

CIRG@UP OptiBench: A statistically sound framework for benchmarking optimisation algorithms

E. S. Peer

Department of Computer Science
University of Pretoria
South Africa
espeer@cs.up.ac.za

A. P. Engelbrecht

Department of Computer Science
University of Pretoria
South Africa
engel@cs.up.ac.za

F. van den Bergh

Rapid Mobile
Pretoria
South Africa
frans@rapidm.com

Abstract- This paper is a proposal, by the Computational Intelligence Research Group at the University of Pretoria (CIRG@UP), for a framework to benchmark optimisation algorithms. This framework, known as OptiBench, was conceived out of the necessity to consolidate the efforts of a large research group. Many problems arise when different people work independently on their own research initiatives. These problems range from duplicating effort to, more seriously, having conflicting results. In addition, less experienced members of the group are sometimes unfamiliar with the necessary statistical methods required to properly analyse their results. These problems are not limited internally to CIRG@UP but are also prevalent in the research community at large. This proposal aims to standardise the research methodology used by CIRG@UP internally (initially in the optimisation subgroup and later in subgroups working in other paradigms of computational research). Obviously this paper cannot dictate the methodologies that should be used by other members of the broader research community, however, the hope is that this framework will be found useful and that others will willingly contribute and become involved.

“He uses statistics as a drunken man uses lamp-posts – for support rather than illumination.”
— Andrew Lang (1844-1912), Treasury of Humorous Quotations

1 Introduction

The Computational Intelligence Research Group at the University of Pretoria (CIRG@UP), at the time of this writing, consists of just over forty postgraduate students. The group performs research in neural networks [1, 2], swarm intelligence [3], evolutionary computation [4, 5], artificial immune systems [6], data and text mining [7, 8], image analysis [9] and multi-agent systems¹ [10, 11]. A group this large and diverse presents considerable logistical challenges. OptiBench is a framework/tool that aims to meet these challenges by alleviating some common problems that have been identified. These problems are:

- Duplication of effort
- Insufficient testing on problems

¹For more detailed information about the group and its activities consult the CIRG@UP web-site at <http://cirg.cs.up.ac.za/>

- Failure to test against latest developments (comparing to canonical algorithms only)
- Poor choices for parameters
- Conflicting results
- Invalid statistical inference

What is rather frightening is how many of these problems are prevalent in the broader research community (see Section 3 for examples).

The swarm intelligence subgroup, the largest in CIRG@UP, predominantly performs research in optimisation using particle swarms [12, 13]. Since it is the largest, a framework was initially developed to facilitate research in this specific subgroup. For this reason many of the examples presented in this paper will be specific to particle swarm research. Currently, the framework itself is specific to optimisation of unconstrained continuous problems, leading to the name OptiBench. Later, the intention is to expand this framework for other paradigms, catering to the needs of the rest of the group. It should be noted that OptiBench is a work in progress and does not yet fully satisfy all the needs of the researchers in the swarm intelligence subgroup. Nevertheless some of the elements of the framework are already being found useful in the rest of the group and may likely be useful to the broader research community as well.

Without loss of generality, unconstrained optimisation problems can be represented as follows:

$$\text{Given } f : \mathbb{R}^n \rightarrow \mathbb{R} \\ \text{find } \mathbf{x}^* \in \mathbb{R}^n \text{ for which } f(\mathbf{x}^*) \leq f(\mathbf{x}), \forall \mathbf{x} \in \mathbb{R}^n$$

According to the No Free Lunch (NFL) theorem [14, 15] all optimisation algorithms perform equivalently when their performance is averaged over all possible optimisation problems having a finite search space. This means that any given algorithm can not outperform any other algorithm, including a simple random search or linear enumeration of the search space, on all problems. Therefore, designing a benchmark that yields a single number representing the relative performance for each algorithm over all optimisation problems is likely to be an exercise in futility. An algorithm that performs well on some of the problems is likely to perform poorly on others.

Christensen *et al* [16] have shown that a class of “searchable” functions exists for which better algorithms than random search can be found for all problems in this class. Thus

the NFL theorem does not hold for this class of problem. They also conjecture that many if not most problems of interest to optimisation research will exhibit this “searchable” characteristic.

OptiBench tests each algorithm on a large set of problems, much larger than is commonly used in the optimisation research community. This need for larger problem sets has also been recognised by Yao *et al* [17]. The entire problem set will ultimately include: various synthetic function minimisation problems; neural network training, on both synthetic and real world data, for classification as well as function learning problems; and, if the data is available, any other real world optimisation problems. The intension is to collect problems that are as representative as possible of the real world problems that CIRG@UP is interested in solving.

In addition to ranking the performance of each algorithm over the entire set of known benchmark problems (which is possibly dangerous in light of the NFL theorem), OptiBench allows subsets of the problems to be grouped. The ranking can then be calculated for a particular class of problem. This allows us to determine which optimisation algorithms are best suited to each class of problem.

The OptiBench framework consists of two main components, the OptiBench management console² and a Computational Intelligence Library (CILib)³. Scheduling of experiments on a distributed cluster of machines and the analysis of the results of these experiments are handled by the management console. CILib is a free library of problems and algorithms useful to computational intelligence research. The algorithms are written so that they can be used independently of the simulation environment in a real world scenario. Currently the library only includes particle swarm optimisation algorithms, since that is the area of research that its authors are working in, but it has been written in a fashion that is readily extendible. Other members of CIRG@UP have plans to extend the library into their own paradigms of research. Work has already been started on extending CILib in the areas of niching particle swarms, genetic algorithms and neural networks. The library has been released under the terms of the GNU⁴ General Public License (GPL)⁵ for reasons mentioned in Section 4.

A motivation for this proposal that highlights the existence of the problems discussed here is given in Section 3. In Section 4, an overview of OptiBench is presented with reference to how it addresses these problems. Finally some concluding remarks are made in Section 5. Some necessary statistical background is given in the following section.

2 Hypothesis testing

This section contains statistics that are required for the discussion in Section 3. First, some statistical tests that are fre-

²The OptiBench management console is accessible at <http://optibench.cs.up.ac.za/>

³The CILib source code is available at <http://cilib.sourceforge.net/>

⁴GNU is the recursively defined acronym: GNU’s Not Unix. See <http://www.fsf.org/> for details.

⁵The full terms of the GNU GPL are available at <http://www.fsf.org/licenses/gpl.html>

quently used [16, 17, 18] are presented. These tests all rely on random sampling from normal population distributions. Consequently, a test for normal sample distributions is included next. Finally, a test is presented that is valid when samples are not necessarily drawn from a normal population. This last test is used by OptiBench as discussed in Section 4.

The probability density function for the normal distribution $N(\mu, \sigma^2)$ is defined as [19]:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}(x-\mu)^2/\sigma^2} \quad (1)$$

where μ and σ^2 are parameters for the population mean and variance respectively. The standardised form of the normal distribution $N(0, 1)$ is centered around the origin with unit variance and is obtained using the transformation $y = (x - \mu)/\sigma$. The corresponding characteristic function is given by:

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^z e^{-\frac{1}{2}y^2} dy \quad (2)$$

Equation (2) represents the area under the distribution defined by Equation (1) over the transformed domain $-\infty \leq y \leq z$.

Under the assumption that two independent samples of size $n_{i=[1,2]} \geq 30$ are drawn from normal populations, the test statistic [20]:

$$Z = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{S_1^2/n_1 + S_2^2/n_2}} \quad (3)$$

has approximately a $N(0, 1)$ distribution, where \bar{X}_i and S_i^2 are the sample means and variances respectively. This pivotal quantity is used to perform statistical inference regarding the population means μ_1 and μ_2 from which the samples are drawn.

Specifically, for some kind of error metric, a one-sided test is needed to determine if one population’s average error is significantly better than another. The null hypothesis of $H_0 : \mu_1 = \mu_2$ and an alternative hypothesis $H_1 : \mu_1 < \mu_2$ can be formulated to determine if μ_1 is significantly smaller than μ_2 . The hypothesis H_0 can either be rejected or not rejected at some predetermined level of significance α . If H_0 is rejected then it can be concluded that H_1 is probably true. The probability $p = P(Z \leq z) = \Phi(z)$, where z is the calculated value of the test statistic Z , is known as the p -value. In general, H_0 can be rejected if $p \leq \alpha$.

The Student’s t -test [20] is another well known technique for performing inference about the population mean from sample data. Note that the t -test has two different test statistics depending on whether the samples have equal variances or not. The t -test also makes the assumption that the samples are drawn from a normally distributed population. In fact, for smaller samples it relies even more heavily on this assumption.

ANOVA (ANalysis Of VAriance) testing [21] is a popular method for performing multivariate analysis. This technique has the added advantage of handling $n \geq 2$ samples

simultaneously. However, it too makes the assumption that each sample is drawn from a normal population distribution.

Testing using any of these methods when the assumptions of a normal population are not satisfied is dangerous and can lead to inference errors.

The χ^2 goodness of fit test [19, 20] is a well known technique for testing whether a sample is drawn from a population with a given distribution. The test statistic:

$$C = \sum_{i=1}^k \frac{(O_i - E_i)^2}{E_i} \quad (4)$$

has a $\chi^2(k - p - 1)$ distribution where O_i is the observed frequency of the i^{th} event occurring, E_i is the expected frequency under the theoretical distribution being tested, k represents the number of distinct event categories and p represents the number of constraints added and/or parameters estimated, if any. Intuitively, large values of C indicate a deviation from the expected distribution. The problem with this approach is that it relies on the events occurring in discrete categories. Of course the normal distribution, which is continuous, can be made discrete by defining ranges for each category. The choice of these ranges is subjective and for this reason the Kolmogorov-Smirnov (KS) test which works for continuous functions is preferred.

Press *et al* [22] has a description as well as an efficient implementation for the KS test. To test if a sample is drawn from a normal population distribution the test statistic is simply:

$$D = \max_{-\infty < x < \infty} |S_N(x) - \Phi(x)| \quad (5)$$

where $S_N(x)$ is an unbiased estimator for the cumulative sample distribution given by the fraction of data points to the left of x . As for the χ^2 test, large values of D indicate a deviation from the expected normal distribution. In order to get the probability that a sample distribution is drawn from a normal population calculate:

$$P = Q_{KS}(\lfloor \sqrt{N_c} + 0.12 + 0.11/\sqrt{N_c} \rfloor D) \quad (6)$$

where N_c is the size of the sample and Q_{KS} is defined as:

$$Q_{KS}(\lambda) = 2 \sum_{j=1}^{\infty} (-1)^{j-1} e^{-2j^2 \lambda^2} \quad (7)$$

If it is determined that a sample was not drawn from a normal population then non-parametric or distribution free techniques should be employed. These tests typically perform inference on the population medians η_i as opposed to the means μ_i since they are more robust estimators for unknown distributions. While these non-parametric tests are typically less powerful than the parametric tests just presented, they do not make as many assumptions about the form or parameters of the population distribution.

An established non-parametric technique is the Wilcoxon rank sum test [20]. While it does assume that both samples have similar distributions, it does not require that they be normal. To test whether two sample

distributions are drawn from similar populations a slightly modified [22] KS test can be performed.

Given the null hypothesis $H_0 : \eta_1 = \eta_2$, the Wilcoxon test can be used to reject it in favour of $H_1 : \eta_1 < \eta_2$ at a given level of significance α . The two samples of size n_1 and n_2 with $n_1 \leq n_2$ are combined into a single joint sample. The observations in this joint sample are assigned ranks according to their values in ascending order. The test statistic defined as:

$$W = \sum_{(1)} R_i - \frac{n_1(n_1 + 1)}{2} \quad (8)$$

where $\sum_{(1)} R_i$ is the sum of the ranks over the first sample, has an exact distribution for which the p -values $P(W \leq w_{n_1, n_2, \alpha})$ are tabulated for $n_1 \leq n_2 \leq 10$ [20]. For larger samples the test statistic:

$$Z = \frac{W - n_1 n_2 / 2}{\sqrt{n_1 n_2 (n_1 + n_2 + 1) / 12}} \quad (9)$$

has approximately a $N(0, 1)$ distribution. In this case, the decision to reject H_0 can be made as described earlier for Equation (3).

3 Motivation

In light of the problems mentioned in Section 1, it is clear that standardisation of research methodologies within CIRG@UP is necessary. This standardisation can be greatly facilitated by a good tool that automates empirical studies and enables sharing of research software, resources and results. The hope is that OptiBench will prove to be such a tool. However, as noted earlier, these problems are not specific to CIRG@UP. As motivation, a few examples of these problems will be discussed here. It is hoped that the need for higher standards, that apply to the broader research community, will become apparent.

Sadly, example papers used in this section to illustrate these problems did not take much effort to find. In fact, most of them were drawn from a single conference proceedings. The analysis in this section has, for the most part, been limited to very recent papers in the particle swarm field. The situation would likely be far worse if a broader time frame and other computational intelligence paradigms were considered.

The purpose of this section is definitely not to point fingers but rather to illustrate that these problems do indeed exist. It should be noted that the authors of this paper have, up to now, also been guilty of compounding these problems. Awareness that these problems exist is the first step towards constructively tackling them as a community.

Duplication of effort is definitely a problem in the context of any research group. Each time a member of the group has to reinvent the wheel the productivity of the group as a whole is adversely impacted. Sharing a single software library amongst the group can put an end to everyone independently implementing their own versions of the same control algorithms for testing their new developments against.

This allows members of the group to concentrate their efforts on developing new and improved algorithms. In addition to the algorithms, efficiencies can be gained by sharing problem sets and statistical analysis tools as well. The challenge will be extending these beyond the research group.

A problem that exists generally, is researchers benchmarking the performance of their algorithms using only a small set of functions. Yao *et al* [17] is a notable exception. A set of functions that is too small can not be representative of complex real world problems irrespective of whether the NFL theorem (see Section 1) holds. CIRG@UP performs contract research for commercial clients and as a result is interested in solving such real world problems. The standard benchmark functions that permeate optimisation literature are simply not sufficient for testing algorithms. Specifically, they do not aid in determining which algorithms are best suited to particular classes of problems.

Another problem that exists generally, is researchers failing to take into account the latest developments within their field. Specifically, in particle swarm research, numerous (probably in excess of one hundred) improvements to the canonical Kennedy and Eberhart [12, 13] version have been introduced since 1995. Yet now, over eight years later, researchers [18, 23, 24, 25, 26, 27, 28, 29] still compare against outdated algorithms, sometimes claiming that their algorithm is superior without any reference to other, possibly even better, algorithms that have since been developed. Such comparisons are of little use to a person wishing to make use of this research to solve real world problems. In a real world scenario one wishes to use the best algorithm available to solve a particular type of problem.

In addition to testing against the latest developments, researchers should make a reasonable effort to determine the best parameters for the algorithms they make comparisons against. If better parameters cannot be determined or found in other research then those of the canonical version of the algorithm should be used. Crippling an algorithm with poor parameter choices and then making claims that a new development is superior is not fair.

As a specific illustration, consider that Ratnaweera *et al* [24] have noted some consensus within the discipline that swarm sizes should be between 20 and 50 (supported by Shi and Eberhart [30] and Van den Bergh and Engelbrecht [31]). Despite this, Wang [23], Secrest and Lamont [29] have used swarms of 100 particles. The latter authors go as far as to state that this size is commonly used without citing any references to back up their claim. Peram *et al* [28] in turn, have obtained reasonable results using only 10 particles.

Consider, as another example, that Clerc and Kennedy [32] and Van den Bergh [33] have derived acceleration coefficients resulting in faster convergence. Very few researchers [18, 25] take these developments into account. Kennedy [26] is particularly interesting, while the newly proposed algorithm (based on Clerc's analysis) takes these developments into account, it is not clear whether the canonical version, to which it is compared, does. Many others (including Secrest and Lamont [29]) seem to use arbi-

trarily determined parameters. On the other hand, some [23] do not even provide values for these coefficients so their results can not be reproduced.

Conflicting results, one of the most serious problems, are highly prevalent in research literature. Consider some papers [18, 23, 24, 25, 26, 27, 28, 29] making comparisons against the canonical particle swarm. The reported results obtained using the canonical algorithm (on the most common functions) were analysed.

In order to not single out specific papers the median me , mean \bar{x} , standard deviation s , and range for the reported results are tabulated in Table 1. The number of papers N making use of a given function is also shown.

Performing this kind of analysis is somewhat dangerous since all the configurations are not identical. However, the results clearly show that something is wrong. Consider the factors that might cause a given configuration to perform poorly.

Firstly, the dimension of the functions are not the same across all the papers. However, this is unlikely to be a significant contributor to the large variance in the results because at least one of the highest dimension configurations was able to perform well in each case. That is, the average error for at least one of the high dimension configurations falls below the median of all the others for a given function. Similarly, for the search domain of a function, at least one of the widest domain configurations performs well for each function.

Another factor is the configuration of the algorithm itself. In which case, the results are in conflict because of poor parameter choices and the previously discussed problem is emphasised.

The only remaining possibility is that some researchers are using broken implementations of the canonical particle swarm.

As a case in point, results for the Quadric function in Peer *et al* [18] are guaranteed to conflict with others. This is due to a minor error (a single subscript was incorrect) in the implementation of Quadric in CILib. The error was picked up during unit testing performed after publication of the paper. It would likely have remained undetected without the independent testing of individual software components.

The point being, these kind of errors are easily made and difficult to detect making it quite believable that broken implementations may be responsible for conflicting results. In any event, the results in Table 1 show that there is indeed a problem somewhere.

Lastly, a very serious problem that is far too prevalent is performing invalid statistical inferences to substantiate claims. Statistical hypothesis testing is a very useful tool for understanding data and as such should be utilised. The danger is, however, that improper use of these statistics can lead to a false sense of confidence.

Some researchers avoid hypothesis testing [24, 26, 27] which is perfectly acceptable provided they do not make strong claims that one algorithm is significantly superior to another. Drawing any such conclusions from experimental results where only the average performance is presented

Table 1: Degree of conflict for each function

Function	N	me	\bar{x}	s	[min, max]
Sphere	8	2.55e-4	1.289e2	3.601e2	[0, 1.02e3]
Rosenbrock	8	7.49e1	1.375e6	3.889e6	[4.07e-12, 1.1e7]
Griewank	7	0.158	0.258	0.379	[1.184e-16, 1.042]
Rastrigin	7	7.22e1	2.178e2	4.105e2	[1.8e-5, 1.143e3]

[23, 25, 28] is unacceptable.

The parametric tests presented in Section 2 can only be used if samples are drawn from normal population distributions. Van den Bergh [33] has found that neural network training and generalisation errors are typically not normally distributed. Testing performed using OptiBench’s KS test revealed that particle swarm optimisers also do not, in general, produce normally distributed sample distributions. For this reason parametric tests should be avoided unless it can be shown that the samples are drawn from normal populations.

Yao *et al* [17] and Peer *et al* [18] have applied the Student’s t -test without satisfactorily showing that the normality constraint is satisfied. Christensen *et al* [16] employ a Monte Carlo simulation technique in an attempt to reduce the dependency on normality. The non-parametric Wilcoxon rank sum test presented in Section 2 is a good alternative.

Secrest and Lamont [29] incorrectly make use of the Kruskal-Wallis (KW) test [20] to conclude that their new algorithm is superior to the canonical particle swarm. The KW test is a $n \geq 2$ sample non-parametric statistic for testing the null hypothesis $H_0 : \eta_1 = \eta_2 = \dots = \eta_n$ against an alternative hypothesis $H_1 : \text{at least one } \eta_i \text{ differs from the others}$. This test can only be used to conclude that the medians are different - not that one is better than the other.

This section has raised many concerns regarding the problems in Section 1 without offering any concrete solutions. OptiBench, discussed in the next section, can potentially alleviate a lot of these issues.

4 OptiBench overview

OptiBench is a tool for benchmarking optimisation algorithms in a statistically sound manner. It is comprised of two separate components. The first of these is a back-end server supporting a database and the management console. The second component (CILib) is a free library of algorithms and problems. CILib can function independently of the server or run as a node in a cluster of workstations performing processing tasks for the server. Some of the key features provided by OptiBench include:

- An extensible open source library of algorithms and problems
- A centralised result repository
- Configuration and set-up of experiments
- Distributed processing

- Data analysis and statistical inferencing engine

What might potentially be perceived as a drawback of the system is that CILib is implemented in the Java programming language. This controversial decision was made very early in development because of the benefits Java provides. Most notably, it is entirely platform independent opening CILib up to as wide an audience as possible. Secondly, the introspective capabilities of the language were necessary to build a truly extensible library. Every aspect of every algorithm and problem in the library can be configured using a single XML document.

The reason Java might be considered a drawback by some is the common misconception that Java programs execute more slowly than compiled C++. It should be noted that the initial author of CILib also believed this but was prepared to sacrifice up to 50% of the performance for the benefits provided by Java. Surprisingly, with the right choice of JVM (Java Virtual Machine) and careful tuning, the JVM can execute code faster than the equivalent compiled C++ code. When tested against the C++ implementations of particle swarms used by Van den Bergh [33] the equivalent Java CILib implementation was able to slightly outperform the compiled C++ code. Obviously, this does not prove that Java is faster than C++, nor is that the intent. However, it does make a convincing argument that Java implementations of computational intelligence algorithms can indeed be good enough.

The authors would like to extend this library (CILib) to the entire community in the hopes that it will be useful. Obviously this is not done for entirely altruistic reasons. Raymond [34] explains the ecology of open source software and how all involved parties can reap benefits from it. By providing the library under an open source license it is hoped that the development of tools to perform effective research will be accelerated. In addition, by using a common code base the community could greatly reduce the incidence of conflicting and incorrect results. Open source software is arguably less buggy since more developers have the opportunity to inspect the code.

By having a shared centralised result repository conflicting results within CIRG@UP are completely eliminated. In addition, different researchers do not duplicate effort by running identical experiments. A more subtle benefit is that results for the latest algorithm developments will also be readily available so that comparisons against outdated versions can also be eliminated.

The management console can be used to configure and set-up experiments. These configurations are also stored in the database so that all the parameters used for a given experiment are recorded. In addition, OptiBench performs

versioning of configurations so that when configurations are altered the results can automatically be recomputed. Altering algorithm parameters is trivialised so that more experimentation with different parameters is encouraged. Hopefully this will alleviate the problem of poor parameter choices somewhat. Configurations are stored in XML fragments that are used to configure CILib objects using introspection. This means that the moment code is added to the library it can be managed and used by OptiBench.

One of the key design requirements of OptiBench was that it be able to scale up to run on a cluster of workstations. This is to address the issue of insufficient testing of problems raised in Section 3. Every time a new algorithm is configured in OptiBench it is automatically scheduled to run on the cluster for every known problem. This is an enormous number of computations that need to take place.

The most powerful feature of OptiBench will be the data analysis and inferencing engine. A lot of work still needs to be done on this area of the system. Initially, OptiBench will employ a method of ranking similar to that described in Mendes *et al* [27]. Each algorithm is ranked on a given problem set according to three criteria.

The first criterion is simply a ranking of the median performance of each algorithm. The second is a ranking of the number of function evaluations needed to reach a predetermined error threshold. The third criterion is a ranking of the number of individuals in a sample that reach this predetermined error threshold.

An algorithm is only ranked above another if it performs significantly better according to the Wilcoxon rank sum test in Section 2. Otherwise, the algorithms are given the same rank but listed in order of the magnitude of their medians. In future, if it is determined that the two samples satisfy the normal assumption then parametric tests about their means will be employed instead. Initially only non-parametric tests will be used.

5 Conclusion

This paper has critically analysed a small sample of recent publications in the particle swarm paradigm. This analysis illustrated the existence of a number of problems that the research community needs to be aware of if any solutions are to be found.

The framework employed internally by CIRG@UP was proposed. This framework is called OptiBench. The open source nature of the library on which OptiBench is based provides solutions to many of the most important problems highlighted in Section 3. Other problems are averted through standardisation of the statistical methods employed by OptiBench.

OptiBench is potentially the first step towards unifying the efforts and harmonising the results published in the computational intelligence community.

Acknowledgement

The financial assistance of the National Research Foundation towards this research is hereby acknowledged. Opin-

ions expressed in this paper and conclusions arrived at, are those of the authors and are not necessarily attributed to the National Research Foundation.

Bibliography

- [1] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1996.
- [2] S. S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd ed., 1998.
- [3] J. Kennedy, R. Eberhart, and Y. Shi, *Swarm Intelligence*. Morgan Kaufmann, 2002.
- [4] T. Bäck, D. B. Fogel, and T. M. (eds), *Evolutionary Computation I: Basic Algorithms and Operations*. IoP Press, 2000.
- [5] T. Bäck, D. B. Fogel, and T. M. (eds), *Evolutionary Computation 2: Advanced Algorithms and Operations*. IoP Press, 2000.
- [6] L. N. de Castro and J. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer Verlag, 2002.
- [7] J. Han and M. Kamber, *Data Mining Concepts and Techniques*. Morgan Kaufmann, 2001.
- [8] R. S. Michalski, I. Bratko, and M. Kubat, *Machine Learning and Data Mining*. Wiley, 1998.
- [9] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing: Analysis and Machine Vision*. Brooks Cole, 2nd ed., 1998.
- [10] G. Weiss, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 2000.
- [11] M. Wooldridge, *Introduction to MultiAgent Systems*. Wiley, 2002.
- [12] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, (Nagoya, Japan), pp. 39–43, 1995.
- [13] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, vol. IV, (Perth, Australia), pp. 1942–1948, 1995.
- [14] D. H. Wolpert and W. G. Macready, "No free lunch theorems for search," Tech. Rep. SFI-TR-95-02-010, Santa Fe Institute, July 1995.
- [15] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, pp. 67–82, 1997.

- [16] S. Christensen and F. Oppacher, "What can we learn from no free lunch? a first attempt to characterize the concept of a searchable function," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, (San Francisco, California), pp. 1219–1226, July 2001.
- [17] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 82–102, July 1999.
- [18] E. S. Peer, F. van den Bergh, and A. P. Engelbrecht, "Using neighbourhoods with the guaranteed convergence pso," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS)*, (Indianapolis, USA), pp. 235–242, April 2003.
- [19] S. S. Wilks, *Mathematical Statistics*. Wiley, 2nd ed., 1963.
- [20] A. G. W. Steyn, C. F. Smit, S. H. C. du Toit, and C. Strasheim, *Modern Statistics in Practice*. J. L. van Schaik, 2nd ed., 1996.
- [21] A. Rutherford, *Introducing Anova and Ancova: A GLM Approach*. Sage Publications, 2001.
- [22] W. H. Press, S. A. Teukolsky, W. Vetterling, and B. P. F. (eds), *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press, 2nd ed., 2002.
- [23] Y. Wang and Y. Jiao, "A novel genetic algorithm using latin squares for numerical optimization," in *Proceedings of the Asia-Pacific Conference on Simulated Evolution and Learning (SEAL)*, (Singapore), 2002.
- [24] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Particle swarm optimization with self-adaptive acceleration coefficients," in *Proceedings of the International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, (Singapore), 2002.
- [25] N. Higashi and H. Iba, "Particle swarm optimization with gaussian mutation," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS)*, (Indianapolis, USA), pp. 72–79, April 2003.
- [26] J. Kennedy, "Bare bones particle swarms," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS)*, (Indianapolis, USA), pp. 80–86, April 2003.
- [27] R. Mendes, J. Kennedy, and J. Neves, "Watch thy neighbor or how the swarm can learn from its environment," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS)*, (Indianapolis, USA), pp. 88–94, April 2003.
- [28] T. Peram, K. Veeramachaneni, and C. K. Mohan, "Fitness-distance-ratio based particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS)*, (Indianapolis, USA), pp. 174–181, April 2003.
- [29] B. R. Secrest and G. B. Lamont, "Visualizing particle swarm optimization – gaussian particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS)*, (Indianapolis, USA), pp. 198–204, April 2003.
- [30] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimisation," in *Proceedings of the IEEE International Congress on Evolutionary computation*, pp. 101–106, 1999.
- [31] F. van den Bergh and A. P. Engelbrecht, "Effects of swarm size on cooperative particle swarm optimisers," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, (San Francisco, USA), July 2001.
- [32] M. Clerc and J. Kennedy, "The Particle Swarm: Explosion, Stability and Convergence in a Multi-Dimensional Complex Space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.
- [33] F. van den Bergh, *An Analysis of Particle Swarm Optimizers*. PhD thesis, Department of Computer Science, University of Pretoria, South Africa, 2002.
- [34] E. S. Raymond, *The Cathedral and the Bazaar: Musings on Linux and Open Source by an accidental revolutionary*. O' Reilly, 2nd ed., 2001.