# Clever Algorithms: Problem Solving Methodology*

Jason Brownlee
jasonb@CleverAlgorithms.com
The Clever Algorithms Project
http://www.CleverAlgorithms.com

December 08, 2010
Technical Report: CA-TR-20101208-1

## Abstract

The Clever Algorithms project aims to describe a large number of Artificial Intelligence algorithms in a complete, consistent, and centralized manner, to improve their general accessibility. The project makes use of a standardized algorithm description template that uses well-defined topics that motivate the collection of specific and useful information about each algorithm described. This report considers methodology for apply Clever Algorithms for solving practical problems.

**Keywords:** `Clever, Algorithms, Problem, Solving, Methodology`

## 1 Introduction

The Clever Algorithms project aims to describe a large number of algorithms from the fields of Computational Intelligence, Biologically Inspired Computation, and Metaheuristics in a complete, consistent and centralized manner [12]. The project requires all algorithms to be described using a standardized template that includes a fixed number of sections, each of which is motivated by the presentation of specific information about the technique [13].

The field of Data Mining has a clear methodologies that guides a practitioner to solve problems, such as Knowledge Discovery in Databases (KDD) [20]. Metaheuristics and Computational Intelligence algorithms have no such methodology.[1]

This report describes some of the considerations when applying algorithms from the fields of Metaheuristics, Computational Intelligence, and Biologically Inspired Computation to practical problem domains. This discussion includes:

- The suitability of application of a given technique to a given problem and the transferability of algorithm and problem features (Section 2)
- The distinction between strong and weak methods that describes a continuum of methods that use more or less problem specific information respectively (Section 3).
- A summary of problem solving strategies that suggest different ways of applying a given technique in the fields of function optimization and approximation (Section 4).

*© Copyright 2010 Jason Brownlee. Some Rights Reserved. This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 2.5 Australia License.

[1]Some methods can be used for classification and regression and as such may fit into methodologies such as KDD.

## 2 Suitability of Application

From a problem-solving perspective, the tools that emerge from the field of Computational Intelligence are generally assessed with regard to their utility as *efficiently* or *effectively* solving problems. An important lesson from the 'no-free-lunch theorem' was to *bound claims of applicability.* An approach toward this end is to consider the suitability of a given strategy with regard to the feature overlap with the attributes of a given problem domain. From a Computational Intelligence perspective, one may consider the architecture, processes, and constraints of a given strategy as the features of an approach.

The suitability of the application of an 'approach' to a 'problem' takes into considerations concerns such as the *appropriateness* (can the approach address the problem), the *feasibility* (available resources and related efficiency concerns), and the *flexibility* (ability to address unexpected or unintended effects). This section summarizes a general methodology toward addressing the problem of suitability in the context of Computational Intelligence tools. This methodology involves (1) the systematic elicitation of system and problem features, and (2) the consideration of the overlap of problem-problem, algorithm-algorithm, and problem-algorithm overlap of feature sets.

### 2.1 Systematic Feature Elicitation

A *feature* of a system (tool, strategy, model) or a problem is a distinctive element or property that may be used to differentiate it from similar and/or related cases. Examples may include functional concerns such as: processes, data structures, architectures, and constraints, as well as emergent concerns that may have a more subjective quality such as general behaviors, organizations, and higher-order structures. The process of the elicitation of features may be taken from a system or problem perspective:

- *System Perspective*: This requires a strong focus on the lower level functional elements and investigations that work toward correlating specific controlled organizations towards predictable emergent behaviors.

- *Problem Perspective*: May require both a generalization of the specific case to the general problem case, as well as a functional or logical decomposition into constituent parts.

Problem *generalization* and *functional decomposition* are important and well used patterns for problem solving in the broader fields of Artificial Intelligence and Machine Learning as the promotion of simplification and modularity can reduce the cost and complexity of achieving solutions [47, 10].

### 2.2 Feature Overlap

Overlap in elicited features may be considered from three important perspectives: *between systems*, *between problems*, and *between a system and a problem.* Further, such overlap may be considered at different levels of detail with regard to generalized problem solving strategies and problem definitions. These overlap cases are considered as follows:

- *System Overlap*: Defines the suitability of comparing one system to another, referred to as *comparability.* For example, systems may be considered for the same general problems and compared in terms of theoretical or empirical capability, the results of which may only be meaningful if the systems are significantly similar to each other as assessed in terms of feature overlap.

- *Problem Overlap*: Defines the suitability of comparing one problem to another, referred to as *transferability.* From a systems focus for example, transferability refers to the capability of a technique on a given problem to be transferred to another problem, the result of which is only meaningful if there is a strong overlap between the problems under consideration.

- *System-Problem Overlap*: Defines the suitability of a system on a given problem, referred to as *applicability*. For example, a system is considered suitable for a given problem if it has a significant overlap in capabilities with the requirements of the problem definition.

Such mappings are expected to have noise given the subjective assessment and/or complexity required in both the elicitation and consideration overlap of the of features, the noisiest of which is expected to be the mapping between systems and problems. The mapping of salient features of algorithms and problems was proposed as an important reconciliation of the 'no-free-lunch theorem' by Wolpert and Macready [62], although the important difference of this approach is that the system and algorithm are given prior to the assessment. In [61], Wolpert and Macready specifically propose the elicitation of the features from a problem-first perspective, for which specialized algorithms can be defined. Therefore, this methodology of suitability may be considered a generalization of this reconciliation suitable for the altered 'Computational Intelligence' (strategy first) perspective on Artificial Intelligence.

## 3 Strong and Weak Methods

Generally, the methods from the fields of Metaheuristics, Computational Intelligence, and Biologically Inspired Computation may be considered weak methods. They are weak because they are general purpose problem solves that are typically considers black-box solvers for a range of problem domains. The stronger the method, the more that must be known about the problem domain. Rather than discriminating techniques into weak and strong is not a useful categories, it is more useful to consider a continuum of methods from pure block box techniques that have none or few assumptions about the problem domain, to strong methods that exploit most or all of the problem specific information available.

For example, the Traveling Salesman Problem is an example of a combinatorial optimization problem. A naïve (such a Random Search) black box method may simple explore permutations of the cities. Slightly stronger methods may initialize the search with a heuristic-generated technique (such as nearest neighbor) and explore the search space using a variation method that also exploits heuristic information about the domain (such as a 2-opt variation). Continuing along this theme, a stochastic method may explore the search space using a combination of probabilistic and heuristic information (such as Ant Colony Optimization algorithms). At the other end of the scale the stochastic elements are decreased or removed until one is left with pure heuristic methods such as the Lin-Kernighan heuristic [35] and exact algorithms from linear and dynamic programming that focuses on the structure and nature of the problem [59].

Approaching a problem is not as simple as selecting the strongest method available and solving it. The following describes two potential strategies:

- *Start Strong*: Select the strongest technique available and apply it to the problem. Difficult problems can be resistant to traditional methods for many intrinsic and extrinsic reasons. Use products from a strong technique (best solution found, heuristics) to seed the next weaker method in line.

- *Start Weak*: Strong methods do not exist for all problems, and if they do exist, the computation, skill, and/or time resources may not be available to exploit them. Start with a weak technique and use it to learn about the problem domain. Use this information to make better decisions about subsequent techniques to try that can exploit what has been learned.

In a real-world engineering or business scenario, the objective is to solve a problem, or achieve the best possible solution to the problem within the operating constraints. Concerns of algorithm and technique purity become less important as they may be in their respective fields of research. Both of the above strategies suggest an iterative methodology, where the product or knowledge gained from one technique may be used to prime a subsequent stronger or weaker technique.

## 4 Domain-Specific Strategies

An algorithm may be considered a strategy for problem solving. There are a wide range of ways in which a given algorithm can be used to solve a problem. Function Optimization and Function Approximation were presented as two general classes of problems to which the algorithms from the fields of Metaheuristics, Computational Intelligence, and Biologically Inspired Computation are applied [14]. This section reviews general problem problem solving strategies that may be adopted for a given technique in each of these general problem domains.

### 4.1 Function Optimization

This section reviews a select set of strategies for addressing optimization problems from the field of Metaheuristics and Computational Intelligence to provide general insight into the state of the interaction between stochastic algorithms and the field of optimization. This section draws heavily from the field of Evolutionary Computation, Swarm Intelligence, and related Computational Intelligence sub-fields.

#### 4.1.1 Global and Local Optimization

Global Optimization refers to seeking a globally optimal structure or approximation thereof in a given problem domain. Global is differentiated from Local Optimization in that the latter focuses on locating an optimal structure within a constrained region of the decision variable search space, such as a single peak or valley (basin of attraction). In the literature, global optimization problems refers to the class of optimization problems that generally cannot be addressed through more conventional approaches such as gradient decent methods (that require derivatives), and pattern search (that can get ‘stuck’ in local optima and/or may never converge) [45, 57].

A global search strategy provides the benefit of making few if any assumptions about where promising areas of the search space may be, potentially highlighting unintuitive combination’s of parameters. A local search strategy provides the benefit of focus and refinement of an existing candidate solution. It is common to apply a local search method to the solutions located by a global search procedure as an refinement strategy (such as using a Hill Climber after a Genetic Algorithm), and some methods have both techniques built in (such as GRASP).

#### 4.1.2 Parallel Optimization

A natural first step in addressing difficult (large and rugged cost landscapes) is to exploit parallel and distributed hardware, to get an improved result in the same amount of time, or the same result in less time, or both [16]. Towards unifying the myriad of approaches and hardware configurations, a general consensus and taxonomy has been defined by the Parallel Evolutionary Algorithms (PEA) and Parallel Metaheuristics fields that considers the ratio of *communication* to *computation* called *granularity* [15, 4].

This taxonomy is presented concisely by Alba and Tomassini as a plot or trade-off of three concerns: (1) the number of sub-populations (models or parallel strategies working on the problem), (2) the coupling between the sub-populations (frequency and amplitude of communication), and (3) the size of the sub-populations (size or extent of the sub-models) [5].

Two important and relevant findings from the narrower field of Parallel Evolutionary Algorithms include (1) that tightly coupled (frequent migration) between coarse-grained models typically results in worse performance than a non-distributed approach [6], and (2) that loose coupling (infrequent migration) between coarse-grained models has been consistently shown to provide a super-linear increase in performance [3, 7, 15].

### 4.1.3 Cooperative Search

This is a more general approach that considers the use of multiple models that work together to address a difficult optimization problems. Durfee, et al. consider so-called *Cooperative Distributed Problem Solving (CDPS)* in which a network of loosely coupled solvers are employed to address complex distributed problems. Specifically, such systems are desirable to match the processing capabilities of the solver with regard to the attributes of the problem. For example, a given problem may have spatially distributed, functionally distributed, or temporally distributed sub-problems to which a centralized and monolithic system may not be suitable.

Lesser considers CDPS and proposes such models perform *distributed search* on dependent or independent and potentially overlapping sub-problems as a motivating perspective for conducting research into Distributed Artificial Intelligence (DAI)[2] [34]. Lesser points out that in real world applications, it is hard to get a optimal mapping between the allocated resources to the needs or availability of information for a given problem, suggesting that such problems may be caused by a mismatch in processing times and/or number of sub-problems, interdependencies between sub-problems, and local experts whose expertise cannot be effectively communicated. For a more detail on the relationships between parallel and cooperative search, El-Abd and Kamel provide a rigorous taxonomy [19].

### 4.1.4 Hybrid Search

Hybrid Search is a perspective on optimization that focuses on the use of multiple and likely different approaches either sequentially (as in the canonical global and local search case), or in parallel (such as in Cooperative Search). For example in this latter case, it is common in the field of PEA to encourage different levels of exploration and and exploitation across island populations by varying the operators or operator configurations used [55, 2].

Talbi proposed a detailed 4-level taxonomy of Hybrid Metaheuristics that encompassed concerns of parallel and cooperating approaches [54]. The taxonomy encompasses parallel and cooperative considerations for optimization and focuses on the discriminating features in the lowest level such as heterogeneity, and specialization of approaches.

### 4.1.5 Functional Decomposition

Three examples of a functional decomposition of optimization include (1) multiple objectives, (2) multiple constraints, and (3) partitions of the decision variable search space.

Multi-Objective Optimization (MOO) is a sub-field that is concerned with the optimization of decision variables under a set of objective functions. A solution to a MOO conventionally involves locating and returning a set configurations under the objective functions called the optimal or non-dominated Pareto set of solutions [17]. The complexity with MOO problems is in the typically unknown dependencies between decision variables across objectives, that in the case of conflicts, must be traded off (Purshouse and Fleming provide a taxonomy of such complexity [46]).

[2]This perspective provided the basis for what became the field of Multi-Agent Systems (MAS).

Constraint Satisfaction Problem's (CSP) involve the optimization of decision variables under a set of constraints. The principle complexity in such problems is in locating structures that are feasible or violate the least number of constraints, potentially optimizing such feasibility [58, 31].

Search Space Partitioning involves partitioning of the decision variable search space (for example see Multispace Search by Gu, et al [18, 26, 25]). This is a critical consideration given that for equal-sized dimensions bounds on parameters, the increase in decision variables results in an exponential increase in the volume of the space to search.

### 4.1.6 Availability Decomposition

Optimization problems may be partitioned by the concerns of temporal and spatial distribution of (1) information availability, and (2) computation availability. An interesting area of research regarding variable information availability for optimization problems is called Interactive Evolutionary Computation, in which one or a collection of human operators dynamically interact with an optimization process [53]. Example problem domains include but are not limited to computer graphics, industrial design, image processing, and drug design.

There is an increasing demand to exploit clusters of heterogeneous workstations to complete large-scale distributed computation tasks like optimization, typically in an opportunistic manner such as when individual machines are under utilized. The effect is that optimization strategies, such as random partitioning of the search space (independent non-interacting processing) are required to take advantage of such environments for optimization problems [50, 36].

### 4.1.7 Meta Optimization

One may optimization at a level above that considered in previous sections. Specifically, (1) the iterative generation of an inductive model called multiple restart optimization, and (2) the optimization of the parameters of the process that generates an inductive model of an optimization problem. Multiple or iterative restarts involves multiple independent algorithm executions from different (random) starting conditions. It is generally considered as an method for achieving an improved result in difficult optimization problems in which a given strategy is deceived by local or false optima [38, 28], typically requiring a restart schedule [21].

A second and well studied form of meta optimization involves the optimization of the search process itself. Classical examples include the self-adaptation of mutation parameters (step sizes) in the Evolutionary Strategy and Evolutionary Programming approaches. Smith and Fogarty provided a review of genetic algorithms with adaptive strategies providing a taxonomy in which the meta-adaptations are applied at one of three levels: (1) the population (adapting the overall sampling strategy), (2) the individual (adapting the creation of new samples in the decision variable space), and (3) components (modifying component contributions and/or individual step sizes as in ES and EP) [52].

## 4.2 Function Approximation

This section reviews a select set of strategies for addressing Function Approximation problems from the field of Artificial and Computational Intelligence to to provide general insight into the state of the interaction between stochastic algorithms and the field. The review draws heavily from the fields of Artificial Neural Networks, specifically Competitive Learning, as well as related inductive Machine Learning fields such as Instance Based Learning.

### 4.2.1 Vector Quantization

Vector Quantization (VQ) refers to a method of approximating a target function using a set of exemplar (prototype or codebook) vectors. The exemplars represent a discrete sub-set of the problem, generally restricted to the features of interest using the natural representation of the observations in the problem space, typically an an unconstrained $n$-dimensional real valued space. The VQ method provides the advantages of a non-parametric model of a target function (like instance-based and lazy learning such as $k$NN) using a symbolic representation that is meaningful in the domain (like tree-based approaches).

The promotion of compression addresses the storage and retrieval concerns of $k$NN, although the selection of codebook vectors (the so-called quantization problem) is a hard problem that is known to be NP-complete [22]. More recently Kuncheva and Bezdek have worked towards unifying quantization methods in the application to classification problems, referring to the approaches as Nearest Prototype Classifiers (NPC) and proposing a generalized nearest prototype classifier [32, 33].

### 4.2.2 Parallelization

Instance-based approaches are inherently parallel given the generally discrete independent nature in which they are used, specifically in a case or per-query manner. As such, parallel hardware can be exploited in the preparation of the corpus of prototypes (parallel preparation), and more so in the application of the corpus given its read-only usage [1, 39, 43]. With regard to vector quantization specifically, there is an industry centered around the design and development of VQ and WTA algorithms and circuits given their usage to compress digital audio and video data [40, 42].

### 4.2.3 Cooperative Methods

Classical cooperative methods in the broader field of statistical machine learning are referred to as *Ensemble Methods* [41, 44] or more recently *Multiclassifier Systems* [24].

*Boosting* is based on the principle of combining a set of quasi-independent weak learners that collectively are as effective as a single strong learner [30, 48]. The seminal approach is called Adaptive Boosting (AdaBoost) that involves the preparation of a series of classifiers, where subsequent classifiers are prepared for the observations that are misclassified by the proceeding classifier models (creation of specialists) [49].

*Bootstrap Aggregation* (bagging) involves partitioning the observations into $N$ randomly chosen subsets (with re-selection), and training a different model on each [9]. Although robust to noisy datasets, the approach requires careful consideration as to the consensus mechanism between the independent models for decision making.

*Stacked Generalization* (stacking) involves creating a sequence of models of generally different types arranged into a stack, where subsequently added models generalize the behavior (success or failure) of the model before it with the intent of correcting erroneous decision making [60, 56].

### 4.2.4 Functional Decomposition

As demonstrated, it is common in ensemble methods to partition the dataset either explicitly or implicitly to improve the approximation of the underlying target function. A first important decomposition involves partitioning the problem space into sub-spaces based on the attributes, regular groups of attributes called features, and decision attributes such as class labels. A popular method for attribute-based partitioning is called the Random Subspace Method, involving the random partitioning of attributes to which specialized model is prepared for each (commonly used on tree-based approaches) [27].

A related approach involves a hierarchical partitioning of attributes space into sub-vectors (sub-spaces) used to improve VQ-based compression [23]. Another important functional decomposition involves the partitioning of the set of observations. The are many ways in which observations may be divided, although common approaches include pre-processing using clustering techniques to divide the set into natural groups, additional statistical approaches that partition based on central tendency and outliers, and re-sampling methods that are required to reduce the volume of observations.

### 4.2.5 Availability Decomposition

The availability observations required to address function approximation in real-world problem domains motivate the current state of the art in Distributed Data Mining (DDM or so-called Collective Data Mining), Parallel Data Mining (PDM), and Distributed Knowledge Discovery in Database (DKDD) [29]. The general information availability concerns include (1) the *intractable volume of observations*, and (2) the *spatial (geographical) and temporal distribution of information* [63]. It is common in many large real-world problems for it to be infeasible to centralize relevant observations for modeling, requiring scalable, load balancing, and incremental acquisition of information [51].

### 4.2.6 Meta Approximation

The so-called ensemble or multiple-classifier methods may be considered meta approximation approaches as they are not specific to a given modeling technique. As with function optimization, meta-approaches may be divided into restart methods and meta-learning algorithms. The use of restart methods is a standard practice for connectionist approaches, and more generally in approaches that use random starting conditions and a gradient or local search method of refinement.

The method provides an opportunity for over-coming local optima in the error-responds surface, when there is an unknown time remaining until convergence [37], and can exploit parallel hardware to provide a speed advantage [8]. Ensemble methods and variants are examples of meta approximation approaches, as well as the use of consensus classifiers (gate networks in mixtures of experts) to integrate and weight the decision making properties from ensembles.

## 5 Conclusions

This report provided a discussion of some considerations when applying Metaheuristic and Computational Intelligence algorithms. Some of the content of this report was derived from the dissertation work of the author [11].

## References

[1] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications*, 7(1):39–59, 1994.

[2] P. Adamidis, P. Adamidis, and V. Petridis. Co-operating populations with different evolution behaviours. In V. Petridis, editor, *Proc. IEEE International Conference on Evolutionary Computation*, pages 188–191, 1996.

[3] E. Alba. Parallel evolutionary algorithms can achieve super-linear performance. *Information Processing Letters*, 82:7–13, 2002.

[4] E. Alba. *Parallel Metaheuristics: A New Class of Algorithms*. John Wiley, 2005.

[5] E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443–462, 2002.

[6] E. Alba and J. M. Troya. Influence of the migration policy in parallel distributed gas with structured and panmictic populations. *Applied Intelligence*, 12:163–181, 2000.

[7] T. C. Belding. The distributed genetic algorithm revisited. In *Proceedings of the 6th International Conference on Genetic Algorithms*, 1995.

[8] A. Di Blas, A. Jagota, and R. Hughey. Optimizing neural networks on simd parallel computers. *Parallel Computing*, 31:97–115, 2005.

[9] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[10] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal Of Robotics And Automation*, 2(1):14–23, 1986.

[11] Jason Brownlee. *Clonal Selection as an Inspiration for Adaptive and Distributed Information Processing*. PhD thesis, Complex Intelligent Systems Laboratory, Faculty of Information and Communication Technologies, Swinburne University of Technology, 2008.

[12] Jason Brownlee. The clever algorithms project: Overview. Technical Report CA-TR-20100105-1, The Clever Algorithms Project http://www.CleverAlgorithms.com, January 2010.

[13] Jason Brownlee. A template for standardized algorithm descriptions. Technical Report CA-TR-20100107-1, The Clever Algorithms Project http://www.CleverAlgorithms.com, January 2010.

[14] Jason Brownlee. Unconventional optimization algorithms: An introduction. Technical Report CA-TR-20100118-1, The Clever Algorithms Project http://www.CleverAlgorithms.com, January 2010.

[15] E. Cantu-Paz. *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Publishers (Springer), 2000.

[16] T. G. Crainic and N. Hail. Parallel metaheuristics applications. In *Parallel Metaheuristics*. John Wiley & Sons, Inc., 2005.

[17] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley and Sons, 2001.

[18] B. Du, B. Du, J. Gu, W. Wang, and D.H.K. Tsang. Multispace search for minimizing the maximum nodal degree. In Jun Gu, editor, *Proc. Sixth International Conference on Computer Communications and Networks*, pages 364–367, 1997.

[19] M. El-Abd and M. Kamel. A taxonomy of cooperative search algorithms. In *Hybrid Metaheuristics*, 2005.

[20] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, 1996.

[21] A. S. Fukunaga. Restart scheduling for genetic algorithms. In *Parallel Problem Solving from Nature - PPSN V*, 1998.

[22] M. Garey, D. Johnson, and H. Witsenhausen. The complexity of the generalized lloyd - max problem (corresp.). *IEEE Transactions on Information Theory*, 28(2):255–256, Mar 1982.

[23] A. Gersho, A. Gersho, and Y. Shoham. Hierarchical vector quantization of speech with dynamic codebook allocation. In Y. Shoham, editor, *Proc. IEEE International Conference on ICASSP '84. Acoustics, Speech, and Signal Processing*, volume 9, pages 416–419, 1984.

[24] J. Ghosh. Multiclassifier systems: Back to the future. In *Proceedings of the Third International Workshop on Multiple Classifier Systems*, 2002.

[25] J. Gu. Multispace search: A new optimization approach. In *Algorithms and Computation*, 1994.

[26] J. Gu. Multispace search for satisfiability and np-hard problems. In *Satisfiability Problem: Theory and Applications : DIMACS Workshop*, 1997.

[27] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:832–844, 1998.

[28] X. Hu, R. Shonkwiler, and M. Spruill. Random restarts in global optimization. Technical Report Tech Report 110592-015, School of Mathematics, Georgia Institute of Technology, January 1994.

[29] H. Kargupta and P. Chan. *Advances in Distributed and Parallel Knowledge Discovery.* AAAI Press/MIT Press, 2000.

[30] M. Kearns. Thoughts on hypothesis boosting. Unpublished manuscript, 1988.

[31] V. Kumar. Algorithms for constraint-satisfaction problems : a survey. *The AI magazine (AI mag.)*, 13, 1992.

[32] L. Kuncheva and J. Bezdek. An integrated framework for generalized nearest proto- type classifier design. *Int. Journal of Uncertaintly, Fuzziness and Knowledge-Based Systems*, 6:437–457, 1998.

[33] L.I. Kuncheva and J.C. Bezdek. Nearest prototype classification: clustering, genetic algorithms, or random search? *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 28(1):160–164, 1998.

[34] V.R. Lesser. An Overview of DAI: Viewing Distributed AI as Distributed Search. *Journal of Japanese Society for Artificial Intelligence-Special Issue on Distributed Artificial Intelligence*, 5(4):392–400, January 1990.

[35] Shen Lin and B. W Kernighan. An effective heuristic algorithm for the traveling-salesman problems. *Operations Research*, 21:498–516, 1973.

[36] P. Liu, P. Liu, and D. W. Wang. Reduction optimization in heterogeneous cluster environments. In Da-Wei Wang, editor, *Proc. 14th International Parallel and Distributed Processing Symposium IPDPS 2000*, pages 477–482, 2000.

[37] M. Magdon-ismail and A. F. Atiya. The early restart algorithm. *Neural Computation*, 12:1303–1312, 2000.

[38] M. Muselli. A theoretical approach to restart in global optimization. *Journal of Global Optimization*, 10:1–16, 1997.

[39] M. V. Nagendra Prasad, V. R. Lesser, and S. E. Lander. Retrieval and reasoning in distributed case bases. *Journal of Visual Communication and Image Representation, Special Issue on Digital Libraries*, 7(1):74–87, 1996.

[40] A. Nakada, A. Nakada, T. Shibata, M. Konda, T. Morimoto, and T. Ohmi. A fully parallel vector-quantization processor for real-time motion-picture compression. *IEEE Journal of Solid-State Circuits*, 34(6):822–830, 1999.

[41] D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

[42] K.K. Parhi, K.K. Parhi, F.H. Wu, and K. Genesan. Sequential and parallel neural network vector quantizers. *IEEE Transactions on Computers*, 43(1):104–109, 1994.

[43] E. Plaza and J. Llus A. F. Martn. Cooperative case-based reasoning. In *Distributed Artificial Intelligence Meets Machine Learning Learning in Multi-Agent Environments*, 1997.

[44] R. Polikar. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):21–45, 2006.

[45] W. L. Price. A controlled random search procedure for global optimisation. *The Computer Journal*, 20(4):367–370, 1977.

[46] R. C. Purshouse and P. J. Fleming. Conflict, harmony, and independence: Relationships in evolutionary multi-criterion optimisation. In *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization (EMO)*, pages 16–30, 2003.

[47] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, third edition, 2009.

[48] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1992.

[49] R. E. Schapire. The boosting approach to machine learning: An overview. In D. D. Denison, M. H. Hansen, C. Holmes, B. Mallick, and B. Yu, editors, *Nonlinear Estimation and Classification*, 2003.

[50] T. Schnekenburger. Parallel randomized algorithms in heterogeneous environments. In *Int. Conference on Systems Engineering*, 1993.

[51] D. Skillicorn. Strategies for parallel data mining. *IEEE Concurrency*, 7(4):26–35, 1999.

[52] J. E. Smith and T. C. Fogarty. Operator and parameter adaptation in genetic algorithms. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 1:81–87, 1997.

[53] H. Takagi. Interactive evolutionary computation: fusion of the capabilities of ec optimization and human evaluation. *Proceedings of the IEEE*, 9(89):1275–1296, September 2001.

[54] E. Talbi. A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8:541–564, 2001.

[55] R. Tanese. Distributed genetic algorithms. In *Proceedings of the third international conference on Genetic algorithms*, 1989.

[56] K. M. Ting and I. H. Witten. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289, 1999.

[57] A. Törn, M.M. Ali, and S. Viitanen. Stochastic global optimization: Problem classes and solution techniques. *Journal of Global Optimization*, 14:437–447, 1999.

[58] E. Tsang. *Foundations of Constraint Satisfaction.* Academic Press, 1993.

[59] G. J. Woeginger. Exact algorithms for np-hard problems: A survey. *Combinatorial Optimization – Eureka, You Shrink!*, 2570:185–207, 2003.

[60] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(LA-UR-90-3460):241–259, 1992.

[61] D. H. Wolpert and W. G. Macready. No free lunch theorems for search. Technical report, Santa Fe Institute, Sante Fe, NM, USA, 1995.

[62] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(67):67–82, 1997.

[63] M. J. Zaki. Parallel and distributed data mining: An introduction. In *Revised Papers from Large-Scale Parallel Data Mining, Workshop on Large-Scale Parallel KDD Systems, SIGKDD*, pages 1–23, 1999.