2 Background and Related Work

This chapter presents the background and related work. In Section 2.1, we introduce the background of this thesis. This includes a brief introduction of the development of artificial intelligence (AI), how AI and robotics are combined, and the research that has been done in recent years in the areas of automation science. Section 2.2 reviews the development of evolutionary computation which is the main technique used in this thesis. This section includes an introduction of biological evolution, principles, strengths and caveats of evolutionary computation and its applications. Section 2.3 introduces how AI/robotics and animal behavior study benefit from each other, which includes how animal behavior can be used as inspiration for AI and robotics and the methods to investigate animal behavior using AI/robotics techniques.

2.1 Background

2.1.1 The Development of AI and Robotics

Intelligence is a natural part of life. Humans and other biological creatures exhibit many intelligent behaviors such as pattern recognition and decision making. However, intelligence is not a property that is limited to biological creatures. It should be equally applicable to computers or machines. The term AI emerged in a conference in 1956 at Dartmouth College, where several pioneers of this filed including Marvin Minsky, John McCarthy, etc., discussed the development of digital computers and the future of AI. The definition of AI is still a disputed topic. Some researchers argued that AI is to simulate the intelligent behaviors which are observed in humans and other biological creatures using computers or machines. That is, an intelligent machine should be able to exhibit behaviors similar to that of live creatures when encountering the same problems [23]. Others gave the following definition: "Artificial Intelligence is the study of

mental faculties through the use of computational models" [24]. According to Fogel [25], an intelligent system should know how to make decisions in order to fulfill a goal (e.g., solving a problem). In other words, instead of pre-programming the machine using human's knowledge, it should be able to learn and adapt. In [26], Minsky even argued, "Why can't we build, once and for all, machines that grow and improve themselves by learning from experience? Why can't we simply explain what we want, and then let our machines do experiments or read some books or go to school, the sorts of things that people do?" In 1950, Turing [27] proposed an imitation game which is nowadays known as *Turing test* to discuss a question: "Can machine think?". Although whether a machine could pass the *Turing test* or not is beyond the consideration at that time, it was accepted as a notion that a machine could learn and adapt to mimic human behavior. Many promising achievements have been made to enable machines to accomplish a variety of intelligent actions since then.

In the 1970s, the emergence of expert system—a computer program that mimics a human expert's decision-making capability [28], significantly promoted the development of AI. An expert system can solve complicated problems through reasoning about the knowledge (which is mainly represented as *if-else* rules) it has. One of the most representative examples is IBM's chess program (Deep Blue). It defeated the champion of the world chess (Gary Kasparov) in 1997 [29], which provides evidence that a computer program can even outperform a human expert in term of decision-making ability. An expert system consists of two components: knowledge base and inference engine. Knowledge base contains facts and rules that are known to the system. Inference engine uses the knowledge to make decisions and derive new rules, which are then stored in the system to update the knowledge base. Expert systems have many commercial applications such as medical diagnosis [30], prediction [31] and monitoring [32], etc. The rules in an expect system can be expressed using Boolean logic or Fuzzy logic. In Boolean logic, every condition in the rules is either true or false. Fuzzy logic was introduced by Zadeh [33] to describe "degree of truth". For example, a cup with water is described as "full (1)" or "empty (0)" using Boolean logic; however, in Fuzzy logic, it can also be described using some fuzzy expressions such as "almost full", "half full", "near empty", etc. Fuzzy logic is used commonly in our daily life. An example rule in an expert system using Fuzzy logic is: *IF the temperature is cold, THEN turn the heater on.* In stock market an old saying is: "buy low, sell high". However, whether the stock value can be considered as low or high depends on the stock curves in a particular situation. Fuzzy systems have many commercial applications in such as air conditioner, digital camera, and and hand

writing recognition, etc. Another representation of AI is neural network, which mimics the processing ability of nervous systems of biological creatures (especially human brain). Through a combination of weights and excitation functions (e.g., sigmoid function), neural networks can accomplish many tasks observed in humans, such as pattern recognition and image processing.

Robotics is a field about making machines that can move in several ways to accomplish certain tasks. While AI and robotics are not essentially connected, they are often used together to make the robots "smarter". For example, a robot with AI can move autonomously and make decision while interacting with the environment it is operating on. Through combining AI and robotics, machines can be created to behave more like humans or animals. Instead of only executing fixed programs as in a car assembling line, robots can learn and adapt to the changing environment.

There are two common architectures adapted for the control of a robot: deliberative architecture and subsumption architecture [34]. In deliberative architecture, the robot operates on a top-down fashion and its action mainly depends on planning. A typical cycle is: *sensor* $\rightarrow$ *plan* $\rightarrow$ *act*. In the sensor stage, the robot gets the information from the world based on sensors such camera, infrared sensors, etc. After pre-processing, this information would be passed to the central control architecture which integrates all the sensing information and reasons about it. Based on the knowledge the robot has to decide which action to take to fulfill a goal (e.g., maximize its reward). This architecture has led to many successful applications. The pioneer work is *Shakey the robot* which is capable of reasoning about its own actions [35]. In this work, the environment the robot is operating on is simplified and the experimental conditions are well controlled (e.g. uniform color and flat floor). More work has been done since then to enable robots to tackle complex and changing environments [36]. Another architecture adapted widely nowadays is subsumption architecture [34], in which the robot makes decision based on *sensor* $\rightarrow$ *act* without deliberate reasoning or planing. Instead of building a central reasoning system to integrate all the sensory information, the robot could process it in parallel by each layer. This could enhance the robustness of the robotic control system. Some famous robots using subsumption architecture are Allen [34], Herbert [37] and Genghis [38].

2.1.2 Introduction of Automation Science

With the development of AI and robotics, intelligent and automation systems were commonly used for assisting scientific research. Since the first clinical automated laboratory management system [39] was created in 1993, such systems are increasingly used in drug discovery, agriculture and energy labs, etc. Nowadays, it is the demand that machines could automate the whole process of scientific research. The question of whether it is possible to automatically conduct scientific research is very interesting in theory, and it also involves a lot of practical work which needs to be solved (e.g., communication and coordination). In order to speed up experimental process, researchers should take advantage of intelligent and automation systems to help, for example, collect and analyze thousands of data, because these things are very time-consuming and boring if only carried out manually. The ideal situation is to make a machine conduct scientific research automatically without or with little human intervention, and it can do experiments day and night in a constant manner without any tiredness and complain.

The field of automation science has been developed to a great extent because of the increasing demands of drug industry and relevant fields of biology and chemistry. High-throughput screening (HTS) systems [40] are one of the early efforts. HTS systems could do many things such as preparation, observation and analysis of assay, greatly enhancing the speed of data collection and analysis process in a short time. Recently, King, et al. have built a Robot Scientist—Adam, which can automatically generate functional genomics hypotheses about the yeast *Saccharomyces cerevisiae* and carry out experiments to test and refine the hypotheses based on the techniques developed in AI and robotics [41, 7]. This Robot Scientist is able to do plenty of experiments and observations a day. The experiments are performed automatically by machines, which makes it possible to test all aspects of experimental process. Adam could automatically conduct the cycles of scientific experiments: forming hypothesis, initializing the experiments, explaining the results and verifying the hypothesis, and then repeating the cycle. The functional genomics hypotheses are autonomously generated by intelligent software and the experiments are conducted coordinating different components in the automated system [41].

In system identification area, Gauld et al. have developed a digital automated identification system (DAISY) to identify biological species automatically with high accuracy using advanced image processing technique [42]. This technology has gone though great

improvement in recent years, raising the possibility of automation, or at least semi-automation, in the process of routine taxonomic identification. In [43], MacLeod et al. reported that an imaging system that is originally designed for identifying marine zooplankton was used by the US government for monitoring horizon oil spill in the deep water. They argue that taxonomists and researchers in machine learning, pattern recognition as well as artificial intelligence should collaborate with each other in order to better identify and name biological species.

Drawing on approaches from various research areas especially AI and robotics, intelligent and automation systems are playing a vital role in scientific research, allowing researchers to conduct experiments more efficiently. It is argued that the revolution of automation science would emerge in a few decades [7].

2.2 Evolutionary Computation

Evolutionary computation is a technique which draws inspiration from biological evolution. It is a stochastic search method and uses trial-and-error to guide the search process. We use evolutionary computation technique to automate generation of models during the system identification process. Section 2.2.1 briefly introduces the biological evolution. Section 2.2.2 details the principles, strengths, weaknesses of evolutionary computation, including evolutionary algorithms and coevolutionary algorithms. Section 2.2.3 presents three main applications of evolutionary computation and the related work.

2.2.1 Biological Evolution

Biological evolution is about how living creatures evolve to adapt to their changing environment. According to Darwin's Theory of Evolution [44], species fight for survival. The species that can fit the environment survive, and others that can not would die. This phenomenon is regarded as *survival of the fittest* or *natural selection*. There are heritage (e.g., through sexual reproduction) and random mutation among species' genes. The genes that help the species survive would have a higher chance of be preserved and passed on to the next generation, while the genes that are harmful or not useful would be abandoned.

Table 2.1: The relationship between different species

	the influence of species A		
the influence of species B	+,+ (reciprocity)	+,0 (symbiosis)	+,- (predation)
	0,+ (symbiosis)	0,0 (neutral)	0,- (amensalism)
	-,+ (predation)	-,0 (amensalism)	-,-(antibiosis)

Natural selection tends to reserve and accumulate small beneficial genetic mutations. Suppose that some members in a species have evolved a functional organism that is very useful (e.g., a wing that can fly). This makes these members easier to find food or avoid threat from predators. Their offspring are more likely to inherit such advantageous function and this function would be passed to the next generation. The other members without the advantageous function are more likely to die out. Natural selection helps the species to compete and adapt better in the environment. At the same time, it also accelerates the extinction of the species that can not fit the environment. The dinosaur used to be a dominant species in the ancient world due to its big body and flying ability. This was a big advantage when the climate was mild. However, as the climate changed dramatically (e.g., extremely cold or hot), the big body was no longer an advantage as it needed to consume much more energy. This led to accelerate the extinction of dinosaurs.

Coevolution is special form of evolution, which involves in the simultaneous evolution of two or more species. A typical example of coevolution is fox and rabbit or parasite and host. In nature, the survival ability between species is coupled. That means the survival ability of one particular individual in a species depends not only on its chromosome, but also the interaction with the individuals from other species. Although the correlation between different species is complicated, for a specific species, there are three possibilities: beneficial, injured and neutral. Therefore, through permutation and combination, the relationship between two species can be summarized in Table 2.1. It includes 6 concrete relationship: reciprocity, neutral, symbiosis, amensalism, predation and antibiosis [45].

Where, symbol '+', '-' and '0' represent beneficial, injured and neutral. For example, "+, -" indicates A benefits and B gets injured in the relationship.

2.2.2 Introduction of Evolutionary Computation

2.2.2.1 Principles

Based on the principle of biological evolution, a bi-inspired algorithm — genetic algorithm (GA) was proposed by Halland in 1960s [46]. GA simulates the heritage, mating, and mutation of the natural evolution. In GA, the solution for a given problem is represented as chromosome, which contains several genes. Each gene could be a binary number, integer, or floating-point number. Heritage is also called reproduction in GA. Mating corresponds to crossover/mate, in which different individuals exchange genes. Mutation is normally realized by randomly changing a particular gene. For example, some gene in the chromosome may be randomly replaced by another gene. Mutation in GA serves the same function as it is in natural evolution. It creates the diversity and creativity among the population. GA is driven by a fitness function, which defines the goal or solution to be achieved. The evolutionary process is to optimize (e.g., maximize) the fitness of the individuals in the population. There are other types of evolutionary algorithms that based on the basic idea of natural evolution. Fogel, Owens and Walsh invented *evolutionary programming (EP)* [47]. Rechenberg and Schwefel introduced *evolution strategies (ES)* [48], which are mainly dealing with real-value continuous optimization problems. In the early 1990s, another new evolutionary algorithm called *genetic programming (GP)* was presented by Koza [49]. Fig. 2.1 shows a brief classification of evolutionary algorithms.

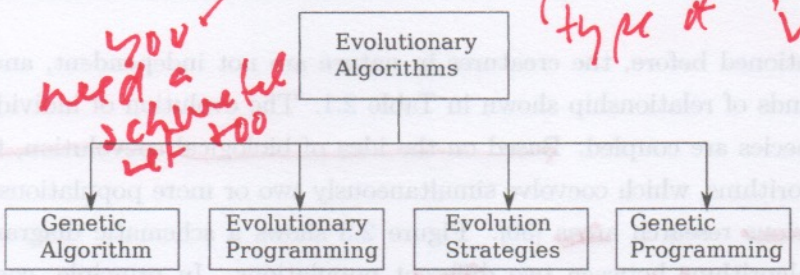

Figure 2.1: Classification of evolutionary algorithms

The implementation of evolutionary algorithms follow a general flow during the operation process. They can be divided into 5 steps: initialization, evaluation, mutation, selection and termination. At the beginning, a random population of individuals is initialized. These could be several random strings (chromosomes or individuals), each of which contains several genes (e.g., floating point number). These strings are encoded of the

solutions to be achieved. Different chromosomes are then evaluated using the predefined fitness function. The fitness of individuals in the population is only decided by their own chromosomes. That means in different generations, the fitness of individuals who own the identical chromosome is constant. Selection happens after the evaluation of each individual, and the ones with higher fitness normally have a higher chance of being selected to the next generation and have offsprings. The offsprings would go through mutation, which helps to maintain diversity of the whole population. Fig. 2.2 show a diagram of how the evolutionary algorithms proceed.

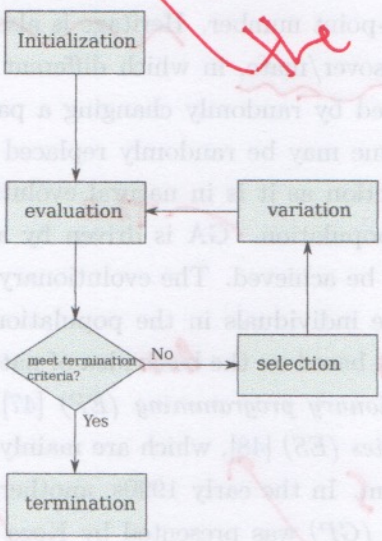

Figure 2.2: This diagram shows the flow of evolutionary algorithms.

As we mentioned before, the creatures in nature are not independent, and they have different kinds of relationship shown in Table 2.1. The evolution of individuals among different species are coupled. Based on the idea of biological coevolution, the coevolutionary algorithms, which coevolve simultaneously two or more populations, are widely used in various research areas [50]. Figure 2.3 shows a schematic diagram of coevolutionary algorithms between two different populations. In principle, coevolutionary algorithms can be considered of comprising several sub-algorithms, each of which could be an evolutionary algorithm. These sub-algorithms interact with each other in the fitness calculation process. In other words, the fitness of individuals in one population not only depends on its own chromosome, but also on the performance of other individuals from another population during the coevolutionary process.

The essential difference between evolutionary algorithms and coevolutionary algorithms

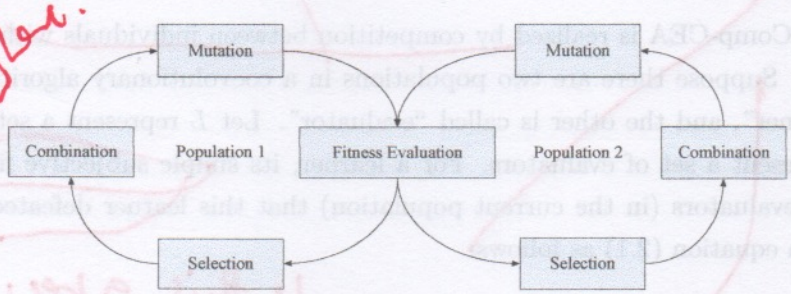

Figure 2.3: A schematic diagram of coevolutionary algorithms between two different populations.

is the way of fitness evaluation. While the fitness of an individual in evolutionary algorithm depends only one its chromosome, in coevolutionary algorithms the individual's fitness depends on its performance when evaluated by the individuals from the other populations. According to the way of evaluation, the coevolutionary algorithms are generally divided into two categories—competitive coevolutionary algorithm (Comp-CEA) and cooperative coevolutionary algorithm (Coop-CEA). Comp-CEA assesses each individual by its competitive performance with respect to its opponents, while Coop-CEA assesses each individual by its cooperative performance with respect to its co-operators. As discussed by Dawkins and Krebs [51], competitive coevolution can produce a phenomena of "arm races" by increasing the complexity of each population in the coevolution. The evolution of one population may drive another population to evolve new strategies, which makes both populations evolve a higher level of complex behavior. Generally, Coop-CEA is applied to the situation in which the problem can be divided into several sub-problems. In Coop-CEA, there are several cooperative species evolving simultaneously, and each sub-species represent a part of the whole solution, which is the combination of the eventual solution in each sub-species according to a certain sequence. In the rest of the thesis, we only discuss Comp-CEAs, which will also be referred to as coevolutionary algorithms in general.

In coevolutionary algorithms, the fitness of individuals is called subjective fitness [52]. An individual's subjective fitness is based on the performance of its temporary opponents from the current generation or a combination of current and past generations. The fitness of individuals with the same chromosome in different generations may vary because of changing opponents. Conversely, the fitness in evolutionary algorithms is called absolute or objective fitness.

Comp-CEA can also be applied in single population or multiple populations. The single-

population Comp-CEA is realized by competition between individuals within the same population. Suppose there are two populations in a coevolutionary algorithm. One is called "learner", and the other is called "evaluator". Let L represent a set of learners and E represent a set of evaluators. For a learner, its simple subjective fitness is the number of evaluators (in the current population) that this learner defeated [53]. It is described in equation (2.1) as follows:

$$\forall i \in L \Rightarrow CF_i = \sum_{j \in E, i defeats j} 1 \tag{2.1}$$

CF_i is the fitness of learner i.

Another fitness calculation approach is called competitive fitness sharing [50] as described in the following:

$$\forall j \in E \Rightarrow N_j = \sum_{k \in L, k defeats j} 1 \tag{2.2}$$

$$\forall i \in L \Rightarrow CF_i = \sum_{j \in E, i defeats j} \frac{1}{N_j} \tag{2.3}$$

N_j represents the number of learners that could defeat evaluator j.

In competitive fitness sharing [50], the learner that could defeat the more competitive evaluator get higher reward. For example, if a learner, i, in a population is the only individual to defeat an evaluator, j, this learner's accumulative fitness is added by 1, as N_j is equal to 1 in Equation (2.3). The aim of using competitive fitness sharing is to preserve/award the learner that possesses important genetic materials which are worth passing to the next generation.

There are many ways to choose the evaluators (temporary opponents). Random Paring [54] means finding a random temporary opponent for each leaner. In single elimination tournament [55], all the individuals randomly match, and the losers are taken out and winners are selected into next round of random match. Round Robin [54] means all the evaluators are the temporary opponents of each leaner. There are also other ways such as K-random opponent [55] and fitness sampling [50]. In fitness sampling, the selected

temporary opponents should have relatively higher fitness value in the last generation. The evaluation time for random paring is the shortest, but the performance is the worst; the calculation time for round robin is the longest, but the performance is the best. In our thesis, we chose to use the simple subjective fitness calculation and Round Robin.

2.2.2.2 Strengths

The primary advantage of evolutionary computation is it is conceptually simple. Evolutionary computation draws inspiration from biological evolution, and it can be implemented using simple operators as mentioned before. Moreover, evolutionary computation is task-independent, which means it can be applied to virtually any task that can be formulated as a function optimization problem [56]. The advantage of function optimization as well as black-box optimization will be detailed in Section 2.2.3.1. Evolutionary computation (especially evolutionary algorithms) can be run in parallel (e.g., using GPU computing) to accelerate the evolutionary process. Each individual can be evaluated independently according to the predefined fitness function. Only the selection and recombination process need the serial computing. Evolutionary computation is robust to changes of circumstances. Once the circumstance has changed, the evolved solutions can be used as a start for further development without the need to restarting the whole process [57]. Apart from the general advantages of evolutionary computation, when compared with evolutionary algorithms, coevolutionary algorithms have the following two advantages:

- *Open-Ended Evolution* Coevolutionary algorithms can create an open-ended evolution for each population due to the complex interaction between the competing populations during the coevolutionary process. Such open-ended evolution could encourage the appearance of new building blocks, thus maintaining the diversity of populations. In Darwin's natural selection, this phenomenon is referred to as "arm race" [51], which leads each species to continuously improve.
- *No Absolute Fitness Needed* Coevolutionary algorithms can be applied to solve problems in which absolute fitness can not be effectively defined. For example, when evolving a chess program, it would be challenging to define a fitness to determine which program is better. An effective way of evaluating a chess program is making it play with other programs and then calculating its subjective fitness [53, 58].

2.2.2.3 Weaknesses

The main weakness of evolutionary computation is that it requires a sufficient number of generations to obtain good solutions. In simulation, this would not be a problem; however when the evaluation needs to be conducted on the real system, it would take a long time and may also cost a lot. If the fitness function can not be defined properly, the obtained solutions may be biased. Although coevolutionary algorithms have some advantages over evolutionary algorithms, there are three main pathologies:

- *Red Queen Effect* During the coevolutionary process, two populations keep competing with each other. For a particular population, when *Red Queen Effect* happens, the variation tendency of subjective fitness and objective (absolute) fitness is opposite [59]. For example, the increase of its subjective fitness may correspond to the decrease of its objective fitness. In other words, the objective fitness does increase, but the landscape of subjective fitness does not reflect such situation.

- *Cycling* In coevolutionary algorithms, the aim of each individual is to defeat its temporary opponents. The optimal solution which is obtained in the previous generations would probably be lost. After some generations, the optimal solution may be found but lost again. The coevolution is trapped into this endless cycling, failing to find the optimal solution [52]. The general way of overcoming the problem of "cycling" is "hall of fame"[50]. "Hall of fame" obtains the excellent individuals from the previous generations, and these obtained individuals may be selected to be temporary components to evaluate the individuals from the competing population in the current generation.

- *Disagreement* During the coevolutionary process, when one species is entirely better than the other, disengagement will occur. In this case, the selection criteria won't make any sense and selection gradient will disappear, since the subjective fitness of each population is constant. The diversity of populations will converge into zero, and it is impossible to form "arm race" [51]. The common solution for solving the disengagement problem is "resource sharing" and "reducing virulence"[52]. "Resource sharing" keeps the diversity of populations, and "reducing virulence" selects the evaluators to keep the gradient of learners' fitness.

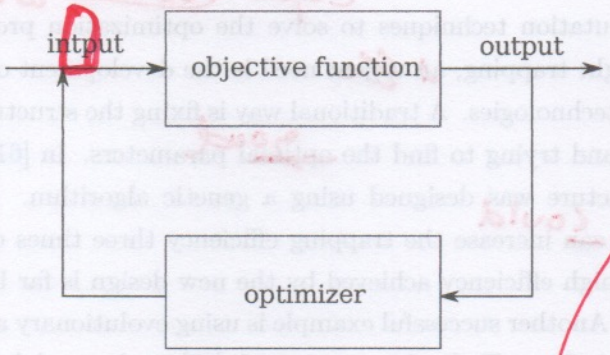

Figure 2.4: This diagram shows the process of black box optimization. The task is to find a candidate solution (input) that can optimize (e.g. maximize or minimize) the objective function.

2.2.3 Applications of Evolutionary Computation

Evolutionary algorithms are widely used for solving various engineering tasks ranging from optimization, control, pattern recognition, robotics and system identification/modelling. In the following sections, we will focus on three main fields in which evolutionary computation technique plays a role.

2.2.3.1 Black Box Optimization

One of the promising application of evolutionary computation is black box optimization. Black box optimization refers to such problems that the optimization algorithm aims to optimize an objective function without assuming the hidden structure of that function (e.g., linear or differentiable). In particular, the aim is to found a set of inputs that can maximize or minimize the output of the objective function. For example, in a traveling salesman problem, in which the task is to find the best combination of route of cities (input) that can minimize the length of tour of visiting all cities. Fig. 2.4 shows a diagram of the black box optimization.

Due to the complexity of the real problem in nature, the stochastic search algorithms such as evolutionary algorithms provide us an efficient and relatively 'perfect' solutions. Although evolutionary algorithms could not guarantee the best solution would be found every time, they are still superior to many traditional search algorithms such as the greedy local search algorithm [60]. There are many real-world examples that using

evolutionary computation techniques to solve the optimization problem. In the area of nanophotonic light trapping, an urgent need is the development of low cost thin film solar photovoltaic technologies. A traditional way is fixing the structure according to the physical intuition and trying to find the optimal parameters. In [61], a highly efficient light-trapping structure was designed using a genetic algorithm. It was shown that this new structure can increase the trapping efficiency three times comparing with the classic limit. The high efficiency achieved by the new design is far beyond the reach of traditional design. Another successful example is using evolutionary algorithms to design antennas for NASA's Space Technology spacecraft [57], and one of the antennas was used in the mission. The antennas is a critical device for the spacecraft to communicate with the ground, as faulty communication may cause a lot of data lost or the crash of the spacecraft. The antennas designed using evolutionary algorithms are significantly better than those designed by human experts.

In the following, we list several cases (but not all) that evolutionary computation could be applied to solve the 'tough' optimization problems.

- *High-dimension* As the dimension, n, of the objective function increases, the search space increases exponentially. This is called "curve of dimension" by Bellman [62]. For example, if we have to optimize a function that has 30 dimensions, and each dimension only has 20 parameters to be selected. For a grid search in which all the possible solution is evaluated, it will take 20^{30} evaluations. Suppose that each evaluation takes $1\mu s$, it would more than the $3 \cdot 10^{31}$ years. However, if using evolutionary computation, it probably takes hours to find the optimal solution.
- *Multi-Model* Multi-model means a system (function) has more than one optimums (e.g., Rastrigin Function). The one/s with the best fitness value is/are considered as global optimum/s, and the other are considered as local optimums. These local optimums around the global optimum/s are very misleading for the gradient-based search algorithms (e.g., *hill climbing* algorithm), as the solutions may easily get trapped at the local optimums. Evolutionary algorithms are shown to be very efficient to find the global optimum/s [63].
- *Non-separable and non-differential* A function, $g(x_1, x_2, \cdots, x_n)$ is non-separable, if it can not be expressed as: $g(x_1, x_2, \cdots, x_n) = g(x_1)g(x_2)\cdots g(x_n)$. For the separable function, it would be much easier to optimize, as we can treat each variable separately. However, for non-separable system, the variables are normally