OPPOSITIONAL BIOGEOGRAPHY-BASED OPTIMIZATION

MEHMET ERGEZER

Bachelor of Engineering in Electrical and Computer Engineering

Youngstown State University

May, 2003

Master of Science in Electrical and Computer Engineering

Youngstown State University

May, 2006

submitted in partial fulfillment of the requirements for the degree

DOCTOR OF ENGINEERING

at the

CLEVELAND STATE UNIVERSITY

May 2014

We hereby approve the dissertation

of

Mehmet Ergezer

Candidate for the Doctor of Engineering degree.

This dissertation has been approved for the

Department of Electrical and Computer Engineering

and CLEVELAND STATE UNIVERSITY

College of Graduate Studies by

_____

Dan Simon, Dissertation Committee Chairperson

_____

Department/Date

_____

Murad Hizlan, Dissertation Committee Member

_____

Department/Date

_____

Hanz Richter, Dissertation Committee Member

_____

Department/Date

_____

Iftikhar Sikder, Dissertation Committee Member

_____

Department/Date

Sailai Shao, Dissertation Committee Member

Department/Date

Dan Simon, Doctoral Program Director

Department/Date

Chansu Yu, Department Chair

Department/Date

Student‘s Date of Defense

ACKNOWLEDGMENTS

THERE are many people that I would like to thank for their help and support to help me become the person that I am today. Foremost, I must acknowledge my adviser Dr. Simon for always leading by example, through his hard work and ethical and optimal decision making skills. I am forever grateful to him for his mentoring style of encouraging investigative thinking and allowing us the freedom to do research.

It gives me great pleasure in acknowledging the support and help of all my committee members, Dr. Hizlan, Dr. Richter, Dr. Sikder and Dr. Shao for all their suggestions and recommendations.

Special thanks to Dawei Du, Rick Rarick, George Thomas, Berney Montavon, Paul Lozovyy and all of the remaining past and present members of the Embedded Controls Lab. Being part of a team that solves real-world problems has been one of the most rewarding experiences of my life. Also, many thanks to our industrial partners, including Jeff Abell from General Motors, Nick Mastandrea from Innovative Developments and Arun Venkatesan from Cleveland Medical Polymers, for working with us on these projects.

I consider it an honor to work with my fellow engineers at ARCON Corporation. I am indebted to them for encouraging me the complete my research.

Last, but not least, I owe a debt of gratitude to my family, especially my parents Güngör and Dr. Yalçın Ergezer. I could not achieve this without their unconditional love, encouragement and guidance. I must also thank Slava and her parents for welcoming me and tolerating me through this long adventure and for providing me with the much needed relief from the technical world.

OPPOSITIONAL BIOGEOGRAPHY-BASED OPTIMIZATION

MEHMET ERGEZER

ABSTRACT

THIS dissertation outlines a novel variation of biogeography-based optimization (BBO), which is an evolutionary algorithm (EA) developed for global optimization. The new algorithm employs opposition-based learning (OBL) alongside BBO migration to create oppositional BBO (OBƃO). Additionally, a new opposition method named quasi-reflection is introduced. Quasi-reflection is based on opposite numbers theory and we mathematically prove that it has the highest expected probability of being closer to the problem solution among all OBL methods that we explore. Performance of quasi-opposition is validated by mathematical analysis for a single-dimensional problem and by simulations for higher dimensions. Experiments are performed on benchmark problems taken from the literature as well as real-world optimization problems provided by the European Space Agency. Empirical results demonstrate that with the assistance of quasi-reflection, OBƃO significantly outperforms BBO in terms of success rate and the number of fitness function evaluations required to find an optimal solution for a set of standard continuous domain benchmarks.

The oppositional algorithm is further revised by the addition of fitness-dependent quasi-reflection which gives a candidate solution that we call $\hat{x}_{Kr}$. In this algorithm, the amount of reflection is based on the fitness of the individual and can be non-uniform. We find that for small reflection weights, $\hat{x}_{Kr}$ has a higher probability of being closer to the solution, but only by a negligible amount. As the reflection weight increases, $\hat{x}_{Kr}$ gets closer (on average) to the solution of an optimization problem as the probability of being closer decreases.

In addition, we extend the idea of opposition to combinatorial problems. We introduce two different methods of opposition to solve two types of combinatorial optimization problems. The first technique, open-path opposition, is suited for combinatorial problems where the final node in the graph does not have be connected to the first node such as the graph-coloring problem. The latter technique, circular opposition, can be employed for problems where the endpoints of a graph are linked such as the well-known traveling salesman problem (TSP). Both discrete opposition methods have been hybridized with biogeography-based optimization (BBO). Simulations on standard graph-coloring and TSP benchmarks illustrate that incorporating opposition into BBO improves performance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

MANY engineering problems involve nonlinearities or other complexities which render mathematical methods and even local optimization algorithms futile. However, nature has become an expert in “optimizing" difficult, convoluted problems through evolution. Evolutionary computing (EC) attempts to replicate nature’s success by representing solutions as encoded individuals and allows them to evolve through a selection mechanism. Where mathematics can guide toward a unique solution, solutions provided by nature are diverse. For instance, deer are known to have 34 species and many more subspecies. This variety, represented by their coats, size or antlers, enables them to adopt to various diets, predators and landscapes. While a white-tail deer hides and sprints away from predators, mule deer pronks away by jumping using all four feet. Similarly, EC allows global minimization by creating a population of solutions that are robust and adaptive. These solutions may not be perfectly optimal but they are evolved to be suitably fit solutions to the optimization problem. As stated by an anonymous quote “Perfection would be a fatal flaw for evolution. Life’s hold on life depends on God losing his grip on life every once in a while."

This chapter outlines the purpose of this dissertation. Section 1.1 in-

troduces evolutionary computation, presents its history and common applications. Section 1.2 gives an overview of biogeography-based optimization as an evolutionary algorithm and Section 1.3 discusses opposition as a tool for optimization. Section 1.4 lists the pseudo code for oppositional biogeography-based optimization. The motivation for this research is discussed in Section 1.5 and the problem statement is broached in Section 1.6.

1.1 Evolutionary Computation

Evolutionary computation is an umbrella term, that is, a hypernym, conceived in 1991 [1] to unite the various evolutionary techniques that were being simultaneously developed around the world. This section will discuss the development of EC, present an overview of its methodology, explore some controversies in academia (namely the No Free Lunch Theorem) and its applications as reported in today's literature.

1.1.1 History of Evolutionary Computation

The evolution of evolutionary computation can be summarized as follows.

- Evolutionary simulations:
  - 1954: The first implementation of EC is commonly credited to Baricelli [2], who modeled cells migrating in a grid and competing for survival.
  - 1958: On the opposite corner of the world, in Australia, another researcher [3] modeled sexual reproduction by recombining solutions.
- Evolutionary algorithms:
  - 1962: David Fogel developed evolutionary programming (EP) [4] in order to replicate intelligent behavior by predicting the environment.

In *Artificial Intelligence through Simulated Evolution*, he explains [5]:

> Intelligent behavior is a composite ability to predict one's environment coupled with a translation of each prediction into a suitable response in light of some objective.

EP relies solely on mutation for reproduction, not on recombination, and applies tournament style selection based on fitness. Also, unlike most other EAs, EP enables population size to evolve.

- In 1962, Holland published an article outlining a theory of adaptive systems [6]. Later, he published *Adaptation in Natural and Artificial Systems* [7] which was instrumental in the development of genetic algorithms (GA). In GA, solution candidates were represented as chromosomes in a DNA in binary code and evolved by single point crossover and mutation. Holland's GA gained popularity in part due to his Schema Theorem [8], also referred as the Fundamental Theorem of Genetic Algorithms: "Short, low-order schemata with above average fitness increase exponentially in successive generations."
- 1964: Evolution strategies (ES) [9, 10] was designed by three students as an automatic parameter selection algorithm for a laboratory experiment to minimize the drag in wind tunnel [11]. During the laborious experiment, researchers discovered that heuristic search outperformed a discrete gradient-oriented method. They applied their algorithm to 2D and 3D air flow [12] and 3D hot water nozzle problems [13]. Their proposed "cybernetic solution path" algorithm had two rules [14, 15]:
  - Mutation: "Change all variables at a time, mostly slightly and at random."
  - Survival of the fittest: "If the new set of variables does not diminish the goodness of the device, keep it, otherwise return to

the old status."

- Swarm intelligence:
  - Ant colony optimization was first published as Dorigo's PhD dissertation in 1992 [16]. He was inspired by the probabilistic behavior of ants [17] and specially the double bridge experiment [18]. In this experiment, a colony of ants must cross back and forth one of the two bridges to collect food from the other side. In time, ants converge to the shorter path by following the concentration of pheromone left behind by the previous colonists. Goss et al. [18] also proposed a mathematical model for the probability of an ant choosing a bridge based on the previously made decisions by the ants.
  - 1995: Particle swarm optimization [19, 20] is a swarm intelligence method [21] that is based on the models of bird flocking [22]. It was originally designed to model social behavior where subjects altered their perspectives to better fit in with their peers. It has later been simplified to a heuristic optimization algorithm where each particle's velocity determines its position based on information received from its neighborhood.
- Miscellaneous EC methods:
  - Differential evolution (DE) is developed by Storn in 1995 [23, 24] and is considered to be a robust EA for avoiding premature convergence found in GA [25, 26]. In DE, an individual is created based on the weighted difference of two other solution candidates added to a third random solution candidate [27]. If this new individual is more fit than an individual randomly selected from the current generation, it replaces that individual. Performance of DE depends on the selected weight parameters. Reference [28] proposes a set of weights for DE based on the problem dimension and the number of fitness

evaluations.

- Genetic programming (GP) is born in 1985 when Cramer created an algorithm that develops simple sequential programs [29]. He utilized GA to manipulate tree-like structures that represented randomly generated functions. His work was later expanded by Koza to evolve more complicated programs [30, 31, 32]. GP has evolved from being solely a program creator. It is also a popular method for automatic circuit design where given a set of requirements, GP generates the desired circuit routing, placement and size [33, 34].
- Simulated annealing (SA) is independently developed by two scientists in the mid-1980s [35, 36] and is a generalization of the Metropolis-Hastings algorithm (MH) [37]. MH is a Monte Carlo method that allows sampling from a probability distribution and only requires density function evaluation. Annealing is the process of heating a thermodynamic system and then slowly cooling it. The goal of SA is to minimize the system's energy by moving from current state s to a neighboring state based on an acceptance probability function which depends on states' energies and a global decay parameter that represents the temperature. SA began as an optimizer for combinatorial problems [35, 38, 39] and its variations include quantum annealing [40] and stochastic tunneling [41].
- Tabu search (TS), published by Grover in 1985 [42, 43, 44], explores the neighborhood of an individual in search of a more fit solution while remembering a list of recently visited neighbors, marked as taboo, to avoid revisiting them. Therefore, if the algorithm is stuck in a local minima, instead of retreating, it is forced to explore in a new direction. TS can solve combinatorial problems including graph coloring [45, 46, 47].

1.1.2 Evolutionary Computation Methodology

Biomimicry, drawing inspiration from nature for developing new technology, is now employed in many scientific fields. Recently, NBD Nano has designed a water bottle that refills itself by extracting moisture from the air [48]. This technology imitates the Namib Desert beetle's wings' coating which catches the water from the morning fog. However, this is not the first example of biomimicry. In the fifteen century, Leonardo Da Vinci studied birds' anatomy to design his flying machine [49]. Many of today's inventions, from Velcro to nose of Shinkansen (Japan's bullet train), mimic solutions from nature [50]. Universities and corporations have started research centers for nature inspired future development ideas [51, 52]. As seen by EC's history, many evolutionary algorithms and other machine intelligence learning methods are also inspired by nature. For example, genetic algorithms (GA) [8] mimic evolution, ant colony optimization (ACO) [16] approximates animal behavior in colonies, and artificial neural networks [53] are modeled after the biological nervous system. Other examples include particle swarm optimization [19], artificial immune systems [54] and hill climbing [55]. The majority of these EAs follow a similar methodology which could be outlined as:

- Initialize population
- Selection
- Recombination
- Random variation

Generally, the process starts by creating an initial random population of possible solutions. The population is then processed in a way which is motivated by the natural model. Based on this natural model's properties, such as genetic inheritance and survival of the fittest, the population will evolve and adapt to its environment while attempting to get closer to the solution after

each generation. The algorithms generally quit once an acceptable solution is found or when the available computing resources are exhausted.

1.1.3 Controversies and No Free Lunch Theorem

One of the benefits that made EC popular is that it can be applied to various types of problems. However, generally, EAs are not modified to match the cost functions of the problem at hand and the same search algorithm is used regardless of a problem's particulars. Reference [56] shows that the differences in cost functions are crucial. The authors prove that when we ignore the particular biases or properties of a cost function, the expected performance of all algorithms over all cost functions is precisely identical. This is called the No Free Lunch Theorem (NFL).

Their main theorem is that the probability of obtaining a particular histogram of cost values given a specific number of cost evaluations is independent of the algorithm used given that we have no prior information about the optimization problem. This implies that if we have no prior knowledge about the cost function, the expected performance will be independent of the chosen algorithm. The theorem relies on the assumption that since nothing is known about the cost functions, then, on average, all cost functions have the same probability distribution. They further conclude that the expected distribution of the histogram will be the same regardless of the selected algorithm. Therefore, the EA should be chosen based on the distribution of the cost function.

The theorem is named No Free Lunch Theorem (NFL) and is applied to search [56], supervised learning [57, 58] and optimization [59]. Further development lists the necessary conditions for NFL [60, 61]. NFL theorem created controversy about the credibility of EC [62, 63]. However, not everyone agrees with NFL's applicability to real-world problems. Reference [64] disputes the validity of the NFL in black box scenarios and proposes the Almost No Free Lunch Theorem.

1.1.4 Evolutionary Computation Applications

EC has been used to assist in solving countless problems in a variety of fields from geophysics to financial markets. This section will discuss some of this research. In aerospace engineering, EC has been applied to wing shape design of an aircraft [65, 66] and maneuvers of a spacecraft while minimizing time [67, 68]. In chemistry, it has been used in the design of new molecules to meet given set of specifications [69] and creating new antimicrobial compounds for cleaners [70]. Another area where EC is applied is control systems. It has been employed in online controller design [71] and many offline ones including linear quadratic-Gaussian and H_∞ control [72, 73, 74], as well as control of chemical reactors [75, 76]. EC has been utilized for motion planning in robotics [77, 78, 79, 80] and network design in communications [81, 82, 83]. In finance, EC has been employed for bankruptcy [84, 85, 86] and stock predictions [87, 88, 89]. In geophysics, EC has been applied to seismic wave inversion [90, 91, 92] and groundwater monitoring [93, 94]. Holland and Miller draw a parallel between economic systems and complex adaptive systems and employ artificial adaptive agents to predict economic phenomena [95]. Another popular application is protein building and folding simulations in biology [96, 97, 98]. Reference [99] develops fuzzy rules tuned by EA for linguistic modeling. This list can be expanded to add materials science, law enforcement, data mining and countless other fields. As one can see, EC has been improving our lives in various fields, no less than any other established science. One can expect it to have even more applications in the future as the theory behind it is further developed and as computing power continues to become cheaper.

1.2 Biogeography-based Optimization

Biogeography-based optimization (BBO) is a generalization of biogeography to EC. Island biogeography helped the development of evolutionary theory and is a compelling area of study because islands are discrete environments.

That is, they sustain their own distinctive organisms and they are numerous. There are more islands than there are continents or oceans [100]. Due to the variations they provide (size, ecology, length and degree of isolation), islands can offer the necessary tools for studying evolution. Charles Darwin, credited as the formulator of the theory of evolution, conceived his hypotheses on natural selection after studying/eating giant tortoises on the Galapagos Islands [101].

Seeing that biogeography helped the development of theory of evolution, it stands to reason that biogeography would be a solid candidate for evolutionary computation. Population biology studies the impact of immigration, emigration and extinction on the number of species. BBO is modeled after the immigration and emigration of species between islands. The fitness of each island is measured by its habitat suitability index, HSI [102]. A habitat with a high HSI indicates a desirable living environment in biogeography and a good solution in BBO. This type of habitat will host many species and spread its species frequently to other habitats. Because a high HSI island hosts a large number species, it will be harder to immigrate there and this type of solution will be less susceptible to alterations and therefore its HSI will remain more static throughout many generations.

On the other hand, a habitat with a low HSI will be hosting a limited number of species and these species will have a lesser chance of being accepted to other islands. It will be very easy for the species from other islands to migrate to low HSI habitats. Therefore, the species distribution on low HSI islands will change more frequently.

The independent variables of the HSI are called the suitability index variables (SIVs). SIVs are the climatic and topographic features offered by the island and can include such factors as precipitation, temperature, elevation and slope.

Fig. 1 illustrates linear BBO immigration and emigration curves. In this figure, the estimated solutions are sorted by fitness from worst to best. The

worst solution candidate, with a low HSI, has the highest immigration rate; hence, it has a very high chance of borrowing features from other solutions, helping it to improve for the next generation. The best solution candidate, with a high HSI, has a very low immigration rate, indicating that it is less likely to be altered by the other individuals. The emigration rate works in the opposite direction.

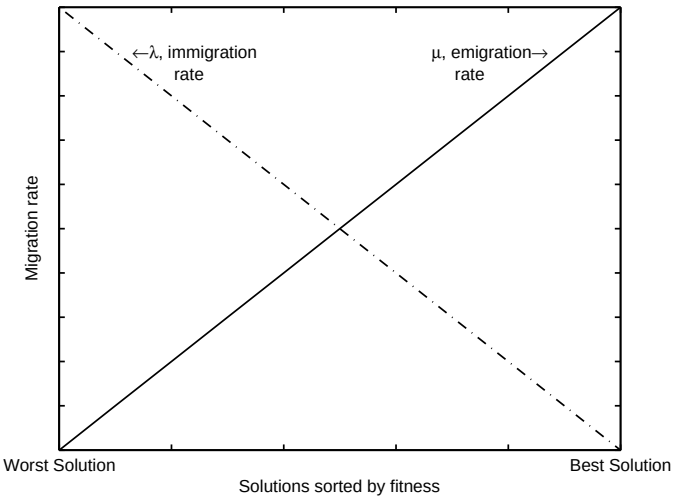


Figure 1. Linear migration rates plotted against the sorted population. Better solution candidates possess a low immigration rate and a high emigration rate.

BBO migration functions are programmed as described above. The other area of biogeography, extinction, is implemented indirectly. When fitter species immigrate to an island, lesser fit species must go extinct to accommodate the new ones. However, note that emigration in BBO does not symbolize a move, but rather a copy. For example, if a feature in island 1 migrates to island 2, then both islands 1 and 2 have this feature. The worst solution candidate is assumed to have the worst features; thus, it has a very low emigration rate and a low chance of sharing its features. On the other hand, the fittest solution candidate has the best features and the highest probability of sharing them.

One of the distinguishing features of BBO is that the original population is not discarded after each generation; rather, it is modified by migrations

and continues to survive. Also, when updating the population, BBO considers the fitness of the immigrating and emigrating islands via the emigration and immigration curves.

Mathematical modeling and convergence properties of EC are still being investigated as modeling the dynamics of an adaptive system is difficult. One approach to confirm convergence is to formulate the EA as a finite state Markov chain. While [103] derived the necessary conditions for asymptotic convergence to optimum for GA and ES, [104] proved their convergence. This proof is accomplished by finding the limit of the probability of nearing the global optimum as the number of iterations goes to infinity. The proof illustrates that EA will be in a certain vicinity of the optimal point with a probability of 1. However, practicality of this proof is rather limited in the real world as it assumes infinite time for convergence. Reference [105] extends this work to BBO and derives the limiting probabilities for all possible population distributions.

Despite being a new algorithm, BBO has already been implemented in many fields of engineering. It has been applied to the power flow problem [106, 107, 108, 109], economic dispatch [110, 111, 112], image classification [113, 114, 115], communications [116, 117, 118] and robotics [119, 120, 121]. While statistical foundations for BBO are being developed [122, 123], BBO has been combined with other EAs such as ES [124], DE [125], PSO [120], and flower pollination by artificial bees [126] to form hybrids. In addition, it has been utilized to optimize other EC methods, such as fuzzy [127] and neuro-fuzzy [128] systems.

1.3 Opposition-based Learning

In this section we discuss the numerous definitions of opposition in various areas of culture and science, and explain how it can be applied to optimization problems. Study of opposition has been going on for millennia. The opposite forces have been studied by humanity for a long time on a philosophical

level. Dualities found in many religions are an example of this. Dualities have different interpretations in different cultures. In Taoism, yin-yang (shown in Fig. 2) reflects the harmony of opposite forces and seeks balance in complementary forces. Two ancient Persian religions, Zoroastrianism and Manichaeism, are also considered dualistic. Manichaeism was one the most predominant religions of its time, spreading from Roman Empire to China. In Manichaeism, dualism existed as a struggle between good and evil. As Manichaeism gained popularity, it was declared a heresy in Christianity, oppressed by Islam and forbidden in China by Ming dynasty.

What might have started as a theological debate (yin vs. yang and good vs. evil), still exists today in the scientific world. In electrical engineering, duality refers to the relationship between capacitance and impedance or open and short circuits. In mechanical engineering, duality indicates the relationships between stress and strain, stiffness and flexibility. In magnetism, the dual of magnetic field is the electric field and the dual of permittivity is permeability. Furthermore, in mathematics, duality is studied in logic, set and order theories.

Another example of opposition in today's scientific world is the study of antimatter. Physicists believe that all particles have a mirror image in the universe, called antimatter. International groups of researchers at CERN are conducting the world's most expensive science experiment to create such antiparticles. They believe that studying and experimenting with antimatter will allow them to test the doctrine of modern physics and standard model of particle physics [129]. This research is so crucial to the field that based on its outcomes "the textbooks ... [may] have to be rewritten," according to Jeffrey Hangst from CERN [130]. Even though we do not fully understand antimatter, certain applications of it are seen in today's technology (for example, in medicine, anti-electrons are used for tomography scanning).

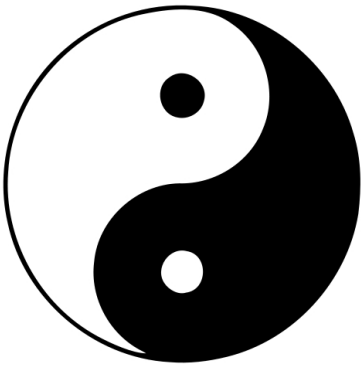


Figure 2. Yin-yang representing the harmony of opposing forces in Eastern philosophy

Opposition is encountered in different fields under different names. In Euclidean geometry it is referred as inverse geometry, in physics it is the parity transformation and in mathematics, it denotes reflection. All of these definitions involve isometric self-mapping of a function. Other examples include astronomy where planets that are 180° apart are considered to be opposing each other. Opposites also have a significant meaning in semantics as generalization of antonyms. Where antonyms are limited to gradable terms, such as thin and thick, the term opposite can be applied to gradable, non-gradable and pesudo-opposite terms.

The idea of OBƎO is derived from opposition-based learning (OBL). The creators of the OBL believe that a shortcoming of natural learning is that it is time consuming since it is modeled after a very slow process. For instance, it requires countless life cycles for species to evolve. On the other hand, human society progresses at a much faster rate via “social revolutions." Hence, the learning process could be improved based on such a model. Describing revolutions as fast and fundamental changes, whether in politics, economics or any other context, Tizhoosh maps this theory to machine learning and proposes to use opposite numbers instead of random ones to quickly evolve the population [131].

The main principal of OBL is to utilize opposite numbers to approach

the solution. The inventors of OBL advocate that given a random number, generally, its opposite has a higher chance of being closer to the solution than a random point in the search space. Thus, by comparing a number to its opposite, a smaller search space is needed to converge to the right solution(s). In this research, we develop the proofs measuring the effectiveness of opposite points against random numbers.

OBL has its roots in reinforcement learning [132, 133] and has been applied to various soft computing methods such as neural networks [134, 135, 136, 137] and fuzzy systems [138, 139]. To date, OBL has been employed to accelerate the convergence properties of numerous evolutionary algorithms such as differential evolution [140, 141, 142, 143], particle swarm optimization [144, 145, 146, 147], ant colony optimization [148, 149] and simulated annealing [150] in a wide range of fields from image processing [151, 152, 139] to system identification [153, 154].

The algorithm is implemented as two functions. The first one is called only once per simulation during initialization to create the initial population. This function compares the initial random population and its opposite to select the most fit among them. The second function is called every J_r generations, where J_r, jumping rate, is a control parameter set by the user to jump, or skip, opposite population creation at certain generations. Since the opposition function is called twice, OBƁO is classified as an “initializing and somatic explicit opposition-based computing algorithm" [155]. Because the opposite population’s fitness has to be evaluated, OBƁO will have to converge faster than original BBO (in terms of generation count) in order to maintain the same CPU load. A benchmark method based on number of cost function calls is introduced in Section 3.1 to take this into consideration.

1.4 Algorithms

In this section, we provide an outline of the main function which evaluates the EA and opposition algorithms, as well a brief overview of each EA. Whether we employ GA, DE or BBO as the optimization algorithm, Algorithm 1 is used to seek for the global minimum.

Algorithm 1 Pseudocode for EA with opposition where rand $\in [0, 1]$ is a uniform random number

1: ***Main EA Function***
2: Create an initial random population
3: Replace duplicate individuals with random ones
4: Calculate the cost of each individual
5: Sort the population
6: Execute the opposition algorithm (Algorithm 2)
7: **while** Optimal solution is not found or cost evaluation limit is not reached **do**
8: Perform EA selection/recombination (Algorithms $4-6$)
9: Replace duplicate individuals with random ones
10: Ensure that each individual is valid
11: Calculate the cost of the updated individuals
12: Sort the population
13: **if** rand $\leq$ opposition jumping rate **then**
14: Execute the opposition algorithm (Algorithm 2)
15: **end if**
16: Apply elitism by replacing the worst of current generation with the best of the previous generation
17: Ensure that each individual is valid
18: Sort the population
19: **end while**

In Algorithm 1, line 8 creates a function call for the desired EA: GA, DE

or BBO which are described in following subsections and Algorithms 4, 5 and 6. Line 14 calls the opposition opposition function as outlined in Algorithm 2.

Algorithm 2 Pseudocode for opposition logic

1: ***Opposition Function***
2: Create an opposite population, $\hat{x}_o$, $\hat{x}_{qo}$, $\hat{x}_{qr}$ or $\hat{x}_{Kr}$, as defined in Chapter II
3: Calculate the cost of each opposite individual
4: Select the fittest individuals amongst the EA and opposite populations
5: **return** $Fittest\ Individuals$

1.4.1 Genetic Algorithms

GA is one of the most popular EA and many variations of it exist in the literature [156]. We employ GA with uniform crossover and roulette-wheel selection as described in Algorithms 3, 4. The probability of selection with roulette wheel is directly proportional to each individual's fitness. The crossover rate is set to 50%; thus, on average, each child will have half of each parent's genes.

Algorithm 3 Pseudocode for roulette-wheel selection of parents

1: ***Roulette-Wheel Function***
2: Cumulative sum of all costs, Σ_c
3: Running sum, Σ_s = 0
4: **for** Each Solution Candidate, S **do**
5: $\Sigma_s = \Sigma_s + Cost(S)/\Sigma_c$
6: **if** rand(0,1) $< \Sigma_s$ **then**
7: Parent$_i = S$
8: **end if**
9: **end for**

Algorithm 4 Pseudocode for one generation of genetic algorithm function

GA Function
Select parents using roulette-wheel (Algorithm 3)
Produce children:
for Each Pair of Parents, P_1 and P_2 **do**
    for Each Problem Dimension, d **do**
        if rand(0,1) $<$ Crossover rate **then**
            $C1_d = P1_d$
            $C2_d = P2_d$
        else
            $C1_d = P2_d$
            $C2_d = P1_d$
        end if
    end for
    Form two new solution candidates from children
end for
Mutation:
for Each Solution Candidate, S **do**
    for Each Problem Dimension, d **do**
        if rand(0,1) $<$ Mutation rate **then**
            $S_d =$ rand(min_d, max_d)
        end if
    end for
end for
return $Best\ Individual$

1.4.2 Differential evolution

While most EA's start with recombination, DE begins each generation with mutation operation by creating the donor vector:

$$v = r_1 + F(r_2 - r_3) \tag{1.1}$$

where r_1, r_2 and r_3 are randomly selected, distinct solution candidates and F is the weighting factor. Then, based on the crossover probability, CR, a trial vector, u_d, is formed from the donor vector and the current solution candidate, S_d:

$$u_d = \begin{cases} v_d & \text{if } \mathbf{rand}(0,1) \leq CR \text{ OR } d = \mathbf{rand}(1, D) \\ S_d & \text{otherwise} \end{cases} \tag{1.2}$$

where $_d$ is the independent variable and D is the problem dimension. The rand function returns a uniformly distributed random integer within the given closed interval. The logical OR statement ensures that at least one variable is taken from the donor vector while forming the trial vector. Finally, if the trial vector is fitter than the the current solution candidate, the trial vector replaces it in the next generation. This flavor of the DE algorithm is commonly referred as DE/rand/1/bin [157] and is outlined in Algorithm 5.

Algorithm 5 Pseudocode for one iteration of differential evolution function

DE Function
for Each Solution Candidate, S **do**
    Select 3 unique individuals from the population: r_1, r_2, r_3
    Form the donor vector, v:
    $v = r_1 + F(r_2 - r_3)$
    for Each Problem Dimension, d **do**
        Form the trial vector, u:
        if rand$(0, 1) \leq CR$ OR $d =$ rand$(1, D)$ **then**
            $u_d = v_d$
        else
            $u_d = S_d$
        end if
    end for
    The fitter of the two survives:
    if Cost(u) $\leq$ Cost(S) **then**
        $S' = u$
    else
        $S' = S$
    end if
end for
$S' = S$
return $Best\ Individual$

1.4.3 Biogeography-based Optimization

For this research, we implement partial immigration-based BBO as described in [122]. Partial immigration indicates that the initial selection of islands is based on immigration rates, λ, and emigration decisions are made at the level of each independent variable via roulette wheel selection. BBO's reproduction scheme is named blended migration as proposed in [158]. Blended

migration is based on blended crossover which was developed for genetic algorithms [159]. Blending refers to the act of combining the reproducing individuals using a blending parameter, α. The BBO migration scheme is presented in Algorithm 6.

Algorithm 6 Pseudocode for one iteration of biogeography-based optimization function.

BBO Function
Assign immigration rates: $\lambda_i \propto \text{rank}_i$
Assign emigration rates: $\mu_i = 1 - \lambda_i$
for Each Solution Candidate, S_i **do**
    for Each Problem Dimension, d **do**
        Select immigrating feature $S_{i,d} \propto \lambda_i$
        Select emigrating feature $S_{j,d} \propto \mu_j$
        $S_{i,d} = \alpha S_{i,d} + (1 - \alpha) S_{j,d}$
    end for
end for
Perform Mutation:
for Each Solution Candidate, S **do**
    for Each Problem Dimension, d **do**
        if rand(0,1) $<$ Mutation rate **then**
            $S_d = rand(min, max)$
        end if
    end for
end for
return $Best\ Individual$

1.5 Motivation for this Research

EAs are applied when traditional methods are inadequate- for instance when the fitness landscape has many local minima. Applying OBƎO to such

difficult problems yields promising results; however, there is still a need for development in EAs, especially in mathematical understanding. Based on the presented literature review, we see the following lack.

- Opposition theory has already been proposed for solving continuous time optimization problems. However, there is a need for analyzing the effectiveness of choosing opposition over random numbers. Therefore, in Chapter II and Section 2, we study the statistical properties of opposition for heuristic optimization algorithms.

- The statistical analysis yield to the proposal of new oppositional algorithms. Mathematical analysis of the proposed algorithms are presented in Chapter II and Sections 3-4. The validity of these novel methods is furthered analyzed in Chapter III with the help of real-world and benchmark problems.

- Many manufacturing and combinatorial problems are defined in discrete domain. However, the current definition of opposition is not valid for these type of problems. Therefore, in Chapter IV , we extend opposition to discrete domain problems.

1.6 Contributions of This Research

BBO is a newer evolutionary algorithm, but it already has proven itself a worthy competitor to the better known EAs, such as genetic algorithms, differential evolution, and ant colony optimization. BBO is a great way to approach complex nonlinear problems because it can outperform or match other EAs with less computational effort. However, there is still some room left for improving BBO since many other techniques exist in the literature that are utilized to enhance other EAs. Our goal is to experiment with these algorithms and adapt them to BBO to demonstrate BBO's highest potential. In order to achieve this goal, we introduce quasi-reflection as a new opposition method

and mathematically prove that it yields the highest expected probability of being closer to the solution among all OBL methods.

In this research, probabilistic analysis of OBL is introduced in Chapter II where we mathematically compare all existing opposition techniques and introduce a novel opposition method that is mathematically proven to be better than previous methods. Chapter III presents the results of our empirical analysis comparing the existing and new oppositional algorithms. The performance of the algorithms are tested on low and variational dimensional benchmark problems taken from the literature and real-world space trajectory optimization problems provided by European Space Agency. The significance of our findings are also discussed by employing statistical tests. Chapter IV extends opposition to discrete domain optimization problems. Chapter V discusses future work and presents concluding remarks. The detailed mathematical proofs for the results presented in Chapter II are given in Appendix A. Appendix B defines the low and variable dimensional benchmark functions and Appendix C lists the publications resulted from this research.

CHAPTER II

PROBABILISTIC ANALYSIS OF OPPOSITION-BASED LEARNING

THIS chapter presents up-to-date definitions of the opposition methods as reported in the literature and introduces new ones. We statistically compare existing and new oppositional techniques in one-dimensional space. Section 2.1 presents the definitions of various oppositional points. Section 2.2 derives the proofs of how often the quasi-opposite and reflected points are closer to the solution of an optimization problem than an EA individual or its opposite. Section 2.3 introduces a new, fitness-dependent quasi-reflection method and proves how often this new variable is closer to the solution than an EA individual. Section 2.4 derives the expected distance between the fitness-dependent quasi-reflection method and the optimal solution. Finally, Section 2.5 summarizes the proofs derived in the chapter.

2.1 Definitions of Oppositional Points

In [142], Rahnamayan introduced quasi-opposition-based learning and proved that a quasi-opposite point is more likely to be closer to the solution of

the optimization problem than the opposite point. In this section, we extend on this proof to show how much a quasi-opposite point is better than an opposite point. First, let us define opposite and quasi-opposite numbers in one dimensional space. These definitions can easily be extended to higher dimensions.

Definition Let $\hat{x}$ be any real number $\in [a, b]$. Its opposite, $\hat{x}_o$, is defined as

$$\hat{x}_o = a + b - \hat{x} \tag{2.1}$$

Notice that similar definitions already exist in mathematics. In Euclidean geometry, the opposite is referred as the inversion of point x. In addition, if the center of the domain is 0, then the opposite can be simplified as the additive inverse where $-x$ is the additive inverse of x. In Euclidean space, inversive geometry studies other such transformations such as circle and curve inversion. Since after these transformations, the distance is preserved, opposition as defined in Eq. (2.1) can be described as an isometric mapping.

Definition Let $\hat{x}$ be any real number $\in [a, b]$. Its quasi-opposite point, $\hat{x}_{qo}$, is defined as follows [131]:

$$\hat{x}_{qo} = \text{rand}(c, \hat{x}_o) \tag{2.2}$$

where c is the center of the interval $[a, b]$ and can be calculated as $(a+b)/2$, and rand$(c, \hat{x}_o)$ is a random number uniformly distributed between c and $\hat{x}_o$.

Note that unlike opposition, the quasi transformation is not a linear transformation because it involves the random function. It is also not an isometric transformation since the quasi-opposite point is not always placed equally far from the reflection point.

Since we reflect $\hat{x}$ to obtain $\hat{x}_o$ to accelerate the EA exploration process, we propose to apply the same logic and reflect the quasi-opposite point, $\hat{x}_{qo}$, to obtain the quasi-reflected point, $\hat{x}_{qr}$.

Definition Let $\hat{x}$ be any real number $\in [a, b]$. Then the quasi-reflected point, $\hat{x}_{qr}$, is defined as [160]

$$\hat{x}_{qr} = \text{rand}(c, \hat{x}) \tag{2.3}$$

where rand$(c, \hat{x})$ is a random number uniformly distributed between c and $\hat{x}$.

If x is the unknown solution to an optimization problem and $\hat{x}$ is an individual in an EA, then $\hat{x}_o$ is the opposite of the EA individual and $\hat{x}_{qo}$ and $\hat{x}_{qr}$ are the quasi-opposite and quasi-reflected individuals, respectively. Fig. 3 illustrates a point $\hat{x}$, its opposition, $\hat{x}_o$, its quasi-opposition, $\hat{x}_{qo}$ and its quasi-reflection, $\hat{x}_{qr}$ as defined in Eqs. 2.1-2.3. Earlier we discussed that opposition has different meanings in different fields. We can interpret the opposite points defined in Fig. 3 with an example from semantics. Let $\hat{x}$ be the statement that "Jane is short"; then the opposite statement, $\hat{x}_o$ would be "Jane is not short" or "Jane is tall". The quasi definitions are more fuzzy. $\hat{x}_{qo}$ would indicate that "Jane is taller than most" and $\hat{x}_{qr}$ would mean the opposite of $\hat{x}_{qo}$: "Jane is shorter than most". This explanation is comparable to a fuzzy membership degree from fuzzy set theory. Also, it is analogous to the categorization of opposition in the Aristotelian logic where the square of opposition (Fig. 4) illustrates the relationship among the contradictory propositions.

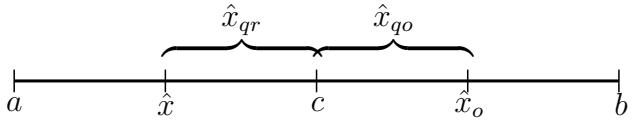


Figure 3. Opposite points defined in domain $[a, b]$. c is the center of the domain and $\hat{x}$ is an EA individual. $\hat{x}_o$ is the opposite of $\hat{x}$, and $\hat{x}_{qo}$ and $\hat{x}_{qr}$ are the quasi-opposite and quasi-reflected points, respectively.

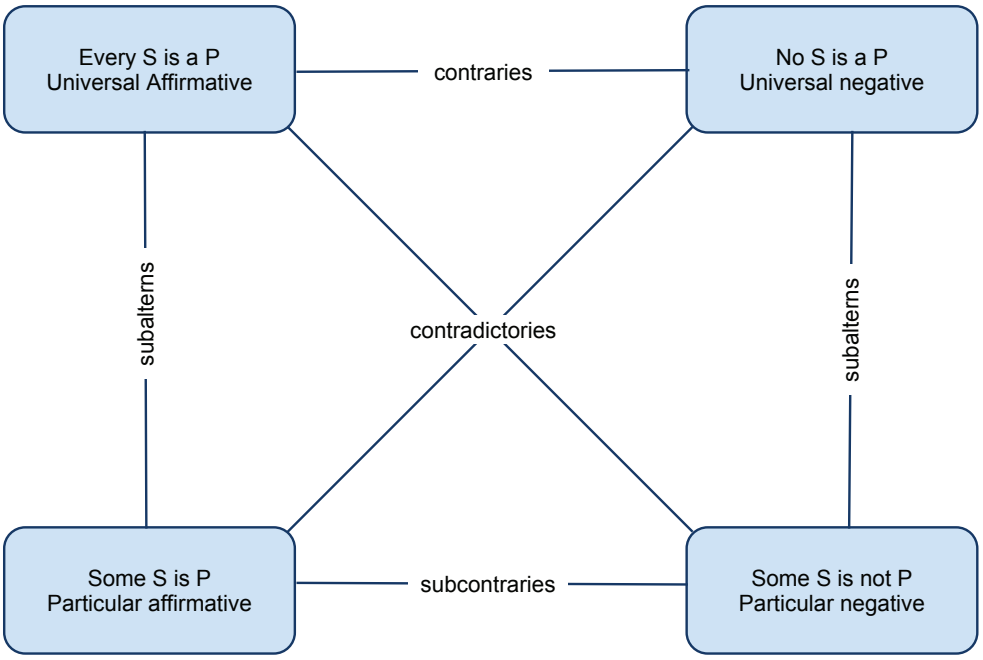


Figure 4. Square of opposition, conceived by Aristotle, classifies the relationships between opposing propositions [155].

Notice that in Fig. 3, the degree of opposition increases as we move further away from $\hat{x}$. The term degree of oppostion is defined in [155] and a crude proposal for quantifying the level of opposition is presented in Table I. We can say that in OBL, points with a higher degree of opposition dominate over the lesser degrees. Super opposition, $\hat{x}_s$, is defined in [155] as all points between $[a, b]$ except $\hat{x}$, therefore it is a superset of all defined opposite points and more. For the semantic example given above, $\hat{x}_s$ would include the statement “Jane is the shortest" as well as “Jane is the tallest". Super opposition is not discussed any further in this research.

Table I. Assignment of opposition degrees to the defined opposite points based on the opposition distance from the reflected point.

Degree of opposition	Opposition method
0	Solution estimate, $\hat{x}$
1	Quasi-reflection, $\hat{x}_{qr}$
2	Quasi-opposition, $\hat{x}_{qo}$
3	Opposition, $\hat{x}_o$
4	Super opposition, $\hat{x}_s$

2.2 Probabilistic Overview of Opposition

This section will derive the following expected probabilities, where x is the unknown solution to an optimization problem, $\hat{x}$ is an EA candidate solution, and the expected value is taken over the probability density functions of x and $\hat{x}$.

- $\Pr\left[|\hat{x}_{qo} - x| < |\hat{x}_o - x|\right]$: In Theorem 2.2.1, we prove how likely it is that a quasi-opposite point is closer than the opposite of an EA individual to the solution of an optimization problem.

- $\Pr\left[|\hat{x}_{qr} - x| < |\hat{x}_o - x|\right]$: In Theorem 2.2.2, we prove how likely it is that a quasi-reflected point is closer than the opposite of an EA individual to the solution of an optimization problem.

- $\Pr\left[|\hat{x}_{qo} - x| < |\hat{x} - x|\right]$: In Theorem 2.2.3, we prove how likely it is that a quasi-opposite point is closer than an EA individual to the solution of an optimization problem.

- $\Pr\left[|\hat{x}_{qr} - x| < |\hat{x} - x|\right]$: In Theorem 2.2.4, we prove how likely it is that a quasi-reflected point is closer than an EA individual to the solution of an optimization problem.

We should note that all our proofs are in one dimensional space and we assume that the solution x of the optimization problem has a uniform distribution.

Our assumption of uniformity is validated by the Principle of Insufficient Reason proposed by Bernoulli [161] and Laplace [162], although neither mathematician named the principle. The name is given by the critics of the theorem as a wordplay of Leibnitz's Principle of Sufficient Reason [163] which states that "nothing happens without a reason". According to the Principle of Insufficient Reason, "in the absence of prior knowledge, we must assume that events A_i have equal probabilities" [164]. As an example, one can consider tossing a coin. Probabilities of obtaining a head or a tail are assumed to be equal because we presume that the probability of occurrence of one over the other is unlikely. Another example would be picking a card from a deck. Since we don't have any knowledge of the distribution of the cards in the deck, we assume that all cards have equal probability of being picked.

Finally, we assume that the problem domain is symmetric about 0, thus $b = -a$. This assumption is made for ease of notation, and can be relaxed without losing the generality of the results.

Theorem 2.2.1. *Assume that the solution x of an optimization problem is uniformly distributed in a one-dimensional search space. Then the probability averaged over all x and all $\hat{x}$ that a quasi-opposite point $\hat{x}_{qo}$ is closer to the solution than the opposite of an EA individual $\hat{x}_o$ is $^{11}/_{16}$.*

Proof. See Appendix A.1. □

Theorem 2.2.2. *Assume that the solution x of an optimization problem is uniformly distributed in a one-dimensional search space. Then the probability averaged over all x and all $\hat{x}$ that a quasi-reflected point $\hat{x}_{qr}$ is closer to the solution than the opposite $\hat{x}_o$ of an EA individual is $^{9}/_{16}$.*

Proof. See Appendix A.2. □

Now that we obtained the performance of quasi-opposition versus opposition, we investigate the probability of quasi-opposition against the evolution-

ary algorithm individual in more detail. First, we compute the probability of $\hat{x}_{qo}$ being closer than $\hat{x}$ to the solution of an optimization problem, x, and the expected value of this probability under certain conditions.

Theorem 2.2.3. *Assume that the solution x of an optimization problem is uniformly distributed in a one-dimensional search space. Then the probability averaged over all x and all $\hat{x}$ that a quasi-opposite point $\hat{x}_{qo}$ is closer to the solution than an EA individual is $^9/_{16}$.*

Proof. See Appendix A.3. □

The final lemma in this section is the probability of $\hat{x}_{qr}$ being closer than $\hat{x}$ to the solution of an optimization problem, x, and the expected value of this probability.

Theorem 2.2.4. *Assume that the solution x of an optimization problem is uniformly distributed in a one-dimensional search space. Then the probability averaged over all x and all $\hat{x}$ that a quasi-reflected point $\hat{x}_{qr}$ is closer to the solution than an EA individual is $^{11}/_{16}$.*

Proof. See Appendix A.4. □

2.3 Fitness-Weighted Quasi-Reflection

In this section, we introduce a new opposite point named fitness-dependent quasi-reflection or $\hat{x}_{Kr}$. Unlike $\hat{x}_{qr}$, $\hat{x}_{Kr}$ is a not an independent random variable. Instead, it is defined as the function of the fitness of $\hat{x}$. This way we can control the amount of reflection based on the fitness of the individual. Thus, fit solutions can be reflected by a smaller amount than less fit solutions. $\hat{x}_{Kr}$ is defined as

$$\hat{x}_{Kr} = \begin{cases} \hat{x} + (c - \hat{x})K & \text{if } \hat{x} \leq c \\ c + (\hat{x} - c)(1 - K) & \text{if } \hat{x} > c \end{cases} \tag{2.4}$$

where $K \in (0, 1]$ is the reflection weight and can further be described as:

$$K = \frac{\text{Solution rank}}{\text{Population size}} \tag{2.5}$$

and solution rank = 1 for the best individual in the population.

Even though, fitness-dependent reflection is applied to $\hat{x}_{qr}$ here, it can easily be applied to any other opposition method. However, since $\hat{x}_{qr}$ is shown to have the highest probability of being closer to the solution, it is taken as the base for the $\hat{x}_{Kr}$ algorithm.

Eq. 2.4 can be redefined by using the unit step function, $\mathrm{U}(x)$. The unit step function of x is a discontinuous function that is defined as 0 for negative values of x and 1 for the remaining values of x.

$$\hat{x}_{Kr} = [\hat{x} + (c - \hat{x})K]\,\mathbf{U}(c - \hat{x}) + [c + (\hat{x} - c)(1 - K)]\,\mathbf{U}(\hat{x} - c) \tag{2.6}$$

$\hat{x}_{Kr}$ eliminates the need for the previously defined random function by considering the relative fitness of the individual. Let the center of the domain, c, be zero. Then Eq. 2.4 can be simplified as

$$\hat{x}_{Kr} = \hat{x}(1 - K) \tag{2.7}$$

This section derives results that are analogous to Theorem 2.2.4 for the fitness-weighted quasi-reflected point and computes the probability of $\hat{x}_{Kr}$being closer than $\hat{x}$ to the solution of an optimization problem, x, and the expected value of this probability as a function of the reflection weight, K.

Theorem 2.3.1. *Assume that the solution of an optimization problem is uniformly distributed in a one-dimensional search space. Then the probability averaged over all x and all $\hat{x}$ that a fitness-weighted quasi-reflected point is closer than an EA individual $\hat{x}$ to the solution x is* $(6-K)/8$.

Proof. See Appendix A.5. □

In the previous section, we solved for the probabilities for $\hat{x}_{qr}$ and $\hat{x}$. In this section we used $\hat{x}_{Kr}$, which depends on $\hat{x}_{qr}$ and $\hat{x}$ and the individuals

ranking among the population of solution candidates. Here, we would like to validate our findings for $\hat{x}_{Kr}$ by comparing it to those of $\hat{x}_{qr}$. Recall that in Theorem 2.2.1, we obtained

$$\Pr\left[|\hat{x}_{qr} - x| < |\hat{x} - x|\right] = \frac{11}{16} \tag{2.8}$$

In this section, when we defined $\hat{x}_{Kr} = \hat{x} - K\hat{x}$, we proved that

$$\Pr\left[|\hat{x}_{Kr} - x| < |\hat{x} - x|\right] = \frac{6 - K}{8} \tag{2.9}$$

for $K \in (0, 1]$. If we assume that K, the reflection weight, is uniformly distributed, then $\mathrm{E}[K] = \frac{1}{2}$ and the expected value of Eq. 2.9 becomes equal to Eq. 2.8:

$$\mathbf{E}_K\left\{\Pr\left[|\hat{x}_{Kr} - x| < |\hat{x} - x|\right]\right\} = \mathbf{E}_K\left\{\frac{6 - K}{8}\right\} = \frac{11}{16} \tag{2.10}$$

where E_K indicates calculating the expected value with respect to$K$. Furthermore, K can be designed to have a non-uniform distribution so different reflection patterns can be developed to better fit a given problem. Equation 2.5 defines the reflection weight as a linear function of individual fitness. However, based on our expertise on a given problem, we can choose different K values. Table II lists four complementary functions, quadratic and sinusoidal, that could be used to create the reflection weights. Plots of these nonlinear functions are presented in Fig. 5. These functions are inspired from the BBO migration models presented in [165].

Table II. Example of quadratic and sinusoidal functions that can be used to create reflection weights where r is the rank of an individual, where 1 is best, and p is the population size.

Label	Reflection weight
f1	$\left(\frac{r}{p}\right)^2$
f2	$\left(\frac{r}{p} - 1\right)^2$
f3	$\frac{1}{2}\left(\cos\left(\frac{r\pi}{p}\right) + 1\right)$
f4	$\frac{1}{2}\left(-\cos\left(\frac{r\pi}{p}\right) + 1\right)$

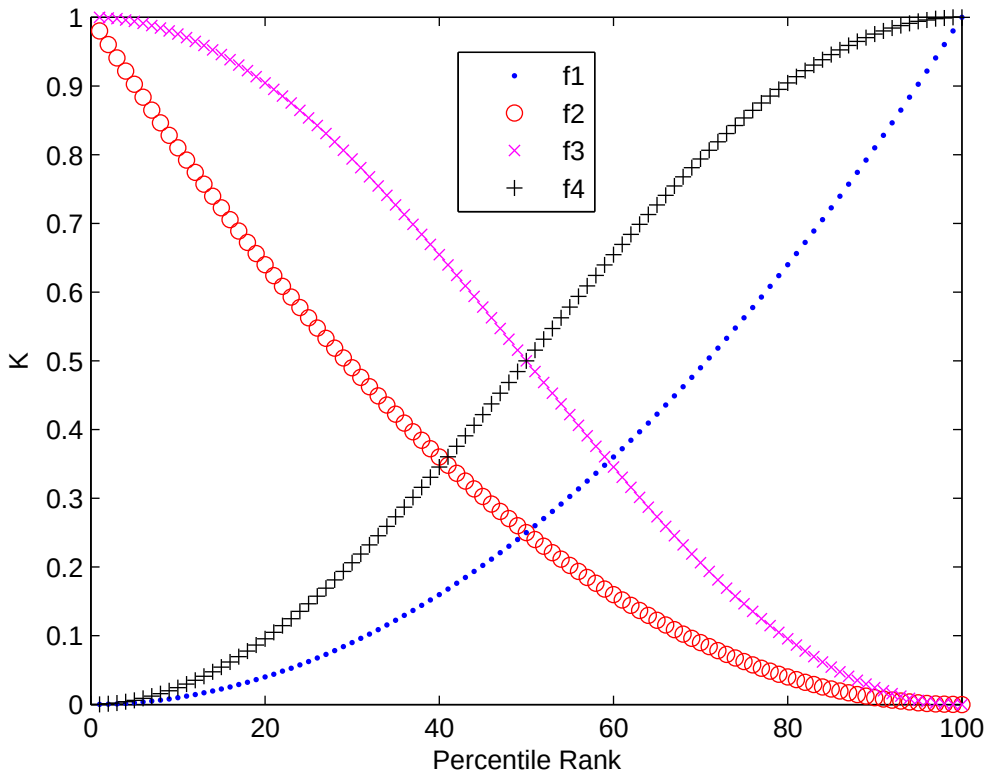


Figure 5. Four possible nonlinear reflection weights based on individual rankings.

Fig. 6 plots the expected probability of $\hat{x}_{Kr}$ being closer than $\hat{x}$ to the solution as a function of reflection weight. The results are derived theoretically and verified via simulation. Note that there is a discontinuity in Fig. 6 when $K = 0$, where the probability is 0. After this point, at $K = 0_+$, the probability jumps to about 75%.

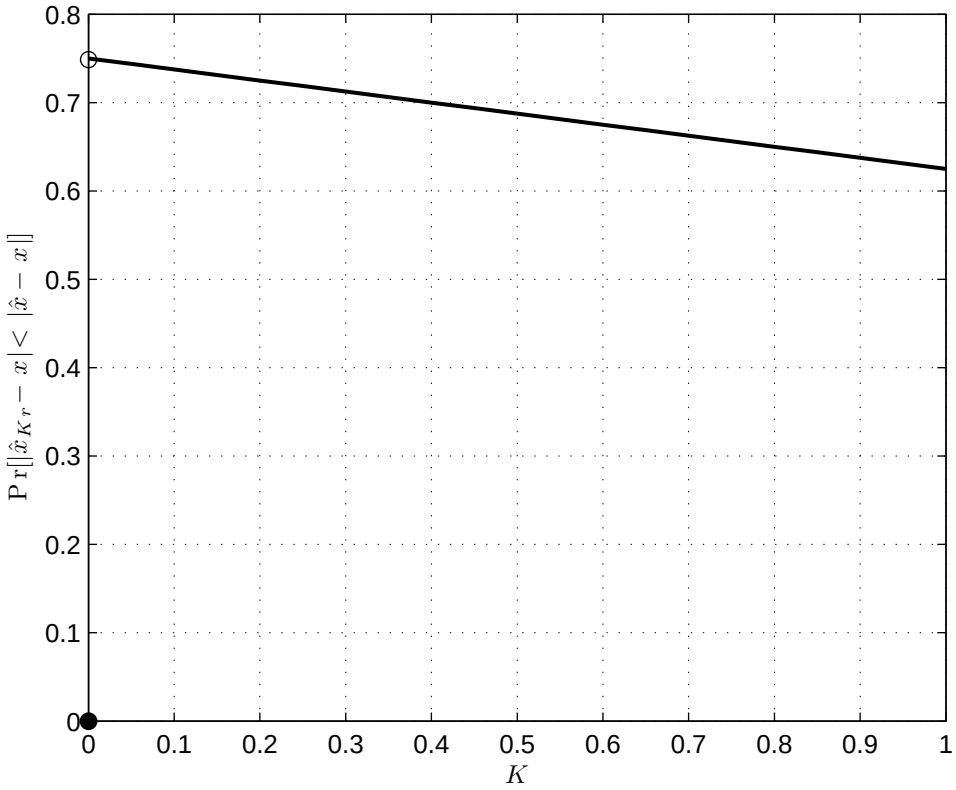


Figure 6. Expected probability that $\hat{x}_{Kr}$ is closer to the solution of an optimization problem than an EA individual

2.4 Distance Between a Fitness-Dependent Quasi-Reflected Point and the Solution

In Section 2.2, we compared the probability of $\hat{x}_{qr}$ being closer than $\hat{x}$ to the optimal solution x. Later, in Section 2.3 we defined $\hat{x}_{Kr}$ to be a fitness-weighted quasi-reflection point that is a function of the reflection weight K and $\hat{x}$. We then calculated the expected probability of $\hat{x}_{Kr}$ being closer than $\hat{x}$ to the optimal solution x as a function of the reflection weight K. In this section, we calculate $\hat{x}_{Kr}$'s distance to the optimal solution as a function of the reflection weight K and $\hat{x}$.

Recall that

$$\hat{x}_{Kr} = \hat{x}(1 - K) \tag{2.11}$$

Appendix A.6.1 presents the probability distribution functions (pdf) necessary for our calculations in the subsequent sections. We employ these pdf's in Appendix A.6.2-A.6.4 to calculate the expected distance between the fitness-weighted quasi-reflected point and the optimal solution, where the distance is defined as

$$|\hat{x}_{Kr} - x| = |\hat{x}(1 - K) - x| \tag{2.12}$$

Lemma 2.4.1. *Assume that the solution of an optimization problem is uniformly distributed in a one-dimensional search space between $-b$ and b. Then the expected distance between $\hat{x}_{Kr}$ and x is* $\left[3bK^2 - 2b(K-1)(2+K)\right]/6$.

Proof. See Appendix A.6.2-A.6.4. □

In Appendix A.6.5-A.6.7, we derive the expected distance between the EA individual and the solution, obtaining the following lemma.

Lemma 2.4.2. *Assume that the solution of an optimization problem is uniformly distributed in a one-dimensional search space. Then the expected distance between $\hat{x}$ and x is*

$$\boldsymbol{E}\left[|\hat{x} - x|\right] = \frac{2b}{3} \tag{2.13}$$

Proof. See Appendix A.6.5-A.6.7. □

In Appendix A.6.4, we calculate the expected distance between the fitness-weighted quasi-reflected individual and the minimum of an optimization problem as a function of the reflection weight, K. Then, in Appendix A.6.7, we calculate the expected distance between the EA individual and the minimum. We can now combine these two findings and calculate the expected difference in distance. The difference between these two distances can be written as

$$\begin{aligned} \mathbf{E}\left[|\hat{x}_{Kr} - x|\right] - \mathbf{E}\left[|\hat{x} - x|\right] &= \frac{bK^2}{2} - \frac{b(K-1)(2+K)}{3} - \frac{2b}{3} \\ &= \frac{bK(K-2)}{6} \end{aligned} \tag{2.14}$$

Theorem 2.4.3. *Assume that the solution of an optimization problem is uniformly distributed in a one-dimensional search space. Then the expected distance between* $\hat{x}_{Kr}$ *and* x *and* $\hat{x}$ *and* x *is* $\frac{bK(K-2)}{6}$

Fig. 7 shows the theoretical and simulation results of the calculated distances. The simulation results are obtained by generating 100,000 random points for the solution x, solution candidate $\hat{x}$ and fitness-based quasi-reflected point $\hat{x}_{Kr}$. The average difference of these simulated points are indicated with markers $+$, $*$ and o. The straight and dashed lines represent the results of our mathematical findings. The results of the Monte Carlo simulations and theoretical equations are well-aligned. Recall that when the reflection weight is 0, a fitness-weighted quasi-reflected point is identical to the opposite of an EA individual. However, as K increases, the fitness-weighted quasi-reflected point gets closer (on average) to the solution of an optimization problem.

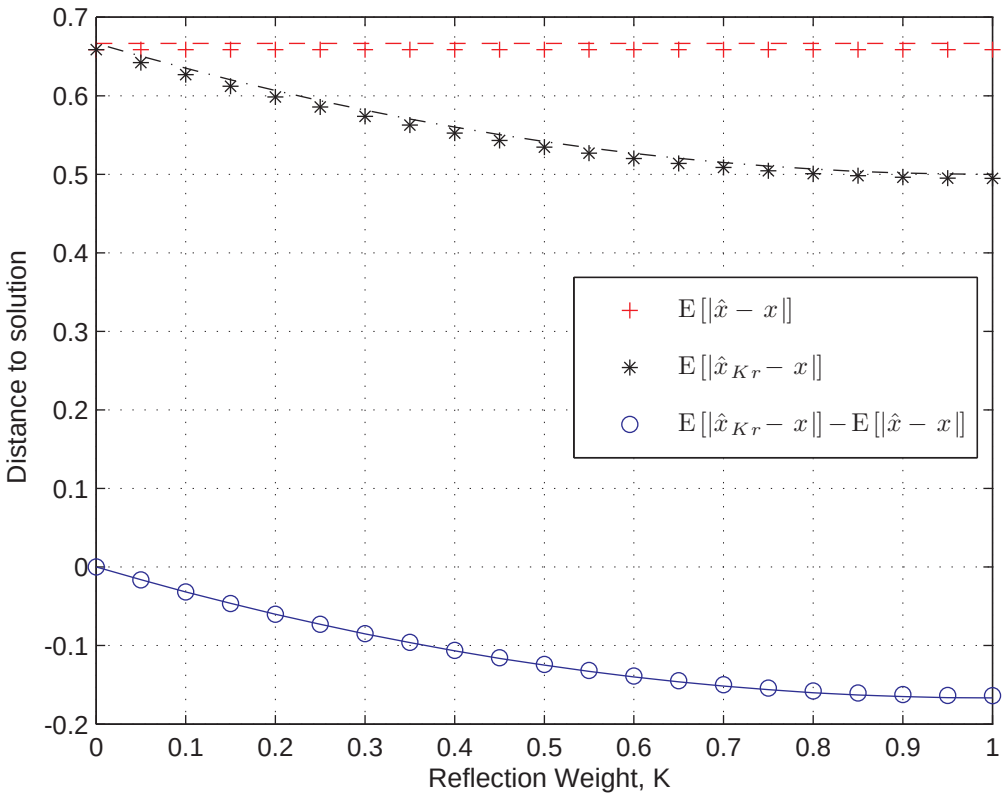


Figure 7. The expected distance between the fitness-weighted quasi-reflected individual and the solution of an optimization problem, and the EA individual and that solution.

2.5 Summary

The presented results assume that the problem space is one-dimensional; however, they can be be extended for higher dimensions. We assumed that the solution and estimate have uniform distributions as in [166] and that the problem domain is symmetric such that $b = -a$ to simplify the resulting mathematical expressions. Finally, we limit the reflection weight to $K \in (0, 1]$. Varying the range of K will create different opposition algorithms and can be a topic of further research.

2.5.1 Probabilities

Table III lists the probabilities of being closer to the solution of an optimization problem for all of the discussed opposition points. Rows 1-3 compare the probability of the opposition points relative to an EA individual, and Rows 4 and 5 compare the probability of the quasi-opposite points relative to the opposition of the EA individual. Row 6 lists the probability of fitness-dependent quasi-reflection being closer than an EA individual to the solution.

Table III. Probability that opposite point is closer than an EA individual to the solution of an optimization problem.

Row		Probability
1	$\Pr\left[\lvert\hat{x}_o - x\rvert < \lvert\hat{x} - x\rvert\right]$	$\frac{1}{2}$
2	$\Pr\left[\lvert\hat{x}_{qo} - x\rvert < \lvert\hat{x} - x\rvert\right]$	$\frac{9}{16}$
3	$\Pr\left[\lvert\hat{x}_{qr} - x\rvert < \lvert\hat{x} - x\rvert\right]$	$\frac{11}{16}$
4	$\Pr\left[\lvert\hat{x}_{qo} - x\rvert < \lvert\hat{x}_o - x\rvert\right]$	$\frac{11}{16}$
5	$\Pr\left[\lvert\hat{x}_{qr} - x\rvert < \lvert\hat{x}_o - x\rvert\right]$	$\frac{9}{16}$
6	$\Pr\left[\lvert\hat{x}_{Kr} - x\rvert < \lvert\hat{x} - x\rvert\right]$	$\frac{6-K}{8}$

From Table III, we observe that in an optimization problem, the highest probability of being closer to the solution than an EA individual is the quasi-reflected point, presented in Row 3. Row 4, the quasi-opposite point, also has the same probability; however, it is compared to opposite of an EA individual, not the EA individual itself. Therefore, quasi-reflection should be the preferred opposition algorithm when working with the available EA individuals to yield the highest probability of being closer to the solution. The probability of fitness-dependent quasi-reflection being closer to the solution presented in Row 6, is dependent on the reflection weight K, which in turn depends on the individual's relative fitness in the population. Note that an average individual ($K = 1/2$) will have the same probability of being closer to the solution as the quasi-reflected point.

2.5.2 Expected Distance of Fitness-Weighted Quasi-Reflection Compared to an EA Individual

We have shown in Section 2.3 that the probability that a fitness-weighted quasi-reflected point is closer than a random EA individual to the solution of an optimization problem is

$$\begin{aligned} \Pr\left[|\hat{x}_{Kr} - x| < |\hat{x} - x|\right] &= \Pr\left[|\hat{x}(1-K) - x| < |\hat{x} - x|\right] \\ &= \frac{6-K}{8} \end{aligned}$$

Table IV summarizes the findings of Section 2.4 where we derived the expected distance of a fitness-weighted quasi-reflected point to the solution of an optimization problem compared to a random EA individual's distance to that solution.

Table IV. Distance to solution as a function of reflection weight, where the problem domain is $[-b, b]$

	Probability
$\mathbf{E}\left(\lvert\hat{x} - x\rvert\right)$	$\frac{2b}{3}$
$\mathbf{E}\left(\lvert\hat{x}_{Kr} - x\rvert\right)$	$\frac{bK^2}{2} - \frac{b(K-1)(2+K)}{3}$
$\mathbf{E}\left(\lvert\hat{x}_{Kr} - x\rvert - \lvert\hat{x} - x\rvert\right)$	$\frac{bK}{6}(K-2)$

Fig. 7 combines the results from Sections 2.3 and 2.4 and plots the expected distance and probability for a fitness-weighted quasi-reflected point as a function of reflection weight, K. From this figure, we notice that the expected probability of being closer to the solution of an optimization problem, and the distance to the solution, both decrease with K. While a shorter distance to the solution is desirable, having a smaller chance of being closer to the solution is not desirable.

The linear reflection weight equation as defined in Eq. 2.5 yields a large K for less fit solutions to enable higher reflection. Fig. 8 plots the distance

and probability of being closer to solution for $\hat{x}_{Kr}$ and $\hat{x}$ with respect to K. The straight and dashed lines represent the theoretical results and the markers o and $*$ are obtained via randomly generating x, $\hat{x}$ and $\hat{x}_{Kr}$ points and calculating their expected distance and probabilities. The simulation and theoretical results are well-aligned. Based on Fig. 8, when using a larger K, the individual has a less change of being closer to the solution but the expected distance to the solution is less than a random EA point.

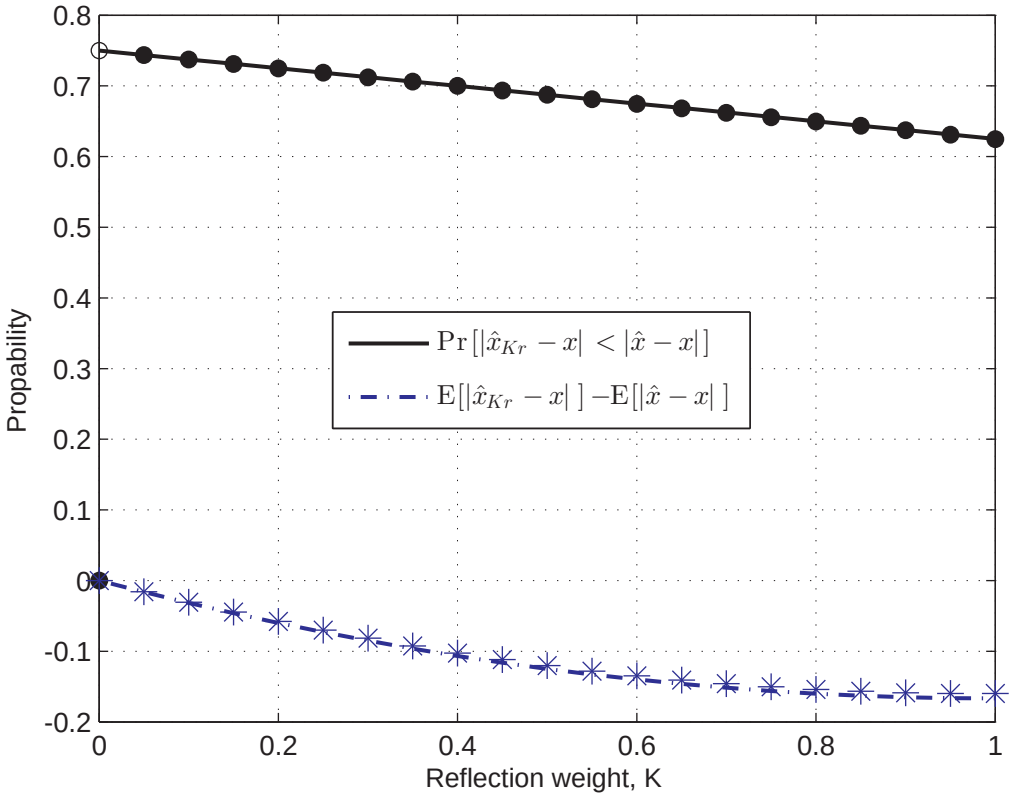


Figure 8. Relative distance to the solution of an optimization problem, and probability of being closer to the solution, between $\hat{x}_{Kr}$ and $\hat{x}$. Notice that when K is small, $\hat{x}_{Kr}$ has the highest probability of being closer to solution. However, for small K, $\hat{x}_{Kr}$ is closer by a negligible amount.

CHAPTER III

EMPIRICAL RESULTS OF OPPOSITION-BASED LEARNING

THE probabilities calculated in the previous chapter are studied in this chapter using standard benchmark functions from the literature as well as the real-world. The first half of this chapter will focus on the problems from the literature. Section 3.1 explains the metrics utilized to compare the performance of various EAs and Section 3.2 introduces the benchmark functions in more detail. The results of the benchmark problems are presented in Section 3.3. The second half of the chapter analyzes the performance of the oppositional algorithms on real world problems. Section 3.4 introduces the global optimization problems provided by the European Space Agency (ESA). The simulation metrics for these problems are presented in Section 3.5 and the performance of the EA is discussed in Section 3.6. Section 3.7 analyzes the significance of the presented results.

3.1 Simulation Settings

This section outlines the methodology utilized to measure the performance of OBƎO . Performance analysis of the presented algorithms is based on the number of cost function evaluations, F_c, performed before reaching the desired solution range, because generally, the cost function evaluation is the most CPU intensive task of an EA [167]. The following method, which we employ to test our algorithms, is published in [168, 169]. This method compares the number of cost function evaluations required for an EA to converge to a value near the solution. The desired convergence value is calculated by:

$$|f - \hat{f}| < \epsilon_1 |f| + \epsilon_2 \tag{3.1}$$

where f is the known solution, $\hat{f}$ is the best solution candidate at the current generation, and ϵ_1 and ϵ_2 are small positive numbers.

We now explain the various parameters for the presented simulations in more detail. As we increase the population size, we are increasing the number of cost function evaluations. This generally helps converge to the solution at the cost of simulation time. Therefore, we increase the population size with the dimension of the benchmark problem. Table V lists the settings for the presented results. Note that this table is just a rule of thumb. For example, a more demanding problem, such as Perm, can be evaluated with a population of 100 for 10 dimensions.

Table V. Problem dimension vs. population size

Problem Dimension	Population Size
2	4
4	8
20	50
30	98
100	350

In order to avoid unbounded run times, we introduce an upper limit on function calls, $MaxFc$. If the best solution has not reached the desired solution range by the set number of cost function evaluations, we quit the simulation. Vesterstrom and Thomsen [170] used an evaluation limit of 500,000 for 30-dimensional problems and an evaluation limit of 5,000,000 for 100-dimensional problems. Keeping these settings as our reference, we set $MaxFc$ as shown in Table VI.

Table VI. Problem dimension vs. maximum function calls

Problem Dimension	Maximum Function Calls
≤ 4	1×10^5
> 4	5×10^6

The tolerance level for acceptable solutions is also based on the problem dimension [169]. In Eq. 3.1, we define ϵ_1 as 10^{-4} for all dimensions and let ϵ_2 be determined by the problem dimension as shown in Table VII.

Table VII. Problem dimension vs solution tolerance

Problem Dimension	ϵ_1	ϵ_2
$\leq$4	10^{-4}	10^{-4}
$>$ 4	10^{-4}	10^{-6}

OBƎO 's jumping rate constant, J_r, is set to 0.3 [141]. This means that at each generation, we have a 30% chance of calculating opposite populations. J_r can also decrease with each generation so that the number of cost function calls due to the oppositional algorithm decreases with time.

Finally, the best two solution candidates in each generation are preserved using elitism for BBO and OBƎO . For future work, one can keep track of the standard deviation over each Monte Carlo run and apply a statistical hypothesis test, such as a chi-square test, to analyze the effects of varying these parameters.

3.2 Benchmark Functions

This section introduces the 22 continuous-domain benchmark functions employed to compare the performance of OBƎO and BBO. These problems are selected to provide a variety of challenges to OBƎO as each function includes different characteristics: multimodality, nonseparability, or irregularity. Multimodal functions are functions which have many local minima, nonseparable functions have inter-dependencies among the variables for an added challenge and irregular functions are nondifferentiable. In this dissertation, we further categorize these functions as low-dimensional and variable-dimensional. More information on these functions can be found in [170, 141, 171, 172] or on Appendix C where a definition and a two-dimensional plot of each benchmark function is provided.

Section 3.2.1 presents the low-dimensional functions. These functions are two- or four-dimensional. The variable-dimensional functions are presented in Section 3.2.2. In these problems, the problem dimension is adjustable. The majority of the functions employed are in this category since variable-dimensional functions can also be utilized for low-dimensional simulations. Some of these functions, such as quartic, include random noise to simulate real world applications. Finally, constrained functions are left for future work. Considered examples include:

1. Keane's bump function [173]
2. Appendix C of [174]

3.2.1 Low-dimensional Benchmark Problems

Table VIII presents an overview of the low-dimensional benchmark problems used.

Table VIII. Low-dimensional benchmark functions. The superscript is the problem dimension

Function	Domain	argmin	min f(x)
Beale	$(-4.5, 4.5)^2$	$(3, 0.5)$	0
Colville	$(-10, 10)^4$	1^4	0
DeJong F5	$(-65.536, 65.536)^2$	$(-32, 32)$	0.998
Easom	$(-100, 100)^2$	(π, π)	-1
Tripod	$(-100, 100)^2$	$(0, -50)$	0

3.2.2 Variable-dimension Benchmark Problems

An overview of these functions is listed in Table IX. Note that the Penalty 1 and Penalty 2 functions, also called Generalized Penalized Functions [172], have typographical errors in many publications [175, 176, 177, 178], including

some heavily-referenced articles [172, 170]. Readers should refer to Equations 25 and 26 in the original publication [179] for the correct equations.

Also, two of the variable dimension problems, Fletcher and Perm, are set as four-dimensional problems, instead of twenty, owing to the fact that their minimum could not be located within the listed boundaries for higher dimensions.

Table IX. Variable-dimensional benchmark functions, where n is the problem dimension

Function	Domain	argmin	min f(x)
Ackley	$(-30, 30)^n$	0^n	0
Alpine	$(-10, 10)^n$	0^n	0
Fletcher/Powell	$(-\pi, \pi)^n$	rand$(-\pi, \pi)^n$	0
Griewank	$(-600, 600)^n$	0^n	0
Penalty1	$(-50, 50)^n$	1^n	0
Penalty2	$(-50, 50)^n$	1^n	0
Perm	$(-n, n)^n$	$(1, 2, ..., n)$	0
Quartic	$(-1.28, 1.28)^n$	0^n	0
Rastrigin	$(-5.12, 5.12)^n$	0^n	0
Rosenbrock	$(-30, 30)^n$	1^n	0
Schwefel 1.2	$(-65.536, 65.536)^n$	0^n	0
Schwefel 2.21	$(-100, 100)^n$	0^n	0
Schwefel 2.22	$(-10, 10)^n$	0^n	0
Schwefel 2.26	$(-512, 512)^n$	420.9687^n	$-418.9829n$
Sphere	$(-5.12, 5.12)^n$	0^n	0
Step	$(-100, 100)^n$	0^n	0
Zakharov	$(-5, 10)^n$	0^n	0

3.3 Simulation Results

In this section, we provide preliminary simulation results. Section 3.3.1 compares the performance of all the oppositional algorithms presented and Section 3.3.2 explores the effects of static and dynamic reflection on quasi-reflection and quasi-opposition.

3.3.1 Experimental Comparison of Oppositional Algorithms

As noted in Table IX, many benchmark functions have a symmetric domain and their optimizing argument is located at the center of the domain. This is not a very realistic scenario and an EA can be designed to take advantage of that. Thus, in order to test the effectiveness of BBO, we randomly shift the solution. One way to achieve this goal, while maintaining the original range of a benchmark problem, is to randomly shift the domain of the problem for each Monte Carlo run. Shifting the domain of the problem yields the illusion of shifting the solution, without modifying the problem equation. The ten benchmarks with shifted domain are Ackley, Alpine, Griewank, Quartic, Rastrigin, Schwefel 1.2 - 2.22, Sphere and Step. The shifted domain is calculated as follows. Let a problem have a domain of $[-a, a]$ with solution located at the center of the domain, 0. Then the range of the solution domain is $2a$. The shifted domain is defined as $[r - 2a, r]$ where r is a random point uniformly distributed in $[0, 2a]$

For example, Ackley is defined in $[-30, 30]$ with min f$(x) = 0$ located at $(0, 0, .., 0)$. Then, r is a random number in $(0, 60)$ and shifted Ackley is defined in $[r - 60, r]$ with min f$(x) = 0$ at $(0, 0, .., 0)$. As a result, based on the value of r, the domain of shifted Ackley can be anywhere in $[-60, 0]$ to $[0, 60]$ while the solution will still be at $(0, 0, .., 0)$.

Other simulation settings:

- Number of Monte Carlo runs: 50

- Mutation rate: 0
- Variable dimension: 20

Table X compares the proposed oppositional algorithms, $\hat{x}_o$, $\hat{x}_{qo}$, $\hat{x}_{qr}$, $\hat{x}_{Kr}$, alongside BBO for the lower dimensional benchmark functions, and Table XI lists the results for the variable dimension problems with twenty dimensions.

Table X. The mean of the best results from BBO, $\hat{x}_o$, $\hat{x}_{qo}$, $\hat{x}_{qr}$, $\hat{x}_{Kr}$ for lower dimension benchmark problems. The maximum number of function calls is limited to 100,000. SR is the success rate (that is, the proportion of simulations that found a solution to the desired accuracy). Mean Fc is the average number of function calls before a solution was found.

	BBO/$\hat{x}$		BBO/$\hat{x}_o$		BBO/$\hat{x}_{qo}$		BBO/$\hat{x}_{qr}$		BBO/$\hat{x}_{Kr}$	
Benchmark	SR	Mean Fc	SR	Mean Fc	SR	Mean Fc	SR	Mean Fc	SR	Mean Fc
Beale	0	-	0	-	0	-	**0.16**	**50343**	0.06	285
Colville	0.98	3504	**0.98**	**330**	0.92	10176	0.88	18676	0.78	18445
DeJong F5	1	400	**1**	**64**	1	700	1	504	1	616
Easom	0	-	0	-	0	-	0	-	0	-
Fletcher	0	-	0	-	0	-	0	-	0	-
Perm	**0.92**	**25497**	0.46	554	0.08	26228	0.16	26993	0.2	25039
Tripod	0.28	727	**1**	**99**	1	4505	0.68	15092	0.44	981
Mean	0.45	7532	**0.49**	**262**	0.43	10402	0.41	22321	0.35	9073

According to Table X, BBO augmented with standard opposition, BBO/$\hat{x}_o$, reduces the number of function calls necessary by 96.5% while increasing BBO's success rate. While none of the algorithms can solve Easom and Fletcher, BBO/$\hat{x}_{qr}$ and BBO/$\hat{x}_{Kr}$ are the only algorithms that are able to provide some successful solutions to the Beale problem. Also, all of the opposition techniques outperform BBO/$\hat{x}$ on the Tripod problem. On the other hand, BBO/$\hat{x}_{qr}$ and BBO/$\hat{x}_{Kr}$ cannot successfully solve the Perm problem as often as BBO/$\hat{x}$ and this causes their success rate to be below BBO/$\hat{x}$'s.

Figs. 9-10 provide some sample runs from these benchmarks. In Fig 9,

we display the best results from all algorithms for the first 10 generations of Colville. Recall that it takes BBO/$\hat{x}_{qr}$ five times more function calls than BBO to solve this problem. However, Fig 9 shows that the oppositional algorithms start converging faster than BBO.

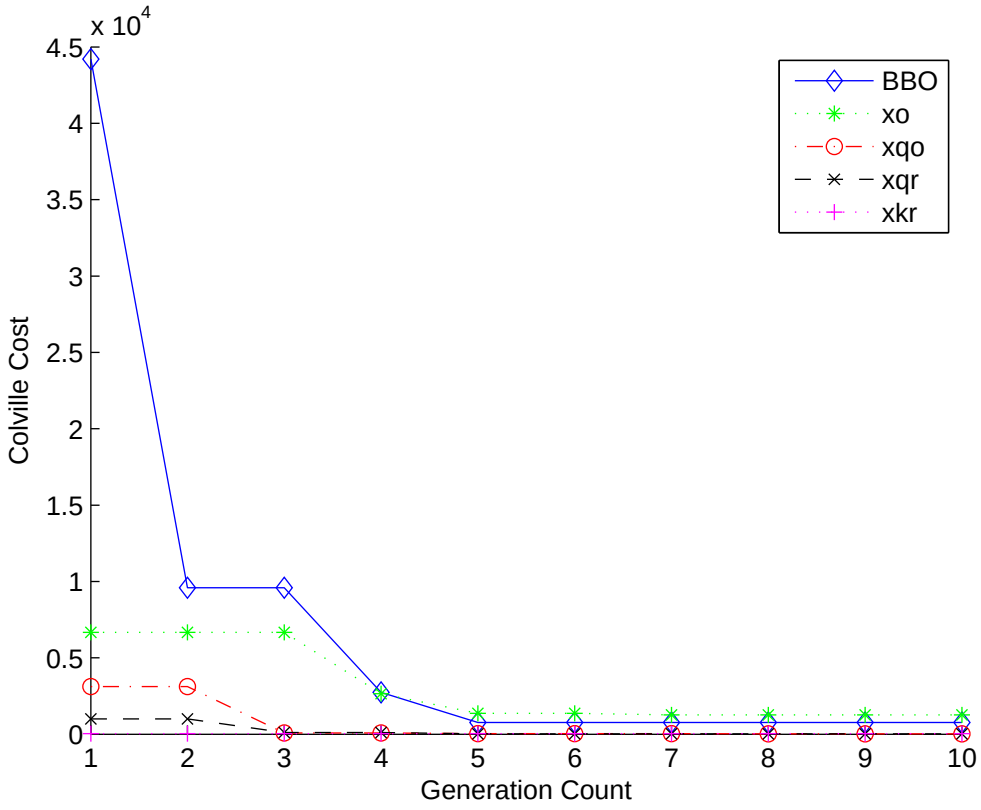


Figure 9. Fist ten generations of best results obtained for Colville. Oppositional algorithms start strong.

Fig 10 plots the best results from the Colville problem between generations 90 and 100. Notice that the oppositional algorithms still provide better solutions than BBO.

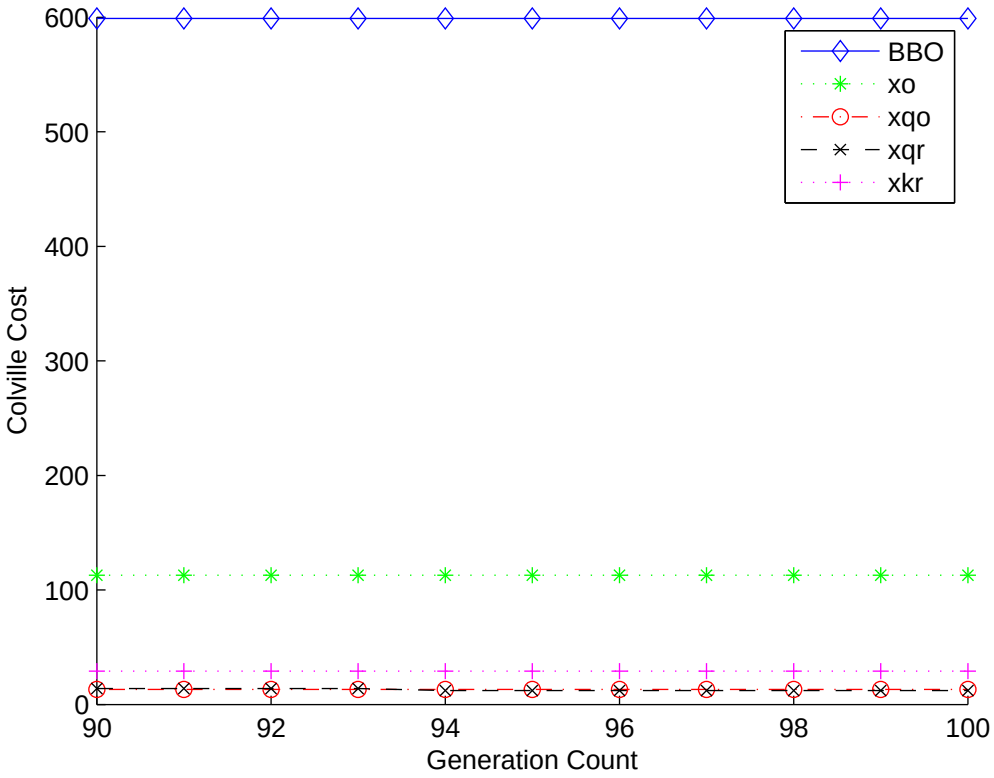


Figure 10. Generations 90-100 of best results obtained for Colville. Oppositional algorithms still provide better solutions.

Table XI. The mean of the best results from BBO, $\hat{x}_o$, $\hat{x}_{qo}$, $\hat{x}_{qr}$, $\hat{x}_{Kr}$ for twenty dimensional benchmark problems. The maximum number of function calls is limited to 5,000,000. Superscript s indicates that the domain of the problem is shifted randomly for each Monte Carlo simulation. SR is the success rate (that is, the proportion of simulations that found a solution to the desired accuracy). Mean Fc is the average number of function calls before a solution was found.

	BBO/$\hat{x}$		BBO/$\hat{x}_o$		BBO/$\hat{x}_{qo}$		BBO/$\hat{x}_{qr}$		BBO/$\hat{x}_{Kr}$	
Benchmark	SR	Mean Fc	SR	Mean Fc	SR	Mean Fc	SR	Mean Fc	SR	Mean Fc
Ackley^s	**1**	**19506**	1	24718	1	23466	1	23933	1	32428
Alpine^s	1	31174	1	39131	1	11994	1	9774	**1**	**4407**
Griewank^s	**0.18**	**537629**	0.16	371364	0.06	321172	0.12	429915	0.04	847383
Penalty1	**1**	**29193**	1	40644	1	44294	1	38129	1	44345
Penalty2	**1**	**26838**	1	47391	1	47147	1	45110	1	43215
Quartic^s	**1**	**188232**	1	255268	0.92	431632	0.76	833877	0	-
Rastrigin^s	**1**	**5121**	1	6682	1	6696	1	6853	1	7441
Rosenbrock	0	-	0	-	0	-	0	-	0	-
Schwefel 1.2^s	0.94	1651347	0.92	2140428	**0.96**	**2044892**	0.92	2424489	0.82	2074367
Schwefel 2.21^s	0	-	0	-	0	-	0	-	0	-
Schwefel 2.22^s	**1**	**6732**	1	9145	1	9132	1	8931	1	12008
Schwefel 2.26	**1**	**90753**	1	116924	1	122869	1	115415	1	121479
Sphere^s	**1**	**4920**	1	6762	1	6660	1	6998	1	11006
Step^s	**1**	**55203**	1	74758	1	68296	1	69203	1	71719
Zakharov	0.94	1490104	0.92	1722772	**0.96**	**1620825**	0.82	1815811	0.32	3525365
Mean	**0.80**	**318212**	0.80	373537	0.79	366083	0.77	448341	0.68	566264

Based on Table XI, we notice that the opposition methods hinder BBO's performance on twenty dimensional problems. Their results are not as successful nor as efficient as original BBO. This is contrary to our intuition. Fig 11 displays the best results from the first ten generations of the Schwefel 1.2^s benchmark. In Table XI, we showed that BBO/$\hat{x}_{qr}$ requires 50% more functions calls to solve this problem. However, according to Fig 11, oppositional algorithms perform better than BBO.

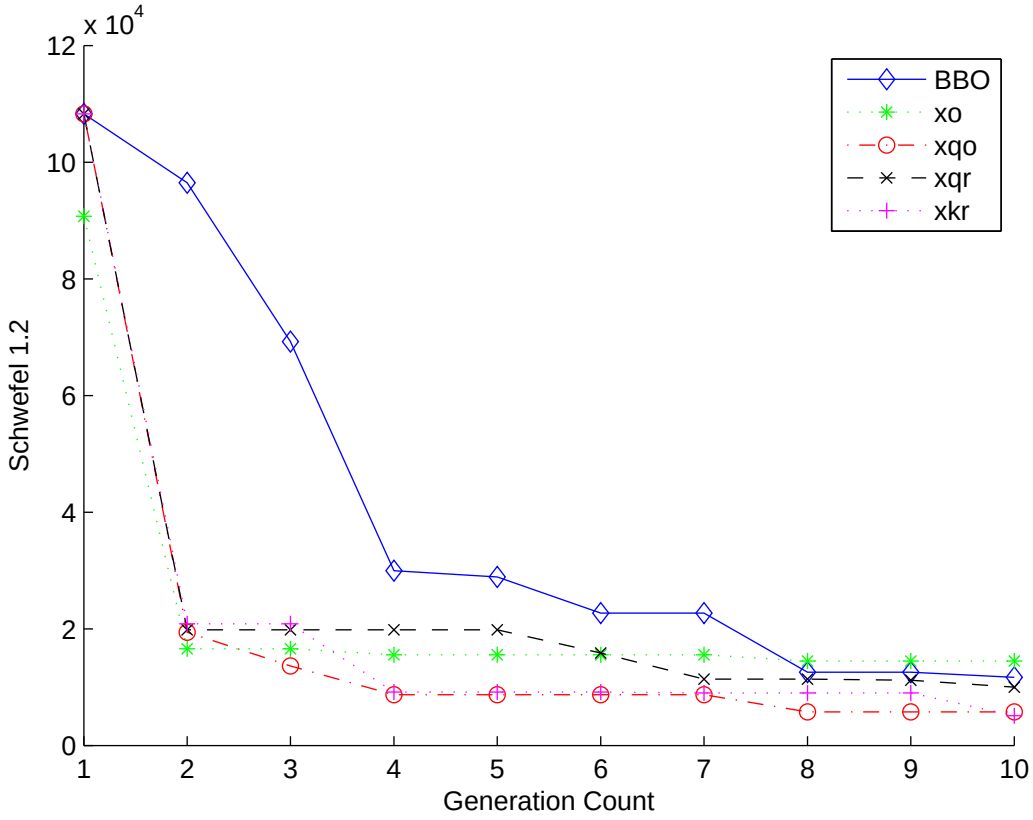


Figure 11. Fist ten generations of best results obtained for Schwefel 1.2^s. Oppositional algorithms start strong.

Fig 12 displays the best results from generations 90-100 for Schwefel 1.2^s. The oppositional algorithms are still in the lead, producing solution candidates that cost less than half of BBO. However, notice that the oppositional algorithms seem to have reached steady-state as they do not continue converging.

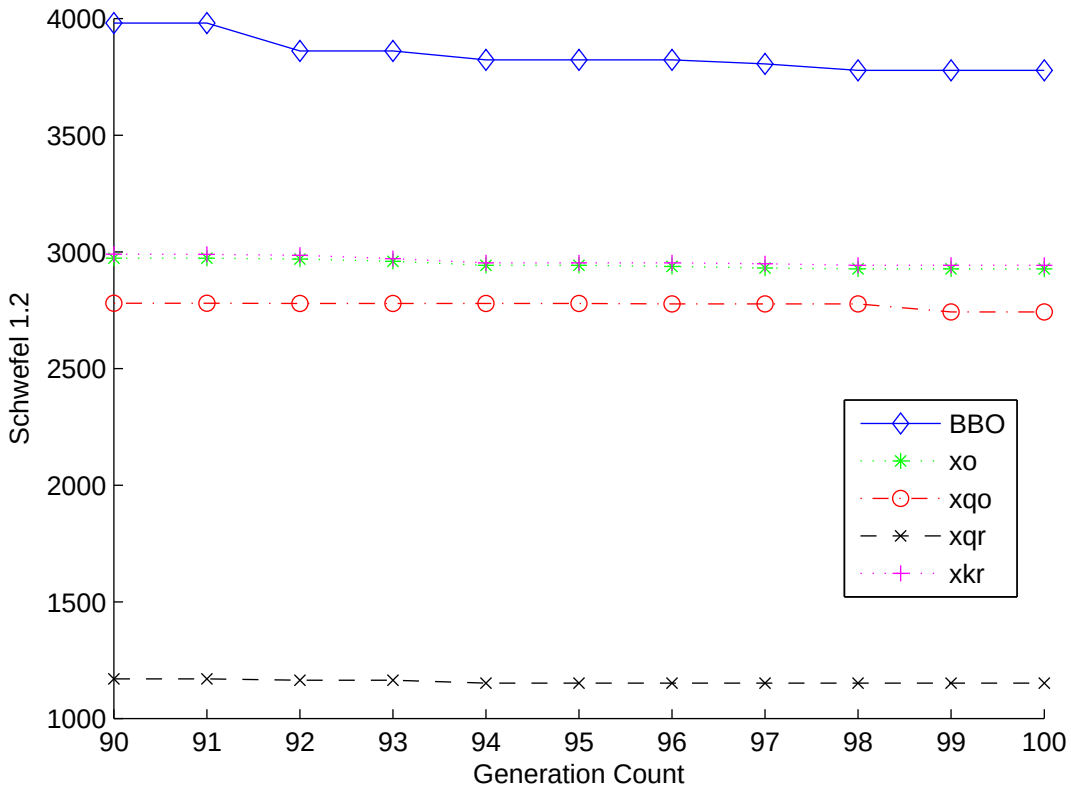


Figure 12. Generations 90-100 of best results obtained for Schwefel 1.2^s. Oppositional algorithms still provide better solution candidates.

Further inspection of the remaining benchmarks suggest that oppositional BBO algorithms fail to converge to the exact solution even though they will approach it closely. In order to test this hypothesis, we ran the simulations for BBO/$\hat{x}_{qr}$ and limited the possibility of opposition to the first 40 generations. The results of our findings are listed in Table XII.

We observe that BBO/$\hat{x}_{qr}^{40}$ is not just a good improvement on BBO/$\hat{x}_{qr}$, it also improves average success rate and number of function calls. Nevertheless, there are still problems such as the quartic function where opposition seems to delay convergence. Thus, a more intelligent oppositional algorithm needs to established. Note that the oppositional generation limit 40 is chosen arbitrarily and further research should be performed on the convergence issues with opposition.

Table XII. The mean of the best results from BBO, BBO/$\hat{x}_{qr}$ for twenty dimensional benchmark problems. BBO/$\hat{x}_{qr}^{40}$ is BBO with quasi-reflection limited to first 40 generations. The maximum number of function calls is limited to 5,000,000. Superscript s indicates that the domain of the problem is shifted randomly for each Monte Carlo simulation. SR is the success rate (that is, the proportion of simulations that found a solution to the desired accuracy). Mean Fc is the average number of function calls before a solution was found.

	BBO		BBO/$\hat{x}_{qr}$		BBO/$\hat{x}_{qr}^{40}$	
Benchmark	SR	Mean Fc	SR	Mean Fc	SR	Mean Fc
Ackley^s	1	19506	1	23933	**1**	**19272**
Alpine^s	1	31174	**1**	**9774**	1	16577
Griewank^s	0.18	537629	0.12	429915	**0.22**	**314295**
Penalty1	1	29193	1	38129	**1**	**24094**
Penalty2	1	26838	1	45110	**1**	**25299**
Quartic^s	**1**	**188232**	0.76	833877	1	341941
Rastrigin^s	**1**	**5121**	1	6853	1	5728
Rosenbrock	0	-	0	-	0	-
Schwefel 1.2^s	0.94	1651347	0.92	2424489	**1**	**1483430**
Schwefel 2.21	0	-	0	-	0	-
Schwefel 2.22^s	**1**	**6732**	1	8931	1	7589
Schwefel 2.26^s	1	90753	1	115415	**1**	**88344**
Sphere^s	**1**	**4920**	1	6998	1	5722
Step^s	1	55203	1	69203	**1**	**53578**
Zakharov	**0.94**	**1490104**	0.82	1815811	0.92	1635391
Mean	0.80	318212	0.77	448341	**0.81**	**309328**

3.3.2 Reflection Range

In this section, we explore the results of different ways of selecting the opposition range for quasi-oppositional algorithms. Reference [144] presents the first findings in domain analysis for oppositional algorithms where the authors introduced dynamic opposition. In dynamic opposition, the opposite point is calculated using the minimum and maximum of a given population, instead of the problem's predefined search domain. We will investigate the affects of dynamic opposition on quasi-reflected and quasi-opposite BBO.

Recall that a quasi-reflected point is calculated by reflecting the individual between itself and the center of the domain. Therefore, we are interested in different mid-domain calculations. The first method, named SM, is static and calculated by finding the midpoint of the problem domains as given in Table IX. The second method is dynamic, referred as DM, and is calculated by finding the center of the domain based on a given population for each problem dimension.

To compare the effects of these two definitions of mid-point, we selected six benchmarks of interest and ran 50 Monte Carlo simulations for each one. The chosen benchmarks have an uneven range or a minimum that is not centered. We employed OBƎO with quasi-reflection, two member elitism and without mutation or reflection weight (see Section 2.3). Our findings are presented in Table XIII .

Table XIII. The effects of static and dynamic population range on quasi-reflection. SM: Midpoint is calculated based on the search domain of the benchmark problem, as defined by Table IX. DM: Midpoint is calculated dynamically for each generation.

	BBO/QR/SM		BBO/QR/DM	
Benchmark Functions	Mean Fc	SR	Mean Fc	SR
Ackley	**1,784**	**1**	63,282	1
Penalty2	52,003	1	**46,886**	**1**
Quartic	**14,635**	**1**	184,537	1
Schwefel 2.21	**2,206**	**1**	3,700,663	1
Schwefel 2.26	145,140	1	**55,416**	**1**
Zakharov	1,870,749	0.8	**1,865,230**	**1**
Mean	347,753	0.97	986,002	1.00
Geometric Mean	30,552	0.96	243,693	1.00

From Table XIII, we note that BBO/QR/DM outperforms the static population by a larger margin on the benchmarks that do not have their minimum at the center of the domain, such as Schwefel 2.26. Also, the dynamic mid-domain calculation increases our success rate on more challenging problems, such as Zakharov, at the cost of performance on simpler problems, such as Ackley.

In Eq. 2.2, we defined a quasi-opposite point as a random point between the center of the domain and the opposite of the individual. Therefore, to calculate a quasi-opposite point, the domain is necessary not only for calculating the midpoint but for calculating the opposite point as well. The solution domain can be defined as:

- The solution domain is same as the given domain of the problem which is defined by the user and in our case given in Table IX. This is the SM

method.

- The domain of the current generation for each independent variable is utilized as the domain for the opposite points. For an n-dimensional problem, at each generation, we would calculate n different domains. Assuming that our solution converges with time, the domain of the problem should shrink. This is the DM method.

- The domain of the current generation for the whole population is utilized as the domain for the opposite points. For an n-dimensional problem, we would calculate the minimum and maximum values at each generation and that would define the domain for the whole population. If the problem is not scaled, users should be wary of this method since different independent variables might have widely varying domains and should not be combined.

Also, for each of these domain definitions, there is a corresponding center point definition. However, the last definition has an unfair advantage compared to the first two, since for our benchmark problems the minimum argument is the same for all of the dimensions. This is not a very realistic scenario and therefore, this option is not included in the following simulations. If we allow the possibility of mixing the first two definitions for domain and midpoint calculations, there will be a potential of four combinations.

To compare the effects of the first two definitions of opposite-point domain and mid-point, we selected the same six benchmarks as for quasi-reflection and ran 50 Monte Carlo simulations for each one. We employed OBƎO with quasi-opposition, two member elitism and without mutation or reflection weight (see Section 2.3). Table XIV presents our findings.

Table XIV. The effects of static and dynamic population range on quasi-opposition. SR/SM: Domain/Midpoint is calculated based on the domain of the benchmark problem, as defined by Table IX. DR/DM: Domain/Midpoint of each independent variable is calculated dynamically for each generation.

Benchmark	BBO/QO/SR/SM		BBO/QO/DR/DM		BBO/QO/SR/DM		BBO/QO/DR/SM	
Functions	Mean Fc	SR	Mean Fc	SR	Mean Fc	SR	Mean Fc	SR
Ackley	**1,806**	**1**	36,170	1	2,784	1	4,527	1
Penalty2	59,022	1	**46,102**	**1**	50,816	1	52,744	1
Quartic	**13,243**	**1**	75,445	1	15,127	1	17,620	1
Schwefel 2.21	**2,298**	**1**	4,459,940	0.54	3,374	1	5,256	1
Schwefel 2.26	149,426	1	**76,255**	**1**	143,826	1	147,226	1
Zakharov	1,817,245	0.76	**1,138,275**	**1**	2,049,700	0.84	2,400,623	0.62
Mean	**340,507**	0.96	972,031	0.92	377,605	**0.97**	437,999	0.94
Geometric Mean	**30,961**	0.96	191,098	0.90	35,865	**0.97**	44,549	0.92

For Schwefel 2.26, a problem with a larger domain and a skewed minimum argument, all algorithms had a 100% success rate, but on average BBO/QO/DR/DM can solve it in approximately half the function calls as the others. For Zakharov, a problem with a smaller non-symmetrical domain, DR/DM is the only method with a perfect success rate. On the other hand, for Schwefel 2.21 DR/DM is the only method that failed to achieve a perfect success rate. We can see that the DR/DM combination is hit or miss since it had the fewest function calls for half of the benchmarks and the most function calls for the rest of them. This shows that there is no single perfect method for all problems [59].

Note that for the non-homogenous matching, such as SR/DM and DR/SM, the midpoint is based on a different domain than the opposite point. For example, if SR/DM method is applied to a problem with SR of $(-100, 100)^n$, but the current generation has a domain average of -10 for a given dimension, then the opposite-reflected point calculated from this data may be skewed.

3.4 Real-world problems

The oppositional algorithms are tested on real-world problems provided by the European Space Agency (ESA). An overview of these problems is summarized in Table XV. More details about the models employed in these problems can be found at [180] and [181]. These problems provide a good benchmark for global optimization as the provided parameters are claimed to be compatible with the current missions of ESA and NASA [182]. These problems model the interplanetary space trajectories from the ESA missions. For instance, the goal of the esa4 problem is to calculate the best possible trajectory that the spacecraft Messenger should follow to orbit Mercury. This problem has nonlinear constraints that are known to cause difficulties for optimization algorithms. For esa5, the maneuvers that will yield the optimal path to Saturn for a fly by mission past Earth, Jupiter and Venus must be determined.

The selected problems were created and made available by ESA and are called as blackbox functions by the EA. EA generated solution candidates with a structure of variables is passed to an ESA function, which then evaluates these variables, handles the constraints when applicable and returns the corresponding objective function value. Some of these trajectory optimization problems are more complete as they include deep-space maneuvers.

Table XV. Overview of ESA global trajectory optimization problems. Min(J) is the best cost value known at the time of this writing.

Problem	Dimension	min(J)	Key
cassini1	6	4.9307	esa1
cassini2	22	8.383	esa2
gtoc1	8	$-1,581,950$	esa3
messenger	18	8.630	esa4
messenger full	26	2.113	esa5
rosetta	22	1.343	esa6
sagas	12	18.19	esa7

3.5 Simulation Settings

The simulation parameters are presented in Table XVI where population size is the number of EA individuals that are maintained each generation. Ideal generation limit is an approximation, assuming a single objective function evaluation per individual per generation. This may not be equal to the actual generation limit for oppositional algorithms since each time an opposite population is generated, its objective function values have to be evaluated. This means that oppositional algorithms require more computational resources for the same number of generations. To make a fair comparison, we set the termination condition to be a specific number of cost function evaluations. Simulations are run until the number of function calls reaches the product of ideal generation limit and population size. Elite population is the number of fittest individuals preserved after each generation. They replace the least fit individuals for the next generation. The blending amount is only employed for GA and BBO [158]. It indicates the recombination weight factor of the parent and the child. The opposition jumping rate is the probability of creating an opposite population per generation.

Table XVI. Simulation settings for real-world problems

Parameter	Value
Population size	100
Ideal generation limit	1000
Elite population	2
Blending amount	0.25
Opposition jumping rate	0.5

3.6 Simulation Results

Results are analyzed by two different approaches. First, in Tables XVII-XIX, we list the minimum for each EA and OBL algorithm, averaged over a set of 25 Monte Carlo simulations. We also include the standard deviations. The best mean is indicated with a boldface typeset. These tables provide us with insight regarding the expected performance of each algorithm. Next, in Tables XX-XXII, we list the best result obtained by each approach. Since, generally, our goal is to find a single optimal trajectory, these tables provide us with the results that would be employed from each algorithm in the real world.

Table XVII. Mean (and standard deviation) of final cost function value after 25 Monte Carlo simulations with GA and its oppositional versions.

Problem	GA	GA/$\hat{x}_o$	GA/$\hat{x}_{qo}$	GA/$\hat{x}_{qr}$
esa1	18.63(4.74)	**10.56(3.88)**	14.42(4.34)	17.95(3.26)
esa2	31.44(4.66)	**24.97(1.68)**	26.54(2.09)	26.32(1.44)
esa3	-280922(185322)	**-768476(218896)**	-461517(368769)	-339683(369646)
esa4	21.92(3.07)	**16.86(1.64)**	17.87(2.73)	19.25(2.20)
esa5	27.16(4.57)	**20.00(1.72)**	24.92(4.72)	27.36(7.60)
esa6	15.71(2.60)	10.25(2.86)	11.97(3.34)	**9.29(2.37)**
esa7	1710.89(160.57)	**1059.45(155.34)**	1131.1(288.32)	1135.52(277.07)

Table XVIII. Mean (standard deviation) of final cost function after 25 Monte Carlo simulations with DE and its oppositional versions.

Problem	DE	DE/$\hat{x}_o$	DE/$\hat{x}_{qo}$	DE/$\hat{x}_{qr}$
esa1	10.36(3.55)	**8.40(3.55)**	11.21(3.02)	11.14(2.98)
esa2	19.76(4.67)	**18.55(2.77)**	18.67(2.39)	20.39(2.26)
esa3	-459008(386101)	-835807(386714)	**-951889(424493)**	-878495(392723)
esa4	15.55(3.41)	**13.01(1.91)**	13.19(2.249)	13.19(2.15)
esa5	22.94(7.11)	15.94(4.37)	**14.49(1.39)**	14.79(1.957)
esa6	7.08(4.06)	5.08(2.70)	4.57(2.49)	**4.56(2.34)**
esa7	987.79(31.34)	951.46(57.78)	964.93(19.19)	**905.72(226.75)**

Table XIX. Mean (standard deviation) of final cost function after 25 Monte Carlo simulations with BBO and its oppositional versions.

Problem	BBO	BBO/$\hat{x}_o$	BBO/$\hat{x}_{qo}$	BBO/$\hat{x}_{qr}$
esa1	19.72(7.52)	**10.73(3.95)**	12.11(4.24)	14.71(3.87)
esa2	30.51(3.928)	**24.75(2.83)**	24.91(1.35)	26.05(2.27)
esa3	-303629(220240)	**-654887(269287)**	-560691(411569)	-548817(446057)
esa4	21.05(2.72)	16.95(1.66)	**16.53(2.05)**	17.62(2.03)
esa5	31.43(7.05)	21.22(3.80)	21.84(4.18)	**20.13(3.38)**
esa6	15.94(3.39)	**10.30(2.86)**	11.02(3.33)	10.54(3.38)
esa7	1711.32(234.26)	937.21(175.90)	889.59(228.82)	**888.73(235.47)**

The GA results, Table XVII, are dominated by GA/$\hat{x}_o$'s performance which provides the lowest mean for six out of seven problems. The other minimum is achieved by GA/$\hat{x}_{qr}$.

The mean results of DE are listed in Table XVIII and the success rate is more uniformly distributed among the OBL algorithms. Three of the best results are obtained by DE/$\hat{x}_o$, two are obtained by DE/$\hat{x}_{qr}$, and two are obtained by DE/$\hat{x}_{qo}$. For esa1, the expected DE performance is better that DE/$\hat{x}_{qo}$ and DE/$\hat{x}_{qr}$. However, DE performs worse than all of its oppositional variations, on

average.

From Table XIX, we observe that the lowest mean for four out of seven problems is obtained with BBO/$\hat{x}_o$, the lowest means for two problems are achieved with BBO/$\hat{x}_{qr}$, and the lowest mean for one problem is achieved with BBO/$\hat{x}_{qo}$. Notice that on average, the expected performance of each OBL is significantly better that BBO.

If we compare the means from the three EAs, we note that DE-based approaches provide the lowest mean cost for six out of seven problems and BBO has the lowest mean for esa7. Among the OBL methods for the three EAs, $\hat{x}_o$ has three of the lowest means while $\hat{x}_{qr}$ and $\hat{x}_{qo}$ each have two. On the other hand, none of the original EAs performed best for any of the seven problems.

Minimum results for GA, Table XX, indicate that GA/$\hat{x}_o$ finds the lowest cost for three of the problems, GA/$\hat{x}_{qo}$ for two of the problems, and GA and GA/$\hat{x}_{qr}$ each find the lowest cost for one of the problems. For esa2, GA returns a lower cost than any of the OBL enhanced algorithms.

Table XX. Minimum cost achieved after 25 Monte Carlo simulations with GA and its oppositional versions.

Problem	GA	GA/$\hat{x}_o$	GA/$\hat{x}_{qo}$	GA/$\hat{x}_{qr}$
esa1	9.21	**5.56**	6.36	8.78
esa2	**19.88**	21.19	23.03	24.05
esa3	-701380	**-1224444**	-1079288	-1005413
esa4	15.96	13.46	**12.36**	14.48
esa5	20.53	**17.33**	17.85	17.55
esa6	10.19	6.17	6.60	**5.00**
esa7	1418.62	981.57	**438.39**	825.47

Minimums achieved by DE, Table XXI, show that DE, DE/$\hat{x}_{qo}$ and DE/$\hat{x}_{qr}$ each find the lowest minimum for two problems while DE/$\hat{x}_o$ only finds the

best minimum for one of the problems. For esa1, all three OBL algorithms seems to get stuck at the same local minima whereas DE reaches a slightly lower cost. For esa7, the minimum achieved by DE/$\hat{x}_{qr}$ is very close to the best-known global minimum (regardless of the low iteration count) and is significantly lower than the other results.

Table XXI. Minimum cost achieved after 25 Monte Carlo simulations with DE and its oppositional versions.

Problem	DE	DE/$\hat{x}_o$	DE/$\hat{x}_{qo}$	DE/$\hat{x}_{qr}$
esa1	**4.93**	5.30	5.30	5.30
esa2	**12.13**	12.81	13.92	13.42
esa3	-1203488	-1309175	**-1452754**	-1358905
esa4	10.47	10.20	10.24	**9.87**
esa5	14.66	**9.61**	11.29	11.29
esa6	1.96	2.04	**1.93**	1.98
esa7	932.61	691.59	932.58	**20.93**

Among the minimums found by BBO, Table XXII, we observe that both BBO/$\hat{x}_{qr}$ and BBO/$\hat{x}_{qo}$ find the best result for three of the problems. BBO/$\hat{x}_o$ reaches a better solution for one of the problems. For esa6 and esa7, OBL algorithms return much lower cost results than BBO.

Table XXII. Minimum cost achieved after 25 Monte Carlo simulations with BBO and its oppositional versions.

Problem	BBO	BBO/$\hat{x}_o$	BBO/$\hat{x}_{qo}$	BBO/$\hat{x}_{qr}$
esa1	6.86	5.66	**5.61**	6.15
esa2	23.61	**16.60**	22.24	24.12
esa3	-776421	-1262728	-1253848	**-1325004**
esa4	16.20	13.98	**13.40**	14.32
esa5	21.15	16.86	15.00	**14.47**
esa6	9.42	5.60	6.09	**3.18**
esa7	1166.76	431.04	**223.10**	253.53

Comparing the minimum results for the three EAs, we observe that DE-based algorithms found the best result for all of the problems. The highest number of minimums achieved by the EAs is about uniformly distributed between DE, DE/$\hat{x}_{qo}$ and DE/$\hat{x}_{qr}$.

3.7 Statistical Tests

In order to analyze the significance of our findings, we perform t-tests on our simulation results. For each problem, we compare the results of the 25 Monte Carlo simulations for the oppositional algorithms to that of the original EA. The null hypothesis is that the mean of the EA and that of the OBL enhanced EA are equal and the alternative hypothesis is that their means are different. Tables XXIII-XXV provide the two-tailed p-value of the t-distribution. We can reject the null hypothesis if the p-value is significant, i.e., less than 0.05.

For our analysis, we assume that the data sets have equal sample size and variance. These assumptions are valid as each data set has 25 data points and the standard deviations listed in Tables XVII-XIX are, in general, of the

same order of magnitude.

The p-values for the GA and DE algorithms are listed in Tables XXIII-XXIV and indicate a similar performance. $\hat{x}_o$ has the lowest p-values and all $\hat{x}_o$ algorithms reject the null hypothesis successfully. On the other hand, there are a few instances where $\hat{x}_{qo}$ or $\hat{x}_{qr}$ fail to reject the null hypothesis. This might be due to the fact that the jumping rate was adjusted based on the performance of $\hat{x}_o$.

Table XXV lists the p-values of the null hypothesis for BBO and its oppositional versions. $\hat{x}_{qr}$ provides the least significant results, while $\hat{x}_o$ has the highest number of null-hypothesis rejections. This, again, could be the result of tuning the oppositional algorithms based on the performance of $\hat{x}_o$. A different opposition rate might provide a higher significance for $\hat{x}_{qr}$ and $\hat{x}_o$.

Table XXIII. P-values of two-tailed t-tests comparing the statistical significance of the GA results to those of each oppositional algorithm. Results that are not statistically significant are shown in bold font.

Problem	GA/$\hat{x}_o$	GA/$\hat{x}_{qo}$	GA/$\hat{x}_{qr}$
esa1	3.18E-8	1.98E-3	**5.57E-1**
esa2	3.94E-8	1.62E-5	3.44E-6
esa3	3.91E-11	3.36E-2	**4.81E-1**
esa4	2.79E-9	9.93E-6	8.90E-4
esa5	2.29E-9	**9.44E-2**	**9.12E-1**
esa6	5.70E-9	5.52E-5	4.37E-12
esa7	0.00E+0	1.48E-11	7.50E-12

Table XXIV. P-values of two-tailed t-test comparing the statistical significance of DE's results to that of each oppositional algorithm. Results that are not statistically significant are shown in bold font.

Problem	DE/$\hat{x}_o$	DE/$\hat{x}_{qo}$	DE/$\hat{x}_{qr}$
esa1	**5.71E-2**	**3.65E-1**	**4.02E-1**
esa2	**2.73E-1**	**3.07E-1**	**5.40E-1**
esa3	1.19E-3	8.47E-5	3.97E-4
esa4	2.08E-3	5.78E-3	5.13E-3
esa5	1.17E-4	4.48E-7	1.33E-6
esa6	4.55E-2	1.12E-2	9.68E-3
esa7	8.09E-3	3.14E-3	**7.94E-2**

Table XXV. P-values of two-tailed t-test comparing the statistical significance of BBO's results to that of each oppositional algorithm. All results are statistically significant.

Problem	BBO/$\hat{x}_o$	BBO/$\hat{x}_{qo}$	BBO/$\hat{x}_{qr}$
esa1	2.97E-6	5.92E-5	4.80E-3
esa2	2.94E-7	1.78E-8	1.06E-5
esa3	6.83E-6	8.30E-3	1.74E-2
esa4	5.73E-8	2.74E-8	6.91E-6
esa5	6.79E-8	4.28E-7	3.38E-9
esa6	7.37E-8	4.49E-6	9.02E-7
esa7	0.00E+0	0.00E+0	2.22E-16

CHAPTER IV

DISCRETE AND COMBINATORIAL OPPOSITION

The previous chapters discussed continuous domain optimization problems. Recently, there has been research to extend BBO to combinatorial problems such as the traveling salesman problem (TSP) [183, 184, 185]. Oppositional learning, created for accelerating continuous search spaces, can also be modified and integrated with BBO to solve combinatorial problems, such as graph-coloring and TSP.

We recognize that applying opposition to a TSP path by simply reversing that path is meaningless because the reversed path will yield the same cost as the original path. For example, in a TSP problem, if a tour between cities (1, 2, 3, 4) has a cost of c, so would its opposite, (4, 3, 2, 1) because all of the cities preserve their neighbors. Therefore a new definition of opposition is needed. For TSP problems, we define an opposite path as a path that seeks to (or approximately) maximizes the distance between the adjacent vertices in the original path. Based on this definition, a tour may have more than one possible opposite.

We propose two new definitions of opposition in discrete space. The first

proposed definition is for open graph problems, where the final node may be disconnected from the first node, such as the graph-coloring problem and is presented in Appendix 4.1. The latter opposition method is for closed walk problems, where the endpoints of the graph are linked, such as the traveling salesman problem. We named this method cycle opposition and introduce it in Appendix 4.2. The combinatorial biogeography-based optimization is proposed in Appendix 4.3. The effectiveness of these algorithms are tested on vertex coloring and traveling salesman problems and their results are discussed in Appendix 4.4. Appendix 4.5 lists possible directions for future research.

4.1 Open-path Opposition

The first method of opposition for discrete domain problems that we propose is open-path opposition. Open path indicates that we complete the path when we reach the last vertex on the path. An example of such a problem would be the vertex coloring problem. Refer to Section 4.4.1 for more information on graph coloring.

In order to implement open-path opposition, proximities between nodes are calculated. If nodes share an edge so that they are directly connected, their proximity is taken as one. If nodes connect through another node, their distance is two; and if nodes connect through two nodes, their proximity is three; and so on. Consider a path of four nodes, sorted as (1, 2, 3, 4). Table XXVI lists the proximity between each nodes.

Table XXVI. Distances of nodes (1, 2, 3, 4) for calculating the opposite path.

Node 1	Node 2	Proximity
1	2	1
1	3	2
1	4	3
2	3	1
2	4	2
3	4	1

The opposite of this path would be a path that maximizes the proximity between adjacent nodes while minimizing the proximity between further nodes. Table XXVII lists the original path and its calculated opposite. Numbers above the arrows indicate the proximity between the nodes in the original path as shown in Table XXVI. The goal of open-path opposition is to maximize the total proximity traveled by a path by spreading the adjacent nodes apart. We can say that the greater the total proximity, the greater is the opposition. The maximum opposite achievable for our example is seven and it is shown in the table as the exact opposite path. A lesser opposite path, named greedy opposite, is also shown in the table. The greedy opposite path uses a greedy algorithm to quickly calculate the approximate opposite of a given path; however, it might not yield the highest degree of opposition.

Table XXVII. Opposite path of nodes in a tour (1, 2, 3, 4) .

Tour	Path with proximities	Total Proximity
Original Path	$1 \xrightarrow{1} 2 \xrightarrow{1} 3 \xrightarrow{1} 4$	3
Exact Opposite	$3 \xrightarrow{2} 1 \xrightarrow{3} 4 \xrightarrow{2} 2$	7
Greedy Opposite	$1 \xrightarrow{3} 4 \xrightarrow{2} 2 \xrightarrow{1} 3$	6

Notice that calculating the optimal or exact opposite is a combinatorial

problem of its own; therefore a greedy approximation is developed. The greedy opposition is implemented to maximize the proximity one city at a time. For this example, based on Table XXVI, nodes 1 and 4 have the highest distance between them, so they start the greedy opposite tour. Then, we find the node with the highest distance that can continue the tour, node 2, and continue until the tour is completed. Because the greedy algorithm seeks the local optimum at each step, it is unsuccessful in finding the exact opposite even for such a small problem.

Since there is no randomness involved in the definition of opposite path, a greedy opposite path can be defined at the beginning of a program based on node count and the opposite population can always be created based on this path to save processing time. Reconsider our example of 4 nodes. Seeing that the output of the greedy opposition algorithm is deterministic, we can use our greedy path from Table XXVII to calculate the opposite of any other 4-node path. To do this, we refer to (1, 2, 3, 4) as a list of node indices, instead of a list of nodes. Therefore, we can map any 4-node map to its opposite.

For a given number of variables in a combinatorial problem, we can calculate its greedy opposite by using Algorithm 7.

Algorithm 7 Open-path greedy opposite algorithm

procedure GREEDY OPPOSITE PATH(n) ▷ n is the number of nodes
  Initialize odd index counter, $odd_{idx} = 1$
  Initialize even index counter, $even_{idx} = n$
  for each node index c_i from 1 to n **do**
    if c_i is odd **then**
      Opposite node index $O_{c_i} = odd_{idx}$
      Increment odd_{idx}
    else c_i is even
      $O_{c_i} = even_{idx}$
      Decrement $even_{idx}$
    end if
  end for
  return O_{c_i}
end procedure

For the 4-node problem, the presented greedy algorithm would yield the greedy opposite path: $1 \rightarrow 4 \rightarrow 2 \rightarrow 3$. This greedy algorithm can be used to accelerate the convergence rate of various combinatorial problems, including the graph-coloring problem. Also, as possible future work, different mapping algorithms that create different degrees of opposition can be developed, similar to $\hat{x}_{qr}$, $\hat{x}_{qo}$ and $\hat{x}_{Kr}$, and their statistical significance can be analyzed.

4.2 Cycle Opposition

In the previous section, we discussed opposition on a open path. However, some problems, such as the symmetric TSP, are closed since the endpoints of the graph are connected. Open-path opposition will not yield a high degree of opposition for these cases as it assumes that the extreme vertices have maximal separation when they are actually adjacent. Therefore, here we propose

opposite cycle as an alternative to opposite path for problems with closed paths.

On a symmetric TSP, starting at any city on a path, moving in either direction, we will return to our starting point and travel the same amount. Thus, a closed path can be seen as a circular tour to reflect the symmetry of progressing in opposite directions on the path and yet returning to the same point and traveling the same distance. Fig. 13 illustrates a symmetric TSP with eight cities on a circular path. This is an intuitive representation of this problem.

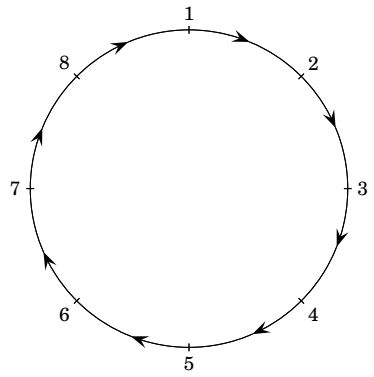


Figure 13. 8-city closed path problem where the path is represented as a circle.

Based on Fig. 13, we can see that to maximize the proximity between the adjacent vertices, we must travel to the opposite side of the circle. This is our definition of opposition for problems with closed path. Fig. 14 illustrates the opposite of each city in the tour.

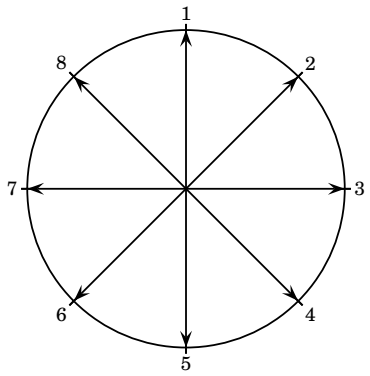


Figure 14. 8-city closed path problem with opposite cities indicated across the circular path.

Although Fig. 14 shows the opposite of each city, it does not indicate an opposite path. It reveals that if we start at city 1, its opposite is city 5. But where do we go from there? The opposite of city 5 is 1, but we cannot revisit the same city. The next best thing for us is to travel to city 2 or 8 since both yield the same amount of opposition. We can choose either of these cities randomly or based on the opposition order, which is explained below. We continue this process until all cities are visited.

We can define permutations on our opposite circuits based on the direction in which we move around the circular path. We call this the order of opposition and four possibilities of it are presented in Table XXVIII. These permutations are named according to the direction we choose to advance. For example, CCW opposite indicates that after reaching an opposite city, we would always move counter-clockwise around the circle to progress on the path. Thus, after we visit city 5, we would start moving counter-clockwise to find the furthest vertex, in this case city 2. The CW opposite is similar, but advances in the clockwise direction to form an opposite cycle. Notice that both the CW and CCW paths following our choice on city 5 are mirror images of each other and yield the same amount of opposition. We can define the CW opposite path as follows.

Definition Let n be the number of nodes in a graph and P be a cycle with an

even number of nodes n. CW opposite path, P_o^{CW} of P is defined as

$$\begin{aligned} P &= [1, 2, \ldots, n] \\ P_o^{CW} &= \left[1, 1+\frac{n}{2}, 2, 2+\frac{n}{2}, \ldots, \frac{n}{2}-1, n-1, \frac{n}{2}, n\right] \end{aligned} \tag{4.1}$$

The other two techniques, CW-CCW and CCW-CW oppositions, reverse direction after each decision. So if CW-CCW opposition moves clockwise to get to city 2, it would then advance counter-clockwise and link to city 6. Notice that CW-CCW and CCW-CW oppositions create less opposition as we progress around the circle. Table XXVIII lists the possible CW-CCW and CCW-CW cycles for a 8-city TSP.

Table XXVIII. Permutations of opposite tour of cities (1, 2, . . . , 7, 8). Tours are named after the direction followed around the opposition circle after each city visit. For example, CW opposite indicates that from the current location, we must travel clockwise around the circle to find the largest opposition.

Path Name	Path Followed	Total Proximity
Original path	$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8$	7
CW Opposite	$1 \to 5 \to 2 \to 6 \to 3 \to 7 \to 4 \to 8$	31
CCW Opposite	$1 \to 5 \to 8 \to 4 \to 7 \to 3 \to 6 \to 2$	25
CW-CCW Opposite	$1 \to 5 \to 2 \to 6 \to 8 \to 4 \to 3 \to 7$	30
CCW-CW Opposite	$1 \to 5 \to 8 \to 4 \to 2 \to 6 \to 7 \to 3$	26

Notice that we cannot assign opposite cities as defined in Eq. 4.1 if n is odd. If we follow the opposite circle (Fig. 14) in an odd-length cycle, the opposite point would end up being between two cities. Then, the CW or CCW option would specify which direction to travel around the circle to find the opposite city.

One way of implementing CW opposition in odd-length graphs is to add an auxiliary node to the end of the path to force the city count to an even number. We then calculate the CW opposite of the tour and remove the auxiliary

city from the end of tour. This procedure yields the same result as following the opposite circle in the CW direction to find the opposite city.

Algorithm 8 lists the pseudocode for generating the CW opposite path for even- and odd-length TSP cycles. In this algorithm, we define the middle node to be the reflection point, r_p, calculate opposite cities based on r_p and link every city to its opposite. As future work, different reflection points can be selected to create different levels of opposition, analogous to $\hat{x}_{qo}$ and $\hat{x}_{qr}$ in the continuous domain.

Algorithm 8 CW opposite cycle

procedure CW OPPOSITION(n) ▷ n is the number of nodes
  if n is odd **then**
    $v = n + 1$
  else n is even
    $v = n$
  end if
  $r_p = \frac{v}{2}$ ▷ r_p is the reflection point
  $idx = 1$
  while $v_i \leq r_p$ **do**
    $O_{idx} = v_i$ ▷ O is the opposite cycle
    $O_{idx+1} = v_i + r_p$
    $idx = idx + 2$
  end while
  if n is odd **then**
    Remove last node from O
  end if
  return O
end procedure

4.3 Combinatorial Biogeography-based Optimization

To solve an optimization problem in a continuous domain, we search for the best solution that exists within a given domain. Combinatorial problems, such as the ones discussed in this chapter, are ordering type problems. We are given a list of all vertices that must be part of the solution and we are to find the best sequence of these vertices that will minimize the cost function. In this chapter, we follow Du's TSP migration pattern which is inspired by the inver-over operator [186].

In the spirit of BBO, all the islands are assigned emigration and immigration rates based on their fitness. We then perform roulette wheel to select an immigrating and an emigrating island, I_i and I_e, and randomly choose a city in the immigrating island to be our migration point, M_p. Next, we seek the migration point in the emigrating island and locate the adjacent vertex as the flipping point, F_p. A new island is created from the immigrating island by flipping the sequence of vertices between M_p and F_p. Algorithm 9 demonstrates the pseudocode for combinatorial BBO.

Algorithm 9 Combinatorial BBO migration

1: **procedure** MIGRATION(I_i, I_e)
2: $M_p = \text{rand}(I_i(city))$ ▷ Random migration point
3: $F_p = I_e(M_p + 1)$ ▷ Flip point is adjacent to M_p
4: $I_{new} =$Flip $I_i(M_p + 1 : Fc)$
5: **return** I_{new}
6: **end procedure**

Algorithm 9 can be illustrated with the following example. Let the randomly selected migration point be $M_p = 3$ and immigrating and emigrating

islands be

$$I_i = [1 \rightarrow 3^{M_p} \rightarrow 4 \rightarrow 6 \rightarrow 2 \rightarrow 5]$$

$$I_e = [6 \rightarrow 4 \rightarrow 3 \rightarrow 2^{F_p} \rightarrow 1 \rightarrow 5]$$

Considering that in I_e, M_p is followed by 2, $F_p = 2$. We then flip the cities between M_p and F_p in I_i to follow the same sequence from I_e and obtain

$$I_{new} = [1 \rightarrow 3^{M_p} \rightarrow 2^{F_p} \rightarrow 6 \rightarrow 4 \rightarrow 5]$$

4.4 Experimental Results

All benchmark problems for vertex coloring and traveling salesman problems, are simulated in MATLAB® with the settings listed in Table XXIX. The tabulated results are the best findings over 20 independent Monte Carlo simulations at the end of 100 generations.

Table XXIX. Simulation settings for graph-coloring problems.

Variable	Value
Population size	50
Generation limit	100
Number of elites	3
Monte Carlo runs	20

4.4.1 Vertex Coloring

We selected vertex coloring [187] as our combinatorial benchmark because it is the most popular graph-coloring problem. Furthermore, other coloring problems can be transformed into vertex coloring. Graph-coloring has many real-world applications related to scheduling including register allocation [188], wireless network testing [189] and final exam timetables at universities [190].

In vertex coloring, we are giving a a graph $G(V, E)$ denoting a list of countries on a map (vertices) and their neighbors (edges). The neighboring cities are represented as vertices that are linked with an edge. Connected vertices cannot share the same color. The goal is to find the minimum number of colors needed to color the vertices. This number is denoted as the chromatic number, $\chi(G)$. Vertex coloring is an NP-complete problem.

Fig. 15 illustrates a 3-color graph-coloring problem and its solution. In this problem, there are eight countries (vertices) and 13 connections (edges). The minimum number of colors needed is $\chi(G) = 3$.

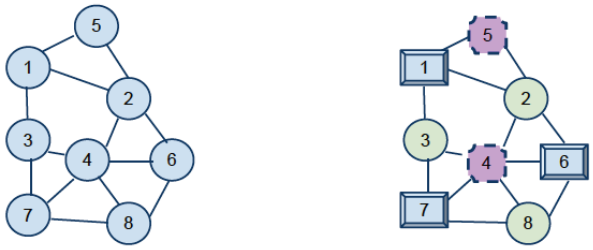


Figure 15. Example of a three-color map with eight vertices and 13 edges. The figure on the right is the properly colored map.

Various evolutionary approaches have been created to solve the graph-coloring problem [191, 192, 193]. Our method is a hybrid between an evolutionary algorithm (BBO) and the greedy algorithm described in Algorithm 11. The role of BBO is to sort the list of countries and to provide this re-ordered list to the greedy algorithm which quickly assigns a color to each country. This simple methodology does not guarantee that an optimal solution is found, but it stores the vertices as a list so that open-path opposition can be easily applied.

Each BBO individual in the population stores a list of vertices as its solution features (islands). Vertices are rearranged from one generation to the next and conveyed to the greedy algorithm to minimize the chromatic number. Algorithm 10 outlines the hybrid BBO/Greedy algorithm.

Algorithm 10 Vertex coloring with BBO

```
1: procedure BBO VERTEX(V, E)
2:     Initialize population by shuffling the order of vertices
3:     while Generation count is not reached do
4:         BBO migration for each individual
5:         Cost function calls Greedy Vertex (Algorithm 11)
6:     end while
7: end procedure
```

The goal of the greedy algorithm is to quickly assign a valid color to each country based on the order of vertices generated by BBO. Algorithm 11 presents the pseudocode for the greedy vertex coloring algorithm.

Algorithm 11 Greedy vertex coloring

```
1: procedure GREEDY VERTEX(V, E)
2:     for Each vertex do
3:         Find all of its neighbors
4:         Find the colors of all the neighbors
5:         Assign the smallest available color index not assigned to a neighbor
6:     end for
7:     return number of colors
8: end procedure
```

Table XXX lists the benchmark problems borrowed from [194] which are assembled from various resources, including [195, 191, 196]. The table lists the number of vertices and edges for each problem along with the chromatic number, $\chi(G)$, if one was available.

Table XXX. List of benchmark problems along with their optimal solution for vertex coloring. "NA" indicates not available (i.e., not known).

Benchmark	$\chi(G)$	# Vertices	# Edges
anna	11	138	493
david	11	87	406
DSJC125.1	NA	125	1472
DSJR500.1	NA	500	7110
games120	9	120	638
huck	11	74	301
le450.5a	5	450	5714
miles750	31	128	2113
myciel3	4	11	20
myciel4	5	23	71
myciel5	6	47	236
myciel6	7	95	755
queen10.10	NA	100	2940
queen11.11	11	121	3960
queen5.5	5	25	160
queen6.6	7	36	290
queen7.7	7	49	476

Simulation results for graph-coloring benchmarks are depicted in Table XXXI. These are the best results obtained from each algorithm after 20 independent Monte Carlo simulations. We note that BBO augmented with open-path opposition (BBO/OPO) performs no worse than BBO. BBO/OPO achieved a better minimum than BBO for three of the benchmark problems and both algorithms reached the optimal solution for 6 of the problems.

BBO/OPO is a hybrid between BBO and a greedy algorithm and thus, BBO migration might not have been as effective as it could. As future work,

BBO/OPO can be restructured to solve the graph-coloring problem without the help of the greedy algorithm.

Table XXXI. Best results obtained by BBO and BBO/OPO (open-path opposition) algorithms after 100 generations for graph-coloring problems.

Benchmark	BBO	BBO/OPO
anna	11	11
david	11	11
DSJC125.1	11	11
DSJR500.1	19	19
games120	11	11
huck	11	11
le450.5a	31	**30**
miles750	35	35
myciel3	4	4
myciel4	5	5
myciel5	6	6
myciel6	12	**10**
queen10.10	22	22
queen11.11	26	26
queen5.5	7	7
queen6.6	10	**9**
queen7.7	13	13

4.4.2 Traveling Salesman Problem

The traveling salesman problem (TSP) [197] is a well-known closed path combinatorial problem. The TSP is classified as a NP-hard problem and currently there is no polynomial-time algorithm that can guarantee an optimal solution. In the TSP, we are given a list of cities and their coordinates. We

sort this list to minimize the length of the path traveled while visiting each city only once and returning to the starting city. This problem is based on the challenge faced by the traveling salesman who tries to find the shortest route which would allow him to visit all the cities once before returning to the departure city. The TSP represents many real-world applications such as vehicle routing (i.e., for postal services or buses) [198, 199, 200], and printed circuit board (PCB) drilling problems [201, 202]. For instance, to manufacture a PCB, tens of thousands of holes must be drilled to place components. The solution of the TSP, where the cities represent the holes, would portray the path the drill must follow from one hole to the next.

In this section, we will focus solely on the symmetric traveling salesman problem where the distance between two nodes is identical when traveling from either direction. The set of TSP benchmark problems employed are borrowed from TSPLIB [203]. Table XXXII lists these benchmark problems, their dimensions and minimum costs. For our simulations, we chose to implement clockwise (CW) circular opposition, Table XXVIII, as our opposite algorithm.

Table XXXII. Symmetric TSP benchmark problems and their optimal results as posted by TSPLIB [203].

Benchmark	Optimal Solution	Dimension
att532	27686	532
berlin52	7542	52
bier127	118282	127
ch130	6110	130
d18512	645238	18512
gr202	40160	202
kroA150	26524	150
kroA200	29368	200
kroC100	20749	100
lin105	14379	105
lin318	42029	318
p654	34643	654
rat575	6773	575
st70	675	70
usa13509	19982859	13509
vm1084	239297	1084

The best results obtained from both algorithms is represented in Table XXXIII along with their geometric mean. BBO with CW circular opposition, BBO/CO, is able to find a shorter route for 14 of the benchmark problems while BBO had a better route for 1 problem.

Table XXXIII. Best results obtained by BBO and BBO with circular opposition (BBO/CO) for symmetric TSP benchmark problems.

Benchmark	BBO	BBO/CO
att532	1413346	**1377743**
berlin52	14493	14493
bier127	455327	**444256**
ch130	**31226**	31955
d18512	58765326	**58590744**
gr202	2467	**2443**
kroA150	175305	**168672**
kroA200	253296	**244031**
kroC100	98980	**95513**
lin105	72946	**72162**
lin318	481978	**480803**
p654	1765633	**1742126**
rat575	99960	**98846**
st70	1956	**1880**
usa13509	2123405375	**2115346146**
vm1084	7918373	**7891132**
Geometric Mean	343679	**338639**

4.5 Conclusions on Combinatorics

In this section, we introduced open-path and circular opposition techniques to assist our evolutionary algorithm, BBO, to solve combinatorial optimization problems. The objective of both opposition methods was to create an opposite path by maximizing the proximity between adjacent nodes. The open-path opposition was developed for open-ended combinatorics and was tested on

17 graph-coloring problems. BBO was able to reach the optimal solution in 6 of these benchmarks without the aid of opposition, while open-path opposition surpassed BBO on three of the remaining problems.

The circular opposition technique was developed for graphs where the last node was connected to the first one. The circular opposition was tested on 16 traveling salesman problems and was found to outperform standard BBO in 14 of them.

Further research could focus on combining the proposed methods with other EAs for combinatorial opposition and exploring different degrees of opposition for open- and closed-path combinatorics. Also, future research efforts could concentrate on removing BBO's dependency on the greedy algorithm in the graph-coloring problem. Effects of such modifications on open-path opposition's performance should be investigated.

CHAPTER V

CONCLUSIONS AND FUTURE WORK

THIS section provides concluding remarks as well as direction for future work. Section 5.1 discusses the extension of the mathematical proofs to higher dimensions. Section 5.2 proposes a method to handle constraints with BBO and Section 5.3 explores possible extensions to BBO inspired by biogeography and coevolution.

Evolutionary algorithms are tools for heuristically solving global optimization problems. As new approaches are developed, their success is commonly measured based on empirical analysis. In this research, we developed the mathematical proofs that allow us to quantify the effectiveness of employing opposite points in EAs. We derived the probability that the distance between an OBL point and the solution is less than the distance between an EA solution candidate and the solution. Our investigations for three OBL algorithms (opposition, quasi-opposition and quasi-reflection) showed that quasi-reflection is the most likely OBL method to be closer to the solution of an optimization problem.

We also modified the quasi-reflection algorithm to allow the opposition amount to be a function of the solution candidate's ranking. This algorithm is named fitness-weighted quasi-reflection. We obtained the probability of $\hat{x}_{Kr}$being

closer than an EA individual to the solution as a function of the reflection weight. We then derived the expected distance to the solution and concluded that the probability of being closer to the solution and the expected distance to the solution both decrease with the reflection weight.

After the theoretical analysis, we compared the performance of these OBL algorithms on three popular EAs (GA, DE, BBO) through empirical studies. As benchmark problems, we selected seven space trajectory problems provided by the ESA, as well as 22 well-known problems from the literature, and showed the statistical significance of our results. For lower dimensional problems, we found that compared to BBO, BBO/$\hat{x}_o$, reduces the number of function calls necessary by 96.5% while providing a higher success rate. Inspection of the variable-dimensional benchmarks suggested that oppositional BBO algorithms failed to converge to the exact solution even though they approach it closely. By limiting opposition to first 40 generations for BBO/$\hat{x}_{qr}$, we reduced its number of function calls by 31% and increased its average success rate by 4% . Thus, for future work, a more intelligent oppositional jumping-rate algorithm needs to established. For the ESA problems, we found that, on average, none of the original EAs (GA, DE or BBO) could outperform any of the oppositional algorithms.

5.1 Opposition Probabilities in Higher Dimensions

In Chapter 2, we defined opposite points in one-dimensional space. Then, we derived the probability of the opposite of a point being closer than the point itself to the solution. We extended this proof for the quasi-opposite points, again for one-dimensional problems. However, since meta-heuristic algorithms, such as OBBO, are generally employed for multidimensional problems, we need to show the validity of our results in higher dimensions.

In Fig. 16, we present the probabilities of success as the problem dimension increases. Figure legends have been abbreviated for clarification purposes. $\Pr[\hat{x}_o, \hat{x}]$ is shorthand for $\Pr[|\hat{x}_o - x| < |\hat{x} - x|]$ or the probability of $\hat{x}_o$ being closer than $\hat{x}$ to the solution. These results are obtained using a MATLAB simulation as as described in Algorithm 12.

Algorithm 12 Pseudocode for simulating the high dimensional probabilities of the oppositional algorithms

```
for dimensions between 1 to 100 do
    for 201 uniformly distributed solution candidates, x ∈ [a, b] do
        Randomly select 5000 points as EA individuals, x̂
        Compute their corresponding opposite points, x̂_o , x̂_qo and x̂_qr , as defined in Chapter II
        Calculate the distances between each point and the solution
        if the opposite point is closer than x̂ to x then
            Increment counter
        end if
    end for
end for
```

According to these findings, for a 20-dimensional problem, such as the ones presented in Section 3.2, the quasi-reflected estimate has a 91% probability of being closer than the EA individual to the solution. More importantly, Fig. 16 demonstrates that the effectiveness of quasi populations increases with the problem dimension.

Based on Fig. 16 and the empirical results presented on Section 3.3, conjecture that the theorems presented in Section 2.5 are qualitatively valid in higher dimensions. However, future research can focus on extending the mathematical proofs to support these findings.

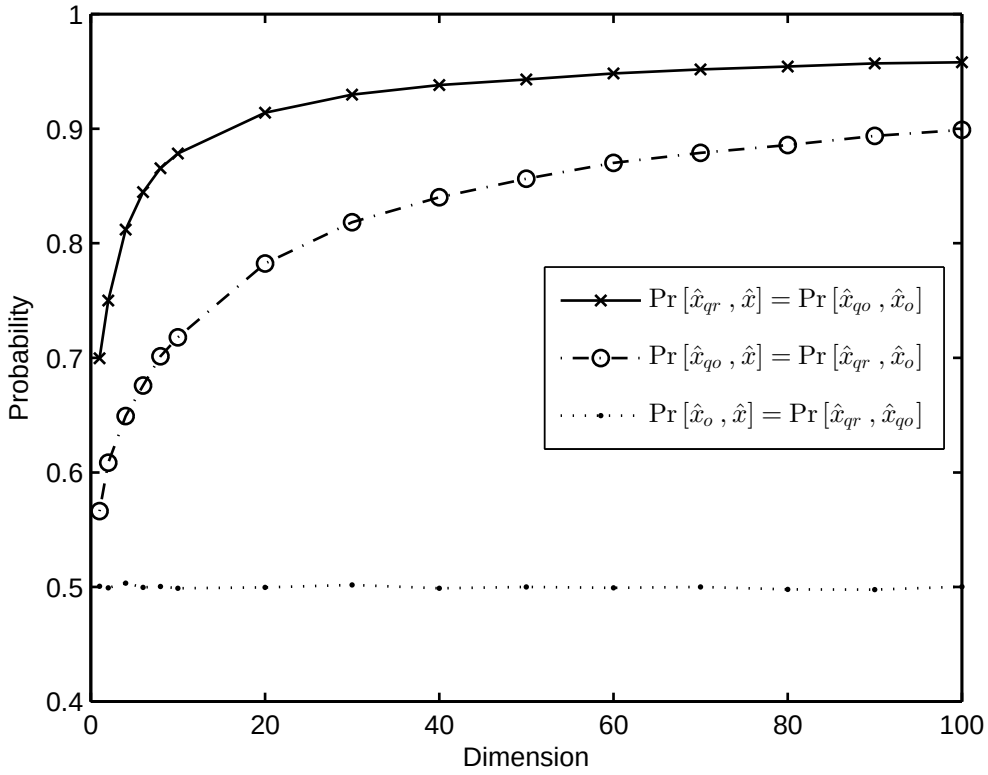


Figure 16. Effects of dimension on the probabilities of various opposition methods. $\Pr[\hat{x}_{qr}, \hat{x}])$ is the probability that $\hat{x}_{qr}$ is closer to the solution than $\hat{x}$. Other legends can be read similarly.

5.2 Constrained Optimization

Nonlinear programming problems (NLP) with numerous constraints are complicated and [204] argues that it is impossible to develop a single deterministic method that would work effectively for all NLPs. He argues that such a study would concludes in performing exhaustive search which is computationally expensive. Therefore, meta-heuristic search algorithms, such as BBO, are commonly employed in solving constrained optimization problems.

A modified version of BBO, blended BBO, has been employed for constrained optimization in [158]. One generation of blended BBO where $\alpha \in [0, 1]$ is the blending parameter is outlined in Algorithm 13.

Algorithm 13 Pseudocode for blended migration

```
for each solution candidate S_i do
    for each parameter p do
        Select an immigrating variable, S_i(p)
        Select an emigrating variable, S_k(p)
        S_i(p) ← αS_i(p) + (1 − α)S_k(p)
    end for
end for
```

The definition for constrained optimization is the following. Given the objective function f, feasible region F and search space S, our goal is to

$$\text{optimize } f(\vec{x}), \text{ s.t. } \vec{x} = (x_1, \ldots, x_n) \in \Re^n$$
$$\text{and } \vec{x} \in F \subseteq S$$

where the feasible region has m constraints. q of these constraints are inequality constraints and the rest of them are equality constraints:

$$\begin{aligned} g_j(\vec{x}) &\leq 0, \text{ for } j = (1, \ldots, q) \\ h_j(\vec{x}) &= 0, \text{ for } j = (q+1, \ldots, m) \end{aligned}$$

These equality constraints are commonly rewritten as inequality constraints

$$h_j(\vec{x}) \leq \epsilon \text{ and } h_j(\vec{x}) \geq -\epsilon$$

for small $\epsilon > 0$.

A comprehensive survey of existing constrained optimization methods is presented in [205]. A large number of real-world problems involve constraints and these constraints are generally handled by penalizing the infeasible solutions based on the distance from the feasible region. However other methods such as decoders and separation of feasible and infeasible solutions also exist in the literature. A future research goal is to select one these methods and combine it with oppositional theory to create a new constraint handling algorithm. We could then study the effects of oppositional BBO on constrained optimization problems and compare with existing methods.

5.3 Biogeographical Extensions

BBO is inspired by biogeography and can be extended by its discoveries. We can acquire motivations from biogeography, such as the effects of island isolation, island size, the types of islands created or coevolution (for multi-objective optimization).

In future work, I would like to focus on a more specific type of ecology: island biogeography. From the point of view of species, islands have a special place in biogeography. They host species that are endemic, native to the island or archipelago and exist exclusively on their native land and provide a lot more species than mainlands proportional to their size [206]. Because of these characteristics they are referred as “biodiversity hotspots". Since BBO models the migration of species, we can study island biogeography and supplement BBO with the developments in island biogeography.

In island biogeography, islands are classified in two categories based on their formation [207]. Continental islands are pieces of mainland that get isolated, so they already accommodate species before they are formed. The other type of islands are called oceanic. Oceanic islands are formed by the elevation of the ocean floor and they are devoid of any species when they are formed. As time passes, continental islands fail to retain their original number of species whereas the oceanic islands gain species. In BBO, all islands (candidate solutions) are created in the same manner at the beginning of the program. One way to implement the differences in island formation in BBO could be to create inhabited, oceanic islands during optimization and give them time to evolve their own endemic species. Also, we can create new continental islands by separating islands from the main population. These new continental islands can contain a subset of species from their “mainland".

Other dichotomies in island speciation are related to the age, size and isolation of the islands. Evolution requires isolation and extensive periods of time [207] so an island might host a larger number of species as it gets older.

In BBO this idea can be integrated with the immigration and emigration functions. For example, as islands get more mature, their emigration rate can increase. On the other hand, a newly formed island should be more open to immigration even if it has a high fitness value.

Another effect that helps determine the species richness on an island is the area of an island. The number of species hosted by an island is directly proportional to the island size [208]. The last effect that we examine is called the distance effect. An example is presented in [209] where the author compares the number of species on islands of equivalent area and finds that the further the island is from the mainland, the fewer species it hosts. According his results, an island 2000 km away from the mainland is expected to have half the species found on an island near the mainland. Thus, immigration rates are inversely proportional to the island's isolation. In order to apply size and isolation affects in BBO, further research must be conducted to develop emergent/adaptive species (independent solution variables) where the number of individuals vary during the program.

Furthermore, future work can focus on coevolution where multiple sets of populations evolve independently. Individuals in these populations can cooperate and compete with each other. Definitions presented for simulated cooperative coevolution by different authors are conflicting. We examine Potter and De Jong's model [210] because, like BBO, it is based on the evolution of species. In this model, each variable in a given problem is represented by a different species and each species evolves separately in its own population. After each iteration, a group of representatives are selected from their own population to form solution candidates to be evaluated. This scheme motivates the species to compete with each other in their own population to be selected as representatives and it encourages different species to cooperate to survive. Fig. 17 illustrates an iteration of the cooperative coevolutionary evolution [211]. His simulations showed that coevolution reduces computational costs. Also, due to the modularity of the algorithm, it would be a strong candidate for paral-

lel programming. The coevolution theory can be also be combined with the archipelago algorithm presented by [212]. An OBL approach to coevolution is presented in [213] where the authors find a Pareto front for multi-objective optimization. They implement opposition in the competitive level by creating a set of opponents to compete against the representatives.

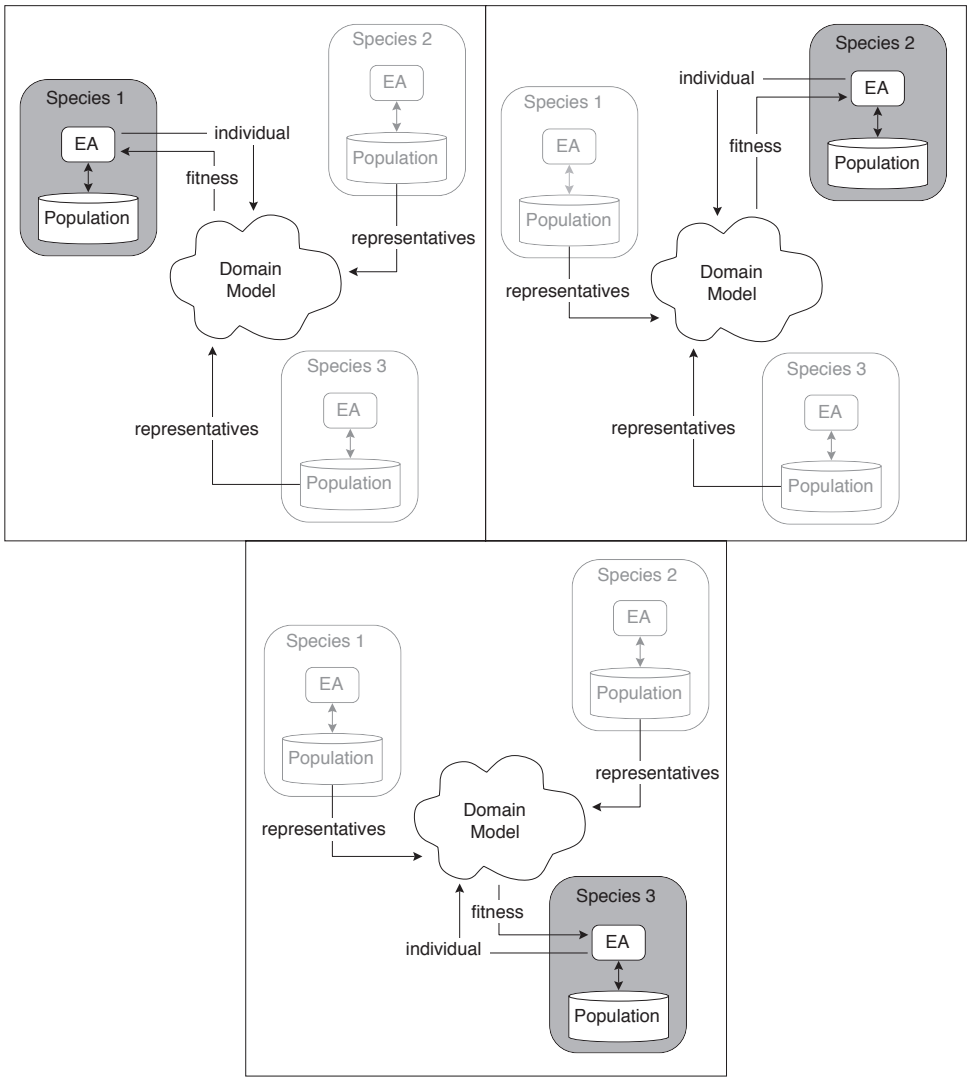


Figure 17. Potter and De Jong's model for cooperative coevolution [211]. Reprinted by permission of MIT Press Journals. The figure illustrates a problem with three variables split into three independent populations. Each population takes a turn to select representatives that form individuals with the help of other populations.

Another recent advance in coevolution is proposed by ESA's advanced concepts team which allows parallel programming on multi-core processors. Their approach is named the generalized island-model (GIM) paradigm [214]. GIM enables evolutionary and non-heuristic optimization algorithms to co-operate with each other to solve constrained/un-constrained, single/multiple-objective global optimization problems. These algorithms can form various topologies and asynchronously exchange information to accelerate their convergence properties. Reference [215] illustrates an example where an archipelago of seven islands is created where each island represents an optimization algorithm. In this case, three differential evolution and three simulated annealing instances are placed on an outer ring while subplex, a local optimization algorithm [216], is in the center of a wheel rim topology as shown in Fig. 18. Each island is executed on a separate thread; thus, each optimization algorithm can run on its own processor. A python/C++ implementation of GIM called PyGMO is made available at [217]. PyGMO gives the user the ability to combine various algorithms depending on the problem and processing power. I believe such implementations, although hard to implement, are the future of global optimization.

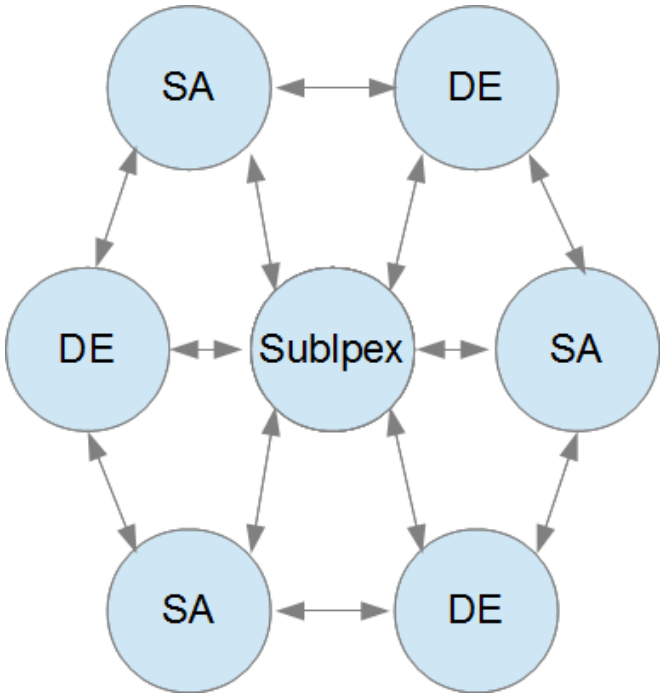


Figure 18. An archipelago of seven islands connected with wheel rim topology as discussed in [215]. Each island represents a solver: differential evolution, simulated annealing or subplex. The islands are fully and bi-directionally connected.

APPENDICES

APPENDIX A

PROOFS

A.1 Quasi-Opposition vs. Opposite

Theorem 2.2.1 *Assume that the solution of an optimization problem is uniformly distributed in a one-dimensional search space. Then the probability averaged over all x and all $\hat{x}$ that a quasi-opposite point is closer than the opposite of an EA individual to the solution is $^{11}/_{16}$.*

Proof. Given the scenario in Fig. 3 where a and b are the end points of the solution domain and c is the center of this domain, the solution x is located in one of these four sections: (A) $x \in [a, \hat{x}]$, (B) $x \in [\hat{x}, c]$, (C) $x \in [c, \hat{x}_o]$ or (D) $x \in [\hat{x}_o, b]$. We examine each scenario separately in Cases A, B, C and D below.

Case (A)

$x \in [a, \hat{x}]$ as illustrated in Fig. 19. From Fig. 19, we note that $\hat{x}_{qo}$ is always closer than $\hat{x}_o$ to solution, x. Hence,

$$\Pr[\ |\ \hat{x}_{qo} - x\ |\ <\ |\ \hat{x}_o - x\ |\ ] = 1 \text{ for } x \in [a, \hat{x}] \tag{A.1}$$

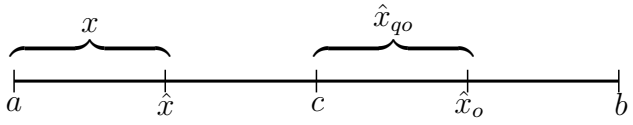


Figure 19. Solution domain if $x \in [a, \hat{x}]$

Case (B)

$x \in [\hat{x}, c]$ as illustrated in Fig. 20. x is still always closer than $\hat{x}_o$ to $\hat{x}_{qo}$. Hence,

$$\Pr[\ |\ \hat{x}_{qo} - x\ |\ <\ |\ \hat{x}_o - x\ |\ ] = 1 \text{ for } x \in [\hat{x}, c] \tag{A.2}$$

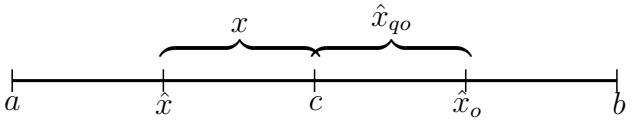


Figure 20. Solution domain if $x \in [\hat{x}, c]$

Case (C)

$x \in [c, \hat{x}_o]$ as illustrated in Fig. 21. From Fig. 21, we see that $\hat{x}_o$ is always greater than x, hence we can remove the absolute value in $|\ \hat{x}_o - x\ |$:

$$\Pr[\ |\ \hat{x}_{qo} - x\ |\ <\ |\ \hat{x}_o - x\ |\ ] \ = \ \Pr[\ |\ \hat{x}_{qo} - x\ |\ < \hat{x}_o - x] \text{ for } x \in [c, \hat{x}_o] \tag{A.3}$$

We can now employ the total probability theorem from [164] (Eq. 2-41) to rewrite Eq. A.3 as:

$$\begin{aligned}
\Pr[\ |\ \hat{x}_{qo} - x\ |\ < \hat{x}_o - x] \ = \ & \Pr[\ |\ \hat{x}_{qo} - x\ |\ < \hat{x}_o - x \mid \hat{x}_{qo} - x < 0] \times \\
& \Pr[\hat{x}_{qo} - x < 0] + \\
& \Pr[\ |\ \hat{x}_{qo} - x\ |\ < \hat{x}_o - x \mid \hat{x}_{qo} - x > 0] \times \\
& \Pr[\hat{x}_{qo} - x > 0]
\end{aligned} \tag{A.4}$$

Eliminating the remaining absolute values in $\mid \hat{x}_{qo} - x \mid$ in (A.4) and combining similar terms yields:

$$\begin{aligned} \Pr[\mid \hat{x}_{qo} - x \mid \; < \hat{x}_o - x] \;\; &= \;\; \Pr[\hat{x}_{qo} > 2x - \hat{x}_o \mid \hat{x}_{qo} < x] \Pr[\hat{x}_{qo} < x] + \\ &\quad \Pr[\hat{x}_{qo} < \hat{x}_o \mid \hat{x}_{qo} > x] \Pr[\hat{x}_{qo} > x] \end{aligned} \tag{A.5}$$

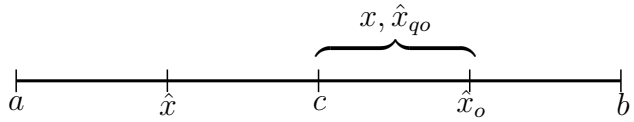


Figure 21. Solution domain if $x \in [c, \hat{x}_o]$

We solve Eq. (A.5) in three parts:

1. If we assume that x and $\hat{x}_{qo}$ have uniform distribution in $[c, \hat{x}_o]$, then $\Pr[\hat{x}_{qo} < x] = \Pr[\hat{x}_{qo} > x] = \frac{1}{2}$.

2. From Fig. 21, note that $\Pr[\hat{x}_{qo} < \hat{x}_o \mid \hat{x}_{qo} > x] = 1$.

3. We can solve the first of the two expressions on the right side of Equation (A.5) as

$$\begin{aligned} \Pr[\hat{x}_{qo} > 2x - \hat{x}_o \mid \hat{x}_{qo} < x] \Pr[\hat{x}_{qo} < x] \;\; &= \;\; \frac{\Pr[\hat{x}_{qo} > 2x - \hat{x}_o, \, \hat{x}_{qo} < x]}{\Pr[\hat{x}_{qo} < x]} \Pr[\hat{x}_{qo} < x] \\ &= \;\; \frac{\Pr[2x - \hat{x}_o < \hat{x}_{qo} < x]}{\Pr[\hat{x}_{qo} < x]} \Pr[\hat{x}_{qo} < x] \\ &= \;\; \Pr[2x - \hat{x}_o < \hat{x}_{qo} < x] \end{aligned} \tag{A.6}$$

The probability region for this inequality is shown in Fig. 22.

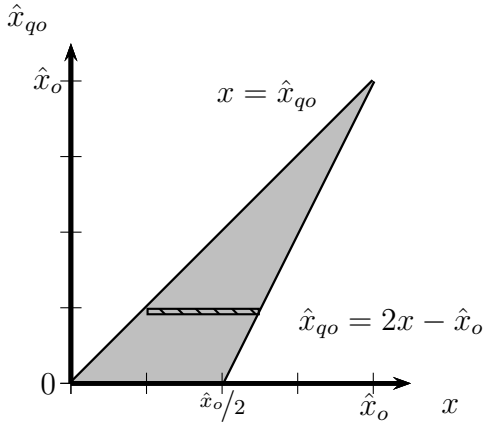


Figure 22. Integration region of $2x - \hat{x}_o < \hat{x}_{qo} < x$

Suppose that the center of the domain is 0; then based on Fig. 22, Eq. (A.6) can be solved as

$$\begin{aligned}
\Pr\left[2x - \hat{x}_o < \hat{x}_{qo} < x\right] &= \iint \mathrm{f}(x, \hat{x}_{qo})\, \mathrm{d}x\, \mathrm{d}\hat{x}_{qo} \\
&= \int_0^{\hat{x}_o} \int_{\hat{x}_{qo}}^{\frac{\hat{x}_{qo}+\hat{x}_o}{2}} \mathrm{f}(x)\mathrm{f}(\hat{x}_{qo})\, \mathrm{d}x\, \mathrm{d}\hat{x}_{qo} \\
&= \int_0^{\hat{x}_o} \int_{\hat{x}_{qo}}^{\frac{\hat{x}_{qo}+\hat{x}_o}{2}} \frac{1}{\hat{x}_o^2}\, \mathrm{d}x\, \mathrm{d}\hat{x}_{qo} \\
&= \frac{1}{2\hat{x}_o^2} \int_0^{\hat{x}_o} \hat{x}_o - \hat{x}_{qo}\, \mathrm{d}\hat{x}_{qo} \\
&= \frac{1}{2\hat{x}_o^2} \left[\hat{x}_o^2 - \frac{\hat{x}_o^2}{2}\right] \\
&= \frac{1}{4} \qquad \text{(A.7)}
\end{aligned}$$

where $\mathrm{f}(x, \hat{x}_{qo})$ is the joint density function of x and $\hat{x}_{qo}$. We can now solve Equation (A.3):

$$\Pr\left[\, |\, \hat{x}_{qo} - x\, |\, <\, |\, \hat{x}_o - x\, |\, \right] = \frac{1}{4} + (1)\left(\frac{1}{2}\right) = \frac{3}{4} \text{ for } x \in [c, \hat{x}_o] \qquad \text{(A.8)}$$

Case (D)

$x \in [\hat{x}_o, b]$ as illustrated in Fig. 23. From Fig. 23, we see that $\hat{x}_o$ is always closer than $\hat{x}_{qo}$ to x. Hence,

$$\Pr\left[\, |\, \hat{x}_{qo} - x\, |\, <\, |\, \hat{x}_o - x\, |\, \right] = 0 \text{ for } x \in [\hat{x}_o, b] \qquad \text{(A.9)}$$

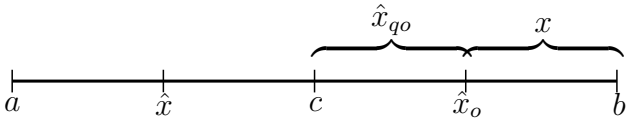


Figure 23. Solution domain if $x \in [\hat{x}_o, b]$

Conditional Probability of Quasi-Opposition vs. Opposite

Equations (A.1), (A.2), (A.8), and (A.9) can be combined to calculate the conditional probability of the quasi-opposition point being closer than the opposite point to the solution in the domain $[a, b]$:

$$\begin{aligned}\Pr\left[\, |\, \hat{x}_{qo} - x\, |\, <\, |\, \hat{x}_o - x\, |\ \ \middle|\ \hat{x}_o\right] &= \frac{1(\hat{x} - a) + 1(c - \hat{x}) + \frac{3}{4}(\hat{x}_o - c) + 0(b - \hat{x}_o)}{b - a} \\ &= \frac{\frac{1}{4}c - a + \frac{3}{4}\hat{x}_o}{b - a} \end{aligned} \tag{A.10}$$

Since $c = (a + b)/2$, we can rewrite Eq. (A.10) as

$$\Pr\left[\, |\, \hat{x}_{qo} - x\, |\, <\, |\, \hat{x}_o - x\, |\ \ \middle|\ \hat{x}_o\right] = \frac{\frac{1}{8}(a + b) - a + \frac{3}{4}\hat{x}_o}{b - a} \tag{A.11}$$

Assuming that the domain is symmetric (that is $b = -a$), Eq. (A.11) becomes

$$\Pr\left[\, |\, \hat{x}_{qo} - x\, |\, <\, |\, \hat{x}_o - x\, |\ \ \middle|\ \hat{x}_o\right] = \frac{b + \frac{3}{4}\hat{x}_o}{2b} \text{ for } x \in [a, b] \tag{A.12}$$

Probability of Quasi-Opposition vs. Opposite

We now take the previous results to prove Theorem 2.2.1. Let $\hat{x}_o$ have a uniform distribution; we can then calculate the probability as

$$\Pr\left[\, |\, \hat{x}_{qo} - x\, |\, <\, |\, \hat{x}_o - x\, |\,\right] = \int_{-b}^{b} \frac{b + \frac{3}{4}\hat{x}_o}{2b} \mathrm{f}(\hat{x}_o)\, \mathrm{d}\hat{x}_o \tag{A.13}$$

Since $\hat{x}_o$ is uniformly distributed between 0 and b, that is, $\hat{x}_o \sim \mathrm{U}[0, b]$,

Eq. (A.13) becomes

$$\begin{aligned}\Pr[\,|\,\hat{x}_{qo}-x\,|\;<\;|\,\hat{x}_o-x\,|\,] &= \frac{1}{b}\int_0^b \frac{b+\frac{3}{4}\hat{x}_o}{2b}\,\mathbf{d}\hat{x}_o \\ &= \frac{\frac{3}{16}\hat{x}_o(\hat{x}_o+\frac{16}{6}b)}{b^2}\Bigg|_0^b \\ &= \frac{11}{16}\end{aligned} \tag{A.14}$$

This gives the result stated in Theorem 2.2.1 □

A.2 Quasi-Reflection vs. Opposite

Theorem 2.2.2 *Assume that the solution of an optimization problem is uniformly distributed in a one-dimensional search space. Then the probability averaged over all x and all $\hat{x}$ that a quasi-reflected point is closer than the opposite of an EA individual to the solution is $9/16$.*

Proof. We compute the probability of $\hat{x}_{qr}$ being closer than $\hat{x}_o$ to the solution, x, and the expected value of this probability under certain conditions.

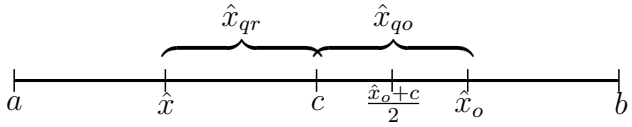


Figure 24. Opposite points defined in domain $[a,b]$. c is the center of the domain and $\hat{x}$ is an EA individual, generated by an EA. $\hat{x}_o$ is the opposite of $\hat{x}$, and $\hat{x}_{qo}$ and $\hat{x}_{qr}$ are the quasi-opposite and quasi-reflected points, respectively.

Given the scenario in Fig. 24 where a and b are the end points of the solution domain and c is the center of this domain, the solution x is in one of these five sections: (A) $x \in [a,\hat{x}]$, (B) $x \in [\hat{x},c]$, (C) $x \in [c,\frac{\hat{x}_o+c}{2}]$, (D) $x \in [\frac{\hat{x}_o+c}{2},\hat{x}_o]$ or (E) $x \in [\hat{x}_o,b]$. We examine each scenario separately in Cases A, B, C, D and E below.

Case (A)

$x \in [a, \hat{x}]$ as illustrated in Fig. 25. From Fig. 25, we note that $\hat{x}_{qr}$ is always closer than $\hat{x}_o$ to solution, x. Hence,

$$\Pr[\ |\ \hat{x}_{qr} - x\ |\ <\ |\ \hat{x}_o - x\ |\ ] = 1 \text{ for } x \in [a, \hat{x}] \tag{A.15}$$

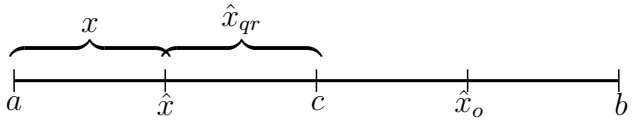


Figure 25. Solution domain if $x \in [a, \hat{x}]$

Case (B)

$x \in [\hat{x}, c]$ as illustrated in Fig. 26. x is always closer to $\hat{x}_{qr}$ than $\hat{x}_o$. Hence,

$$\Pr[\ |\ \hat{x}_{qr} - x\ |\ <\ |\ \hat{x}_o - x\ |\ ] = 1 \text{ for } x \in [\hat{x}, c] \tag{A.16}$$

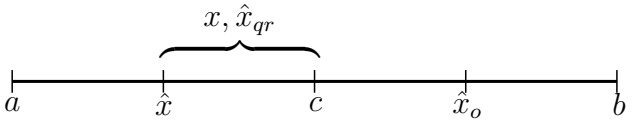


Figure 26. Solution domain if $x \in [\hat{x}, c]$

Case (C)

$x \in [c, \hat{x}_o]$ as illustrated in Fig. 27. We eliminate absolute value signs, knowing that $\hat{x}_{qr} < x < \hat{x}_o$:

$$\Pr[\ |\ \hat{x}_{qr} - x\ |\ <\ |\ \hat{x}_o - x\ |\ ] = \Pr[2x - \hat{x}_o < \hat{x}_{qr}] \text{ for } x \in [c, \hat{x}_o] \tag{A.17}$$

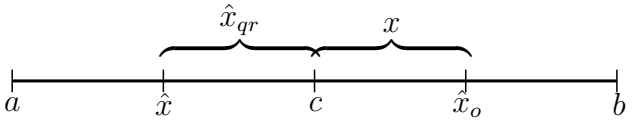


Figure 27. Solution domain if $x \in [c, \hat{x}_o]$

The probability region for this inequality is shown in Fig. 28.

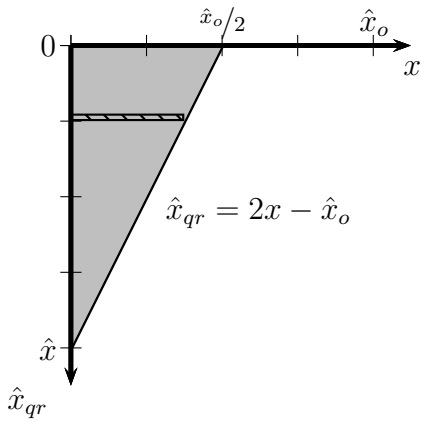


Figure 28. Integration region of $2x - \hat{x}_o < \hat{x}_{qr}$

Once again, we suppose that the end points of the solution domain are equal in magnitude, or $b = -a$. Thus, $\hat{x}_o = -\hat{x}$. We can solve Eq. (A.17) based on Fig. 28:

$$\begin{aligned}
\Pr\left[2x - \hat{x}_o < \hat{x}_{qr}\right] &= \iint \mathrm{f}(x, \hat{x}_{qr})\, \mathrm{d}x\, \mathrm{d}\hat{x}_{qr} \\
&= \int_{-\hat{x}_o}^{0} \int_{0}^{\frac{\hat{x}_{qr}+\hat{x}_o}{2}} \mathrm{f}(x)\mathrm{f}(\hat{x}_{qr})\, \mathrm{d}x\, \mathrm{d}\hat{x}_{qr} \\
&= \int_{-\hat{x}_o}^{0} \int_{0}^{\frac{\hat{x}_{qr}+\hat{x}_o}{2}} \frac{1}{\hat{x}_o^2}\, \mathrm{d}x\, \mathrm{d}\hat{x}_{qr} \\
&= \frac{1}{\hat{x}_o^2} \int_{-\hat{x}_o}^{0} \frac{\hat{x}_{qr} + \hat{x}_o}{2}\, \mathrm{d}x\, \mathrm{d}\hat{x}_{qr} \\
&= \frac{1}{2\hat{x}_o^2} \left[\hat{x}_o^2 - \frac{\hat{x}_o^2}{2}\right] \\
&= \frac{1}{4}
\end{aligned} \tag{A.18}$$

Case (D)

$x \in [\hat{x}_o, b]$ as illustrated in Fig. 29. From Fig. 29, we see that $\hat{x}_o$ is always closer than $\hat{x}_{qr}$ to x. Hence,

$$\Pr\left[\, |\, \hat{x}_{qr} - x\, |\, <\, |\, \hat{x}_o - x\, |\, \right] = 0 \text{ for } x \in [\hat{x}_o, b] \tag{A.19}$$

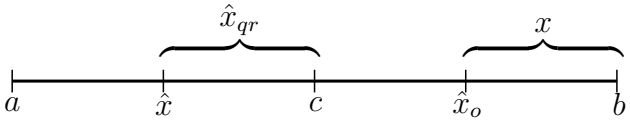


Figure 29. $\hat{x}_{qr}$ solution domain, $x \in [\hat{x}_o, b]$

Conditional Probability of Quasi-Reflection vs. Opposite

Equations (A.15), (A.16), (A.18) and (A.19) can be combined to calculate the probability of the quasi-opposition point being closer than the opposite point to the solution in the domain $[a, b]$:

$$\Pr\left[\, |\, \hat{x}_{qr} - x\, |\, <\, |\, \hat{x}_o - x\, |\, \,\middle|\, \hat{x}_o\right] = \frac{1(\hat{x} - a) + 1(c - \hat{x}) + \frac{1}{4}(\hat{x}_o - c) + 0(b - \hat{x}_o)}{b - a}$$

$$= \frac{b + \frac{1}{4}\hat{x}_o}{2b} \text{ for } x \in [a, b] \tag{A.20}$$

Probability of Quasi-Reflection vs. Opposite

We now take the previous results to prove Theorem 2.2.2. Let $\hat{x}_o$ have a uniform distribution; we can then calculate the probability as

$$\Pr[\, |\, \hat{x}_{qr} - x\, |\, <\, |\, \hat{x}_o - x\, |\,] = \frac{1}{b}\int_{-b}^{b} \frac{b + \frac{1}{4}\hat{x}_o}{2b} \mathrm{f}(\hat{x}_o)\, \mathrm{d}\hat{x}_o \tag{A.21}$$

and since $\hat{x}_o \in [0, b]$, Eq. (A.21) becomes

$$\begin{aligned} \Pr[\, |\, \hat{x}_{qo} - x\, |\, <\, |\, \hat{x}_o - x\, |\,] &= \frac{1}{b}\int_{0}^{b} \frac{b + \frac{1}{4}\hat{x}_o}{2b}\, \mathrm{d}\hat{x}_o \\ &= \frac{9}{16} \text{ for } x \in [\hat{x}_o, b] \end{aligned} \tag{A.22}$$

This gives the result stated in Theorem 2.2.2 □

A.3 Quasi-Opposition vs. EA Individual

Theorem 2.2.3 *Assume that the solution of an optimization problem is uniformly distributed in a one-dimensional search space. Then the probability av-*

eraged over all x and all $\hat{x}$ that a quasi-opposite point is closer than an EA individual to the solution is $9/16$.

Proof. Given the scenario in Fig. 3 where a and b are the end points of the solution domain and c is the center of this domain, there are four possibilities for the solution, x: (A) $x \in [a, \hat{x}]$, (B) $x \in [\hat{x}, c]$, (C) $x \in [c, \hat{x}_o]$ or (D) $x \in [\hat{x}_o, b]$. We examine each scenario separately in Cases A, B, C and D below.

Case (A)

$x \in [a, \hat{x}]$ as illustrated in Fig. 30. From Fig. 30, we note that $\hat{x}$ is always closer than $\hat{x}_{qo}$ to solution, x. Hence, when $x \in [a, \hat{x}]$, the probability that the quasi-opposition point is closer than the opposite point to the solution is

$$\Pr[\ |\ \hat{x}_{qo} - x\ |\ <\ |\ \hat{x} - x\ |\ ] = 0 \text{ for } x \in [a, \hat{x}] \tag{A.23}$$

Also, note that this case is a reflection of Case D from Section A.2.

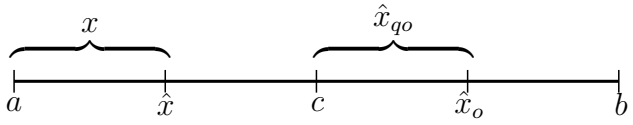


Figure 30. Solution domain if $x \in [a, \hat{x}]$

Case (B)

$x \in [\hat{x}, c]$ as illustrated in Fig. 31. From Fig. 31, we note that $\hat{x} < x < \hat{x}_{qo}$. We then eliminate the absolute value signs:

$$\Pr[\ |\ \hat{x}_{qo} - x\ |\ <\ |\ \hat{x} - x\ |\ ] = \Pr[\hat{x}_{qo} < 2x - \hat{x}_o] \text{ for } x \in [\hat{x}, c] \tag{A.24}$$

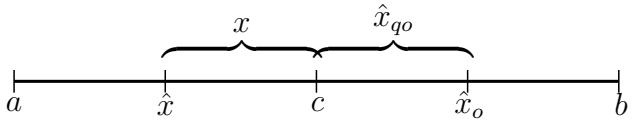


Figure 31. Solution domain if $x \in [\hat{x}, \frac{\hat{x}+c}{2}]$

The probability region for this inequality is shown in Fig. 32.

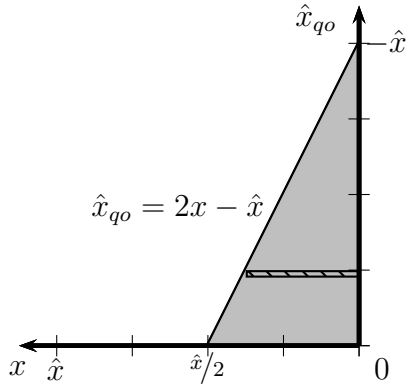


Figure 32. Integration region of $\hat{x}_{qo} < 2x - \hat{x}$

Assuming that center of the solution domain c is 0 and $\hat{x}_{qo}$ and x are uniformly distributed, we can solve Eq. A.24 using Fig. 32:

$$\begin{aligned}
\Pr\left[\hat{x}_{qo} < 2x - \hat{x}\right] &= \iint \mathrm{f}(x, \hat{x}_{qo})\, \mathrm{d}x\, \mathrm{d}\hat{x}_{qo} \\
&= \int_0^{-\hat{x}} \int_{\frac{\hat{x}_{qo}+\hat{x}}{2}}^{0} \mathrm{f}(x)\mathrm{f}(\hat{x}_{qo})\, \mathrm{d}x\, \mathrm{d}\hat{x}_{qo} \\
&= \int_0^{-\hat{x}} \int_{\frac{\hat{x}_{qo}+\hat{x}}{2}}^{0} \frac{1}{\hat{x}^2}\, \mathrm{d}x\, \mathrm{d}\hat{x}_{qo} \\
&= \frac{-1}{\hat{x}^2} \int_{-\hat{x}_o}^{0} \frac{\hat{x}_{qo} + \hat{x}}{2}\, \mathrm{d}\hat{x}_{qo} \\
&= \frac{-1}{2\hat{x}^2} \left[\frac{\hat{x}^2}{2} - \hat{x}^2\right] \\
&= \frac{1}{4}
\end{aligned} \tag{A.25}$$

Also, note that this case is similar to Case C of Section A.2.

Cases (C) and (D)

If we look back at Section A.2, we note that $\hat{x}_{qr}$ versus $\hat{x}_o$ is the mirrored version of $\hat{x}_{qo}$ versus $\hat{x}$. Thus, Case (A) of Section A.3 is equivalent to Case (D) of Section A.2. Table XXXIV summarizes the results of these findings.

Solution region if $[\,\lvert\,\hat{x}_{qo}-x\,\rvert\,<\,\lvert\,\hat{x}-x\,\rvert\,]$	Solution region if $[\,\lvert\,\hat{x}_{qr}-x\,\rvert\,<\,\lvert\,\hat{x}_{o}-x\,\rvert\,]$	Pr
$x \in [a, \hat{x}]$	$x \in [\hat{x}_o, b]$	0
$x \in [\hat{x}, c]$	$x \in [c, \hat{x}_o]$	$1/4$
$x \in [c, \hat{x}_o]$	$x \in [\hat{x}, c]$	1
$x \in [\hat{x}_o, b]$	$x \in [a, \hat{x}]$	1

Table XXXIV. Similar probabilities of different opposite points: $\hat{x}_{qo}$ vs. $\hat{x}$ and $\hat{x}_{qr}$ vs. $\hat{x}_{qo}$

Conditional Probability of Quasi-Opposition vs. EA Individual

We will now use the the probabilities derived in Equations (A.15), (A.16), (A.18) and (A.19) to calculate the probability of the quasi-opposition point being closer than the EA individual to the solution in the domain $[a, b]$.

$$\begin{aligned}\Pr\left[\,\lvert\,\hat{x}_{qo}-x\,\rvert\,<\,\lvert\,\hat{x}-x\,\rvert\;\middle|\;\hat{x}\right] &= \frac{0(\hat{x}-a)+\frac{1}{4}\,(c-\hat{x})+1\,(\hat{x}_o-c)+1(b-\hat{x}_o)}{b-a} \\ &= \frac{4b-3c-\hat{x}}{4(b-a)} \\ &= \frac{b-\frac{1}{4}\hat{x}}{2b} \text{ for } x \in [a,b] \end{aligned} \tag{A.26}$$

Probability of Quasi-Opposition vs. EA Individual

We now take the previous results to prove Theorem 2.2.3. Let $\hat{x}$ have a uniform distribution; we can then calculate the probability as

$$\Pr[\,\lvert\,\hat{x}_{qo}-x\,\rvert\,<\,\lvert\,\hat{x}-x\,\rvert\,] = \int_{-b}^{b} \frac{b-\frac{1}{4}\hat{x}}{2b}\mathrm{f}(\hat{x})\,\mathrm{d}\hat{x} \tag{A.27}$$

and since $\hat{x} \in [-b, 0]$, Eq. (A.27) becomes

$$\Pr[\ |\ \hat{x}_{qo} - x\ |\ <\ |\ \hat{x} - x\ |\ ] = \frac{1}{b}\int_{-b}^{0} \frac{b - \frac{1}{4}\hat{x}}{2b}\, \mathrm{d}\hat{x} = \frac{\hat{x}(b - \frac{1}{8}\hat{x})}{2b^2}\ \Bigg|_{-b}^{0}$$

$$= \frac{9}{16} \tag{A.28}$$

This gives the result stated in Theorem 2.2.3 □

A.4 Quasi-Reflection vs. EA Individual

Theorem 2.2.4 *Assume that the solution of an optimization problem is uniformly distributed in a one-dimensional search space. Then the probability averaged over all x and all $\hat{x}$ that a quasi-reflected point is closer than an EA individual to the solution is $^{11}/_{16}$.*

Proof. Given the scenario in Fig. 3 where a and b are the end points of the solution domain and c is the center of this domain, there are four possibilities for the solution, x: (A) $x \in [a, \hat{x}]$, (B) $x \in [\hat{x}, c]$, (C) $x \in [c, \hat{x}_o]$ or (D) $x \in [\hat{x}_o, b]$. We examine each scenario separately in Cases A, B, C and D below.

Case (A)

$x \in [a, \hat{x}]$ as illustrated in Fig. 33.

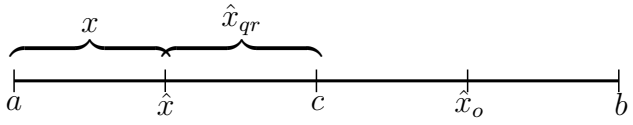


Figure 33. Solution domain if $x \in [a, \hat{x}]$

From Fig. 33, we note that $\hat{x}$ is always closer than $\hat{x}_{qr}$ to solution, x. Hence, when $x \in [a, \hat{x}]$, the probability that the quasi-reflected point is closer than the opposite point to the solution is

$$\Pr[\ |\ \hat{x}_{qr} - x\ |\ <\ |\ \hat{x} - x\ |\ ] = 0 \text{ for } x \in [a, \hat{x}] \tag{A.29}$$

Also, note that this case is similar to Case D of Section A.1.

Case (B)

$x \in [\hat{x}, c]$ as illustrated in Fig. 34.

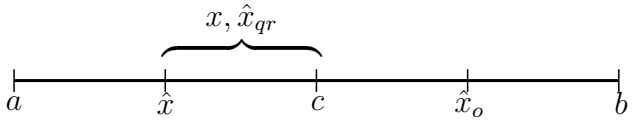


Figure 34. Solution domain if $x \in [\hat{x}, c]$

From Fig. 34, we note that this case is a reflection of Case C of Section A.1 shown in Fig. 21. This results in

$$\Pr[\,|\,\hat{x}_{qr} - x\,|\,<\,|\,\hat{x} - x\,|\,] = 3/4 \text{ for } x \in [\hat{x}, c] \tag{A.30}$$

Cases (C) and (D)

If we look carefully at Section A.1, we realize that $\hat{x}_{qr}$ versus $\hat{x}$ is the mirrored version of our analysis in Section A.1. Thus, Case (A) of Session A.4 is equivalent to Case(D) of Section A.1. Table XXXV summarizes the results of these findings.

Solution region if $[\,\|\,\hat{x}_{qr} - x\,\|\,<\,\|\,\hat{x} - x\,\|\,]$	Solution region if $[\,\|\,\hat{x}_{qo} - x\,\|\,<\,\|\,\hat{x}_o - x\,\|\,]$	Pr
$x \in [a, \hat{x}]$	$x \in [\hat{x}_o, b]$	0
$x \in [\hat{x}, c]$	$x \in [c, \hat{x}_o]$	$3/4$
$x \in [c, \hat{x}_o]$	$x \in [\hat{x}, c]$	1
$x \in [\hat{x}_o, b]$	$x \in [a, \hat{x}]$	1

Table XXXV. Similar probabilities of different opposite points: $\hat{x}_{qr}$ vs. $\hat{x}$ and $\hat{x}_{qo}$ vs. $\hat{x}_o$

Conditional Probability of Quasi-Reflection vs. EA Individual

Equations (A.1), (A.2), (A.8) and (A.9) can be combined to calculate the probability of the quasi-reflected point being closer than the EA individual to the solution in the domain $[a, b]$:

$$\begin{aligned}\Pr\left[\, |\, \hat{x}_{qr} - x\, |\, <\, |\, \hat{x} - x\, |\, \;\middle|\; \hat{x}\right] &= \frac{0(\hat{x} - a) + \frac{3}{4}(c - \hat{x}) + 1\,(\hat{x}_o - c) + 1(b - \hat{x}_o)}{b - a} \\ &= \frac{4b - c - 3\hat{x}}{4(b - a)} \\ &= \frac{b - \frac{3}{4}\hat{x}}{2b} \end{aligned} \tag{A.31}$$

Probability of Quasi-Reflection vs. EA Individual

We now take the previous results to prove Theorem 2.2.4. Let $\hat{x}$ have a uniform distribution; we can calculate the probability as

$$\Pr[\, |\, \hat{x}_{qr} - x\, |\, <\, |\, \hat{x} - x\, |\,] = \int_{-b}^{b} \frac{b - \frac{3}{4}\hat{x}}{2b} \mathrm{f}(\hat{x})\, \mathrm{d}\hat{x} \tag{A.32}$$

Since $\hat{x} \in [-b, 0]$, Eq. (A.32) becomes

$$\begin{aligned}\Pr[\, |\, \hat{x}_{qr} - x\, |\, <\, |\, \hat{x} - x\, |\,] &= \frac{1}{b}\int_{-b}^{0} \frac{b - \frac{1}{2}\hat{x}}{2b}\, \mathrm{d}\hat{x} = \left.\frac{\hat{x}(8b - 3\hat{x})}{16b^2}\right|_{-b}^{0} \\ &= \frac{11}{16} \end{aligned} \tag{A.33}$$

This gives the result stated in Theorem 2.2.4 □

A.5 Probabilistic Analysis of Fitness-Weighted Quasi-Reflection

Theorem 2.3.1 *Assume that the solution of an optimization problem is uniformly distributed in a one-dimensional search space. Then the probability averaged over all x and all $\hat{x}$ that a quasi-reflected point (as a function of the*

reflection weight, K) is closer than an EA individual to the solution is $^{(6-K)}/_{8}$ *when* $K \in [0, 1]$.

Proof. Given the scenario in Fig. 35 where a and b are the end points of the solution domain and c is the center of this domain, the solution, x, will always be in one of these four segments: (A) $x \in [a, \hat{x}]$, (B) $x \in [\hat{x}, c]$, (C) $x \in [c, \hat{x}_o]$ or (D) $x \in [\hat{x}_o, b]$. We examine each scenario separately in Cases A, B, C and D below.

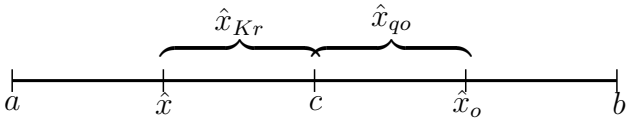


Figure 35. Opposite points defined in domain $[a, b]$. c is the center of the domain and $\hat{x}$ is an EA individual. $\hat{x}_o$ is the opposite of $\hat{x}$, and $\hat{x}_{qo}$ and $\hat{x}_{Kr}$ are the quasi-opposite and quasi-reflected points, respectively.

A.5.1 Case (A)

For this case, $x \in [a, \hat{x}]$ as shown in Fig. 36. From Fig. 36, we note that $\hat{x}$ is always closer than $\hat{x}_{Kr}$ to solution, x. Hence, when $x \in [a, \hat{x}]$, the probability that the quasi-reflected point is closer than the opposite point to the solution is

$$\Pr[\ |\ \hat{x}_{Kr} - x\ |\ <\ |\ \hat{x} - x\ |\ ] = 0 \text{ for } x \in [a, \hat{x}] \tag{A.34}$$

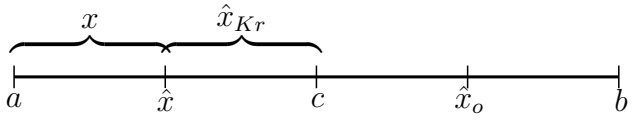


Figure 36. Solution domain if $x \in [a, \hat{x}]$

A.5.2 Case (B)

For this case we investigate the probability if $x \in [\hat{x}, c]$ as seen in Fig. 37.

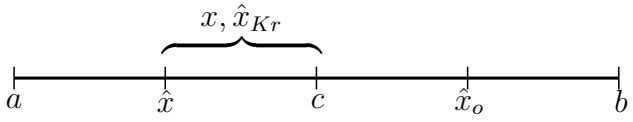


Figure 37. Solution domain if $x \in [\hat{x}, c]$

$$\Pr[\ |\ \hat{x}_{Kr} - x\ |\ <\ |\ \hat{x} - x\ |\ ] = \Pr[\ |\ \hat{x}(1-K) - x\ |\ <\ |\ \hat{x} - x\ |\ ] \tag{A.35}$$

From Fig. 37, we note that $\hat{x} < x$. Then, Eq. A.35 can be simplified as

$$\Pr[\ |\ x - \hat{x}(1-K)\ |\ < x - \hat{x}] \tag{A.36}$$

We now use the Total Probability Theorem from [164], Eq. 241, and obtain four probabilities:

$$\begin{aligned}\Pr[\ |\ x - \hat{x}(1-K)\ |\ < x - \hat{x}] \;=\; & \Pr[\ |\ \hat{x}(1-K) - x\ |\ < x - \hat{x} \mid \hat{x}(1-K) < x] \times \\ & \Pr[\hat{x}(1-K) < x] + \\ & \Pr[\ |\ \hat{x}(1-K) - x\ |\ < x - \hat{x} \mid \hat{x}(1-K) > x] \times \\ & \Pr[\hat{x}(1-K) > x] \end{aligned} \tag{A.37}$$

The subsequent sections analyzes these four terms individually.

Case (B1)

This case involves the term $\Pr[\ |\ \hat{x}(1-K) - x\ |\ < x - \hat{x} \mid \hat{x}(1-K) < x]$ from Eq. A.37

$$\begin{aligned}\Pr[|\ \hat{x} - K\hat{x} - x| < x - \hat{x} \mid \hat{x}(1-K) < x] \;&=\; \Pr[x - \hat{x} + K\hat{x} < x - \hat{x} \mid \hat{x}(1-K) < x] \\ &=\; \Pr[\hat{x}K < 0 \mid \hat{x}(1-K) < x]\end{aligned} \tag{A.38}$$

Since by definition $K > 0$ and $\hat{x} < 0$

$$\Pr[\ |\ \hat{x}(1-K) - x\ |\ < x - \hat{x} \mid \hat{x}(1-K) < x] = 1 \text{ for } x \in [\hat{x}, c] \tag{A.39}$$

Case (B2)

This case involves the term $\Pr[\hat{x}(1-K) < x]$ from Eq. A.37 and it is solved in two steps. We first hold $\hat{x}$ fixed and find the probability over x. Then, we let $\hat{x}$ vary and calculate the corresponding expected probability.

When $\hat{x}$ is fixed and x is uniform in $[\hat{x}, 0]$, that is, $x \sim \mathbf{U}[\hat{x}, 0]$, we obtain:

$$\begin{aligned}\Pr[\hat{x}(1-K) < x] &= \int_{\hat{x}(1-K)}^{\infty} \mathbf{f}(x)\,\mathbf{d}x = \int_{\hat{x}(1-K)}^{0} -\frac{1}{\hat{x}}\,\mathbf{d}x \\ &= -\frac{1}{\hat{x}}x \Big|_{\hat{x}(1-K)}^{0} \\ &= 1-K \end{aligned} \tag{A.40}$$

Now, we let $\hat{x} \sim \mathbf{U}[-b, 0]$ and calculate the expected probability:

$$\begin{aligned}\mathbf{E}[\Pr[\hat{x}(1-K) < x]] &= \int_{-b}^{0} \Pr[x > \hat{x}(1-K)]\,\mathbf{f}(\hat{x})\,\mathbf{d}\hat{x} \\ &= \int_{-b}^{0} (1-K)(\frac{1}{b})\,\mathbf{d}\hat{x} = \frac{1-K}{b}\hat{x} \Big|_{-b}^{0} \\ &= 1-K \end{aligned} \tag{A.41}$$

Case (B3)

Here we solve the term $\Pr[\,|\,\hat{x}(1-K) - x\,|\, < x - \hat{x} \mid \hat{x}(1-K) > x]$ from Eq. A.37.

$$\begin{aligned}\Pr[\,|\,\hat{x}(1-K) - x\,|\, < x - \hat{x} \mid \hat{x}(1-K) > x] &= \frac{\Pr[\hat{x}(2-K) < 2x\,,\, \hat{x}(1-K) > x]}{\Pr[\hat{x}(1-K) > x]} \\ &= \frac{\Pr\left[\frac{2-K}{2}\hat{x} < x\,,\, \hat{x}(1-K) > x\right]}{\Pr[\hat{x}(1-K) > x]} \\ &= \frac{\Pr\left[\frac{2-K}{2}\hat{x} < x < \hat{x}(1-K)\right]}{\Pr[\hat{x}(1-K) > x]} \end{aligned} \tag{A.42}$$

Eq. A.42 consists of two probabilities. The second probability is solved in Section A.5.2. The first probability will be calculated as an expected probabil-

ity. This will be done in two steps where we hold $\hat{x}$ fixed and $x \sim \mathbf{U}[\hat{x}, 0]$:

$$\begin{aligned}
\Pr\left[\frac{2-K}{2}\hat{x} < x < \hat{x}(1-K)\right] &= \int_{\frac{2-K}{2}\hat{x}}^{(1-K)\hat{x}} \mathbf{f}(x)\,\mathbf{d}x = \int_{\frac{2-K}{2}\hat{x}}^{(1-K)\hat{x}} -\frac{1}{\hat{x}}\,\mathbf{d}x \\
&= -\frac{1}{\hat{x}}x\,\Big|_{\frac{2-K}{2}\hat{x}}^{(1-K)\hat{x}} = -\frac{1}{\hat{x}}\left[\hat{x} - K\hat{x} - \hat{x} + \frac{K\hat{x}}{2}\right] \\
&= \frac{K}{2} \qquad \text{(A.43)}
\end{aligned}$$

We let $\hat{x} \sim \mathbf{U}[-b, 0]$ and calculate the expected probability:

$$\begin{aligned}
\mathbf{E}\left[\Pr\left[\frac{2-K}{2}\hat{x} < x < \hat{x}(1-K)\right]\right] &= \int_{-b}^{0} \Pr\left[\frac{2-K}{2}\hat{x} < x < \hat{x}(1-K)\right] f(\hat{x})\,\mathbf{d}\hat{x} \\
&= \int_{-b}^{0} \frac{K}{2}\,\frac{1}{b}\,\mathbf{d}\hat{x} = \frac{K}{2b}\hat{x}\,\Big|_{-b,0} \\
&= \frac{K}{2} \qquad \text{(A.44)}
\end{aligned}$$

We then combine Eq. A.44 and Eq. A.47 to solve Eq. A.42:

$$\begin{aligned}
\Pr[\mid \hat{x}(1-K) - x \mid \; < x - \hat{x} \mid \hat{x}(1-K) > x] &= \frac{\Pr\left[\frac{2-K}{2}\hat{x} < x < \hat{x}(1-K)\right]}{\Pr\left[\hat{x}(1-K) > x\right]} \\
&= \frac{K}{2}\,\frac{1}{K} = \frac{1}{2} \qquad \text{(A.45)}
\end{aligned}$$

Case (B4)

This case solves the term $\Pr[\hat{x}(1-K) > x]$ from Eq. A.37. This is solved in two steps. We first hold $\hat{x}$ fixed and find the probability over x. Then, we let $\hat{x}$ vary and calculate the corresponding expected probability.

When $\hat{x}$ is fixed and $x \sim \mathbf{U}[\hat{x}, 0]$, we obtain:

$$\begin{aligned}
\Pr[\hat{x}(1-K) > x] &= \Pr[x < \hat{x}(1-K)] = \mathbf{F}_x(\hat{x}(1-K)) \\
&= \int_{-\infty}^{\hat{x}(1-K)} \mathbf{f}(x)\,\mathbf{d}x = \int_{\hat{x}}^{\hat{x}(1-K)} -\frac{1}{\hat{x}}\,\mathbf{d}x \\
&= -\frac{1}{\hat{x}}x\,\Big|_{\hat{x}}^{\hat{x}(1-K)} = -\frac{1}{\hat{x}}\left[\hat{x} - K\hat{x} - \hat{x}\right] \\
&= K \qquad \text{(A.46)}
\end{aligned}$$

Now, we let $\hat{x} \sim \mathbf{U}[-b, 0]$ and calculate the expected probability:

$$\begin{aligned}
\mathbf{E}\left[\Pr\left[\hat{x}(1-K) > x\right]\right] &= \int_{-b}^{0} \Pr\left[x < \hat{x}(1-K)\right] \mathbf{f}(\hat{x})\, \mathbf{d}\hat{x} \\
&= \int_{-b}^{0} K \frac{1}{b}\, \mathbf{d}\hat{x} = \frac{K}{b}\, \hat{x} \Big|_{-b}^{0} = \frac{K}{b}\, b \\
&= K
\end{aligned} \tag{A.47}$$

Case (B) Conclusion

We can now solve Eq. A.35 and Eq. A.37 using Eq. A.39, Eq. A.41, Eq. A.45 and Eq. A.47:

$$\begin{aligned}
\Pr[\,|\,\hat{x}_{Kr} - x\,|\,<\,|\,\hat{x} - x\,|\,] &= \Pr[\,|\,\hat{x}(1-K) - x\,|\,<\,|\,\hat{x} - x\,|\,] \\
&= \Pr[\,|\,\hat{x}(1-K) - x\,|\,< x - \hat{x} \mid \hat{x}(1-K) < x] \times \\
&\quad \Pr[\hat{x}(1-K) < x] + \\
&\quad \Pr[\,|\,\hat{x}(1-K) - x\,|\,< x - \hat{x} \mid \hat{x}(1-K) > x] \times \\
&\quad \Pr[\hat{x}(1-K) > x] \\
&= 1(1-K) + \frac{1}{2}(K) \\
&= 1 - \frac{K}{2}
\end{aligned} \tag{A.48}$$

Thus,

$$\Pr[\,|\,\hat{x}_{Kr} - x\,|\,<\,|\,\hat{x} - x\,|\,] = 1 - \frac{K}{2} \text{ for } x \in [\hat{x}, c] \tag{A.49}$$

A.5.3 Case (C)

For this case $x \in [c, \hat{x}_o]$ as shown in Fig. 38.

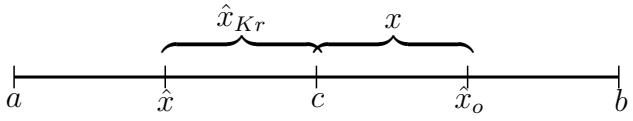


Figure 38. Solution domain if $x \in [c, \hat{x}_o]$

When $x \in [c, \hat{x}_o]$, the probability that the quasi-reflected point is closer than the estimated point to the solution is

$$\Pr[\ |\ \hat{x}_{Kr} - x\ |\ <\ |\ \hat{x} - x\ |\ ] = \Pr[\ |\ \hat{x}(1-K) - x\ |\ <\ |\ \hat{x} - x\ |\ ] \tag{A.50}$$

From Fig. 38, we note that $\hat{x} < x$ and $\hat{x}(1-K) < x$. Then, Eq. A.35 can be simplified as

$$\Pr[x - \hat{x}(1-K) < x - \hat{x}] = \Pr[\hat{x}(K+1-1) < x(1-1)] = \Pr[K\hat{x} < 0] \tag{A.51}$$

Thus,

$$\Pr[\ |\ \hat{x}_{Kr} - x\ |\ <\ |\ \hat{x} - x\ |\ ] = 1 \text{ for } x \in [c, \hat{x}_o] \tag{A.52}$$

A.5.4 Case (D)

This is the case if $x \in [\hat{x}_o, b]$ as shown in Fig. 39.

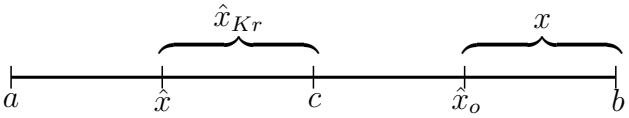


Figure 39. Solution domain if $x \in [\hat{x}_o, b]$

This case is very similar to Case (C). From Fig. 39, we again note that $\hat{x} < x$ and $\hat{x}(1-K) < x$.

$$\begin{aligned} \Pr[\ |\ \hat{x}_{Kr} - x\ |\ <\ |\ \hat{x} - x\ |\ ] &= \Pr[K\hat{x} < 0] \\ &= 1 \text{ for } x \in [\hat{x}_o, b] \end{aligned} \tag{A.53}$$

A.5.5 Conditional Probability

We can now combine all of the cases to calculate the conditional probability in the domain $[a, b]$.

$$\begin{aligned} \Pr\left[\ |\ \hat{x}_{Kr} - x\ |\ <\ |\ \hat{x} - x\ |\ \ \middle|\ \hat{x}\right] &= \frac{0(\hat{x}+b) + (1-\frac{K}{2})(0-\hat{x}) + 1(\hat{x}_o - 0) + 1(b - \hat{x}_o)}{2b} \\ &= \frac{-\hat{x}(1-\frac{K}{2}) + b}{2b} \end{aligned} \tag{A.54}$$

A.5.6 Probability

We now take the previous results to prove Theorem 2.3.1. The probability for uniform $\hat{x}$ can be calculated as

$$\begin{aligned}
\Pr[\,|\,\hat{x}_{Kr}-x\,|\,<\,|\,\hat{x}-x\,|\,] &= \int_{-b}^{0} \Pr[\,|\,\hat{x}_{Kr}-x\,|\,<\,|\,\hat{x}-x\,|\,]\,\mathrm{f}(\hat{x})\,\mathrm{d}\hat{x} \\
&= \int_{-b}^{0} \left(\frac{-\hat{x}(1-\frac{K}{2})+b}{2b}\right)\left(\frac{1}{b}\right)\,\mathrm{d}\hat{x} \\
&= \frac{\hat{x}(K-2)+4b}{8b^2}\hat{x}\,\Big|_{-b}^{0} \\
&= \frac{6-K}{8} \qquad \text{(A.55)}
\end{aligned}$$

This gives the result stated in Theorem 2.3.1 □

A.6 Expected Distance of Fitness-Weighted Quasi-Reflected Point

A.6.1 Probability Distribution Functions

This section defines the expected distance between a fitness-weighted quasi-reflected point, $\hat{x}_{Kr}$, and the solution as a new random variable, Z, which is a function of two RVs.

Distribution of x

We assume that x is uniformly distributed in $[-b, b]$ so $x \sim \mathrm{U}[-b, b]$ and $\mathrm{f}(x) = \frac{1}{2b}$. Fig. 40 illustrates the distribution of x.

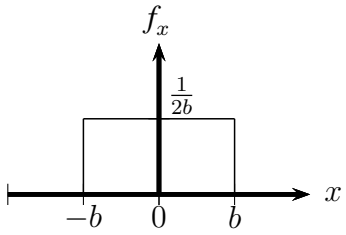


Figure 40. Distribution of x in domain $[-b, b]$.

Distribution of $\hat{x}$

Let us assume that $\hat{x}$, the EA individual, is uniformly distributed in $[-b, 0]$ so $\hat{x} \sim \mathrm{U}[-b, 0]$ and $\mathrm{f}(x) = \frac{1}{b}$. Fig. 41 illustrates the distribution of $\hat{x}$.

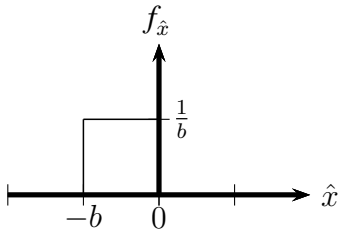


Figure 41. Distribution of $\hat{x}$ in domain $[-b, 0]$.

Distribution of $\hat{x}_{Kr}$

It can be shown that the quasi-reflection, $\hat{x}_{Kr}$, is a function of one random variable, $\hat{x}$, and is uniformly distributed in $[b(K-1), 0]$, or $\hat{x}_{Kr} \sim \mathrm{U}[b(K-1), 0]$. Fig. 42 shows the distribution of $\hat{x}_{Kr}$.

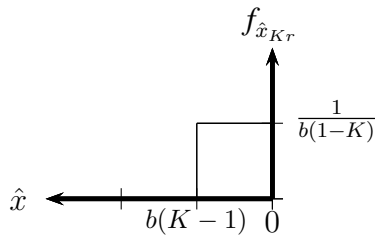


Figure 42. Distribution of $\hat{x}_{Kr}$ in domain $[a, b]$ where K is the reflection weight.

Distribution of two random variables

We find the distribution of two random variables using the following equation from [164]. If X and Y are uniformly distributed random variables and $Z = X - Y$,

$$\mathrm{f}_Z(z) = \int \mathrm{f}_X(z+y)\,\mathrm{f}_Y(y)\,\mathrm{d}y \tag{A.56}$$

If we let $Z = \hat{x}_{Kr} - x$, then $Z \in [b(K-2), b]$ and

$$\mathrm{f}_Z(z) = \int_{-\infty}^{\infty} \mathrm{f}_{\hat{x}_{Kr}}(z+y)\ \mathrm{f}_x(y)\ \mathrm{d}y \tag{A.57}$$

and the expected distance equation can be written as

$$\mathbf{E}[\,|\,\hat{x}_{Kr} - x\,|\,] = \mathbf{E}[\,|\,z\,|\,] = \int\ |\,z\,|\ \ \mathrm{f}_{|\,Z\,|}(z)\,\mathrm{d}z \tag{A.58}$$

Calculation of $\mathbf{f}_{\hat{x}_{Kr}}(z+y)$

Notice that Eq. A.57 is convolution of the random variables $\mathrm{f}_{\hat{x}_{Kr}}(z+y)$ and $\mathrm{f}_x(y)$ as shown in Fig. 44. This convolution requires the pdf $\mathrm{f}_{\hat{x}_{Kr}}(z+y)$. This distribution is obtained by shifting the distribution of $\mathrm{f}_{\hat{x}_{Kr}}(y)$ by z. The result is shown in Fig. 43.

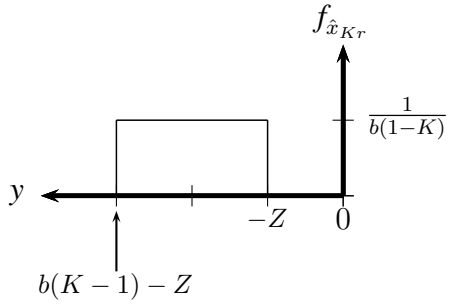


Figure 43. $\mathrm{f}_{\hat{x}_{Kr}}(z+y)$ in domain $[a, b]$ where K is the reflection weight.

A.6.2 Distance between $\hat{x}_{Kr}$ and x

We can now convolve $\mathrm{f}_{\hat{x}_{Kr}}$ and $\mathrm{f}_{\hat{x}}$ to find f_Z. This convolution is calculated graphically based on Fig. 44 as Z shifts from $-b$ to b.

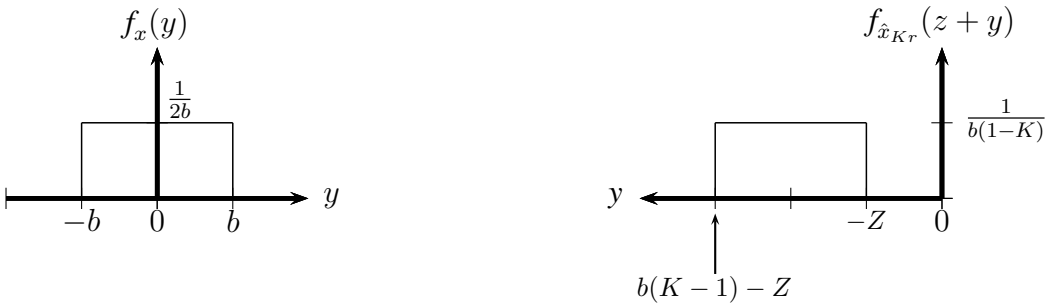


Figure 44. $\mathrm{f}_{\hat{x}_{Kr}-x}(y)$ can be obtained by convolving $\mathrm{f}_{\hat{x}_{Kr}}$ and f_x as Z shifts from $-b$ to b

We will shift Z in $\mathrm{f}_{\hat{x}_{Kr}}(z+y)$ in four steps. Note that each case corresponds to its respective section in Fig. 45. For example, the region calculated in Case A corresponds to Ⓐ in Fig. 45.

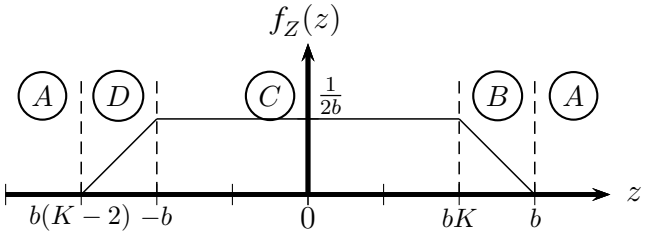


Figure 45. Convolution of $\mathrm{f}_{\hat{x}_{Kr}}$ and f_x.

A) The two end points, as shown in Fig. 46:

i) if $-z < -b$, then $z > b$

ii) if $b(K-1) - z > b$, then $z < b(K-2)$

and it is clear from the figure that the two distributions do not intersect, so the area intersected by the two densities is zero.

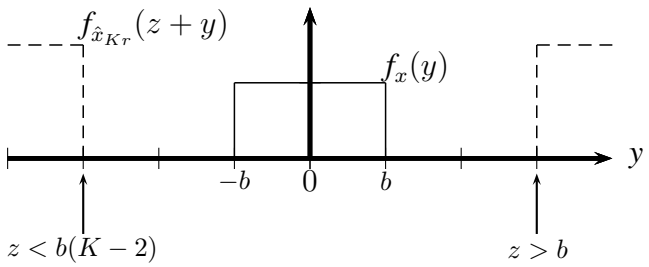


Figure 46. Convolution of $\mathrm{f}_{\hat{x}_{Kr}}$ and f_x. Shifting at the end points

B) Shift in the first leg from the left hand side:

For this case $-z > -b$ and $b(K-1)-z < -b$ as shown in Fig. 47. Thus $bK < z < b$

This case corresponds to Section Ⓑ in Fig. 45. The density of $\mathrm{f}_Z(z)$ in this section can be calculated using the equation for a line:

$y(Z) = mZ + l$

where m is the slope of y and l is the z-axis intersection.

$$m = \frac{y_2 - y_1}{Z_2 - Z_1} = \frac{0 - \frac{1}{2b}}{b - bK} = \frac{-1}{2b^2(1-K)}$$

To calculate the intersection point l, we evaluate y at bK:

$$\begin{aligned} y(bK) &= \frac{1}{2b} = \frac{-1}{2b^2(1-K)} bK + l \\ l &= \frac{1}{2b} + \frac{bk}{2b^2(1-K)} = \frac{b}{2b^2(1-K)} \end{aligned}$$

We can now write $\mathrm{f}_Z(z)$ in Section Ⓑ as

$$\mathrm{f}_Z(z) = \frac{-z+b}{2b^2(1-K)} \text{ for } bK < z < b \tag{A.59}$$

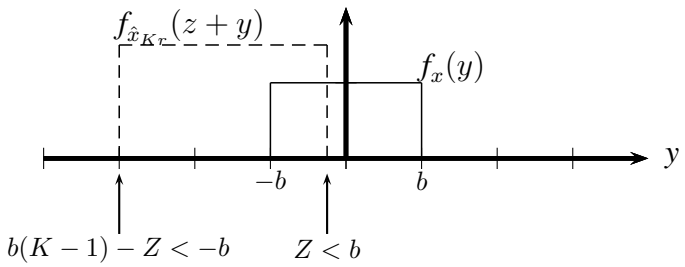


Figure 47. Convolution of $\mathrm{f}_{\hat{x}_{Kr}}$ and f_x. As z is increased, $\mathrm{f}_{\hat{x}_{Kr}}(z+y)$ is overlapping $\mathrm{f}_x(y)$

C) Both legs of $\mathrm{f}_{\hat{x}_{Kr}}(z+y)$ shifted in $\mathrm{f}_{\hat{x}}(y)$:

This case corresponds to Section Ⓒ in Fig. 45. For this case $-z > -b$ and $b(K-1)-z > -b$ as shown in Fig. 48. Thus $-b < z < bK$

For Section Ⓒ, we can calculate the distribution of $\mathrm{f}_Z(z)$ as

$$\begin{aligned} \mathrm{f}_Z(z) &= \frac{1}{2b}\frac{1}{b(1-K)}\left[(z-bK)-(z+b)\right] \\ &= \frac{1}{2b} \text{ for } -b < z < bK \end{aligned} \tag{A.60}$$

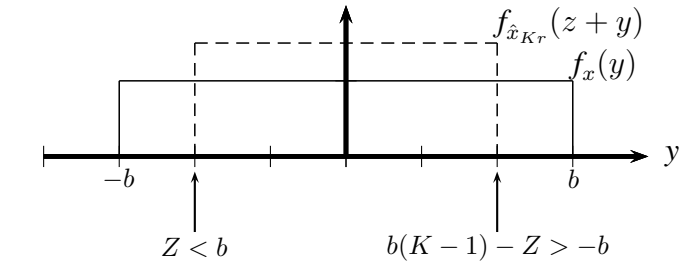


Figure 48. Convolution of $\mathrm{f}_{\hat{x}_{Kr}}$ and f_x. $\mathrm{f}_{\hat{x}}(z+y)$ is enclosed in $\mathrm{f}_{\hat{x}}(y)$ as z is increased

D) Shift out the first leg from right hand side:

This case corresponds to Section Ⓓ in Fig. 45. For this case $-z > b$ and $b(K-1)-z < b$ as shown in Fig. 49. Thus $b(K-2) < z < -b$

Once again, we use the equation for a line $y(z) = mz + l$ where the slope, m is

$$m = \frac{y_2 - y_1}{Z_2 - Z_1} = \frac{\frac{1}{2b} - 0}{b - bK} = \frac{1}{2b^2(1-K)}$$

To calculate the intersection point l, we evaluate y at $b(K-2)$:

$$y(b(K-2)) = 0 = \frac{1}{2b^2(1-K)} b(K-2) + l$$
$$l = \frac{-b(K-2)}{2b^2(1-K)}$$

We can now write $\mathrm{f}_Z(z)$ in Section Ⓓ as

$$\mathrm{f}_Z(z) = \frac{z - b(K-2)}{2b^2(1-K)} \text{ for } b(K-2) < z < -b \tag{A.61}$$

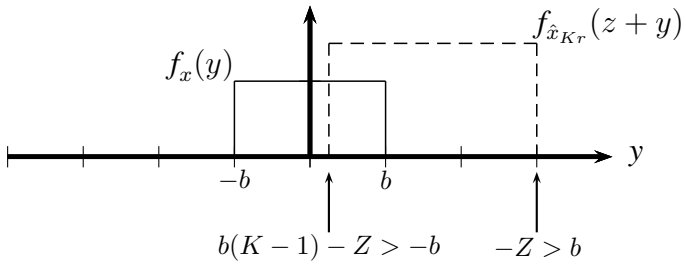


Figure 49. Convolution of $\mathrm{f}_{\hat{x}_{Kr}}$ and f_x. $\mathrm{f}_{\hat{x}}(z+y)$ starts shifting out of $\mathrm{f}_x(y)$ as z is increased

Combining Cases A-D, we obtain $\mathrm{f}_Z(z)$ as:

$$\mathrm{f}_Z(z) = \int_{-\infty}^{\infty} \mathrm{f}_{\hat{x}_{Kr}}(z+y)\ \mathrm{f}_x(y)\ \mathrm{d}y = \begin{cases} 0 & \text{if } z < b(K-2) \\ \frac{z-b(K-2)}{2b^2(1-K)} & \text{if } b(K-2) < z < -b \\ \frac{1}{2b} & \text{if } -b < z < bK \\ \frac{-z+b}{2b^2(1-K)} & \text{if } bK < z < b \\ 0 & \text{if } z > b \end{cases} \tag{A.62}$$

These results are also presented in Fig. 45.

A.6.3 Absolute Value of Distance between $\hat{x}_{Kr}$ and x

Based on Eq. A.62, we calculate $\mathrm{f}_{|Z|}(z)$ graphically as shown in Fig. 50.

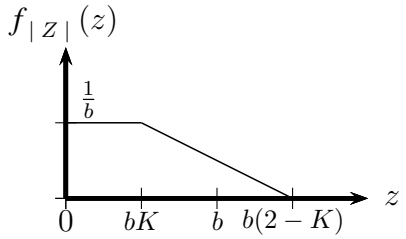


Figure 50. $\mathrm{f}_{|Z|}(z)$

Fig. 50 can be mathematically defined as:

$$\mathrm{f}_{|Z|}(z) = \begin{cases} 0 & \text{if } z < 0 \\ \frac{1}{b} & \text{if } 0 < z < bK \\ \frac{-z+b(K-2)}{2b^2(K-1)} & \text{if } bK < z < b(2-K) \\ 0 & \text{if } z > b(2-K) \end{cases} \tag{A.63}$$

A.6.4 Expected Distance between $\hat{x}_{Kr}$ and x

Lemma 2.4.1 *Assume that the solution of an optimization problem is uniformly distributed in a one-dimensional search space, the expected distance between* $\hat{x}_{Kr}$ *and* x *is* $\left[3bK^2 - 2b(K-1)(2+K)\right]/6$.

Proof. The expected distance can be calculated using Eq. A.63 as

$$\begin{aligned} \mathbf{E}\left[\,|\,\hat{x}_{Kr} - x\,|\,\right] &= \mathbf{E}\left[\,|\,z\,|\,\right] = \int |\,z\,|\;\mathrm{f}_{|Z|}(z)\,\mathrm{d}z \\ &= \int_0^{b(2-K)} z\,\mathrm{f}_{|Z|}(z)\,\mathrm{d}z \\ &= \int_0^{bK} z\,\mathrm{f}_{|Z|}(z)\,\mathrm{d}z + \int_{bK}^{b(2-K)} z\,\mathrm{f}_{|Z|}(z)\,\mathrm{d}z \\ &= \int_0^{bK} z\,\frac{1}{b}\,\mathrm{d}z + \int_{bK}^{b(2-K)} z\,\frac{z+b(K-2)}{2b^2(K-1)}\,\mathrm{d}z \\ &= \frac{1}{b}\,\frac{z^2}{2}\,\Big|_0^{bK} + \frac{1}{2b^2(K-1)}\,\frac{z^3}{3} + \frac{z^2(b-K)}{2}\,\Big|_{bK}^{b(K-2)} \\ &= \frac{bK^2}{2} - \frac{b(K-1)(2+K)}{3} \end{aligned} \tag{A.64}$$

This gives the result stated in Lemma 2.4.1 □

A.6.5 Distance between $\hat{x}$ and x

To find the expected distance between EA individual and the solution, we will again refer to Eq. A.56. This time, we let $Z = \hat{x} - x$, then

$$\mathrm{f}_Z(z) = \int_{-\infty}^{\infty} \mathrm{f}_{\hat{x}}\,(z+y)\ \mathrm{f}_x\,(y)\ \mathrm{d}y \tag{A.65}$$

Eq. A.65 reflects a convolution of two random variables as Z shifts from $-b$ to b and is done graphically based on Fig. 51.

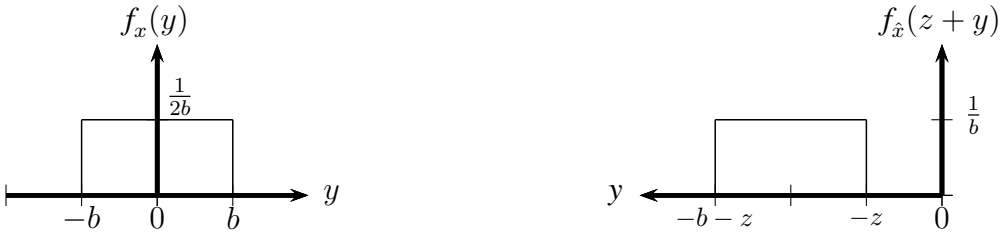


Figure 51. $\mathrm{f}_{\hat{x}-x}(y)$ can be obtained by convolving $\mathrm{f}_{\hat{x}}$ and f_x as z shifts from $[-b, b]$

Cases as z shifts $-b$ to b are shown in Fig. 52.

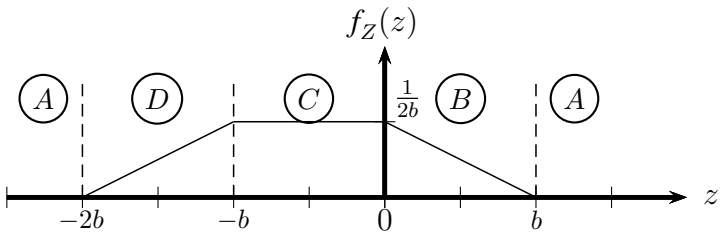


Figure 52. Convolution of $\mathrm{f}_{\hat{x}}$ and f_x.

A) The two end points, as shown in Fig. 53.

i) if $-z < -b$, then $z > b$

ii) if $-b - z > b$, then $z < -2b$

Based on Fig. 51, we can see that that two densities do not intersect, hence $\mathrm{f}_Z(z)$ is zero for this case. This case corresponds to Section Ⓐ in Fig. 52.

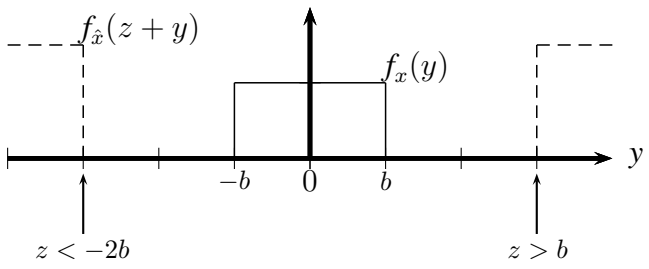


Figure 53. Convolution of $\mathrm{f}_{\hat{x}_{Kr}}$ and f_x. Shifting at the end points

B) Shift in the first leg from left hand side:

For this case, we increase z as $-z > -b$ until $-b - z < -b$ as shown in Fig. 54. Thus $0 < z < b$

The density of $\mathrm{f}_Z(z)$ in this section increases as z increases until the densities overlap when $-b - z = -b$. This case corresponds to Section Ⓑ in Fig. 52.

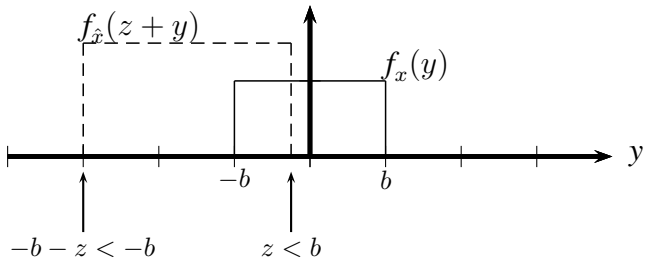


Figure 54. Convolution of $\mathrm{f}_{\hat{x}}$ and f_x. As z is increased, $\mathrm{f}_{\hat{x}}(z+y)$ is overlapping $\mathrm{f}_x(y)$

C) Both legs of $\mathrm{f}_{\hat{x}}(z+y)$ shifted in $\mathrm{f}_{\hat{x}}(y)$:

We continue to increase z while $\mathrm{f}_{\hat{x}}(z+y)$ is within $\mathrm{f}_{\hat{x}}(y)$. For this case $-z < b$ and $-b - z > -b$ as shown in Fig. 55. Thus $-b < z < 0$

For Section Ⓒ, we can see that $\mathrm{f}_Z(z)$ stays constant at its peak as seen in Fig. 52.

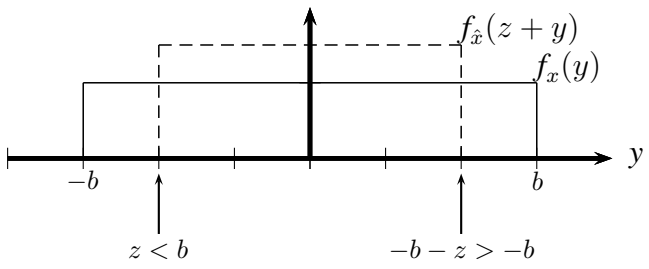


Figure 55. Convolution of $\mathrm{f}_{\hat{x}_{Kr}}$ and f_x. Shifting at the end points

D) Shift out the first leg from the right hand side:

For this case, $\mathrm{f}_{\hat{x}}(z+y)$ starts shifting out of $\mathrm{f}_{\hat{x}}(y)$. The boundaries for z are $-z > b$ and $-b - z < b$ as shown in Fig. 56. This corresponds to $-2b < z < -b$ in Fig. 52, also labeled as Section Ⓓ .

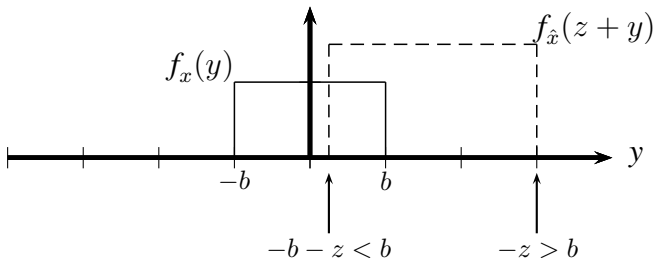


Figure 56. Convolution of $\mathrm{f}_{\hat{x}}$ and f_x. $\mathrm{f}_{\hat{x}}(z+y)$ starts shifting out of $\mathrm{f}_x(y)$ as z is increased.

Combining the results from Cases A-D, we obtain $\mathrm{f}_Z(z)$ as:

$$\mathrm{f}_Z(z) = \int_{-\infty}^{\infty} \mathrm{f}_{\hat{x}}\,(z+y)\;\mathrm{f}_x\,(y)\;\mathrm{d}y = \begin{cases} 0 & \text{if } z < -2b \\ \frac{z+2b}{2b^2} & \text{if } -b < z < 0 \\ \frac{1}{2b} & \text{if } -b < z < 0 \\ \frac{b-z}{2b^2} & \text{if } 0 < z < b \\ 0 & \text{if } z > b \end{cases} \tag{A.66}$$

A.6.6 Absolute Value of Distance between $\hat{x}$ and x

We calculate $\mathrm{f}_{|Z|}(z)$ graphically based on Eq. A.66. The result is shown in Eq. A.67 and Fig. 57.

$$\mathrm{f}_{|Z|}(z) = \begin{cases} 0 & \text{if } z < 0 \\ \frac{-z+2b}{2b^2} & \text{if } 0 < z < 2b \\ 0 & \text{if } z > 2b \end{cases} \tag{A.67}$$

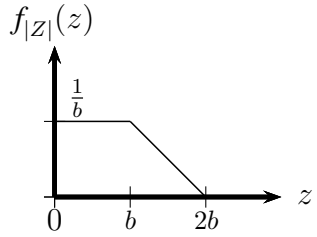


Figure 57. $\mathrm{f}_{|Z|}(z)$

A.6.7 Expected Distance between $\hat{x}$ and x

Lemma 2.4.2 *Assume that the solution of an optimization problem is uniformly distributed in a one-dimensional search space, the expected distance between $\hat{x}$ and x is*

$$\boldsymbol{E}\left[\,|\,\hat{x}-x\,|\,\right] = \frac{2b}{3} \tag{A.68}$$

Proof. The expected distance can be calculated using Eq. A.67 as

$$\begin{aligned} \mathbf{E}\left[\,|\,\hat{x}-x\,|\,\right] &= \mathbf{E}\left[\,|\,z\,|\,\right] = \int\ |\,z\,|\ \ \mathrm{f}_{|\,Z\,|}\,(z)\,\mathrm{d}z \\ &= \int_0^{2b} z\,\mathrm{f}_{|\,Z\,|}\,(z)\,\mathrm{d}z = \int_0^{2b} z\,\frac{-z+2b}{2b^2}\,\mathrm{d}z \\ &= \left.\frac{z^2}{2b} - \frac{z^3}{6b^2}\,\right|_0^{2b} \\ &= \frac{2b}{3} \end{aligned} \tag{A.69}$$

This gives the result stated in Lemma 2.4.2 □

APPENDIX B

BENCHMARK FUNCTIONS

B.1 Low-Dimensional Benchmark Problems

B.1.1 Beale

The equation representing the Beale function [174] is given in Eq. B.1 and is plotted as a mesh contour plot in Fig. 58. Table XXXVI provides the overview of the problem.

$$F(\vec{x}) = [1.5 - x_1(1 - x_2)]^2 + \left[2.25 - x_1(1 - x_2^2)\right]^2 + \left[2.625 - x_1(1 - x_2^3)\right]^2 \quad \text{(B.1)}$$

Table XXXVI. Beale function overview

Function	Domain	argmin	min f(x)
Beale	$(-4.5, 4.5)^2$	$(3, 0.5)$	0

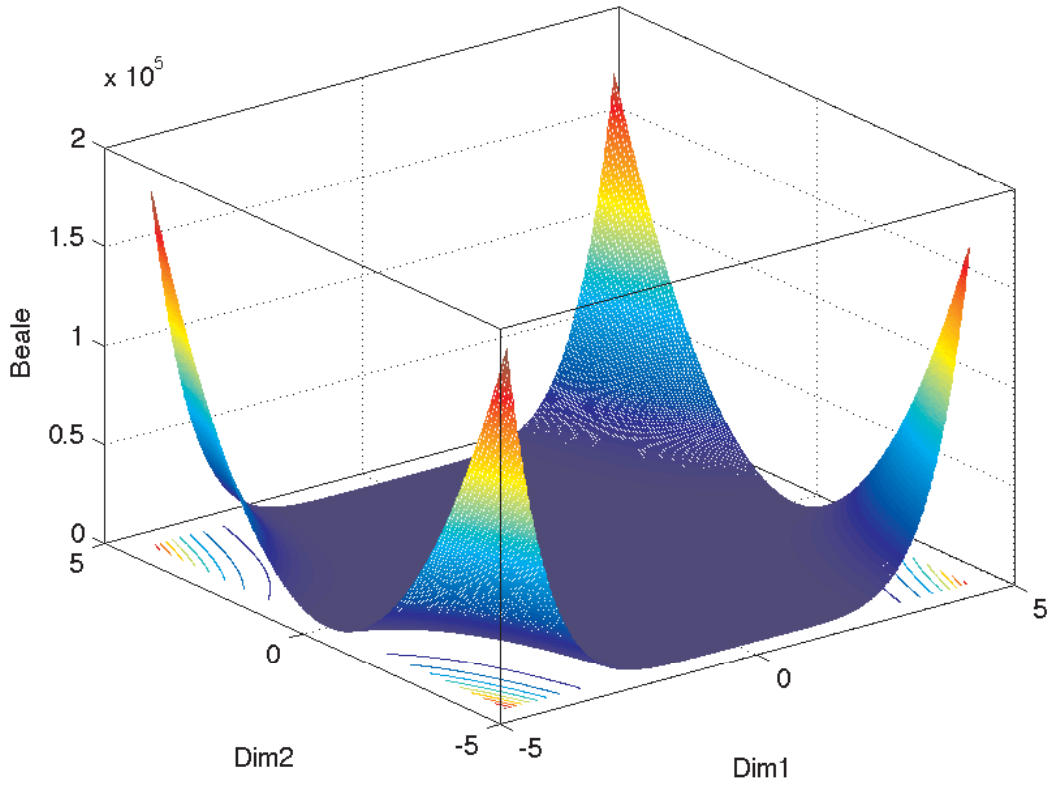


Figure 58. Two dimensional plot of the Beale Function

B.1.2 Colville

The equation representing the Colville function [174] is given in Eq. B.2. Table XXXVII provides the overview of the problem. Because of the dimension size, we cannot include a plot of this function.

$$\begin{aligned} F(\vec{x}) &= 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + \\ &\quad 10.1\left((x_2 - 1)^2 + (x_4 - 1)^2\right) + 19.8(x_2 - 1)(x_4 - 1) \end{aligned} \tag{B.2}$$

Table XXXVII. Colville function overview

Function	Domain	argmin	min f(x)
Colville	$(-10, 10)^4$	1^4	0

B.1.3 DeJong F5

The equation representing Shekel's Foxhole function [218], also known as DeJong F5, is given in Eq. B.3 and is plotted as a mesh contour plot in Fig. 59. The function is currently set to have 25 foxholes. Table XXXVIII provides the overview of the problem.

$$F(\vec{x}) = \frac{1}{0.002 + \sum_{i=1}^{25} \frac{1}{i + \sum_{j=1}^{2}(x_j - a_{ji})^6}} \tag{B.3}$$

$$\text{where } [a_{ji}] = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & \vdots & -32 & -16 & ... & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & \vdots & -16 & -16 & ... & 32 & 32 & 32 \end{bmatrix}$$

Table XXXVIII. DeJong F5 function overview

Function	Domain	argmin	min f(x)
DeJong F5	$(-65.536, 65.536)^{25}$	$(-32, 32)^{25}$	0.998

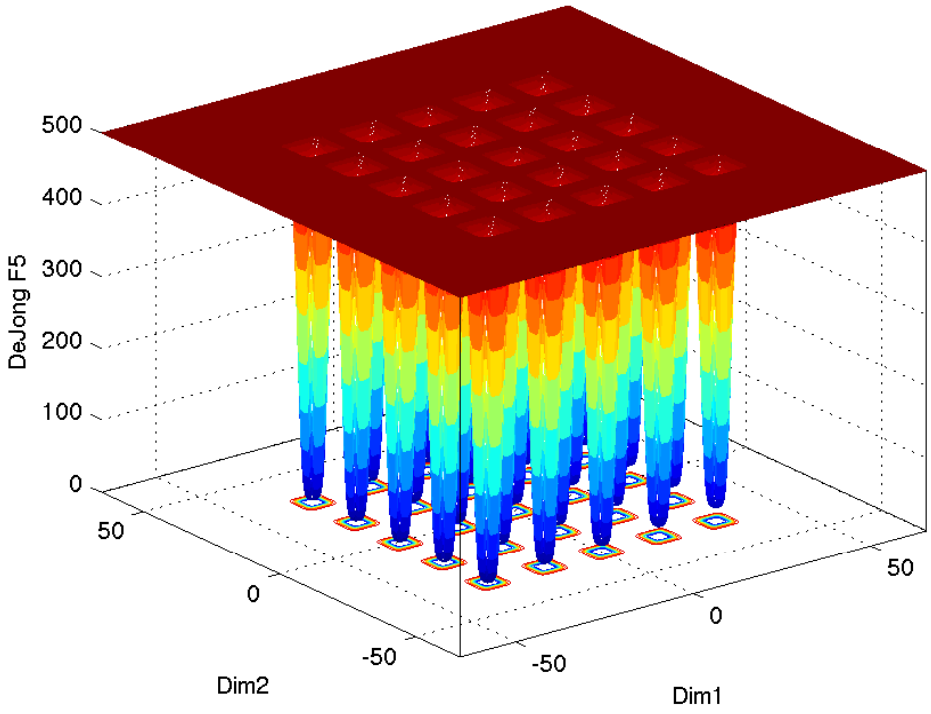


Figure 59. Two dimensional plot of the DeJong F5 function

B.1.4 Easom

The equation representing the Easom function [219] is given in Eq. B.4 and is plotted as a mesh contour plot in Fig. 60. Table XXXIX provides the overview of the problem.

$$F(\vec{x}) = -\cos(x_1)\cos(x_2)e^{-(x_1-\pi)^2-(x_2-\pi)^2} \tag{B.4}$$

Table XXXIX. Easom function overview

Function	Domain	argmin	min f(x)
Easom	$(-100, 100)^2$	(π, π)	-1

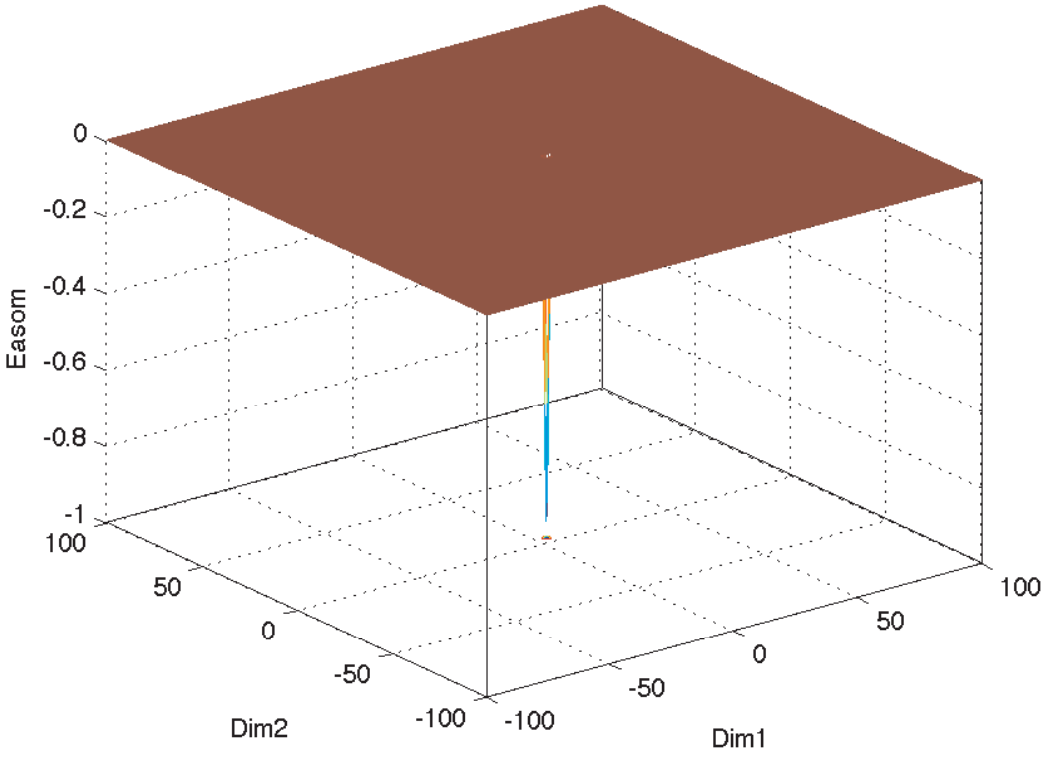


Figure 60. Two dimensional plot of the Easom function

B.1.5 Perm

The equation representing the Perm function [141] is given in Eq. B.5 and is plotted as a mesh contour plot in Fig. 61. Table XL provides the overview of the problem. Even though this is a multidimensional function, it could not be solved at 20 dimensions and therefore, only used as a low dimension benchmark.

$$F(\vec{x}) = \sum_{k=1}^{n} \left[\sum_{i=1}^{n} \left(i^k + 0.5\right) \left(\left(\frac{x_i}{i}\right)^k - 1 \right) \right]^2 \tag{B.5}$$

Table XL. Perm function overview

Function	Domain	argmin	min f(x)
Perm	$(-n, n)^n$	$(1, 2, ..., n)$	0

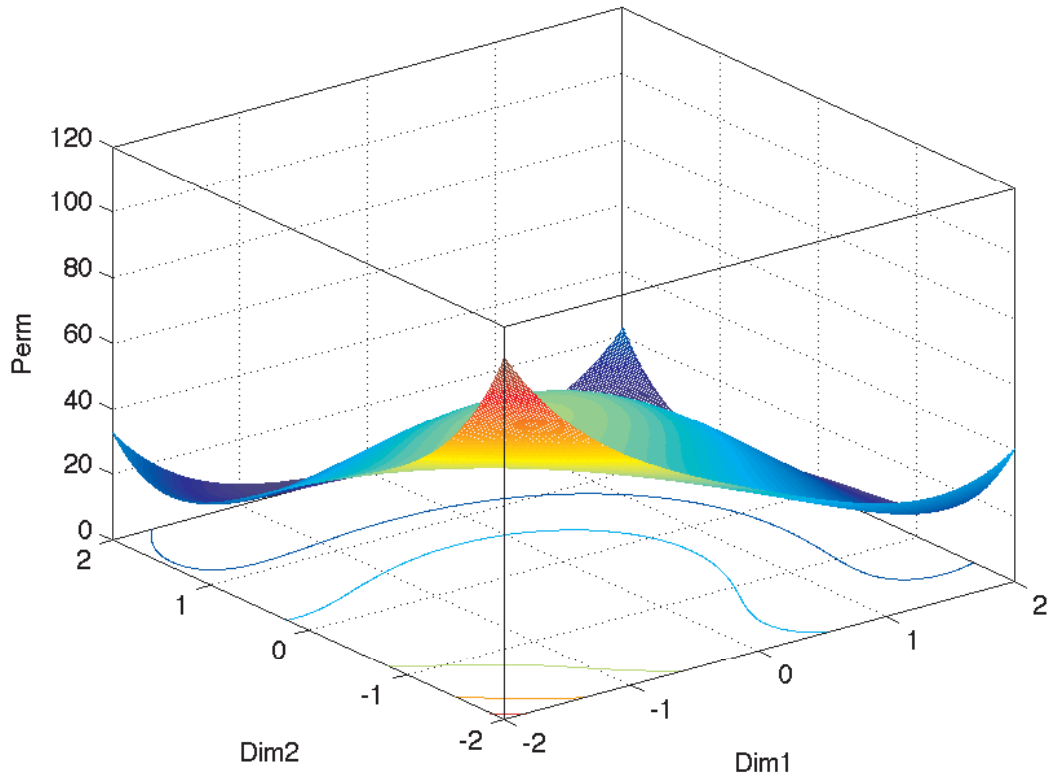


Figure 61. Two dimensional plot of the Perm function

B.1.6 Tripod

This function is taken from [140]. The equation representing the Tripod function is given in Eq. B.6 and is plotted as a mesh contour plot in Fig. 62. Table XLI provides the overview of the problem.

$$\begin{aligned} F(\vec{x}) &= \mathrm{p}(x_2)\left(1 + (x_1)\right) + |x_1 + 50\mathrm{p}(x_2)\left(1 - 2\mathrm{p}(x_1)\right)| + \\ &\quad |x_2 + 50\left(1 - 2\mathrm{p}(x_2)\right)| \end{aligned} \tag{B.6}$$

$$\text{where } \mathrm{p}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Table XLI. Tripod function overview

Function	Domain	argmin	min f(x)
Tripod	$(-100, 100)^2$	$(0, -50)$	0

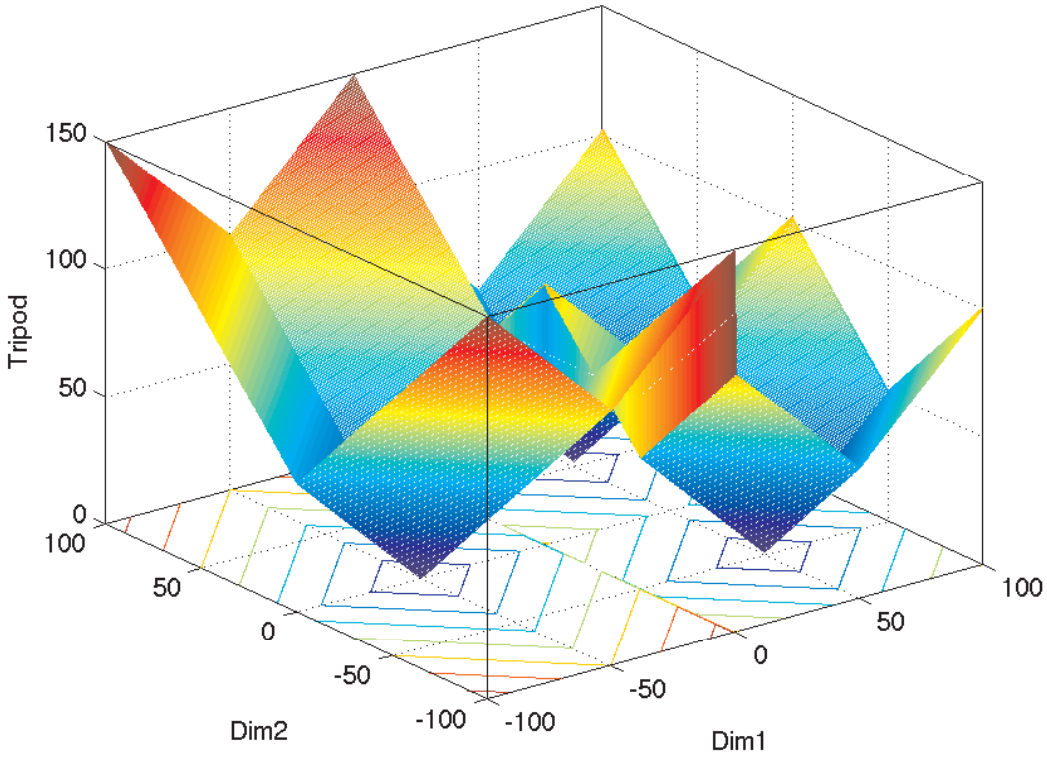


Figure 62. Two dimensional plot of the Tripod function

B.2 Variable-Dimensional Benchmark Problems

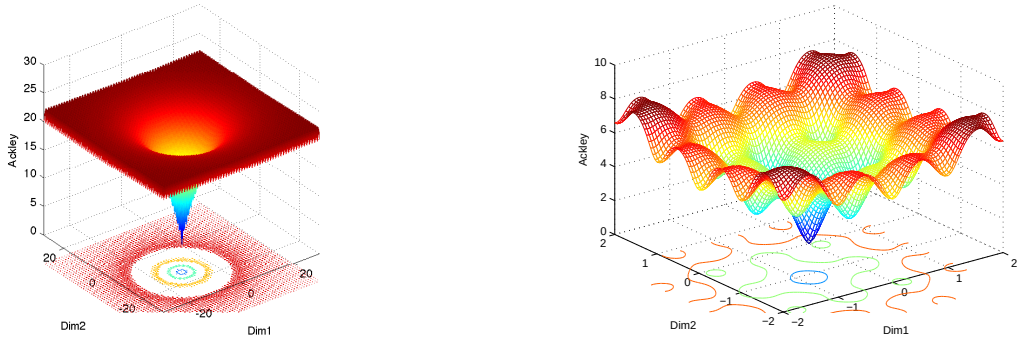
B.2.1 Ackley

This popular function is first published in [220] as a two-dimensional problem and later extended to n-dimensions in [221]. The equation representing the Ackley function is given in Eq. B.7 and is plotted as a mesh contour plot in Fig. 63. Table XLII provides the overview of the problem.

$$F(\vec{x}) = -20 \cdot exp\left(-0.2\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2}\right) - exp\left[\frac{1}{n} \cdot \sum_{i=1}^{n} \cos(2\pi x_i)\right] + 20 + exp(1) \quad \text{(B.7)}$$

Table XLII. Ackley function overview

Function	Domain	argmin	min f(x)
Ackley	$(-32.768, 32.768)^n$	0^n	0



(a) (b)

Figure 63. Two dimensional plot of the Ackley function. Figure a) illustrates the function in its full domain while b) is zoomed in to $[-2, 2]$

B.2.2 Alpine

The equation representing the Alpine function [141] is given in Eq. B.8 and is plotted as a mesh contour plot in Fig. 64. Table XLIII provides the overview of the problem.

$$F(\vec{x}) = \sum_{i=1}^{n} |x_i \sin(x_i) + 0.1x_i| \tag{B.8}$$

Table XLIII. Alpine function overview

Function	Domain	argmin	min f(x)
Alpine	$(-10, 10)^n$	0^n	0

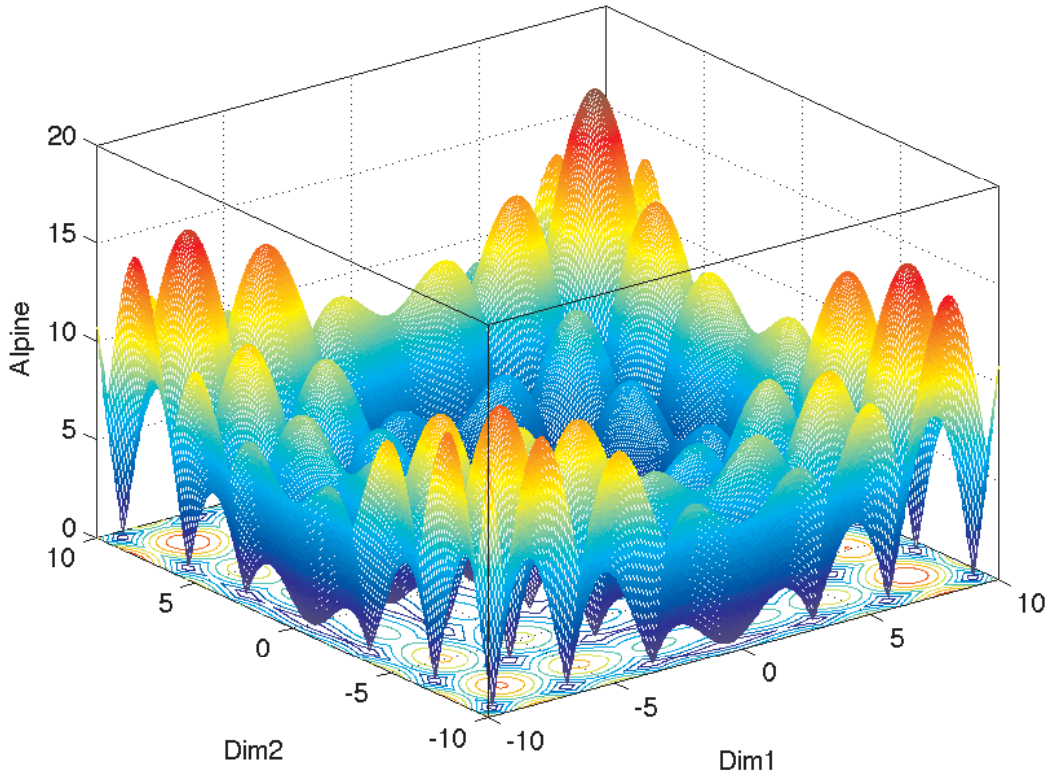


Figure 64. Two dimensional plot of the Alpine function

B.2.3 Fletcher/Powell

The Fletcher function is given in Eq. B.9 [222] and is plotted as a mesh contour plot in Fig. 65. Table XLIV provides the overview of the problem.

Note that the parameters of Fletcher functions and its minimum, α is randomly distributed. Therefore, the function will have a different contour plot for each simulation. The plotted results are obtained when $\alpha = [0.97, 0.20]$.

$$\begin{aligned} A_i &= \sum_{j=1}^{n} a_{i,j} \sin(\alpha_j) + b_{i,j} \cos(\alpha_j) \quad i = 1, 2, \ldots, n \\ B_i &= \sum_{j=1}^{n} a_{i,j} \sin(x_j) + b_{i,j} \cos(x_j) \quad i = 1, 2, \ldots, n \\ F(\vec{x}) &= \sum_{i=1}^{n} (A_i - B_i)^2 \end{aligned} \tag{B.9}$$

where $a_{i,j}$ and $b_{i,j}$ are random numbers $\in [-100, 100]$ and α_j is random $\in [-\pi, \pi]$.

Table XLIV. Fletcher function overview

Function	Domain	argmin	min f(x)
Fletcher	$(-\pi, \pi)^n$	α^n	0

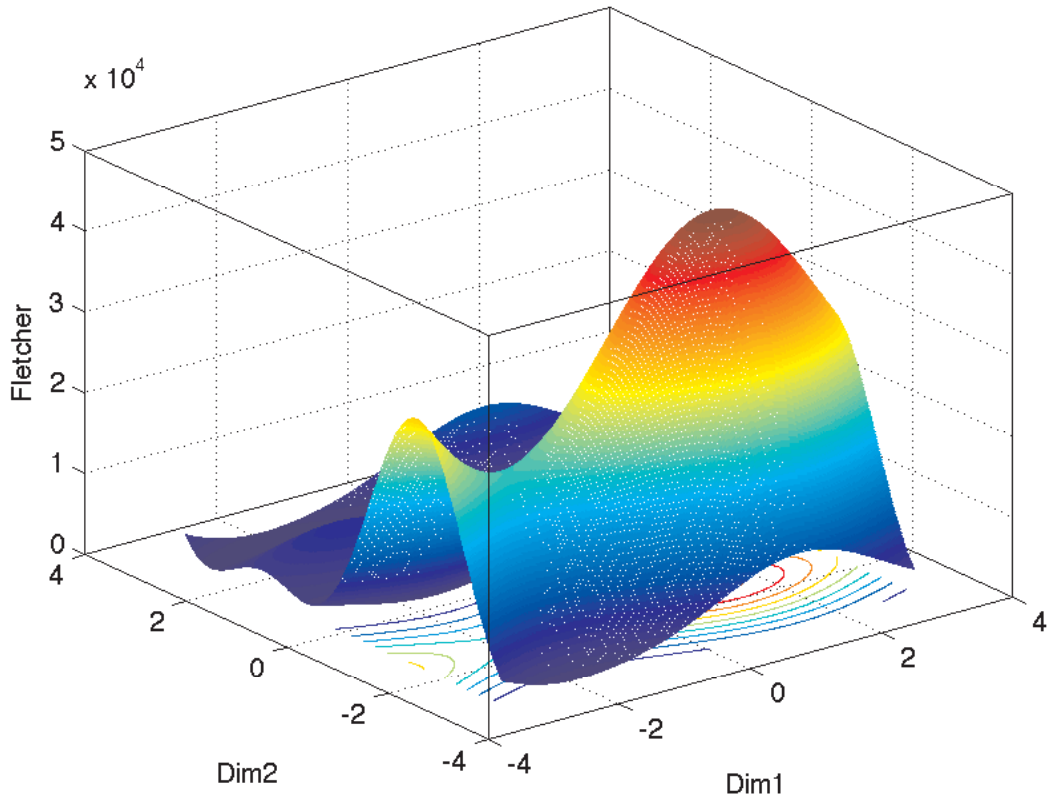


Figure 65. Two dimensional plot of the Fletcher function when $\alpha = [0.97, 0.20]$.

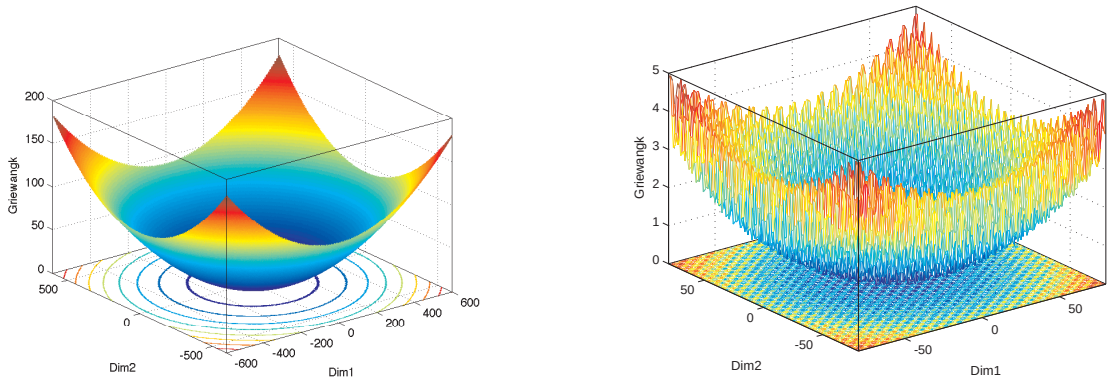
B.2.4 Griewangk

The Griewangk function is given in Eq. B.10 [223] and is plotted as a mesh contour plot in Fig. 66. Table XLV provides the overview of the problem. The surface of Griewang has an abundance of local minima and to present these, we zoom in and plot a smaller section of its domain in Fig. 66.

$$F(\vec{x}) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) \tag{B.10}$$

Table XLV. Griewangk function overview

Function	Domain	argmin	min f(x)
Griewangk	$(-600, 600)^n$	0^n	0



(a) (b)

Figure 66. Two dimensional plot of the Griewangk function. Figure a) illustrates the function in its full domain while b) is zoomed in to $[-80, 80]$

B.2.5 Penalty 1

Penalty 1 function is inspired from Problem 7 in [179] and has approximately 5^n local minimums. Penalty 1 function is given in Eq. B.11 and is plotted as a mesh contour plot in Fig. 67. Table XLVI provides the overview of the problem.

$$
\begin{aligned}
g(x) &= \frac{\pi}{n}\Bigg[10\sin^2(\pi y_1) + \\
&\quad \sum_{i=1}^{n-1}(y_i-1)^2\left(1+10\sin^2(\pi y_{i+1})\right) + (y_n-1)^2\Bigg] \\
y_i &= 1+\frac{x_i-1}{4} \quad i=1,2,\ldots,n \\
u_i &= \begin{cases} 100(x_i-10)^4 & \text{if } x_i > 10 \\ 0 & \text{if } -10 \le x_i \le 10 \quad i=1,2,\ldots,n \\ 100(-x_i-10)^4 & \text{if } x_i < -10 \end{cases} \\
F(\vec{x}) &= g(x) + \sum_{i=1}^{n} u(x_i)
\end{aligned}
\tag{B.11}
$$

Table XLVI. Penalty 1 function overview

Function	Domain	argmin	min f(x)
Penalty 1	$(-50, 50)^n$	1^n	0

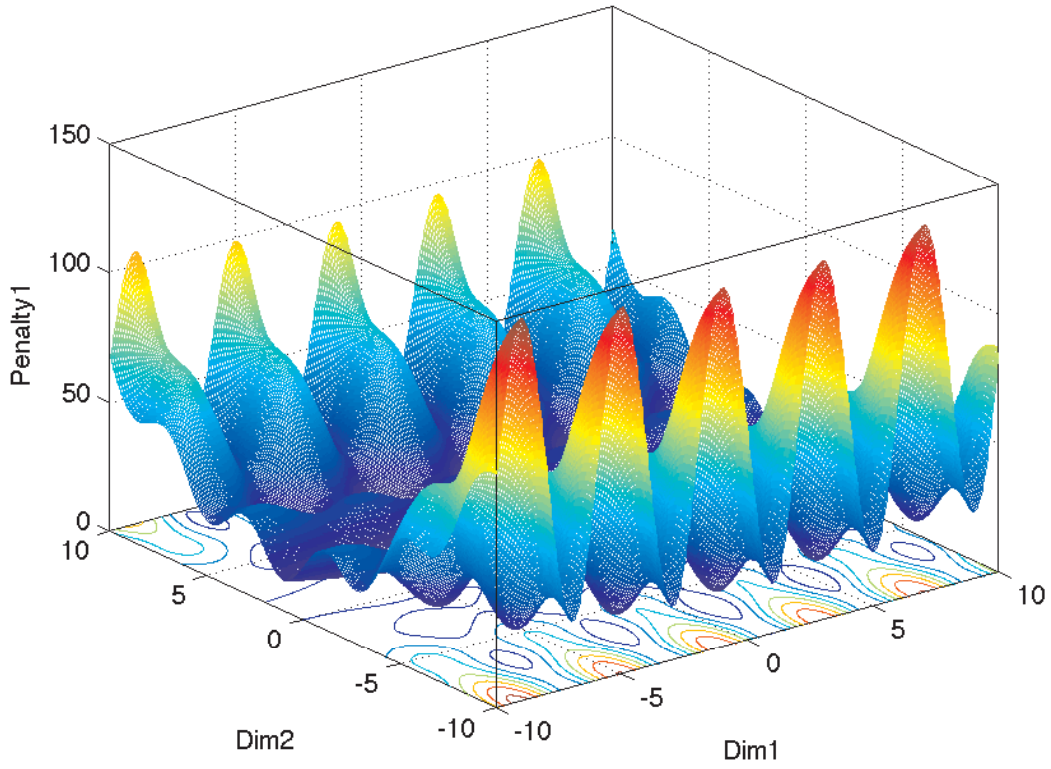


Figure 67. Two dimensional plot of the Penalty 1 function.

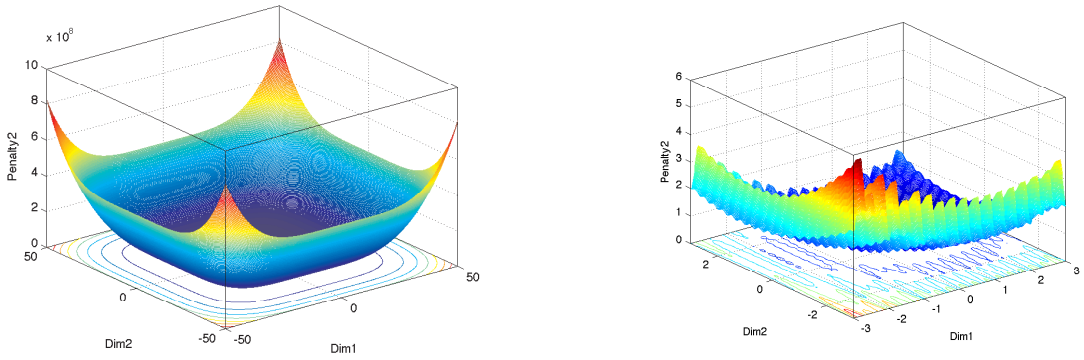
B.2.6 Penalty2

Penalty 2 function is inspired from Problem 18 in [179] and has approximately 30^n local minimums. Penalty 2 function is given in Eq. B.12 and is plotted as a mesh contour plot in Fig. 68. Table XLVII provides the overview of the problem.

$$
\begin{aligned}
g(x) &= 0.1\Bigg[\sin^2(3\pi x_1) + \\
&\quad \sum_{i=1}^{n-1}(x_i-1)^2\left(1+\sin^2(3\pi x_{i+1})\right) + \\
&\quad (x_n-1)^2\left(1+\sin^2(2\pi x_n)\right)\Bigg] \\
u_i &= \begin{cases} 100(x_i-5)^4 & \text{if } x_i > 5 \\ 0 & \text{if } -5 \le x_i \le 5 \quad i=1,2,\ldots,n \\ 100(-x_i-5)^4 & \text{if } x_i < -5 \end{cases} \\
F(\vec{x}) &= g(x) + \sum_{i=1}^{n} u(x_i)
\end{aligned}
\tag{B.12}
$$

Table XLVII. Penalty 2 function overview

Function	Domain	argmin	min f(x)
Penalty 2	$(-50,50)^n$	1^n	0



(a) (b)

Figure 68. Two dimensional plot of the Penalty 2 function. Figure a) illustrates the function in its full domain while b) is zoomed in to $[-3,3]$

B.2.7 Quartic

We employ the quartic function with uniformly distributed noise on the open interval $(0, 1)$. Quartic function, also referred as the forth DeJong function [224], is given in Eq. B.13 and is plotted as a mesh contour plot in Fig. 69. Table XLVIII provides the overview of the problem.

$$F(\vec{x}) = \sum_{i=1}^{n} i x_i^4 + \text{rand}(0, 1) \tag{B.13}$$

where rand is pseudorandom Gaussian noise.

Table XLVIII. Quartic function overview

Function	Domain	argmin	min f(x)
Quartic	$(-1.28, 1.28)^n$	0^n	0

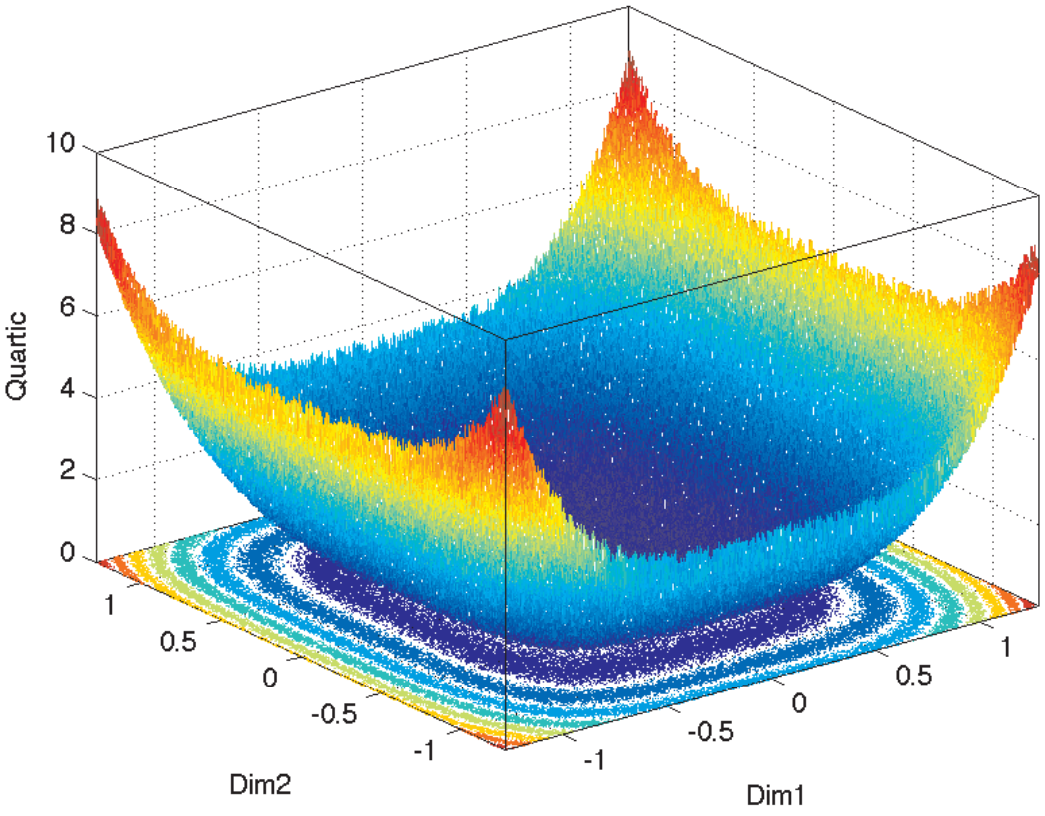


Figure 69. Two dimensional plot of the Quartic function.

B.2.8 Rastrigin

Rastrigin function is a modified version of the sphere problem [225]. It has been made multimodal with the addition of the cosine term. Rastrigin function is given in Eq. B.14 and is plotted as a mesh contour plot in Fig. 70. Table XLIX provides the overview of the problem.

$$F(\vec{x}) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i)\right) \tag{B.14}$$

Table XLIX. Rastrigin function overview

Function	Domain	argmin	min f(x)
Rastrigin	$(-5.12, 5.12)^n$	0^n	0

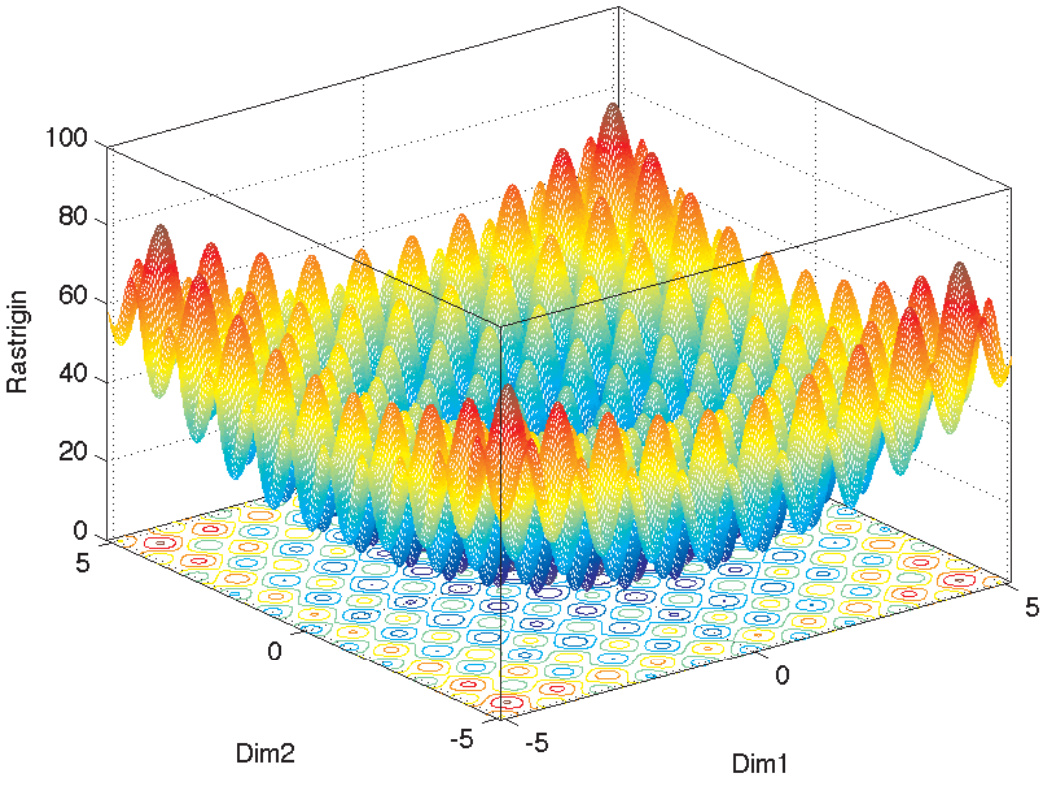


Figure 70. Two dimensional plot of the Rastrigin function.

B.2.9 Rosenbrock

Rosenbrock function is proposed in [226] as a two-dimensional benchmark problem. Due to the valley-like shape of the function, it is challenging to converge to the global optimum. It has been extended to higher dimensions by different authors. Rosenbrock function is given in Eq. B.15 and is plotted as a mesh contour plot in Fig. 71. Table L provides the overview of the problem.

$$F(\vec{x}) = \sum_{i=1}^{n-1} \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right) \tag{B.15}$$

Table L. Rosenbrock function overview

Function	Domain	argmin	min f(x)
Rosenbrock	$(-2.048, 2.048)^n$	1^n	0

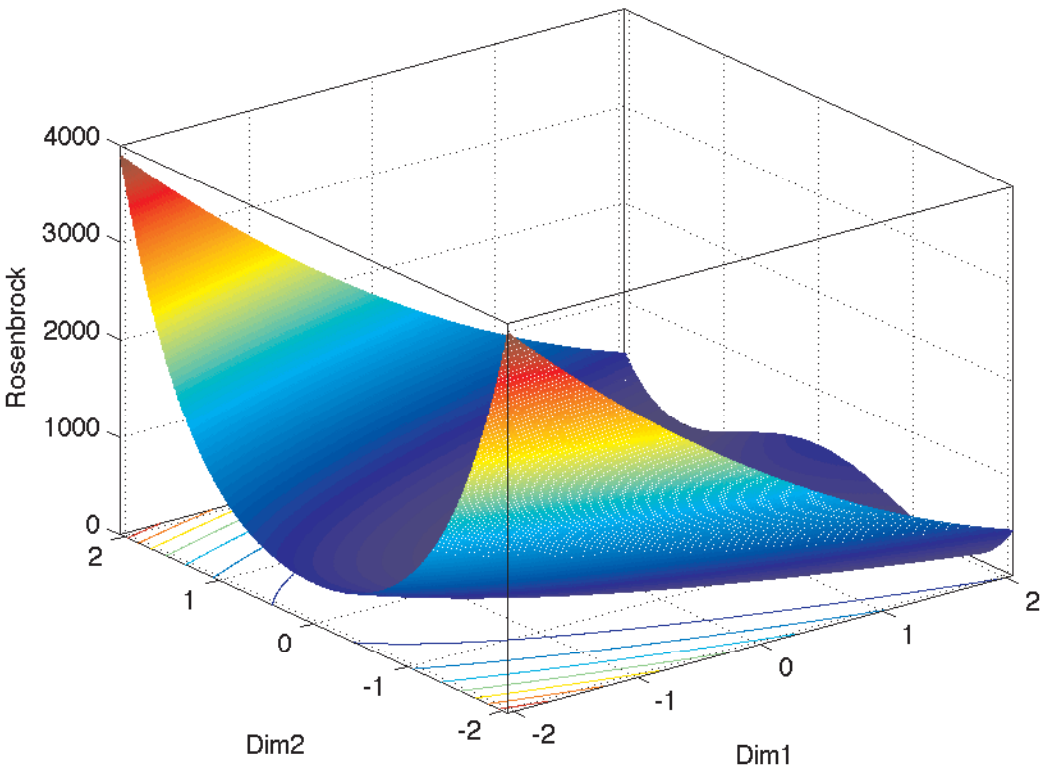


Figure 71. Two dimensional plot of the Rosenbrock function.

B.2.10 Schwefel 1.2

Schwefel 1.2, also referred as Schwefel's double sum function, is another popular benchmark [227]. Schwefel 1.2 function is given in Eq. B.16 and is plotted as a mesh contour plot in Fig. 72. Table LI provides the overview of the problem.

$$F(\vec{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2 \tag{B.16}$$

Table LI. Schwefel 1.2 function overview

Function	Domain	argmin	min f(x)
Schwefel 1.2	$(-65.536, 65.536)^n$	0^n	0

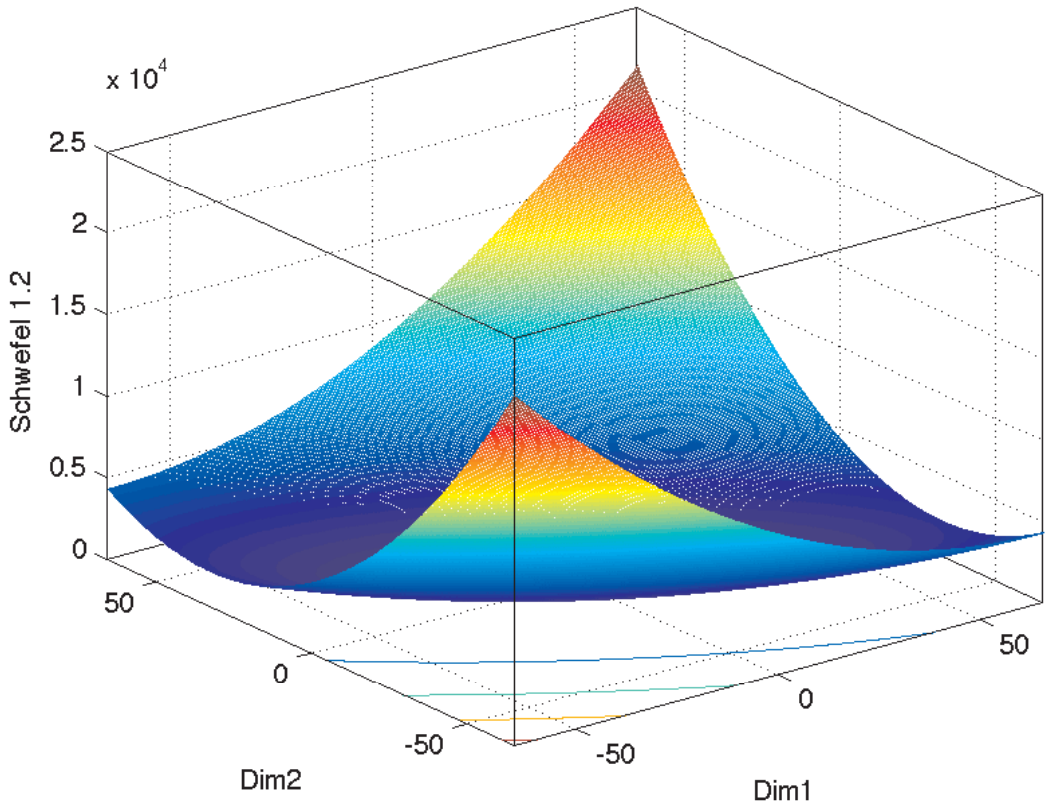


Figure 72. Two dimensional plot of the Schwefel 1.2 function.

B.2.11 Schwefel 2.21

Schwefel 2.21 function is given in Eq. B.17 and is plotted as a mesh contour plot in Fig. 73 [227]. Table LII provides the overview of the problem.

$$F(\vec{x}) = \max_i \{|x_i|, 1 \leq i \leq n\}, \quad i = 1, 2, \ldots, n \tag{B.17}$$

where the max function returns the largest of its parameters.

Table LII. Schwefel 2.21 function overview

Function	Domain	argmin	min f(x)
Schwefel 2.21	$(-100, 100)^n$	0^n	0

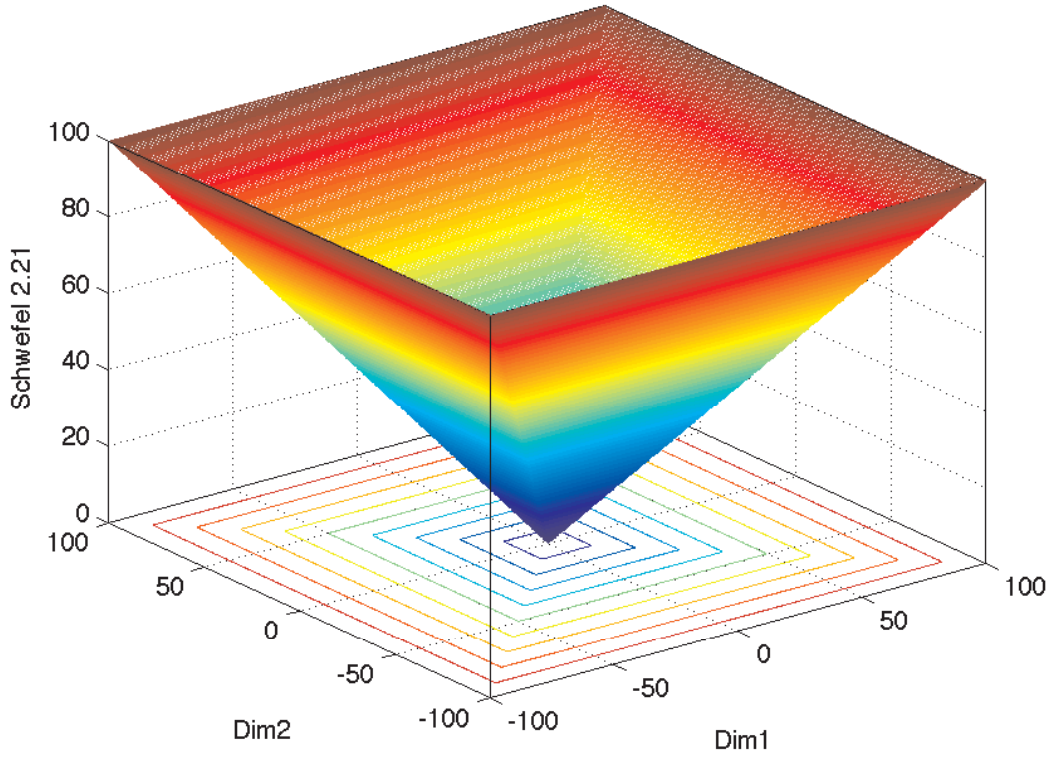


Figure 73. Two dimensional plot of the Schwefel 2.21 function.

B.2.12 Schwefel 2.22

Schwefel 2.22 function is given in Eq. B.18 and is plotted as a mesh contour plot in Fig. 74 [227]. Table LIII provides the overview of the problem.

$$F(\vec{x}) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i| \tag{B.18}$$

Table LIII. Schwefel 2.22 function overview

Function	Domain	argmin	min f(x)
Schwefel 2.22	$(-10, 10)^n$	0^n	0

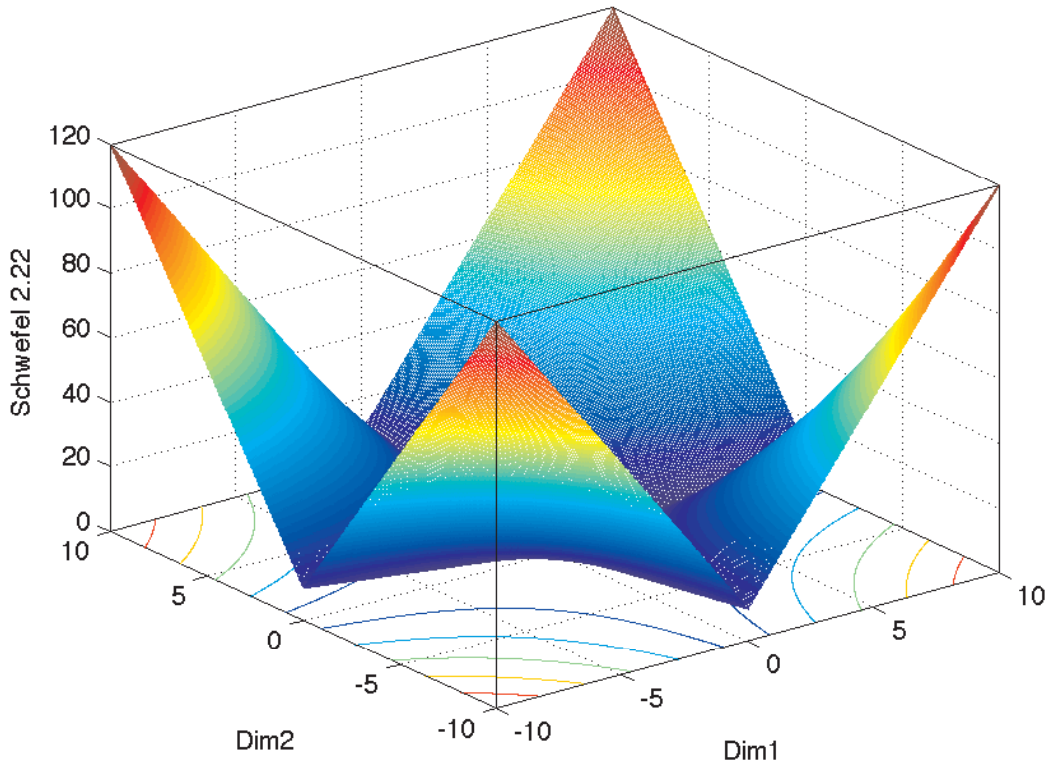


Figure 74. Two dimensional plot of the Schwefel 2.22 function.

B.2.13 Schwefel 2.26

Schwefel 2.26 function is given in Eq. B.19 and is plotted as a mesh contour plot in Fig. 75 [228]. Table LIV provides the overview of the problem.

$$F(\vec{x}) = -\sum_{i=1}^{n} x_i \sin\left(\sqrt{|x_i|}\right) \tag{B.19}$$

Table LIV. Schwefel 2.26 function overview

Function	Domain	argmin	min f(x)
Schwefel 2.26	$(-512, 512)^n$	420.9867^n	f(argmin)n

For the two-dimensional case, global minimum can be calculated as:

$$\min \mathrm{f}(x) = 2\, F([420.9867, 420.9867]) = -837.9658 \tag{B.20}$$

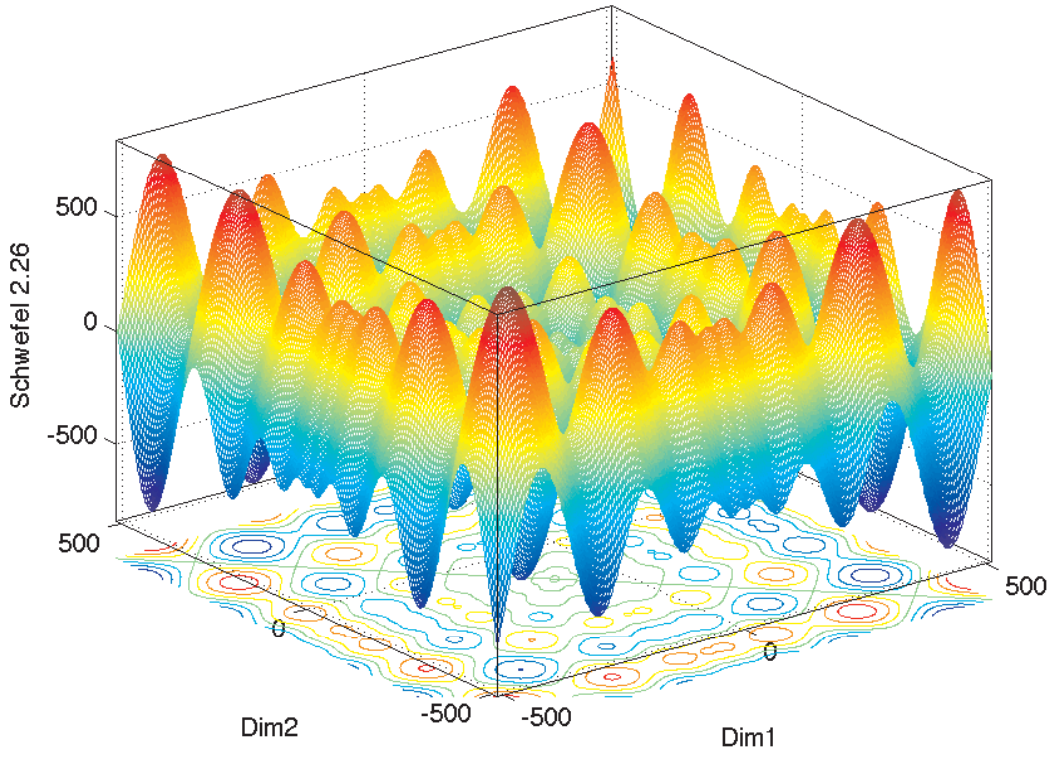


Figure 75. Two dimensional plot of the Schwefel 2.26 function.

B.2.14 Sphere

Sphere function is one of the earliest EA benchmarks [15]. Sphere function is given in Eq. B.21 and is plotted as a mesh contour plot in Fig. 76. Table LV provides the overview of the problem.

$$F(\vec{x}) = \sum_{i=1}^{n} x_i^2 \tag{B.21}$$

Table LV. Sphere function overview

Function	Domain	argmin	min f(x)
Sphere	$(-5.12, 5.12)^n$	0^n	0

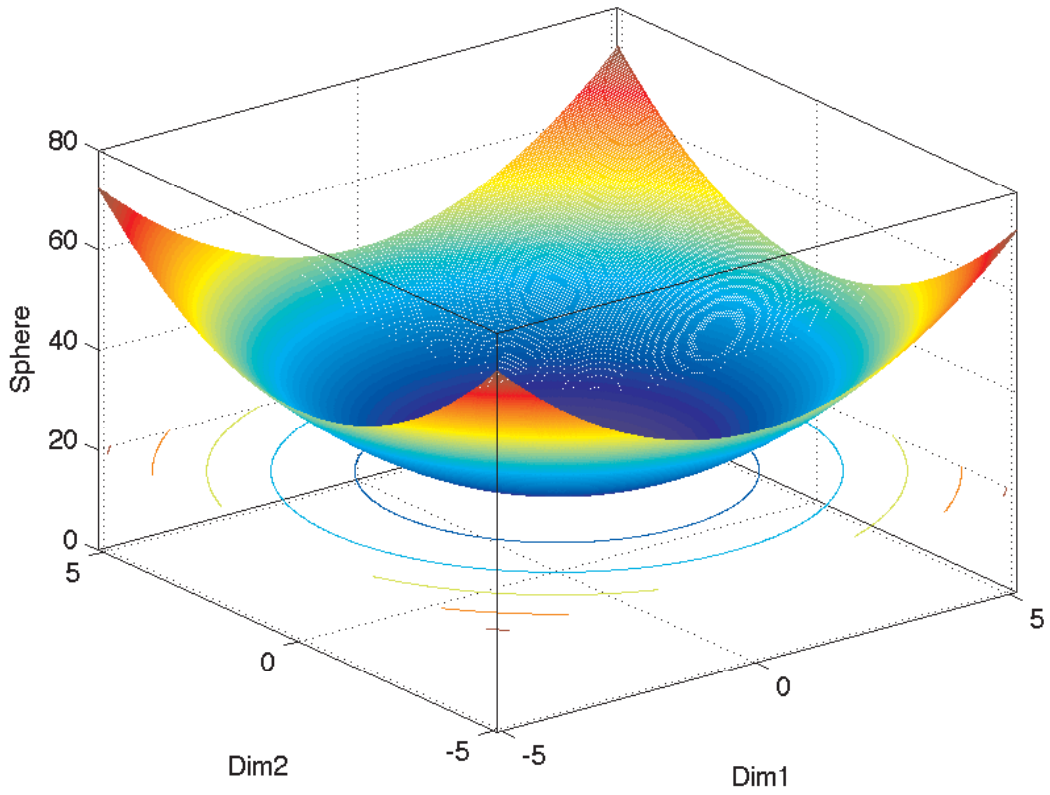


Figure 76. Two dimensional plot of the Sphere function.

B.2.15 Step

Step function is based on De Jong's F3 which was created to test EA performance on discontinuous functions [224]. Sphere function is given in Eq. B.22 and is plotted as a mesh contour plot in Fig. 77. Table LVI provides the overview of the problem.

$$F(\vec{x}) = \sum_{i=1}^{n} \text{floor}(x_i + 0.5)^2 \tag{B.22}$$

where floor function rounds towards minus infinity.

Table LVI. Step function overview

Function	Domain	argmin	min f(x)
Step	$(-100, 100)^n$	$(-0.5, 0.5)^n$	0

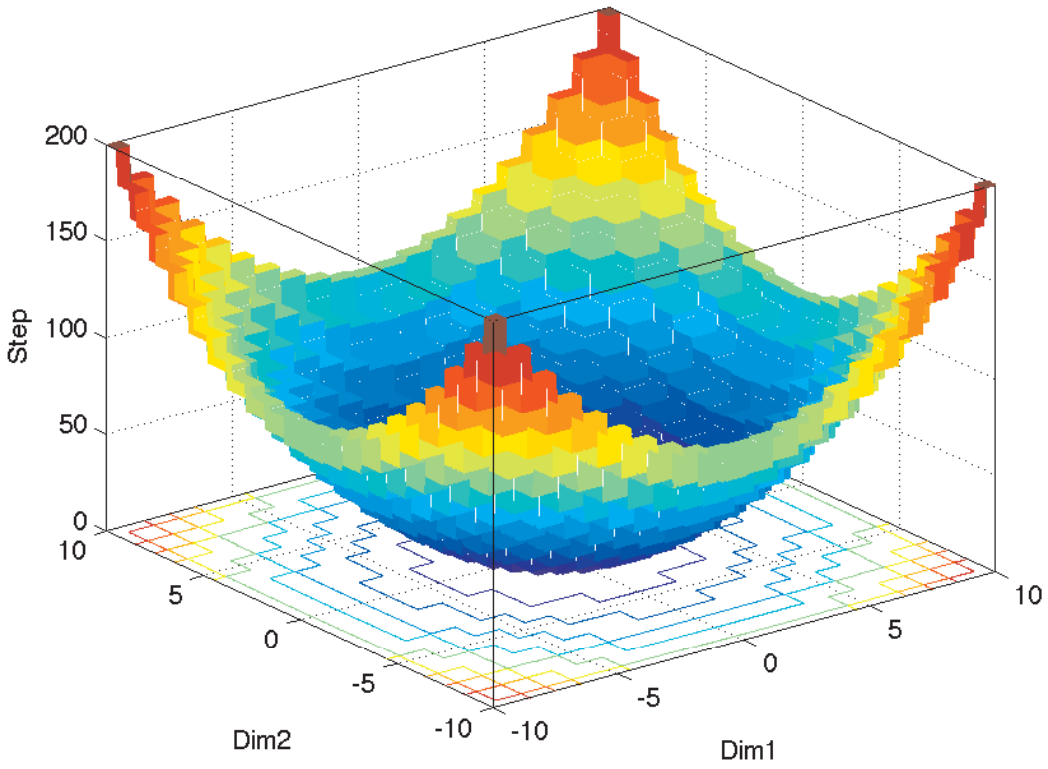


Figure 77. Two dimensional plot of the Step function. $Dim1, Dim2$ zoomed in to $[-10, 10]$ to illustrate the piecewise-constant steps of the funciton.

B.2.16 Zakharov

Zakharov function [141] is unimodal and flat. However, due to its relative uniform distribution of the solution, it is challenging to find the global optima located at the corner of the domain. Zakharov function is given in Eq. B.23 and is plotted as a mesh contour plot in Fig. 78. Table LVII provides the overview of the problem.

$$F(\vec{x}) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^4 \tag{B.23}$$

Table LVII. Zakharov function overview

Function	Domain	argmin	min f(x)
Zakharov	$(-5, 10)^n$	0^n	0

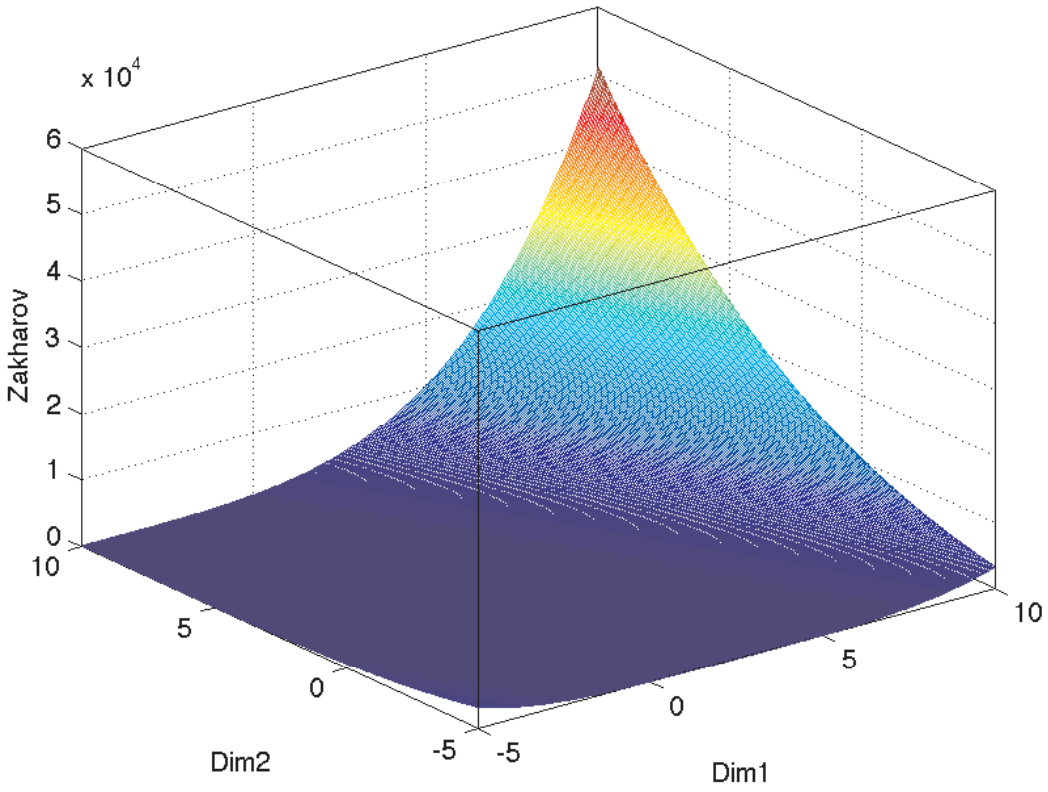


Figure 78. Two dimensional plot of the Zakharov function.

APPENDIX C

PUBLISHED, PRESENTED, AND SUBMITTED RESULTS FROM THIS RESEARCH

M. Ergezer, D. Simon, “Mathematical and Experimental Analyses of Oppositional Algorithms,” *IEEE Transactions on Cybernetics*, Accepted.

M. Ergezer, D. Simon, “Probabilistic Properties of Fitness-based Quasi-reflection to Accelerate the Performance of Evolutionary Algorithms," *Computers & Operations Research*, Submitted.

M. Ergezer, and I. Sikder, “Survey of Oppositional Algorithms,” *14th International Conference on Computer and Information Technology*, pp. 623–628, 2011.

M. Ergezer, and D. Simon, “Oppositional biogeography-based optimization for combinatorial problems," *IEEE Congress on Evolutionary Computation.* pp. 1496–1503, 2011.

D. Simon, M. Ergezer, D. Du, and R. Rarick, “Markov models for biogeography-based optimization," *IEEE Transactions on Systems, Man, and Cybernetics,*

Part B: Cybernetics, pp. 299–306, 2011.

D. Simon, R. Rarick, M. Ergezer, and D. Du, “Analytical and numerical comparisons of biogeography-based optimization and genetic algorithms," *Information Sciences*, vol. 181, pp. 1224–1248, 2011.

M. Ergezer, B. E. Abali, and D. Simon, “Biogeography-based Optimization Identifies Material Coefficients as an Inverse Problem," poster session presented at NSF CMMI Research and Innovation Conference, Atlanta, GA, January 2011.

D. Du, D. Simon, and M. Ergezer, “Biogeography-based optimization combined with evolutionary strategy and immigration refusal," *IEEE International Conference on Systems, Man and Cybernetics*, pp. 997–1002, 2009.

M. Ergezer, D. Simon, and D. Du, “Oppositional biogeography-based optimization," *IEEE International Conference on Systems, Man and Cybernetics*, pp. 1009–1014, 2009.

D. Simon, M. Ergezer, and D. Du, “Population distributions in biogeography-based optimization algorithms with elitism," *IEEE International Conference on Systems, Man and Cybernetics*, pp. 991–996, 2009.

BIBLIOGRAPHY

[1] F. Biscani, D. Izzo, and C. H. Yam, "A global optimisation toolbox for massively parallel engineering optimisation," *arXiv:1004.3824*, 2010.

[2] T. Back, D. Fogel, and Z. Michalewicz, *Evolutionary Computation: Basic algorithms and operators*. IOP Publishing Ltd. Bristol, UK, UK, 1999.

[3] N. Barricelli *et al.*, "Esempi numerici di processi di evoluzione," *Methodos*, vol. 6, pp. 45–68, 1954.

[4] J. Barker, "Simulation of genetic systems by automatic digital computers. I. Introduction," *Australian J. Biological Sciences*, vol. 10, pp. 484–491, 1958.

[5] L. Fogel, "Autonomous automata," *Industrial Research*, vol. 4, no. 2, pp. 14–19, 1962.

[6] L. Fogel, A. Owens, M. Walsh *et al.*, *Artificial intelligence through simulated evolution*. Wiley New York, 1966, vol. 26.

[7] J. Holland, "Outline for a logical theory of adaptive systems," *Journal of the ACM*, vol. 9, no. 3, pp. 297–314, 1962.

[8] J. H. Holland, *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[9] J. Holland, *Adaptation in Natural and Artificial Systems*. MIT Press, 1992.

[10] I. Rechenberg, "Cybernetic solution path of an experimental problem," *Library Translation*, vol. 1122, 1964.

[11] H. Schwefel, “Kybernetische evolution als strategie der experimentellen forschung in der stromungstechnik,” Thesis, Technische Universität Berlin, 1965.

[12] H. Beyer and H. Schwefel, “Evolution strategies–a comprehensive introduction,” *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.

[13] H. Schwefel, “Experimentelle optimierung einer zweiphasenduse,” *AEG Research Institute, Project MHD-Straustrahlrohr*, vol. 11034, p. 68, 1968.

[14] H. P. Schwefel and J. Klockgether, “Two phase nozzle and hollow core jet experiments,” *Proceedings of the Symposium on the Engineering Aspects of Magnetohydrodynamics*, 1970.

[15] I. Rechenberg, “Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution,” Dissertation, Technical University of Berlin, 1971.

[16] I. Rechenberg and M. Eigen, “Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution,” 1973.

[17] M. Dorigo, “Optimization, learning and natural algorithms,” Dissertation, Politecnico di Milano, Italy, 1992.

[18] J. Deneubourg, J. Pasteels, and J. Verhaeghe, “Probabilistic behaviour in ants: a strategy of errors?” *Journal of Theoretical Biology*, vol. 105, no. 2, pp. 259–271, 1983.

[19] S. Goss, S. Aron, J. Deneubourg, and J. Pasteels, “Self-organized shortcuts in the argentine ant,” *Naturwissenschaften*, vol. 76, no. 12, pp. 579–581, 1989.

[20] J. Kennedy, R. Eberhart *et al.*, “Particle swarm optimization,” in *Proceedings of IEEE International Conference on Neural Networks*, vol. 4. Perth, Australia, 1995, pp. 1942–1948.

[21] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 2002, pp. 39–43.

[22] R. Eberhart, Y. Shi, J. Kennedy, and E. Corporation, *Swarm intelligence*. Elsevier, 2001.

[23] C. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 1987, pp. 25–34.

[24] R. Storn and K. Price, "Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces," International Computer Science Institute Publications, Tech. Rep., 1995.

[25] ——, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[26] J. Andre, P. Siarry, and T. Dognon, "An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization," *Advances in Engineering Software*, vol. 32, pp. 49–60, 2001.

[27] A. Hrstka, O.; Kucerova, "Improvement of real coded genetic algorithm based on differential operators preventing premature convergence," *Advances in Engineering Software*, vol. 35, pp. 237–246, 2004.

[28] K. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series)*. Springer, 2005.

[29] M. Pedersen, "Good parameters for differential evolution," Hvass Laboratories, Tech. Rep. HL1002, 2010.

[30] N. Cramer, "A representation for the adaptive generation of simple sequential programs," in *Proceedings of the First International Conference on Genetic Algorithms*, vol. 183, 1985, p. 187.

[31] J. R. Koza, "Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems," Stanford University, Tech. Rep. STAN-CS-90-1314, 1990.

[32] J. Koza, "Evolving a computer program to generate random numbers using the genetic programming paradigm," in *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991, pp. 37–44.

[33] ——, *Genetic programming: On the programming of computers by means of natural selection*. MIT Press, 1992.

[34] J. Koza, F. Bennett III, D. Andre, M. Keane, and F. Dunlap, "Automated synthesis of analog electrical circuits by means of genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 2, pp. 109–128, 2002.

[35] J. Koza, F. Bennett III, D. Andre, and M. Keane, "Automated WYWIWYG design of both the topology and component values of electrical circuits using genetic programming," in *Proceedings of the First Annual Conference on Genetic Programming*, 1996, pp. 123–131.

[36] S. Kirkpatrick, "Optimization by simulated annealing: Quantitative studies," *Journal of Statistical Physics*, vol. 34, pp. 975–986, 1984.

[37] V. Černỳ, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," *Journal of Optimization Theory and Applications*, vol. 45, no. 1, pp. 41–51, 1985.

[38] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller *et al.*, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, p. 1087, 1953.

[39] S. Nahar, S. Sahni, and E. Shragowitz, "Simulated annealing and combinatorial optimization," in *Conference on Design Automation*, 1986, pp. 293–299.

[40] P. Van Laarhoven, E. Aarts, and J. Lenstra, "Job shop scheduling by simulated annealing," *Operations Research*, vol. 40, no. 1, pp. 113–125, 1992.

[41] C. Apolloni *et al.*, "Quantum stochastic optimization," *Stochastic Processes and their Applications*, vol. 33, no. 2, pp. 233–244, 1989.

[42] W. Wenzel and K. Hamacher, "Stochastic tunneling approach for global minimization of complex potential energy landscapes," *Physical Review Letters*, vol. 82, no. 15, pp. 3003–3007, 1999.

[43] F. Glover, C. McMillan, and B. Novick, "Interactive decision software and computer graphics for architectural and space planning," *Annals of Operations Research*, vol. 5, pp. 557–573, 1985.

[44] F. Glover, "Tabu search, Part I," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–207, 1989.

[45] ——, "Tabu search, Part II," *ORSA Journal on Computing*, vol. 2, no. 1, pp. 4–32, 1990.

[46] A. Hertz and D. Werra, "Using tabu search techniques for graph coloring," *Computing*, vol. 39, pp. 345–351, 1987.

[47] N. Zufferey, P. Amstutz, and P. Giaccari, "Graph colouring approaches for a satellite range scheduling problem," *Journal of Scheduling*, vol. 11, pp. 263–277, 2008.

[48] M. Levin, "Combinatorial optimization in system configuration design," *Automation and Remote Control*, vol. 70, pp. 519–561, 2009.

[49] BBC News, "Namib desert beetle inspires self-filling water bottle," Online, November 2012, retrieved from http://www.bbc.co.uk/news/technology-20465982 on November 2012.

[50] C. H. Young, "Aviation and medicine," *Bulletin of the Medical Library Association*, vol. 28, no. 3, p. 132, 1940.

[51] BBC News, "Biomimicry: Beaks on trains and flipper-like turbines," Online, October 2011, retrieved from http://www.bbc.co.uk/news/business-15480620 on February 2013.

[52] California Institute of Technology, "Bioinspired design and engineering," Online, July 2012, retrieved from http://www.gharib.caltech.edu/bioinspired-design/index.html on February 2013.

[53] George Washington University, "Center for biomimetics and bioinspired engineering," Online, September 2009, retrieved from http://cobre.seas.gwu.edu/ on February 2013.

[54] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biology*, vol. 5, no. 4, pp. 115–133, 1943.

[55] J. Farmer, N. Packard, and A. Perelson, "The immune system, adaptation and machine learning," *Physica D*, vol. 2, pp. 187–204, 1986.

[56] S. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Prentice hall, 2009.

[57] D. H. Wolpert and W. G. Macready, "No free lunch theorems for search," Santa Fe Institute, Tech. Rep. SFI-TR-95-02-010, 1995.

[58] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural Computation*, vol. 8, no. 7, pp. 1341–1390, 1996.

[59] ——, "On bias plus variance," *Neural Computation*, vol. 9, no. 6, pp. 1211–1243, 1997.

[60] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, April 1997.

[61] C. Igel and M. Toussaint, "A no-free-lunch theorem for non-uniform distributions of target functions," *Journal of Mathematical Modelling and Algorithms*, vol. 3, no. 4, pp. 313–322, 2004.

[62] T. English, "No more lunch: Analysis of sequential search," in *Congress on Evolutionary Computation*, vol. 1, 2004, pp. 227–234.

[63] J. Culberson, "On the futility of blind search: An algorithmic view of no free lunch," *Evolutionary Computation*, vol. 6, no. 2, pp. 109–127, 1998.

[64] N. Radcliffe and P. Surry, "Fundamental limitations on search algorithms: Evolutionary computing in perspective," *Computer Science Today*, pp. 275–291, 1995.

[65] S. Droste, T. Jansen, and I. Wegener, "Optimization with randomized search heuristics–the (A)NFL theorem, realistic scenarios, and difficult functions," *Theoretical Computer Science*, vol. 287, no. 1, pp. 131–144, 2002.

[66] S. Obayashi, D. Sasaki, Y. Takeguchi, and N. Hirose, "Multiobjective evolutionary computation for supersonic wing-shape optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 2, pp. 182–187, 2002.

[67] D. Sasaki, M. Morikawa, S. Obayashi, and K. Nakahashi, "Aerodynamic shape optimization of supersonic wings by adaptive range multiobjective genetic algorithms," in *Evolutionary Multi-Criterion Optimization*. Springer, 2001, pp. 639–652.

[68] W. W. Gibbs, "Programming with primordial ooze," *Scientific American*, pp. 48–50, 1996.

[69] C. L. Karr and L. M. Freeman, "Genetic-algorithm-based fuzzy control of spacecraft autonomous rendezvous," *Engineering Applications of Artificial Intelligence*, vol. 10, no. 3, pp. 293–300, 1997.

[70] R. Glen and A. Payne, "A genetic algorithm for the automated generation of molecules within constraints," *Journal of Computer-Aided Molecular Design*, vol. 9, no. 2, pp. 181–202, 1995.

[71] B. Lemley, "Machines that think." *Discover*, vol. 22, no. 1, pp. 74–79, 2001.

[72] K. De Jong, "Adaptive system design: a genetic approach," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 10, no. 9, pp. 566–574, 2007.

[73] K. J. Hunt, "Polynomial LQG and H_∞ controller synthesis: a genetic algorithm solution," in *Proceedings of the 31st IEEE Conference on Decision and Control*, vol. 4, 1992, pp. 3604–3609.

[74] J. Lin, H. Sheu, and S. Chao, "Lqg/ga design of active noise controllers for a collocated acoustic duct system," *Journal of Sound and Vibration*, vol. 228, no. 3, pp. 629–650, 1999.

[75] J. Herrmann, "A genetic algorithm for minimax optimization problems," in *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 2, 2002.

[76] A. Assion, T. Baumert, M. Bergt, T. Brixner, B. Kiefer, V. Seyfried, M. Strehle, and G. Gerber, "Control of chemical reactions by feedback-optimized phase-shaped femtosecond laser pulses," *Science*, vol. 282, no. 5390, p. 919, 1998.

[77] S. K. Karr *et al.*, "Fuzzy control of an exothermic chemical reaction using genetic algorithms," *Engineering Applications of Artificial Intelligence*, vol. 6, no. 6, pp. 575–582, 1993.

[78] Y. Davidor, *Genetic Algorithms and Robotics: A heuristic strategy for optimization*. World Scientific Pub Co Inc, 1991.

[79] K. Sugihara and J. Smith, "Genetic algorithms for adaptive motion planning of an autonomous mobile robot," in *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 2002, pp. 138–143.

[80] J. Ahuactzin, E. Talbi, P. Bessiere, and E. Mazer, "Using genetic algorithms for robot motion planning," *Geometric Reasoning for Perception and Action*, pp. 84–93, 1993.

[81] Y. Hu and S. Yang, "A knowledge based genetic algorithm for path planning of a mobile robot," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings*, vol. 5, 2004, pp. 4350–4355.

[82] C. Ergun and K. Hacioglu, "Multiuser detection using a genetic algorithm in CDMA communications systems," *IEEE Transactions on Communications*, vol. 48, no. 8, pp. 1374–1383, 2002.

[83] H. Chou, G. Premkumar, and C. Chu, "Genetic algorithms for communications network design-an empirical study of the factors that influence performance," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 3, pp. 236–249, 2002.

[84] M. Ericsson, M. Resende, and P. Pardalos, "A genetic algorithm for the weight setting problem in OSPF routing," *Journal of Combinatorial Optimization*, vol. 6, no. 3, pp. 299–333, 2002.

[85] K. Shin and Y. Lee, "A genetic algorithm application in bankruptcy prediction modeling," *Expert Systems with Applications*, vol. 23, no. 3, pp. 321–328, 2002.

[86] B. Back, T. Laitinen, and K. Sere, "Neural networks and genetic algorithms for bankruptcy predictions," *Expert Systems with Applications*, vol. 11, no. 4, pp. 407–413, 1996.

[87] S. Min, J. Lee, and I. Han, "Hybrid genetic algorithms and support vector machines for bankruptcy prediction," *Expert Systems with Applications*, vol. 31, no. 3, pp. 652–660, 2006.

[88] R. Kuo, C. Chen, and Y. Hwang, "An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network," *Fuzzy Sets and Systems*, vol. 118, no. 1, pp. 21–45, 2001.

[89] K. Kim and I. Han, "Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index," *Expert Systems with Applications*, vol. 19, no. 2, pp. 125–132, 2000.

[90] S. Chen, *Genetic algorithms and genetic programming in computational finance*. Kluwer Academic Pub, 2002.

[91] M. Sambridge and G. Drijkoningen, "Genetic algorithms in seismic waveform inversion," *Geophysical Journal International*, vol. 109, no. 2, pp. 323–342, 1992.

[92] P. Stoffa and M. K. Sen, "Nonlinear multiparameter optimization using genetic algorithms: inversion of plane-wave seismograms," *Institute for Geophysics*, vol. 56, p. 1794, 1991.

[93] P. Gerstoft, "Inversion of seismoacoustic data using genetic algorithms and a posteriori probability distributions," *Journal of the Acoustical Society of America*, vol. 95, no. 2, pp. 770–782, 1994.

[94] D. McKinney *et al.*, "Genetic algorithm solution of groundwater management models," *Water Resources Research*, vol. 30, no. 6, pp. 1897–1906, 1994.

[95] S. Cieniawski, J. Eheart, and S. Ranjithan, "Using genetic algorithms to solve a multiobjective groundwater monitoring problem," *Water Resources Research*, vol. 31, no. 2, pp. 399–409, 1995.

[96] J. Holland and J. Miller, "Artificial adaptive agents in economic theory," *The American Economic Review*, vol. 81, no. 2, pp. 365–370, 1991.

[97] R. Unger and J. Moult, "Genetic algorithms for protein folding simulations," *Journal of Molecular Biology*, vol. 231, no. 1, pp. 75–81, 1993.

[98] T. Dandekar and P. Argos, "Potential of genetic algorithms in protein folding and protein engineering simulations," *Protein Engineering Design and Selection*, vol. 5, no. 7, p. 637, 1992.

[99] J. Pedersen and J. Moult, "Protein folding simulations with genetic algorithms and a detailed molecular description," *Journal of Molecular Biology*, vol. 269, no. 2, pp. 240–259, 1997.

[100] O. Cordón and F. Herrera, "A proposal for improving the accuracy of linguistic modeling," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 3, pp. 335–344, 2002.

[101] R. MacArthur and E. Wilson, *The theory of island biogeography*. Princeton University Press, 2001.

[102] C. Darwin, "Geological Observations on the Volcanic Islands visited during the voyage of HMS Beagle, together with some brief notices on the geology of Australia and the Cape of Good Hope; being the second part of the Geology of the Voyage of the Beagle, under the command of Capt. Fitzroy, RN, during the years 1832 to 1836," *Quarterly Journal of the Geological Society*, vol. 1, no. 1, p. 556, 1845.

[103] T. Wesche, G. Goertler, and W. Hubert, "Modified habitat suitability index model for brown trout in southeastern Wyoming," *North American Journal of Fisheries Management*, vol. 7, pp. 1133–1136, 1987.

[104] A. Eiben, E. Aarts, and K. Van Hee, "Global convergence of genetic algorithms: A markov chain analysis," in *Parallel Problem Solving from Nature*, ser. Lecture Notes in Computer Science, H.-P. Schwefel and R. MÃd'nner, Eds. Springer Berlin / Heidelberg, 1991, vol. 496, pp. 3–12.

[105] D. B. Fogel, "Asymptotic convergence properties of genetic algorithms and evolutionary programming: Analysis and experiments," *Cybernetics and Systems: An International Journal*, vol. 25, pp. 389–407, 1994.

[106] D. Simon, M. Ergezer, and D. Du, "Markov models of biogeography-based optimization," *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, vol. 41, pp. 299–306, 2011.

[107] R. Rarick, D. Simon, F. Villaseca, and B. Vyakaranam, "Biogeography-based optimization and the solution of the power flow problem," in *IEEE International Conference on Systems, Man and Cybernetics*, 2009, pp. 1003–1008.

[108] A. Bhattacharya and P. Chattopadhyay, "Biogeography-based optimization for solution of optimal power flow problem," in *International Conference on Electrical Engineering/Electronics Computer Telecommunications and Information Technology*, 2010, pp. 435–439.

[109] P. K. Roy, S. P. Ghoshal, and S. S. Thakur, "Biogeography based optimization approach for optimal power flow problem considering valve loading effects," *International Journal of Recent Trends in Engineering*, vol. 3, no. 3, 2010.

[110] ——, "Multi-objective optimal power flow using biogeography-based optimization," *Electric Power Components and Systems*, vol. 38, no. 12, pp. 1406–1426, 2010.

[111] A. Bhattacharya and P. Chattopadhyay, "Biogeography-based optimization for different economic load dispatch problems," *IEEE Transactions on Power Systems*, vol. 25, no. 2, pp. 1064–1077, 2010.

[112] ——, "Economic dispatch solution using biogeography-based optimization," in *2009 Annual IEEE India Conference*, 2010, pp. 1–4.

[113] P. K. Roy, S. P. Ghoshal, and S. S. Thakur, "Combined economic and emission dispatch problems using biogeography-based optimization," *Electrical Engineering (Archiv fur Elektrotechnik)*, vol. 92, pp. 173–184, 2010.

[114] V. Panchal, P. Singh, N. Kaur, and H. Kundra, "Biogeography based satellite image classification," *International Journal of Computer Science and Information Security*, vol. 6, no. 2, pp. 269–274, 2009.

[115] V. Panchal, S. Goel, and M. Bhatnagar, "Biogeography based land cover feature extraction," in *World Congress on Nature & Biologically Inspired Computing*, 2010, pp. 1588–1591.

[116] R. Sumalatha and M. Subramanyam, "Region based coding of 3D magnetic resonance images for telemedicine applications," *International Journal of Computer Applications*, vol. 5, no. 12, pp. 46–51, 2010.

[117] U. Singh, H. Kumar, and T. Kamal, "Linear array synthesis using biogeography based optimization," *Progress In Electromagnetics Research*, vol. 11, pp. 25–36, 2010.

[118] M. R. Lohokare, S. S. Pattnaik, S. Devi, K. M. Bakwad, and J. G. Joshi, "Parameter calculation of rectangular microstrip antenna using biogeography-based optimization," in *Applied Electromagnetics Conference*, 2010, pp. 1–4.

[119] U. Singh, H. Singla, and T. Kamal, "Design of Yagi-Uda antenna using biogeography based optimization," *IEEE Transactions on Antennas and Propagation*, vol. 58, no. 10, pp. 3375–3379, October 2010.

[120] P. Lozovyy, G. Thomas, and D. Simon, "Biogeography-based optimization for robot controller tuning," in *Computational Modeling and Simulation of Intellect: Current State and Future Perspectives*, B. Igelnik, Ed. Hershey, PA: IGI Global, 2011, pp. 162–181.

[121] H. Kundra and M. Sood, "Cross-Country Path Finding using Hybrid approach of PSO and BBO," *International Journal*, vol. 7, 2010.

[122] M. Azab, "Global maximum power point tracking for partially shaded pv arrays using particle swarm optimisation," *International Journal of Renewable Energy Technology*, vol. 1, no. 2, pp. 211–235, 2009.

[123] D. Simon, "A probabilistic analysis of a simplified biogeography-based optimization algorithm," *Evolutionary Computation*, vol. 19, no. 2, pp. 167–188, 2011.

[124] D. Simon, M. Ergezer, and D. Du, "Population distributions in biogeography-based optimization algorithms with elitism," in *IEEE International Conference on Systems, Man and Cybernetics*, 2009, pp. 991–996.

[125] D. Du, D. Simon, and M. Ergezer, "Biogeography-based optimization combined with evolutionary strategy and immigration refusal," in *IEEE International Conference on Systems, Man and Cybernetics*, 2009, pp. 997–1002.

[126] W. Gong, Z. Cai, and C. Ling, "DE/BBO: a hybrid differential evolution with biogeography-based optimization for global numerical optimization," *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, pp. 1–21.

[127] N. Johal, S. Singh, and H. Kundra, "A hybrid FPAB/BBO Algorithm for Satellite Image Classification," *International Journal of Computer Applications*, vol. 6, no. 5, 2010.

[128] S. Kumar, P. Bhalla, and A. Singh, "Fuzzy rule base generation from numerical data using biogeography-based optimization," *Institution of Engineers J. of Electron. and Telecomm. Eng*, pp. 8–13, 2009.

[129] M. Ovreiu and D. Simon, "Biogeography-based optimization of neuro-fuzzy system parameters for diagnosis of cardiac disease," in *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. ACM, 2010, pp. 1235–1242.

[130] G. B. Andresen, M. D. Ashkezari, and M. Baquero-Ruiz, "Trapped antihydrogen," *Nature*, vol. 468, no. 7324, pp. 673–676, 12 2010. [Online]. Available: http://dx.doi.org/10.1038/nature09610

[131] J. Palmer, "Antimatter atom trapped for first time, say scientists," BBC News, BBC News, November 2010, retrieved from http://www.bbc.co.uk/news/science-environment-11773791 on November 16th, 2010.

[132] H. Tizhoosh, "Opposition-based learning: A new scheme for machine intelligence," in *Proceedings of International Conference on Computational Intelligence for Modelling Control and Automation*, vol. 1, 2005, pp. 695–701.

[133] ——, "Reinforcement learning based on actions and opposite actions," in *International Conference on Artificial Intelligence and Machine Learning*, 2005, pp. 94–98.

[134] ——, "Opposition-based reinforcement learning," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 10, no. 4, pp. 578–585, 2006.

[135] M. Ventresca and H. Tizhoosh, "Improving the convergence of backpropagation by opposite transfer functions," in *IEEE International Joint Conference on Neural Networks*, 2006, pp. 9527–9534.

[136] ——, "Opposite transfer functions and backpropagation through time," in *IEEE Symposium on Foundations of Computational Intelligence*, 2007, pp. 570–577.

[137] M. Shokri, H. Tizhoosh, and M. Kamel, "Opposition-based q (lambda) with non-markovian update," in *Proc. IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 2007, pp. 288–295.

[138] M. Rashid and A. Baig, "Improved opposition-based PSO for feedforward neural network training," in *International Conference on Information Science and Applications*, 2010, pp. 1–6.

[139] H. Tizhoosh and F. Sahba, "Quasi-global oppositional fuzzy thresholding," in *IEEE International Conference on Fuzzy Systems*, 2009, pp. 1346–1351.

[140] H. Tizhoosh, "Opposite fuzzy sets with applications in image processing," in *Proceedings of International Fuzzy Systems Association World Congress*, Lisbon, Postugal, 2009, pp. 36–41.

[141] S. Rahnamayan, "Opposition-based differential evolution," Dissertation, University of Waterloo, 2007.

[142] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 64–79, 2008.

[143] ——, "Quasi-oppositional differential evolution," in *IEEE Congress on Evolutionary Computation*, 2007, pp. 2229–2236.

[144] S. Rahnamayan and G. G. Wang, "Solving large scale optimization problems by opposition-based differential evolution (ODE)," *WSEAS Transactions on Computers*, vol. 7, pp. 1792–1804, October 2008.

[145] H. Wang, Y. Liu, S. Zeng, H. Li, and C. Li, "Opposition-based particle swarm algorithm with cauchy mutation," in *IEEE Congress on Evolutionary Computation*, 2007, pp. 4750–4756.

[146] F. Shahzad, A. Baig, S. Masood, M. Kamran, and N. Naveed, "Opposition-Based Particle Swarm Optimization with Velocity Clamping," *Advances in Computational Intelligence*, pp. 339–348, 2009.

[147] J. Tang and X. Zhao, "An enhanced opposition-based particle swarm optimization," in *WRI Global Congress on Intelligent Systems*, vol. 1, 2009, pp. 149–153.

[148] Y. Chi and G. Cai, "Particle swarm optimization with opposition-based disturbance," in *International Asia Conference on Informatics in Control, Automation and Robotics*, vol. 2, 2010, pp. 223–226.

[149] A. R. Malisia, "Investigating the application of opposition-based ideas to ant algorithms," Thesis, University of Waterloo, 2007.

[150] A. Malisia, *Improving the Exploration Ability of Ant-Based Algorithms*, H. Tizhoosh and M. Ventresca, Eds. Springer, 2008.

[151] M. Ventresca and H. Tizhoosh, "Simulated annealing with opposite neighbors," in *IEEE Symposium on Foundations of Computational Intelligence*, 2007, pp. 186–192.

[152] F. Khalvati, H. Tizhoosh, and M. Aagaard, "Opposition-based window memoization for morphological algorithms," in *IEEE Symposium on Foundations of Computational Intelligence*, 2007, pp. 425–430.

[153] S. Rahnamayan and H. R. Tizhoosh, "Image thresholding using micro opposition-based Differential Evolution (Micro-ODE)," in *IEEE Congress on Evolutionary Computation*, 2008, pp. 1409–1416.

[154] B. Subudhi and D. Jena, "Nonlinear system identification using opposition based learning differential evolution and neural network techniques," *IEEE Journal of Intelligent Cybernetic Systems*, vol. 1, pp. 1–13, 2009.

[155] ——, "A differential evolution based neural network approach to nonlinear system identification," *Applied Soft Computing*, vol. 11, pp. 861 – 871, 2011.

[156] H. Tizhoosh, M. Ventresca, and S. Rahnamayan, *Opposition-Based Computing*, H. Tizhoosh and M. Ventresca, Eds. Springer, 2008.

[157] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[158] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55 –66, feb. 2011.

[159] H. Ma and D. Simon, "Blended biogeography-based optimization for constrained optimization," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 3, pp. 517–525, 2011.

[160] H. Muhlenbein and D. Schlierkamp-Voosen, "Predictive models for the breeder genetic algorithm i. continuous parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 25–49, 1993.

[161] M. Ergezer, D. Simon, and D. Du, "Oppositional biogeography-based optimization," in *IEEE International Conference on Systems, Man and Cybernetics*, 2009, pp. 1009–1014.

[162] J. Bernoulli, *Ars conjectandi [The art of conjecture]*. Basel, 1713.

[163] P. de Laplace, *Théorie analytique des probabilités*. Courcier, 1820.

[164] A. Lovejoy and P. Stanlis, *The great chain of being: A study of the history of an idea*. Transaction Publishers, 2009.

[165] A. Papoulis and S. Pillai, *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill New York, 1965.

[166] H. Ma, "An analysis of the equilibrium of migration models for biogeography-based optimization," *Information Sciences*, vol. 180, no. 18, pp. 3444 – 3464, 2010.

[167] W. Dembski and R. Marks, "Conservation of information in search: measuring the cost of success," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 39, no. 5, pp. 1051–1061, 2009.

[168] E. Mezura-Montes and C. Coello, "A simple multimembered evolution strategy to solve constrained optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 1, pp. 1–17, Feb. 2005.

[169] C. Grosan, A. Abraham, M. Chis, and T. Chang, "Evolutionary elementary cooperative strategy for global optimization," in *Knowledge-Based Intelligent Information and Engineering Systems*, ser. Lecture Notes in Computer Science, B. Gabrys, R. J. Howlett, and L. C. Jain, Eds., vol. 4253. Springer, 2006, pp. 677–683.

[170] A. Hedar and M. Fukushima, "Minimizing multimodal functions by simplex coding genetic algorithm," *Optimization Methods and Software*, vol. 18, pp. 265 – 282, 2003.

[171] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on nu-

merical benchmark problems," in *IEEE Congress on Evolutionary Computation*, vol. 2, 2004, pp. 1980–1987.

[172] D. Simon, "Biogeography-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, 2008.

[173] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[174] A. J. Keane, "Experiences with optimizers in structural design," in *Proceedings of the Conference on Adaptive Computing in Engineering Design and Control*, vol. 94, 1994, pp. 14–27.

[175] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs (3rd ed.)*. London, UK: Springer-Verlag, 1996.

[176] H. Zhang and J. L., "Adaptive evolutionary programming based on reinforcement learning," *Information Sciences*, vol. 178, no. 4, pp. 971 – 984, 2008.

[177] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 107–125, 2008.

[178] S. He, Q. Wu, J. Wen, J. Saunders, and R. Paton, "A particle swarm optimizer with passive congregation," *Biosystems*, vol. 78, no. 1-3, pp. 135 – 147, 2004.

[179] Z. Tu and Y. Lu, "A robust stochastic genetic algorithm (StGA) for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 5, pp. 456–470, Oct. 2004.

[180] F. Aluffi-Pentini, V. Parisi, and F. Zirilli, "Global optimization and stochastic differential equations," *Journal of Optimization Theory and Applications*, vol. 47, no. 1, pp. 1–16, 1985.

[181] T. Vinko and D. Izzo, "Global optimisation heuristics and test problems for preliminary spacecraft trajectory design," European Space Agency, the Advanced Concepts Team, ACT technical report, Tech. Rep. GO-HTPPSTD, 2008.

[182] D. Izzo, *Global optimisation and space pruning for spacecraft trajectory design*, B. A. Conway, Ed. Cambridge University Press, 2009.

[183] S. Das and P. N. Suganthan, "Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems," IEEE Computational Intelligence Society, Tech. Rep., 2010.

[184] D. Du, "Biogeography-based optimization: Synergies with evolutionary strategies, immigration refusal and Kalman filters," Thesis, Cleveland State University, 2009.

[185] H. Mo and L. Xu, "Biogeography migration algorithm for traveling salesman problem," *Advances in Swarm Intelligence*, pp. 405–414, 2010.

[186] Y. Song, M. Liu, and Z. Wang, "Biogeography-based optimization for the traveling salesman problems," in *2010 Third International Joint Conference on Computational Science and Optimization*, 2010, pp. 295–299.

[187] G. Tao and Z. Michalewicz, "Inver-over operator for the TSP," in *Parallel Problem Solving from Nature*. Springer, 1998, p. 803.

[188] F. Leighton, "A graph coloring algorithm for large scheduling problems," *Journal of Research of the National Bureau of Standards*, vol. 84, no. 6, pp. 489–503, 1979.

[189] G. Chaitin, M. Auslander, A. Chandra, J. Cocke, M. Hopkins, and P. Markstein, "Register allocation via coloring," *Computer Languages*, vol. 6, no. 1, pp. 47–57, 1981.

[190] S. Even, O. Goldreich, S. Moran, and P. Tong, "On the NP-completeness of certain network testing problems," *Networks*, vol. 14, no. 1, pp. 1–24, 1984.

[191] M. Carter, "A survey of practical applications of examination timetabling algorithms," *Operations Research*, vol. 34, no. 2, pp. 193–202, 1986.

[192] D. Johnson, C. Aragon, L. McGeoch, and C. Schevon, "Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning," *Operations Research*, pp. 378–406, 1991.

[193] P. Galinier and J. Hao, "Hybrid evolutionary algorithms for graph coloring," *Journal of Combinatorial Optimization*, vol. 3, no. 4, pp. 379–397, 1999.

[194] D. Costa and A. Hertz, "Ants can colour graphs," *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 295–305, 1997.

[195] M. Trick. (1998, January) Graph coloring instances. Retrieved from http://mat.gsia.cmu.edu/COLOR/instances.html on October 31th, 2010. [Online]. Available: {http://mat.gsia.cmu.edu/COLOR/instances.html}

[196] F. Leighton, "A graph coloring algorithm for large scheduling problems," *Journal of Research of the National Bureau of Standards*, vol. 84, no. 6, pp. 489–503, 1979.

[197] D. Knuth, *The Stanford GraphBase: a platform for combinatorial computing*. ACM Press, 1993.

[198] J. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48–50, 1956.

[199] N. Christofides, A. Mingozzi, and P. Toth, "The vehicle routing problem," *Revue Française d'Informatique et de Recherche Opérationnelle*, vol. 10, no. 2, pp. 55–70, 1976.

[200] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 3, pp. 345–358, 1992.

[201] P. Toth and D. Vigo, *The vehicle routing problem*. Society for Industrial Mathematics, 2002.

[202] V. Magirou and A. Nicolitsas, "The efficient drilling of printed circuit boards," *Interfaces*, vol. 16, no. 4, pp. 13–23, 1986.

[203] G. C. Onwubolu and M. Clerc, "Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization," *International Journal of Production Research*, vol. 42, no. 3, pp. 473–491, 2004.

[204] G. Reinelt, "TSPLIB–A traveling salesman problem library," *INFORMS Journal on Computing*, vol. 3, no. 4, p. 376, 1991.

[205] J. Gregory, "Nonlinear programming faq, usenet sci. answers," *Retrieved from ftp://rtfm. mit. edu/pub/usenet/sci. answers/nonlinear-programming-faq on August 13th, 2010*, 1995.

[206] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 1, pp. 1–32, 1996.

[207] R. J. Whittaker and J. M. Fernández-Palacios, *Island Biogeography: Ecology, Evolution, and Conservation*, 2nd ed. Oxford University Press, USA, 2 2007.

[208] D. Quammen, *The Song of the Dodo: Island Biogeography in an Age of Extinction*. Scribner, 1997.

[209] P. Darlington, *Zoogeography: the geographical distribution of animals*. New York: John Wiley & Sons, 1957.

[210] J. Diamond, "Biogeographic kinetics: estimation of relaxation times for avifaunas of southwest pacific islands," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 69, no. 11, p. 3199, 1972.

[211] M. A. Potter and K. De Jong, "A cooperative coevolutionary approach to function optimization," *Parallel Problem Solving from Nature*, vol. 866, pp. 249–257, 1994.

[212] ——, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evolutionary Computation*, vol. 8, no. 1, pp. 1–29, 2000, (c) 2000 by the Massachusetts Institute of Technology.

[213] J. Abell and D. Du, "A framework for multi-objective, biogeography based optimization of complex system families," in *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*. Forth Worth, TX: AIAA, September 2010.

[214] T. Tan and J. Teo, "Evolving Opposition-Based Pareto Solutions: Multiobjective Optimization Using Competitive Coevolution," in *Oppositional Concepts in Computational Intelligence*, ser. Studies in Computational Intelligence, H. Tizhoosh and M. Ventresca, Eds. Springer Berlin / Heidelberg, 2008, vol. 155, pp. 161–206.

[215] D. Izzo, M. Ruciński, and F. Biscani, "The generalized island model," *Parallel Architectures and Bioinspired Algorithms*, pp. 151–169, 2012.

[216] T. H. Rowan, "Functional stability analysis of numerical algorithms," Dissertation, The University of Tennessee, Knoxville, 1990.

[217] D. Izzo and F. Biscani, "Python parallel global multiobjective optimizer," Online, 2011, retrieved from http://pagmo.sourceforge.net/pygmo/index.html on February 14th, 2013.

[218] J. Shekel, "Test functions for multimodal search techniques," in *Fifth Annual Princeton Conference on Information Science and Systems*, 1971.

[219] E. Easom, "A survey of global optimization techniques," Thesis, University of Louisville, 1990.

[220] D. H. Ackley, *A connectionist machine for genetic hillclimbing*. Norwell, MA, USA: Kluwer Academic Publishers, 1987.

[221] T. Bäck, *Evolutionary algorithms in theory and practice*. Oxford University Press, 1996.

[222] R. Fletcher and M. Powell, "A rapidly convergent descent method for minimization," *The Computer Journal*, vol. 6, no. 2, p. 163, 1963.

[223] A. Torn and A. Pilinskas, "Global optimization," in *Lecture Notes in Computer Science*, vol. 350. Springer-Verlag, 1989, berlin Heidelberg.

[224] K. De Jong, "Analysis of the behavior of a class of genetic adaptive systems," Dissertation, University of Michigan, 1975.

[225] L. A. Rastrigin, "Extremal control systems," *Theoretical foundations of engineering cybernetics series*, 1974.

[226] H. H. Rosenbrock, "An Automatic Method for Finding the Greatest or Least Value of a Function," *The Computer Journal*, vol. 3, no. 3, pp. 175–184, 1960. [Online]. Available: http://comjnl.oxfordjournals.org/content/3/3/175.abstract

[227] H. Schwefel, *Evolution and Optimum Seeking: The Sixth Generation*. John Wiley & Sons, Inc. New York, NY, USA, 1993.

[228] ——, *Numerical optimization of computer models*. John Wiley & Sons, Inc. New York, NY, USA, 1981.