2

Opposition-Based Computing

H.R. Tizhoosh[1], Mario Ventresca[1], and Shahryar Rahnamayan[2]

[1] Department of Systems Design Engineering, University of Waterloo, Canada
{tizhoosh,mventres}@pami.uwaterloo.ca

[2] Faculty of Engineering and Applied Science, University of Ontario Institute of Technology (UOIT), Canada
Shahryar.Rahnamayan@uoit.ca

Summary. Diverse forms of opposition are already existent virtually everywhere around us but the nature and significance of oppositeness is well understood only in specific contexts within the fields of philosophy, linguistics, psychology, logic and physics. The interplay between entities and opposite entities is apparently fundamental for maintaining universal balance. However, it seems that there is a gap regarding oppositional thinking in engineering, mathematics and computer science. Although many opposition-based techniques exist in these fields, the oppositional properties they employ are not usually directly studied. A better understanding of opposition could potentially establish new search, reasoning, optimization and learning schemes with a wide range of applications. For instance, improving convergence rate for hyperdimensional problems could be improved through the use of oppositional strategies.

A large number of problems in engineering and science cannot be approached with conventional schemes and are generally handled with intelligent techniques such as evolutionary, neural, reinforcing and swarm-based techniques. These methodologies, however, suffer from high computational costs. In this work, the outlines of opposition-based computing, a proposed framework for computational intelligence, will be introduced. The underlying idea is simultaneous consideration of guess and opposite guess, estimate and opposite estimate, viewpoint and opposite viewpoint and so on in order to make better decisions in a shorter time, something that all aforementioned techniques could benefit from. The goal is to better understand the role of opposition within a computational intelligence context with the intention of improving existing or developing more powerful and robust approaches to handle complex problems.

Due to its diversity, it is difficult, if not impossible, to uniquely and universally define the nature and the scope of what we call *opposition*. However, there is no doubt that we need a mathematical formalism if we are going to, at least to some degree, exploit the oppositional relationships in real-world systems. Hence, we attempt to establish a generic framework for computing with opposites in this chapter, a framework which may not mathematically capture the very essence of oppositeness in all systems accurately but it will be, as we hope, a point of departure for moving toward opposition-based computing. The ideas put forward in this chapter may not be shared by all those who work on some aspect of opposition (and usually without labeling it opposition), nevertheless, we believe that this framework can be useful for understanding and employing opposition.

2.1 Introduction

The idea of explicitly and consiously using opposition in machine intelligence was introduced in [21]. The paper contained very basic definitions and three application examples providing preliminary results for genetic algorithms, reinforcement learning and neural networks. The main scheme proposed was to not only to look at chromosomes, actions and weights but simultaneously consider anti-chromosomes, counter-actions and opposite weights. The paper argued that by considering these opposites and by simultaneous analysis of quantities and their opposite quantities we can accelerate the task in focus by increasing the chances of discovering the basins of attraction for optimal (or high quality) solutions.

In this chapter, the framework of opposition-based computing (OBC) will be provided, new definitions will be established and a rudimentary classification of oppositional concepts in computational intelligence will be provided. In section 2.2 the nature of opposition will be briefly discussed. Section 2.3 provides introductory definitions which we propose to be employed to understand opposition. We also show an example based on low-discrepancy sequences, of the type of improvements opposition can potentially deliver. In Section 2.4 we formally describe opposition-based computing using our definitions from Section 2.3. In section 2.5 a rough guideline or classification of OBC algorithms is provided. Section 2.6 summarizes the work with some conclusions.

2.2 What Is Opposition?

According to the American Heritage Dictionary *opposite* is defined as, "being the other of two complementary or mutually exclusive things" and *oppositional* as "placement opposite to or in contrast with another" [1]. Generalizing these definitions we can infer that opposites are two complementary elements of some concept or class of objects/numbers/abstractions, and the relationship which defines their complementarity is the notion of opposition[1]. More formally, two elements $c_1, c_2 \in \mathcal{C}$ are considered opposite relative to some mapping $\varphi \colon \mathcal{C} \rightarrow \mathcal{C}$, where φ is a one-to-one function and $\mathcal{C}$ is some non-empty set of concepts.

The class $\mathcal{C}$ can represent any form of concept, whether it be a physical entity (i.e. a state or estimate) or more abstract notion (i.e a hypothesis or viewpoint). The mapping between the opposite elements is also very robust, indeed the very notion of opposite is defined relative to this mapping, which can essentially be any function determining which entities are opposites. Therefore, changing φ consequently changes how the concepts are percieved.

Consider the following two simple situations, letting $\mathcal{C}$ be a continuous interval between $[0, 1]$ and $X \in \mathcal{C}$: .

1. Let $\varphi(X) = (X + 0.5)\%1$. The modulo operator guarantees a constant distance of 0.5 between $X = x$ and its opposite $\breve{x} = \varphi(x)$[2].
2. Let $\varphi(X) = 1 - X$. Under this definition as $X = x \rightarrow 0.5$, $\breve{x} \rightarrow 0.5$ (the center of the search interval). Therefore φ is less concerned with physical distance within the natural ordering of $\mathcal{C}$ and more related to similarity.

These situations highlight an important facet of oppositional thinking. Namely, the natural ordering of concepts from class $\mathcal{C}$ is not necesarily the only manner in which

[1] See the introduction, Chapter 1, for a more general motivation on opposition.
[2] We denote the opposite of x with $\breve{x}$.

opposites can be defined. More often than not the problem definition may indicate or hint towards something more appropriate. In fact, the second scenario contains a paradox in that if $x = 0.5$ then $\breve{x} = 0.5$, leaving a debate as to whether a concept can be its own opposite or whether it is possible for the concept to not have an opposite at all[3]. This situation must also be appropriately addressed through the definition of φ.

The above examples were restricted to a single dimensional problem. A logical next step is to examine how oppositional thinking changes as the problem dimensionality increases. Let $C =< c_1, c_2, \ldots, c_n >$ be an arbitrary concept in $\mathcal{C}$ and c_i be the i^{th} dimensional component of the n-dimensional system. We can then define $\Phi(C) =< \varphi^1(c_1), \varphi^2(c_2), \ldots, \varphi^n(c_n) >$ where φ^i defines opposition solely for dimension i and Φ is the resultant mapping over all n single dimension mappings. That is, the opposite of a multi-dimensional system is defined with respect to the opposite of each component. Also, as with the single dimension situation, Φ must be a one-to-one function.

Let S be an arbitrarily selected subset of integers from $\{1, 2, \ldots, n\}$, corresponding to one of $\{\varphi^1, \varphi^2, \ldots, \varphi^n\}$. In total there exist 2^n possible subsets, of which only one has all n integers (denoted S^*). Then, for every S we can calculate

$$\tau(S) = \frac{|S|}{|S^*|} = \frac{|S|}{n}, \tag{2.1}$$

to represent the *degree of opposition* between the concepts represented by S and the true opposite S^*, where $|\cdot|$ is the cardinality of the respective set. It is important to notice that in this case each component is given equal weighting, meaning that there does not exist a component which contributes more "oppisitional tendency" than any other (i.e. each dimension of the problem is equally important). However, allowing unequal weighting allows for greater flexibility and freedom when employing opposition in computational intelligence (as will be shown throughout this book).

By introducing the idea of varying degrees of oppositeness we also can ask whether opposition is a static or dynamic concept. This refers to the redefinition of opposition as undesirable elements of $\mathcal{C}$ are excluded through a search, optimization or learning process (i.e. $|\mathcal{C}|$ can vary at each iteration). We denote a system with a dynamic concept of opposition as Φ_t, indicating there exists a time component to the mapping (where each component is also time varying and denoted by φ^i_t).

Exploiting *symmetry* is in many cases a good example for incorporating opposition within learning and search operations. In its essence, considering oppositional relationships is nothing but integration of *a-priori knowledge*. There exist a large number of works on using a-priori or expert knowledge in computational intelligence, why then, one may ask, should we concern ourselves with opposition since we are integrating a-priori knowledge anyway? The knowledge of oppositeness in context of a given problem is a very special type of a-priori knowledge, which, if we comprehend the oppositional relationships between quantities and abstractions, its processing bears virtually no uncertainty. If, for instance, we know that the quantity/abstaction X has a high evaluation in context of the problem in focus, then its opposite $\breve{X}$ can be confidently dismissed as delivering a low evaluation. By recognizing oppositional entities we

[3] This is a very old debate in logic and philosophy which everybody may know via the example "is the glass half full or half empty?".

acquire a new type of a-priori knowledge that, if processed as general, non-oppositional knowledge, its contributions to problem solving remain widely unexploited.

A final major issue to address is whether opposite concepts are required to have an antagonistic nature (mutually exclusive), or can they interact in a constructive manner (balance maintenance). Is the system constantly deciding between which of two concepts is most desirable or are these two concepts used together to produce a more desirable system? Furthermore, is this interaction a constant or dynamic phenomena? That is, can competitive and cooperative relationships exist between opposites at different times? Most likely antipodal answers to these questions are true for different problem settings.

Most of the ideas behind these notions can be found scattered in various research areas. Chapters 3 and 4 further discuss some of the statistical and logical motivations and interpretations of opposition, respectively.

2.3 Computing with Opposites

In this section we attempt to establish a generic framework for computing with opposites which will be used in subsequent sections when discussing opposition-based computing. We have left the definitions abstract to provide flexibility and robustness in their practical application. First we provide important definitions followed by a simple example of using the outlined concepts.

2.3.1 Formal Framework

In this subsection we will provide abstract definitions pertaining to type-I and type-II opposition. For each definition we discuss its implications and provide examples where appropriate.

Definition 1 (Type-I Opposition Mapping). *Any one-to-one mapping function, $\Phi\colon \mathcal{C} \rightarrow \mathcal{C}$, which defines an oppositional relationship between two unique*[4] *elements C_1, C_2 of concept class $\mathcal{C}$ is a type-I opposition mapping. Furthermore, the relationship is symmetric in that if $\Phi(C_1) = C_2$, then $\Phi(C_2) = C_1$.*

This mapping is understood to define the opposite nature between C_1, C_2 with respect to the problem and/or goal of employing the relationship. In some cases this may be directly related to solving the problem at at hand (i.e. with respect to minimizing an evaluation function). Another possible use is to implcitly aid in solving the problem, for example to provide a more diverse range of solutions.

Definition 2 (Type-I Opposition). *Let $C \in \mathcal{C}$ be a concept in n-dimensional space and let $\Phi\colon \mathcal{C} \rightarrow \mathcal{C}$ be an opposition mapping function. Then, the type-I opposite concept is determined by $\breve{C} = \Phi(C)$.*

[4] The assumption is that an element cannot be its own opposite, although it is technically possible to relax this condition.

For convenience, we often omit the explicit reliance on Φ. For example, if $\breve{x}_i = \Phi(x_i) = a_i + b_i - x_i$, we simply write this as $\breve{x}_i = a_i + b_i - x_i$. Analogously, we can also denote type-I opposites as $\breve{x}_i = -x_i$ or $\breve{x}_i = 1 - x_i$ depending on the range of the universe of discourse.

Type-I opposites capture *linear opposition* and are simple and easy to calculate. For instance, if we regard *Age* as an integer set $A = \{0, 1, \dots, 100\}$ and define $\Phi(age) = 100 - age$, then the opposite of a 10 year old child is an 90 year old adult, where age is some random element of A (the sex could provide additional oppositional attributes).

Definition 3 (Type-I Super-Opposition). *Let C be an n-dimensional concept set $\mathcal{C}$. Then, all points $\breve{C}^s$ are type-I super-opposite of C when $d(\breve{C}^s, C) > d(\breve{C}, C)$ for some distance function $d(\cdot,\cdot)$.*

Type-I super-opposition plays a role in systems where the opposite is defined such that $|\breve{C}^s| \geq 1$ corresponding to the existence of at least one point further in distance from concept C, but not with respect to logical meaning. Consider $\Phi(X) = -X$ where $-\infty < X < \infty$. For $X = x = 0.1$, $\breve{x} = -0.1$, but there exists an infinite number of values further from x than its opposite $\breve{x}$. These extreme (or super) points are in super-opposition to C. Note that $|\breve{C}^s| \geq 0$ for any Φ (i.e. super-opposition is not required to exist). For instance, if we regard the *age* as an integer set $A = \{0, 1, \dots, 100\}$, then the super-opposites of a 10 years old are all those members with an age above 90 years. On the other hand, consider an adult of exactly 100 years; his/her opposite is an unborn child (exactly equal to age 0). So in this case when $age = 100$ there is no super-opposition (i.e. $|\breve{C}^s| = 0$).

As another example of super-opposition, let $\mathcal{C} = \mathbb{R}$ and use $\breve{x} = a + b - x$ as the relationship between opposites. Then for $x \in [a, b]$

$$\breve{x}^s \in \begin{cases} [a, \breve{x}) & \text{for} \quad x > (a+b)/2 \\ [a, b] - \{x\} & \text{for} \quad x = (a+b)/2 \\ (\breve{x}, b] & \text{for} \quad x < (a+b)/2 \end{cases} \tag{2.2}$$

represents the corresponding super-oppostion relationship. In other words, for $x = \frac{a+b}{2}$ the entire interval except x becomes the super-opposite of x. This means that for $x \to \frac{a+b}{2}$ the type-I opposites converge to the same value and the range of super-opposite values increases. Using our above child/adult age example, then a member with an age of 50 years has all elements $\neq 50$ as its super-opposites, including the 10 year old and the 90 year old, simultaneously.

Definition 4 (Type-I Quasi-Opposition). *Let C be an n-dimensional concept in set $\mathcal{C}$. Then, all points $\breve{C}^q$ are type-I quasi-opposite of C when $d(\breve{C}^q, C) < d(\breve{C}, C)$, for some distance function $d(\cdot,\cdot)$.*

Similar to super-opposition, quasi-opposition will not enter into all systems. For example, consider a binary problem where each $x_i \in \{0, 1\}$. In this case, neither quasi- nor super-opposition are of any relevance. However, for a more practical situation consider the single dimensional hypothecial case where $\breve{x} = a + b - x$, then

$$\breve{x}^q \in \begin{cases} (\breve{x}, \breve{x} + \min[d(a,\breve{x}), d(\breve{x},c)]) & \text{for} \quad x > (a+b)/2 \\ \emptyset & \text{for} \quad x = (a+b)/2 \\ (\breve{x} - \min[d(b,\breve{x}), d(\breve{x},c)], \breve{x}) & \text{for} \quad x < (a+b)/2 \end{cases} \tag{2.3}$$

defines the set of quasi-oppositional values with $c = \frac{a+b}{2}$.

Whereas the super-opposition is naturally bounded by the extremes of the universe of discourse, the quasi-opposites can be bounded differently. A convenient way is to mirror the same distance that the opposite has to the closest interval limit in order to set a lower bound for quasi-opposites. This is illustrated in Figure 2.1. Of course, one could define a degree of quasi-oppositeness and determine the slope of gradual increase/decrease based on problem specifications. This, however, will fall into the scope of similarity/dissimilarity measures, which has been extensively investigated for many other research fields [5,10,19,20].

We have defined how type-I opposition deals with the relationship between concepts based on features of the concepts without regard to the actual quality of the concept. Although, there does exist a relationship between these two cases, they are at heart, different. Quality is measured with respect to the question being asked of the opposites,in contrast to the physical structure of the two concepts. We can summarize that type-I opposition models/captures oppositional attributes in a linear or almost linear fashion, is easy to compute and, most likely, can be regarded as an approximation of the real opposites (type-II opposition) for non-linear problems.

Definition 5 (Type-II Opposition Mapping). *Any one-to-many function* $\Upsilon\colon f(\mathcal{C}) \rightarrow f(\mathcal{C})$ *which defines an oppositional relationship between the evaluation* f *of a concept*

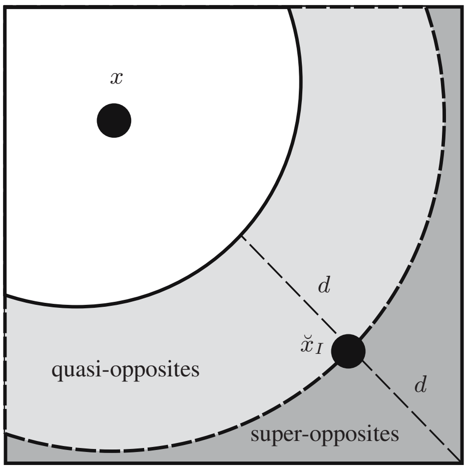

Fig. 2.1. Simplified representation of quasi- and super-opposites in a 2-dimensional space. The distance d of the type-I opposite $\breve{x}_I$ from the closest corner can be used to define the quasi- and super-oppositional regions. The boundary between these two region (bolded dashed circle segment) can be regarded quasi-opposite for a strict interpretation.

$C \in \mathcal{C}$ to a set S of all other evaluations of concepts also in $\mathcal{C}$ such that $C \in \Upsilon(s)$ $\forall s \in S$.

The focus on evaluation f of concepts, instead of focusing on the concepts themselves, leads to loss of information regarding the true relationship of opposite concepts (i.e. type-II is a phenotypical relationship as opposed to a structural one). Given some concept C and its associated evaluation $f(C)$, the type-II opposition mapping will return a set of opposite evaluations which can then lead to concepts (although the specific true type-I opposite concept cannot be determined with certainty). Here, Υ returns a set because f is not restricted to any functional form. For example, if f is a periodic function such as the sine wave defined over $[-2\pi, 2\pi]$, then for any $f(x)$ there will be at least two opposites since a horizontal plane will cut the wave twice at each of $-3/2\pi, 1/2\pi$ and $-1/2\pi, 3/2\pi$, respectively (5 points at $-2\pi, -\pi, 0, \pi, 2\pi$ and will cut the wave at 4 points elsewhere).

Definition 6 (Type-II Opposition). *Let concept C be in set $\mathcal{C}$ and let $\Upsilon: f(\mathcal{C}) \rightarrow f(\mathcal{C})$ be a type-II opposition mapping where f is a performance function (such as error, cost, fitness, reward, etc.) Then, the set of type-II opposites of C are completely defined by Υ. Type-II opposition can also be coined* non-linear opposition.

An example of type-II opposition is in the following. Let $y = f(x_1, x_2, \cdots, x_n) \in \mathbb{R}$ be an arbitrary function where $y \in [y_{\min}, y_{\max}]$ are the extreme values of f. Then, for every point $C = (c_1, c_2, \cdots, c_n)$ we can define the type-II opposite point $\breve{C} = (\breve{c}_1, \breve{c}_2, \cdots, \breve{c}_n)$ according to

$$\breve{C} = \Upsilon(C) = \{c \mid \breve{y} = y_{\min} + y_{\max} - y\}. \tag{2.4}$$

We can assume that f is unknown, but that $y_{\min}$ and $y_{\max}$ are given or can be reasonably estimated.

Alternatively, at sample t we may consider a type-II opposite with respect to $t-1$ previous samples. Using Equation (2.4), this temporal type-II opposite can be calculated according to

$$\breve{c}_i(t) = \left\{ c \mid \breve{y}(t) = \min_{j=1,\ldots,t} y(j) + \max_{j=1,\ldots,t} y(j) - y(t) \right\}. \tag{2.5}$$

Figure 2.2 depicts the difference between type-I and type-II opposition for a simple case where we are trying to find the extremes of a non-linear function. As apparent, type-II opposition can provide more reliable information.

In general, a type-II opposition scenario implies a deep understanding of the tartget concept, which is typically difficult to attain for real-world applications. However, type-II opposition mappings can be approximated online as the learning/search is in progress. We term this situation *opposition mining*.

Definition 7 (Opposition Mining). *Let $C^* \in \mathcal{C}$ be a target concept and Υ a type-II opposition mapping. Then, opposition mining refers to the online discovery of $\bar{\Upsilon}$, which represents an approximation to Υ.*

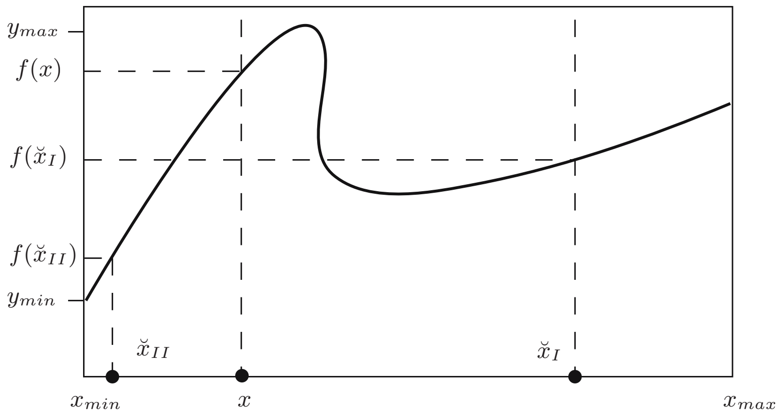

Fig. 2.2. Type-I versus type-II opposition. A nonlinear function $f : x \rightarrow [y_{min}, y_{max}]$ has the value $f(x)$ at the position x. The type-I opposite $\breve{x}_I = x_{max} + x_{min} - x$ generates the function value $f(\breve{x}_I)$. The type-II opposite $\breve{x}_{II}$, however, corresponds to the real opposite value $f(\breve{x}_{II}) = \breve{f}(x) = y_{max} + y_{min} - f(x)$. Evidently, $f(\breve{x}_I) \rightarrow f(\breve{x}_{II})$ for a linear or almost linear f.

The definitions for super- and quasi-oppostion can also be extended to the type-II domain, although we have not provided definitions here. Similarily, we can discuss the notion of a degree of opposition for both type-I and type-II scenarios (we will assume, without loss of generality a type-I situation).

Definition 8 (Degree of Opposition). *Let Φ be a type-I opposition mapping and $C_1, C_2 \in \mathcal{C}$ arbitrary concepts such that $C_1 \neq \Phi(C_2)$. Then, the relationship between C_1 and C_2 is not opposite, but can be described as partial opposition, determined by a function $\tau\colon C_1, C_2 \rightarrow [0, 1]$. As $\tau(C_1, C_2)$ approaches 1 the two concepts are closer to being opposite of each other.*

So, it is possible to compare elements $C_1, C_2 \in \mathcal{C}$ within the framework of opposition. Calculating the degree of opposition $\tau(C_1, C_2)$ will depend on how concepts are represented for the problem at hand. Nevertheless, this definition allows for the explanation of the relationship between elements that are not truly opposite. For example, consider the statements

$$\begin{aligned} s_1 &= \text{“Everybody likes apples”} \\ s_2 &= \text{“Nobody likes apples”} \\ s_3 &= \text{“Some people like apples”} \\ s_4 &= \text{“Some people do not like apples”} \end{aligned}$$

We can describe the relationship between these concepts with respect to s_1 using the degree of opposition. Doing so, we can determine that $\tau(s_1, s_4) < \tau(s_1, s_3) < \tau(s_1, s_2)$, where the exact value of each relationship is ignored but is related to the logical meaning of the statements. It is important to distinguish between the degree of opposition and super- and quasi-opposition. The latter relationships are related to physical distance whereas the former is concerned with the logical relationship. In some cases these may overlap, and others there may be no such similarity.

As an example of how to combine the degree of opposition with a temporal situation as described above, assume only the evaluation function $g(X)$ is available. This is accomplished by adding the temporal variable t to the definition of the τ function. Then, using the same scenario as Equation (2.5) the type-II temporal opposite for variables X_1 and X_2 could then be computed according to

$$\tau(X_1 = x_1, X_2 = x_2, t) = \frac{|g(x_1) - g(x_2)|}{\max\limits_{j=1,\ldots,t} g(x_j) - \min\limits_{j=1,\ldots,t} g(x_j)} \in [0,1]. \tag{2.6}$$

A more practical example of employing the degree of opposition can be found in [22], where opposite actions were utilized for a reinforcement learning (RL) algorithm. In order to make additional updates to the Q matrix for the Q-learning algorithm, an RL agent has to know how to find opposite actions and opposite states. Clearly, this is an application-dependant problem although in some applications such as control and navigation a good definition of opposite actions is generally straightforward (*increase temperature* is the opposite of *decrease temperature*, and *move left* is the opposite of *move right*). Nonetheless, general procedures may be defined to facilitate the situation where a good opposite action may not be so apparent.

In [22] this was accomplished by defining the degree of opposition τ as a measure in how far two actions a_1 and a_2 are opposite of each other in two different states s_i and s_j, respectively, such as

$$\tau(a_1|_{s_i}, a_2|_{s_j}) = \eta \times \left[1 - \exp\left(-\frac{|Q(s_i,a_1) - Q(s_j,a_2)|}{\max\limits_k \left(Q(s_i,a_k), Q(s_j,a_k)\right)}\right)\right], \tag{2.7}$$

where the Q matrix contains the accumulated discounted rewards (which guide learning), and η is the *state similarity* and can be calculated based on state clustering or by a simple measure such as

$$\eta(s_i,s_j) = 1 - \frac{\sum\limits_k |Q(s_i,a_k) - Q(s_j,a_k)|}{\sum\limits_k \max\left(Q(s_i,a_k), Q(s_j,a_k)\right)}. \tag{2.8}$$

In [22] limited examples are provided to intuitively verify the usefulness of this measure in context of the reinforcement learning problems. However, online opposition mining based on this measure, embedded inside the parent algorithm, has not been investigated.

2.3.2 Example for Using Opposition

There are numerous manifestations of oppositional relationships in both physical world and the human society, too many to show an example for each. Instead, we will restrict ourselves to a random sampling scenario. We assume at least one optimal solution exists within a given interval $[a, b]$, and that we are given feedback regarding the distance of a guess to the closest optimal solution. To make comparing the approaches easier we group guesses into pairs.

It has been shown that for a problem with a single optima, where distance feedback is given as evaluation criteria that guesses made in a specific dependent manner (in contrast to purely random) have a higher probability of being closer to the optimal solution [17,18].

Theorem 1 (Searchig in n-Dimensional Space). *Let $y = f(X^n)$ be any unknown function with $X^n = (x_1, \ldots, x_n)$ such that $x_i \in [a_i, b_i]$, where $a_i, b_i \in \mathbb{R}$. Also, let y have some extreme values $S = \{s_1, \ldots, s_n\}$, without loss of generality assumed the minima of y. Generate X, Y based on a uniform distribution with respect to the interval $[a_i, b_i]$ and let $\breve{X} = a_i + b_i - X_i$. Then*

$$Pr\left(\|\breve{X}, S\| < \|Y, S\|\right) > Pr\left(\|Y, S\| < \|\breve{X}, S\|\right)$$

where $\|\cdot\|$ denotes the Euclidean distance.

Similarly, if $\breve{X} = (0.5+X)\%1$ we can also achieve a similar expected behavior. Indeed any low discrepancy sequence (see Chapter 3) should yield an improvement, in terms of probability of making guesses closer to the optima, over purely random sampling. This concept has been employed in quasi-Monte Carlo methods [8,9,15,23], evolutionary algorithms [7,11,12,17], particle swarms [13,14,16] and neural networks [3,6], to name a few. The underlying concept is similar in all cases. Although in relation to opposition-based computing, typically the focus is on pairs of samples *(guess, opposite guess)*.

Table 2.1 shows probabilities of randomly generating a solution in 2, 5 and 10 dimensions over two opposition mappings, both compared to purely random sampling. For each dimension we assume the interval is $[0, 1]$, without loss of generality. The first method, called “Mirror”, uses $\breve{x}_i = 1 - x_i$, and the second method, “Modulo”, is defined as $\breve{x}_i = (0.5 + x_i)\%1$. To make comparing the approaches simpler, we examine pairs of guesses. The table shows triples $< P(\min(X, Y) = \min(\breve{X}, Y)), P(\min(X, \breve{X}) < \min(X, Y)), P(\min(X, \breve{X}) > \min(X, Y) >$ which compare the probability of two guesses being equal, opposition-based technique is more desirable and random guesses are better, respectively. The data was gathered by randomly selecting 1000 solutions, where 10,000 guessing pairs were considered. The value provided represents the average over these experiments.

In all experiments, the non-random guessing yields a higher probability of being closer to one of the optimal solutions. Although, as the number of optimal solutions increases, this benefit decreases, especially for the Mirror opposition mapping. But, the Mirror function seems to be the best of the two mappings when there only exists one optimal solution.

In general for distance-based problems, given m optimal solutions represented as the set $S = \{S_1, ..., S_m\}$, the goal is to devise an opposition mapping such that if we let $g_1 = d(X, s^{1,*})$, $g_2 = d(Y, s^{2,*})$ and $g_3 = d(\Phi(X), s^{3,*})$ then we desire the expected value relationship,

Table 2.1. Generating guesses where $< P(\min(X,Y) = \min(\breve{X},Y)), P(\min(X,\breve{X}) < \min(X,Y)), P(\min(X,\breve{X}) > \min(X,Y) >$ for variables X, Y. In most cases, a Modulo function outperforms the Mirror function, but both show an improvement over purely random sampling. See Chapter 3 for more on low-discrepancy sequences.

	Number of optimal Solutions		
Method	1	2	5
	2 Dimensions		
Mirror	$< 0.358, 0.358, 0.283 >$	$< 0.344, 0.343, 0.313 >$	$< 0.338, 0.338, 0.321 >$
Modulo	$< 0.364, 0.364, 0.273 >$	$< 0.355, 0.355, 0.291 >$	$< 0.344, 0.344, 0.312 >$
	5 Dimensions		
Mirror	$< 0.361, 0.361, 0.278 >$	$< 0.346, 0.346, 0.309 >$	$< 0.334, 0.334, 0.332 >$
Modulo	$< 0.359, 0.359, 0.282 >$	$< 0.353, 0.354, 0.293 >$	$< 0.347, 0.347, 0.307 >$
	10 Dimensions		
Mirror	$< 0.361, 0.362, 0.277 >$	$< 0.345, 0.345, 0.310 >$	$< 0.332, 0.332, 0.336 >$
Modulo	$< 0.344, 0.360, 0.296 >$	$< 0.343, 0.358, 0.299 >$	$< 0.340, 0.356, 0.304 >$

$$E[\min(g_1, g_3)] < E[\min(g_1, g_2)] \tag{2.9}$$

where $d(\cdot,\cdot)$ is a distance function and $s^{1,*}, s^{2,*}, s^{3,*}$ are the solutions closest to each of X, Y and $\Phi(X)$, respectively. Thus, the most desirable opposition map Φ^* will be one that maximizes the difference

$$\Phi^* = \underset{\Phi}{\operatorname{argmax}}\, E[\min(g_1, g_2)] - E[\min(g_1, g_3)], \tag{2.10}$$

although improved performance should be observed for any Φ satisfying equation 2.9. Notice though that it is possible to design poor opposition mapping functions which actually decrease the performance of the algorithm.

2.4 Opposition-Based Computing

Having introduced the foundational definitions for opposition-based computing, we will now provide its definition as well as a first attempt at a classification strategy for opposition-based computing methods in the following section.

Definition 9 (Opposition-Based Computing). *We speak of opposition-based computing, when a computational method or technique implicitly or explicitly employs oppositional relations and attributes either at its foundation (purely oppositional algorithms) or to improve existing behavior of some parent algorithm (opposition-based extensions of existing algorithms).*

Situations in which opposition is inherent to the algorithm existence (for instance switching between opposite alternatives in a decision-making scenario) is generally termed *implicit* because the underlying concept tends to be overlooked in favor of improving its behavior by other means. For example, adversarial searching using game tree searching typically follows a turn-based rule which alternates between two players. The focus of research in this area tends towards decreasing computational time via improving the decision as to which nodes are to be explored instead of changing the underlying adversarial structure of the algorithm. A similar concept also applies to other competition-based methods and paradigms.

In contrast to implicit opposition, explicit forms occur when the goal is to utilize the non-apparent or imperceptible opposition as a means of improving an existing, usually non-oppositional algorithm. An example of this is the use of antithetic sampling to reduce the variance and improve convergence rate of Monte Carlo integration methods. The integration method works without the use of any opposites, but by explicitly using antithetic variates we can observe improved behavior in terms of reduced variance leading to faster convergence.

In both implicit and explicit algorithms it is possible to further determine the manner in which opposites interact with respect to the scale $[0 = \mathit{cooperative}, \ldots, 1 = \mathit{competitive}]$.

Definition 10 (Degree of Competition). *Let $C_1, C_2 \in \mathcal{C}$ be arbitrary concepts where $C_1 = \Phi(C_2)$. With inherent respect to the algorithm employing opposition, Σ, the degree at which C_1, C_2 compete for use in Σ is given by the function $\zeta\colon C_1, C_2|\Sigma \rightarrow [0,1]$. Values of ζ closer to 1 indicate a more competive situation. For readability we often drop the Σ, though it is implied.*

Basically, this definition allows us to measure how cooperative or competitive the consideration of two opposite guesses/concepts are. For a game tree search $\zeta \rightarrow 1$ (totally competitive), and in case of antithetic variates we take an average of the results such that $\zeta \rightarrow 0$ (totally cooperative).

We can also speak of the degree of competition between concepts which have a degree of opposition < 1. The above definition changes slightly in that the requirement of $C_1 = \Phi(C_2)$ is dropped and we adopt a subscript ζ_τ to represent the case that the concepts are not true opposites.

Whether wishing to describe the degree of opposition or the degree of competition between opposites, there may be a time componet to the interaction. In such cases the goal is to describe the degree of opposition between aribitrary concepts $C_1, C_2 \in \mathcal{C}$ as time (or algorithm iterations) t varies,

$$\Upsilon_t(C_1, C_2) = \frac{\mathrm{d}\Upsilon(C_1, C_2)}{\mathrm{d}t}. \tag{2.11}$$

In a similar fashion we can describe the behavior of the degree of competition between concepts with respect to algorithm Σ at time t as

$$\zeta_t(C_1, C_2) = \frac{\mathrm{d}\zeta(C_1, C_2)}{\mathrm{d}t}. \tag{2.12}$$

These definitions are very broad and there are many possible manifestations of opposition that can arise. We provide possibilities below, but this is by no means a comprehensive list. Of course, some of these areas overlap, and can be hybridized.

2.5 OBC Algorithms

Opposition can be embedded into existing algorithms implicitly or explicitly. For instance, the Proof Number Search (Chapter 6) is an example of implicit opposition usage, whereas the ODE algorithm (Chapter 8) is an explicit implementation. In this section we establish a general framework for explicit employment of OBC within existing methodologies. For this purpose we begin general definitions of *OBC algorithms*.

Definition 11 (Implicit OBC Algorithms). *Any algorithm that incorporates oppositional concepts without explicitly using type-I or type-II opposites is an implicit OBC algorithm, or short an I-OBC algorithm.*

The range of I-OBC techniques is large since they have been in use since quite some time. For example, the well-known bisection method [4] for solving equations can be considered an I-OBC algorithm since it shrinks the search interval by looking at positive versus negative sign change. As well, using Bayes theorem [2] is an implicit usage of oppositional relationship between the conditional probabilities $p(A|B)$ and $p(B|A)$[5]. A different, stronger version of implicit incorporation of oppositional concepts is Bayesian Yin-Yang Harmony Learning which is based on the alternative viewpoints of Bayes Rule (Chapter 10).

Definition 12 (Explicit OBC Algorithms). *Any algorithm dealing with an unknown function $Y = f(X)$ and a known evaluation function $g(X)$ (with higher values being more desirable) extended with OBC to calculate type-I or type-II opposite $\breve{X}$ of numbers, guesses or estimates X for considering $\max\Big(g(X), g(\breve{X})\Big)$ in its decision-making is an explicit OBC algorithm, or short an E-OBC algorithm.*

Generally, three classes of E-OBC algorithms can be distinguished:

1. **Initializing E-OBC Algorithms –** The concept of opposition is only used during the initialization. The effect is a better start (initial estimates closer to solution vicinity). Further, since it is done before the actual learning/searching begins, this creates no additional overhead.
2. **Somatic E-OBC Algorithms –** This approach is based on modification of the body of an existing algorithm. For instance, changing the weights in a neural network during the learning based on integration of opposites weights changes the way the algorithm works in every step. The effect of OBC will be much more visible since opposition is considered in every iteration/episode/generation.
3. **Initializing and Somatic E-OBC Algorithms –** This class uses OBC both during initialization and actual learning/search by combining the two previous approaches.

Algorithm 1 provides the generic structure of E-OBC algorithms. However, it seems that the selection of an appropriate OBC algorithm and its specific steps directly depend on the problem at hand and cannot be universally established.

[5] The underlaying idea of Bayes theorem is looking at events from opposite perspectives when analyzing a cause-effect problem. For example, if we have the prior probability of cancer, $p(\text{cancer})$, and on smoking, $p(\text{smoking})$, then the conditional posterior probability $p(\text{cancer}|\text{smoking})$ can be determined when we make an antipodal observation for the conditional probability $p(\text{smoking}|\text{cancer})$.

Algorithm 1. General pseudo code of initializing and somatic E-OBC algorithms. Different schemes may/shall be defined for specific parent algorithms.

```
1: Establish evaluation function $g(\cdot)$ (e.g. error, fitness, reward etc.)
2: Initialize parameters P=$\{p_1, \cdots, p_n\}$ (e.g. weights, chromosomes etc.)
3: Find opposites $\breve{P} = \{\breve{p}_1, \cdots, \breve{p}_n\}$

4: **if** First option **then**
5:     Establish $M = P \cup \breve{P}$
6:     Sort $M$ in descending order with respect to $g(\cdot)$
7:     Choose the $n$ first parameters $m_1, \cdots, m_n$ from $M$
8: **end if**

9: **if** Second option **then**
10:     Run the algorithm with $P$
11:     Run the algorithm with $\breve{P}$
12:     Choose the better one with respect to $g(\cdot)$
13: **end if**

14: **for** each learning/search loop of the parent algorithm **do**
15:     Calculate the estimate $X$ based on schemes provided by the parent algorithm
16:     **if** OBC-condition satisfied **then**
17:         Calculate $\breve{X}$
18:         Perform the parent algorithm's steps for $\breve{X}$
19:         Take $X$ or $\breve{X}$ based on $\max(g(X), g(\breve{X}))$
20:     **end if**
21: **end for**
```

Line 1 in Algorithm 1 controls the frequency of relaying on opposition while the parent algorithm runs. The following cases can be distinguished:

1. **OBC condition always true –** OBC can be applied during the entire learning/search, meaning that in all iterations/episodes/generations, proper opposite quantities are calculated and comparisons with original quantities are conducted. It could appear that in this way we can fully exploit the benefit of oppositional concept. However, in some cases the additional overhead for calculation of opposite entities can exceed its advantage; the extended algorithm becomes less effective than its original version in terms of convergence speed (the accuracy will be at least as high as the regular call without OBC due to $\max(g(X), g(\breve{X}))$).
2. **OBC condition as a threshold –** If we decide to activate OBC only for a fraction of time, then the most straightforward condition for opposite estimates is to set a threshold θ. If number of iterations/episodes/generations is less than θ, for instance, then opposite estimates will be calculated and considered. This will eliminate the overhead complexity but adds a degree of freedom which needs empirical knowledge to be filled.
3. **OBC condition as a probabilistic constraint –** Opposite consideration could be triggered by a probability function, which ideally is coupled with the evaluation function $g(X)$. Empirical and uniform probabilities may be used as well.

Generally, probabilistic control of opposition frequency should be either a decreasing or increasing function. For former, we rely on opposition rather at the begining stages of learning/search and do not use it towards the end to suppress the computational overhead. For latter case, we increase the opposition frequency as learning/search progresses since, perhaps, we do not know the true opposites initially. This would correspond to dynamic/online opposition mining.

In following subsections we try to categorize different OBC algorithms.

2.5.1 Search and Reasoning

Generally speaking, a search algorithm explores possible solutions to a problem in a systematic manner with the goal of discovering some solution. Although, in many instances the goal may not be obtainable in a reasonable amount of time. A popular approach to deal with this scenario is to incorporate heuristic knowledge about the problem or solution to guide the searching algorithm, instead of an uninformed search which does not use any such knowledge. Reasoning is a process of searching for some explanation for a belief or action. Therefore search and reasoning are closely coupled processes. Examples include:

1. Implicit: adversarial search and reasoning (see Chapter 6).
2. Explicit: understanding or compromising between opposing arguments, considering opposite reasonings (see Chapter 5).

2.5.2 Optimization

The purpose of optimization algorithms is to maximize or minimize some real-valued function. Without loss of generality, consider a minimization problem where we are given a function $f : A \to \mathbb{R}$ from some set A we seek to find an element $a^* \in A$ such that $f(a^*) \leq f(a)\ \forall a \in A$. Of course, this generalizes all searches, but the reader should keep important aspects such as constraint satisfation, multiple objectives, etc in mind. Examples include:

1. Implicit: Competitive coevolution (see Chapter 9).
2. Explicit: Using a low-discrepancy paired guessing strategy to improve diversity and/or convergence rate (see Chapter 3).

2.5.3 Learning

Learning refers to the process of modifying behavior as a result of experiences and/or instructions. In other words, given some initial hypothesis, learning allows for its modification as more data are observed. This contrasts with search and reasoning in that learning is not concerned with explaining why the data is observed, only to perform some action when it occurs. Often learning can be seen as an optimization problem, however, here we distinguish them because they do not always imply one another. Examples include:

1. Implicit: model selection using opposing viewpoints of a problem (see Chapter 10).
2. Explicit: considering two opposing actions or states during active learning (see Chapter 11).

2.6 Summary and Conclusions

In this work, the general framework of opposition-based computing was introduced. Basic definitions of type-I and type-II opposites along with categorization of oppositional algorithms were provided as well. Generally, OBC provides a straightforward framework for extension of many existing methodologies as this has been demonstrated for neural nets, evolutionary algorithms, reinforcement learning and ant colonies. The OBC extensions of existing machine learning schemes seem to primarily accelerate search, learning and optimization processes, a characteristic highly desirable for hyperdimensional complex problems.

Opposition-based computing, however, encompasses multiple challenges as well. First and foremost, a solid and comprehensive formalism is still missing. Besides, the definition of opposition may not be straightforward in some applications such that the need for opposition mining algorithms should be properly addressed in future investigations. On the other side, the cost of calculating opposite entities can exceed the time saving so that the OBC-extended algorithm becomes slower than the parent algorithm. Extensive research is still required to establish control mechanisms to regulate the frequency of opposition usage within the parent algorithms.

Due to our imperfect understanding of interplay between opposite entities, this work will most likely be a preliminary investigation. Hence, more comprehensive elaborations with a solid mathematical understanding of opposition remain subject to future research.

The motivation of this chapter was not to just provide a framework for oppositional concepts claiming to cover all possible forms of opposition (due to the diversity of oppositeness this seems to be extremely difficult, if not impossible), but also to establish a general umbrella under which other, existing techniques can be classified and studied. This should not be understood to just put a new label on existing methodologies and algorithms but to consciously comprehend the oppositional nature of problems and exploit a-priori knowledge in light of inherent antipodality and complementarity of entities, quantities and abstractions.

2.7 Contributions and Acknowledgements

The idea of opposition-based computing and its applications for genetic algorithms, neural nets, reinforcement learning and ant colonies has been introduced by Dr. Hamid R. Tizhoosh. Mario Ventresca has designed and developed OBC versions of backpropagation algorithm and simulated annealing, and provided some formalism to establish the OBC definitions. Dr. Shahryar Rahnamayan has introduced opposition-based differential evolution (ODE) with practical and theoretical contributions to some of the formalism in this work. The authors would like to thank Dr. Don McLeish for valuable discussions with respect to the relationship between antithetic variates and opposite quantities. We also are grateful to Maryam Shokri and Dr. Farhang Sahba for their contributions to the OBC research. As well, we would like to thank Alice Malisia and Masoud Mahootchi who have provided examples how OBC can be practically implemented.

References

1. The American Heritage Dictionary of the English Language. Houghton Mifflin (2000)
2. Carlin, B.P., Louis, T.A., Carlin, B.: Bayes and Empirical Bayes Methods for Data Analysis. Chapman and Hall/CRC, Boca Raton (2000)
3. Cervellera, C., Muselli, M.: Deterministic Design for Neural Network Learning: An Approach based on Discrepancy. IEEE Transactions on Neural Networks 15(3), 533–544 (2004)
4. Chapra, S.C., Canale, R.P.: Numerical Methods for Engineers. McGraw-Hill Education, New York (2005)
5. Cheng, V., Li, C.H., Kwok, J.T., Li, C.K.: Dissimilarity learning for nominal data. Pattern Recognition 37(7), 1471–1477 (2004)
6. Jordanov, I., Brown, R.: Neural Network Learning Using Low-Discrepancy Sequence. In: Foo, N.Y. (ed.) Canadian AI 1999. LNCS, vol. 1747, pp. 255–267. Springer, Heidelberg (1999)
7. Kimura, S., Matsumura, K.: Genetic Algorithms using Low-Discrepancy Sequences. In: Genetic and Evolutionary Computation Conference, pp. 1341–1346 (2005)
8. Kucherenko, S.: Application of Quasi Monte Carlo Methods in Global Optimization. In: Global Optimization, pp. 111–133 (2006)
9. Kucherenko, S., Sytsko, Y.: Application of Deterministic Low-Discrepancy Sequences in Global Optimization. Computational Optimization and Applications 30, 297–318 (2005)
10. Li, M., Chen, X., Li, X., Ma, B., Vitanyi, P.M.B.: The similarity metric. IEEE Transactions on Information Theory 50(12), 3250–3264 (2004)
11. Maaranen, H., Miettinen, K., Makela, M.M.: Quasi-random Initial Population for Genetic Algorithms. Computers and Mathematics with Applications 47(12), 1885–1895 (2004)
12. Maaranen, H., Miettinen, K., Penttinen, A.: On Initial Populations of a Genetic Algorithm for Continuous Optimization Problems. Journal of Global Optimization 37, 405–436 (2007)
13. Nguyen, Q., Nguyen, X., McKay, R., Tuan, P.: Initialising PSO with Randomised Low-Discrepancy Sequences: The Comparative Results. In: Congress on Evolutionary Computation, pp. 1985–1992 (2007)
14. Nguyen, X., Nguyen, Q., McKay, R.: Pso with Randomized Low-Discrepancy Sequences. In: Genetic and Evolutionary Computation Conference, p. 173 (2007)
15. Niederreiter, H., McCurley, K.: Optimization of Functions by Quasi-random Search Methods. Computing 22(2), 119–123 (1979)
16. Pant, M., Thangaraj, R., Grosan, C., Abraham, A.: Improved Particle Swarm Optimization with Low-Discrepancy Sequences. In: Congress on Evolutionary Computation (to appear, 2008)
17. Rahnamayan, S., Tizhoosh, H.R., Salamaa, M.: A Novel Population Initialization Method for Accelerating Evolutionary Algorithms. Computers and Mathematics with Applications 53(10), 1605–1614 (2007)

18. Rahnamayan, S., Tizhoosh, H.R., Salamaa, M.: Opposition versus randomness in soft computing techniques. Applied Soft Computing 8(2), 906–918 (2008)
19. Sarker, B.R., Saiful Islam, K.M.: Relative performances of similarity and dissimilarity measures. Computers and Industrial Engineering 37(4), 769–807 (1999)
20. Stewart, N., Brown, G.D.A.: Similarity and dissimilarity as evidence in perceptual categorization. Journal of Mathematical Psychology 49(5), 403–409 (2005)
21. Tizhoosh, H.R.: Opposition-Based Learning: A New Scheme for Machine Intelligence. In: International Conference on Computational Intelligence for Modelling Control and Automation, vol. 1, pp. 695–701 (2005)
22. Tizhoosh, H.R.: Opposition-based reinforcement learning. Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII) 10(4), 578–585 (2006)
23. Yaochen, Z.: On the Convergence of Sequential Number-Theoretic Method for Optimization. Acta Mathematicae Applicatae Sinica 17(4), 532–538 (2001)