# A Cost Benefit Operator for Efficient Multi Level Genetic Algorithm Searches

George G. Mitchell, Barry McMullin, James Decraene

***Abstract*— In this paper we present a novel cost benefit operator that assists multi level genetic algorithm searches. Through the use of the cost benefit operator, it is possible to dynamically constrain the search of the base level genetic algorithm, to suit the user's requirements. Initially we review meta-evolutionary (multi-level genetic algorithm) approaches. We note that the current literature has abundant studies on meta-evolutionary GAs. However these approaches have not identified an efficient approach to termination of base GA search or a means to balance practical consideration such as quality of solution and the expense of computation. Our *Quality time tradeoff* operator (QTT) is user defined, and acts as a base level termination operator and also provides a fitness value for the meta-level GA. In this manner the amount of computation time spent on less encouraging configurations can be specified by the user. Our approach has been applied to a computationally intensive test problem which evaluates a large set of configuration settings for the base GAs. This approach should be applicable across a wide range of practical problems (e.g. routing, logistic and biomedical applications).**

## I. Introduction

A number of different approaches have been taken to the development of Genetic Algorithms for different application areas. The genetic algorithms typically consist of a number of core parts: representation, fitness evaluation, crossover and mutation operators. Many differing forms of operators have been developed. These include for crossover: Order Crossover [1], Modified Crossover [2], Partially Mapped Crossover [3], Cycle Crossover [4], 2-quick / 2-repair [5], plus a number of less frequently used crossover operators [6]. The mutation operators include: Displacement Mutation [7], Exchange Mutation [8], Insertion Mutation [7, 9], Simple Inversion Mutation [10, 11], Inversion Mutation [12, 13] and other less well known mutation operators [14].

When encountering a new problem that has not been evaluated with a genetic algorithm, one is presented with the question: How does one select appropriate genetic operators and parameters to obtain the most desirable solution?

This question can be addressed through the use of parameter-less genetic algorithms and meta-GAs'. These techniques can help in the selection of efficient parameters for a given problem.

Such parameter-less techniques include: The Harik's *parameter-less genetic algorithm* (PLGA) [15] and the *meta-evolutionary approach* proposed by Mernik [16]. However, these techniques present significant limitations. For example Harik only focused on a single parameter: population size. Mernik evaluated only a small group of crossover operators and their associated crossover rates. Similarly Freisleben in 1993 [17] used a meta-evolutionary approach to determine components (selection method, elitist model and a restricted set of crossover and mutation methods) and parameters (population size, crossover probability, mutation probability) for a genetic algorithm to solve instances of the traveling salesman problem.

All of the foregoing techniques have attempted to identify the optimal configuration settings (parameters and operator settings) for genetic algorithms applied to a particular problem instance.

A modification to the traditional meta-evolutionary approach [17] is the multi-level genetic algorithm. In a similar way to the meta-evolutionary search, the meta level (or level one GA) controls a population of candidate solutions. Each of these solutions is in turn evaluated through the use of a Base level (or level two GA). The multi-level GA is not limited to only two levels in the hierarchy and can have as many levels as are required to reach the minimal computation unit required. This form of system has been used to model biological networks [18].

One practical issue that exists in all of these approaches is how to balance the large amount of computational effort (time, space etc) that is required to evaluate parameters and operators that yield quality solutions that meet a user's needs.

We have developed a multi level genetic algorithm, in essence combining the best of Harik's and Freisleben's techniques together with a *cost benefit* (Quality-Time Tradeoff) operator.

Manuscript received March 15, 2007. This work was supported in part by the European Union as part of the FP6 ESIGNET project contract No. 12789.

G. G. Mitchell is with the Research Institute for Network Communications Engineering (RINCE), Dublin City University, Dublin 9, Ireland. (phone: ++353-1-7005000; fax: ++353-1-7007767; e-mail: george.mitchell@dcu.ie or geo.mit@gmail.com).

B. McMullin is with the Research Institute for Network Communications Engineering (RINCE), Dublin City University, Dublin 9, Ireland. (phone: ++353-1-7005000; fax: ++353-1-7007767; e-mail: barry.mcmullin@dcu.ie).

J. Decraene is with the Research Institute for Network Communications Engineering (RINCE), Dublin City University, Dublin 9, Ireland. (phone: ++353-1-7005000; fax: ++353-1-7007767; e-mail: james.decraene@eeng.dcu.ie).



1-4244-1340-0/07/$25.00 ©2007 IEEE

The multi level genetic algorithm approach can be summarized by the following properties:

- It uses and evaluates a wide variety of genetic operators selected from the *Path*, *Binary* and *Adjacency* representations.
- It evolves parameters such as population size, crossover rate, mutation rate, constraint handling techniques, type of crossover and mutation operators and selection technique. This removes the need of users to be experts in the field of designing GAs.
- It provides a cost benefit function that performs a dual role of termination condition (base level genetic algorithm) and fitness evaluation (meta level genetic algorithm).
- Most importantly the cost benefit function is specified by the user. The latter can define a tradeoff between quality of solution and the amount computation effort expended. This allows one to incorporate realistic commercial constraints on searching for solutions.

The paper examines the multi-level GA, and then introduces our Quality-Time Tradeoff operator and its potential application areas. Following this we present a case study applying the multi level genetic algorithm with the Quality time tradeoff operator to one problem instance.

## II. Multi Level Genetic Algorithm

A number of different Genetic Algorithm variants have been developed that attempt to take advantage of distributed computing properties, these can take a number of different forms: distributed GA[19], parallel GA[20], Divide and Conquer GA [21, 22] and Multi level GA [18, 23].

However, all of these approaches rely on some predetermined terminating condition (in the form of a fixed number of time steps, reaching a solution quality threshold, or reaching an evolutionary plateau for example). An illustration of a meta-evolutionary genetic algorithm (which is a multi level genetic algorithm with only two levels) that we have used is presented in figure 1.

These approaches while being widely accepted in the computational optimization field, present significant problems from a practical point of view. Indeed in the case of specific critical applications (where we have a constraint on time, financial resources, computational effort, etc), an efficient balance between the quality and time tradeoff is required.

To address this quality time tradeoff issue, we designed a *n*-level GA coupled with the Quality-Time Tradeoff mechanism (see figure 1). The latter is a technique which permits the user to decide what they consider to be the most important factor for their search: i.e. a *tradeoff* between the quality of solution versus speed of return of solution. We term the operator that accomplishes this, the QTT Q*uality-Time Tradeoff* operator.

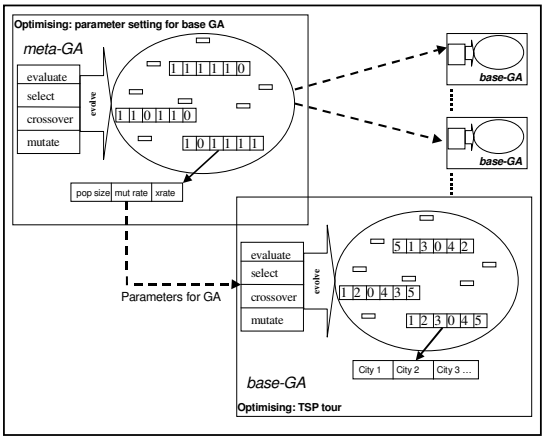


Fig. 1. Multi level genetic algorithm for identifying *cost effective* parameters for GAs solving the TSP.

## III. Quality-Time Tradeoff Operator

The use of a *tradeoff* operator (cost-benefit operator) for optimization problems is a novel approach for the GA community, however within the optimization research area it has been used for some years. Sosič [24] developed a tradeoff operator for a *local optimization algorithm, Duty (as termed by* Sosič) which minimized the *excess* (error) of a present solution compared to the benchmark known optimal solution. An optimal present solution would be found to have an excess of zero. The generalized Duty measure gives equal weight to the quality of the solution and the computing time as follows:

$$D_k(t) = t^k * E(t) \quad (1)$$

The tradeoff operator used in the experiment of the multi level GA was as follows:

$$\text{tradeoff} = t + wQ \quad (2)$$

Where *t*, the computation time is measured as the number of generations, *Q* the quality of the solution is the raw fitness value for the individual solution and *w* the weighting factor placed on the quality of the solution. By weighting the quality of solutions the tradeoff operator can be set by the user to terminate the solution at the desired *cost benefit point* (where time spent searching for a better solution does not outweigh the improvement in the quality of the solution).

The tradeoff operator should produce a 'u' shaped curve and then by identifying the minimum of this curve it is possible to halt the base level GA search. The tradeoff operator accomplishes this through the use of two key mechanisms:

1. A low pass filter (for a minimization problem) is applied to the solution fitness values generated by the base level GA. This determines *the best solution found so far*. This best solution is computed in real-time.
2. A sliding window grace period is applied. This gives an opportunity to encounter fitter candidate solutions. The size of the window is user defined and affects the running time of the search. Typically a small window returns a result more rapidly. This is tightly coupled to the *quality time tradeoff*.

## IV. Case Study

We present a case study to explore configuration settings for genetic algorithms applied to selected TSP problems. Firstly we describe the traveling salesman problem, and then we present how the quality tradeoff operator is instantiated to solve this problem. Finally we present the result obtained from our multi level genetic algorithm which uses the tradeoff operator to balance quality and time efficiently.

### A. Quality Time Tradeoff and TSP configuration setting.

Hamiltonian search optimization has for some time remained one of the key benchmarks for any optimization algorithms [25]. The Traveling Salesman Problem (TSP) is a form of Hamiltonian problem. The TSP consists of a minimal distance Hamiltonian cycle of a complete graph visiting all $N$ nodes (where $N$ represents the number of cities in the problem). The TSP is a classical example of an NP-hard problem and exhibits a factorial growth in number of search size as the number of cities in the problem increases. In cases where $N$ is very large an algorithm which generates near-optimal solutions is at present the most applicable search technique. Many different approaches have been taken to finding sub and near optimal solutions for the TSP such as: 2-opt, 3-opt, Ant colony, Tabu search, multiple heuristic search enhanced GA and many more [9, 26-30].

With a genetic algorithm applied to the TSP some implementation details differ from the more general genetic algorithm. In particular these concern the construction of solutions that are valid according to the constraints of the TSP - (visit all $N$ cities once and once only). It is common for genetic algorithms to have some form of validity preserving operators. These can be validity preserving crossover and mutation operators that do not allow the creation of invalid tours (according to the TSP constraints) or some form of repair, penalty or decoder function can be used [6].

#### 1) QTT setup.

The initial step is to instantiate a suitable QTT for the optimization problem at hand, here this is the TSP. this involves selecting suitable parameters for the window size and the weighting factor.

Experiments were performed on the well known problems: eil51, st70 and eil101, these are benchmark problems from the TSPLIB problem set [31]. Although these selected TSP problems are trivial (in that current evolutionary approaches are applied to TSP problem in excess of 7000 nodes) they allow for more quick evaluation of our approach. It is important to note that the purpose of this research is to demonstrate the use of the quality time tradeoff operator, as opposed to developing a competitive state of the art *optimal* search technique.

An example of the typical output from a genetic algorithm for the eil51 problem is given in figure 2. It is apparent from this graph that after a period of time the solution enters a plateau phase where little or no improvements are made to the fitness of the solution. With multi level genetic algorithms, similar plateau phases occur during the evaluation of the base level GAs. In figure 3, a graph is shown which shows the effect of applying the tradeoff operator. In this case, a weighting factor was selected that required solutions be found in relatively small number of generations. This weighting factor is totally arbitrary and represents some constraint (financial, time etc.).

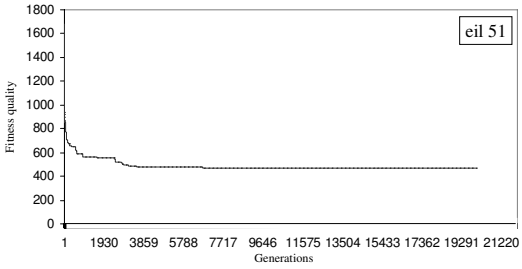


Fig. 2. Example GA search for eil-51 TSP.

The appropriate windowing size for a given problem must be selected so as to allow the GA sufficient time to encounter better solutions. As the search of a GA is not a linear improvement (figure 2) it is not easy to suggest a generalized windowing size and this is yet again a user selected variable. The approach taken in this case study was to produce an indicative graph of the genetic search by computing one run of a GA for the required problem. This provided the user with *some* insight into the dynamics of the problem (proximity to a known optimal solution after $n$ generations). This technique was found to be a very raw technique for selecting appropriate window sizes for small TSP problems.

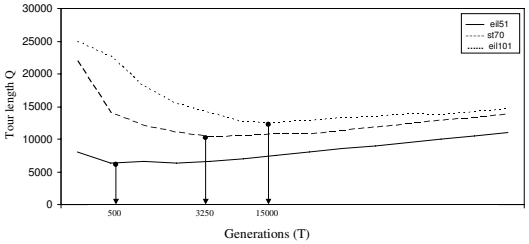


Fig. 3. U curves for Tour length $Q$ vs. generations $T$.

*2) The Problem: GA configuration setting for the TSP.*
The selection of appropriate operators and their associated parameters for the TSP is itself a complex and time consuming problem. With the use of our multi-level genetic algorithm it is possible to evaluate a large number of configurations. The amount of available computation power and time available are factors that affect the quality of solution that can be expected to be found. In our situation 84 Pentium III PCs were available for use, this allowed our multi level GA to operate with one PC designated as the meta-GA machine and 83 computers as base level GAs. An arbitrary weighting for the tradeoff operator was selected.

The Meta level-GA specified the *parameter strings* for each of the base level-GAs. Each of the elements in the parameter string encoded the value for a specific parameter in integer form. The string consisted of eight individual parameters as illustrated below:

| **Locus** | **1** | **2** | **3** | **4** |
|---|---|---|---|---|
| **Gene Function** | Population Size | Selection Type | Mutation Type | Mutation Rate |
| **Possible Alleles** | 0-9 | 0-9 | 0-9 | 00-99 |
| **Locus** | **5** | **6** | **7** | **8** |
| **Gene Function** | Crossover Type | Crossover Rate | Adaptive Mutation | Repair Type |
| **Possible Alleles** | 0-9 | 0-9 | 0-9 | 0-99 |

Fig. 4. Parameters for configuration string

- The *first gene* specified the size of the Base-GA population.
- The *second gene* specified the selection type.
- The *third gene* specified the mutation operator.
- The *fourth gene* specified the mutation rate.
- The *fifth gene* specified the type of crossover.
- The *sixth gene* specified the crossover rate.
- The *seventh gene* specified the use of any adaptive mutation.
- The *eighth gene* in the string specified the type of repair

**S1 = 3, 4, 2, 58, 1, 7, 8, 34**

Fig. 5. Example configuration string as passed to base GA

The Genetic operators that were considered were:
(1) crossover operators - Classical, n-point, Uniform, PMX, CX, OX, Voting Recombination, MPX, Masked, Modified, Position, Modified PMX, Complete Subtour Exchange, Subtour Chunking, Path random combination, Binary crossovers *random combination*, Adjacency crossovers *random combination*, *random combination* of all crossovers (see Larranaga et.al. [14]).

(2) Mutation operators: reciprocal exchange (swap) mutation, insertion mutation, 2-opt mutation, inversion mutation, *random combination* of mutation operators (see Larranaga et.al. [14]).

(3) repair- different repair configurations were also explored these repair operators used a number of techniques for the correction of invalid tours, when repair was not selected then a penalty function was used (see Mitchell et.al. [32-34] for details of GeneRepair techniques).

The parameter string therefore had six operator settings (excluding the mutation rate and crossover rates) that specified particular operators. The total number of permutations for these six settings is 45,000. The inclusion of the mutation and crossover rates increases the total number of permutations to approximately 45 million. This is a relatively small number of permutations in comparison to a reasonable sized TSP problem.
The following problem sizes were used for the experiments presented in this paper: 50, 70 and 100 cities. The 100 city size problem set was formed from the KRO series of TSPLIB benchmark problems [31], these were titled: 100a, 100b, 100c, 100d, 100e. The 70 city size problem set was constructed from varying sized data sets available from TSPLIB with the exception of one data set st70. These data sets therefore are not TSPLIB benchmark problems. The four data sets constructed that were titled 70a, 70b, 70c and 70d. The 50 city TSP problems data sets were all constructed from varying sized data sets available from TSPLIB these data sets were titled 50a, 50b, 50c, 50d and 50e.

These results were produced with a meta GA population size of 100 parameter strings and the meta GA was permitted to run for 25 days or approximately between 1000 and 1500 generations (depending on TSP problem size). Although this is a relatively low number of generations this permitted up to 1500000 different parameter strings to be considered.
Results suggest that a *set* of operators and associated parameters for individual TSP problems exist rather than one single configuration setting suiting *all* of the tested problems. The weighting for the tradeoff operator was varied widely so as to consider a wide range of values. Again it is important to note that these tests were not performed to produce optimal result but rather to meet the user's *quality time* tradeoff need. The GA Tradeoff values and the *W* weightings are presented in table 1.
The most consistent configuration settings that were found using the specified weightings and window sizes for the 50, 70 and 100 city problems are presented in table 2.
The result indicate that a large population size is most consistent with a population size the square of the number of cities being selected for both the 50 and 70 city problems and a population size of 4000 individuals being selected for the 100 city problem. Result found that were common to all problems were: tournament selection, insertion mutation, crossover rate of 1, a high initial mutation rate and crossover operators were a combination of all path representation crossover operators.

TABLE I
GA TRADEOFF RESULTS

| *Problem* | *Best Distance obtained* | *QTT Value* | *Weight w* |
|---|---|---|---|
| ***50a*** | 14162 | 1780 | 0.1 |
| ***50b*** | 1113 | 1122 | 1 |
| ***50c*** | 826 | 4452 | 5 |
| ***50d*** | 786 | 1983 | 2 |
| ***50e*** | 954 | 11274 | 11 |
| ***70a*** | 1090 | 17356 | 15 |
| ***70b*** | 130166 | 26954.8 | 0.2 |
| ***70c*** | 769 | 10556 | 13 |
| ***70d*** | 1266 | 9767 | 7 |
| ***70e*** | 1511 | 8373 | 5 |
| ***100a*** | 891 | 14337 | 15 |
| ***100b*** | 1947 | 30028 | 15 |
| ***100c*** | 38313 | 4814.2 | 0.1 |
| ***100d*** | 40348 | 12833.9 | 0.2 |
| ***100e*** | 37871 | 8428.6 | 0.2 |

TABLE II
GA WITH TRADEOFF PARAMETER SETTING

| *Setting* | *50 city* | *70 city* | *100 city* |
|---|---|---|---|
| ***Population size*** | Problem size$^2$ | Problem size$^2$ | 4000 |
| ***Selection type*** | Tournament | Tournament | Tournament |
| ***Mutation*** | Insertion | Insertion | Insertion |
| ***Mutation rate*** | 0.007 | 0.007 | 0.008 |
| ***Crossover*** | Random Combination - Path | Random combination – Path | Random Combination - Path |
| ***Crossover rate*** | 1 | 1 | 1 |
| ***Adaptive mutation*** | On | On | On |
| ***Repair type*** | Combination | Combination | Dynamic random repair |

Constraints for the problem could be satisfied by validity preserving crossover and mutation operators or when non validity preserving operators were employed one of 25 possible repair techniques could be employed or a penalty function to penalize invalid solutions could be selected. Results indicate that the multi level GA found that a combination of repair methods were most effective in tests.

## V. FUTURE WORK

We have identified another problem that requires the use of a multi level genetic algorithm and would benefit from the use of a cost benefit tradeoff operator to limit the amount of time spent determining base level solutions. Cell Signaling Networks (CSNs) are highly complex biochemical networks occurring in living cells. These complex systems of interacting molecules can be regarded as special purpose computers which take for input a variety of chemical signals emitted *from* or *outside* the cell (e.g.. hormones, proteins, ions etc). The reception of these input signals triggers a series of chemical reactions which can be viewed as a form of signal processing. The output of such systems can constitute a change in the state of the cell leading to specific cellular responses such as: the cell cycle, chemotaxis or apoptosis (programmed death of the cell). These cellular activities are crucial for the survival and adaptation of the organism.

One current goal of Synthetic Biology is to simulate and evolve Cell Signaling Networks in-silico. The ability to simulate and direct the evolution of CSNs may allow the design of *molecular computers* which may be programmed to perform specific tasks (e.g. smart drugs). However simulating CSNs in-silico is computationally expensive due to the high number of interacting molecules and the intricate and multi-level nature of these biological networks. This is why specific techniques from Evolutionary Computation (EC) are required to assist this enterprise. We believe that the tradeoff technique proposed in this paper may contribute to such projects.

In this evolutionary computation problem, we may observe a hierarchy of processes which constitutes the multi-level nature of the problem. At least 3 network levels can be distinguished:

- *Molecular level*: this is the lowest level in which the molecules are considered (nodes in the networks). These molecules may interact with one another, these chemical reactions constitute the networks arcs.
- *Cellular compartment level*: the molecules are located in compartments within the cell (membrane, cytosol, nucleus, etc). Compartments (nodes) may communicate (arcs) to each other by transferring molecules to one another.
- *Cellular level*: At this level, we consider the inter cellular communication, where cells may broadcast a signal to other cells.

For each network level, optimization of the network topology is required. Furthermore, additional features in the simulation need to be accounted for, such as: Brownian motion, chemical kinetics and other physical and chemical properties. For these various reasons, it is easy to understand why building such an evolutionary simulation platform requires adapted EC techniques to make it feasible.

Withstanding practical considerations, our proposed tradeoff operator is a mechanism that could assist in the simulation and evolution of CSNs by reducing and optimizing the effort needed during the evolutionary computational process,

assisting the search by examining those solutions that offer the most promising prospects.

Secondly, because we combine our tradeoff technique with a multi-level Genetic Algorithm, it is possible to use our approach to evolve the multi-level structure of CSNs.

Although the application of the tradeoff operator has previously only been evaluated on combinatorial optimization problems such as the TSP the application in synthetic biology would be a very worth while experiment and one which could assist in the better understanding of this complex problem area.

## VI. CONCLUSION

In this paper we presented our quality time tradeoff operator. This operator allows for more effective searching of problem with respect to the users requirements with regard quality of solution and amount of computation time. These needs are a daily challenge for many practical applications of multi level genetic algorithms.

Through the use of a user specified window size, solutions are provided with a grace period during which any improvement will be considered and can affect the running of the search. A second user specified value is the weighting. This value allows the user to place an importance on the quality of end solution verses the amount of computation time expended searching.

Following a presentation of the current literature and a review of our proposed technique we then presented a case study where the *tradeoff* operator was applied to a large complex problem: configuration setting for selected TSP problems.

Finally we outlined where we believe the tradeoff operator might be applied in the future.

## References


[1] G. Syswerda, "Schedule Optimization Using Genetic Algorithms," in *Handbook of Genetic Algorithms*, L. D. Davis, Ed. New York: Van Nostrand Reinhold, 1991, pp. 332-349.

[2] L. D. Davis, "Applying Adaptive Algorithms to Epistatic Domains," *Proc. of Int. Joint Conference on Artificial Intelligence IJCAI*, pp. 162-164, 1985.

[3] D. E. Goldberg and R. Lingle, "Alles, Loci and the TSP," *Proc. of the First International Conference on Genetic Algorithms and Their Applications*, pp. 154-159, 1985.

[4] I. M. Oliver, D. J. Smith, and J. R. Holland, "A study of permutation crossover operators on the traveling salesman problem," *Proc. 2nd International Conference on Genetic Algorithms and their Applications*, pp. 224-230, 1987.

[5] M. Gorges-Schleuter, "ASPARAGOS an asynchronous parallel genetic optimization strategy," *Proc. 3rd International Conference on Genetic Algorithms*, pp. 422-427, 1989.

[6] K. D. Crawford and R. Wainwright, "Research Question: How does one go about developing a new crossover operator with an a priori expectation of its merit? (A Survey of Crossover Operators for Genetic Algorithms)," The University of Tulsa, Tulsa, USA Technical Report UTULSA-MCS-96-2,, 1996.

[7] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin Germany: Springer Verlag, 1996.

[8] W. Banzhaf, "The "Molecular" Travelling Salesman," *Biological Cybernetics*, vol. 64, pp. 7-14, 1990.

[9] D. B. Fogel, "An Evolutionary Approach to the Traveling Salesman Problem," *Biological Cybernetics*, vol. 60, pp. 139-144, 1988.

[10] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*: The University of Michigan Press, 1975.

[11] J. Grefenstette, R. Gopal, B. Rosimaita, and D. V. Gucht, "Genetic Algorithms for the Traveling Salesman Problem," *Proc. 1st International Conference on Genetic Algorithms and their Applications*, pp. 160-165, 1985.

[12] D. B. Fogel, "Applying Evolutionary Programming to Selected Traveling Salesman Problems," *Cybernetics and Systems*, vol. 24, pp. 27-36, 1993.

[13] D. B. Fogel, "Empirical Estimation of the Computation Required to Reach Approximate Solutions to the Traveling Salesman Problem Using Evolutionary Programming," *Proc. of the Second Annual Conf. on Evolutionary Programming*, pp. 56-61, 1993.

[14] P. Larranaga, C. Kuijpers, R. Murga, I. Inza, and S. Dizdarevic, "Genetic Algorithms for the Traveling Salesman Problem: A Review of Representations and Operators," *Artificial Intelligence Review*, vol. 13, pp. 129-170, 1999.

[15] G. Harik and F. Lobo, "A Parameter-Less Genetic Algorithm," presented at Genetic and Evolutionary Computation Conference, 1999. GECCO 1999, 1999.

[16] M. Mernik, M. Crepinsek, and V. Zumer, "A Meta Evolutionary Approach in Searching of the Best Combination of Crossover Cperators for the TSP.," *Proc. of the International Conference on Neural Networks (NN'2000)*, pp. 32-36, 2000.

[17] B. Freisleben, "Metaevolutionary approaches," in *Handbook of Evolutionary Comutation*, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds. Oxford, UK: IOP Press, 1997, pp. C7.2:1.

[18] A. Kosorukoff, J. Mittenthal, and D. E. Goldberg, "Modeling of evolution of signaling networks in living cells by evolutionary computation," University of Illinois at Urbana Champaign, Urbana IL, USA, IlliGAL Report No. 2001006, 2001.

[19] T. C. Belding, "The Distributed Genetic Algorithm

Revisited.," *Proc. 6th International Conference on Genetic Algorithms*, pp. 114-121, 1995.
[20] H. Mühlenbein, *Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization*: Morgan Kaufmann Publishers, 1989.
[21] G. Cesari, "Divide and conquer strategies for parallel TSP heuristics," *Computers & Operations Research*, vol. 23, pp. 681-694, 1996.
[22] S. A. Mulder and I. Wunsch, Donald C., "Million city traveling salesman problem solution by divide and conquer clustering with adaptive resonance neural networks," *Neural Networks*, vol. 16, pp. 827-832, 2003.
[23] J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 16 (1), pp. 122-128, 1986.
[24] R. Sosic and G. D. Wilby, "Using the Quality-Time Tradeoff in Local optimization.," *Proceedings of the IEEE Second ANZIIS Conference*, pp. 253- 257, 1994.
[25] Z. Michalewicz and D. B. Fogel, *How to SolveIt: Modern Heuristics*. Berlin, Heidelberg, New York: Springer Verlag, 2000.
[26] G. Dantzig, R. Fulkerson, and R. Johnson, "Solution of a large-scale travelling salesman problem," *Operations Research*, vol. 2, pp. 393-410, 1954.
[27] S. Lin and B. W. Kernighan, "An Effective Heuristic Algorithm for the Traveling Salesman Problem,," *Operations Research*, vol. 21, pp. 498-516, 1973.
[28] J. N. MacGregor and T. C. Ormerod, "Human performance on the traveling salesman problem," *Perception and Psychophysics,*, vol. 58, pp. 527-539, 1996.
[29] K. Katayama and H. Narihisa, "An Efficient Hybrid Genetic Algorithm for the Traveling Salesman Problem," *Electronics and Communications in Japan, Part 3*, vol. 84, pp. 1783-1791, 2001.
[30] K.-S. Leung, H.-D. Jin, and Z.-B. Xu, "An expanding self-organizing neural network for the traveling salesman problem," *Neurocomputing*, vol. 62, pp. 267-292, 2004.
[31] G. Reinelt, "TSPLIB - A Traveling Salesman Problem Library," *ORSA Journal on Computing*, vol. 3, pp. 376-384, 1991.
[32] G. G. Mitchell, D. O'Donoghue, and A. Trenaman, "A New Operator for Efficient Evolutionary Solutions to the Travelling Salesman Problem," *Proc. Applied Informatics*, pp. 771--774, 2000.
[33] G. G. Mitchell, D. O'Donoghue, D. Barnes, and M. McCarville, "GeneRepair- A Repair Operator for Genetic Algorithms," *Proc. Genetic and Evolutionary Computation Conference (GECCO) Late Breaking Papers*, pp. 235--239, 2003.
[34] G. G. Mitchell, "Validity Constraints and the TSP - GeneRepair of Genetic Algorithms.," *Artificial Intelligence and Applications*, pp. 306-311, 2005.