Hindawi Publishing Corporation
Journal of Mathematics
Volume 2016, Article ID 4015845, 10 pages
http://dx.doi.org/10.1155/2016/4015845

Research Article

Improving Genetic Algorithm with Fine-Tuned Crossover and Scaled Architecture

Ajay Shrestha and Ausif Mahmood

Department of Computer Science and Engineering, University of Bridgeport, 126 Park Avenue, Bridgeport, CT 06604, USA

Correspondence should be addressed to Ajay Shrestha; shrestha@my.bridgeport.edu

Received 26 November 2015; Revised 13 March 2016; Accepted 21 March 2016

Academic Editor: Niansheng Tang

Genetic Algorithm (GA) is a metaheuristic used in solving combinatorial optimization problems. Inspired by evolutionary biology, GA uses selection, crossover, and mutation operators to efficiently traverse the solution search space. This paper proposes nature inspired fine-tuning to the crossover operator using the untapped idea of Mitochondrial DNA (mtDNA). mtDNA is a small subset of the overall DNA. It differentiates itself by inheriting entirely from the female, while the rest of the DNA is inherited equally from both parents. This unique characteristic of mtDNA can be an effective mechanism to identify members with similar genes and restrict crossover between them. It can reduce the rate of dilution of diversity and result in delayed convergence. In addition, we scale the well-known Island Model, where instances of GA are run independently and population members exchanged periodically, to a Continental Model. In this model, multiple web services are executed with each web service running an island model. We applied the concept of mtDNA in solving Traveling Salesman Problem and to train Neural Network for function approximation. Our implementation tests show that leveraging these new concepts of mtDNA and Continental Model results in relative improvement of the optimization quality of GA.

1. Introduction

Genetic Algorithm is a nature inspired metaheuristic used to solve optimization and search problems which would otherwise take a long time to solve using brute force methods. GA provides us the means to traverse the solution search space intelligently and to come up with a near optimal solution in a substantially short amount of time. Genetic Algorithms are used beyond computer science, engineering, and mathematics, in areas such as economics, bioinformatics, life sciences, and manufacturing. GA is well suited for combinatorial optimization problems. One such problem where we can deploy GA is the Traveling Salesman Problem (TSP).

The goal of Genetic Algorithm is to come as close as possible to the optimal solution. Since the solution search space is so huge, the major difficulty in reaching this goal is the convergence into local minima before exploring the entire search space for global minima. This is where we could exploit the concept of mtDNA to help add some order in the random search for near optimal solution.

2. Genetic Algorithm

The idea of GA was proposed by Holland in his 1975 book [1]. Since then GA has been an active field of research and there has been numerous publications on it. TSP is one of the problems where GA has been successfully used.

As shown in Figure 1, GA has two primary functions: population selection and crossover. Selection algorithm describes the methodology to pick parents that will create children for the next generation. There are four strategies shown in the diagram: elite, roulette, rank, and tour. The elite strategy gives preference to selecting the best members from the current population itself [2]. In roulette selection, members are mapped to a roulette wheel occupying space that is proportional to their fitness and members are selected randomly from it avoiding duplicates [3]. Rank selection method is similar to roulette, but instead of proportional representation of the pie based on fitness, members are ranked in ascending order based on their fitness [2]. In tournament selection, n population members are chosen to compete and the best one

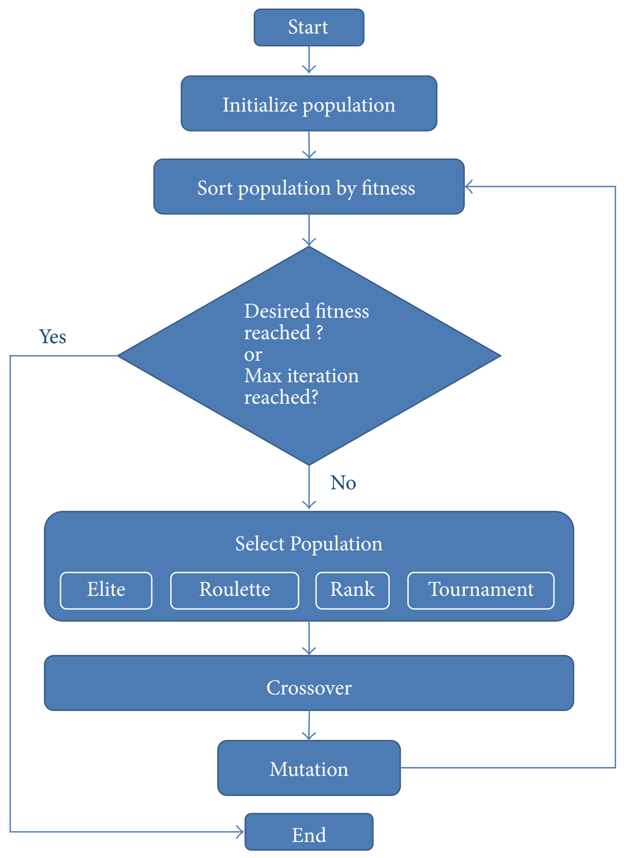

Figure 1: Genetic Algorithm.

is selected to be a parent from the pool of n members [2]. This process is continued until all members have been examined.

The crossover is the process of intermixing the genetic representation of the parent population members with the intension of creating a better fitness in the resulting offspring. There are different types of crossover operators based on the tour representation. Partially mapped crossover (PMX), proposed by Goldberg and Lingle [4], is a popular operator. Here a section of one parent's genes is mapped to the other parent's and the rest are interchanged to produce the offspring [5].

Mutation adds addition value to GA by introducing random change which could assist in overcoming local minima in the search exploration. One way to implement mutation is to randomly select a small percentage of population members and interchange a unit of genes with an adjacent one.

The solution can be enhanced by utilizing Island Model. Here the population is broken into smaller groups or islands and GA is run separately on each in separate threads allowing us to exploit multiple processors or even multiple distributed servers to solve a large problem. This not only speeds up the processing time but also improves the quality of the solution in most cases because it eliminates the sampling bias, if any present in the initial population [3] when run in a single thread. In addition, a small number of population members from different islands can be exchanged to exploit diversity and prevent premature convergence.

Another method to improve the quality of the solution is to perform a more thorough local search. We can achieve this by using K-Opt algorithm, where K is a numerical value, which is usually 2 or 3. In 2-Opt method, two edges are removed from the tour and reconnected in the other way that is possible to retain a valid tour [7]. The advantage of 2-Opt is that it is fast and efficient. When used in combination with standard GA operators, 2-Opt's probability of getting stuck in local minima is also mitigated. Every execution of 2-Opt requires $n \times m$ operations, where n is the number of cities and m is the number of members. Therefore, 2-Opt should not be run on every iteration but rather every X (e.g., 100) iterations to limit the execution time.

2.1. Using GA to Solve TSP.

In a symmetric TSP problem, a salesman has to visit a number of cities and return back to the original (first) city with the shortest route [5]. TSP is a classic NP-hard problem and the worst case run time for solving it exhaustively increases superpolynomially or exponentially with the increase of number of cities.

When using GA to solve TSP, every city is denoted by a unique number. Every solution is a random sequence (or a population member in case of GA) of unique nonrepeating numbers, representing a possible route or tour. Every population member with a unique genetic makeup represents a solution to GA. Likewise every route represents a solution in TSP. GA starts out with an initial set of randomly generated population members that go through several iterations of selection and crossover function in the hope of improving the solution in the subsequent iteration or generation. For TSP, we generate random sets of routes, each of which consists of vector sequence of cities. The selection method picks pairs of routes or population members which are allowed to become inputs to the crossover function. The crossover function then exchanges the unique numbers (cities) of each route pair to generate two children (new population members) for each pair of parent population. The children, who themselves represent solutions (routes) to the TSP, then replace the parents as the new set of routes (population members). This iterative process of selection and crossover can continue until we do not get any better results in the next generation.

Additional enhancements are provided by mutations and island models. To mimic mutations in GA, a small percentage of route solutions are randomly picked to have their sequence interchanged by switching two adjacent cities. Mutation contributes to retaining diversity [9]. In Island Model, multiple GAs are independently run and a small number of population members (routes in case of TSP) are exchanged between these islands after certain number of iterations (generations). Island model also allows us to exploit all available (abundant) computing resources by running multiple GAs in parallel [3]. Both these mechanisms have proven to positively impact the outcome of the solution.

2.2. Train Artificial Neural Network Using GA.

Artificial Neural Networks (ANNs) are learning systems that are inspired by biological nervous system. ANNs mimic the anatomy and function of the brain. ANNs consist of interconnected processing units/nodes (similar to neurons) that are organized

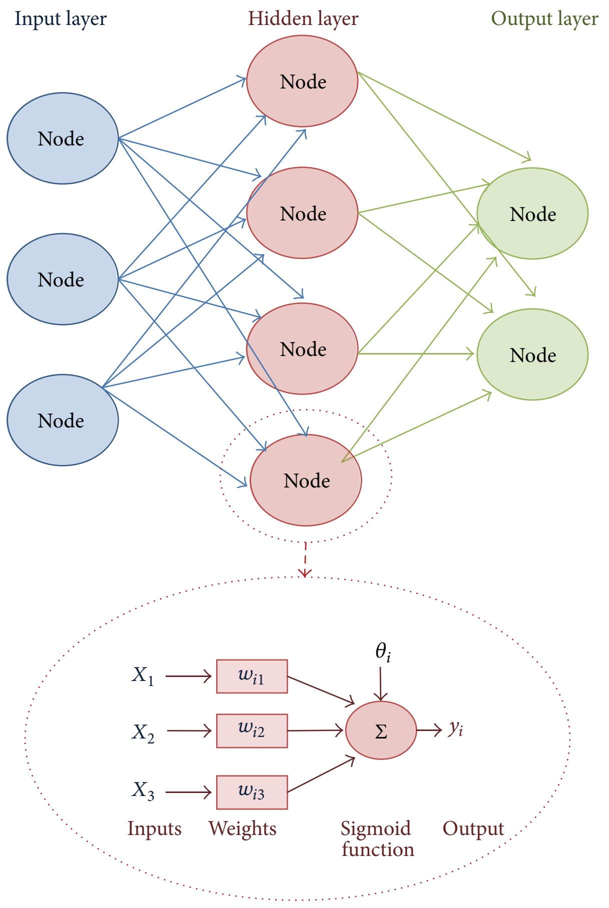

Figure 2: Artificial Neural Network.

in layers as shown in Figure 2. Each node's incoming connections have weights assigned to them and the summation of the incoming signal's weights is processed through the node and the result feed to the subsequent node[s] in the next layer.

In Figure 2, each node in the input layer is connected to all the nodes in the hidden layer and subsequently all nodes in the hidden layer are connected to the output layer nodes. Each incoming connection of the node i is represented by x_j (i.e., jth input to the node) and each connection also has a weight, w_{ij}, associated with it [10].

The following equation shows the mathematical representation of y_i, the output of the processing of the node, where θ_i is the bias and f_i is a nonlinear function such as sigmoid function [10].

ANN Node Function. Consider

$$y_i = f_i\left(\sum_{j=1}^{n} w_{ij}x_j - \theta_i\right) \quad (1)$$

(see [10]).

The ANN in Figure 2 is a Feedforward Neural Network, where the connections between the nodes do not form a feedback loop. There are different ways to train a Neural Network. During the training phase the value of the weights and biases are optimized to solve a particular problem. Gradient descent is typically used to adjust the weights based on the difference between the desired output and the current one [11]. Genetic Algorithm can also be used to train the Neural Network. The selection and fitness criteria can be aptly applied in training the Neural Network. Function approximation is one of the areas where Neural Networks can be used effectively.

3. Related Work

There has been considerable amount of research to improve the GA operators to solve TSP. The development of several selection strategies mentioned earlier, that is, elite, roulette, rank, and tournament, is a testimony of that effort. These strategies have been implemented and run against TSPLIB benchmarks [8] by different researchers. Each of these selection operators has its own characteristics, benefits, and shortcomings. Razali and Geraghty [6] concluded in the paper that rank based selection strategy yielded better results but took more computation time, while tour method is faster for small sized problems [6]. Selection methods represent only one side of the TSP problem. The other major side is the crossover functions, which contributes significantly to the success of the algorithm. There are about eleven crossover operators reviewed by Larrañaga et al. [5] in their paper. Majority of them are based on specific patterns of information mixing and interchange between the parents, for example, Order Crossover (OXI), introduces several uniform length cut points in the path of the parents and produces offspring with several subpaths from the parents intact and assimilated in the children [12]. Another crossover operator, that is, Genetic Edge Recombination crossover, adds more meaningful logic in its workings by assuming the edges of the tour are important and attempts to preserve them in the offspring [13].

There are published literatures on restrictive crossover. Galán et al. [14] proposed a mating strategy that balances between exploration (selection criteria) and exploitation (fitness criteria) by developing a parameter called *mating index*, which controls the degree of exploration (or diversity) of parents based on the hardness of the problem. Strategies like *incest prevention* [15] prevent mating between similar individuals. Assortative mating is another strategy used to improve GA results. Ochoa et al. [16] demonstrate the relation between mutation rates and assortative mating choices; that is, higher mutation rates work well with assortative mating whereas lower mutation rates work well with dissortative mating to confer better fitness. The idea behind these strategies is based on the principle that offspring of similar individuals do not result in higher fitness. Introducing controlled mating based on similarity of genes does yield better results but they are also computationally costly as the lengthy chromosomes have to be compared.

This paper presents a further optimized idea of restrictive mating to complement the standard crossover operators. The idea is based on the premise that it would not be beneficial to select the offspring of the same parents (or close lineage) as new parents to cross over with each other. In fact, it could be detrimental to maintaining diversity and exploring greater search space. As an alternate to exhaustive comparison of the genes to determine genetic diversity between the parents, we present an algorithm that is computationally lean. We exploit the concept of mtDNA to enhance the GA.

4. Proposed Idea and Its Implementation

4.1. Mitochondrial DNA (mtDNA). Humans have 23 pairs of chromosomes with one copy of each pair inherited separately from each parent [6]. The DNA in these chromosomes is referred to as nuclear DNA [17]. In addition, humans also have mtDNA [17], which consists of only 1% of the total DNA [18], thus coding for far less genes. Though insignificant by orders of magnitude when compared to the nuclear DNA in their contribution to inheritable traits (genes), mtDNA's unique characteristic in inheritance can play an important role in guiding the search for optimal solution. The DNA sequence in the 23 pairs of chromosomes is inherited equally into the offspring from both the parents during reproduction, whereas the sequence in mtDNA is inherited only from the maternal side [18]. This allows us to keep track of population members with similar genetic traits and common inheritance via maternal lineage. Diversity is the key to preventing premature convergence and achieving near optimal solution. Crossovers between similar population members with close DNA proximity will not yield results better than the prior generation in most cases. The idea in this paper is to create a data structure to tag and track the mtDNA in every population member and restrict the crossover between population members with similar mtDNA. mtDNA is widely used in evolutionary genetics and population study [19], and its concept could potentially be beneficial to GA search exploration.

4.2. Using mtDNA GA and Scaled Architecture to Solve TSP. The primary objectives of GA are to help get us better solution after every iteration and to prevent solution exploration from prematurely converging into local minima. The primary way to address the later goal is to introduce the right amount of diversity in the parents.

Most of the crossover operators tend to be very refined and granular at the city or node information level and seem to overlook the bigger picture. As the GA undergoes several iterations of crossover, the risk of convergence increases too and in some cases, crossovers between the same population members' offsprings would not yield results any better than the previous generation because their parents would have similar genetic information to begin with. With less genetic variance in the parents, we cannot expect better or different results in the offspring. It is self-evident that genetic variability sows the seed for evolution and newer offspring [20]. One way to track genetic similarity is by tracking the family lineage. And the most effective way to track inheritance in the real world is through mtDNA [21].

The concept of mtDNA (Mitochondrial DNA) is implemented in this paper to control the crossover function to prevent population members with same mtDNA from reproducing for n number of generations. To avoid the overreaching consequences of this condition, this requirement is dictated only on a percentage of crossovers. mtDNA is defined as a separate attribute of the population member class. Since mtDNA gets inherited solely from the female parent, it does not alter as it is passed down to the offspring. This attribute was exploited to guide and control crossovers. Here is a high level overview of the mtDNA algorithm and pseudocode.

All of the four selection methods (tour, elite, roulette, and rank) described earlier were utilized during the implementation. To transfer genes to children during crossover, 1/4 to 3/4 tour cut was made on parent one and transmitted to the children. The rest was transferred in cyclic order from the second parent, skipping any cities that were already derived from the first parent, thus ensuring every city is represented in the child with no repetition. In addition to leveraging mtDNA in the implementation, various selection methods, Island Model, 2-Opt, and distributed processing using multiple servers (Continental Model), were also utilized.

Figure 3 provides a high-level workflow of the GA implementation in this paper. Custom version of GA was run on each of the four threads on each server.

Island Model was implemented with multicore processors in server by running multiple threads in parallel. Each thread ran its own version of GA. Periodically after every X number of iterations/generations on each of the threads running the GA, a handful of randomly selected population members were exchanged between the threads. This process not only added more computing resources and improved the execution time of GA but also increased diversity and reduced initial sampling bias.

2-Opt was implemented by selecting the population member with the best fitness so far in the particular thread of GA execution every X iterations/generations. The selected member then underwent local optimization. Two links/edges of the best member were swapped exhaustively to check if it improves the solution.

Island Model was further scaled with distributed processing by executing the abovementioned implementation on several servers using Web Services (Service Oriented Architecture (SOA)). We aptly named it *Continental Model*. Population members were randomly exchanged between these independently run Island Model GA implementations in different servers after a fixed number of iterations to achieve diversity and to reduce the likelihood of premature convergence.

The implementation was run against known TSP instances (dantzig42, eil51, rd100, ch150, and kroB200) from TSPLIB [8]. The numerical value in the name of the benchmark denotes the number of cities in it; for example, eil51 has 51 cities. The results from the mtDNA implementation of GA were compared against the results of implementation by Razali and Geraghty [6] and known best solutions.

4.3. mtDNA GA in Artificial Neural Network. We used GA to train the Neural Network for function approximation. A multilayer feedforward ANN, with 1 node in the input layer, 26 nodes in the first hidden layer, 26 nodes in the second hidden layer, and 1 node in the output layer, was chosen. The GA implementation for Neural Network is similar to GA implementation for TSP. In place of the city number (in TSP), the value of the weights is randomly initialized in a solution set and crossed over with another set of weights in the case of

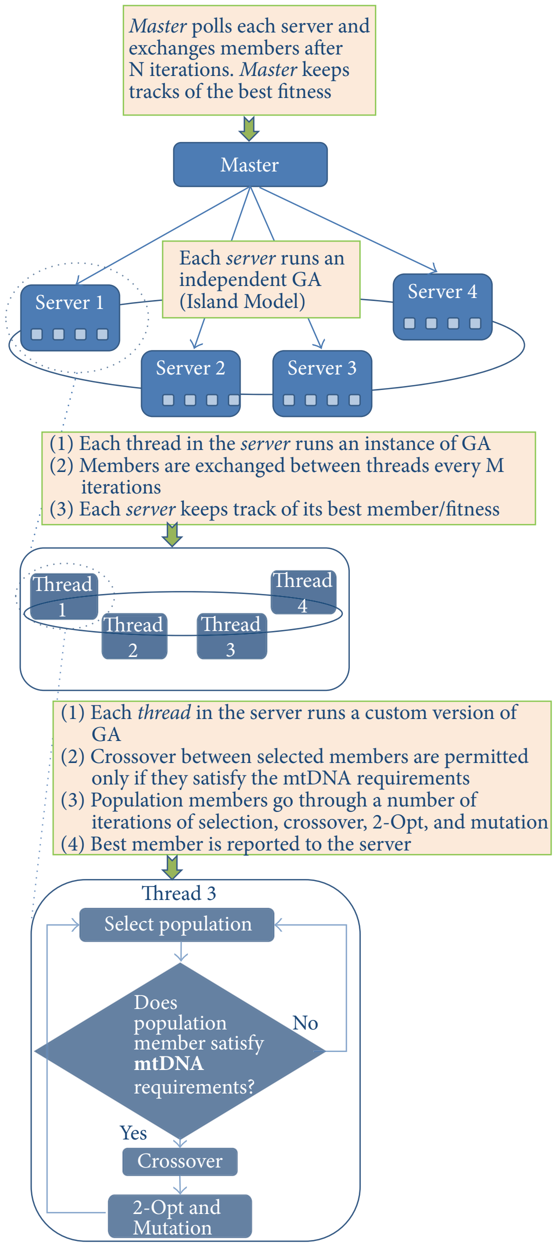
Master polls each server and exchanges members after N iterations. *Master* keeps tracks of the best fitness
Master
Each *server* runs an independent GA (Island Model)
Server 1
Server 2
Server 3
Server 4
(1) Each thread in the *server* runs an instance of GA
(2) Members are exchanged between threads every M iterations
(3) Each *server* keeps track of its best member/fitness
Thread 1
Thread 2
Thread 3
Thread 4
(1) Each *thread* in the server runs a custom version of GA
(2) Crossover between selected members are permitted only if they satisfy the mtDNA requirements
(3) Population members go through a number of iterations of selection, crossover, 2-Opt, and mutation
(4) Best member is reported to the server
Thread 3
Select population
Does population member satisfy **mtDNA** requirements?
No
Yes
Crossover
2-Opt and Mutation

Figure 3: Continental Model GA with mtDNA.

Neural Network. But unlike in TSP, the values of the weights do not need to be unique within a solution set.

mtDNA was introduced in GA here just like in TSP. The resulting children from crossovers were tagged with the same mtDNA attribute of the mother for X iterations as defined in Algorithm 1 and crossovers prevented between those with same mtDNA. mtDNA value was reset after X iterations to ensure that crossovers were not too restrictive. We used mtDNA implementation of GA to train the Neural Network for function approximation.

Here are the details from the implementation:

(1) Population size = 200.

(2) Selection ratio = 20.

Table 1: mtDNA GA solution (fitness) versus others.

TSPLIB benchmarks	Known best solution [6]	Razali and Geraghty [6]	mtDNA + 2-Opt + multinode
dantzig42	679	679	669
eil51	425	430	413
eil76	538	N/A	536
rd100	7910	N/A	7990
ch150	6528	N/A	6739
kroB200	29437	N/A	30706

(3) Mutation ratio = 4.

(4) mtDNA reset every X iterations, where $X = 15$.

(5) Nodes are as follows: 1 (input layer), 26 (hidden layer 1), 26 (hidden layer 2), and 1 (output layer).

GA was used to train ANN for the following functions with and without the mtDNA logic (Figure 4):

Function A: $f(x) = x\sin(50x)/e^2$.

Function B: $f(x) = 250\sin(2x)\sin(x)$.

Function C: 1D version of Schewefel function $f(x, y) = -x\sin(\sqrt{|x|}) - y\sin(\sqrt{|y|})$.

Function D: 1D version of $f(x,y) = (16x(1-x)y(1-y)\sin(9\pi x)\sin(9\pi y))^2$.

5. Results

5.1. GA in Traveling Salesman Problem. The TSPLIB column in Table 1 indicates the benchmark names. The 2nd and the 3rd columns represent the fitness from the known best solution and Razali et al. [6] paper, respectively. The last column lists the fitness value from the GA implementation with mtDNA together with 2-Opt and continental model in this paper. In case of TSP, the fitness function is the distance traveled by the salesman through all the cities and back to the first one. Solutions are evaluated based on the fitness value; that is, the lower the value the better.

Figure 5 shows the comparison between this paper's mtDNA GA implementation and the known solutions [8] of the TSP benchmarks. mtDNA GA implementation scored better on benchmarks with relatively less number of cities, that is, dantzig42 (42 cities), eil51 (51 cities), and eil76 (76 cities).

Figures 6, 7, and 8 provide the results of mtDNA GA implementation on TSPLIB benchmarks with higher (100, 150, and 200) number of cities. While the results of mtDNA GA on these higher benchmarks are slightly behind than the known best solution, they are significantly better than the results when mtDNA logic and scaled (multinode) architecture were not used. Thus, introducing these two concepts adds value to solving TSP by consistently lowering fitness of the solution, even in TSPLIB benchmarks with greater than 99 cities.

Table 2 provides the best route/tour results that were received by running GA with mtDNA for the respective

```
(1) Initialization
      Assign mtDNA attribute to each population member
         If population = initial
            then mtDNA ← random unique value
         If population = offspring of crossover
            then mtDNA ← mtDNA of female (2nd parent)
(2) for $i$ ← 1 to Maximum Iterations
    (a) Selection
        Check mtDNA attributes of crossover pairs at (total iteration mod 100) $< N$ iterations, where $N < 100$
         If parent 1 mtDNA = parent 2 mtDNA
            then abort crossover & find another pair
         If parent 1 mtDNA ≠ parent 2 mtDNA
            then allow crossover
         Reset mtDNA attribute of all members to unique values after $X$ iterations.
               $X < \log_2 P$, where $P$ = total population
    (b) Crossover & mtDNA transfer
        Children's mtDNA ← mtDNA of the female parent
```

Algorithm 1: mtDNA pseudocode and algorithm.

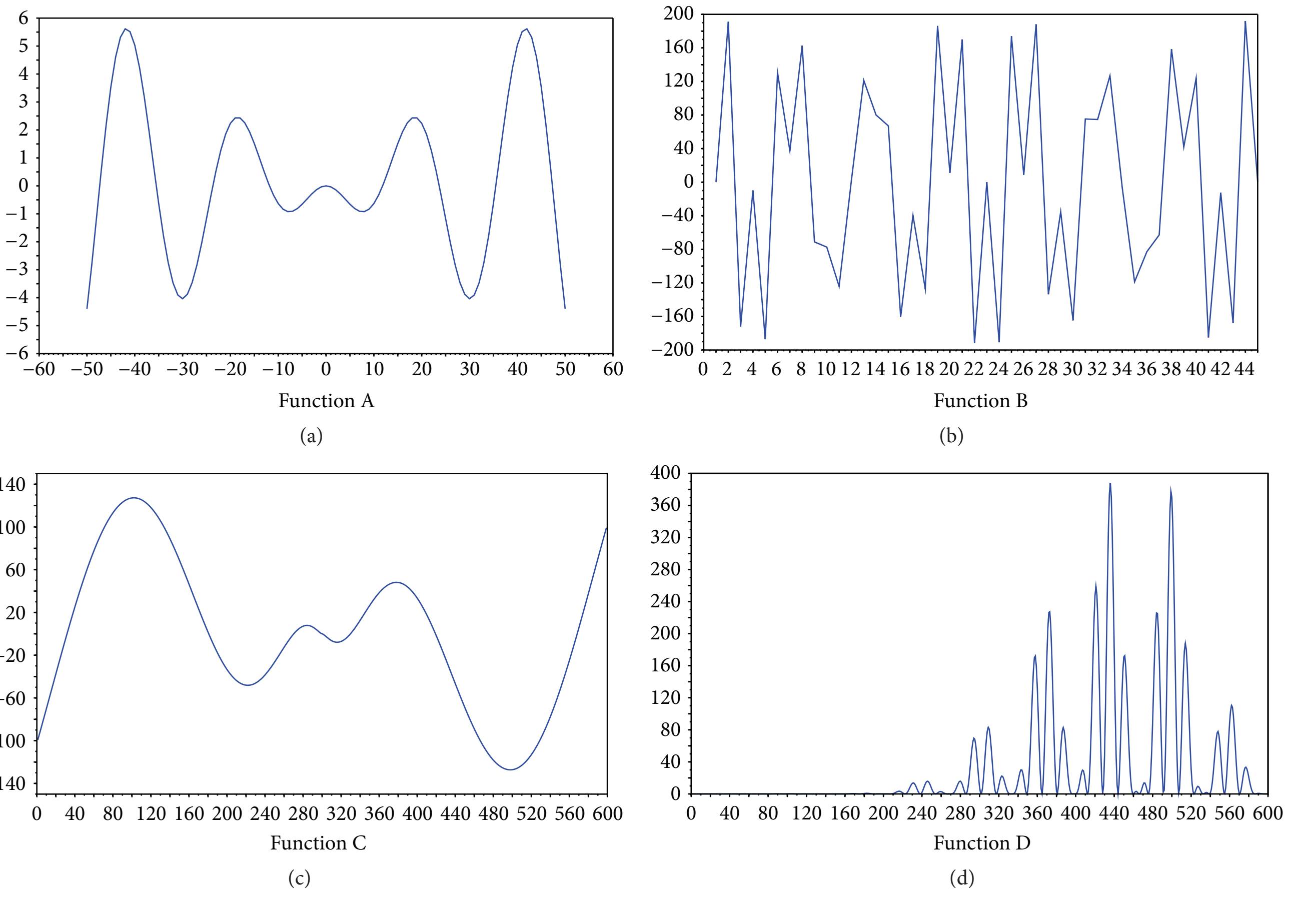

Figure 4: Four functions used for training Neural Network.

benchmarks. The result from Table 1 demonstrates that mtDNA yields results that are better than Razali and Geraghty [6] and the published solution posted on TSPLIB [8] for dantzig42 and eil51 benchmarks. When mtDNA and Continental Model were used with other known operators and algorithms, it resulted in solution that was better than the published solution for eil76 and very close to known best solutions for rd100, ch150, and kroB200 benchmarks.

Table 2: Best route/tour for mtDNA fitness values for various benchmarks.

TSPLIB	mtDNA fitness	mtDNA route/tour with city sequence
dantzig42	669	0>1>3>2>4>5>6>7>8>9>24>25>26>23>10>11>22>21>16>15>12>13>14>17>18>19>20>27>28>29>30>31>32>33>34>35>36>37>38>39>40>41>0
eil51	413	0>31>10>37>4>36>16>3>17>46>11>45>50>26>5>47>22>6>42>23>13>24>12>40>39>18>41>43>14>44>32>38>9>48>8>29>33>20>49>15>1>28>19>34>35>2>27>30>25>7>21>0
eil76	536	0>72>32>62>15>2>43>31>8>38>71>57>11>39>16>50>5>67>3>74>75>66>25>6>52>13>58>65>10>64>37>9>30>54>24>49>17>23>48>22>55>40>42>41>63>21>27>60>20>46>47>28>4>36>35>68>70>59>69>19>14>56>12>53>18>34>7>45>33>51>26>44>29>73>1>61>0
rd100	7990	0>59>68>7>70>67>82>50>89>90>63>35>56>46>98>52>78>5>93>97>65>15>44>51>10>60>4>80>79>64>47>29>21>40>33>6>41>23>24>42>39>54>95>38>99>43>28>34>22>1>88>83>30>87>57>75>58>94>76>92>27>36>18>53>37>69>71>16>72>49>45>55>91>26>9>32>2>77>19>73>25>8>31>3>11>13>84>81>74>20>48>12>66>96>85>62>14>86>61>17>0
ch150	6739	0>97>102>81>94>106>4>99>142>96>123>34>92>125>32>51>110>104>91>53>133>137>45>89>19>24>140>82>55>145>25>74>17>141>84>64>131>136>101>113>98>107>69>134>49>54>57>80>109>28>85>18>1>36>5>27>8>41>119>46>138>39>52>11>23>117>126>68>35>60>10>147>129>16>65>59>139>116>56>38>40>100>115>42>50>108>66>37>22>31>130>76>121>13>79>132>15>120>93>87>78>58>14>77>20>149>114>70>43>63>111>135>144>71>48>146>143>128>26>30>122>73>12>105>90>118>67>127>44>3>103>21>124>148>61>2>112>9>95>88>7>6>83>29>62>47>72>75>33>86>0
kroB200	30706	0>94>173>11>106>97>31>166>174>58>135>28>7>98>75>96>90>128>27>157>2>10>92>121>84>72>141>122>9>67>110>188>52>169>148>69>197>38>39>189>116>193>196>66>4>131>61>25>99>55>100>149>68>74>29>79>19>180>133>48>85>137>165>107>151>37>145>45>24>8>33>56>6>83>57>168>71>36>64>123>126>187>78>112>80>178>46>140>183>114>172>138>190>153>53>162>115>51>185>195>42>184>117>158>49>163>176>142>170>87>22>21>192>103>54>143>159>152>23>76>102>139>105>182>15>104>17>161>134>18>150>136>95>118>91>101>199>129>43>147>40>191>16>35>44>77>154>12>186>125>62>30>47>194>155>50>127>111>14>32>146>120>81>109>108>1>41>119>124>13>63>5>3>82>181>88>59>73>164>86>113>167>132>65>144>198>177>130>156>171>34>60>93>179>175>26>70>160>89>20>0

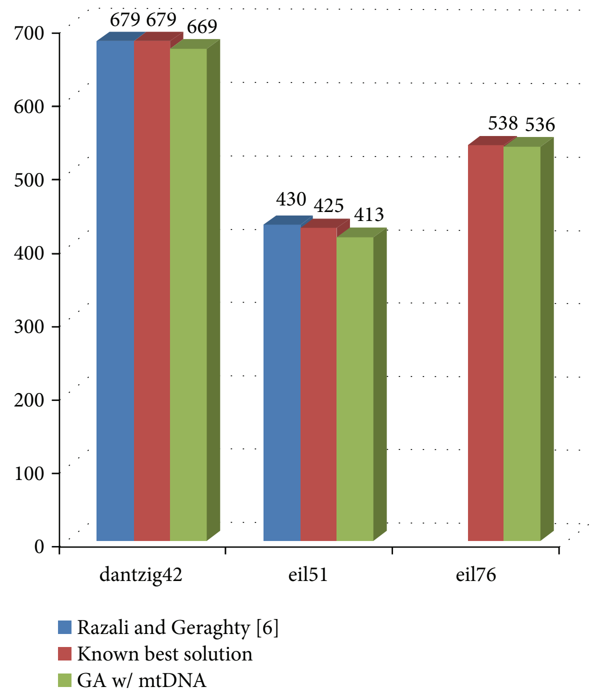

Figure 5: Graph shows GA tour/fitness results of this paper (mtDNA) (Razali and Geraghty) paper [6] and the known best solution [8].

Table 3: Best route/tour for mtDNA fitness values for various benchmarks.

Function	Training algorithm	RMS error per iteration					
		100	200	300	500	1000	5000
A	Standard GA	1.84	1.78	1.78	1.77	1.75	0.232
	mtDNA GA	1.79	1.77	1.77	1.76	0.92	0.203
B	Standard GA	2.94	2.92	2.90	2.87	2.82	2.666
	mtDNA GA	2.93	2.91	2.90	2.87	2.80	2.610
C	Standard GA	0.48	0.46	0.42	0.37	0.29	0.282
	mtDNA GA	0.47	0.41	0.35	0.30	0.21	0.121
D	Standard GA	0.64	0.59	0.59	0.59	0.58	0.371
	mtDNA GA	0.61	0.58	0.58	0.58	0.53	0.314

5.2. GA in Artificial Neural Networks. The ANN was trained separately using mtDNA implementation of GA and GA by itself. After the weights were set, the ANN was used to approximate the four functions (Figure 4) and the square error was computed. The results from the mtDNA implementation of GA as listed in Table 3 were better than the results when GA was used by itself across all four functions. Table 3 shows the results from the several (100, 200, 300, 500, 1000, and 5000) iterations/executions using both the training methods.

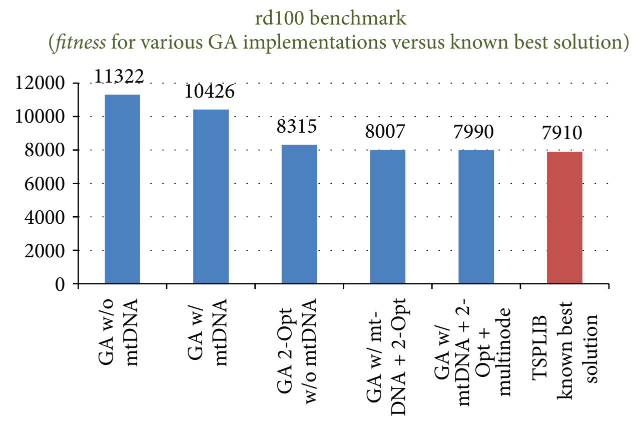

Figure 6: Graph shows various GA/mtDNA results along with the published best solution for rd100.

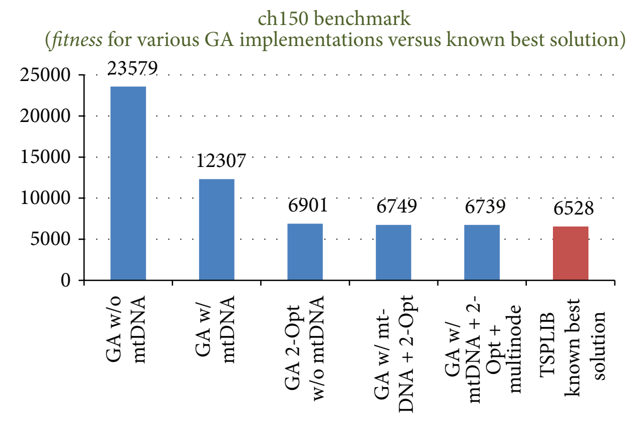

Figure 7: Graph shows various GA/mtDNA results along with the published best (known) solution for ch150.

Figures 9–12 represent the data from Table 3 in graphical form.

The results from mtDNA incorporated GA trained ANN consistently outperforms the GA-only trained ANN for the given four function approximations.

6. Conclusion

We have presented two important ideas of mtDNA and a *Continent Model* in improving the optimization quality of GA. The mtDNA logic introduced in the paper is novel idea and is inspired by nature just like many of the optimization algorithms, for example, Genetic Algorithm, Swarm Intelligence, Ant Colony Optimization, and Neural Network. Like these nature inspired algorithms and systems, the concept of mtDNA is not very complex but can be instrumental in improving the quality of metaheuristics. Maintaining diversity is the key to preventing premature convergence into local minima. The characteristics of mtDNA can be exploited to track diversity and restrict crossover between parents of same genetic traits, thus yielding better fitness value in the offspring. The mtDNA concept articulated and implemented

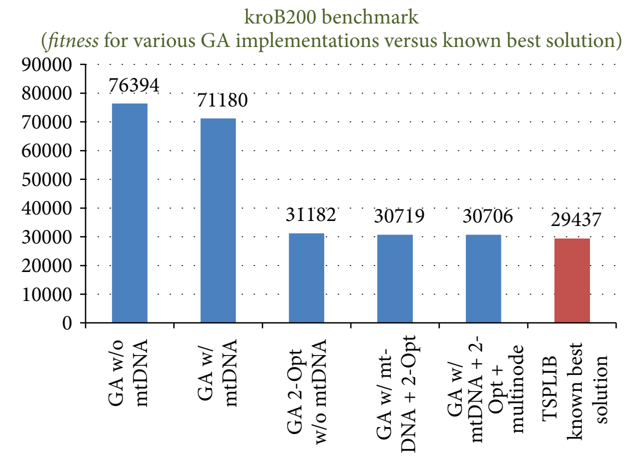

Figure 8: Graph shows various GA/mtDNA results along with the published best (known) solution for kroB200.

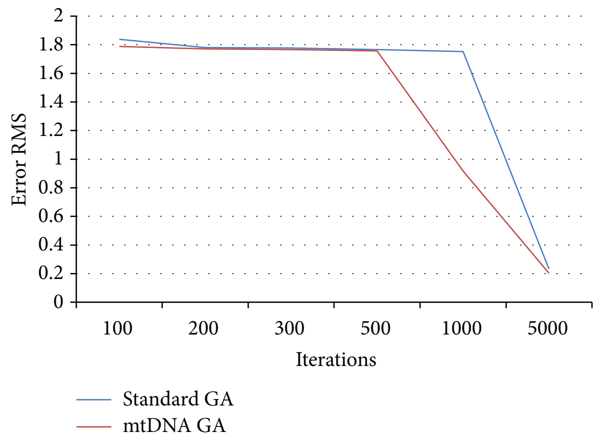

Figure 9: Graph shows error with GA mtDNA and standard GA for Function A.

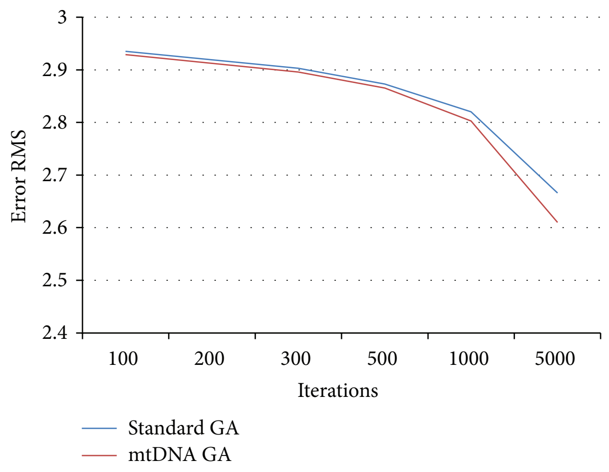

Figure 10: Graph shows error with GA mtDNA and standard GA for Function B.

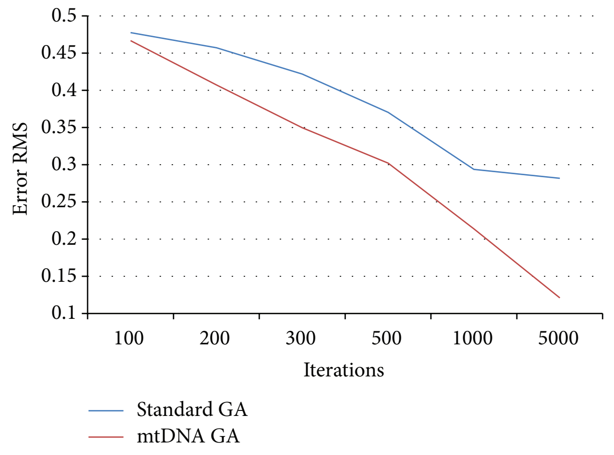

FIGURE 11: Graph shows error with GA mtDNA and standard GA for Function C.

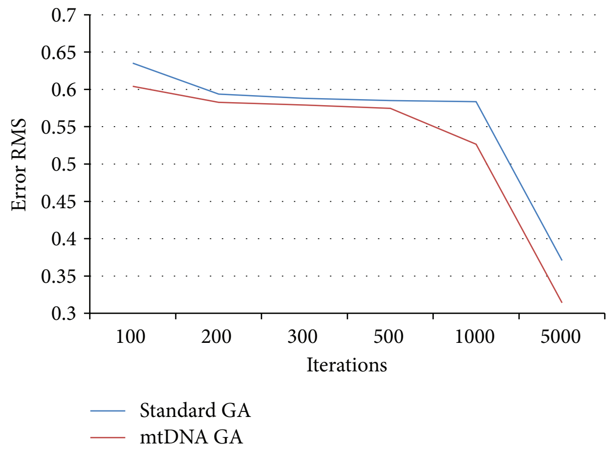

FIGURE 12: Graph shows error with GA mtDNA and standard GA for Function D.

in this paper mimics the natural order where it is an established fact that biodiversity favors evolution and produces more adaptable offspring. Thanks to faster hardware, parallel/distributed processing, and algorithms, TSP benchmarks with smaller number of cities have been solved optimally in short runtime. Larger benchmarks/problems still provide opportunities to improve algorithms. The implementation of mtDNA on small to medium sized TSP benchmarks in this paper supports its contribution in relatively improving the quality of solution. The goal of this implementation is not only necessarily to beat the runtime record of algorithms on benchmarks that have already been optimally solved, but also to provide a proof of concept of a technique that can be exploited to get better results. Likewise with *Continental Model*, we improve our results with greater exploration of the search space afforded by an additional layer of randomness and exchanges between independent implementations of Island GAs. Continental Model multiplies the benefits of Island Model by injecting more diversity and reduces the negative impact of any inherent initial biases in the individual silos of GA implementations in different systems. In addition, we were able to use the concept of mtDNA in GA beyond TSP to improve the outcome of Neural Network learning. Thus, it can be concluded from the results of this paper that Continental Model and the incorporation of mtDNA to control crossover are constructive modifications that contribute to further optimize the GA by yielding relatively better results.

7. Future Work

To extend the validity of mtDNA in GA as a generally more acceptable technique, it can be implemented and tested in other combinatorial optimization problems with much larger data (population) sets. Other novel methods of distributed and parallel computation algorithms can also be leveraged to get closer to optimal solution. The idea of mtDNA to guide the crossover function can be further refined and ingrained into the GA algorithm to achieve better results. The value of mtDNA can be made relative to the variance between the nuclear DNA sequences (city route sequences) of population members and we can restrict crossovers between members with close mtDNA proximity in addition to members with same mtDNA. We can combine the concept of mtDNA with other crossover operators and explore further optimization strategies. In addition, instead of outright prevention of crossovers between population members with same mtDNA, we can employ special operators to such crossovers to maximize diversity. To further validate the use of mtDNA concepts, we can extend the scope of the tests with more experiments and other optimization problems.

Competing Interests

The authors declare that they have no competing interests.

References

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, Mich, USA, 1975.

[2] D. Fouskakis and D. Draper, “Stochastic optimization: a review,” *International Statistical Review*, vol. 70, no. 3, pp. 315–349, 2002.

[3] D. Whitley, *A Genetic Algorithm Tutorial*, Department of Computer Science, Colorado State University, Fort Collins, Colo, USA, 1993.

[4] D. E. Goldberg and R. Lingle Jr., “Alleles, loci and the TSP,” in *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, J. J. Grefenstette, Ed., pp. 154–159, Lawrence Erlbaum Associates, Pittsburgh, Pa, USA, July 1985.

[5] P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, “Genetic algorithms for the travelling salesman problem: a review of representations and operators,” *Artificial Intelligence Review*, vol. 13, no. 2, pp. 129–170, 1999.

[6] N. M. Razali and J. Geraghty, “Genetic algorithm performance with different selection strategiesin solving TSP,” in *Proceedings of the World Congress on Engineering (WCE ’11)*, vol. 2, pp. 1134–1139, London, UK, July 2011.

[7] C. Nilsson, “Heuristics for the traveling salesman problem,” Tech. Rep., Linköping University, Linköping, Sweden, 2003.

[8] G. Reinelt, “TSPLIB—a traveling salesman problem library,” *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, 1991.

[9] S. Li, H. Chen, and Z. Tang, "Study of Pseudo-Parallel genetic algorithm with ant colony optimization to solve the TSP," *IJCSNS International Journal of Computer Science and Network Security*, vol. 11, no. 3, 2011.

[10] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[11] D. H. Nguyen and B. Widrow, "Neural networks for self-learning control systems," *IEEE Control Systems Magazine*, vol. 10, no. 3, pp. 18–23, 1990.

[12] L. Davis, "Applying adaptive algorithms to epistatic domains," in *Proceedings of the International Joint Conferene on Artificial Intelligence (IJCAI '85)*, vol. 1, pp. 162–164, 1985.

[13] D. Whitley, T. Starteather, and D. Shaner, "The traveling salesman and sequence scheduling: quality solutions using genetic edge recombination," in *Handbook of Genetic Algorithms*, L. Davis, Ed., pp. 350–372, D. Van Nostrand Reinhold Company, New York, NY, USA, 1991.

[14] S. F. Galán, O. J. Mengshoel, and R. Pinter, "A novel mating approach for genetic algorithms," *Evolutionary Computation*, vol. 21, no. 2, pp. 197–229, 2013.

[15] L. J. Eshelman and J. D. Schaffer, "Preventing premature convergence in genetic algorithms by preventing incest," in *Proceedings of the 4th International Conference on Genetic Algorithms (ICGA '91),* pp. 115–122, Morgan Kaufmann, San Diego, Calif, USA, July 1991.

[16] G. Ochoa, C. Madler-Kron, R. Rodriguez, and K. Jaffe, "Assortative mating in genetic algorithms for dynamic problems," in *Applications of Evolutionary Computing*, vol. 3449 of *Lecture Notes in Computer Science*, pp. 617–622, Springer, Berlin, Germany, 2005.

[17] I. J. Wilson, M. E. Weale, and D. J. Balding, "Inferences from DNA data: population histories, evolutionary processes and forensic match probabilities," *Journal of the Royal Statistical Society Series A: Statistics in Society*, vol. 166, no. 2, pp. 155–201, 2003.

[18] Y. Guo, Q. Cai, D. C. Samuels et al., "The use of next generation sequencing technology to study the effect of radiation therapy on mitochondrial DNA mutation," *Mutation Research/Genetic Toxicology and Environmental Mutagenesis*, vol. 744, no. 2, pp. 154–160, 2012.

[19] M. Li, A. Schonberg, M. Schaefer, R. Schroeder, I. Nasidze, and M. Stoneking, "Detecting heteroplasmy from high-throughput sequencing of complete human mitochondrial DNA genomes," *The American Journal of Human Genetics*, vol. 87, no. 2, pp. 237–249, 2010.

[20] W. Amos and J. Harwood, "Factors affecting levels of genetic diversity in natural populations," *Philosophical Transactions of the Royal Society B*, vol. 353, no. 1366, pp. 177–186, 1998.

[21] E. Hagelberg, M. Kayser, M. Nagy et al., "Molecular genetic evidence for the human settlement of the Pacific: analysis of mitochondrial DNA, Y chromosome and HLA markers," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 354, no. 1379, pp. 141–152, 1999.