ON EVOLUTIONARY ALGORITHMS FOR EFFECTIVE QUANTUM COMPUTING

By

Markus Gustav Kruger

Thesis presented in partial fulfilment of the requirements for the degree of
Master of Science in Physics at Stellenbosch University

Supervisor : Professor Hendrik B. Geyer

Co-supervisor : Professor Stephan E. Visagie

March 2012

DECLARATION

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

March 2012

ABSTRACT

The goal of this thesis is to present evolutionary algorithms, and demonstrate their applicability in quantum computing. As an introduction to evolutionary algorithms, it is applied to the simple but still challenging (from a computational viewpoint) Travelling Salesman Problem (TSP). This example is used to illustrate the effect of various parameters like selection method, and maximum population size on the accuracy and efficiency of the evolutionary algorithms.

For the sample problem, the 48 continental state capitals of the USA, solutions are evolved and compared to the known optimal solution. From this investigation tournament selection was shown to be the most effective selection method, and that a population of 200 individuals per generation gave the most effective convergence rates.

In the next part of the thesis, evolutionary algorithms are applied to the generation of optimal quantum circuits for the following cases:

- The identity transformation : Picked for its simplicity as a test of the correct implementation of the evolutionary algorithm. The results of this investigation showed that the solver program functions correctly and that evolutionary algorithms can indeed find valid solutions for this kind of problem.
- The work by Ding et al. [16] on optimal circuits for the two-qubit entanglement gate, controlled-S gate as well as the three qubit entanglement gate are solved by means of EA and the results compared. In all cases similar circuits are produced in fewer generations than the application of Ding et al. [16]. The three qubit quantum Fourier transform gate was also attempted, but no convergence was attained.
- The quantum teleportation algorithm is also investigated. Firstly the nature of the transformation that leads to quantum teleportation is considered. Next an effective circuit is sought using evolutionary algorithms. The best result is one gate longer than Brassard [11], and seven gates longer than Yabuki [61].

OPSOMMING

Die doel van hierdie tesis is om evolusionêre algoritmes te ondersoek en hulle toepaslikheid op kwantumkomputasie te demonstreer. As 'n inleiding tot evolusionêre algoritmes is die eenvoudige, maar steeds komputasioneel uitdagende handelsreisigerprobleem ondersoek. Die invloed van die keuse van 'n seleksie metode, sowel as die invloed van die maksimum aantal individue in 'n generasie op die akkuraatheid en effektiwiteit van die algoritmes is ondersoek.

As voorbeeld is die 48 kontinentale hoofstede van die state van die VSA gekies. Die oplossings wat met evolusionêre algoritmes verkry is, is met die bekende beste oplossings vergelyk. Die resultate van hierdie ondersoek was dat toernooi seleksie die mees effektiewe seleksie metode is, en dat 200 individue per generasie die mees effektiewe konvergensie tempo lewer.

Evolusionêre algoritmes word vervolgens toegepas om optimale oplossings vir die volgende kwantumalgoritmes te genereer:

- Die identiteitstransformasie: Hierdie geval is gekies as 'n eenvoudige toepassing met 'n bekende oplossing. Die resultaat van hierdie toepassing van die program was dat dit korrek funksioneer, en vinnig by die korrekte oplossings uitkom.

- Vervolgens is daar ondersoek ingestel na vier van die gevalle wat in Ding et al. [16] bespreek word. Die spesifieke transformasies waarna gekyk is, is 'n optimale stroombaan vir twee kwabis verstrengeling, 'n beheerde-S hek, 'n drie kwabis verstrengelings hek, en 'n drie kwabis kwantum Fourier transform hek. In die eerste drie gevalle stem die oplossings ooreen met die van Ding et al. [16], en is die konvergensie tempo vinniger. Daar is geen oplossing vir die kwantum Fourier transform verkry nie.

- Laastens is daar na die kwantumteleportasiealgoritme gekyk. Die eerste stap was om te kyk na die transformasie wat in hierdie geval benodig word, en daarna is gepoog om 'n effektiewe stroombaan te evolueer. Die beste resultaat was een hek langer as Brassard [11], en sewe hekke langer as Yabuki [61].

Dedication and acknowledgement

I dedicate this thesis to Sir Terry Pratchett, who more than any other person showed me the value of thought and the sad consequences of a lack thereof. His writing led me to question not only my environment, but to critically look at my responses to it, at least such is my aspiration.

My thanks as well to my supervisors, Professors S.E.Visagie and H.B.Geyer, for their patient and understanding academic as well as financial support of this work.

Then many thanks to my family and friends who's support has been invaluable in this time: My parents for their motivation and expectations, Anton Kruger, my brother who is an inspiration for the simple way he gets on with any task, Neil and Elaine Hattingh, and Ivan Louw, the friendships from school that passed the test of time, Tessa Silberbauer, for her advice and support, especially as far as writing is concerned, Jessica Chamier, Andries Gie and Bertie Barnard for being hope in times when I lost mine, Astrid Buica for being the mirror that would not allow me to hide reality from myself, Jessica and Conan Ablewhite, for sharing so much of their lives with me, Helena Wiehahn, from who I learn much and lastly to Hannes Kriel, the Oracle at the Merensky building, without the discussions with you there is no doubt that this thesis would never have happened.

Newton's first law of motion: "An object will remain at rest, or at constant velocity, unless a resultant force is applied to it." is just as applicable to life in general. To paraphrase: "Things stay the same, unless you do something to change it." Often we fall into the habit of expecting things to change for the better without any active input from us. Such is almost never the case.

CONTENTS

LIST OF FIGURES

CHAPTER 1

Project Introduction

1.1 Objective

The Goals of this thesis are as follows:

- Write an evolutionary algorithmic program to solve general TSP problems.
  - Test this program on the known TSP ATT48.
  - Investigate the influence of the program parameters on the efficiency and accuracy of the program.
- Write an evolutionary algorithmic program to solve general quantum circuit model problems.
  - Test this program on the identity transformation.
  - Investigate the influence of the program parameters on the efficiency and accuracy of the program.
  - Use the program to verify the work done by Ding et al. [16].
  - Investigate quantum teleportation using the program and compare the results with those reported by Yabuki [61].

1.2 Overview

Evolutionary methods are a subset of optimization methods known as metaheuristics. Other methods in this set include ant colony optimization, bee colony optimization, cuckoo search, and many others, as well as various combinations of these methods. As an introduction to this method I chose to apply it to the travelling salesman problem (TSP). The TSP is a very well researched problem [2, 6, 40] with known methods (such as integer programming) for solving it exactly. These methods are however very computationally expensive.

Metaheuristics provide less resource intensive methods of finding (local) optimum solutions, without the guarantee that it is the best possible solution. Metaheuristic methods are search methods and thus the size of the search space is important and in the TSP case the number of possible solutions are $(n-1)!$, where n is the number of cities that the salesman needs to visit. This result is derived by application of elementary combinatorics. Even for cases only consisting of a few cities this is a very hard problem (computationally classified as NP-complete [2]).

For the purpose of this project a general metaheuristic solver for the TSP problem was developed in Matlab, and this was tested on the problem known as ATT48, a well known problem involving the state capital cities of the continental USA with the exception of Juneau. The optimum solution to this problem is known and was included in the library that was downloaded from the University of Heidelberg [58]. This example is used to illustrate the impact of some of the control parameters for the evolutionary method on the convergence rate of the solver for this problem. These parameters are:

- Population size of a given generation
- Maximum number of generations
- The mutation probability of any reproduction operation
- The probability that crossover reproduction occurs
- The choice of selection method for breeding, if crossover occurs

In the TSP application only the influence of the selection method is considered, investigation of the other parameters is done at a later stage. Although clear and interesting trends emerge from these simulations, the investigation was not exhaustive, as this is not the focus of the project.

The next step in the project was to apply the knowledge gained from the work with the TSP solver to a Matlab simulator for quantum computer algorithms, specifically in the form of the quantum circuit model. This means that if you are presented with a unitary transformation, the solver is required to find the simplest (in terms of number of primitive gates) circuit that would be equivalent to this transformation.

To test the solver it was applied to the problem of an identity transformation of which the unique optimal circuit solution is known. It also provides a good basis for testing the algorithm parameters mentioned above, to see how they influence the convergence percentage of an ensemble of runs, as well as the convergence rate of a single run. The influence of island models was also investigated in the form of a stasis effect for the best individuals in a mature generation.

After the correct functioning of the software had been established, it was applied to the four circuits listed below, to which evolutionary algorithms where applied in the article by Ding et al. [16]:

- A circuit for maximally entangling two qubits
- A controlled S-gate
- A circuit for maximally entangling three qubits
- A quick Fourier transform gate for three qubits

The last part was an investigation into optimizing the circuit for quantum teleportation. For comparison the work by Yabuki [61] is used.

1.3 Thesis Structure

Chapter 2 starts with a discussion on metaheuristics in general and evolutionary algorithms specifically, and a brief background on their historic applications, as well as a brief description of their strengths and weaknesses. This is followed by a short history of the development of the quantum computer, and a summary of the major developments in the field.

Chapter 3 deals with the general method of applying evolutionary algorithms to a problem, and specifically in this case, that of the travelling salesman. For the purpose of this investigation only a single TSP problem with a known optimal solution is considered. This allows the investigation of the impact of changing the runtime parameters of the evolutionary algorithms, on the convergence percentage and convergence speed of the method.

The application of evolutionary algorithms to the challenges of quantum computer programming is discussed in Chapter 4. The first step to applying evolutionary algorithms is to cast the problem in a form that is compatible with such an approach. For this reason a discussion of the quantum circuit model is presented as well as the choice of representative genes, which in this case are the gates in the primitive gate set. With those steps in place the chapter moves on to the development of a software solver for the problem.

Testing of correctness and efficiency of the quantum gate solver is undertaken in Chapter 5. The influence of the solver parameters like the population of each generation, the maximum number of generations for the simulation, the mutation rate, etc. are investigated to find the influence they have on both the convergence probability as well as the convergence rate. The quantum gate solver is far more resource intensive and thus finding optimal settings is of more importance than in the case of the TSP solver.

From the paper by Ding et al. [16] we know that four gates for the quantum circuit model are commonly required in quantum computing namely, the two-qubit entanglement gate, the three qubit entanglement gate, the controlled S-gate and the quick Fourier transform gate for a three qubit system. As the equivalent transformations for these gates are known the optimization of these circuits, which are carried out in Chapter 6, is simply an application of the solver program.

Unlike the previous application of the solver, the required transformation for the quantum teleportation algorithm needs to be found, and this is done together with the application of the software solver on this transformation. The reasoning behind the form of the quantum teleportation transformation and its concurrent optimization is discussed in Chapter 7.

Chapter 8 gives a summary of the results of the investigations in this project as well as a discussion on some to the questions that arose during the process, but which falls outside the scope of this project.

CHAPTER 2

Foundations

2.1 Evolutionary algorithms

2.1.1 General metaheuristics

Evolutionary algorithms are a part of a more general set of optimization methods known as metaheuristics, which in turn is a subset of stochastic optimization [37]. Most metaheuristics have the structure shown, in Algorithm 1, in common. Metaheuristics are used when there is no clear analytical way of finding the solution to a problem, or when brute force or the analytical methods are computationally too expensive.

Algorithm 1: `General metaheuristics`

```
Input: An initial set of solutions $S_0 = \{\boldsymbol{x}_0\}$ chosen randomly or deterministically from the search space $\mathcal{S}$.
Output: A set of approximations of a global optimum solution.
1:  Set $k = 0$ and calculate the objective function value $f(\boldsymbol{x}_k) \forall \boldsymbol{x}_k \in S_k$ .
2:  repeat
3:      Generate a new set $S_{k+1}$ of candidate solutions $\hat{\boldsymbol{x}}_{k+1}$ from the search space using a
        refinement of the solutions in $S_k$ according to some optimization rules.
4:      if $f(\hat{\boldsymbol{x}}_{k+1}) > f(\boldsymbol{x}_k)$ then
5:          $\boldsymbol{x}_{k+1} = \hat{\boldsymbol{x}}_{k+1}$
6:      else
7:          $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$
8:      end if
9:      $k = k + 1$
10: until Stopping criterion is reached.
```

For a more intuitive picture of heuristics, think about Newton's method for finding the optimal value of a function. It starts with a random guess and then uses the derivative of the function to find a better solution, which then becomes the input guess for the next iteration. In Newton's method the derivative indicates the best direction to move to find a better solution. Metaheuristics aren't limited to functions with well defined derivatives though, as the indication of the best direction to move for a better solution is step can always be approximated by the difference between the fitness function at the old and new points in solution space.

In fact metaheuristics are like the A-team of optimization, often criticized for not having a solid analytical foundation, and thus they aren't guaranteed to find the optimal solution, or even get close. Further they don't have a rigid method of application, more a list of guidelines. But when you are in a tight spot and nothing else can help you, metaheuristics are often your only option.

2.1.2 Types of metaheuristics

There are many different applications of, and variations on, metaheuristic methods, and to discuss all of them is beyond the scope of this work. However, a short introduction is supplied for a few of the most important and widely used ones. It is noteworthy that many of the mechanisms that differentiate between methods have the same aim, namely to achieve faster convergence to a good solution and escaping local optima.

Random search The random search algorithm is the simplest application of metaheuristics. It corresponds to an implementation of Algorithm 1, where the second step is just another random selection of a test solution. The old and new solutions are then compared and the best is kept. This method is guaranteed to find the optimal solution if it is allowed to run for an infinite time, but it has nothing directing its progress and is therefore very slow to converge.

If the system contains no inherent structure in the fitness landscape, and you have no information about the structure of a good solution, other than how to calculate its fitness, this method will still provide results. By limiting the likelihood of choosing new solutions far away from current ones the convergence speed may be improved. If this is strongly limited the algorithm is known as a local search.

Limiting the range of the random changes may lead to the search getting stuck in a local optima. A solution to this problem is simulated annealing.

Simulated annealing By combining the solution space coverage of a random search with the convergence ability of a local search, simulated annealing tries to get the best of both methods, without their individual drawbacks. To facilitate this, the if statement in line 3 of Algorithm 1 is modified to what is shown in Algorithm 2:

Algorithm 2: Modified IF statement for simulated annealing.

```
if $f(\hat{\boldsymbol{x}}_{k+1}) > f(\boldsymbol{x}_k)$ then
    $\boldsymbol{x}_{k+1} = \hat{\boldsymbol{x}}_{k+1}$
else if with probability $1 - e^{-(\frac{f(\hat{\boldsymbol{x}}_{k+1}) - f(\boldsymbol{x}_k)}{t})}$ then
    $\boldsymbol{x}_{k+1} = \hat{\boldsymbol{x}}_{k+1}$
else
    $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$
end if
```

The idea is that if the new solution isn't better than the current one, there is still a probability of going in the wrong direction, an action that could move the search off a local optimum. The addition of the temperature parameter (t), which is set to a high temperature at the start, means that this reversal is more likely to happen in the beginning of the search, giving the search the appearance of a random walk. As the search progresses the parameter t is decreased, and in doing so the probability of guide reversal is also decreased. The number of iterations through which t goes before decreasing is called the algorithm's cooling schedule [37]. Another variation for this process would be to play around with non-linear ways of decreasing t.

For more on the development of this method see Kirkpatrick et al. [30], where the reason why the probability function resembles the Boltzmann distribution, becomes clearer. This paper also discusses an application of simulated annealing to physical chip design, which has, as an integral part, the problem of the shortest circuit paths connecting all the elements. This is similar to the travelling salesman problem that is discussed in Chapter 3. Another way of forcing the search away from local optima, would be to keep track of where you have already been. This is discussed in the next method.

Tabu list This algorithm was proposed in Glover [23], within the context of what type of algorithms executed on a machine could be viewed as intelligent. The author of this paper was also the first to use the term metaheuristics, according to Luke [37]. The idea is a simple one, keep a first in last out list of the last n good solutions and stop your search from going back to solutions on the list. This forces the search to spread out into previously uncharted directions, and so explore more of the solution space. This method has also been combined with other methods, in for example Glover et al. [24] where tabu lists are combined with evolutionary algorithms. One such way is to incorporate the tabu list into the fitness function of the evolutionary algorithm as a penalty tax, decreasing the fitness of such a individual and making it less likely to compete.

GRASP The GRASP (Greedy Randomized Adaptive Search Procedures) algorithm differs from the previous algorithms in that it repeatedly picks a starting solution and then optimizes it with a local search. The initial solution can be a random solution selected from the solution space, or more likely it is a solution created using inherent knowledge of the system as to what would be a good solution. For example, in a TSP application a good initial guess would be a path that doesn't cross itself. In itself this isn't enough to guarantee a optimal solution, but we know that the optimal solution has this characteristic.

The best solution over different attempts is kept. As each initial random guess is independent of the previous search, it prevents the algorithm from getting stuck in a local optima. This algorithm is easily extendable on a system with parallel processing capability, where many instances of the same algorithm can be run concurrently and only the best solution out of such an ensemble is kept. For further elaboration on this method and tweaks and variations see Festa and Resende [19].

Ant Colony Optimization According to Luke [37], ant colony optimization (ACO) was introduced in a Ph.D thesis by Dorigo [18]. There is a lot of literature from Marco Dorigo and Thomas Stützle, on ACO and various applications of this algorithm. These include application to the TSP in Stutzle and Dorigo [56], which is of interest in this thesis. Ant colony optimization represents a departure from the methods discussed so far.

This is the first system where the building blocks across different solutions are competing with each other on a basis that isn't just the best fitness across the whole ensemble. As with GRASP the first part is a creation of a valid solution using prior information about the problem. In the TSP problem this would be creating a circuit by starting at a specific city and then randomly going from that city to the next, based on a probability that depends on a pheromone strength as an indicator of how good the transition is, and so forth until all cities have been visited only once. Once such a tour has been formed, the transition from one city to another in this tour is incremented with a value (pheromone strength) that is based on the overall fitness of this tour. So transitions used in good tours become more likely to be chosen again and ones that haven't been in use have their probabilities stagnate.

After each round of creating solutions the pheromone strength of each transition is decreased by a set amount. This action is an attempt to counter takeover of early good performers. After this step you could add an additional local search step where the best solutions are refined. Initially this method was only utilized for discrete solution spaces, but has been extended to continuous solution spaces as well [35, 53].

Guided local search Guided local search algorithm uses the pheromone idea in reverse, it marks the components of local optima solutions, and actively discourages these choices during the solution construction part of the algorithm, and thereby favors exploration of the search space.

2.1.3 Evolutionary algorithms.

My interest in this field originated with Thompson [57], where he describes the evolution of a frequency discriminator that is both simpler and uses effects that would never have been used had the circuit been designed by regular engineering principles. The ability of this search to find the - outside the box - solutions fascinates me.

The inspiration for this algorithm comes from nature. Evolution is the change in the characteristics of a population transferred from one generation to the next [22]. It is driven by natural selection of these characteristics that are encoded in genetic material that gets passed from parents to children. In nature this process has been shown to produce individuals that are adapted to their environments. By replicating this process in an optimization setup we could take advantage of this adaptive quality. At its root it is a random search that is given direction by letting individuals compete for survival and reproductive dominance, while keeping a memory of the process in the form of the surviving individual's genetic code.

Evolutionary algorithms also depart from the pattern of the algorithms we have considered thus far. Instead of looking at one solution at a time and from that attempting to improve the solution, evolutionary algorithms work with an ensemble of solutions (called a generation) taken from the entire solution space. Solution are thus not just competing with historical versions of a solution, but are competing with other solutions at the same time. This strengthens the exploration of the search space. A representation of a general EA is provided in Algorithm 3.

Algorithm 3: Algorithmic representation of evolutionary algorithms

```
Input: The control parameters, including the population size, maximum number of
generations, mutation rates, selection methods for parent selection, elitism percentage.
Output: An approximation of a global optimum solution.
1: Set generational number ($G$) to zero.
2: Create an initial population of solutions selected randomly from the search space. ($P_0$)
3: Evaluate the fitness of each individual $f(p_i)$ in the population.
4: repeat
5:     Increment $G$
6:     Generate a new population $P_G$ from $P_{G-1}$ according to the recombination rules of the
       input parameters.
7: until Stopping criterion is reached. This could be a optimal generation, or when a
   solution of sufficient fitness has been evolved.
8: Return best evolved solution.
```

The concepts and parameter names might initially sound foreign to the ear of a mathematician, but when you view this as a biological analogy of the problem, the organic nomenclature becomes almost self explanatory. Each solution becomes an individual organism, and in a single generation all the solutions together represent the population for that generation. To get to the heart of the method, the gene manipulation, one has to first make the connection between the solution of the problem and the genetic representation of such a solution. Such a representation isn't unique and every application of evolutionary algorithms have, as a first step, to find a sensible way of making this conversion.

If the solution is in the form of a vector of a fixed length, then the value of each coefficient could be linked to a gene. If the solution is in the form of a circuit, then the circuit could be subdivided and each part modeled as an individual gene. Even when looking at the same problem this representation may differ, for example the ways in which Ding et al. [16] and Yabuki [61] encode their circuits for evolution.

In this thesis evolutionary algorithms will be applied to the TSP problem as well as to the problem of evolving quantum circuits, and in each case the method of encoding is guided by both the requirements of the problems as well as the structure of the tools for modeling the solutions. In this case the solutions are modeled in Matlab and as such, using vectors and matrices is preferred, as Matlab deals efficiently with these structures.

Once this genetic representation has been fixed, we are left to find ways of reproduction, which are again direct analogies with the reproduction that takes place in nature. Non-sexual reproduction would involve simply copying a solution from one generation to the next, with a chance of mutation during the process. (This corresponds to a local search) Then there is sexual reproduction which is the creation of offspring using gene from multiple parents. This is very effective at exploring the solution space, without losing all the information that made the parents good solutions.

Other methods for countering takeover and false convergence to local optima, are available and often based on techniques taken from nature. These include Island models [1], where populations develop separately for a time before competing in the bigger population.

A new development that is showing some promise is the idea of encoding gene positions as quantum states. Each possible gene is seen as a state, and the value of the gene position is a superpositions of all these possible expressions of these genes. Thus each gene position will be an n-tuple where n is the number of possible gene values for that gene position. This n-tuple will consist of the coefficients of each possible state, and will represents the probability amplitude of finding a specific gene expressed. A rather comprehensive account of all the different implementations of this approach is given in Zhang [62].

2.2 Quantum computers

2.2.1 Historical background

Inspired by the work of Ed Fredkin, the idea of a quantum computer was introduced in a keynote speech presented by Richard Feynman [20]. David Deutsch [13] formalized the ideas, and it has since then grown into a fertile research field. The central idea is to encode the information in a register that contains qubits (a two state quantum system). The states of this register is then allowed to evolve under a controlled transformation that will act as the computational core. Finally, it is required to retrieve the processed data through measurement of the final register. In a noiseless system the computation simply amounts to a unit vector in the Hilbert space $\mathcal{H}^{\otimes n}$ (n is the number of qubits and $\mathcal{H} = span\{|0\rangle, |1\rangle\}$), transforming via a unitary transformation to another unit vector in the same space, thus just complex linear algebra.

In his talk Feynman [20] explained the difficulty of simulating a quantum mechanical system with a classical computer. The problem that he highlighted is that when working with the classical world view, computers can simulate physics to arbitrary accuracy, within certain limits of course. The amount of memory the computer needs scales linearly with the number of particles that we are working with. This is a hard problem for computers as most real systems have a large number of particles, but it is not impossible, given enough computing resources.

The situation changes dramatically when you try and do things on a quantum level though. The memory requirement of a n-particle quantum system grows exponentially with n. Even with large resources this quickly becomes intractable. But instead of giving up at this point, Feynman goes further to ask: "If a classical computer is no good for this task, can we then use a quantum system, to assist in solving problems of a quantum nature?". He then gives reasons why he thinks this is possible.

These are the ideas that David Deutsch [13] further extended into what we today see as the quantum computer. He also came up with the first application of quantum computing that showed that it had advantages over the classical computing model, even for problems that where not strictly quantum mechanical in nature. Enter the era of quantum algorithms.

2.2.2 Quantum algorithms

In his paper [13], Deutsch shows that if you have an algorithm that calculates the value of an input parameter x, encoded in qubits in the input register, to give $f(x)$ encoded in the output register. Then by preparing the input register as an equal superposition of all possible values of x, and operating the same algorithm, the output is an equal superposition of all the possible values of $f(x)$. This is a remarkable result as it enables massive parallel processing on a single processor. Practically however it is not so valuable as you can only recover one of these values, selected at random, by measurement at any given time. It did however show that quantum computers had potential.

This opens the way for further progress in the form of the Deutsch-Jozsa algorithm [14], a black box interrogation algorithm that is much more efficient that any classical method. This idea was further expanded by Berthiaume and Brassard [8]. In 1992 a paper on quantum teleportation was published Bennett et al. [7], highlighting another ability that was unique to quantum computing.

The next high profile advance was that of Shor [51], famous for scaring everyone using public key encryption protocols on the internet. Shor found an algorithm that could factorize large numbers into prime factors, in polynomial time, and hereby circumventing the primary obstacle to cracking RSA-encryption. Two years later Grover [25] published a quantum algorithm for searching through an unordered database with n entries to find a specific entry in order $\sqrt{n}$ passes (with high probability). This is much faster then any classical algorithm.

These are by no means a comprehensive list of quantum algorithms, but it covers the ones that are encountered in the literature most often. All the algorithms for quantum computers would be of no value if there wasn't any quantum computers to execute them on. The next section deals with some of the attempts at creating working quantum computers.

2.2.3 Quantum computers

Creating a quantum computer is a most challenging endeavor. For a system to act as a quantum computer, the first requirement is that the register be isolated from the environment. Only the unitary operation of the algorithm should be able to transform the register. This is known as decoherence of the register states, and in practice this is a very hard problem to solve. Currently the only alternative to isolation is quantum error correction first discussed by Shor [52] and Steane [55] independently.

The first step to a physical quantum computer was the work done by Monroe et al. [41]. In this paper he showed how to apply the 2 qubit conditional not (CNOT) gate to a quantum register comprised of two trapped supercooled atoms. The significance of this is that the CNOT gate together with a single qubit phase gate form a universal gate set, and could be used to approximate any unitary transformation arbitrarily well. This is a necessary step for building a working quantum computer.

In DiVincenzo [17], the properties that a system need to have to be a good candidate for a quantum computer are given. These are:

- A scalable physical system with well characterized qubits: The states represented by n qubits in this space should allow access to all the states in the space $\mathcal{H}_1 \otimes \mathcal{H}_2 \otimes .....\mathcal{H}_n$.
- The ability to initialize the state of the qubits to a simple fiducial state, such as $|000\rangle$. If you can't control the input to your computer then there can't be any sensible computation done.
- Long relevant decoherence times, much longer than the gate operation time. If this was not the case, you would not be able to trust the computed values.
- A universal set of quantum gates. This is the heart of the quantum processor. If you don't have a universal gate set, there will always be operations that can't be executed, or approximated arbitrarily closely.
- A qubit-specific measurement capability. To recover the computed results you need to be able to measure the state of the output register, but more than that, some algorithms like teleportation, require measurement of individual qubits independently.
- The ability to interconvert stationary and flying qubits.
- The ability to faithfully transmit flying qubits between specified locations.

The last two requirement aren't needed for an isolated quantum computer, but information transfer is a critical part of information science, and therefore essential if quantum computers are going to find a place in common information technology.

Finding systems that satisfy these criteria has been the focus of much research. In Ladd et al. [33] an overview of relevant systems is provided, as discussed below.

Photons The first encoding system discussed it that of photon polarization. This type of encoding is very decoherence resistant with lifespans of the order of 0.1s. Creating single qubit gates for this kind of encoding is easily accomplished using wave plates. A two-qubit interaction gate that is necessary for universal computing is more difficult though. The current best method for working with polarization encoded qubit is the Knill-Laflamme-Milburn [32] framework. In Politi et al. [48] it is shown how the number 15 can be factored by Shor's algorithm using this framework.

Trapped atoms Trapped atoms and ions have the longest decoherence stability of any of the methods discussed here. Lifetimes of more than 10s have been achieved [34]. Scaling with this encoding is currently the most challenging part of practical implementation.

According to Ladd et al. [33] an implementation of up to eight trapped ions were realized in Blatt and Wineland [9]. It becomes increasingly difficult to effectively cool more and more ions, but a solution was suggested by in Olmschenk et al. [46]. Using photon interaction the distance between trapped ions could be greatly increased and thus the cooling problem is reduced.

Another method, that of using coherent states in Bose-Einstein condensates to represent qubits, was suggested by Morsch and Oberthaler [42].

Nuclear Magnetic Resonance The initial advantage of NMR technology was that it had been around for half a century, so the operational controls, as well as the theory behind its operation was very well understood. This was then a very good vehicle for experimentation in quantum computing. Initial experiments were done using the spin of atoms in liquid to represent the qubits, and with decoherence lifetimes of more that a second, this initially looked promising. On the scaling side, the method also showed promise, with 12 qubit systems operationally shown by Negrevergne et al. [43], and Vandersypen et al. [59] using another 12-qubit computer to factor 15 using Shor's algorithm.

NMR has a problem with qubit initialization and the current best implementation does not scale well, in liquid NMR, but there are applications of solid state NMR that show more promise [39].

Quantum dots and dopants in solids What quantum dots, dopants and certain impurities in solids have in common is that they create a single electron semi-conductor state that can either be occupied or not, or be in a superposition of the two states. This allows for the encoding of information in the electron occupancy of such structures. Currently the implementation of this method that shows the most promise uses Nitrogen sites in diamond [4].

Having to depend on these impurities occurring naturally is fine for a laboratory and proof of concept system, but for mainstream applications, control over the positioning is very important. To this end Schneider et al. [49] has made valuable progress.

Superconductors Superconductors open up another realm for quantum computing. Superconducting micro circuits in the μm range, allow the manipulation of charges and currents on scales that allow for the encoding of qubits using these quantities. These circuits can be made small enough and close enough to couple inductively and capacitively, thus allowing quantum gates [45]. An application of various algorithms using two qubits encoded with superconducting circuits were done by DiCarlo et al. [15].

The Canadian company D-wave claims to have found a way to construct a 128-qubit adiabatic quantum computer that is made to run a quantum annealing algorithm [21] to solve multidimensional optimization problems. Whether this process works isn't a matter of consensus yet, but this potential amalgamation of quantum computing and heuristic algorithms, seems like a good closing remark for this chapter.

CHAPTER 3

TSP as an application of Evolutionary algorithms

3.1 The travelling salesman problem

In order to apply and test the capabilities of evolutionary algorithms, a problem was needed to apply it to. The ideal problem would be one that was already well investigated with a proven solution, for comparison purposes. It would also be advantageous if the formation of solution lent itself to a genetic description.

The Travelling salesman problem (TSP) was chosen and may be stated as follows: Given a set of n destinations with fixed travel cost between each; find the cheapest tour from the home city, that visits each city exactly once, and then returns to the starting city [2]. For more on the history of this problem, and other related combinatorial problems, see Schrijver [50]. In this paper the roots of this problem is traced back as far as 1838, but the first mathematical treatment seems to be K. Menger in his investigation in 1930 into the messenger's problem.

As an interesting aside, in the introduction to this paper Schrijver mentions how these kinds of pathfinding problems are prevalent even in nature and how various animals, including ants have found ways of finding solutions for it. Many of the current heuristic methods for finding solutions to these types of problems were inspired by natural phenomena, with ant colony algorithms proving especially useful.

This is not the only application to this type of problem. Apart from the various transport optimization related applications, there are applications in the fields of warehouse control, stacking crane priorities, forklift tasking, as well as in production processes, where the drill, solder or welding sequence for robotic arms need to be optimized. As a tool for investigating these, a compilation of various TSP type problems, with their best solution, if available, have been compiled into a single resource called TSPLib. The version that will be used in this investigation comes from the University of Heidelberg [58].

The simplicity of the problem statement belies the complexity of finding an optimal solution. A systematic way for finding solutions only became available with the advent of linear programming in the late 1930s. For an n city system there are $(n-1)$ ways of getting from the starting city to the second, $(n-2)$ ways to the third, etc. Thus there are $(n-1)!$ possible solutions. Even for small values of n this makes a brute force approach ineffective $(O(n!))$. Even methods like linear programming, specifically integer programming, are very resource intensive $(O(2^n))$ [28]. However, there exists heuristic methods like ant colony simulation [56] and evolutionary algorithms [10] that can find very good solutions at a fraction of the computational cost of linear programming methods.

TSP is thus well suited as a test application of the proposed evolutionary approach.

3.2 Evolutionary methods and example application to the TSP

Two important considerations when working with these type of algorithms are the efficiency, which is defined in terms of the convergence rate or half search time, and the accuracy, which describes how close to the optimal solution is to the best evolved tour.

To asses the solver program's performance, it was used to find the shortest route for a salesman visiting all the continental state capitals of the USA as shown in Figure 1[1]. This problem is designated as ATT48 in the TSPLib, which contains the verified best solution for this problem, and as such provides a benchmark for the solutions that the program generates. A representation of the optimal solution is shown in Figure 2.

ATT48 is both complex enough to test the effectiveness of the code, but simple enough that the program can be run with the computational power available.

A single tour (or individual) is represented by a list of labels describing the order in which each of the cities are visited. The starting city is labeled as 1, so the list will start and end with 1 and contain a permutation of the numbers 2 to 48 in between those eg. $\overrightarrow{t} = \{$1, 45, 44, 8, 21, 22, 7, 30, 13, 27, 31, 25, 15, 23, 4, 39, 11, 29, 34, 42, 33, 9, 16, 46, 20, 37, 43, 36, 12, 32, 6, 5, 17, 2, 14, 41, 3, 40, 47, 26, 35, 24, 18, 38, 28, 48, 19, 10, 1$\}$. The vector $\overrightarrow{t}$ describes a tour that starts at city 1 and goes to city 45, then 44 etc. The initial generation contains a thousand of these individuals, chosen randomly. The tour length in kilometers, of each individual can

[1]Large figures, graphs, and tables are published in the appendix

be calculated as the sum of the distances between consecutive cities for the whole tour. The intercity distances are read from a table, which in this case, is symmetric. Other measures of fitness, even asymmetric ones, like the cost of air tickets, or the time taken when flight schedules are incorporated, could easily be incorporated into this model as a different table. This would not cost any processing time beyond that needed to create the initial table.

The program is run for five thousand generations, where each consecutive generation is constructed from the individuals in the previous, with higher priority given to the better solutions in the generation. The tour finding program code and the definition of the parameters files are available by email from MarkusKruger@yahoo.co.uk. These parameters control the various ways of creating the new generation from the current one. The fine tuning of these parameters for solving ATT48 is not the focus of this investigation, but they are discussed briefly, along with their effects on the efficiency and the accuracy of the program, in the next subsections.

3.2.1 Diversity

One of the challenges when using evolutionary algorithms is that of pre-optimal stagnation. Selection pressure tends to lower the diversity from one generation to the next, because the offspring of successful individuals are more likely to be successful than any randomly selected individual. This domination of initial strong individuals is known as a takeover. If this is not countered the population may eventually consist of individuals in close proximity to a set of local optima, a phenomenon known as crowding. As the next generation will then consist of the offspring of this closely related group of individuals, not many new solutions are introduced into the next generations, and thus the solution base stagnates. If, for example, a generation already contains the best possible individual, the algorithm has no choice but to stagnate around this solution, and here stagnation is not a problem, as no new generation needs to be generated. If the same happens around a local optimum solution, however, the algorithm might never be able to get out of this fitness well (a neighborhood of the local optimum).

Each individual may be considered as a memory of what is a good or bad pathway, depending on its fitness. If all the good individuals are the same or closely related, then our knowledge of what a good solution is, is not well developed. However, if there are many good solutions that are distinctly different, then a lot of information about good solutions is available.

The solution to this stagnation problem is thus one of diversification of the solution base in each generation. Diversity is a measure of how well the population of a specific generation is distributed throughout the whole solution space. Initially, if a reasonable metric can be defined on the solution space, diversity can be improved by selecting the initial population so that all individuals are at least a given distance from all the others. A similar result could be attained by putting a net, with a number of holes equal to the population size, over the solution space, and then selecting one individual from each of the holes.

Increasing diversity in later generations is controlled by moving around in the algorithm's parameter space. Much of the time spent on finding a solution is taken up in finding parameters that maximize diversity, and stave off pre-optimal stagnation. The next sections will discuss ways of achieving this goal.

3.2.2 Reproduction

Reproduction is the process of creating individuals in the next generation from the ones in the current generation. Firstly, it needs to attempt to retain as much of the memory of what a good solution is, and at the same time add to this knowledge in the next generation. It is a balancing act between transferring the characteristics of an individual that makes it a fit candidate, but to change it in a way that makes it distinct from its parent(s). There are many ways of attempting to get this balance right.

3.2.2.1 Elitism and incremental replacement

The first form of reproduction, Elitism totally ignores the idea of diversification, and is only concerned with preserving information about the best solution. This is done with the transfer of fittest individuals (the elite) from one generation to the next. This would mean that the current best solution will never be lost, but at the cost of hastening the takeover by these individuals.

Figure 3 shows that selecting these elites from the random population, and carrying it to the next generation (again containing only the Elites and a new set of random individuals) already causes some selection pressure (Elitism (1%), Mutation (0%), Random (99%)). This results in a solution with a fitness that is 199% of the best known fitness and therefore not really usable. The previous search is a random search with a genetic way of representing the individuals. This is intentional to show that the engine that drives evolutionary algorithms is only a random search that is given direction by the feedback it gets from the fitness evaluation.

To counter takeover, one could limit the number of generations an individual could be active in the population. This has the potential of relieving crowding, and stem the normal decrease in diversity. Often used options of this kind is to either kill off older individuals, or if the population needs more selective pressure it is advisable to kill off the worst individuals.

Another strategy is to implement an extinction level event (ELE). In such an event, a very large portion of the population is removed and replaced by random new individuals. This event is a reset of the state of development of the population, and normally leads to improved selection pressure. One adjusted implementation is that of putting the best individuals in stasis for the time it took to reach a stagnating state, and then wipe the entire population. After running for a time close to the time it took to stagnate, these stasis bound individuals will be reintroduced into the population. The development of such an implementation would mirror concurrent development of non-interacting, or weakly interacting, populations. This is known as an island model [1].

Another, less radical, way in which diversity may be increased is by altering the individual from one generation to the next. This is a process known as mutation.

3.2.2.2 Mutation

Mutation is a random or directed change in the genetic structure of an individual that leads to genetically different offspring. The simplicity of this process makes it a common method for population diversification, but it has less diversifying potential than crossover, which is covered next. It can, however, effectively slow crowding.

In the solver, mutation is introduced when a new individual is generated. The mutation rate gives the probability that the mutation operation would be executed. Mutation is implemented by picking a random point in the genome (say city A) as well as a random number of genes (tour length B). From city A the next B cities have their visiting sequence inverted. This is a bit of a cheat as it isn't completely random exchange, but this type of change causes either crossing or uncrossing of routes, and the result is thus either much better or much worse.

The results of adding mutation to elitism is shown in Figure 3. Firstly, 1% of the population is designated as elites and transferred. The rest of the population is built up by creating offspring, either by mutating a parent from the current generation, or generating a new individual. Both options are granted equal probability. The result is a greatly increased selection pressure indicated by the much steeper convergence curve (Elitism (1%) Mutation (49.5%), Random (49.5%)). Secondly, only mutated offspring of the previous generation are used, with no new random individuals (Elitism (1%), Mutation (99%), Random (0%)). This gives even faster convergence. For each of these runs the best individual after five thousand generations gives a difference of 4.85% and 3.42% respectively, relative to the known optimal tour length. Both of these are good approximations, but consistent improvements on the results require contributions from the whole population. This cannot be achieved with elitism and mutation alone. To produce greater diversity, another form of reproduction is needed.

3.2.2.3 Crossover

Picking two (or more) parents and then selecting genes from each to form the next generation is more difficult to implement than mutation, but offspring created this way are more likely to be significantly different from their parents, and thus significantly reduce crowding and loss of diversity. There are multitudes of ways of performing crossover, and testing their influence on the algorithm would be interesting, but it is outside the focus of this thesis.

Finding a sensible way of reproduction by crossover can be challenging where there is interdependence in the genes representing an individual. In this case, the interdependence is caused by the fact that the representations are permutations of the same set of numbers. Selecting a splicing position and exchanging genes before or after that point will not necessarily lead to a valid individual, and thus more care is needed.

In the solver, the problem is addressed by selecting the first half of the tour as represented by parent 1, and then queueing the remaining cities in the sequence that parent 2 would visit them. This solution leads to a greater contribution from parent 1, but the testing run results show that it is still effective.

Finding the best way of selecting parents is the subject of the next part of the chapter.

3.2.2.4 Selection methods

The purpose of a selection method is to select the parents of next generation in such a way as to both optimize the selection pressure of an algorithm, as well as the diversity within any given generation. For an in-depth discussion of the effectiveness of each in a simplified environment, see Hancock [26]. According to Baker [3] there are two approaches to selection that could easily be distinguished, namely sampling and selection. Sampling implies that you go through each individual and choose the number of reproduction opportunities according to its fitness, while selection assigns a probability of reproduction to each individual according to its fitness, and then selects, according to a stochastic process, the parents of the next generation. Sampling selects the parents as a group all at once, while selection selects them one at a time. The advantages of sampling over selection is stressed in Hancock [26], but selection methods are seldom implemented in such an isolated fashion, and with the addition of elitism, some of the negative impact of selection can be negated.

Figures 4, 5 and 6 are the convergence graphs representing the first of two ensembles of five runs for each of the 12 selection methods discussed below. The first ensembles were done with no elitism, no crossover, and 100% chance of mutation. Full mutation was selected as partial mutation in this case would be the same as a probabilistic application of elitism. The second ensemble of graphs, Figures 7, 8 and 9, represents the same parameters with the addition of one percent elitism. With elitism enabled, even the random selection, experiences some selection pressure. The top left graph in Figure 4 gives the results of just randomly selecting a group of individuals for each generation. This shows no improvement in the fitness of the best solution as generations progress, and is in line with what you would expect from randomly searching through the solution space, without retaining any knowledge gained in previous generations. This graph can thus be used as a control for the efficiency of the other selection methods. In contrast to this lack of selection pressure, the top left graph of Figure 7 shows the improvement adding 1% elitism brings to this random search.

Fitness proportionate selection (FPS) FPS is the most direct selection process. The probability of reproduction of an individual is directly proportional to its fitness. This is sensitive to a simple translation of the fitness scale [26]. There are two problems associated with this kind of selection. The first is negative fitness values that need to be treated in a different way as not to cause negative probabilities. A fix for negative fitness values would be to use the relative fitness of individuals relative to the worst individual in any given generation. This would then cause fluctuations in the reproductive probability of the best individual if it where to transfer from one generation to the next. This is an inconsistent way of selection. Setting the fitness of all individuals with a negative fitness to zero could solve both these problems, but still does not address the arbitrariness of this kind of selection.

The second problem is the kind we face in ATT48. The total path length is a natural fitness indicator in this case, but as shorter is better, straight application of FPS with this as fitness measure would lead to the least fit individual getting a better chance at reproduction than the fittest one. To use FPS a redefinition of the fitness function is needed, something like $F = \frac{1}{P}$ where F is the fitness of and individual and P is the total path length of the solution represented by this individual. The top righthand graph on Figure 4 shows that this type of selection performs very poorly with pre-optimal stagnation and terrible accuracy. By adding the 1% elitism the results improve considerably as shown in the top right graph of Figure 7.

Roulette selection If the fitness of the individuals are strictly non negative, this simplest version of fitness proportional selection can be applied. Each individual is assigned an interval proportionate to its fitness, within a larger continuous interval. Parents are then selected by choosing a random number in the total interval and selecting the individual in who's interval the number lies. Just like a roulette wheel with different width spokes. In this experiment FPS and Roulette selection's implementations are virtually the same, and this similarity is reflected in the results shown in bottom right graphs of Figure 4 and Figure 7 respectively.

Windowing To address negative fitness values, and the fluctuations caused by the relative fitness with the least fit individual in a generation, fitness can be redefined as the difference between a given value and the fitness of an individual. This baseline value may be the least fit individual's fitness, or this quantity, averaged over many generations. Alternatively the minimum fitness over the interval could be used. In the case of averaging the baseline, negative fitness values are still possible, but these will then be set to zero, effectively culling these individuals. The number of generations over which this averaging or over which the minimum is done, is known as the window size.

Sigma scaling Sigma scaling starts as FPS, but a baseline value σ equal to the average fitness minus the standard deviation, for the fitness within a single generation, is set, and all fitness are calculated relative to this value. Any individual below σ will have its reproductive probability set to zero. This still suffers from reduced selection pressure as the population fitness approaches the value of the fittest individual in the generation. The results or this type of selection is displayed in the bottom left graphs of Figures 4 and 7 respectively.

Stochastic universal sampling (SUS) SUS is essentially the same as roulette selection except that instead of selecting with a random number in the interval, a net of the right number of holes is placed over the interval and the middle of the holes then select the parents. In contrast to selection methods where the fittest individual still has a chance of non-selection, in a sampling method the fittest always get selected for at least one offspring. The results are plotted in Figure 5 and Figure 8 respectively.

Ranking selection An alternative to using the direct fitness value is simply to depend on the ranking of an individual based on its fitness. Commonly used schemes are presented.

1. **Linear ranking** The individual is assigned a weight of

$$w(p) = \frac{2(P-p)}{(P-1)^2}, \tag{3.1}$$

   where P is the population size and p is the ranking of the individual. This weight is normalized and may be treated as a probability, that can then be used in a selection or sampling method application. The results of this method in selection is represented in top right graphs of Figure 5 and Figure 8, while the results of using sampling is shown in the bottom left graphs of Figure 5 and Figure 8.

2. **Exponential ranking** Following the same procedure as with linear ranking but using an exponential scale with $s \in (0, 1)$, to calculate the weight as:

$$w(p) = \frac{(1-s)s^{(p-1)}}{1-s^P}. \tag{3.2}$$

   Again the weight is normalized and doubles as a probability. In the case of this ATT48 run, with $s = 0.95$, the selection results are shown in the bottom right graphs of Figure 5 and Figure 8, while the sampling results are shown in the top left graphs of Figure 6 and Figure 9.

3. **Gaussian (Normal) ranking** The normal distribution using the ranking position has the added benefit of giving the experimenter two control parameters. By setting the average other than the best individual it is possible to give some of the other individuals a better breeding chance. This could be an advantage when elitism would already ensure the survival of the best individual. Having control over the standard deviation is a useful fine tuning parameter. For these runs the average was set to position one, and the standard deviation was set to 10% of the population, thus 100. The results are shown in graphs at the top right, and bottom left of Figure 6 as well as Figure 9.

Tournament selection This is probably the simplest way of using ranking, although it does have a slight processing overhead. For every parent a random set of a chosen size is selected from the population. Within this set the best individual is then chosen to become a parent. For these runs the tour size was chosen as fifty individuals per tournament. The results is represented in the last graphs on Figure 6 and Figure 9, as the cases without and with 1% elitism, respectively.

3.3 Summary of the work done on ATT48

To compare the accuracy and efficiency of the these reproduction methods, it is assumed that the search far from the stagnation point follows a exponential decline in the difference between the best fitness value in a generation and the true best fitness. This implies the following relationship:

$$F(t) - F_T = \beta 2^{-\alpha t}.$$

In the above equation $F(t)$ is the best fitness in generation t, while F_T is the shortest route, which in this case is 33524 km [58]. The parameter β is a scaling constant for the problem and is related to the way the fitness is defined, while α is the inverse of the half search time. The half search time gives the number of generations that will halve the difference between the current best value and the true best value. The value of α is an indicator of the efficiency of a particular parameter set, in the sense that a bigger value for α would indicate a faster convergence rate.

The results of fitting this model to the data is summarized in Table 1, Table 2 and Table 3. The fitting was done with Matlab's built in curve fitting application cftool, as the straight line fit to the data set $\{t \in [1, t_e] : (t, \log_2(F(t) - F_T))\}$. The parameter t_e is the maximum value for t for which the data followed a straight line pattern (this corresponds to the assumption that it isn't stagnating).

This model fit well for all the methods with elitism enabled, but did less well for the non-elite methods. Without elitism FPS only had two runs that fit with R^2 value > 0.95. Roulette and SUS had no fits better than 0.95. Roulette had a best fit of 0.83, and the worst fit was 0.47. For SUS the best fit was R^2 value of 0.93 and the worst 0.76. Linear Ranking Selection got a single fit less than 0.95, with $R^2 = 0.947$. Apart from these all the other fits had R^2 values exceeding 0.95. It is clear that this model fits much better when elitism in enabled, or when observing the best solution found up to generation t, rather than the best fitness in generation t.

Summaries of these results are represented in Figure 10, Figure 11, Figure 12 and Figure 13. To compare the different methods it is sensible to start with the accuracy as shown in Figure 13. The specific ranking for the different selection methods is given in Table 4. Both normal ranking, sampling (1^{st}) and selection (2^{nd}) did very well, but up to exponential ranking selection (7^{th}), there is very little difference in the performance. All these methods give solutions that are within 500km of the best solution. In general elitism seems to improve accuracy, with the exception being tournament selection and exponential ranking sampling. It is interesting to note that random selection with elitism, manages to produce answers that are within 5% of the best tour length.

The efficiency of the selection methods should be seen in the context of the accuracy, as accuracy is the primary concern in this investigation. Table 5 is sorted by half search time, but also secondarily by accuracy. Again there is very little difference between the first ten selection methods, Exponential ranking sampling is best, but very closely followed by tournament selection without (2^{nd}) and with (3^{rd}) elitism. Elitism seem to not have such a clear influence on the convergence tempo, as it does on the accuracy. It mostly has a detrimental effect on the methods that converge quickly (tournament selection and normal ranking), but an accelerating effect on those that converge slower, like roulette, SUS and random selection.

CHAPTER 4

Quantum computing, primitive gate sets and evolutionary algorithm implementation

The first step in applying evolutionary algorithms to quantum computing is to find a way of casting the problem into a gene representable form. Such a mapping is not a simple matter when looking at the general time evolution of a quantum system, but when it is viewed as an algorithm, there is a structure and a flow that could be what we are looking for.

In the conference presentation [20], R.P. Feynman describes the necessary conditions for using a quantum system as a simulator of quantum phenomena in nature. Deutsch [13] further expanded this idea in showing that a quantum computer is a universal machine, by application of the Church-Turing principle. It states: "Every finitely realizable physical system can be perfectly simulated by a universal model computing machine operated by finite means." The limits this puts on the parts of the machine and the process of computation has direct implications for the computational classes of problems that can be solved with such a computer.

Feynman, very elegantly, showed how a classical universal machine could not simulate a quantum system by finite means, but that a computer working on quantum mechanical principles could. To use a quantum system as an information processor, the information needs to be encoded in the state of particles, which we call qubits (if it is a two-state system), while the processing is done with unitary transformations on these states. For a two state system like spin the state of a qubit (q_1) is described by the two numbers $\alpha, \beta \in \mathbb{C}$, so that $|q_1\rangle = \alpha\,|{\uparrow}\rangle + \beta\,|{\downarrow}\rangle$. Spin up ($|{\uparrow}\rangle$) encodes zero, and spin down ($|{\downarrow}\rangle$) encodes one. Generalization to systems with more than two states simply requires the addition of more coefficients, one for each state. The quantum computer then consists of a register of $n \in \mathbb{N}$ *qudits*, which are $d \in \mathbb{N}$ state objects. In general d takes the value two, but examples of higher than binary systems exist and the operational principles are exactly the same. Objects with more that two states are no longer referred to a qubits, but rather *qudits* (for d state systems).

The user of the quantum computer has control over the initial state of the qubits in the register, thereby encoding the input values for the computation. The system is now allowed to evolve in accordance with a predefined set of unitary transformations (the program of the quantum computer). Finally the processed information is extracted by one or more measurements on the qubits in the register. An assumption is made that there is no interference (decoherence of the register) from any outside sources with this register, changing its state. The realization of this condition is one of the biggest hurdles that needs to be overcome in creating a practical quantum computer.

There are, even for a single qubit, an infinite number of unitary transformations, but according to Steane [54] it is possible to generate all unitary transformations, to within arbitrary accuracy, by repeated applications of a two-qubit controlled not transformation (CNOT gate), and a single qubit rotation transformation ($R(\overrightarrow{\theta})$ gate), where the rotation angles are irrational fractions of both π and each other. The CNOT gate will flip the second qubit's state if the control qubit is in state $|1\rangle$, otherwise it will not change anything, while the rotation gate will rotate the state vector in Hilbert space. To understand the effect of the rotation gate better the analogy of the rotation of a unit vector on a two dimensional circle would be instructive.

By rotating the vector an irrational fraction of π, no integer number of application will ever give exactly an integer number of rotations. It can be shown that for any arbitrary chosen rotation angle there exists and integer number of applications of this rotation that will get you arbitrarily close to the chosen rotation angle. Thus you could use this to rotate any vector to arbitrarily close to any other chosen vector. The single rotation gate $R(\overrightarrow{\theta})$ does the same in the Hilbert space of a single qubit. The CNOT gate will couple different qubits together and together these two gates can be used to create a transformation that is arbitrarily close to any chosen unitary transformation on this space and is thus an example of a universal gate set.

Now that we have a register and a processor toolkit (the universal gate set) it is simple to describe the evolution of the register in terms of the transformations acting in turn on the various qubits in the register. This lends itself to a circuit-like representation, the so-called quantum circuit model.

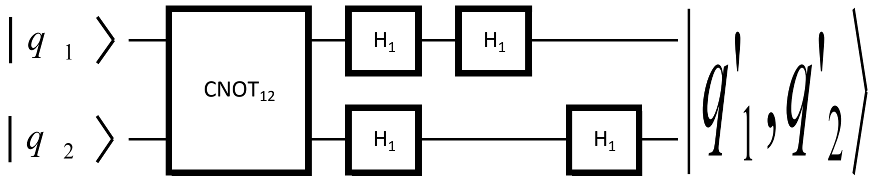


Figure 4.1: Example quantum circuit.

4.1 Quantum circuits

The time evolution of the registry of a quantum computer can be represented as:

$$|Q_1, Q_2, \ldots, Q_n\rangle = \mathbf{U}\,|q_1, q_2, \ldots, q_n\rangle\,.$$

Here $|Q_1, Q_2, \ldots, Q_n\rangle$ are the final qubit states and $|q_1, q_2, \ldots, q_n\rangle$ are initial qubit states. The operator $\mathbf{U}{=}U_tU_{t-1}\ldots U_1$ where U_i is the the unitary transformation of the register from state $i-1$ to state i. The operator U_i for each step is the Kronecker product of all the gates acting on the $i-1$ state's register.

An easier way of keeping track of the individual transformations and which qubits they are acting on, is to use the quantum circuit model. In this model, which closely resembles a classic circuit, the qubits are represented by horizontal lines and the transformations as gates acting on the their specific qubits (horizontal lines). The input register is represented on the left and the output register on the right, while the sequential application of the transformations is given by the relative position between the input and output registers. Figure 4.1 is an example of a two-qubit quantum circuit where a controlled not operation using q_1 as control, is followed by

Hadamard transformations [2] on both qubits, then on the first one only and then only on the second qubit. By using the sequential nature of the quantum circuit, it is easy to transcribe it as a genetic structure where each gene will code the gates for a fixed number of time-steps as is shown in Table 4.1.

Gene 1	Gene 2	Gene 3	Gene 4
$CNOT_{12}$	H	H	I
$CNOT_{12}$	H	I	H

Table 4.1: Figure 4.1 circuit transcribed to genetic structure

Having found a way of representing the evolution of a quantum registry, what is left to find a suitable primitive or universal gate set.

4.2 Choosing a primitive gate set

In Section 4.1 it was mentioned that the two-qubit CNOT gate and a specific class of rotation gates were universal, however they are not the only primitive set. Because irrational ratio rotations are numerically problematic due to rounding effects, it is easier to work with a different set of gates. One such set is given in Yabuki [61], but the set given in Nielsen and Chuang [44], Kempe et al. [29] and Ding et al. [16] with the addition of a reverse CNOT gate was simpler to implement. Table 4.2 shows the gates that will be used in the further simulations.

The number in brackets behind the gate designation gives the number that is used to represent the gates in the genetic code. The genetic representation of Table 4.1 would be $\begin{bmatrix} 9 & 3 & 1 & 3 \\ 9 & 3 & 3 & 1 \end{bmatrix}$ in the solver program.

The problem of decoherence increases with each gate that is introduced into the circuit, and as such circuits with fewer gates are preferred. Thus, although Ding et al. [16] showed that $CNOT_{21}$ can be constructed with an application of $CNOT_{12}$ and the other one qubit gates, it would be more economical to replace them by a single gate which is just a $CNOT_{12}$ with

[2]Hadamard transforms are rotations that take state $|0\rangle$ to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|1\rangle$ to $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$

$$I(1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad S(2) = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \qquad H(3) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$P(4) = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix} \qquad R(5) = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{8}} \end{bmatrix} \qquad S^{\dagger}(6) = \begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix}$$

$$P^{\dagger}(7) = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{-i\pi}{4}} \end{bmatrix} \qquad R^{\dagger}(8) = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{-i\pi}{8}} \end{bmatrix}$$

$$CNOT_{12}(9) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad CNOT_{21}(10) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Table 4.2: Table of primitive gate set

its inputs swapped. The $CNOT$ gates are the only interaction gates in this set and therefore very common. As such the addition of this gate to the set will greatly simplify circuits.

4.3 The implementation of the evolutionary algorithm.

The general structure of this evolutionary algorithm applied in the following chapters, does not differ from that discussed in Section 2.1 and the previous two sections have shown how the genetic structure can be constructed from the quantum circuit model.

Using the genetic structure described in Section 4.2 mutation and crossover can now be implemented. The genetic description for this problem has fewer constraints than the TSP and means both mutation and crossover is easier to effect. Crossover is implemented at gene creation when the new gene is selected from either parent 1 or 2 with equal probability. Mutation is implemented at same stage. It takes the form of a probabilistic choice of either keeping the gene as dictated by the parents, or randomly selecting a gate from the available gate set to replace this gene. This process is repeated for all genes in the child's genome.

What remains is the definition of a metric for measuring the relative fitness of the circuits the algorithm produces. Two factors were taken into account in this investigation, correctness of the circuit for its purpose as well as the number of gates that the circuit contains. The correctness is assessed in one of two possible ways depending on the information that is available about the transformation. If the complete transformation matrix is known, as in the case of the identity experiment, the metric defined in Ding et al. [16] can be used directly. This is however not always the case, and the transformation has to be calculated using the information about the desired registry transformations. This is a linear algebra problem and can be sorted into three possibilities: single solutions, over-constrained system with at best some least squares solution and finally an under-constrained system with multiple solutions. Once this target transformation has been found the previously mentioned metric can be used. The correctness (C) of a circuit is given by the Frobenius inner product [27]:

$$C = \frac{|\operatorname{tr}(T^{\dagger}H)|}{2^n}. \tag{4.1}$$

T is the goal transformation matrix, H is the transformation generated by the circuit and n is the number of qubits in the registry [16].

A measure of the correctness of the circuit, from the point of view of computation would be in calculating the maximum difference between the probability distribution of the output registers of the goal transformation and that of the circuit. A method of doing this would be:

$$C_p(|a\rangle) = \sum_{\langle b| = \langle 000|}^{\langle 111|} \left| \langle b| T |a\rangle^2 - \langle b| H |a\rangle^2 \right|. \tag{4.2}$$

In Equation 4.2 $|b\rangle$ runs over all possible measurable outputs, and $|a\rangle$ is any possible input. In Knill [31] it is shown that $C_p(a) \leq 2C$ and thus by minimizing C the two probability distributions are also forced to converge.

When evaluating two circuits that both give the required transformation, preference is given to the simpler circuit, the one that has fewer active steps (non-identity time-steps) as well as giving a preference for individual time-steps that contain as many identity transformations as possible. The fraction of time-steps that are identity is given a weight of $\frac{S-1}{S}$, while the fraction of the circuit that consists of identity gates, get a weight of $\frac{1}{S}$. These weights were chosen so that a single identity time-step would still outweigh any number of identity gates, while the maximum reward is a number between 0 and 1. Thus reward for efficiency (R) is can be calculated as:

$$\begin{aligned} R &= \frac{S_i}{S} \times \frac{S-1}{S} + \frac{I}{N} \times \frac{1}{S} \\ &= \frac{S_i(S-1)}{S^2} + \frac{I}{SN}. \end{aligned} \tag{4.3}$$

S refers to the maximum number of time-steps in the circuit, S_i is the number of these that are identity transformations and can thus be left out, I is the total number of identity gates in the system, and N is the total number of possible gates $(N = S \times n)$.

Both correctness (C) and efficiency (R) are values between 0 and 1, so for easy distinction on the graphs the fitness value (F) is given by the following expression:

$$F = (C-1) + \begin{cases} R & \text{if } C \geq (1-T) \\ 0 & \text{else} \end{cases}. \tag{4.4}$$

The expression Equation 4.4 would give values of less that zero for circuits that are not valid solutions, and value that are between zero and one for circuits that are valid solutions. The transition from non-correct to correct solution appears on the graphs where the graph crossed the x-axis. T is the fault tolerance for evaluated algorithms, and is a definable input parameter. The fault tolerance gives the maximum difference between the probability distribution of the final register after implementing the goal transformation and that of implementing the circuit. It also provides a way to compensate for rounding errors, or if your experimental setup has a limit on either the accuracy of measurements, or the accurate implementation of gates, then evolving circuits that are correct to this level of accuracy is both sensible and expedient. The fault tolerance parameter also allows for this.

CHAPTER 5

Program testing with the identity transformation

5.1 Motivation for identity tests

The purpose of this set of experimental runs is to test the evolutionary solver program with a problem for which the solution is both known as well as unique. The identity transformation is such a transformation. It can be constructed in many ways using the other gates in the chosen gate set Table 4.2, yet there is a clear best solution that is given by filling all the circuit slots with the identity transformations. The simplicity of the solutions also creates the opportunity to compare several selection methods, test the introduction of a stasis method, and investigate the influence of changing population size on the convergence speed of the program.

The results for effective parameter choices in this case does not imply that they will be equally effective in the cases discussed in later chapters, but they do suggest a sensible starting point in parameter space.

5.2 Experiments

5.2.1 Experiment 1: Functionality test

This first experiment is the testing of the evolutionary program's ability to consistently find the simplest identity circuit with a registry consisting of two qubits. Tournament selection with tours of 11 individuals was used, because it proved effective in the TSP application (see Figures 10 to 13) and is simple to implement (Section 3.2.2.4).

The parameters for these experimental runs are summarized in Table 5.1 and the target transformation is given by Equation 5.1

$$T_g = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.1}$$

Maximum Population Size	200	Register Size	2 qubits
Number of Genes	10	Gene length	1 Gate
Fitness Goal	1	Maximum Generation	1000
Elitism Percentage	10 %	Mutation Rate	15%

Table 5.1: Parameters for experiment A1

Twenty five runs were done using these parameters. The form of the graphs in Figure 14 corresponds to the results in Chapter 3, with allowance made for the fact that the fitness was defined opposite (smaller fitness is better in Chapter 3) to the definition here. This correspondence becomes clearer when looking at the inserted graph of the average fitness for the 25 runs. The more pronounced step characteristic is a result of the much smaller search space, 10^{12} vs 10^{59} for the TSP ATT48 in Chapter 3.

The solver program produced circuits that gave the identity transformation 100% of the time, without any errors in the program. The results can further be broken up into solutions that contain gates other than identity gates (fitness values $\in (0,1)$) and the optimal solution with a fitness value of 1. The left graph in Figure 15 gives a summary of how many generations were needed to get to the optimal result (if this was reached). A large portion of the runs (60%) converged to the best value within 100 generations, while only 16% did not reach optimal convergence by the 1000^{th} generation. 100 generations gives only 20000 solutions that were searched through out of a possible 10^{12}, a most efficient ratio.

The second graph on Figure 15 shows the final fitness distribution of the 25 runs. Here 84% of the runs produced a final fitness of 1 which is optimal, and the rest produced a fitness of between 0.6 and 0.8 which correspond to 3 non-identity gates.

In conclusion, these runs show that the solver program works as expected and is efficient at finding the solutions to this problem.

Maximum Population Size	100	Register Size	3 qubits
Number of Genes	10	Gene length	1 Gate
Fitness Goal	1	Maximum Generation	1000
Elitism Percentage	10 %	Mutation Rate	25%
Normal Distr. μ	1	Normal Distr. σ	25
Tournament size	5		

Table 5.2: Parameters for experiment A2

5.2.2 Experiment 2: Selection method comparison

The second experiment was to test the efficiency and effectiveness of the different selection methods. In an effort to make the results of the test more distinguishable from each other, the search space was increased by adding another qubit to the registry.

The search space for this investigation was considerably larger than for the first set of experiments, and the parameters were not optimized, hence the poor convergence percentages. Despite the non-convergence, trends can still be seen in the results summarized in Figure 16, Figure 17 and Figure 18. The graphs on the left give the distribution of the termination generations of the 25 runs. A shift to the left in these graphs indicate a faster convergence to the optimal solution. The graphs on the right shows the distribution of the quality of the solution. In this case a shift to the right in the histogram points to better solutions. Table 5.3 shows a summary of the results. There is a clear indication that tournament selection is outperforming the other selection methods by a comfortable margin, however neither the efficiency nor the accuracy of any of the methods were satisfactory with the current parameter sets. To improve the performance of these selection methods, some of the diversification enhancements discussed in subsubsection 3.2.2.1 will be tested next.

5.2.3 Experiment 3: Impact of the Stasis effect.

The stasis effect improves diversification of the solution set by restarting the development of a population as soon as it starts to stagnate. To counter the loss of the best solution, these are put in stasis, while the new population develops. When the new population has also developed to the point where they can compete with the best from the previous population, the individuals from stasis are reintroduced. In this way the information about good solutions from the last population isn't lost, but neither do these individuals take over the new population.

	Selection Method				
	Random	Linear	Normal	Roulette	Tournament
Average termination fitness without Stasis	0.51	0.55	0.65	0.61	0.75
Average termination fitness with Stasis	0.65	0.51	0.95	0.92	1.00
Average termination generation without Stasis	1000	1000	963	984	697
Average termination generation with Stasis	1000	1000	538	794	298

Table 5.3: Summary of efficiency and effectiveness of different selection methods, with and without the stasis effect

Maximum Population Size	100	Register Size	3 qubits
Number of Genes	10	Gene length	1 Gate
Fitness Goal	1	Maximum Generation	1000
Elitism Percentage	10 %	Mutation Rate	25%
Normal Distr. μ	1	Normal Distr. σ	25
Tournament size	5	Number of Individuals in stasis	20
Generations to reintroduction	73	Stagnation Factor	0.4

Table 5.4: Parameters for experiment A3

The results from the previous experiment are used as a control for the effect of the stasis parameter. Figure 19 through Figure 23 show the results of the five chosen selection methods with and without the stasis effect. This improvement is evident from the final generation distribution moving to earlier generations, and the higher occurrence of good solutions in the final solution distribution. The results are also summarized in Table 5.3. All the methods, with the exception of linear selection, performed better in both effectiveness and accuracy with the stasis effect activated. Tournament selection still gives the best results, but with stasis activated, Roulette and Normal selection also become options.

Selection methods and measures like the stasis effect can have a large influence on the convergence speed and accuracy of the solver program, but so can simply changing the number of individuals in a generation.

Register Size	3 qubits	Number of Genes	10
Gene length	1 Gate	Fitness Goal	1
Maximum Generation	1000	Elitism Percentage	10 %
Mutation Rate	25%	Max Mutation Rate	31%
Normal Distr. μ	1	Normal Distr. σ	25
Normal Distr. Max σ	31	Max Tournament size	5
Number of Individuals in stasis	20	Generations to reintroduction	73
Stagnation Factor	0.4		

Table 5.5: Parameters for experiment A4

5.2.4 Experiment 4: Population influence on convergence speed

The population size of each generation directly influences the processing power needed to evaluate each generation. It is therefore worth looking at the influence of this parameter on the performance of the solver program. The results of the previous two experiments presented in Table 5.3 show that normal selection and tournament selection, both with the stasis effect enabled, are the optimal methods to use in this simulation and hence this investigation will only make use of these two methods.

Using the common parameters listed in Table 5.5 for each selection method and for each of the maximum populations $\{50, 100, 200, 500\}$ an ensemble of 25 runs were generated. The results are summarized in Table 5.6. The total search space for a 30 gate system is larger that 10^{18}, the coverage column in the table refers to the total number of tried individuals as a fraction of the possible ones. As there are likely to be duplications from generation to generation, this is an upper bound for the searched space. Smaller coverage values indicate more effective searching of the solution space. From the results tournament selection emerges as the optimal selection method, and a maximum population of 200 individuals gives the best compromise between the runtime of the program, and its convergence rate.

5.3 Summary

The main results from this set of experiments are as follows:

- The evolutionary algorithm program works both effectively as well as correctly.
- The optimal selection method seems to be tournament selection.

Selection Method	Population per generation	Average runtime(s)	Average convergence generation	Coverage	Convergence Persentage
Normal Distribution	50	138	529	3×10^-14	76%
	100	187	424	4×10^{-14}	100%
	200	321	368	7×10^{-14}	100%
	500	381	367	2×10^{-13}	100%
Tournament selection	50	124.64	474	2×10^{-14}	72%
	100	132	295	3×10^{-14}	100%
	200	124	139	3×10^{-14}	100%
	500	225	102	5×10^{-14}	100%

Table 5.6: Summary of results for experiment A4

- The addition of stasis method improves the convergence speed as well as the convergence percentage.
- A maximum population of 200 individuals gives the most effective results.

CHAPTER 6

Investigating the results from Ding et al. [16]

6.1 Introduction and goals

After verifying the functionality of the solver program and finding the parameters that give efficient results in Chapter 5, using the solver to investigate more challenging problems can confidently be attempted. The problems selected are four case studies discussed in Ding et al. [16]. They are:

- A circuit for maximally entangling two qubits
- A controlled S-gate
- A circuit for maximally entangling three qubits
- A quick Fourier transform gate for three qubits

The solver was used in an attempt to find the solution to each that requires the least number of gates in the solution, by running an ensemble of 25 runs for each case. The first two cases are the simplest, as they only have two qubits in the register.

6.2 Experiment 1: Gate for maximal entanglement of two qubits

$$T_g = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{bmatrix}. \tag{6.1}$$

The entanglement of qubits is a feature of quantum computing that has no analog in classical computing. An entangled state of two qubits is exactly that, a single state describing the two

Maximum Population Size	100	Register Size	2 qubits
Number of Genes	6	Gene length	1 Gate
Fitness Goal	1	Maximum Generation	100
Elitism Percentage	10 %	Mutation Rate	15%

Table 6.1: Parameters for experiment B1

Gene 1	Gene 2
H	$CNOT_{12}$
I	$CNOT_{12}$

Table 6.2: Best solution for B1

qubits that cannot be described in terms of the states of the individual qubits. This property allows many of the unique actions that a quantum computer can perform. An example of such an operation is the teleportation algorithm investigated in Chapter 7.

The transformation for maximal entanglement[3] of two qubits is given by the expression shown in Equation 6.1. With this transformation as the target, the solver program was run using the parameter set shown in Table 6.1. All 25 runs in the ensemble converged to the same circuit given in Table 6.2, which corresponds precisely to the result that Ding et al. [16] arrived at.

6.3 Experiment 2: a Controlled S-gate transformation

$$T_g = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix}. \tag{6.2}$$

The S-gate is a single qubit phase gate that doesn't change $|0\rangle$ or the probabilities of finding a qubit in the $|0\rangle$ or $|1\rangle$ states, but does change the relative phase of $|1\rangle$ by π. A controlled S-gate is a two-qubit gate that applies the single qubit S-gate to the second qubit, when the first qubit is in the $|1\rangle$ state.

[3]See Plenio and Virmani [47] for a discussion on measures for entanglement.

Maximum Population Size	200	Register Size	2 qubits
Number of Genes	10	Gene length	1 Gate
Fitness Goal	1	Maximum Generation	500
Elitism Percentage	10 %	Mutation Rate	25%

Table 6.3: Parameters for experiment B2

Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Gene 6	Gene 7	Gene 8
P	I	$CNOT_{12}$	I	I	I	I	$CNOT_{12}$
I	P	$CNOT_{12}$	S	S	S	P	$CNOT_{12}$

Table 6.4: Best solution for B2

The experiment's ensemble took longer to run than that of experiment B1. The resulting circuit is considerably more complex and the convergence rate is much poorer. Only 52% of the runs gave a serviceable circuit, and only 40% gave the circuit described by Table 6.4, which was optimal for the gate set that the solver was working with. In the article by Ding et al. [16], the $P^\dagger$-gate (or inverse of the P-gate) is also included in their gate set, whereas it was not included in the solver's gate-set.

Their best solution [16] to this problem contains two P-gates, a $P^\dagger$-gate, and two $CNOT_{12}$-gates as shown in Table 6.5. This would account for the difference in the results from their experiments and the one obtained here, where the optimum solution contained eight gates rather than five.

After including the $S^\dagger$ and $P^\dagger$ gates to the gate set another 25 runs were generated and this time the optimal circuit corresponded to Table 6.5. All 25 run converged to this circuit, and 24 of them converged within 100 generations.

Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
I	$CNOT_{12}$	I	$CNOT_{12}$	P
P	$CNOT_{12}$	$P^\dagger$	$CNOT_{12}$	I

Table 6.5: The best solution for B2 from Ding et al. [16]

Maximum Population Size	100	Register Size	3 qubits
Number of Genes	5	Gene length	1 Gate
Fitness Goal	1	Maximum Generation	200
Elitism Percentage	10 %	Mutation Rate	15%

Table 6.6: Parameters for experiment B3

6.4 Experiment 3: Transformation to maximally entangle three qubits

$$T_g = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{6.3}$$

The register size has increased for the next two simulations. This has the effect of not only increasing the number of possible gate combinations but also increasing the overhead for calculating the effective transformation of a solution. Both of these factors negatively influence the runtime of the simulations.

For this case the transformation that is required is given by the expression shown in Equation 6.3. To find a solution for this transformation an ensemble of 25 runs was done using the parameters given in Table 6.6. In spite of this increased overhead the solver quickly converged, in all cases, to the solution for the maximal entanglement of three qubits given in Ding et al. [16]. This circuit is shown in Table 6.7. For most of the runs this convergence happened within the first twenty generations, even faster than the convergence for the controlled-S transformation.

Gene 1	Gene 2	Gene 3
I	I	$CNOT_{21}$
H	$CNOT_{12}$	$CNOT_{21}$
I	$CNOT_{12}$	I

Table 6.7: Best solution for B3

Maximum Population Size	100	Register Size	3 qubits
Number of Genes	10	Gene length	1 Gate
Fitness Goal	1	Maximum Generation	1000
Elitism Percentage	10 %	Mutation Rate	15%

Table 6.8: Parameters for experiment B4a

6.5 Experiment 4: QFT3 transformation

$$T_g = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \end{bmatrix} \text{where } \omega = e^{\frac{i\pi}{4}}[44]. \tag{6.4}$$

The Quantum Fourier Transform was by far the biggest challenge for the solver. The solver's design was purposefully constrained to not allow non-adjacent qubits to interact, as this is a challenge when constructing even small register size quantum computers. This non-adjacent interaction is required in the circuit for this transformation[4].

[4]see Nielsen and Chuang [44], p. 220

Maximum Population Size	200	Register Size	3 qubits
Number of Genes	15	Gene length	1 Gate
Fitness Goal	1	Maximum Generation	100
Elitism Percentage	10 %	Mutation Rate	15%

Table 6.9: Parameters for experiment B4b

Maximum Population Size	200	Register Size	2 qubits
Number of Genes	5	Gene length	1 Gate
Fitness Goal	1	Maximum Generation	100
Elitism Percentage	10 %	Mutation Rate	25%

Table 6.10: Parameters for the two-qubit SWAP-gate ensemble

The initial attempts did not achieve any success. Two ensembles using the parameters in Table 6.8 and Table 6.9 where run, but neither converged to a workable circuit. To find where the barrier to convergence lay each of the elements of the solution circuit given in Nielsen and Chuang [44] was investigated individually. This circuit shown in Table 6.14 contains a Hadamard gate which is already part of the gate set, a controlled-S gate (CS_{12}) (introduced in Section 6.3), and both a Swap-gate ($SWAP_{13}$) and controlled-P gate (CP_{13}), which has not been introduced thus far.

6.5.1 The $SWAP_{13} - gate$

The Swap gate ($SWAP_{13}$) is a combination of three two-adjacent-qubit swap gates, as discussed in Ding et al. [16].

$$T_g = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.5}$$

The transformation for a two-adjacent-qubit swap gate is given in the expression shown in Equation 6.5, and was verified with the solver program to have a circuit that corresponds to the ones described in Ding et al. [16]. This is just a combination of three $CNOT$-gates as shown in Table 6.13. These two circuits shown in Table 6.13 are totally equivalent in both function as well as number of primary gates, and as such it was not surprising that they were found in equal number in the results of the 25 run ensemble for which the parameters are given in Table 6.10.

From this result it is expected that the $SWAP_{13}$-gate, would be a nine gate combination circuit. To verify this another 25 run ensemble was done with the goal Equation 6.6, and

Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
$CNOT_{12}$	$CNOT_{21}$	$CNOT_{12}$	I	I
$CNOT_{12}$	$CNOT_{21}$	$CNOT_{12}$	$CNOT_{12}$	$CNOT_{21}$
I	I	I	$CNOT_{12}$	$CNOT_{21}$

Gene 6	Gene 7	Gene 8	Gene 9
I	$CNOT_{12}$	$CNOT_{21}$	$CNOT_{12}$
$CNOT_{12}$	$CNOT_{12}$	$CNOT_{21}$	$CNOT_{12}$
$CNOT_{12}$	I	I	I

Table 6.11: Known circuit for $SWAP_{13}$-gate

Maximum Population Size	200	Register Size	3 qubits
Number of Genes	9	Gene length	1 Gate
Fitness Goal	1	Maximum Generation	100
Elitism Percentage	10 %	Mutation Rate	25%

Table 6.12: Parameters for the $SWAP_{13}$-gate ensemble

parameters Table 6.12.

$$T_g = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.6}$$

None of the runs converged to a viable solution, in fact there was very little convergence that happened at all. The best fitness in the initial generation and the best in the final stayed constant in 7 out of the 25 runs, and convergence in the other runs was very slight. This is very unusual behavior for the solver, so in an attempt to find the cause the solver was initialized with a circuit that is known to be a solutions shown in Table 6.11.

Gene 1	Gene 2	Gene 3
$CNOT_{12}$	$CNOT_{21}$	$CNOT_{12}$
$CNOT_{12}$	$CNOT_{21}$	$CNOT_{12}$

Gene 1	Gene 2	Gene 3
$CNOT_{21}$	$CNOT_{12}$	$CNOT_{21}$
$CNOT_{21}$	$CNOT_{12}$	$CNOT_{21}$

Table 6.13: Solutions to the two-qubit SWAP circuit.

Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Gene 6	Gene 7
H	CS_{12}	CP_{13}	I	I	I	$SWAP_{13}$
I	CS_{12}	I	H	CS_{12}	I	I
I	I	CP_{13}	I	CS_{12}	H	$SWAP_{13}$

Table 6.14: Best solution for B4 [44]

In this run there was no improvement on this circuit, which isn't unexpected if this was also the optimum solution, but as this solution isn't unique the lack of alternates is unexpected. For the next run the known circuit was modified by replacing the last gene with the identity gene. This run showed the same behavior as the initial run where there was no incremental improvement as is normally observed, but rather just a jump to the known best after 30 generations. The jump is not convergence behavior, but rather a reflection of the random choice of changing the last gene to a $CNOT$-gate.

The conclusion that is drawn from this behavior is that the fitness defined with respect to the $SWAP_{13}$ transformation is flat over most of the domain, with a spike at the single correct solution. This would be a very hard problem for evolutionary methods to solve. This behavior also leads to further understanding of the initial difficulty in evolving a circuit for the QFT transformation. The non-adjacent interactions require some form of swap gate and it is these that are causing the poor convergence. By adding an adjacent qubit swap gate to the gate set this problem might be alleviated.

Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
R	$CNOT_{21}$	I	$R^\dagger$	$CNOT_{21}$
I	$CNOT_{21}$	R	I	$CNOT_{21}$

Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
$CNOT_{21}$	$R^\dagger$	I	$CNOT_{21}$	R
$CNOT_{21}$	I	R	$CNOT_{21}$	I

Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
R	I	$CNOT_{21}$	$R^\dagger$	$CNOT_{21}$
I	R	$CNOT_{21}$	I	$CNOT_{21}$

Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
I	$CNOT_{12}$	I	R	$CNOT_{12}$
R	$CNOT_{12}$	$R^\dagger$	I	$CNOT_{12}$

Table 6.15: Circuits for CP_{12}-gate

6.5.2 The $CP_{13} - gate$

To investigate the controlled-P (CP_{13}, a gate that applies the P-gate to qubit 3, depending on the state of qubit 1) it is sensible to attempt to find a circuit for the two adjacent-qubit equivalent CP_{12} which could then be combined with a couple of swap gates to form the CP_{13} circuit. Examples of circuits for CP_{12} are shown in Table 6.15. The examples shown here where not the only solutions that was found but all the solutions contained five gates in total, two $CNOT$ gates and the other three R and $R^\dagger$ gates. From the solutions for CP_{12} and the $SWAP_{12}$ gates, it follows that the CP_{13} can be constructed with eleven gates.

6.5.3 Final attempt evolving QFT circuit

With all of these elements one more attempt will be made to evolve a circuit for QFT. Because of the difficulty of evolving the $SWAP_{13}$-gate that is needed at the end of the circuit, it will be factored out of the transformation, as was done by Ding et al. [16]. The goal transformation will thus be given by Equation 6.7. The swap, controlled-S and controlled-P gates will also be

Maximum Population Size	200	Register Size	3 qubits
Number of Genes	9	Gene length	1 Gate
Fitness Goal	1	Maximum Generation	500
Elitism Percentage	10 %	Mutation Rate	25%

Table 6.16: Parameters for the final QFT ensemble

added to the gate set of the solver.

$$T_g = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \end{bmatrix} \text{ where } \omega = e^{\frac{i\pi}{4}} [44]. \tag{6.7}$$

An ensemble of 25 runs were generated using the parameters given in Table 6.16, but still without any success. Another ensemble of 25 runs with the number of genes expanded to 10 also did not give any positive results, even expanding to 20 genes did not achieve any success. It does seem that the solver program does not work efficiently when swap gates are required. Other than identifying the problem no headway has been made in finding a solution.

The best solution that the solver found had a fitness of -0.00192 and is shown in Table 6.17. This comes very close to the required transformation for QFT. From Figure 6.1, it is clear that there is a barrier at a fitness of -0.00192, with none of the attempts finding a solution above this value. When initialized with the solution given in Ding et al. [16], the solver recognizes it as a valid solution, but can't improve upon it. Finding a way around the swap gate problem would be a good problem to concentrate on as a follow-up project.

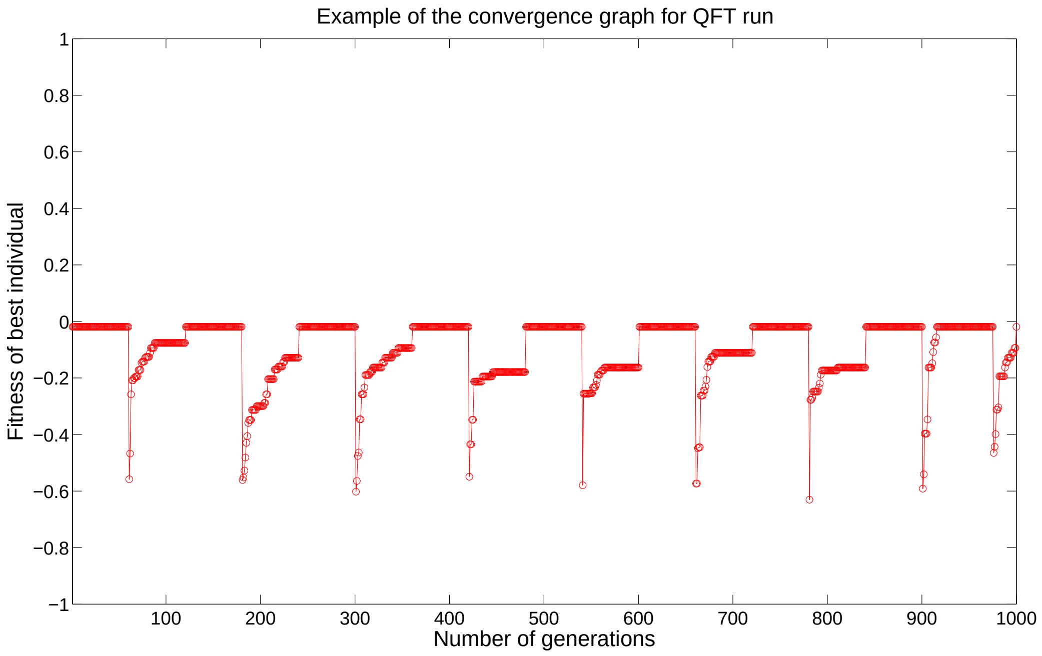


Figure 6.1: Example of a 1000 generation run of the solver, initialized with the best solution.

Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Gene 6	Gene 7	Gene 8
H	I	$CS12$	I	P	I	I	$R^\dagger$
I	I	$CS12$	H	I	CS_{12}	I	I
I	R	I	I	I	CS_{12}	H	I

Table 6.17: Best solution for QFT produced by the solver, with $SWAP_{13}$ gate at the end factored out.

6.6 Conclusion

With the exception of the problems with QFT circuit, the solver performed as well or better than the one used in Ding et al. [16]. It converged to the best solution faster, more regularly than the algorithm used by Ding et al. [16]. Further investigation into the problem with the swap gate is needed.

CHAPTER 7

Quantum Teleportation

7.1 Introduction

In Bennett et al. [7] there is a good description of quantum teleportation, and Brassard [11] frames the idea in terms of the operation of a quantum computer. Teleportation can be described as follows. Say a person, referred to as Alice, needs to send the state of her qubit to another person, called Bob. Alice and Bob both agree on an encoding scheme for information in this particular quantum system. They each obtain one of a pair of entangled qubits. Alice can now perform operations on both her entangled qubit and the one encoding the information

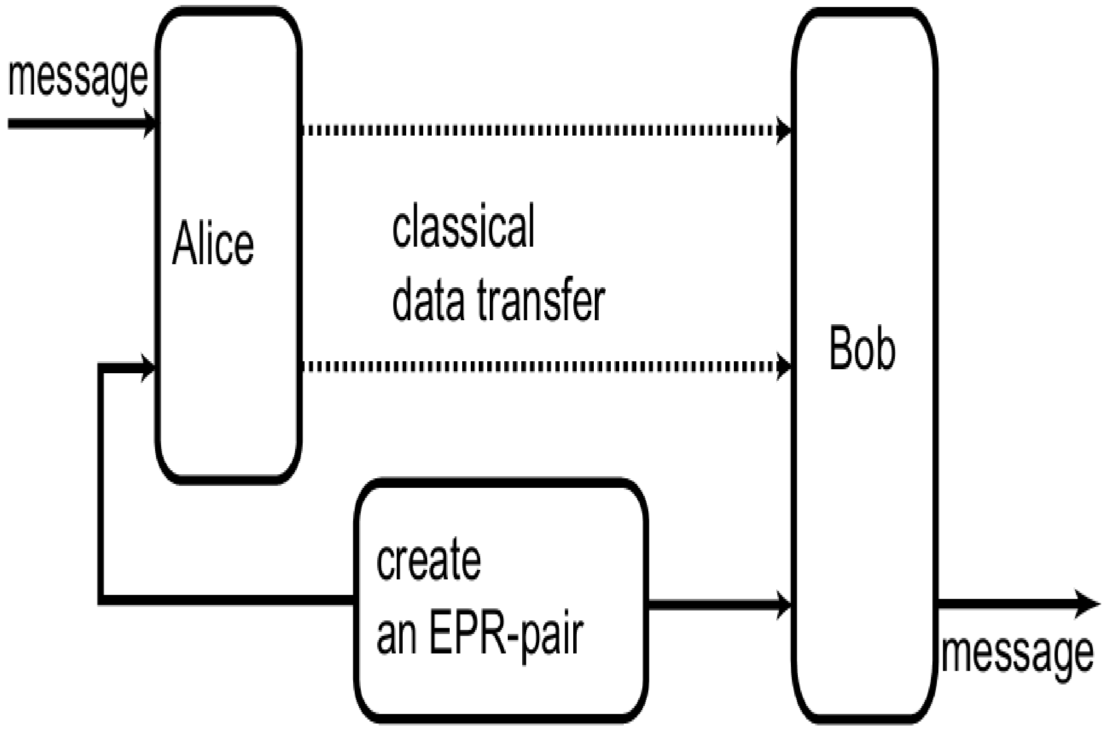


Figure 7.1: A graphical description on the Quantum teleportation process [61].

that she wishes to send to Bob. She then performs measurements on the two qubits she has access to and sends Bob the results of these measurements. By using this information, Bob can apply operators to his qubit that makes it replicate the exact state of the qubit Alice wanted to send him. This process is displayed graphically in Figure 7.1. During this process, Alice's data-encoding qubit is disturbed, and is no longer in its original state. It thus does not contravene the no-cloning theorem [5].

Before evolutionary algorithms can be applied to this problem, the goal transformation has to be described. This is done in the next section.

7.2 The transformation for quantum teleportation

There are many transformations that could lead to teleportation, as we are only interested in what happens to the third qubit, specifically the fact that the last qubit of the output must take on the value of the first qubit in the input register. The transformation (T_g) needed would have to solve the equation :

$$T_g[|a\rangle \otimes |b\rangle \otimes |c\rangle] = |f(a,b,c)\rangle \otimes |a\rangle \,.$$

Thus given that $|a\rangle = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$, $|b\rangle = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$ and $|c\rangle = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$, the transformation equation can be written as:

$$\begin{bmatrix} T_{11} & T_{12} & T_{13} & T_{14} & T_{15} & T_{16} & T_{17} & T_{18} \\ T_{21} & T_{22} & T_{23} & T_{24} & T_{25} & T_{26} & T_{27} & T_{28} \\ T_{31} & T_{32} & T_{33} & T_{34} & T_{35} & T_{36} & T_{37} & T_{38} \\ T_{41} & T_{42} & T_{43} & T_{44} & T_{45} & T_{46} & T_{47} & T_{48} \\ T_{51} & T_{52} & T_{53} & T_{54} & T_{55} & T_{56} & T_{57} & T_{58} \\ T_{61} & T_{62} & T_{63} & T_{64} & T_{65} & T_{66} & T_{67} & T_{68} \\ T_{71} & T_{72} & T_{73} & T_{74} & T_{75} & T_{76} & T_{77} & T_{78} \\ T_{81} & T_{82} & T_{83} & T_{84} & T_{85} & T_{86} & T_{87} & T_{88} \end{bmatrix} \begin{bmatrix} a_1b_1c_1 \\ a_1b_1c_2 \\ a_1b_2c_1 \\ a_1b_2c_2 \\ a_2b_1c_1 \\ a_2b_1c_2 \\ a_2b_2c_1 \\ a_2b_2c_2 \end{bmatrix} = \begin{bmatrix} f_1(b,c)a_1 \\ f_1(b,c)a_2 \\ f_2(b,c)a_1 \\ f_2(b,c)a_2 \\ f_3(b,c)a_1 \\ f_3(b,c)a_2 \\ f_4(b,c)a_1 \\ f_4(b,c)a_2 \end{bmatrix} . \quad (7.1)$$

This can be broken up into eight equations, two for each of the functions f_i. For example

$$\begin{aligned} & T_{11}a_1b_1c_1 + T_{12}a_1b_1c_2 + T_{13}a_1b_2c_1 + T_{14}a_1b_2c_2 \\ & +T_{15}a_2b_1c_1 + T_{16}a_2b_1c_2 + T_{17}a_2b_2c_1 + T_{18}a_2b_2c_2 = f_1(b,c)a_1. \end{aligned} \tag{7.2}$$

These equations must be satisfied for all possible values of $\{a_1, a_2, b_1, b_2, c_1, c_2\}$, and thus also for the following cases:

$$\{a_1, a_2, b_1, b_2, c_1, c_2\} = \begin{cases} S_1 = \{1,0,1,0,1,0\} \\ S_2 = \{1,0,1,0,0,1\} \\ S_3 = \{1,0,0,1,1,0\} \\ S_4 = \{1,0,0,1,0,1\} \\ S_5 = \{0,1,1,0,1,0\} \\ S_6 = \{0,1,1,0,0,1\} \\ S_7 = \{0,1,0,1,1,0\} \\ S_8 = \{0,1,0,1,0,1\} \end{cases}$$

We notice that $f_1(S_n) = f_1(S_{n+4})$, because the values for b and c are the same in theses cases. Each of these select out one of the T's such that

$$T_g = \begin{bmatrix} f_1(S_1) & f_1(S_2) & f_1(S_3) & f_1(S_4) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & f_1(S_1) & f_1(S_2) & f_1(S_3) & f_1(S_4) \\ f_2(S_1) & f_2(S_2) & f_2(S_3) & f_2(S_4) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & f_2(S_1) & f_2(S_2) & f_2(S_3) & f_2(S_4) \\ f_3(S_1) & f_3(S_2) & f_3(S_3) & f_3(S_4) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & f_3(S_1) & f_3(S_2) & f_3(S_3) & f_3(S_4) \\ f_4(S_1) & f_4(S_2) & f_4(S_3) & f_4(S_4) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & f_4(S_1) & f_4(S_2) & f_4(S_3) & f_4(S_4) \end{bmatrix}. \tag{7.3}$$

The transformation that is needed for teleportation is thus a unitary matrix, with definite zero entries as shown above. The function values are fixed with the corresponding entries having to be equal. Furthermore the matrix has to be unitary. As the quantum circuit always guarantees that the transformation will be unitary, the fitness will have to be a function of how close the transformations entries in the zero positions are to zero, and how close the similar functions

Maximum Population Size	200	Register Size	3 qubits
Number of Genes	10	Gene length	1 Gate
Fitness Goal	1	Maximum Generation	500
Elitism Percentage	10 %	Mutation Rate	25%
Tournament size	10% of Max pop	Number of Individuals in stasis	10
Generations to reintroduction	auto	Stagnation Factor	0.5

Table 7.1: Parameters for the Quantum teleportation experiment

Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Gene 6
$CNOT_{21}$	I	I	I	$CNOT_{21}$	I
$CNOT_{21}$	$CNOT_{12}$	$CNOT_{21}$	$CNOT_{12}$	$CNOT_{21}$	$CNOT_{21}$
I	$CNOT_{12}$	$CNOT_{21}$	$CNOT_{12}$		$CNOT_{21}$

Table 7.2: Fittest quantum circuit that satisfies the conditions with the new fitness definition.

values are too each other. If V_1 is the vector formed by the elements in the zero positions, and V_2 the vector formed by the difference between the function values that have to be equal, then the fitness (F) can be described by:

$$F = -\frac{\sqrt{V^{\dagger}V}}{48}.$$

Where V is the concatenation of V_1 and $\frac{V_2}{2}$, and F is the negative of the normal euclidian norm of the vector scaled so it has a minimum of -1.

To test this fitness a single run with the solver was done using the parameters in Table 7.1. This run produced a valid circuit for the transformation, which is shown in Table 7.2.

The transformation shown in Table 7.3 does indeed provide a final state with the state of

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Table 7.3: Transformation matrix for Table 7.2

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 8 & 8 & 8 & 8 & 8 \end{bmatrix}$$

Mask with no information transfer from Alice to Bob after qubit 3 exchange.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix}$$

Mask allowing information exchange between Alice and Bob even after qubit 3 exchange.

Table 7.4: Masks for regulating interaction with spatially separated qubits.

qubit 3 equal to the state of qubit 1 in the input register, but this is far from a teleportation algorithm, as there is no limit on the interaction between the qubits. The next challenge comes from the spacial separation of Alice and Bob, which is discussed in the next section.

7.3 Discussion on qubit interactions

For teleportation Bob never has access to qubit 1 directly and after qubit 3 has been passed to Bob, Alice doesn't have access to it for the rest of the operation. As shown in Figure 7.1 the creation of the EPR-pair, qubit 2 and qubit 3 takes place before Alice has access to the message qubit 1. This effect is incorporated in the circuit model by means of a mask that is placed over the circuit. This mask would mark off-limit interactions, by adding a taboo tax to each circuit that has interaction in these positions, and thus decreasing the fitness of such circuits to the point where they cannot compete with circuits that don't have these disallowed interactions. This is realized by introducing a net of the same dimension as the quantum circuits' matrix representation.

This net is applied to the circuit and when the gate value exceeds the net value for a specific position, 2 is subtracted from the fitness. A single illegal gate would thus reduce the fitness to below a fitness of -1, which is the lower limit for fitness values calculated by the normal metric, and thus remove such circuits from competing in the population.

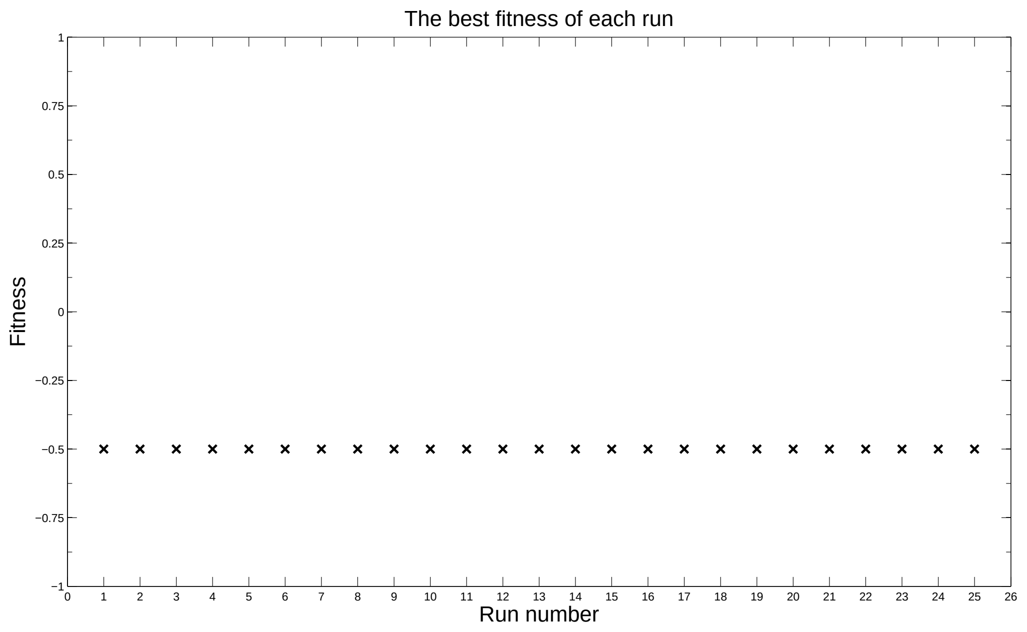


Figure 7.2: Fitness results of 25 ensemble run using mask 1.

Two examples of such masks are shown in Table 7.4. The first mask corresponds to true teleportation as Bob has no interaction with qubits 1 and 2, while Alice has no access to qubit 3. A number 1 in the mask allows only identity gates in that position, 8 allows all possible single qubit gates, and 99 allows any gate in that position.

In the first mask the 1s corresponds to the period of creating the EPR-pair thus only qubit 2 and qubit 3 can be operated on for the first five gate steps. After this step qubit 3 is no longer available for interaction so Bob can only operate any single qubit gates on it and Alice can operate any gate on only qubit 1 and qubit 2.

An ensemble of 25 runs using the parameters of Table 7.1, with this mask applied as well, was done. The results are shown in Figure 7.2, and as can be seen from the graph there is no solution found in any of the runs. There isn't any distribution in the best fitness values either. This behavior, does suggest that there is no solution available to this problem (as would be expected), but it doesn't prove that no solution exists.

The second mask corresponds to the case where there can be interaction between Bob and all the qubits in the register. This would be the case where Alice performs a measurement on

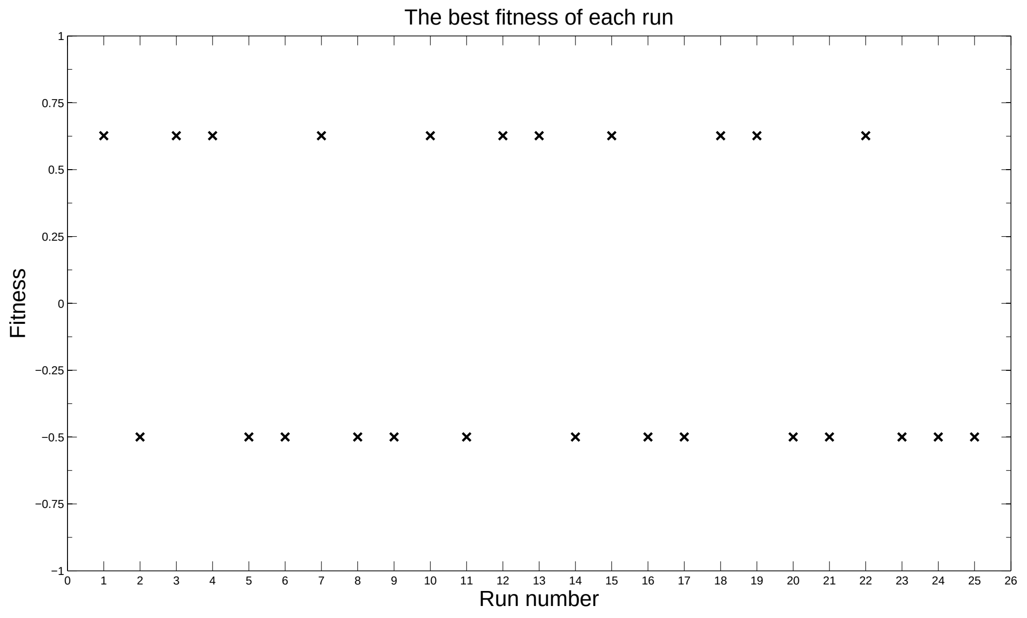


Figure 7.3: Fitness results of 25 ensemble run using mask 2.

Maximum Population Size	200	Register Size	3 qubits
Number of Genes	32	Gene length	1 Gate
Fitness Goal	1	Maximum Generation	500
Elitism Percentage	10 %	Mutation Rate	10%
Tournament size	10% of Max pop	Number of Individuals in stasis	10
Generations to reintroduction	auto	Stagnation Factor	0.01

Table 7.5: Parameters for the Quantum teleportation experiment using the Yabuki [61] and Brassard [11] approach

qubit 1 and qubit 2 and then transmits the results of the measurement to Bob. The results summarized in Figure 7.3 show that in this case there are solutions to the problem. These solutions however turn out to be the same circuit found in Table 7.2, which, although it is a valid solution to the transformation, is not a valid solution to the teleportation problem.

7.3.1 Yabuki [61] and Brassard [11] approach

To simulate quantum teleportation it is necessary to break the process up into two parts. This is what was attempted with the inclusion of the mask, but did not produce the desired result. It is clear that another approach is needed. This approach will have to include a measurement

$$M_{|00\rangle} = \begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{bmatrix} \qquad M_{|01\rangle} = \begin{bmatrix} 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{bmatrix}$$

$$M_{|10\rangle} = \begin{bmatrix} 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{bmatrix} \qquad M_{|11\rangle} = \begin{bmatrix} 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1 \end{bmatrix}$$

Table 7.6: Possible projection operators

step for which the original program was not designed. As the measurements involved in the algorithm give a logical place to break the circuit, this requires that the program generate a unitary transformation before the measurement as well as one after the measurement and then progressively apply them in turn.

The transformations that Alice and Bob applies to the register is T_A and T_B respectively. After Alice applies T_A, she performs a measurement on qubit 1 and qubit 2 of the register and sends the results of this measurement on to Bob. Bob then applies T_B to the register made of qubit 3 and two ancillary qubits created using the classical information sent to him by Alice. For example if Alice measured $|\uparrow\rangle$, $|\downarrow\rangle$, then Bob would use ancillary states $|\uparrow\rangle$, $|\downarrow\rangle$. T_A is such that it allows interaction between qubit 2 and qubit 3 for entangling purposes and then between qubit 1 and qubit 2, but never all three at the same time as Alice never has access to qubit 1 and qubit 3 at the same time. As a further simplification that is in line with the work done by both Yabuki [61] and Brassard [11], qubits 2 and 3 are only allowed to start in the $|0\rangle$ state.

We start again at $T_g[|a\rangle \otimes |b\rangle \otimes |c\rangle] = |f(a,b,c)\rangle \otimes |a\rangle$, but in this case $T_g = T_B M_n T_A$, where M_n is the projection operator simulating the measurement of the first 2 qubits. Depending on the outcome of measurement there are four possible projection operators. These are shown

1	1	1	1	1	1	1	1	99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99	99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99	1	1	1	1	1	1	1	1

100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100

Table 7.7: Mask for Yabuki [61] and Brassard [11] approach run

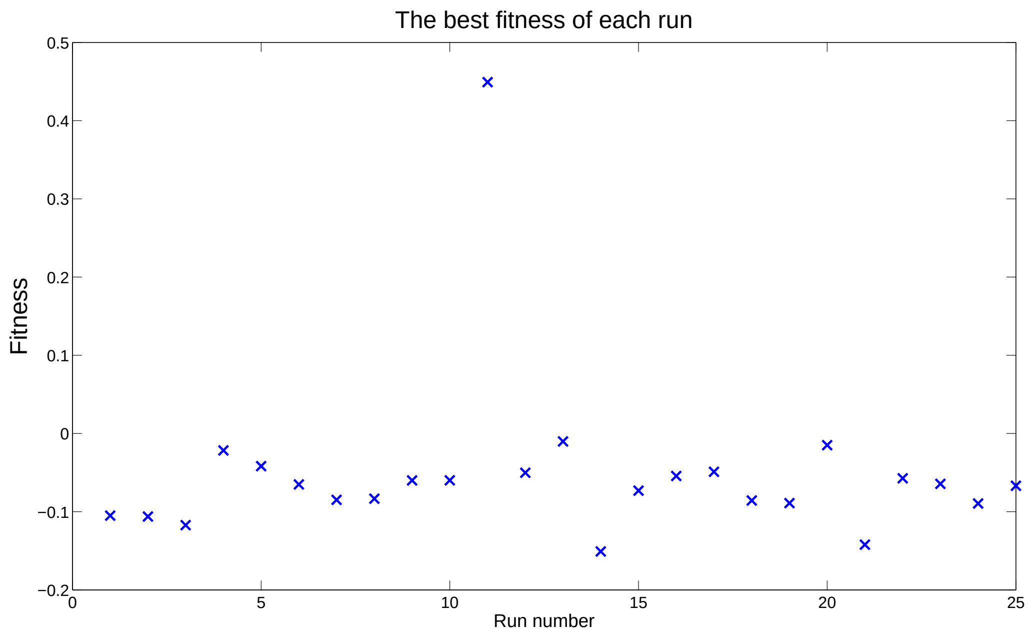


Figure 7.4: Fitness results of 25 ensemble run for Yabuki [61] and Brassard [11] approach.

in Table 7.6. Every circuit would now have four transformations depending on the results of the measurement. Using a density matrix representation allows us to combine all the possible outcomes of measurement into a single representation. The gate set used by Yabuki [61] and Brassard [11] also differs from the standard one discussed in Section 4.2, and was replaced by the one shown in Table 7.8. This change would be needed to compare the circuits given in Yabuki [61], with the ones produced by the program.

The fitness of the circuit would now be assessed using the reduced density matrix of the third qubit in the output register, and comparing it to the density matrix of the input qubit 1. This is done using the normal Frobenius norm, see Section 7.2. For each circuit five random

$$I(1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad S(2) = \begin{bmatrix} i & 0 \\ 0 & 1 \end{bmatrix} \qquad T(3) = \begin{bmatrix} -1 & 0 \\ 0 & -i \end{bmatrix}$$

$$L(4) = \tfrac{1}{\sqrt{2}}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \qquad R(5) = \tfrac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \qquad H(6) = \tfrac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$P(7) = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix} \qquad Q(8) = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{8}} \end{bmatrix} \qquad S^{\dagger}(9) = \begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix}$$

$$P^{\dagger}(10) = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{-i\pi}{4}} \end{bmatrix} \qquad Q^{\dagger}(11) = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{-i\pi}{8}} \end{bmatrix}$$

$$CNOT_{12}(12) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad CNOT_{21}(13) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Table 7.8: Table of primitive gate set for Yabuki [61] and Brassard [11] approach

states for qubit 1 is generated, and then compared with the qubit 3 states after application of all the possible measurements. The circuit fitness is given as the average fitness of all these transformations. To differentiate circuits that work for all states from ones that work only for certain choices, two is subtracted from the fitness when there is no working transformation, and one when at least one of the tested input qubits was successfully teleported.

Using the parameter set given in Table 7.5, and the mask described in Table 7.7, an ensemble of 25 runs were done with the results shown in Figure 7.4. For all the runs, circuits that were very close to teleportation was produced, except for run 11, where a true teleportation circuit was evolved. This circuit, together with the best from literature is shown in Table 7.9, Table 7.10 and Table 7.11.

The circuits from Yabuki [61] and Brassard [11] both contain gates that act on non-adjacent qubits [5]. To make this compatible with the program, swap gates in the form of three $CNOT$

[5]Other examples of circuits that give the teleportation result is given on p.272 of Marinescu and Marinescu [38], p. 201 of Williams and Clearwater [60].

									Gene									
1	2	3	4	5	6	7	8	9	10	11	12		13	14	15	16	17	18
1	1	1	1	1	1	12	13	9	1	1	5	M	1	1	10	13	1	1
1	1	6	2	9	12	12	13	1	6	5	1	M	12	6	1	13	13	1
5	4	1	1	1	12	1	1	1	1	1	1		12	1	1	1	13	9

Table 7.9: Fittest quantum circuit evolved using the Yabuki [61] and Brassard [11] approach.

						Gene					
1	2	3	4		5	6	7	8	9	10	11
1	1	12	4	M	5	1	1	13	1	1	1
4	12	12	1	M	1	12	13	13	13	12	12
1	12	1	1		1	12	13	1	13	12	12

Table 7.10: Best quantum teleportation circuit reported by Yabuki [61].

gates have to be included before and after each of these non-adjacent interacting gates, and thus the structure differ from what was given in Yabuki [61]. Subsequently each of these circuits was fed into the program as initial solutions and further evolved. The effect of this evolution was to reduce the three gate $CNOT$ structure to a two gate effective equivalent. This is reflected in genes $6-7$ and $9-10$ in Table 7.10, and genes $6-7, 9-10, 12-13$ and $15-16$ in Table 7.11. As these are long circuits we will forgo the writing out of gate names and just give the encoding number for each gate. The symbol for a measurement is M.

The genetic code of the circuits shows that Yabuki [61]'s circuit is shorter (11 gates) and simpler than that given by Brassard [11] (17 gates). Both of these are however still shorter and less complex than the one that the solver program evolved (18 gates). Note that the diagram for Brassard's circuit is incorrect in Yabuki [61]. The second $CNOT$-gate should be a $CNOT_{13}$-gate, rather than the $CNOT_{12}$-gate that was reported.

								Gene								
1	2	3	4		5	6	7	8	9	10	11	12	13	14	15	16
1	1	12	5	M	2	1	1	13	1	1	2	1	1	13	1	1
4	12	12	1	M	12	12	13	13	13	12	1	12	13	13	13	12
1	12	1	1		12	12	13	1	13	12	3	12	13	1	13	12

Table 7.11: Best quantum teleportation circuit reported by Brassard [11]

7.4 Conclusion

Even though the results from the solver program isn't as good as had been hoped, the principles of its operation have been shown to be sound, even with the modification that was needed to accommodate both measurement as well as the non-interaction constraints. With more time and effort in code streamlining, or changing the coding language to a faster alternative like C++, the speed of execution and thus the number of experiments in an ensemble could be increased and thus improve the effectiveness of the solver program.

CHAPTER 8

Summary and future work

8.1 Project analysis in terms of set goals

The first goal for the project was to write a program in Matlab so apply evolutionary algorithms to the Travelling Salesman Problem. This goal was attained and is documented in in Chapter 3. Apart from just evolving a solution to a chosen problem from this class, namely the TSP of the 48 state capitals of continental USA, the influence of the various selection methods, and the influence of elitism were tested. The most effective and efficient selection methods were identified, and the impact of elitism on all of them was also investigated. Tournament selection and Normal selection proved the most effective, although the difference between the best seven methods were very slight. Elitism was shown to be advantageous to accuracy, but not universally so for efficiency of the algorithm.

The next step was the programming of a similar solver program for the quantum circuit model. This was done and successfully tested on evolving the identity transformation for a circuit. This also afforded the opportunity to test the impact of some of method parameters on the effectiveness of the program. Firstly we looked at the best selection method for this task, which again was tournament selection. Secondly we looked at the population size of a single generation and how that influences the solution time. A population size of 200 individuals gave the fastest convergence for the algorithm. The influence of island models was also investigated in the form of a stasis effect for the best individuals in a mature generation. This method was highly effective, and was used in all subsequent runs.

Thirdly, after the correct functioning of the software had been established we proceeded to apply the program to four circuits listed below, and to which evolutionary algorithms where applied in the article by Ding et al. [16]:

- A circuit for maximally entangling two qubits

- A controlled S-gate
- A circuit for maximally entangling three qubits
- A quantum Fourier transform gate for three qubits

For some of these circuits the definition of the gate set had to be adjusted, but after doing so all the effective circuits where evolved, with the exception of the quantum Fourier transform for three qubits. The speed of the solver was faster, with the exception of QFT, when compared with the convergence rates reported in Ding et al. [16]. The QFT circuit showed that the solver had problems evolving circuits that needed swap circuits. For evolutionary methods to work, the influence gates have on the circuit have to be comparable. The results from these runs point to the fact that swap circuits for qubits have a much greater influence than other gates.

The last part was an investigation into optimization of the the circuit for quantum teleportation. For comparison the work done by Yabuki [61] is used. The challenges here were to incorporate a measurement process in the solver program, and to designate parts of the circuit that were non-interacting, representing the physical proximity of qubits to the processing party. Firstly the solver was run on a circuit that had no measurement, but where the qubits were spatially separated. This run did not meet with any success. This was expected as a positive result here would have meant that information could be transferred faster that the speed of light.

After incorporating measurement into the model, solutions to the problem were found. The optimal solution found had 18 gates, in comparison to the original solution by Brassard [11] with 17 gates, and the solution found by Yabuki [61] at 11 gates.

8.2 Future work

Although the solver program functioned effectively, it is clear that Matlab, however simple to program, is not the optimal choice for large simulations. Any further research should be started by rewriting the code in a faster language, preferably C++. Having an environment where the individuals are treated as objects with properties, would be a much more natural way of executing such and algorithm.

Once such a program has been realized, I envision a situation where the parameters of the algorithm will not be hard coded, but rather will also be properties of the individuals, and that, with proper fitness and competition rules, will allow the makeup of a specific generation to not only give information about the problem you are trying to solve, but also about which parameters will solve such a problem the best. This would also allow an analysis of how diverse the population is, how many individuals are siblings, and how this influences both take over and premature stagnation.

It would also be interesting to investigate how the shape of the fitness space influences the convergence. I would propose that the optimization is done in a large dimensional space with a single peak value, and a random noise parameter that would introduce many new peaks, or local optima, of values ranging from zero to the the value very close to the true optimal value. Another parameter would be the number of such noise peaks distributed throughout the fitness space.

On the quantum circuit side, comparing solutions to known problems using different gate set would be interesting, also, finding a relation between the size of the gate set and the convergence tempo. Using gate sets that are universal but not primitive, gives the simulator more tools to solve the problems, but it also increases the size of the search space. The influence of different gate sets are apparent in the differences between the results in Section 6.5.

Further research into the strange effect that the swap gates had on the convergence during the investigation of the QFT in Chapter 6 is certainly a worthwhile endeavor. So would be encoding the best known solution to the problems and letting them compete with the best solution that were generated during this thesis, maybe a crossing between these solutions could lead to even better solutions.

And finally, it is known that there is no bit-commitment protocol for a fair quantum coin toss [36]. Investigating alternate methods of achieving the same ends using evolutionary algorithms would be interesting.

BIBLIOGRAPHY

[1] E Alba and M Tomassini. Parallelism and evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 6(5):443–462, October 2002. ISSN 1089-778X. doi: 10.1109/TEVC.2002.800880.

[2] David Applegate, Robert Bixby, Vasek Chvatal, and William Cook. On the Solution of Traveling Salesman Problems. *Documenta Mathematica*, Extra volu:645–656, 1998.

[3] James E Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 14–21, Hillsdale, NJ, USA, 1987. L. Erlbaum Associates Inc. ISBN 0-8058-0158-8.

[4] G Balasubramanian, P Neumann, D Twitchen, M Markham, R Kolesov, N Mizuochi, J Isoya, J Achard, J Beck, J Tissler, and Others. Ultralong spin coherence time in isotopically engineered diamond. *Nature Materials*, 8(5):383–387, 2009.

[5] Michel Le Bellac. *Quantum information and quantum computation*. Cambridge university press, 2006.

[6] M Bellmore and G L Nemhauser. The Traveling Salesman Problem: A Survey. *Operations Research*, 16(3):pp. 538–558, 1968. ISSN 0030364X. URL `http://www.jstor.org/stable/168581`.

[7] C H Bennett, G Brassard, C Crpeau, R Jozsa, A Peres, and W K Wootters. Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. *Physical Review Letters*, 70(13):1895–1899, 1993. URL `www.scopus.com`.

[8] André Berthiaume and Gilles Brassard. Oracle Quantum Computing. *Journal of Modern Optics*, 41(12):2521–2535, December 1994. ISSN 0950-0340. doi: 10.1080/09500349414552351. URL `http://www.tandfonline.com/doi/abs/10.1080/09500349414552351`.

[9] Rainer Blatt and David Wineland. Entangled states of trapped atomic ions. *Nature*, 453(7198):1008–15, June 2008. ISSN 1476-4687. doi: 10.1038/nature07125. URL http://www.ncbi.nlm.nih.gov/pubmed/18563151.

[10] Kenneth D Boese. Cost Versus Distance In the Traveling Salesman Problem. Technical report, UCLA Computer Science Department, 1995.

[11] Gilles Brassard. Teleportation as a quantum computation. *Physica*, D120:43–47, 1998.

[12] CSN. Map of USA with state capitols included., 2000. URL http://www.csgnetwork.com/.

[13] D Deutsch. Quantum theory, the church-turing principle and the universal quantum computer. *Proceedings of The Royal Society of London, Series A: Mathematical and Physical Sciences*, 400(1818):97–117, 1985. ISSN 00804630. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-0022421379&partnerID=40&md5=d5c0a3e85820e3091326aa0483915aa8.

[14] D. Deutsch and R. Jozsa. Rapid Solution of Problems by Quantum Computation. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 439 (1907):553–558, December 1992. ISSN 1364-5021. doi: 10.1098/rspa.1992.0167. URL http://rspa.royalsocietypublishing.org/cgi/doi/10.1098/rspa.1992.0167.

[15] L DiCarlo, J M Chow, J M Gambetta, L S Bishop, B R Johnson, D I Schuster, J Majer, A Blais, L Frunzio, S M Girvin, and Others. Demonstration of two-qubit algorithms with a superconducting quantum processor. *Nature*, 460(7252):240–244, 2009.

[16] S Ding, Z Jin, and Q Yang. Evolving quantum circuits at the gate level with a hybrid quantum-inspired evolutionary algorithm. *Soft Computing*, 12(11):1059–1072, 2008. ISSN 14327643. doi: 10.1007/s00500-007-0273-9. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-45849102047&partnerID=40&md5=3056ada1e3b0efa0ee36105dcc0c5f8d.

[17] David P DiVincenzo. The Physical Implementation of Quantum Computation. *Fortschritte der Physik*, 48(9-11):771–783, 2000. ISSN 1521-3978. doi: 10.1002/1521-3978(200009)48:9/11⟨771::AID-PROP771⟩3.0.CO;2-E. URL http://dx.doi.org/10.1002/1521-3978(200009)48:9/11<771::AID-PROP771>3.0.CO;2-E.

[18] Marco Dorigo. Optimization, Learning and Natural Algorithms. *Ph.D. Thesis, Politecnico di Milano, Italy*, 1992. URL http://ci.nii.ac.jp/naid/10016599043/en/.

[19] P. Festa and M. G. C. Resende. GRASP: basic components and enhancements. *Telecommunication Systems*, 46(3):253–271, March 2010. ISSN 1018-4864. doi: 10.1007/s11235-010-9289-z. URL http://www.springerlink.com/index/10.1007/s11235-010-9289-z.

[20] R P Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, 1982. ISSN 00207748. doi: 10.1007/BF02650179. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-51249180339&partnerID=40&md5=2cffe879999584b6df6cd9c60ecbb2ac.

[21] A B Finnila, M A Gomez, C Sebenik, C Stenson, and J D Doll. Quantum annealing: a new method for minimizing multidimensional functions. *Chemical physics letters*, 219 (5-6):343–348, 1994.

[22] Douglas J Futuyma. *Evolution second edition*. Sinauer, 2009.

[23] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549, January 1986. ISSN 03050548. doi: 10.1016/0305-0548(86)90048-1. URL http://linkinghub.elsevier.com/retrieve/pii/0305054886900481.

[24] Fred Glover, J.P. Kelly, and Manuel Laguna. Genetic algorithms and tabu search: hybrids for optimization. *Computers & Operations Research*, 22(1):111–134, 1995. URL http://www.sciencedirect.com/science/article/pii/0305054893E0023M.

[25] L.K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, New York, New York, USA, 1996. ACM. ISBN 0897917855. doi: 10.1145/237814.237866. URL http://portal.acm.org/citation.cfm?doid=237814.237866http://dl.acm.org/citation.cfm?id=237866.

[26] Peter J B Hancock. An Empirical Comparison of Selection Methods in Evolutionary Algorithms, 1994.

[27] Nicholas J Higham. *Accuracy and Stability of Numerical Algorithms*, volume 11. Society for Industrial and Applied Mathematics, 2002. ISBN 0898715210. doi: 10.1177/1367493507079565. URL http://www.amazon.com/Accuracy-Stability-Numerical-Algorithms-Nicholas/dp/0898715210.

[28] Richard M Karp. Dynamic programming meets the principle of inclusion and exclusion. *Operations Research Letters*, 1(2):49–51, 1982. ISSN 0167-6377. doi: DOI:10.1016/0167-6377(82)90044-X. URL http://www.sciencedirect.com/science/article/pii/016763778290044X.

[29] J Kempe, D Bacon, D P DiVincenzo, and K B Whaley. Encoded Universality from a Single Physical Interaction. *Quantum Information and Computation, vol.1 (Special Issue) (2001) pp. 33-55*, Volume 1:33–55, 2001.

[30] S Kirkpatrick, C D Gelatt, and M P Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983. URL http://www.ncbi.nlm.nih.gov/pubmed/17813860.

[31] E. Knill. Approximation by quantum circuits. *Arxiv preprint quant-ph/9508006*, (May): 1–23, 1995. URL http://arxiv.org/abs/quant-ph/9508006.

[32] E Knill, R Laflamme, and G J Milburn. A scheme for efficient quantum computation with linear optics. *Nature*, 409(6816):46–52, January 2001. ISSN 0028-0836. doi: 10.1038/35051009. URL http://www.ncbi.nlm.nih.gov/pubmed/11343107.

[33] T D Ladd, F Jelezko, R Laflamme, Y Nakamura, C Monroe, and J L O'Brien. Quantum computers. *Nature*, 464(7285):45–53, March 2010. ISSN 1476-4687. doi: 10.1038/nature08812. URL http://www.ncbi.nlm.nih.gov/pubmed/20203602.

[34] C Langer, R Ozeri, J D Jost, J Chiaverini, B DeMarco, A Ben-Kish, R B Blakestad, J Britton, D B Hume, W M Itano, and Others. Long-lived qubit memory using atomic ions. *Physical review letters*, 95(6):60502, 2005.

[35] Tianjun Liao, Marco A Montes De Oca, and Marco Dorigo. An Incremental Ant Colony Algorithm with Local Search for Continuous Optimization. Technical Report April, IRIDIA, 2011.

[36] H Lo and H Chau. Why quantum bit commitment and ideal quantum coin tossing are impossible. *Physica D: Nonlinear Phenomena*, 120(1-2):177–187, 1998. ISSN 01672789. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-0001731260&partnerID=40&md5=9fae9d5377450250b5e572d5ee264815.

[37] Sean Luke. *Essentials of Metaheuristics. 2009*. 2010. ISBN 9780557148592. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Essentials+of+metaheuristics#0.

[38] Dan C Marinescu and Gabriela M Marinescu. *Approcaching Quantum Computing*. Pearson Prentice Hall, 2004.

[39] M Mehring, J Mende, and W Scherer. Entanglement between an electron and a nuclear spin 1/2. *Physical review letters*, 90(15):153001, 2003.

[40] C E Miller, A W Tucker, and R A Zemlin. Integer Programming Formulation of Traveling Salesman Problems. *J. ACM*, 7(4):326–329, October 1960. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/321043.321046. URL http://doi.acm.org/10.1145/321043.321046.

[41] C Monroe, Dm Meekhof, Be King, Wm Itano, and Dj Wineland. Demonstration of a fundamental quantum logic gate. *Physical review letters*, 75(25):4714–4717, December 1995. ISSN 1079-7114. URL http://www.ncbi.nlm.nih.gov/pubmed/10059979.

[42] O Morsch and M Oberthaler. Dynamics of Bose-Einstein condensates in optical lattices. *Reviews of modern physics*, 78(1):179, 2006. URL http://rmp.aps.org/abstract/RMP/v78/i1/p179_1.

[43] C Negrevergne, T S Mahesh, C A Ryan, M Ditty, F Cyr-Racine, W Power, N Boulant, T Havel, D G Cory, and R Laflamme. Benchmarking Quantum Control Methods on a 12-Qubit System. *Phys. Rev. Lett.*, 96(17):170501, May 2006. doi: 10.1103/PhysRevLett.96.170501. URL http://link.aps.org/doi/10.1103/PhysRevLett.96.170501.

[44] Nichel A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge university press, 2000.

[45] A O Niskanen, K Harrabi, F Yoshihara, Y Nakamura, S Lloyd, and J S Tsai. Quantum coherent tunable coupling of superconducting qubits. *Science*, 316(5825):723, 2007.

[46] S Olmschenk, D N Matsukevich, P Maunz, D Hayes, and C Monroe. Quantum Teleportation Between Distant Matter Qubits. *Science*, 323(5913):486–489, 2009.

[47] Martin B Plenio and Shashank Virmani. An introduction to entanglement measures. *Quant. Inf. Comput.*, 7:1–51, 2007.

[48] Alberto Politi, J.C.F. Matthews, and J.L. O'Brien. Shors quantum factoring algorithm on a photonic chip. *Science*, 325(5945):1221, 2009. URL `http://www.sciencemag.org/content/325/5945/1221.short`.

[49] C Schneider, M Strauß, T Sünner, A Huggenberger, D Wiener, S Reitzenstein, M Kamp, S Höfling, and A Forchel. Lithographic alignment to site-controlled quantum dots for device integration. *Applied Physics Letters*, 92:183101, 2008.

[50] Alexander Schrijver. On the History of Combinatorial Optimization (Till 1960). In G L Nemhauser K. Aardal and R Weismantel, editors, *Discrete Optimization*, volume 12 of *Handbooks in Operations Research and Management Science*, pages 1–68. Elsevier, 2005. doi: DOI:10.1016/S0927-0507(05)12001-5. URL `http://www.sciencedirect.com/science/article/pii/S0927050705120015`.

[51] PW Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 124–134. IEEE, 1994. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=365700`.

[52] PW Shor. Scheme for reducing decoherence in quantum computer memory. *Physical review A*, 52(4):2493–2496, 1995. URL `http://link.aps.org/doi/10.1103/PhysRevA.52.R2493`.

[53] Krzysztof Socha and Marco Dorigo. Ant colony optimization for mixed-variable optimization problems. Technical Report October, IRIDIA, 2007. URL `http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2007-019r001.pdf`.

[54] A Steane. Quantum computing. *Reports on progress in physics*, VOL 61; NU:117–173, 1998.

[55] A M Steane. Error Correcting Codes in Quantum Theory. *Phys. Rev. Lett.*, 77(5):793–797, July 1996. doi: 10.1103/PhysRevLett.77.793. URL http://link.aps.org/doi/10.1103/PhysRevLett.77.793.

[56] Thomas Stutzle and Marco Dorigo. ACO Algorithms for the Traveling Salesman Problem, 1999.

[57] Adrian Thompson. Silicon Evolution. In *Stanford University*, pages 444–452. MIT Press, 1996.

[58] University of Heidelberg. TSP lib source, 2010. URL http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

[59] L M K Vandersypen, M Steffen, G Breyta, C S Yannoni, M H Sherwood, and I L Chuang. Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature*, 414(6866):883–887, 2001.

[60] Colin P Williams and Scott H Clearwater. *Explorations in Quantum Computing*. Springer Verlag, 1998.

[61] Taro Yabuki Yabuki. Genetic Algorithms for Quantum Circuit Design –Evolving a Simpler Teleportation Circuit–. In *In Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference*, pages 421–425. Morgan Kauffman Publishers, 2000.

[62] Gexiang Zhang. Quantum-inspired evolutionary algorithms: a survey and empirical study. *Journal of Heuristics*, 17(3):303–351, 2011. ISSN 1381-1231. URL http://dx.doi.org/10.1007/s10732-010-9136-0.

Appendices

PART I

Graphs and figures

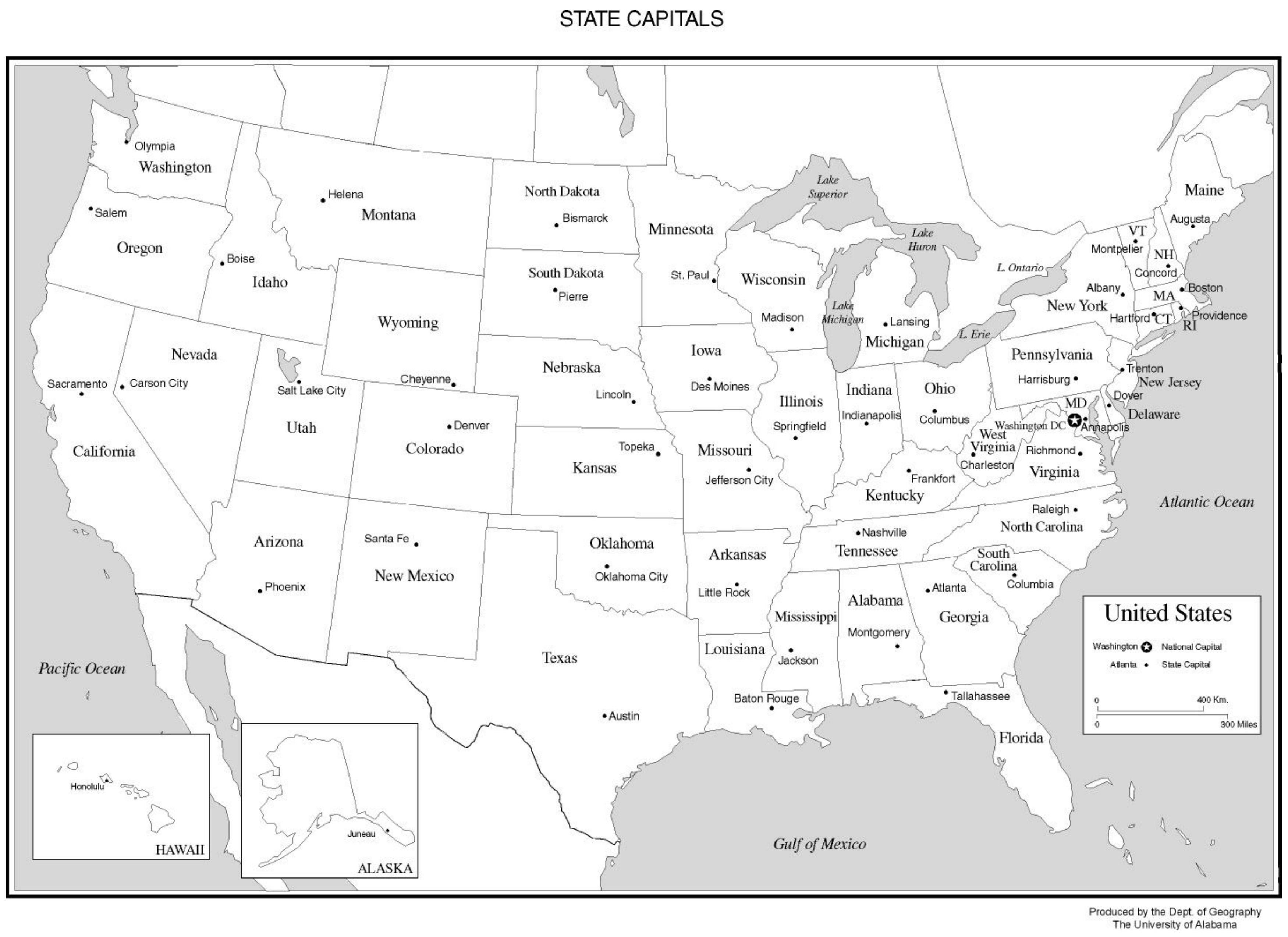


Figure 1: Map of the USA, with state capitals indicated.

CSN [12]

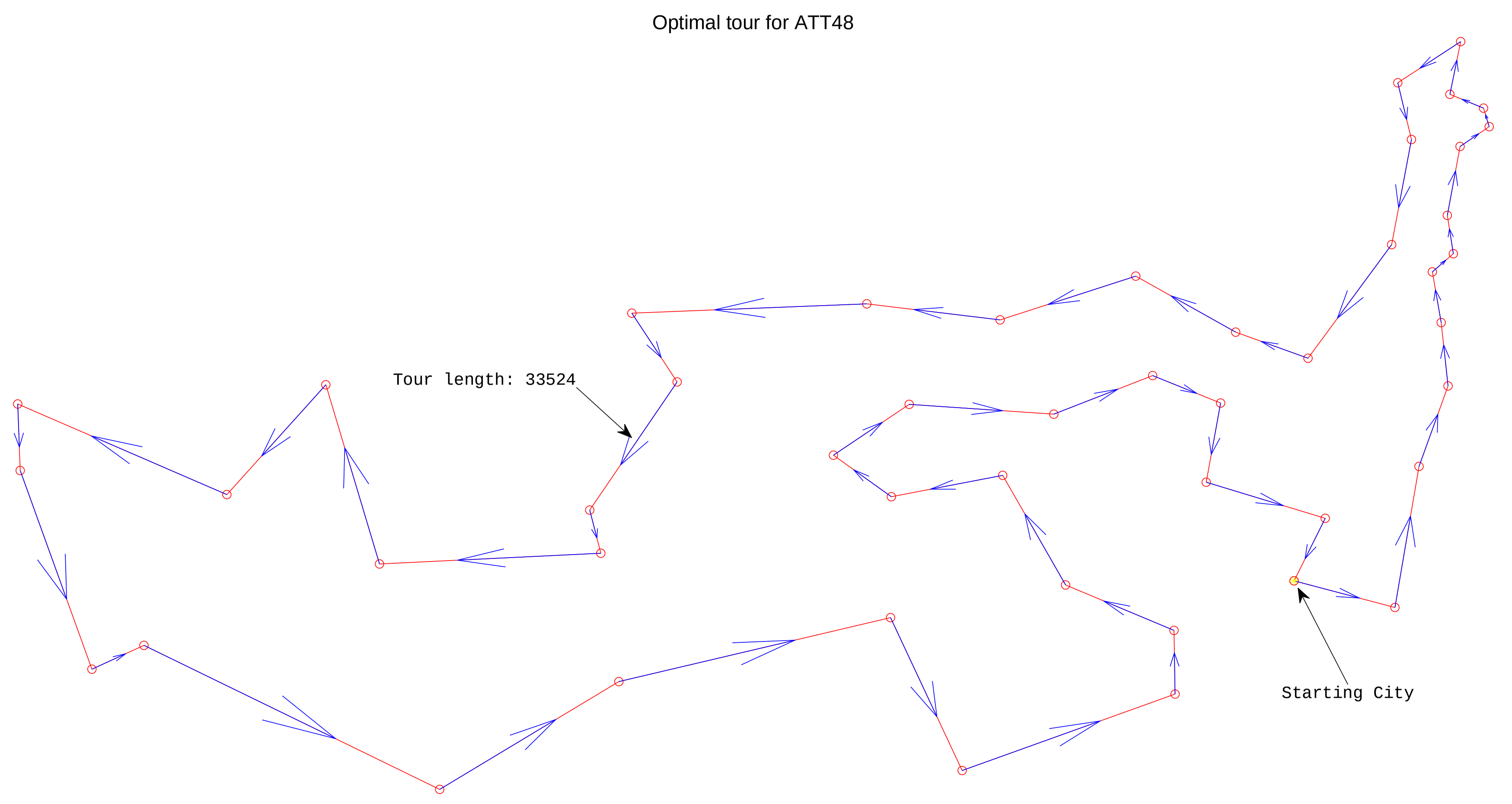


Figure 2: Graphic representation of ATT48 Cities and the optimal tour

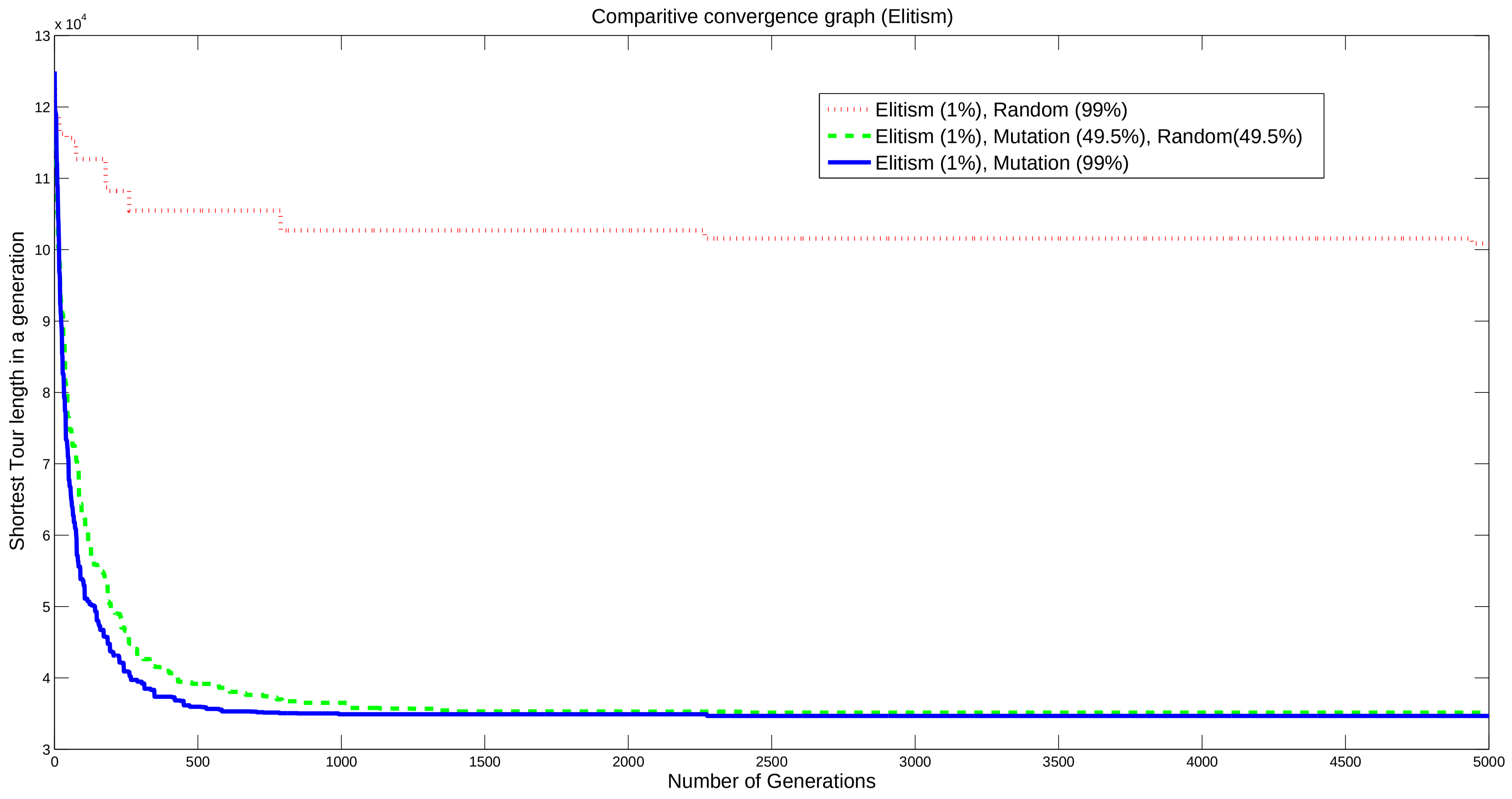


Figure 3: Comparative convergence graph of Elitism effect (Survival of the fittest) on ATT48 Simulation

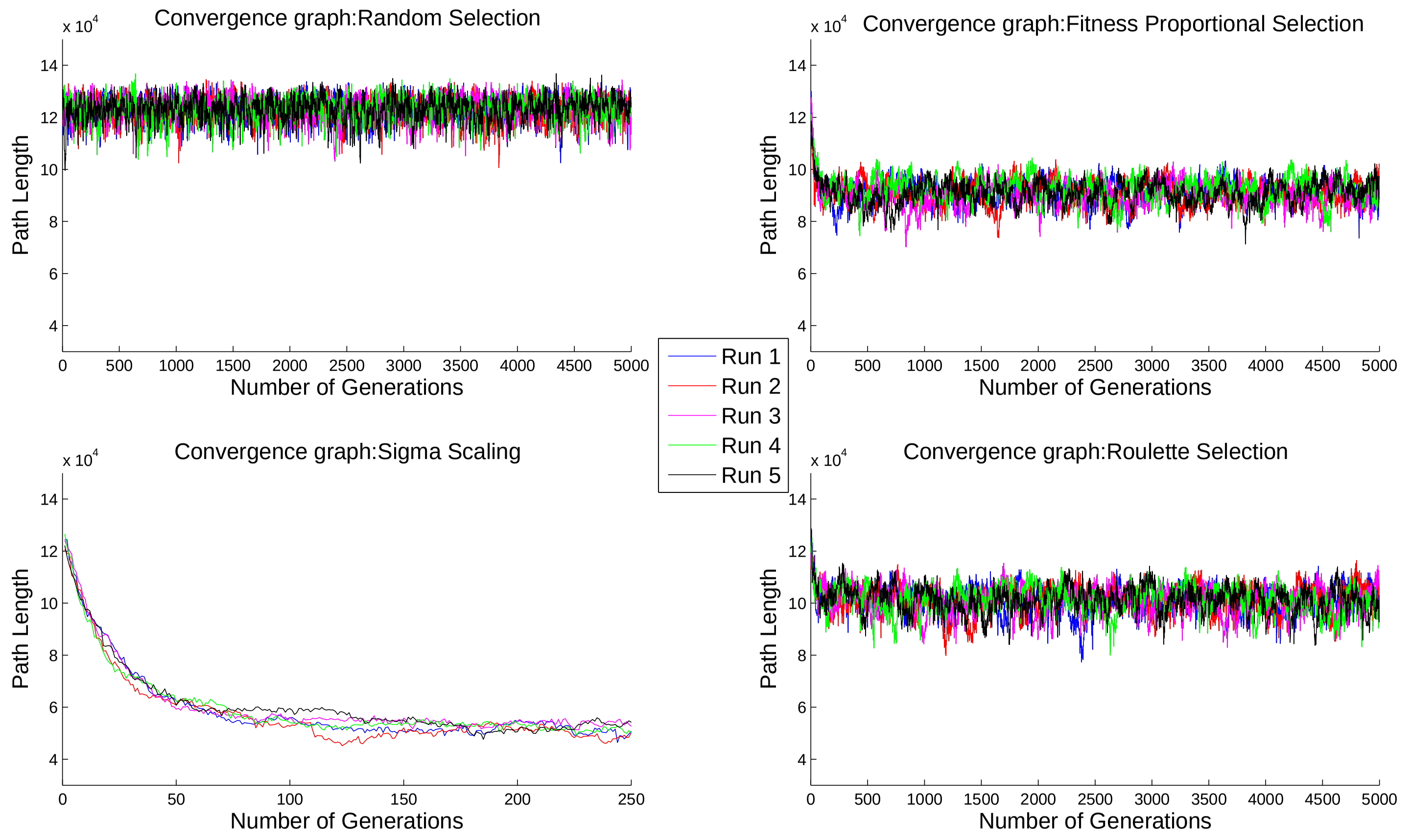


Figure 4: Comparative convergence graph of selection methods using ATT48 Simulation, Methods 0-3

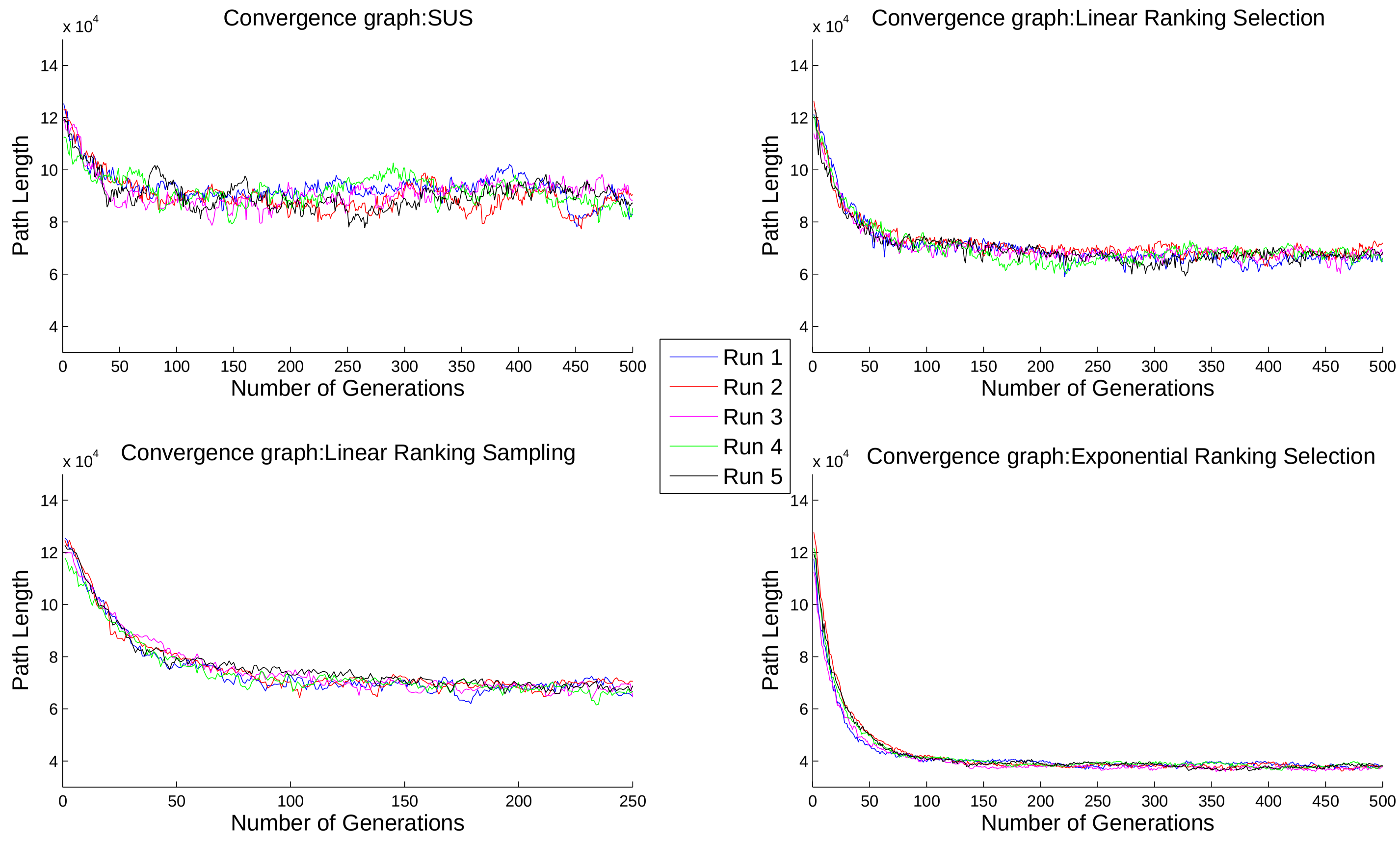


Figure 5: Comparative convergence graph of selection methods using ATT48 Simulation, Methods 4-7

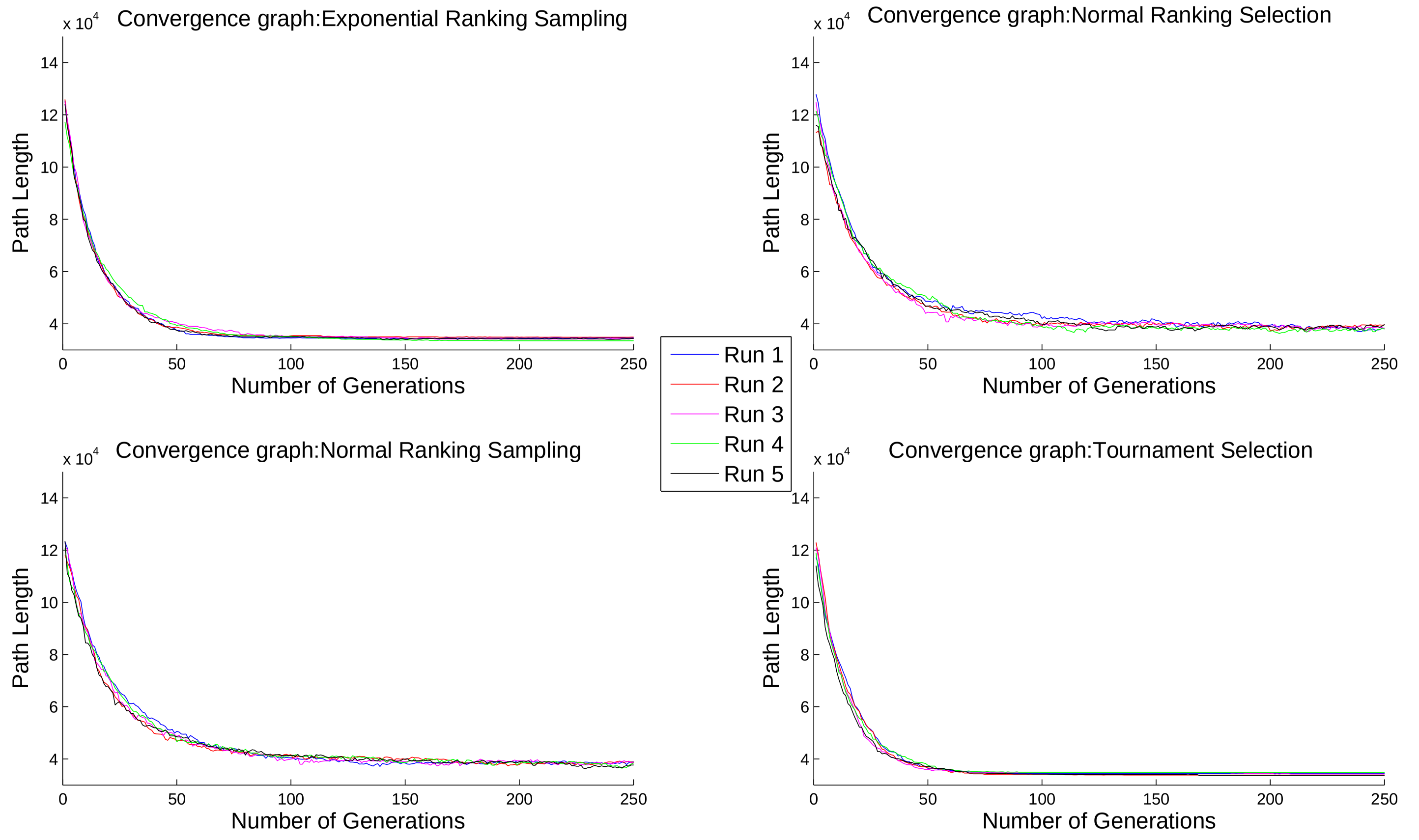


Figure 6: Comparative convergence graph of selection methods using ATT48 Simulation, Methods 8-11

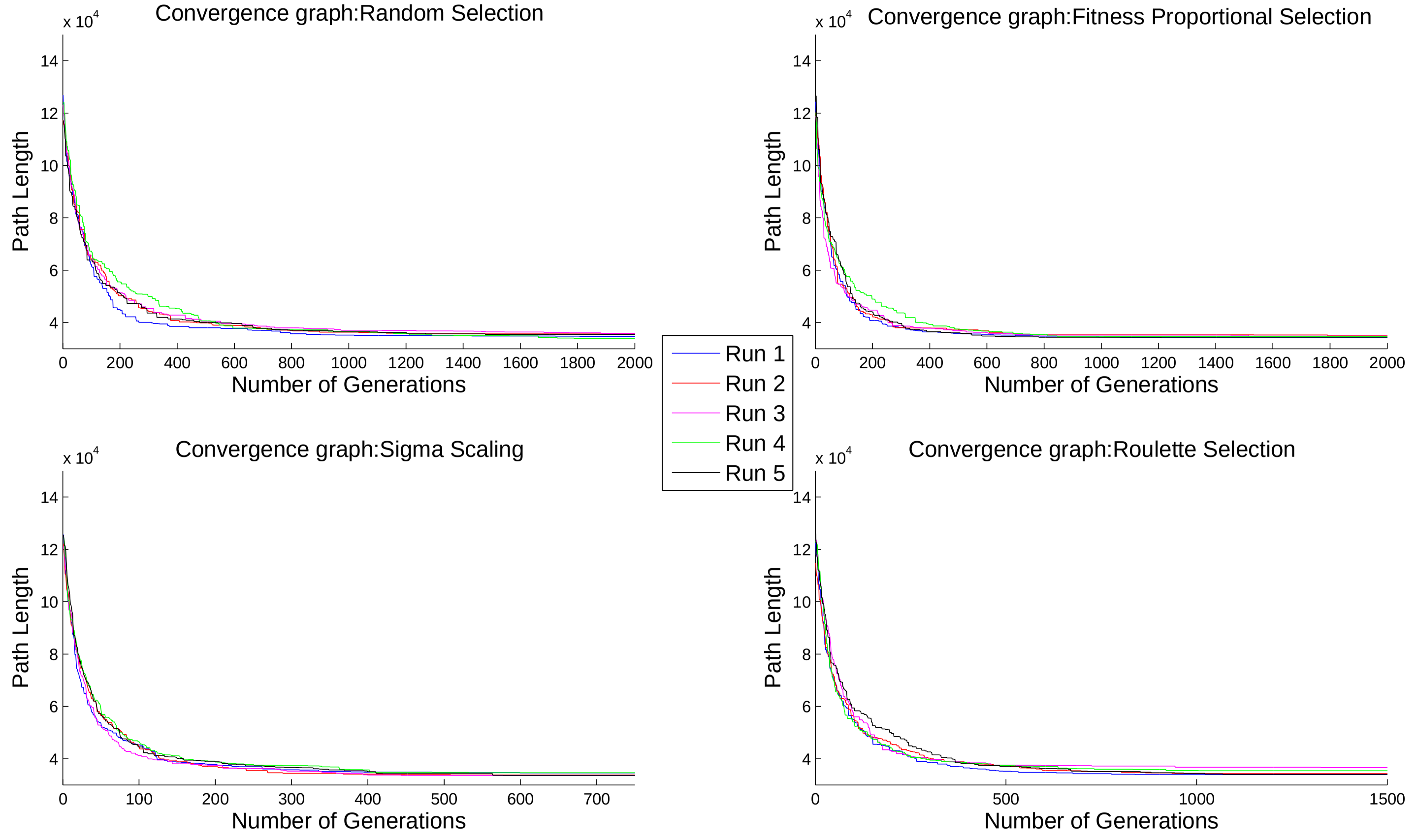


Figure 7: Comparative convergence graph of selection methods using ATT48 Simulation, Methods 0-3

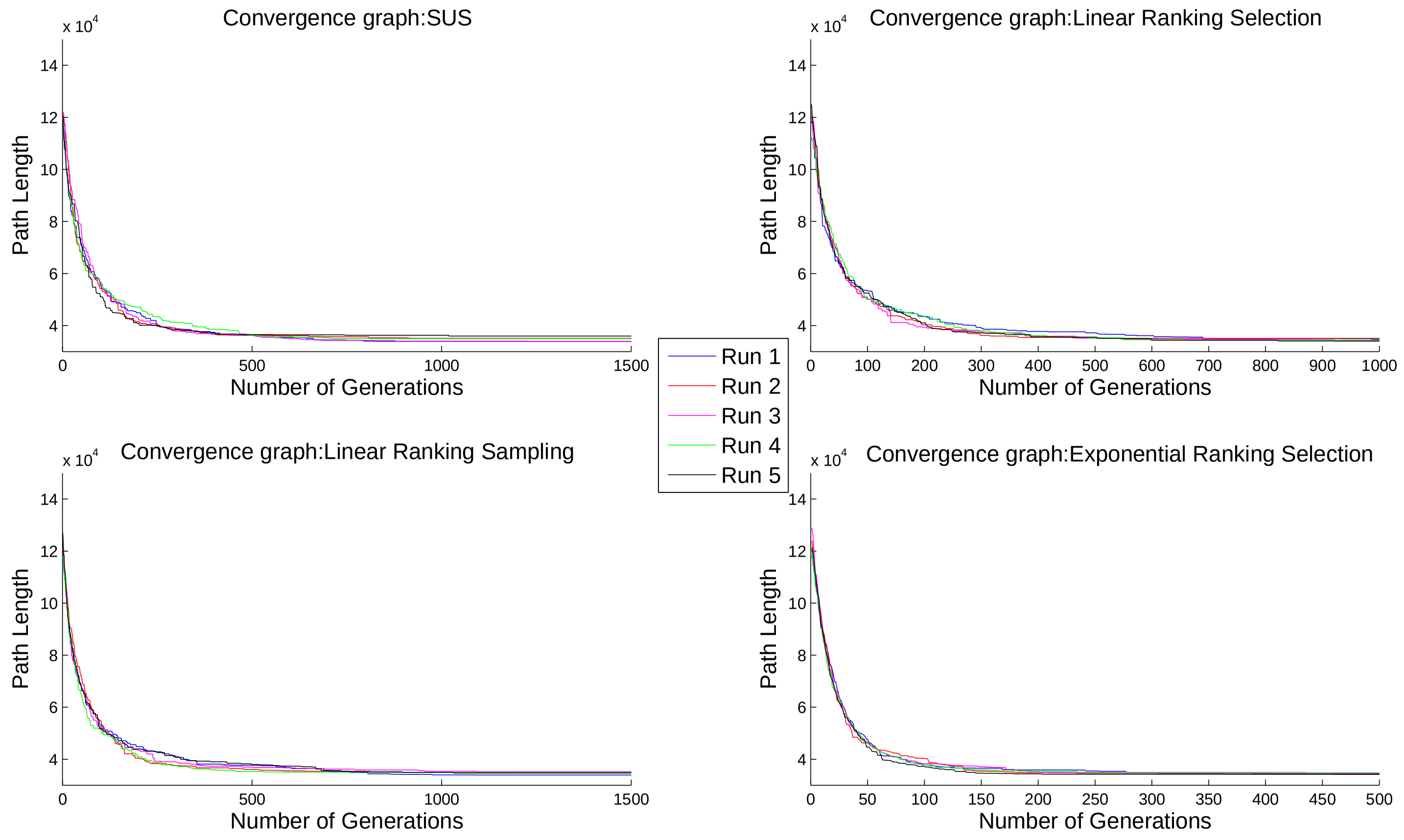


Figure 8: Comparative convergence graph of selection methods using ATT48 Simulation, Methods 4-7

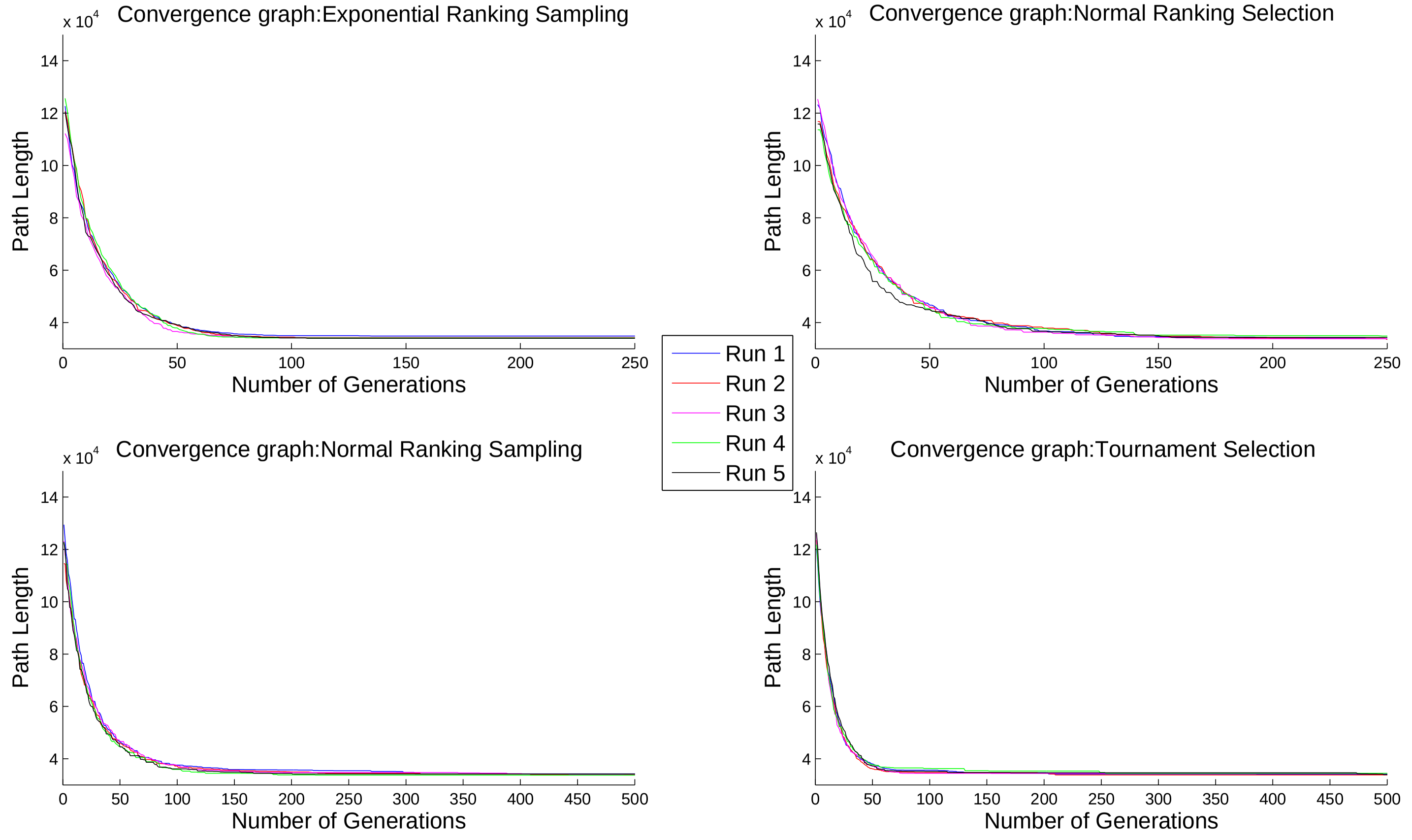


Figure 9: Comparative convergence graph of selection methods using ATT48 Simulation, Methods 8-11

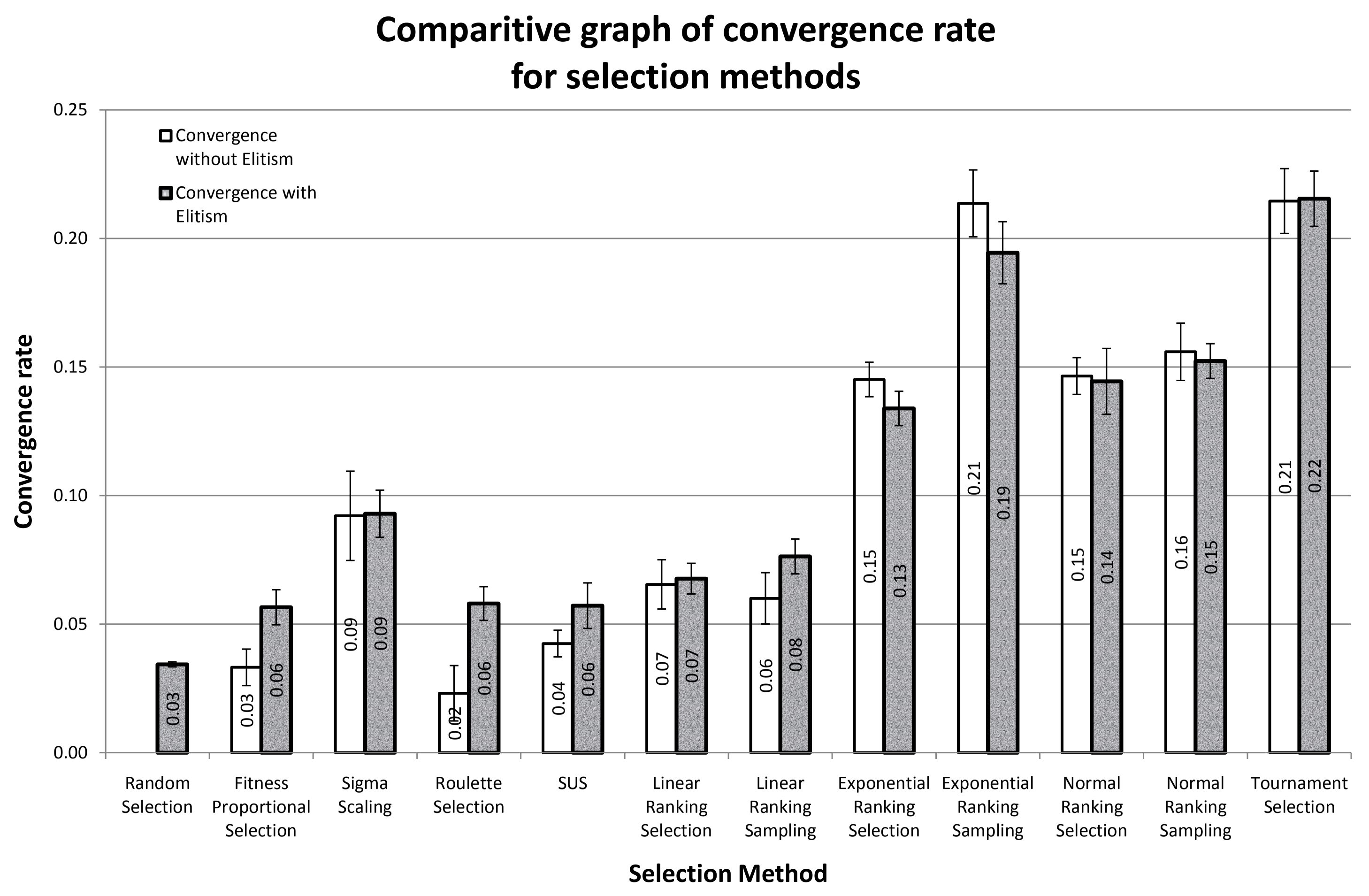


Figure 10: Comparative convergence tempos of the selection methods

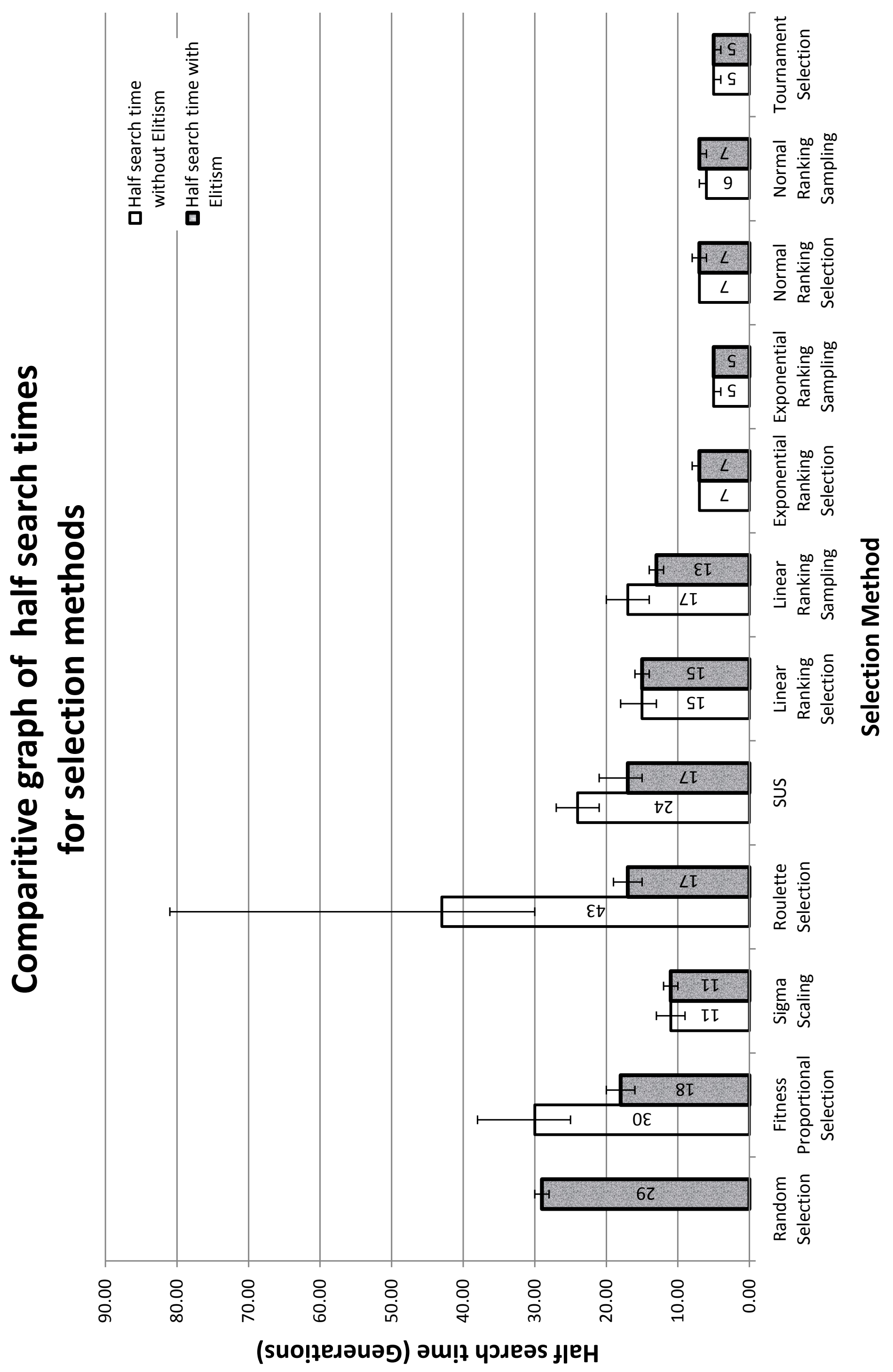


Figure 11: Comparative half search times of the selection methods

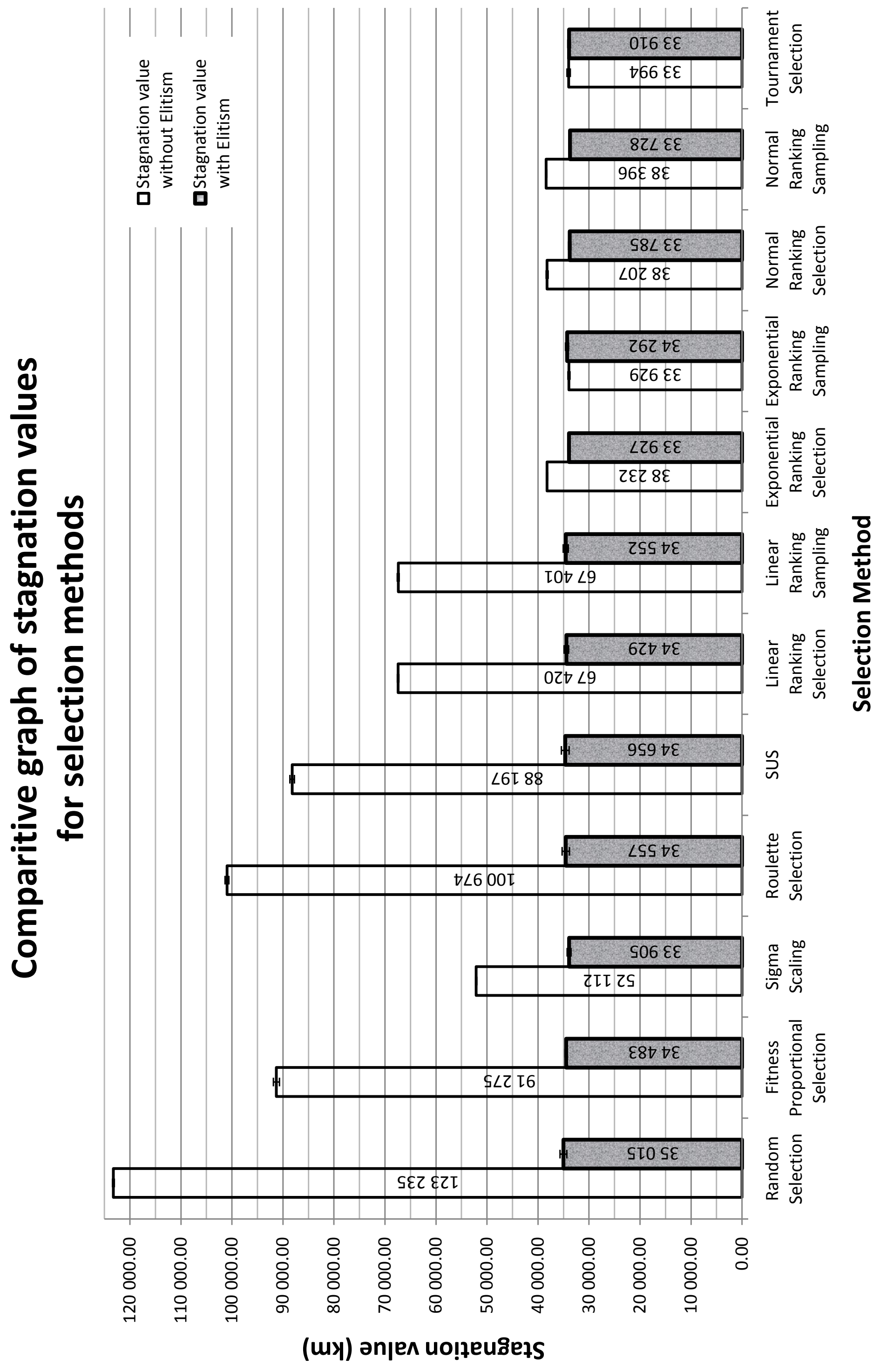


Figure 12: Comparative stagnation values of the selection methods

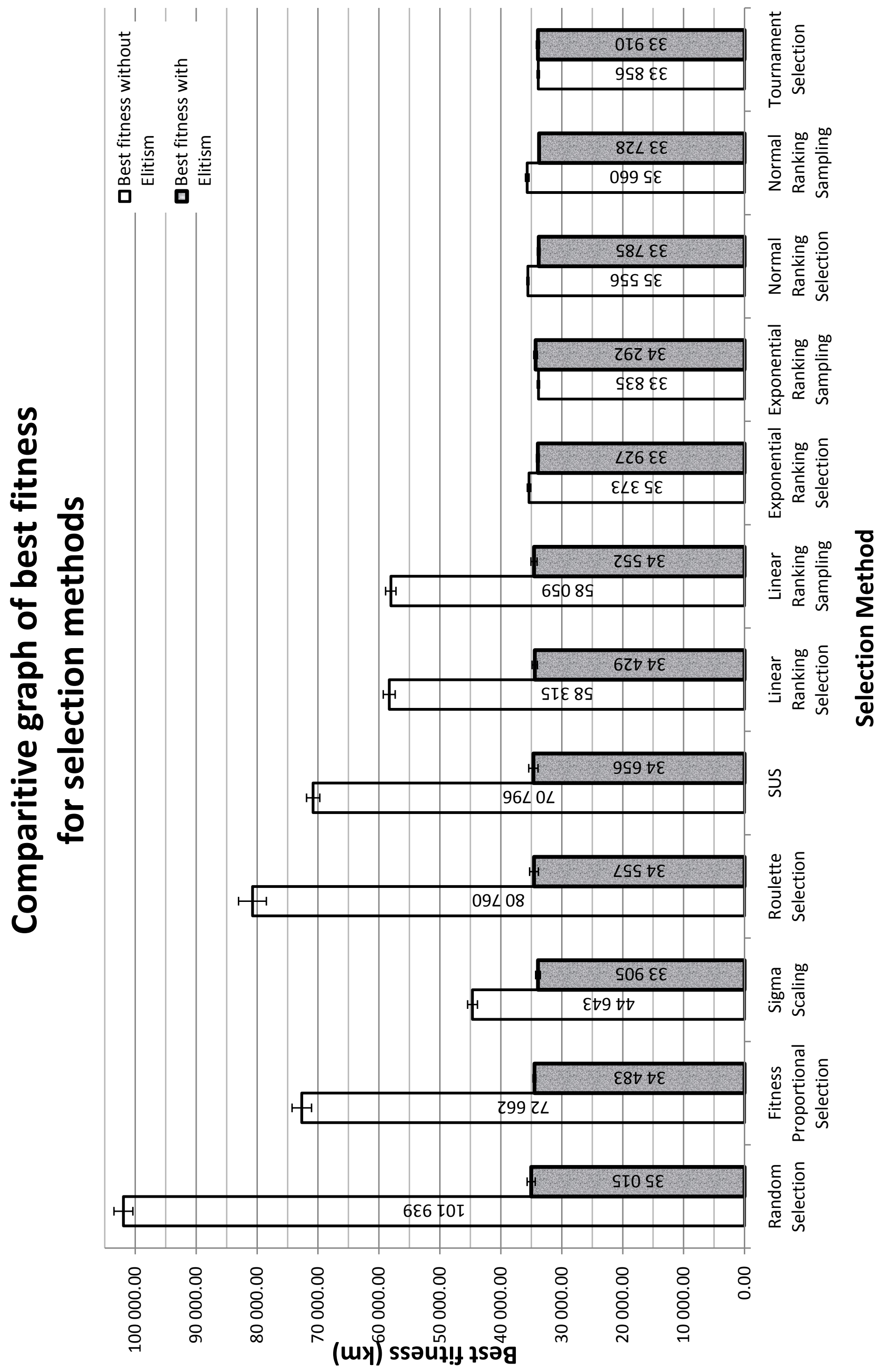


Figure 13: Comparative best fitness of the selection methods

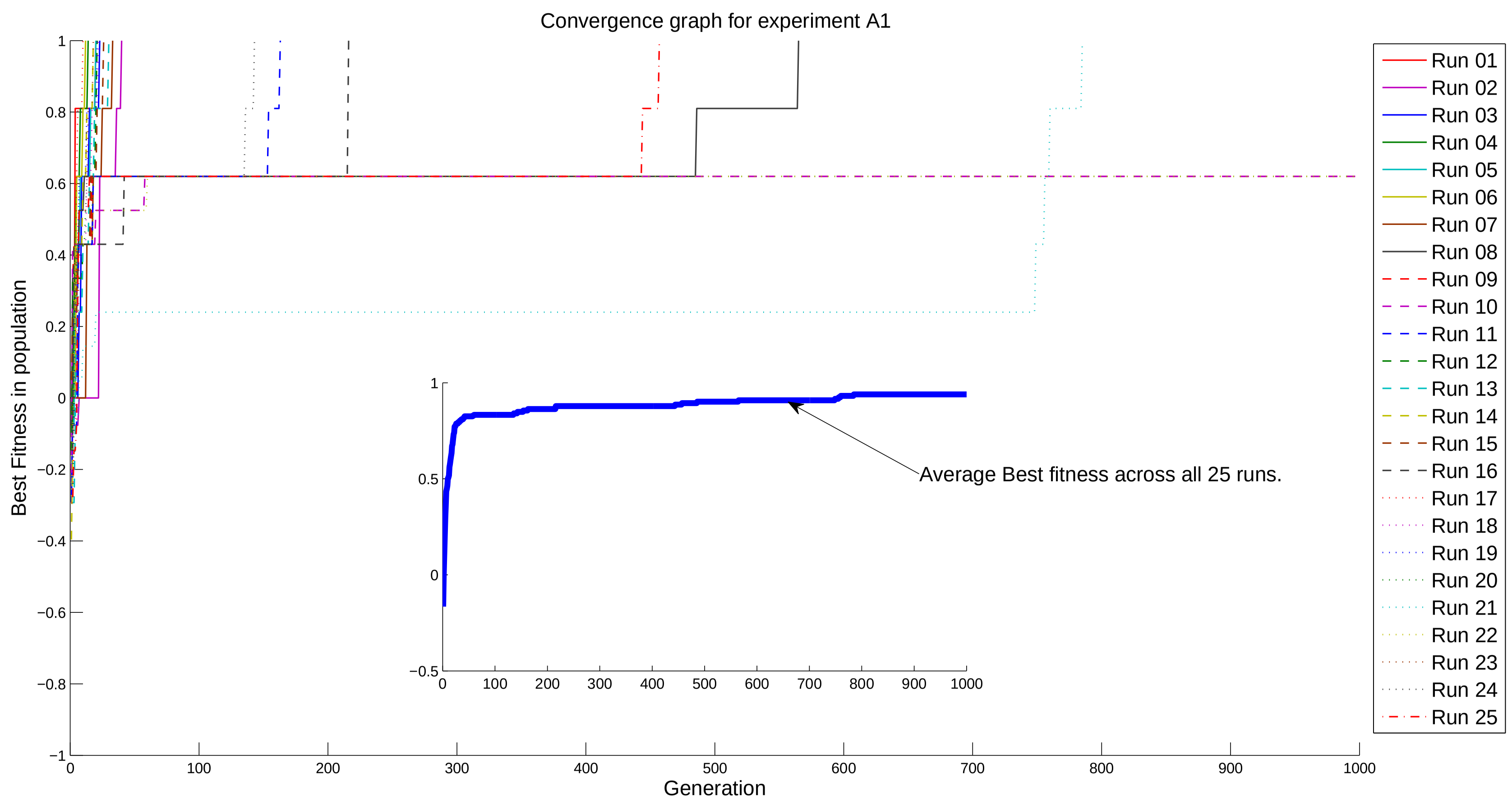
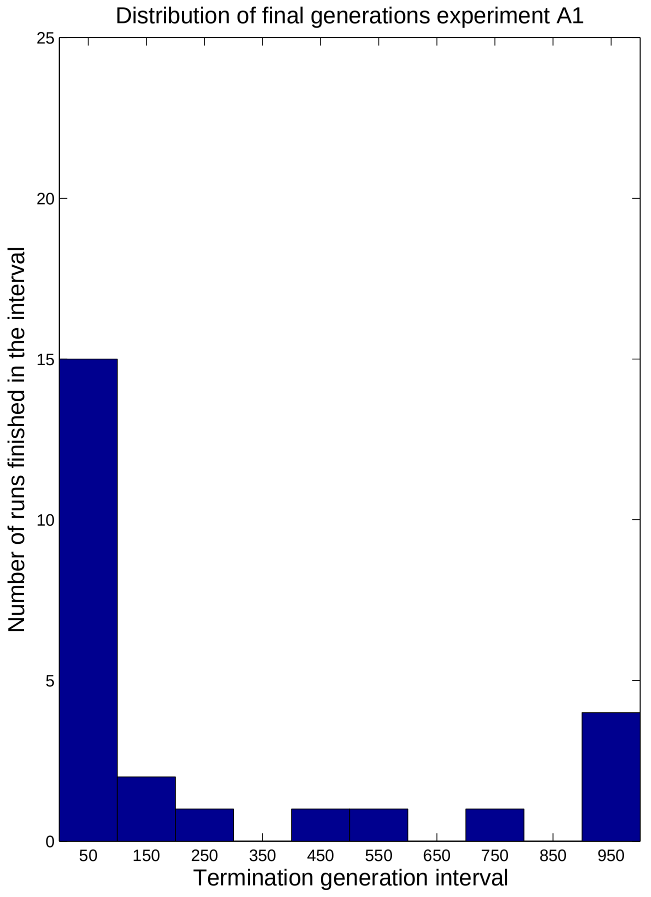


Figure 14: Convergence graph for ensemble of run - Experiment A1



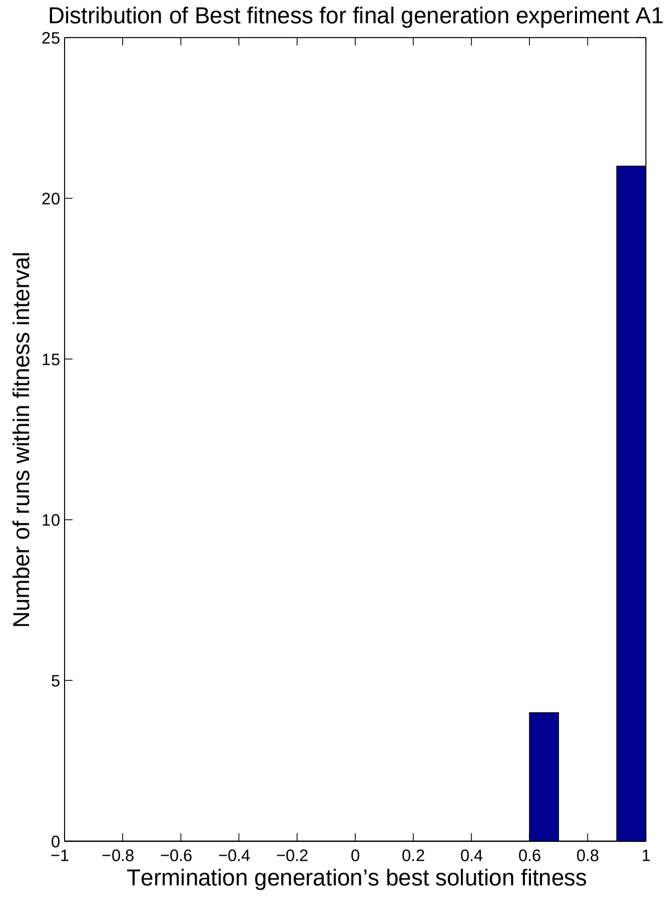


Figure 15: Solution distribution for Experiment A1

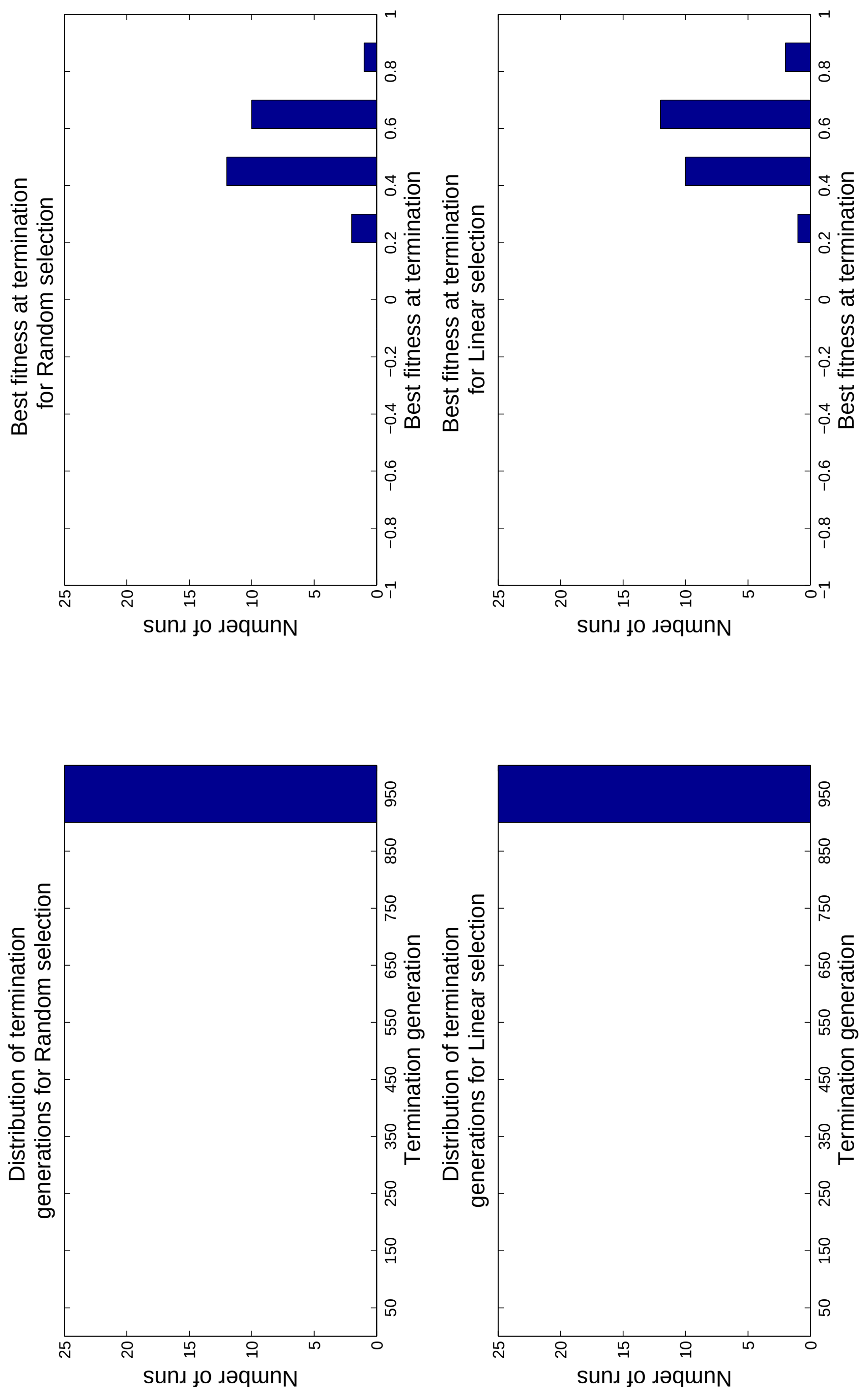


Figure 16: Solution distribution for Experiment A2 - Random and Linear selections

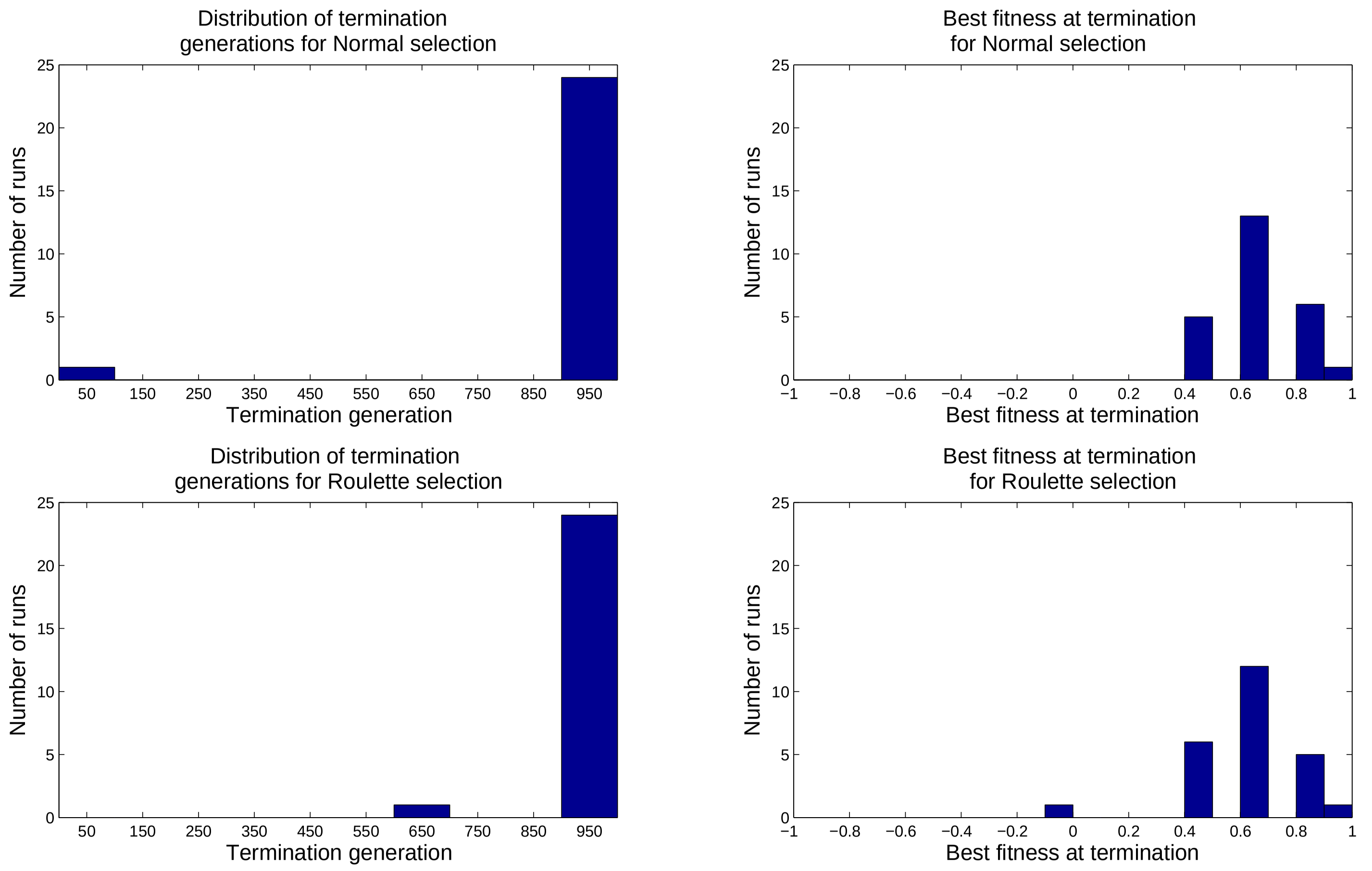


Figure 17: Solution distribution for Experiment A2 - Normal and Roulette selections

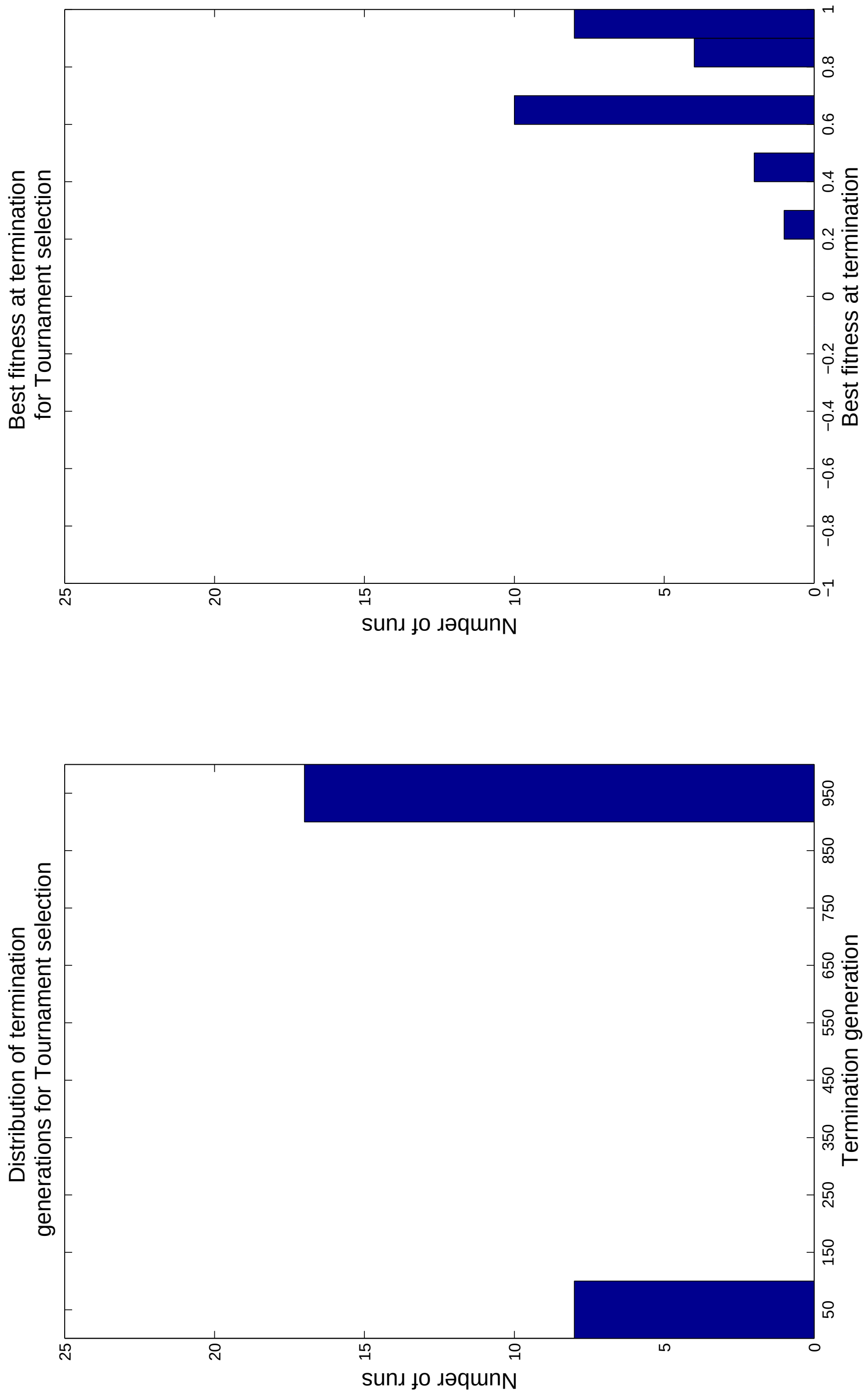


Figure 18: Solution distribution for Experiment A2 - Tournament selection

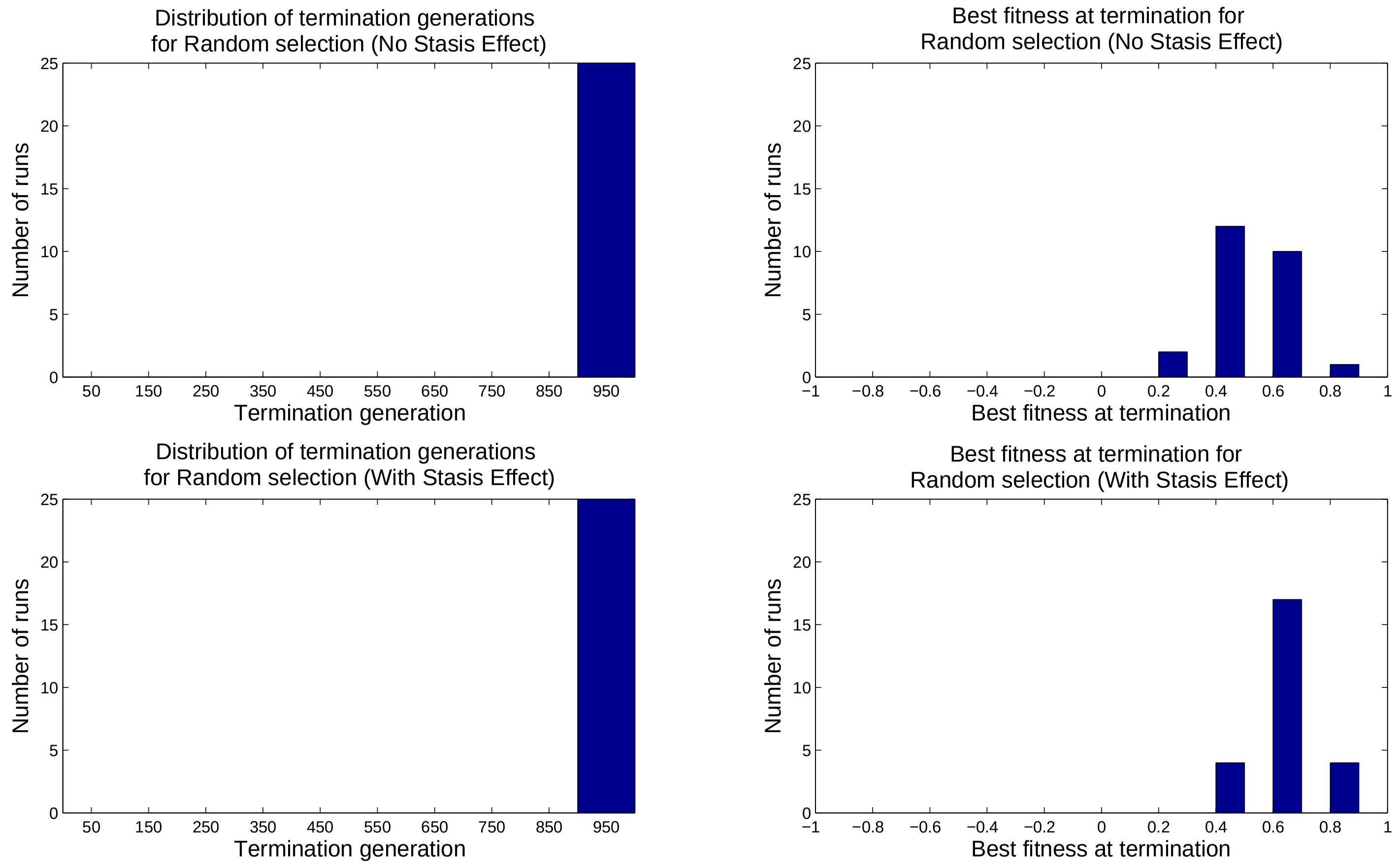


Figure 19: Solution distribution for Experiment A3 - Random selection

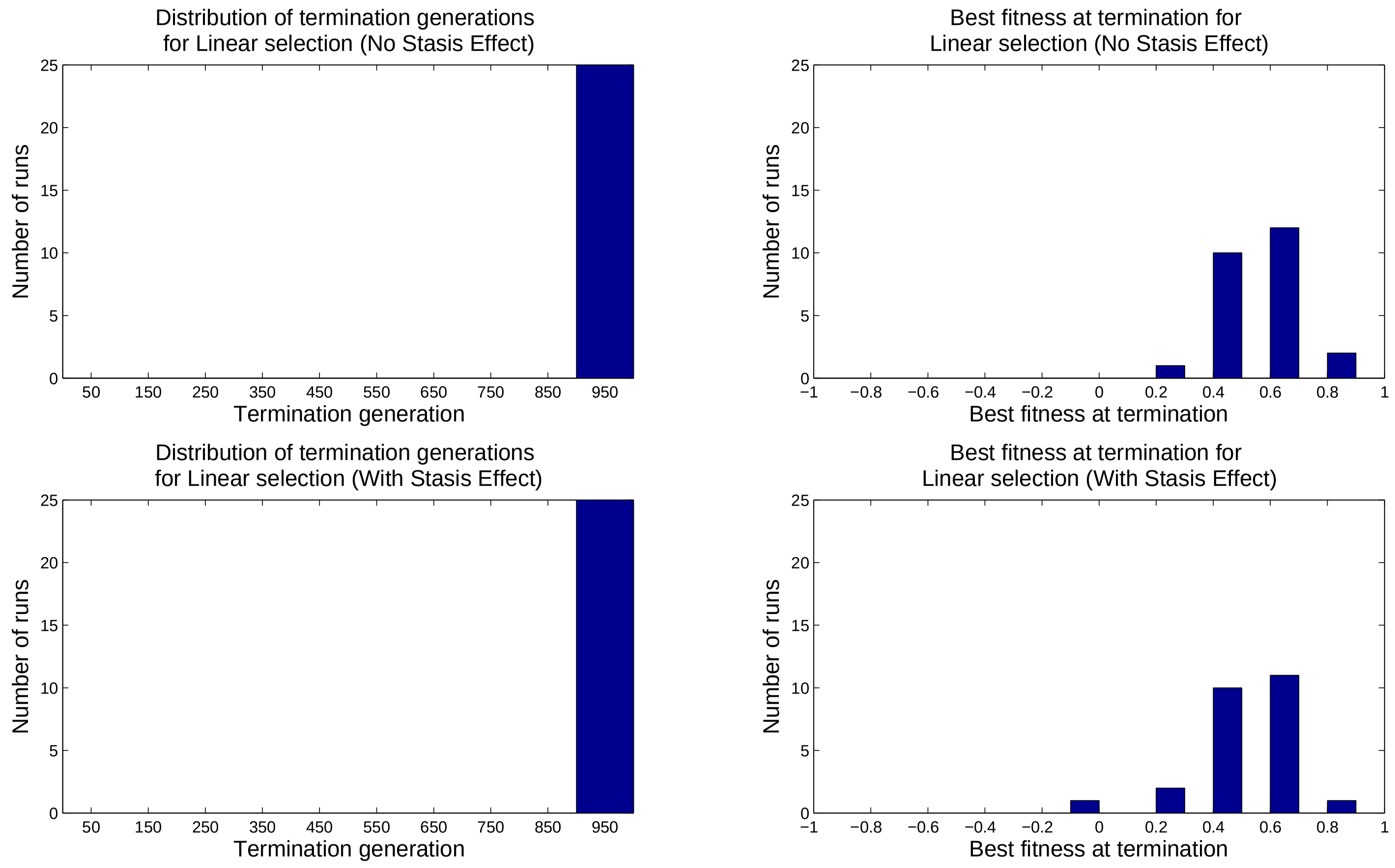


Figure 20: Solution distribution for Experiment A3 - Linear selections

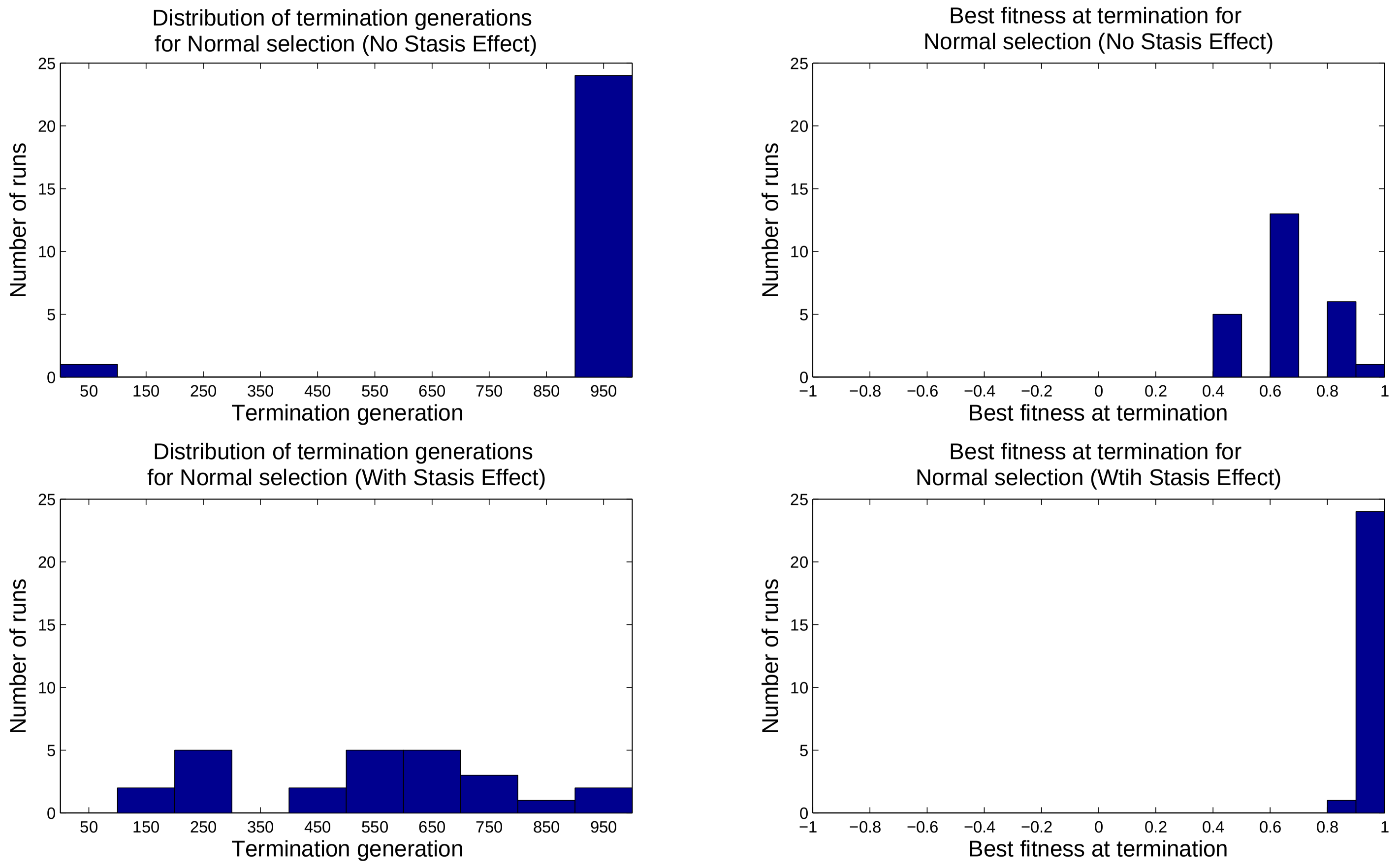


Figure 21: Solution distribution for Experiment A3 - Normal selection

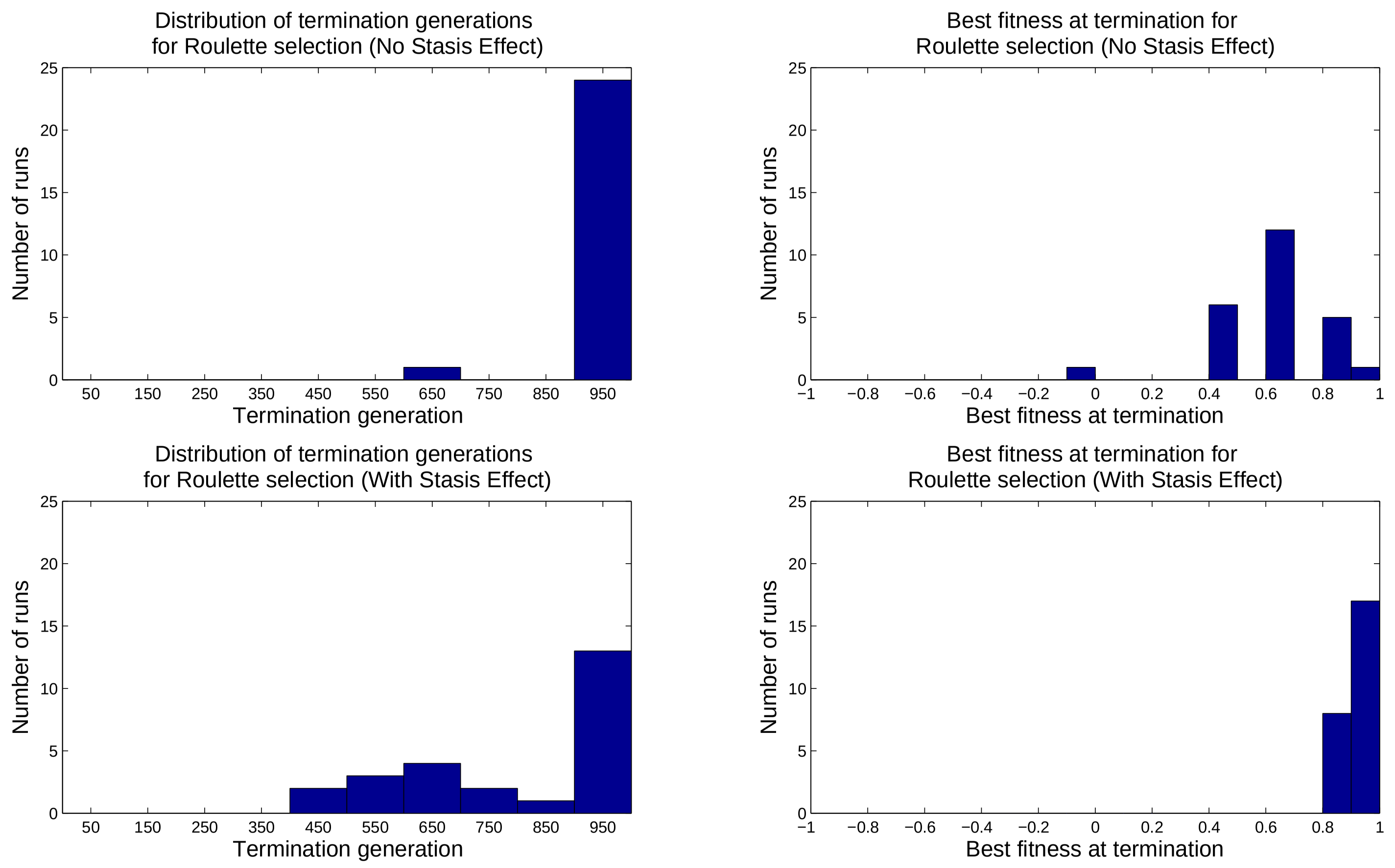


Figure 22: Solution distribution for Experiment A3 - Roulette selection

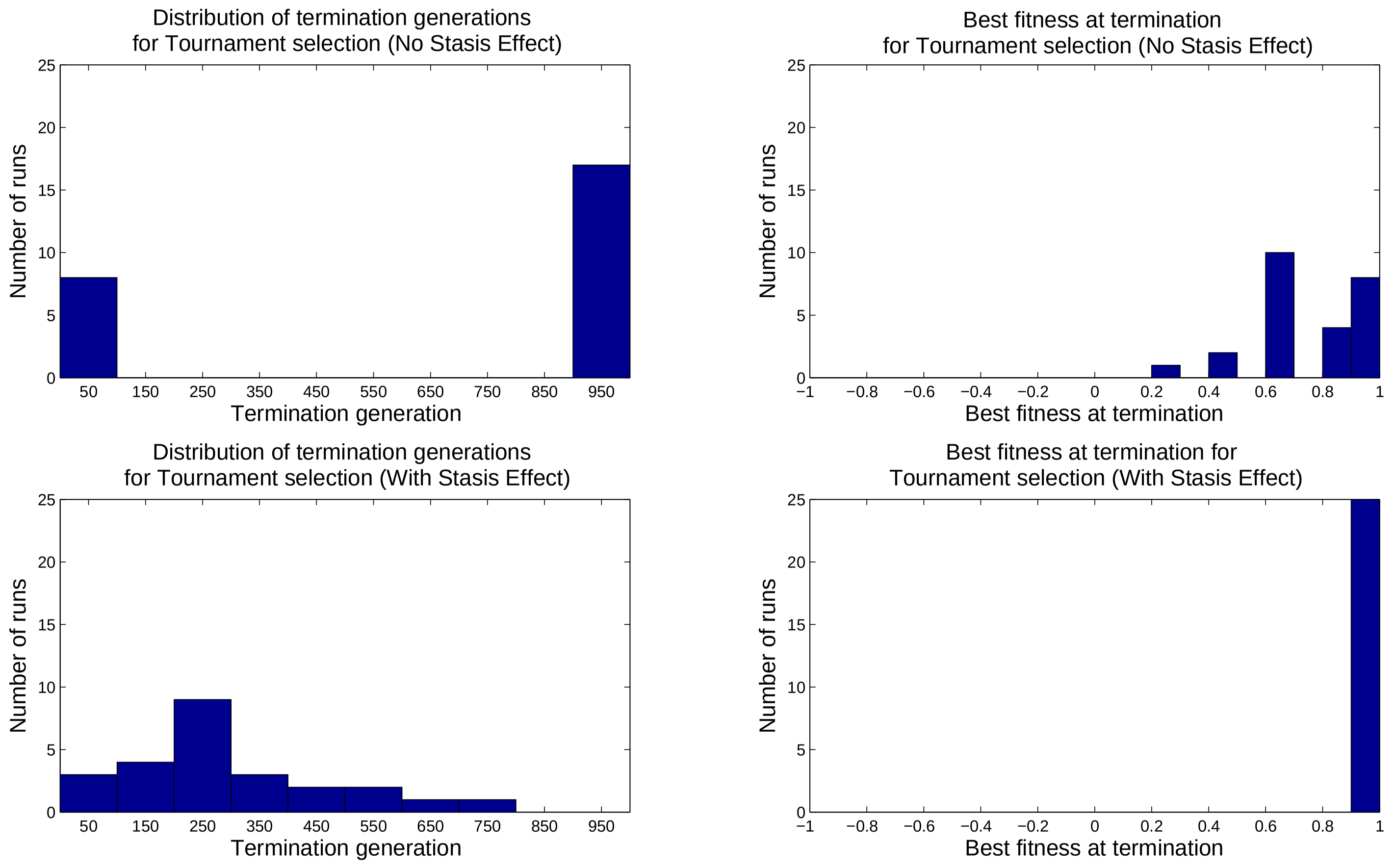


Figure 23: Solution distribution for Experiment A3 - Tournament selection

PART II

Tables

Selection Method	% Elitism	Experimental Run	R-square of Fit	Alpha	95% Confidence Bound	Half Search Time	95% Confidence, Upper bound	95% Confidence, Lower bound	Stagnation Value	95% Confidence Bound	Best Fitness Value	95% Confidence Bound
Random Selection	0	Run 1							123094.4		102513.2	
	0	Run 2							122966.1		100658.6	
	0	Run 3							123463.5		103130.5	
	0	Run 4							123375.6		103817.8	
	0	Run 5							123273.1		99573.44	
	0	Average							123234.6	178.2746	101938.7	1550.389
Fitness Proportional Selection	0	Run 1	0.767129	0.01285	0.0159				91439.24		73625.16	
	0	Run 2	0.945545	0.0123	0.01358				91095.7		73726.17	
	0	Run 3	0.966887	0.007588	0.007941				90348.88		70194.84	
	0	Run 4	0.865532	0.0092	0.01112				92240.63		74474.36	
	0	Run 5	0.955396	0.00803	0.008644				91250.18		71288.47	
	0	Average		0.033198	0.007095	30	8	5	91274.93	596.0803	72661.8	1599.399
Sigma Scaling	0	Run 1	0.986745	0.02448	0.02536				51948.79		44108.44	
	0	Run 2	0.991386	0.0306	0.0316				52022.74		45085.94	
	0	Run 3	0.986598	0.02461	0.02543				52300.81		44337.83	
	0	Run 4	0.988131	0.03682	0.03865				52129.71		43666.94	
	0	Run 5	0.976476	0.02215	0.02315				52158.87		46014.83	
	0	Average		0.092124	0.017367	11	2	2	52112.18	118.2208	44642.79	809.2648
Roulette Selection	0	Run 1	0.697087	0.005172	0.006292				100730.8		77307.01	
	0	Run 2	0.82989	0.004818	0.005451				101217.6		79857.01	
	0	Run 3	0.712377	0.004577	0.005591				100435.4		82971.71	
	0	Run 4	0.466414	0.00686	0.01049				101552.2		79936.53	
	0	Run 5	0.76371	0.01338	0.01706				100934.7		83728.25	
	0	Average		0.023125	0.010767	43	38	13	100974.1	378.1163	80760.1	2282.278
SUS	0	Run 1	0.891565	0.01264	0.01459				87519.09		71064.32	
	0	Run 2	0.901863	0.01239	0.01456				88719.67		71290.65	
	0	Run 3	0.934988	0.01097	0.01191				88526.04		70224.24	
	0	Run 4	0.757959	0.01575	0.023				88408.53		72358.14	
	0	Run 5	0.929957	0.01215	0.01391				87813.11		69040.21	
	0	Average		0.042454	0.005181	24	3	3	88197.29	445.4603	70795.51	1088.323
Linear Ranking Selection	0	Run 1	0.962729	0.01943	0.02091				67652.4		59035.46	
	0	Run 2	0.977933	0.02502	0.02668				67184.93		57005.81	
	0	Run 3	0.989867	0.01594	0.01645				67329.88		59356.39	
	0	Run 4	0.974369	0.01914	0.02048				67440.6		57196.56	
	0	Run 5	0.947862	0.019	0.02069				67494.53		58978.66	
	0	Average		0.065462	0.009582	15	3	2	67420.47	153.9323	58314.58	980.8516
Linear Ranking Sampling	0	Run 1	0.981321	0.01692	0.01763				67569.8		58780.23	
	0	Run 2	0.961474	0.02342	0.02535				67118.64		57780.38	
	0	Run 3	0.989848	0.01635	0.01699				67214.75		56887.81	
	0	Run 4	0.98269	0.01449	0.01509				67509.59		57539.11	
	0	Run 5	0.988201	0.0192	0.01993				67590.34		59305.38	
	0	Average		0.060047	0.009981	17	3	3	67400.62	191.3077	58058.58	853.5674
Exponential Ranking Selection	0	Run 1	0.992783	0.04732	0.04888				38348.22		35334.72	
	0	Run 2	0.994191	0.04387	0.04517				38383.82		35677.34	
	0	Run 3	0.982002	0.04264	0.04488				38128.61		35725.51	
	0	Run 4	0.987575	0.04359	0.04549				38229.32		35071.5	
	0	Run 5	0.986966	0.04107	0.04289				38069.97		35054.11	
	0	Average		0.145162	0.006701	7	0	0	38231.99	118.8238	35372.63	280.9472

Table 1: Summary of data for TSP experiments (Part 1)

Selection Method	% Elitism	Experimental Run	R-square of Fit	Alpha	95% Confidence Bound	Half Search Time	95% Confidence, Upper bound	95% Confidence, Lower bound	Stagnation Value	95% Confidence Bound	Best Fitness Value	95% Confidence Bound
Exponential Ranking Sampling	0	Run 1	0.995699	0.06569	0.06736				33974.66		33710.99	
	0	Run 2	0.995038	0.06674	0.06857				34120.98		33966.14	
	0	Run 3	0.992913	0.06734	0.06954				33962.81		33929.54	
	0	Run 4	0.992901	0.05645	0.0583				33532.14		33523.71	
	0	Run 5	0.992617	0.06536	0.06754				34053.11		34045.64	
	0	Average		0.213653	0.013011	5	0	1	33928.74	202.2588	33835.21	187.3725
Normal Ranking Selection	0	Run 1	0.996474	0.0462	0.04726				38519.26		35732.34	
	0	Run 2	0.995543	0.04444	0.04559				38266.13		35513.37	
	0	Run 3	0.990611	0.04639	0.04814				38249.39		35739.57	
	0	Run 4	0.991896	0.04305	0.04456				37925.9		35534.74	
	0	Run 5	0.989742	0.04045	0.04205				38073.19		35260.67	
	0	Average		0.146517	0.007161	7	0	0	38206.77	195.8607	35556.14	172.1431
Normal Ranking Sampling	0	Run 1	0.996024	0.04567	0.0471				38291.85		35228.62	
	0	Run 2	0.992062	0.05107	0.05333				38314.42		35830.66	
	0	Run 3	0.98451	0.0468	0.04971				38637.44		35901.23	
	0	Run 4	0.989407	0.04133	0.04345				38346.47		35973.16	
	0	Run 5	0.991257	0.04983	0.05215				38389.28		35365.97	
	0	Average		0.155931	0.011137	6	1	0	38395.89	122.6278	35659.93	296.5744
Tournament Selection	0	Run 1	0.996304	0.06606	0.06712				34319.52		34068.81	
	0	Run 2	0.994179	0.06772	0.06909				33831.74		33831.73	
	0	Run 3	0.987145	0.06917	0.07124				33759.79		33701.26	
	0	Run 4	0.994722	0.05991	0.06106				34457.3		34076.62	
	0	Run 5	0.982726	0.0601	0.06219				33600.57		33600.56	
	0	Average		0.21457	0.012608	5	0	1	33993.78	326.9634	33855.8	187.8628
Random Selection	1	Run 1	0.985004	0.01083	0.0111				34568.17		34568.17	
	1	Run 2	0.98509	0.0103	0.01055				35908.28		35908.28	
	1	Run 3	0.985406	0.01036	0.01061				35266.17		35266.17	
	1	Run 4	0.990325	0.009906	0.0101				33952.71		33952.71	
	1	Run 5	0.982149	0.0103	0.01057				35381.47		35381.47	
	1	Average		0.034346	0.000957	29	1	1	35015.36	668.0377	35015.36	668.0377
Fitness Proportional Selection	1	Run 1	0.977288	0.01587	0.01657				34385.13		34385.13	
	1	Run 2	0.990014	0.01595	0.01641				34374.3		34374.3	
	1	Run 3	0.970313	0.02121	0.02229				34870.95		34870.95	
	1	Run 4	0.973555	0.01613	0.0169				34621.52		34621.52	
	1	Run 5	0.960564	0.01598	0.01691				34164.98		34164.98	
	1	Average		0.056566	0.006813	18	2	2	34483.38	236.9108	34483.38	236.9108
Sigma Scaling	1	Run 1	0.9758	0.03093	0.03234				33872.16		33872.16	
	1	Run 2	0.9873	0.02584	0.02669				33784.03		33784.03	
	1	Run 3	0.993257	0.03153	0.03229				33607.68		33607.68	
	1	Run 4	0.973685	0.02433	0.02549				34659.68		34659.68	
	1	Run 5	0.984494	0.02727	0.02826				33600.56		33600.56	
	1	Average		0.092948	0.009174	11	1	1	33904.82	383.6559	33904.82	383.6559
Roulette Selection	1	Run 1	0.981851	0.01887	0.01961				33917.08		33917.08	
	1	Run 2	0.991313	0.0163	0.01674				34346.43		34346.43	
	1	Run 3	0.988427	0.01516	0.01564				35769.88		35769.88	
	1	Run 4	0.993684	0.02068	0.02116				34966.15		34966.15	
	1	Run 5	0.986278	0.01633	0.01689				33784.03		33784.03	
	1	Average		0.058027	0.006554	17	2	2	34556.71	718.777	34556.71	718.777

Table 2: Summary of data for TSP experiments (Part 2)

Selection Method	% Elitism	Experimental Run	R-square of Fit	Alpha	95% Confidence Bound	Half Search Time	95% Confidence, Upper bound	95% Confidence, Lower bound	Stagnation Value	95% Confidence Bound	Best Fitness Value	95% Confidence Bound
SUS	1	Run 1	0.95576	0.01636	0.01738				33929.54		33929.54	
	1	Run 2	0.986071	0.0219	0.02265				34938.79		34938.79	
	1	Run 3	0.974483	0.01441	0.01509				33883.61		33883.61	
	1	Run 4	0.988547	0.01842	0.019				34534.88		34534.88	
	1	Run 5	0.967482	0.01498	0.01578				35992.78		35992.78	
	1	Average		0.057184	0.008855	17	4	2	34655.92	759.8435	34655.92	759.8435
Linear Ranking Selection	1	Run 1	0.970501	0.02183	0.02293				34738.38		34738.38	
	1	Run 2	0.979604	0.02106	0.02195				35111.61		35111.61	
	1	Run 3	0.965435	0.01909	0.02014				33831.73		33831.73	
	1	Run 4	0.990942	0.01749	0.01797				34363.59		34363.59	
	1	Run 5	0.980652	0.02243	0.02334				34099.7		34099.7	
	1	Average		0.067701	0.005964	15	1	1	34429	445.0278	34429	445.0278
Linear Ranking Sampling	1	Run 1	0.979978	0.02421	0.02555				33936.3		33936.3	
	1	Run 2	0.981201	0.01957	0.02061				33979.36		33979.36	
	1	Run 3	0.984653	0.02559	0.02683				35229.77		35229.77	
	1	Run 4	0.982017	0.02185	0.02299				34908.59		34908.59	
	1	Run 5	0.974172	0.02368	0.02517				34704.99		34704.99	
	1	Average		0.076338	0.006785	13	1	1	34551.8	502.9383	34551.8	502.9383
Exponential Ranking Selection	1	Run 1	0.991415	0.03766	0.03867				34195.06		34195.06	
	1	Run 2	0.985547	0.0437	0.04524				33784.03		33784.03	
	1	Run 3	0.989008	0.04065	0.0419				33701.26		33701.26	
	1	Run 4	0.989707	0.03882	0.03997				33723.78		33723.78	
	1	Run 5	0.99044	0.0407	0.04186				34228.44		34228.44	
	1	Average		0.133894	0.006672	7	1	0	33926.51	229.9999	33926.51	229.9999
Exponential Ranking Selection	1	Run 1	0.997168	0.05483	0.05568				34875.15		34875.15	
	1	Run 2	0.995516	0.05711	0.05822				34299.74		34299.74	
	1	Run 3	0.99367	0.06539	0.06691				34148.49		34148.49	
	1	Run 4	0.995207	0.0592	0.0604				34154.23		34154.23	
	1	Run 5	0.988566	0.05614	0.05789				33981.71		33981.71	
	1	Average		0.194446	0.012087	5	0	0	34291.87	302.3598	34291.87	302.3598
Normal Ranking Selection	1	Run 1	0.995905	0.04326	0.04425				33701.26		33701.26	
	1	Run 2	0.994695	0.04023	0.04127				33523.71		33523.71	
	1	Run 3	0.993851	0.04265	0.04384				33588.34		33588.34	
	1	Run 4	0.994722	0.04029	0.04133				34068.81		34068.81	
	1	Run 5	0.994409	0.05096	0.05231				34043.56		34043.56	
	1	Average		0.144431	0.012811	7	1	1	33785.14	224.0538	33785.14	224.0538
Normal Ranking Sampling	1	Run 1	0.997248	0.04571	0.04664				33701.26		33701.26	
	1	Run 2	0.993095	0.04436	0.04579				33600.56		33600.56	
	1	Run 3	0.991212	0.04291	0.04447				33784.03		33784.03	
	1	Run 4	0.995391	0.04831	0.04958				33723.78		33723.78	
	1	Run 5	0.994351	0.04799	0.04939				33831.73		33831.73	
	1	Average		0.15233	0.006753	7	0	1	33728.27	77.00408	33728.27	77.00408
Tournament Selection	1	Run 1	0.982269	0.06226	0.06468				33966.14		33966.14	
	1	Run 2	0.998393	0.07075	0.07157				33831.73		33831.73	
	1	Run 3	0.986632	0.0649	0.06709				34377.78		34377.78	
	1	Run 4	0.990953	0.06125	0.06295				33784.03		33784.03	
	1	Run 5	0.997031	0.06517	0.0662				33588.34		33588.34	
	1	Average		0.21548	0.010756	5	0	1	33909.6	258.3228	33909.6	258.3228

Table 3: Summary of data for TSP experiments (Part 3)

Ranking	Selection Method	Elitism	Percentage Difference
1	Normal Ranking Sampling	1%	0.61%
2	Normal Ranking Selection	1%	0.78%
3	Exponential Ranking Sampling	0%	0.93%
4	Tournament Selection	0%	0.99%
5	Sigma Scaling	1%	1.14%
6	Tournament Selection	1%	1.15%
7	Exponential Ranking Selection	1%	1.20%
8	Exponential Ranking Sampling	1%	2.29%
9	Linear Ranking Selection	1%	2.70%
10	Fitness Proportional Selection	1%	2.86%
11	Linear Ranking Sampling	1%	3.07%
12	Roulette Selection	1%	3.08%
13	SUS	1%	3.38%
14	Random Selection	1%	4.45%
15	Exponential Ranking Selection	0%	5.51%
16	Normal Ranking Selection	0%	6.06%
17	Normal Ranking Sampling	0%	6.37%
18	Sigma Scaling	0%	33.17%
19	Linear Ranking Sampling	0%	73.19%
20	Linear Ranking Selection	0%	73.95%
21	SUS	0%	111.18%
22	Fitness Proportional Selection	0%	116.75%
23	Roulette Selection	0%	140.90%
24	Random Selection	0%	204.08%

Table 4: Accuracy rankings for selection methods

Ranking	Selection Method	Elitism	Alpha	Half search time	95% Confidence, Upper Interval	95% Confidence, Lower Interval
1	Exponential Ranking Sampling	0%	0.2137	5	0	1
2	Tournament Selection	0%	0.2146	5	0	1
3	Tournament Selection	1%	0.2155	5	0	1
4	Exponential Ranking Sampling	1%	0.1944	5	0	0
5	Normal Ranking Sampling	0%	0.1559	6	1	0
6	Normal Ranking Sampling	1%	0.1523	7	0	1
7	Normal Ranking Selection	1%	0.1444	7	1	1
8	Exponential Ranking Selection	1%	0.1339	7	1	0
9	Exponential Ranking Selection	0%	0.1452	7	0	0
10	Normal Ranking Selection	0%	0.1465	7	0	0
11	Sigma Scaling	1%	0.0929	11	1	1
12	Sigma Scaling	0%	0.0921	11	2	2
13	Linear Ranking Sampling	1%	0.0763	13	1	1
14	Linear Ranking Selection	1%	0.0677	15	1	1
15	Linear Ranking Selection	0%	0.0655	15	3	2
16	Roulette Selection	1%	0.0580	17	2	2
17	SUS	1%	0.0572	17	4	2
18	Linear Ranking Sampling	0%	0.0600	17	3	3
19	Fitness Proportional Selection	1%	0.0566	18	2	2
20	SUS	0%	0.0425	24	3	3
21	Random Selection	1%	0.0343	29	1	1
22	Fitness Proportional Selection	0%	0.0332	30	8	5
23	Roulette Selection	0%	0.0231	43	38	13
24	Random Selection	0%				

Table 5: Efficiency rankings for selection methods

PART III

List of acronyms

TSP

The traveling salesman problem, is a well know combinatorics problem, which entails finding the shortest route for a salesman to do a sales tour visiting a number of cities only once.

TSP-Lib

This is a library of problems with the current best solutions for a large number of TSP problems.

ATT48

ATT48 is the designation in TSP-Lib for the TSP problem involving the state capitals of the USA.

NP-complete

A classification in computational complexity theory, where the problems solutions can be verified in polynomial time, but a polynomial time solution is not know.

GRASP

Greedy Randomized Adaptive Search Procedures, is a type of meta-heuristic optimization method.

ACO

Ant Colony Optimization is another type of meta-heuristic optimization method.

EA

Evolutionary Algorithm is also a type of meta-heuristic optimization method.

RSA-encryption

This is an algorithm for public-key cryptography that is effective because factoring large integers is computationally very hard.

CNOT

Controlled Not operation. This is a two input operation that flips the value of the second input, if the first input is true, otherwise it doesn't change anything.

NMR

Nuclear Magnetic Resonance is a physical phenomenon in which magnetic nuclei in a magnetic field absorb and re-emit electromagnetic radiation. This effect is commonly used in analysis, and more recently in construction of quantum computers.

ELE

An Extinction Level Event, is a catastrophic event in a biosphere that kills off most or all life.

QFT

Quantum Fourier Transform is the quantum mechanical equivalent of the quick fourier transform in and clasical electronic circuit.