlin, Heidelberg (2006)
36. Marchetti, F., Minisci, E.: A hybrid neural network-genetic programming intelligent control approach. In: Bioinspired Optimization Methods and Their Applications: 9th International Conference, BIOMA 2020, Brussels, Belgium, 19–20 Nov. 2020, Proceedings, vol 9, pp. 240–254. Springer (2020)
37. Mathavaraj, S., Padhi, R.: Satellite Formation Flying: High Precision Guidance Using Optimal and Adaptive Control Techniques. Springer Singapore (2021)
38. Mereta, A., Izzo, D., Wittig, A.: Machine learning of optimal low-thrust transfers between near-earth objects. In: Hybrid Artificial Intelligent Systems: 12th International Conference, HAIS 2017, La Rioja, Spain, June 21-23, 2017, Proceedings, pp. 543–553. Springer (2017)
39. Miller, D., Saenz-Otero, A., Wertz, J., Chen, A., Berkowski, G., Brodel, C., Carlson, S., Carpenter, D., Chen, S., Cheng, S., Feller, D., Jackson, S., Pitts, B., Perez, F., Szuminski, J., Sell, S.: SPHERES: a testbed for long duration satellite formation flying in micro-gravity conditions. Adv. Astronautical Sci. **105** (2000)
40. Omran, MG.H., Salman, A., Engelbrecht, A.P.: Self-adaptive differential evolution. In: Hao, Y., Liu, J., Wang, Y., Cheung, Y.-M., Yin, H., Jiao, L., Ma, J., Jiao, Y.-C. (eds.), Computational Intelligence and Security, Berlin, Heidelberg, pp. 192–199. Springer, Berlin, Heidelberg (2005)
41. Peniak, M., Bentley, B., Marocco, D., Cangelosi, A., Ampatzis, C., Izzo, D., Biscani, F.: An evolutionary approach to designing autonomous planetary rovers. TAROS 2010, pp. 198 (2010)
42. Peniak, M., Bentley, B., Marocco, D., Cangelosi, A., Ampatzis, C., Izzo, D., Biscani, F.: An Island-model framework for evolving neuro-controllers for planetary rover control. In: The 2010 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2010)

43. Peniak, M., Marocco, D., Ramirez-Contla, S., Cangelosi, A.: Active vision for navigating unknown environments: an evolutionary robotics approach for space research. In: Lacoste, H. (ed.), ESA Special Publication, volume 673 of ESA Special Publication, p. 7 (2009)
44. Peralta, F., Flanagan, S.: Cassini interplanetary trajectory design. Control Eng. Pract. **3**(11), 1603–1610 (1995)
45. Price, K.V.: Differential evolution. Handbook of Optimization: From Classical to Modern Approach, pp. 187–214 (2013)
46. Rommerman, M., Kuhn, D., Kirchner, F.: Robot design for space missions using evolutionary computation. In: 2009 IEEE Congress on Evolutionary Computation, pp. 2098–2105. IEEE (2009)
47. Cuartielles, J.P., Gibbings, A., Snodgrass, C., Green, S., Bowles, N.: Asteroid belt multiple flyby options for m-class missions. In: 67th International Astronautical Congress, p. IAC–16.C1.5.7x33119. International Astronautical Federation (2016)
48. Shen, W.-M., Lu, Y., Will, P.: Hormone-based control for self-reconfigurable robots. In: Proceedings of the Fourth International Conference on Autonomous Agents, AGENTS '00, pp. 1–8. Association for Computing Machinery (2000)
49. Shen, W.-M., Will, P.M., Khoshnevis, B.: Self-assembly in space via self-reconfigurable robots. In: 2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422), **2**, 2516–2521 (2003)
50. Shirazi, A., Holt, H., Armellin, R., Baresi, N.: Time-varying lyapunov control laws with enhanced estimation of distribution algorithm for low-thrust trajectory design. In: Modeling and Optimization in Space Engineering: New Concepts and Approaches, pp. 377–399. Springer (2023)
51. Simões, L.F., Cruz, C., Ribeiro, R.A., Correia, L., Seidl, T., Ampatzis, C., Izzo, D.: Path Planning Strategies Inspired By Swarm Behaviour of Plant Root Apexes. Technical Report Ariadna ID: 09/6401, ESA (2011)
52. Smith, R.: Open Dynamics Engine (2008)
53. Spenneberg, D., Albrecht, M., Backhaus, T., Hilljegerdes, J., Kirchner, F., Zschenker, H.: ARAMIES: A four-legged climbing and walking robot. In: Proceedings of the 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space, vol. 603 (2005)
54. Stubbig, L.J., Cowan, K.J.: Improving the evolutionary optimization of interplanetary low-thrust trajectories using a neural network surrogate model. Adv. Astronaut. Sci. **175** (2021)
55. Telikani, A., Tahmassebi, A., Banzhaf, W., Gandomi, A.H.: Evolutionary machine learning: a survey. ACM Comput. Surv. (CSUR) **54**(8), 1–35 (2021)
56. Vinkó, T., Izzo, D.: Global optimisation heuristics and test problems for preliminary spacecraft trajectory design. Advanced Concepts Team, ESATR ACT-TNT-MAD-GOHTPPSTD (2008)
57. Willis, S., Izzo, D., Hennes, D.: Reinforcement learning for spacecraft maneuvering near small bodies. AAS/AIAA Space Flight Mech. Meet. **158**, 1351–1368 (2016)
58. Xue, Z., Liu, J., Chenchen, W., Tong, Y.: Review of in-space assembly technologies. Chinese J. Aeronaut. **34**(11), 21–47 (2021)
59. Yam, C.H., Lorenzo, D.D., Izzo, D.: Low-thrust trajectory design as a constrained global optimization problem. Proc. Inst. Mech. Eng. Part G: J. Aerosp. Eng. **225**(11), 1243–1251 (2011)
60. Zhang, F., Mei, Y., Nguyen, S., Zhang, M.: Survey on genetic programming and machine learning techniques for heuristic design in job shop scheduling. IEEE Trans. Evol. Comput. (2023)
61. Zuo, M., Dai, G., Peng, L., Wang, M., Liu, Z., Chen, C.: A case learning-based differential evolution algorithm for global optimization of interplanetary trajectory design. Appl. Soft Comput. **94**, 106451 (2020)