DISTRIBUTED COMPUTATIONAL INTELLIGENCE APPLIED IN BIOINFORMATICS

PENG WEI

NATIONAL UNIVERSITY OF SINGAPORE

2004

DISTRIBUTED COMPUTATIONAL INTELLIGENCE APPLIED IN BIOINFORMATICS

PENG WEI
(B. Eng (1st class honors) National University of Singapore)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2004

Acknowledgements

I would like to extend my grateful appreciation to all that helped me in their unique ways throughout the course of this project.

First and most thanks go out to my supervisors, Dr. Vadakkepat, Prahlad and Dr. Tan Kay Chen, for their scrupulous and brilliant supervision, their most-needed encouragement and for their wise suggestion and positive criticisms.

Greatest thanks to Dr. Tay Ee Beng, Arthur for his patiently entertaining all my doubts and requests, for leading me to the inspiring path to explore in the bioinformatics world.

I am also grateful to all the individuals in the Control and Simulation Lab, Department of Electrical and Computer Engineering, National University of Singapore, which provides the research facilities to conduct the research work.

Finally, I wish to acknowledge National University of Singapore (NUS) for the financial support provided throughout my research work.

Summary

DNA microarray is the latest bioinformatics technology which is high-throughput and large-scale, making study complex interplay of all genes simultaneously possible. This thesis reports the effort of applying a newly developed distributed computational intelligence package, Paladin-DES to a real world bioinformatics problem, to search the oligo probe sets of human malaria parasite, *Plasmodium Falciparum* to be printed on the DNA microarrays.

Normal evolutionary computation has changed the traditional single-point gradient-guided search technique to a population-based searching algorithm, which both reduces the searching time and improves the optimum searching results. However, for some very complicated searching problems, even evolutionary computation is also cost impractical or extreme time-consuming.

The Paladin-DES package is developed on the bases of Paladin-DEC package, which exploits the inherent parallelism of evolutionary algorithms by creating an infrastructure necessary to support distributed evolutionary computing using existing Internet and hardware resources. Through the simulation test of searching the probes for the *Plasmodium Falciparum*, Paladin-DES is proven to be a very good candidate in this bioinformatics area.

Plasmodium falciparum, which is the severest cause of human malaria diseases on the earth, whose gene sequence was totally identified in 2002. The distributed package is applied to the gene coding sequence file of this plasmodium to search optimum probes for subsequent medical and biology research. In this research three criteria are proposed to test whether one sequence of gene is a qualified probe or not. The criteria are based on two fundamental considerations of microarray technology, specificity and sensitivity.

Existing methods of searching probes are very rare. The results obtained by the simulation from Paladin-DES are compared with two other methods in terms of effectiveness and efficiency. Effectiveness measures the number of qualified probes found by each method and efficiency measures the time spent by every method for allocating one probe. The Paladin-DES method performs very well in both competition and can be applied for some much larger genomes sequences like plant genome in the later research.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Computational Intelligence Definition

What is computational Intelligence (CI)? What is the difference between CI and AI (Artificial Intelligence)? In 1992, Bezdek first time used the term CI and later in 1994 he gave the following definition:

> A system is computationally intelligent when it: deals only with numerical (low-level) data, has a pattern reorganization component, and does not use knowledge in the AI sense; and additionally, when it (begins to) exhibit (i) computational adaptivity; (ii) computational fault tolerance; (iii) speed approaching human-like turnaround, and (iv) error rates that approximately human performance.

Recently Engelbrecht (Engelbrecht, 2002) declares that CI is a study of adaptive mechanisms to enable or facilitate intelligence behavior in complex and changing environments.

In general, the main objective of Computational Intelligence (CI) is to establish a highly coherent design and analysis environment through a series of synergistic links that give rise to neurofuzzy systems, evolutionary neural networks, fuzzy genetic schemes, granular rough decision systems, and many others in the context of software engineering (Bezdek, 1992; Pedrycz and Peters, 1998).

Computational Intelligence covers mainly 4 paradigms: neural networks, evolutionary computation, swarm intelligence and fuzzy systems. The work in this thesis deals mainly with one of the 4 paradigms: evolutionary computation.

1.2 Project History

This project of distributed computational intelligence was introduced by Tan in 1999. In the first stage Tan and Wang designed a peer-to-peer based genetic algorithm infrastructure over the Internet. Secondly Tan and Cai designed a distributed evolutionary computation system which changed the infrastructure from a peer-to-peer frame to a totally distributed frame with underlying Java based RMI-IIOP (Remote Method Invocation over Internet Inter-ORB Protocol).

In the second phase, a distributed evolutionary computing architecture has been developed to exploit the inherent parallelism of evolutionary algorithms by creating an infrastructure necessary to support distributed evolutionary computing using existing Internet and hardware resources.

There are three evolutionary algorithms packages involved in the system designed by Tan and Cai, which are: Genetic Algorithm, Genetic Programming and Evolutionary Strategy.

This current work is the third phase of the research. In this thesis work one of the evolutionary algorithms package, the evolutionary strategy package has been modified and then applied to a real world bioinformatics problem: to search the oligo sets (probes) of malaria parasite, *Plasmodium Falciparum*.

1.3 Bioinformatics, Microarray

The availability of complete or near-complete catalogs of genes for organisms of increasing complexity has created opportunities for studying numerous aspects of gene function at the genomic level (Baxevanis and Ouellette, 2001). With readily available technology such as DNA Microarray, it is now possible to carry out massively parallel analysis of gene expression on different genomes.

DNA microarrays also referred to as DNA arrays, microarrays, DNA chips, biochips or GeneChips – allow researchers to determine which genes are being expressed in a given cell type at a particular time and under particular conditions (Gershon, 2002).

They can be used to compare the gene expression in 2 different cell types or tissue samples; for example, healthy versus diseased tissues to examine which genes are the causes of the diseases. Unlike conventional nucleic-acid hybridization methods, microarrays can identify thousands of genes simultaneously, which means that genetic analysis can be done on a huge scale (Lockhart and Winzeler, 2000).

DNA molecules, typically in the form of double stranded PCR (Polymerase Chain Reaction) products or oligonucleotides (oligo), can be attached to glass slides or nylon membranes (Schena et al, 1995). These oligo sets are typically optimized sequences of a particular genome which can represent the key characteristics of that genome.

For example, the yeast genome consists of about 6000 genes of varying length; to print all these 6000 genes onto the microarray would not be practical as their varying length results in different melting temperature and thus different processing temperature. The objective is thus to be able to extract 6000 optimized and unique sequences from the original 6000 genes, these 6000 unique sequences is called the olgio sets (probes) of the genome. Optimized oligo sets allow for more efficient analysis of the microarray. However, most of current oligo sets are only available through commercial companies (Operon) involving high cost.

It is our objective in this project to explore computational efficient methods in extracting these optimized sequences to be printed onto the microarray for the subsequent analysis.

In the literature there exist at least two confusing nomenclature systems for referring to hybridization partners. Both use common terms: "probes" and "targets". According to the nomenclature recommended by Phimister (Phimister, 1999), a "probe" is the tethered nucleic acid with known sequence, whereas a "target" is the free nucleic acid sample whose identity/abundance is being detected.

Existing techniques for searching of these probes are not really available; a standard approach one could think of is to select a probe from a sequence and comparing it with all other sequences within the genome. One would expect such a thorough search to be computationally intensive due to its large search space.

Tay and his colleagues have previously demonstrated that the use of computational intelligence techniques such as genetic algorithm and evolutionary strategy can provide us an efficient method for extracting these unique sequences (Joe, 2002 and Xu, 2003). However, most of these approaches become computationally intensive when applied to more complicated genomes.

In this project, we extend the distributed architecture to include evolutionary strategies and apply it to the malaria parasite *Plasmodium falciparum* whose genome sequence was reported recently in October 2002 (Gardner et al, 2002).

1.4 Malaria Parasite, *Plasmodium Falciparum*

The malaria parasite *Plasmodium falciparum* is responsible for hundreds of millions of cases of malaria, and kills more than one million African children annually (Gardner et al, 2002). Immune responses cannot prevent the development of symptomatic infections throughout life, and clinical immunity to the disease develops only slowly during childhood. An understanding of the obstacles to the development of protective immunity is crucial for developing rational approaches to prevent the disease (Urben et al, 1999) and remains an active area of research.

Since detailed coding sequence information about the malaria parasite, *Plasmodium falciparum*, is known, our aim is to develop a program that can search for probes/sequences within each gene so that the probes can be printed onto DNA microarrays for medical research. One probe will identically identify one specific gene, and ideally all genes should be represented by their own probes on the DNA microarray. Difficulties do arise for certain genes that are very similar to each other (may evolve from same ancestor).

1.5 Contribution

This thesis presents a newly developed distributed computational intelligence technique, a Java-based distributed evolutionary strategy package (Paladin-DES). The package has been applied to a complicated bioinformatics problem, to search the probes for the human malaria parasite, *Plasmodium Falciparum*. The traditional searching methods are very troublesome and time-consuming. This project brings the new engineering insight into the bioinformatics field, making the searching more effective and more efficient.

1.6 Thesis Outline

This thesis consists of 6 chapters and is organized as follows: Chapter 2 discusses the background of the computational intelligence, the distributed evolutionary algorithms, together with the updated Paladin-DES package. Some bioinformatics basics and the recently introduced microarray technology are presented in chapter 3. Chapter 4 describes the malaria parasite probes searching problem studied in this project. Results are shown, compared with previously developed methods and discussed in chapter 5. Conclusions are drawn in chapter 6.

Chapter 2

Distributed Computational Intelligence Technique

2.1 Introduction

With the rapidly growing demand for new software systems having increasing complexity and size, research and development work in the area of computational intelligence also grows rapidly. Computational Intelligence (CI) is an area of fundamental and applied research involving numerical information processing (in contrast to the symbolic information processing techniques of Artificial Intelligence (AI)) (Pedrycz and Peters, 1998). Nowadays, CI technologies have been used in various areas to solve problems stemming from increasing complex of forms of software system description and analysis.

Computational Intelligence covers mainly 4 different paradigms: artificial neural networks, evolutionary computation, swarm intelligence and fuzzy systems. The work in this thesis is under one of the 4 paradigms: evolutionary computation.

Evolutionary computation (EC) was first proposed by Holland (Holland, 1975) and Dejong (Dejong, 1975). The objective of EC is to model the real practical problems to natural evolution. The main concept is survival of the fittest. In 1989 Goldberg extended the early work to optimization and machine learning. An evolutionary algorithm (EA) can be considered as an iterative scheme, where each iteration cycle forms a generation of an evolutionary process.

Although EC is a very powerful tool, the computational cost involved in terms of time and hardware is quite high. EC normally needs a large population size and generation number to simulate a more realistic evolutionary model with a better approximation. Sometimes it is cost impractical and not able to be performed without the presence of high performance computing. One solution to overcome this limitation is to exploit the inherent parallel nature of EC by formulating the problem into a distributed computing structure suitable for parallel processing.

The fact is that there are complex problems which are difficult for one computer to solve; on the other hand there are many idle computers which are a large waste of resources. Hence the proposed solution is to divide the task into subtasks and solve the subtasks simultaneously using multiple computation clients, in a divide-and-conquer manner, as shown in Fig 2.1. In this project one of the distributed evolutionary algorithms- Distributed Evolutionary Strategy- is applied to the bioinformatics area.

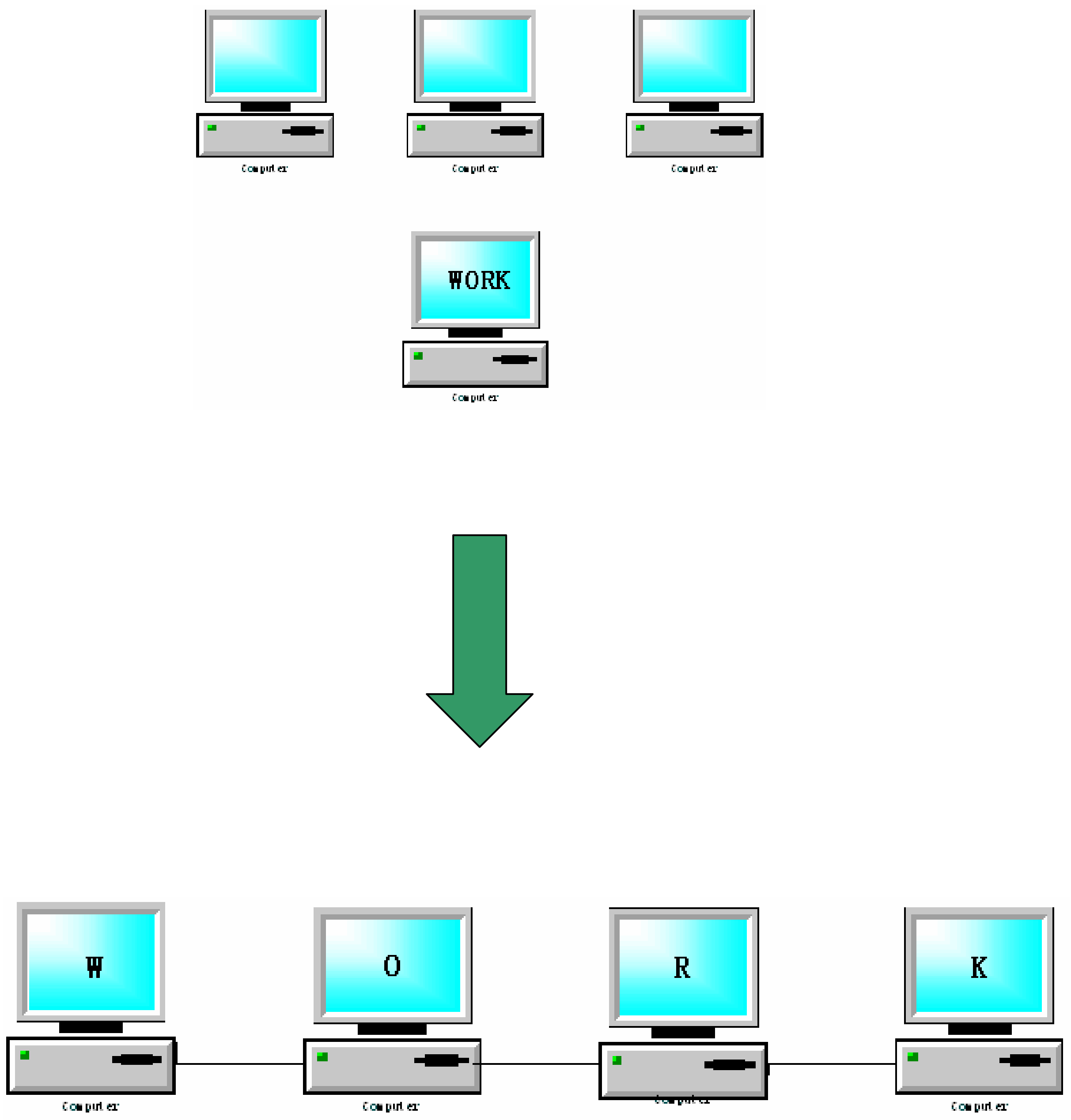


Fig 2.1 Basic concept of distributed EC

In this chapter the concept of Evolutionary Computation and then parallel EC theory is firstly discussed. After that the existing DEC package and the updated DES package are presented in details.

2.2 Evolutionary Computation

The evolutionary computation, which also refers as evolutionary algorithm (EA), attempts to mimic the genetic shift and Darwinians' struggle for survival. Unlike traditional single-point gradient-guided search techniques, the evolutionary algorithm is population-based. It attempts to evolve complex systems concurrently rather than develop one and refine it.

In evolutionary computation a model of a population of individuals is built where each individual is referred to as a chromosome. A chromosome defines the characteristics of individuals in the population. For each generation, individuals compete to reproduce offspring. The survival strength of an individual is measured by a fitness function. Those individuals with the best survival capabilities (fitness value) will have the best opportunity to reproduce. After each generation, individuals may undergo culling, or individuals may survive to the next generation (elitism). There are many types of evolutionary algorithms, among which the best known are 4 types (Engelbrecht, 2002):

Genetic Algorithm (GA)	Modeling genetic evolution
Genetic Programming (GP)	Based on GA, but individuals are programs
Evolutionary Programming (EP)	Derived from the simulation of adaptive behavior in evolution
Evolutionary Strategy (ES)	Geared toward modeling the strategic parameters that control variation in evolution

Table 2.1 Four different types of EC

The implicit parallel property gained by evolving a population of points in the search space concurrently suggests that EAs have a natural mapping onto parallel architectures.

2.3 Parallel Evolutionary Computation

According to Rivera (Rivera, 2001), there are four possible strategies to parallelize EAs, i.e., global parallelization, coarse-grained parallelization, fine-grained parallelization and hybrid parallelization.

In global parallelization, only the fitness evaluations of individuals are parallelized by assigning a fraction of the population to each processor. The genetic operators are often performed in the same manner as traditional EAs since these operators are not as time-consuming as the fitness evaluation. This strategy preserves the behavior of traditional EA and is particularly effective for problems with complicated fitness evaluations.

In coarse-grained parallelization, the entire population is partitioned into subpopulations. This strategy is more complex since it consists of multiple subpopulations and different subpopulations may exchange individuals occasionally (migration). In this parallel EAs model, the whole population is divided into multiple subpopulations, demes, that evolve on their own isolated from each other most of the time. This is also called isolated island model. This class of parallel EAs uses few relatively large demes. Each processor handles a subpopulation by itself. The subpopulations communicate through certain migrant individuals that are transferred from one to another subpopulation periodically, which is migration. The exchange of individuals is produced with low frequency. The migration of individuals from one deme to another is controlled by the topology that defines the connectivity between the subpopulations, by a migrate rate controlling the number of individuals to migrate, by a migration interval that affects the frequency of the migrations. Selection, mutation and crossover operations occur within a deme. Coarse-grained parallel EAs are more difficult to understand since the effects of migration are not fully understood. Often migration in coarse-grained parallel evolutionary algorithms is synchronous occurring at predetermined constant intervals. According to the migration structure chosen, it can increase either, the selection pressure, the diversity or also delay convergence. There is a critical migration rate. Below it, the performance of the algorithm is determined by the isolation of the demes. There are different migration strategies such as to choose emigrants and replace them randomly or alternatively according to fitness. Besides, this strategy introduces fundamental changes in the EA operations and has a different behavior than traditional EAs.

The fine-grained parallelization is often implemented on massively parallel machines, in which the population is divided into many and small demes. In the extreme case one can use a single large population with one individual per processor. Usually each processor controls one or a small amount of individuals and there is intensive communication between demes. The individuals belonging to the whole population are distributed topologically in a grid and are restricted to reproduce in a small environment of its location. Selection and mating are local with neighbors. A critical parameter is the ratio between the radius of the deme and the size of the underlying grid. The genetic operators take place in parallel only among neighborhood processors, and the individuals in each processor are replaced by the new offspring as new generations come out.

In hybrid parallelization, several parallelization approaches are combined, and the complexity of these hybrid parallel EAs depends on the level of hybridization.

2.4 Existing Paladin –DEC Package

The Distributed Evolutionary Computation package Paladin-DEC was first introduced by Tan (Tan, 2002) and had been applied to a case study of drug scheduling in cancer chemotherapy. The distributed implementation of evolutionary algorithms was extended from the coarse-grained parallel evolutionary algorithms with significant modifications, such as migration scheme, task scheduling and fault tolerant, so as to adapt to the features in distributed computing like variant communication overhead, unpredictable node crash and network restrictions. In Paladin-DEC implementation, the whole population is divided into n subpopulations. Each peer computer runs the combined algorithm on its own subpopulations. At each generation, peers run normal EA computation, including selection, crossover and mutation. After a period of time (migration interval), a number (migration rate) of good individuals will be selected and copies of them will be sent to one of its neighbors to perform migration. Every subpopulation also receives copies from its neighbors, which replaces its own low-fitness individuals. After migration next generations' evolutionary computation will go on. The Paladin-DEC package has shown good performance in work-load balancing, robustness, portability and security. Fig 2.2 shows the model of the Paladin-DEC package.

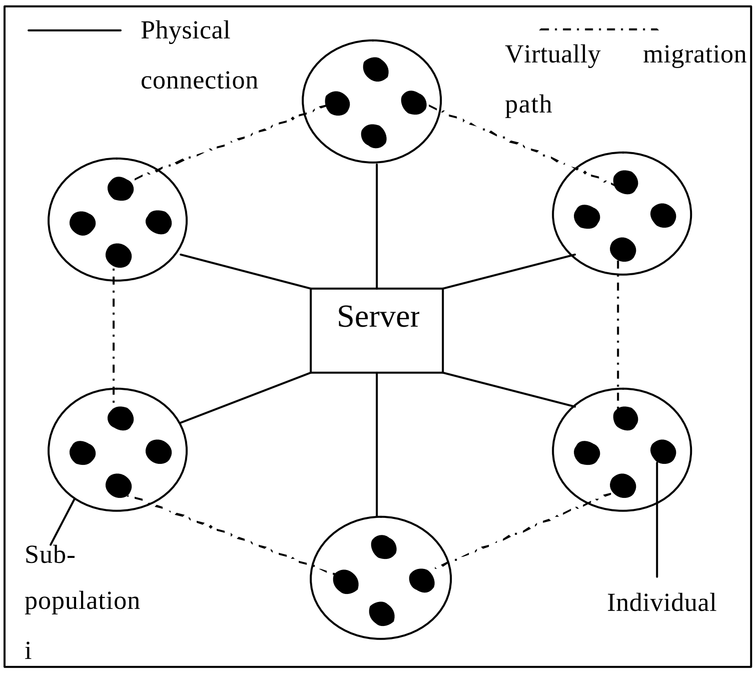


Fig 2.2 A model for distributed evolutionary computing

2.5 Updated Paladin –DES Package

The original version of Paladin was developed to address mainly the distributed genetic algorithm. In this project, the DES package is updated. Some parts are modified in the distributed evolutionary strategy package while the original framework still remains the same. In this section, the main characteristics of evolutionary strategy and how it is implemented in the DES package is discussed.

2.5.1 Evolutionary Strategy

Although both of the algorithms fall into evolutionary algorithms, evolutionary strategy has a big difference with genetic algorithm. Evolutionary Strategies (ES) are often presented and discussed as a technique competing with genetic algorithms. ES was developed to solve real-parameter optimization problem based upon one single genetic operator, i.e., mutation. In ES, a chromosome represents an individual as a pair of float-valued vectors, i.e. $\vec{v} = (\vec{x}, \vec{s})$. Here, the first vector $\vec{x}$ represents a point in the search space; the second vector $\vec{s}$ is a vector of standard deviations. The mutations are realized by replacing $\vec{x}$ by $\vec{x}^{i+1} = \vec{x}^{i} + N(0, \vec{s})$, where $N(0, \vec{s})$ is a vector of independent random Gaussian numbers with a mean of zero and standard deviation $\vec{s}$. The offspring is accepted as a new member of the population if and only if it has better fitness and all constraints are satisfied. The main idea behind these strategies is to allow control parameters to self-adapt rather than changing their values by some deterministic algorithm.

As the original package concentrates on the Genetic Algorithm, to implement the distributed evolutionary strategy package it is essential to clarify the difference between the two algorithms. Table 2.2 lists out the seven most important differences.

Genetic algorithms	Evolutionary strategies
Genotype level of individuals (binary coding)	Phenotype level of individuals (real-value representation)
No knowledge about the objective functions' properties	Knowledge of the dimension of the objective function (i.e. number of variables)
Parameter space restrictions for coding purpose	No parameter restricts apart from machine-dependencies
Dynamic, preservative or static, preservative selection	Static, extinctive selection (equal probabilities); more or less selective
Recombination servers as the main search operator	Mutation servers as the main search operator
Secondary role of mutation	Different recombination schemes
No collective self-learning of parameter settings	Collective self-learning of strategy parameters

Table 2.2 Difference between GA and ES

2.5.2 Updated Paladin-DES Design

Inheriting from the original framework, the updated version also has 4 main parts: Database, server, client and controller. The server part and the database remain the same as the old version, so as the connection between the clients and the server. It continues using the Java-based Remote Method Invocation over Internet Inter-ORB Protocol. (RMI-IIOP)

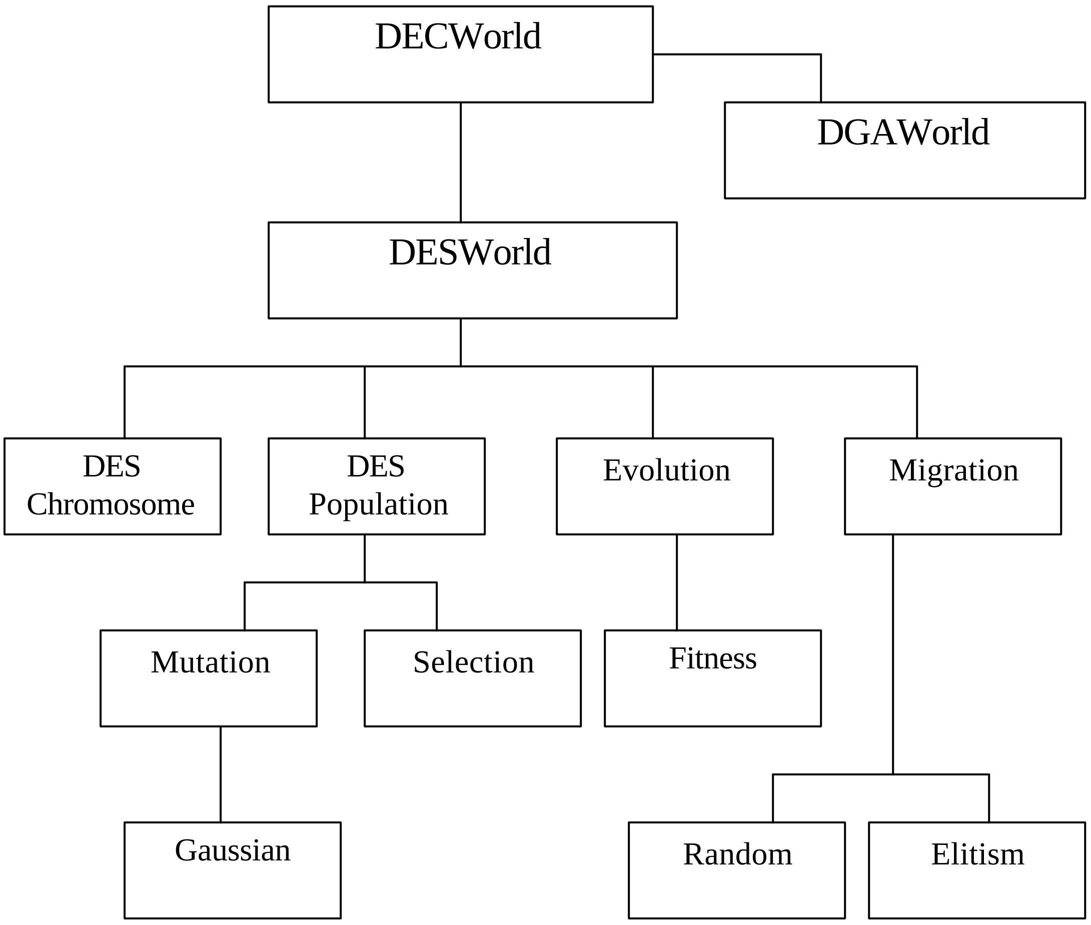


Fig 2.3 Class hierarchy of Distributed Evolutionary Strategy

As can be seen from Fig 2.3, the client DES class hierarchy doesn't contain the crossover computation, since mutation is the only search operator in ES. However, a new fitness sharing scheme has been involved. This scheme is an improvement in the new version of the package. The function of the fitness sharing method is to compare the best individuals in a sub-population, if some of them have much higher fitness values than others, their fitness values will be shared to ensure global optimum to be found instead of local convergence. Fig 2.4 shows the UML of the DESWorld class.

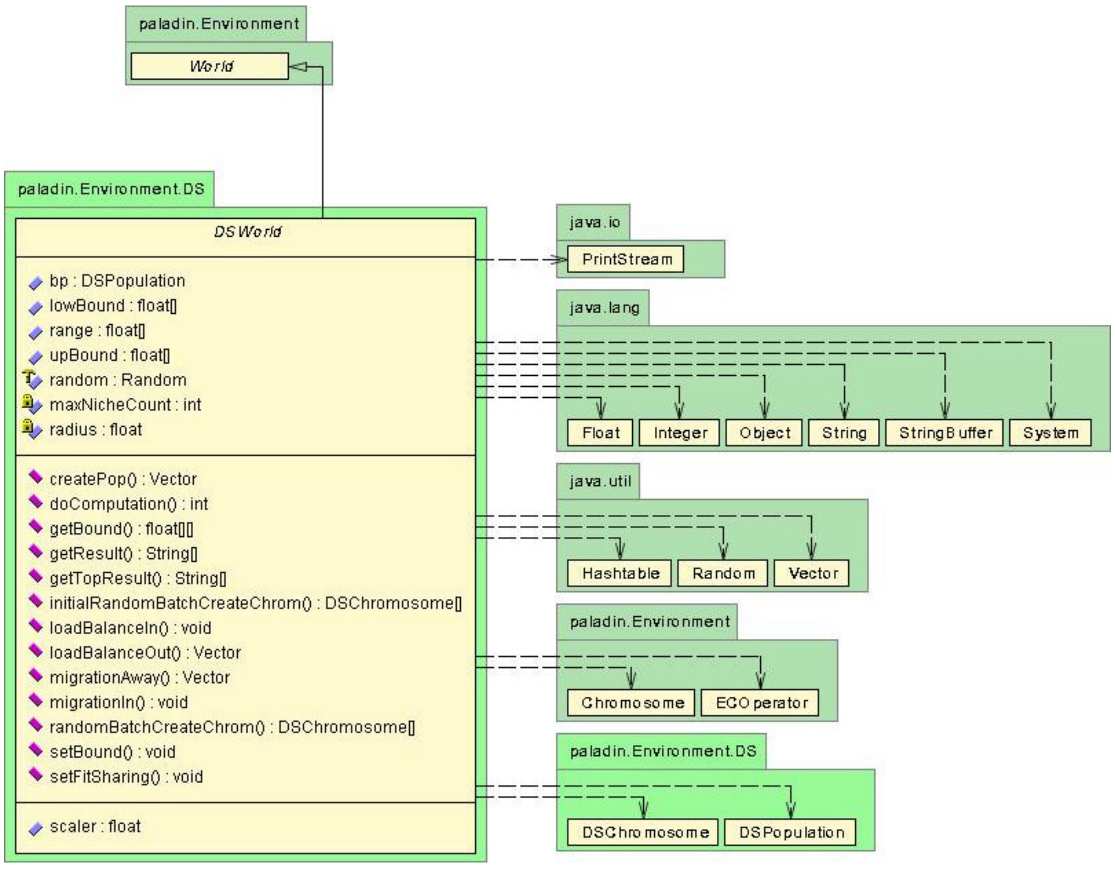


Fig 2.4 UML of DSWorld

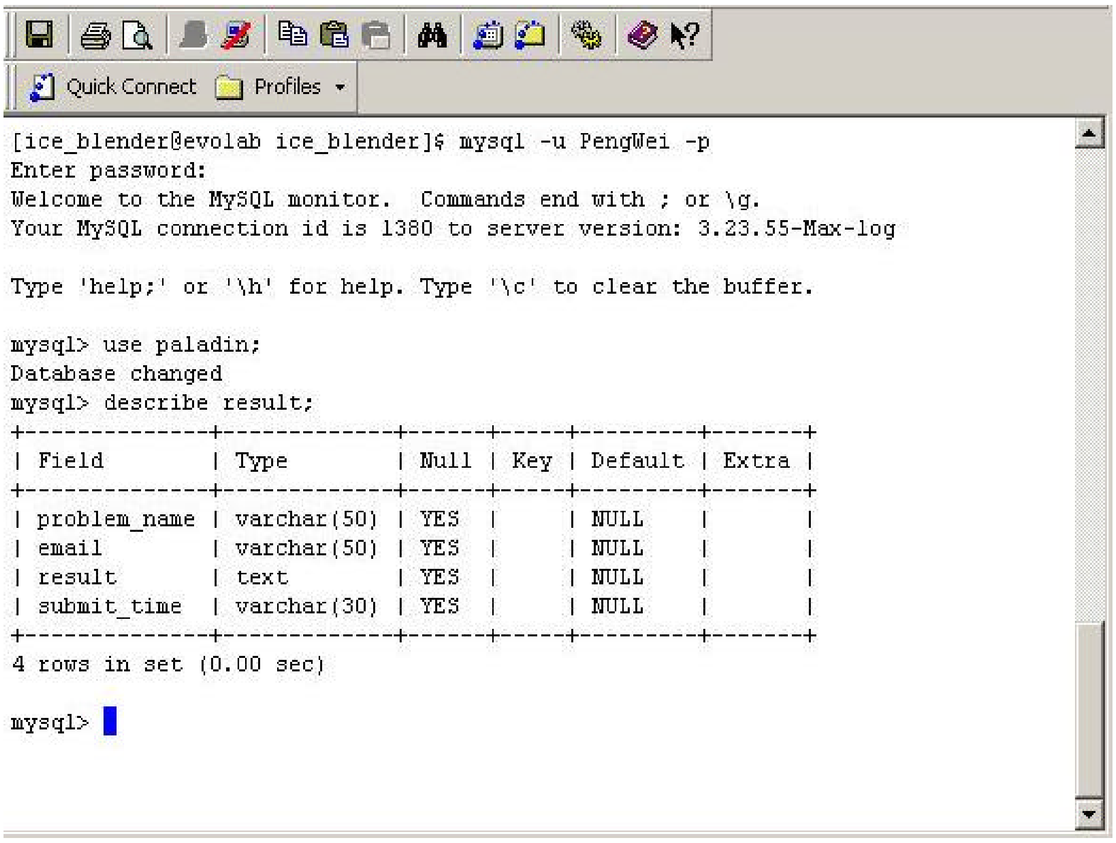
The package is developed in JAVA language based on the latest J2EE technology with JBuilder software. Java Remote Method Invocation over Internet Inter-ORB Protocol technology ("RMI-IIOP") is part of the Java 2 Platform, Standard Edition (J2SE^{TM}). The RMI Programming Model enables the programming of Common Object Request Broker Architecture (CORBA) servers and applications via the rmi API. RMI-IIOP utilizes the Java CORBA Object Request Broker (ORB) and IIOP, so one can write all his own codes in the Java programming language, and use the rmic compiler to generate the code necessary for connecting the applications via the Internet InterORB Protocol (IIOP) to others written in any CORBA-compliant language.

2.5.3 Updated Paladin-DES Implementation

The updated version has 4 main parts: database, server, client and controller.

2.5.3.1 Database

All the final simulation results are stored in the database. Besides storing the final results, the database is also used for peer computers to exchange some intermediate calculation outcomes which are needed perform migration after a period time of migration interval. The database is built on MySQL database technology.



```
[ice_blender@evolab ice_blender]$ mysql -u PengWei -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1380 to server version: 3.23.55-Max-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use paladin;
Database changed
mysql> describe result;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| problem_name | varchar(50) | YES  |     | NULL    |       |
| email        | varchar(50) | YES  |     | NULL    |       |
| result       | text        | YES  |     | NULL    |       |
| submit_time  | varchar(30) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql>
```

Fig 2.5 MySQL Database table description

2.5.3.2 Server

The server is built on a powerful computer with RedHat Linux operating system. It consists of three main functions: Logon server, Resource server and Reception server.

The logon server monitors how many peer computers have logged on the system and can be used to carry out an ES job. All the peer computers logon to the server through a valid email address. One unique and valid email address can only register one peer. In the list of logged on computers, once any email address appear again, the previous logon information is removed while the latest information is updated.

The main usage of the resource server is to manage job files transfer, peer synchronization and agent assigning.

The reception server is responsible for assigning ES parameters, job scheduling and work load balancing, inspecting migrations, final result submission to the database and monitoring the overall ES job performance. As the reception server acts as the main part in server functioning, Table 2.3 shows the methods which are defined in the reception server class and their main operations.

Method Name	Operation performed
getPeerInfo	Get peer computers information, including email address, operating system, memory size and ping value to the server.
getJobInfo	Obtain the normal EA parameters from the class files.
assignJobTo	According to the internal scheduling scheme, assign job to some or all the peer computers logged on the server.
checkJob	Check from the controller whether the job class file needs an agent or not.
cancelJob	Cancel the job from all the peers who have been assigned. Restore the server log on information.
checkPoint	After a migration interval, check the overall computation performance, perform load balancing and get ready for performing migration.
removePeer	Remove the idle peers from servers' logon list. It may be caused by hang of peer computer or other interference.
performMig	Perform migration.
checkFinish	Check whether the terminal condition has been matched.
getBestResult	From all the result submitted to the server, choose the best one.
resultSubmit	Submit the final result to the database.
sendMail	Email the final result to the user who submits the problem class file.

Table 2.3 Main functions defined in the reception server

To accomplish the distributed work, the server part of the DES package involves the latest J2EE Portable Object Adapter technology. An object adapter is the mechanism that connects a request using an object reference with the proper code to service that request. The Portable Object Adapter, or POA, is a particular type of object adapter that is defined by the CORBA specification.

The POA is designed to meet the following goals:

- Allow programmers to construct object implementations that are portable between different ORB products.
- Provide support for objects with persistent identities.
- Provide support for transparent activation of objects.
- Allow a single servant to support multiple object identities simultaneously.

Normal creating and using POA involves 6 steps:

(1) Get the root POA

```
ORB orb = ORB.init( args, null );
POA rootPOA =
POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
```

(2) Create a POA and define the appropriate policies

```
Policy[] tpolicy = new Policy[3];
tpolicy[0] = rootPOA.create_lifespan_policy(
        LifespanPolicyValue.TRANSIENT );
tpolicy[1] = rootPOA.create_request_processing_policy(

RequestProcessingPolicyValue.USE_ACTIVE_OBJECT_MAP_ONLY
 );

tpolicy[2] = rootPOA.create_servant_retention_policy(
        ServantRetentionPolicyValue.RETAIN);

POA tPOA = rootPOA.create_POA("MyTransientPOA", null,
tpolicy);
```

(3) Activate the POA Manager; otherwise all calls to the servant hang because, by default, POAManager will be in the HOLD state.

```
tPOA.the_POAManager().activate();
```

(4) Instantiate the Servant and activate the Tie

```
logonServer logon = new logonServer();
_logonServer_Tie tie1=
(_logonServer_Tie)Util.getTie( logon );
String logOnId = "logonServer";
byte[] id1= logOnId.getBytes();
tPOA.activate_object_with_id( id1, tie1);
```

(5) Publish the object reference using the same object id used to activate the Tie object.

```
Context initialNamingContext = new InitialContext();

initialNamingContext.rebind(messageTag.logonService,
tPOA.create_reference_with_id(id1,
            tie1._all_interfaces(tPOA,id1)[0]) );
System.out.println("Logon Server: Ready...");
```

(6) Get ready to accept requests from the client

```
orb.run();
```

2.5.3.3 Clients/Peers

The linkage between the server and the client inherits the older version of Paladin-DEC, using the Java-based Remote Method Invocation over Internet Inter-ORB Protocol (RMI-IIOP). Normal client peers'working flowchart is shown in Fig 2.6.

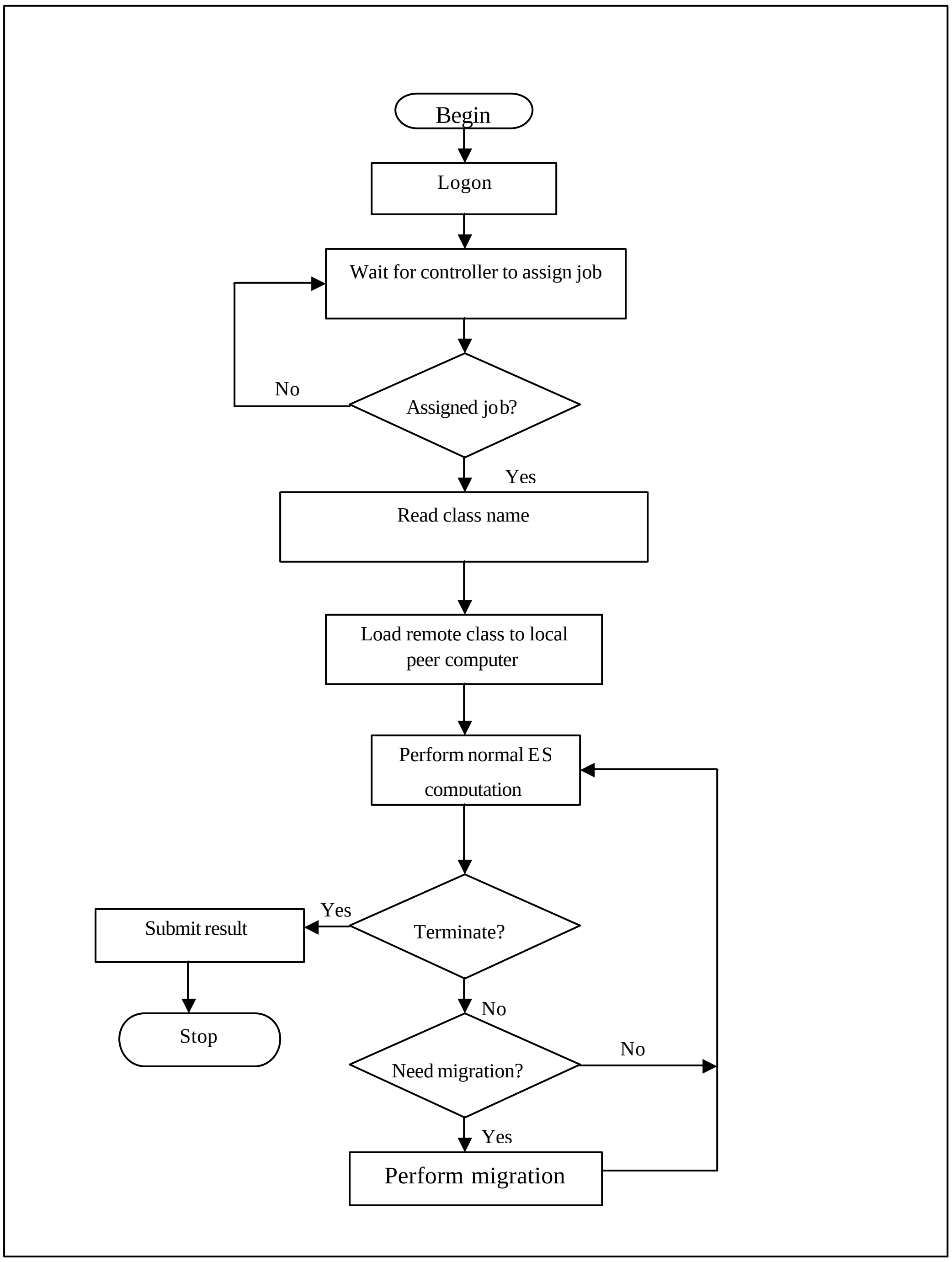
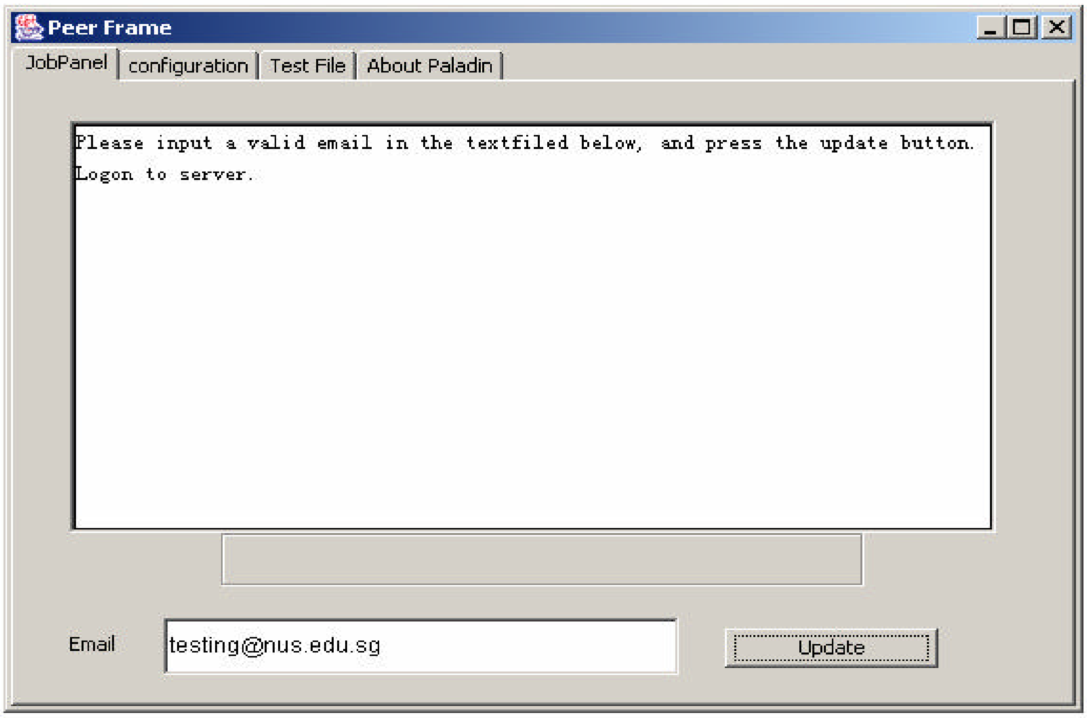


Fig 2.6 Working flowcharts of normal clients

There are two working modes for clients in the updated Paladin-DES package. One is normal working mode; the other is agent-working mode. The difference is that the second method needs an agent to manage data transfer from client to server. The normal client working process begins when a client is started and logon to the server. A valid peer is uniquely identified by its email address. The logon server will check the email address whether have been present in its list and give a response of valid logon or not. After logging on the server, the client is idle and waiting for the controller to assign it an ES job. Fig 2.7 shows the peer computer logon GUI.

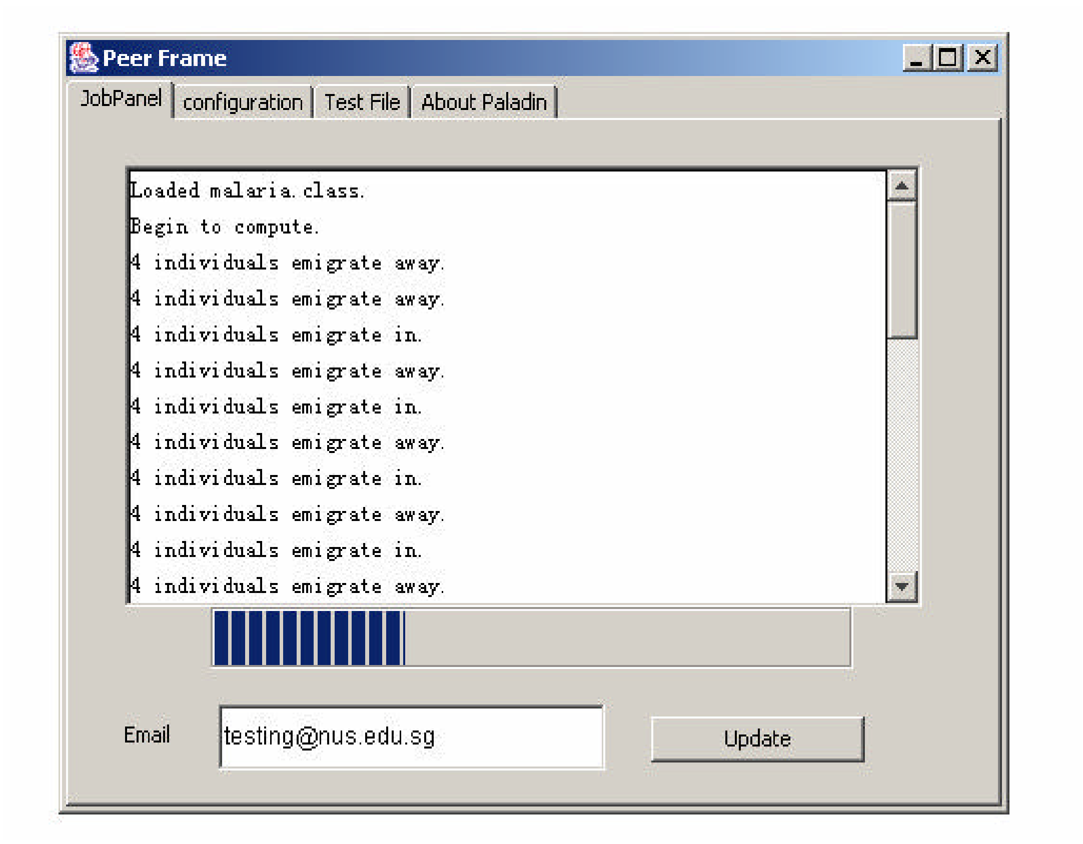


```
Please input a valid email in the textfiled below, and press the update button.
Logon to server.
```

Email: testing@nus.edu.sg

Fig 2.7 Peer computer logon GUI

After getting a job command, it first reads the class name from the controller, and then loads the class from remote resource server to the local peer machine through http. Thereafter it retrieves the ES working parameters from the reception server, and begins to perform normal ES calculation according to the schedule retrieved from reception server. After migration interval, it performs migration if needed. Fig 2.8 shows the working GUI of normal peers.

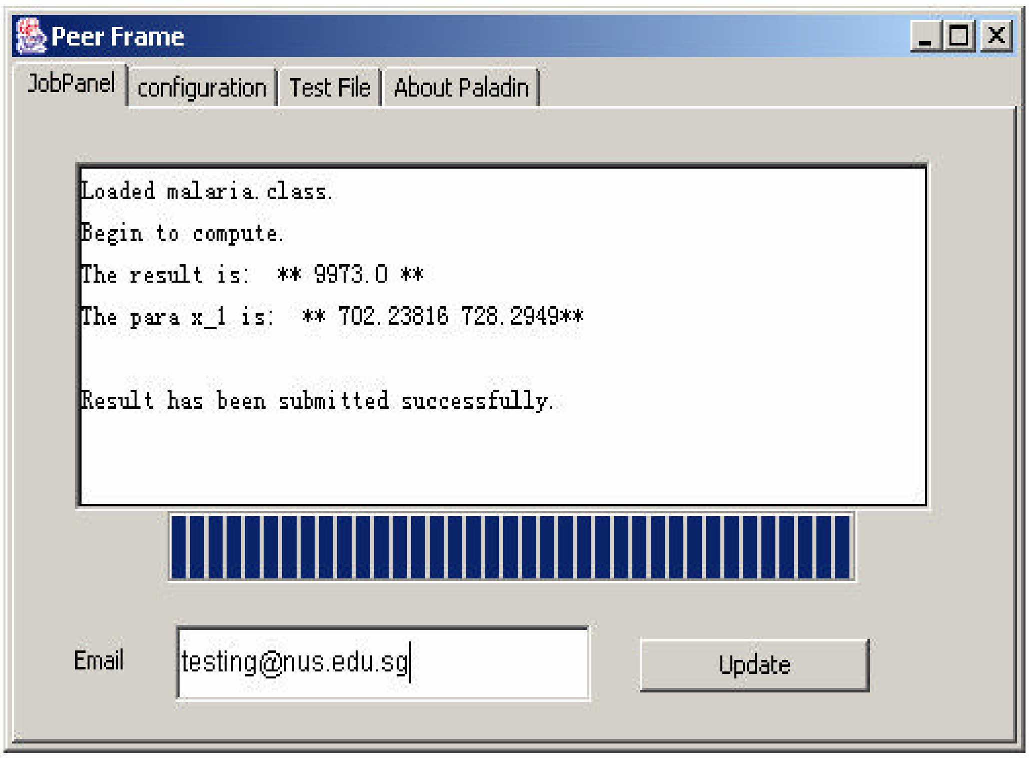


```
Loaded malaria.class.
Begin to compute.
4 individuals emigrate away.
4 individuals emigrate away.
4 individuals emigrate in.
4 individuals emigrate away.
4 individuals emigrate in.
4 individuals emigrate away.
4 individuals emigrate in.
4 individuals emigrate away.
4 individuals emigrate in.
4 individuals emigrate away.
```

Email testing@nus.edu.sg

Fig 2.8 Peers working GUI

When the terminal condition matched, it will submit the results to the reception server and finally the reception server first store the results to the database and then email the user who submits the problem class file the final result. Fig 2.9 shows the GUI where peer computer finishes computation and reports the best individual to the server.



```
Loaded malaria.class.
Begin to compute.
The result is:  ** 9973.0 **
The para x_1 is:  ** 702.23816 728.2949**

Result has been submitted successfully.
```

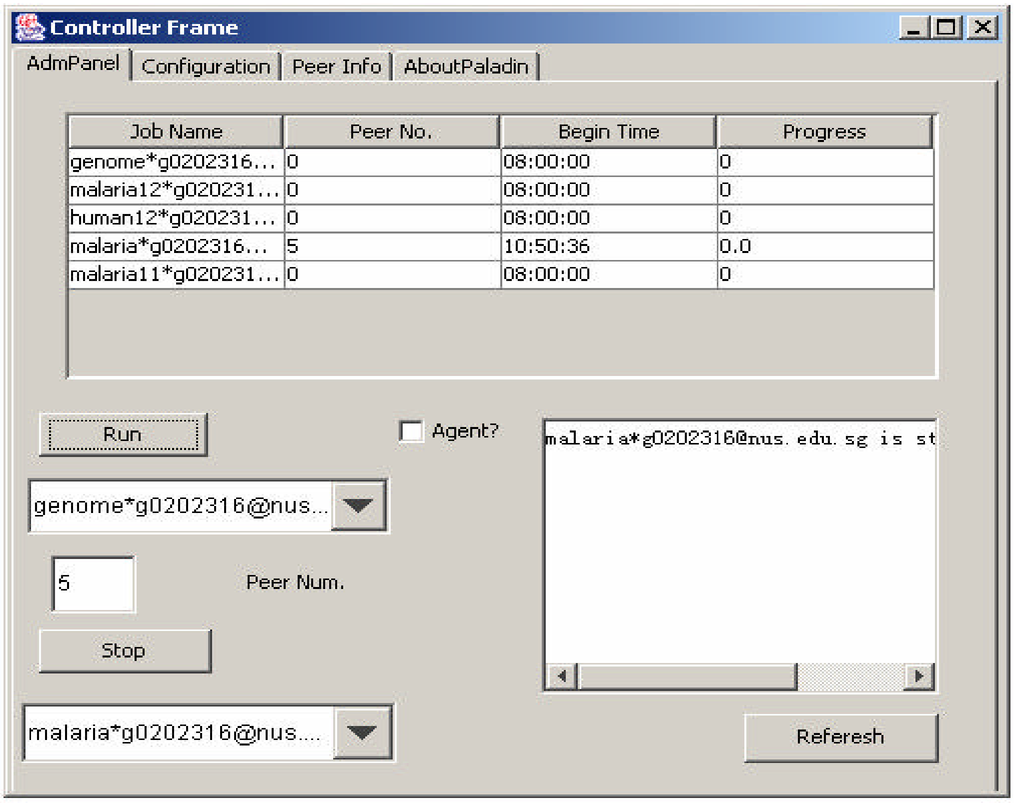
testing@nus.edu.sg

Fig 2.9 Peers finishes working GUI

In agent-working mode, one peer is assigned as an agent according to the resource servers' criteria. This peer will not participate in any E S computation; it will be used as an intermediate node for data transfer, including sending problem file to peers, storing migration individuals for peers to exchange, submitting to server the results obtained from peers, etc. It is the only peer computer which directly handshakes to the server during computation. Other peers, now migration or submitting results, they only need to communicate to the agent peer instead of talking to the server directly. This will reduce the overhead time when more peers are connected to perform the computation.

2.5.3.4 Controller

The controller of the package plays an important surveillance role. It monitors the whole process of the ES problem computation. Fig 2.10 shows the user control panel of the controller. When the controller starts, it first checks the status of the server. If the server operates normally, the controller will display all the job files present on the resource server for peers to download. After user determines the problem file, the number of working peers and whether to use agent or not, the controller will initialize an instance of reception server to perform inspection on the work flowing, including job scheduling, migration process, work load balancing until the final result submission.



Job Name	Peer No.	Begin Time	Progress
genome*g0202316...	0	08:00:00	0
malaria12*g020231...	0	08:00:00	0
human12*g020231...	0	08:00:00	0
malaria*g0202316...	5	10:50:36	0.0
malaria11*g020231...	0	08:00:00	0

```
malaria*g0202316@nus.edu.sg is st
```

genome*g0202316@nus...

Peer Num. 5

malaria*g0202316@nus....

Fig 2.10 Controller GUI

2.6 Conclusion

In this chapter the basic understanding of computational intelligence was presented and then the concept was narrowed down to the project work, evolutionary computation and hence evolutionary strategy. The underlying theory of evolutionary strategy and parallel computation were discussed in details. After that the design and the implementation of the Distributed Evolutionary Strategy package were shown specifically, including the technology involved – JAVA, J2EE, CORBA- and each one of the four parts of the package.

Chapter 3

Bioinformatics Basics

3.1 Introduction

In the last few decades, advances in molecular biology and the equipment available for research in this field have allowed the increasingly rapid sequencing of large portions of the genomes of several species. In fact, to date, several bacterial genomes, as well as those of some simple eukaryotes (e.g., Saccharomyces cerevisiae, or baker's yeast) and more complex eukaryotes (C. elegans and Drosophila) have been sequenced in full. The Human Genome Project, designed to sequence all 24 of the human chromosomes, is also progressing. Popular sequence databases, such as GenBank and EMBL, have been growing at exponential rates. This deluge of information has necessitated the careful storage, organization and indexing of sequence information. Information science has been applied to biology to produce the field called bioinformatics (NCBI Education).

Bioinformatics is conceptualizing biology in terms of molecules and then applying informatics techniques which derived from disciplines such as applied mathematics, computer science, artificial intelligence and statistics to understand and organize the information associated with these molecules, on a large scale. It is the recording, annotation, storage, analysis, and searching/retrieval of nucleic acid sequence (genes and RNAs), protein sequence and structural information. This includes databases of the sequences and structural information as well methods to access, search, visualize and retrieve the information.

Bioinformatics is the field of science in which biology, computer science, and information technology merge into a single discipline. The ultimate goal of the field is to enable the discovery of new biological insights as well as to create a global perspective from which unifying principles in biology can be discerned. There are three important sub-disciplines within bioinformatics involving computational biology:

- The development of new algorithms and statistics with which to assess relationships among members of large data sets;
- The analysis and interpretation of various types of data including nucleotide and amino acid sequences, protein domains, and protein structures; and
- The development and implementation of tools that enable efficient access and management of different types of information.

The most pressing tasks in bioinformatics involve the analysis of sequence information. Computational Biology is the name given to this process, and it involves the following:

- Finding the genes in the DNA sequences of various organisms
- Developing methods to predict the structure and/or function of newly discovered proteins and structural RNA sequences.
- Clustering protein sequences into families of related sequences and the development of protein models.
- Aligning similar proteins and generating phylogenetic trees to examine evolutionary relationships.

The objective of this project is to apply the computational intelligence technique - distributed evolutionary strategy - to search oligo sets (probes) of malaria parasite, *Plasmodium Falciparum* from the nucleotide gene coding sequences. The oligo sets found are to be printed on the microarray for subsequent biology and medical research.

In this chapter the genetic information transfer inside the cell will be firstly presented. Subsequently the microarray technology will be introduced.

3.2 Genetic Information Transfer within cells

This project used the distributed computational intelligence technique, Paladin-DES package to search the malaria parasites' coding sequence file for the qualified oligo sets (probes) to be printed on the microarray. Therefore knowing the genetic information transfer from original DNA to the final pre-translation coding sequence is essential.

As it is well known, DNA exists as a right-handed double helix in which two polynucleotide chains are coiled about one another in a spiral. DNA sequences consist of only four different types of alphabet letters, or 4 bases: Adenine (A), Cytosine (C), Guanine (G), and Thymine (T). They form pairs as AT and GC where AT bases pair is held together with 2 hydrogen bonds while GC pairs have 3.

Although DNA is the source of all the genetic information, it is protein, which contains constituent called amino acids, that finally governs the functionality of the growth and development of an organism. According to central dogma of molecular biology, the transfer of genetic information from DNA to protein during its phenotypic expression in an organism involves two steps. The first step is transcription. It is information transfer from DNA to RNA. The second step is the transfer of information from RNA to protein, this step is called translation translation (Snustad, Simmons and Jenkins, 1997).

During transcription, one strand of DNA of a gene is used as a template to synthesize a complementary strand of RNA called the gene transcript. During translation, the sequence of nucleotides in the RNA transcript is converted into the sequence of amino acids in the polypeptide gene product according to the genetic code. Translation takes place on intricate macromolecular machines called ribosomes which are composed of three to five RNA molecules and 50 to 90 different proteins. The RNA molecules that are translated on ribosomes are called messenger RNAs (mRNA).

3.2.1 Transcription

In eukaryote, where *Plasmodium Falciparums*' cell belongs to, primary transcripts usually are precursors to mRNA and are called pre-mRNAs. Not the whole eukaryote pre-mRNA sequence can be encoded into protein. Inside the pre-mRNA sequence, some are noncoding sequence called introns that separate the coding sequences or exons of these genes. The entire sequence of these split genes are transcribed into pre-mRNA, and the noncoding sequences are subsequently removed by spicing reaction.

3.2.2 Translation

Inside the mRNA, the sequence still exists in the sense of the four bases. After translation, which controlled by the genetic code, every connective three nucleotide bases are translated into one amino acid. Translation is the process of matching amino acids to corresponding sets of three bases (codons) and linking them into a protein. The process of translation is governed by the genetic code. The mystery of genetic code was largely disclosed by the mid-1960s. The genetic code has some very important properties:

- The genetic code is composed of nucleotide triplets.
- The genetic code is nonoverlapping. Each nucleotide in mRNA belongs to just one codon.
- The genetic code is ordered. Multiple codons for a given amino acid and codons for amino acids with similar chemical property are closely related.
- The genetic code is universal. With minor exceptions, the codons have the same meaning in all living organisms, from viruses to humans.

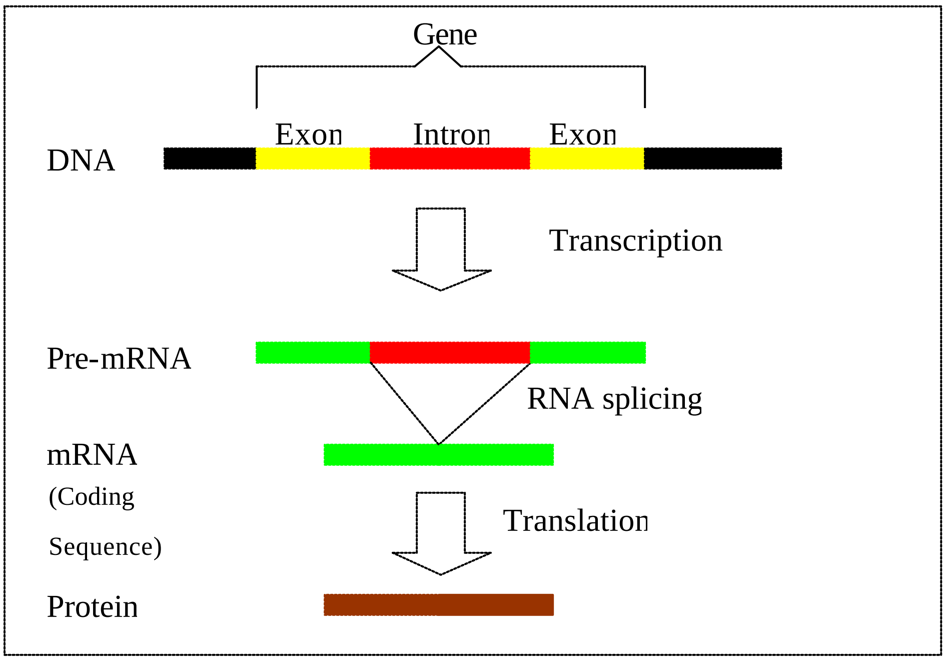


Fig 3.1 Two steps of genetic information transfer from DNA to protein

Fig 3.1 shows the 2 steps genetic information transfer from DNA to protein. The coding sequence (cds) file of malaria parasite, *Plasmodium Falciparum* has been published by *Nature* in 2002. The cds file provides the sequences inside which all the introns have been spliced out and all the exons have been combined together and ready to encode proteins. The target of this project is to find qualified probes of these gene sequences from this coding sequence file to be printed on the DNA microarray. This file contains all useful genetic information in *Plasmodium Falciparum* and nothing else. Therefore probes found from this file are the most useful for the subsequent biology and medical research.

3.3 DNA Microarray

3.3.1 Background

It is widely believed that thousands of genes and their products (i.e., RNA and proteins) in a given living organism function in a complicated and orchestrated way that creates the mystery of life. However, traditional methods in molecular biology generally work on a "one gene in one experiment" basis, which means that the throughput is very limited and the "whole picture" of gene function is hard to obtain.

In the past several years, a new technology, called DNA microarray, has attracted tremendous interests among biologists. The primary applications of microarrays are the study of differential gene expression and gene mapping. This technology promises to monitor the whole genome on a single chip so that it enables the simultaneous analysis of thousands of sequences of DNA for genetic and genomic research and for diagnostics.

The microarray technology is having a significant impact on genomics study. It makes use of the sequence resources created by the genome projects and other sequencing efforts to answer the question, what genes are expressed in a particular cell type of an organism, at a particular time, under particular conditions. Many fields, including drug discovery and toxicological research, have already and will benefit from the use of DNA microarray technology. Fig 3.2 shows an illuminated microarray.

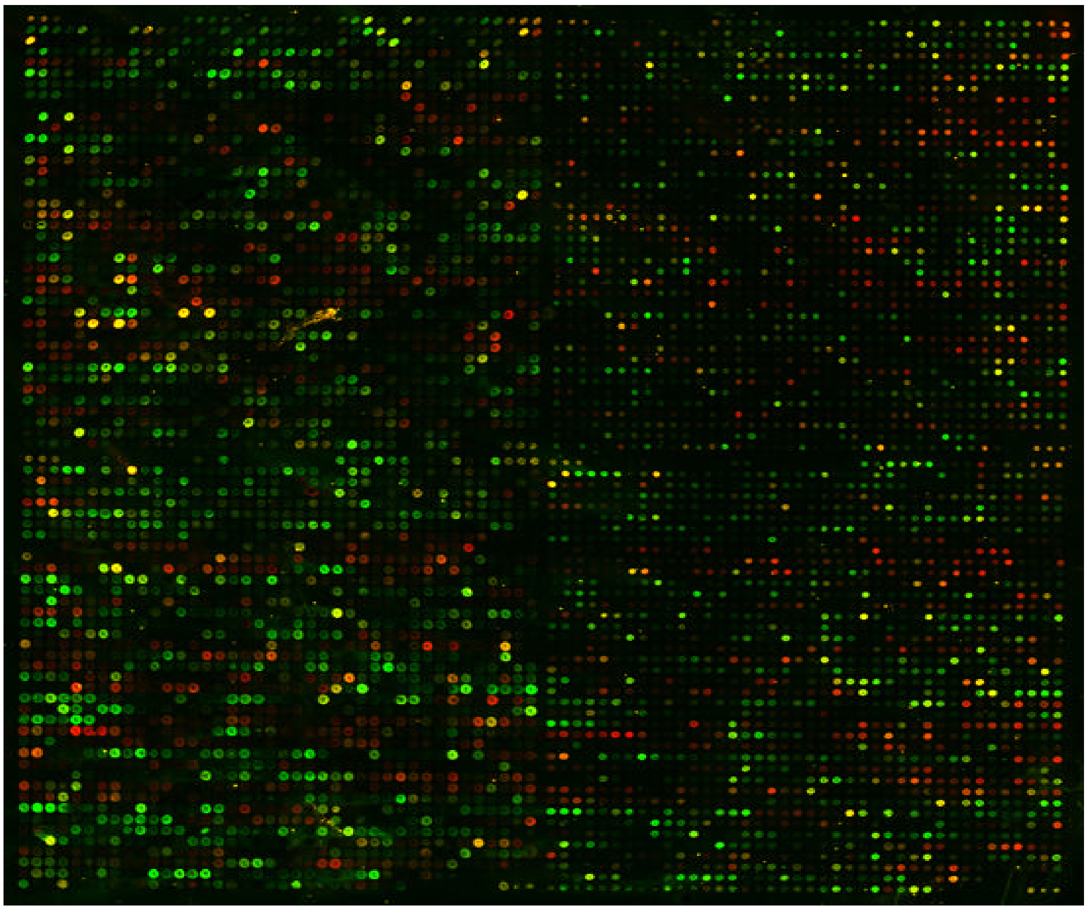


Fig 3.2 An illuminated microarray

There are different ways how microarrays can be used to measure the gene expression levels. One of the most popular microarray applications allows comparing gene expression levels in two different samples, e.g., the same cell type in a healthy and diseased state (see Fig 3.3).

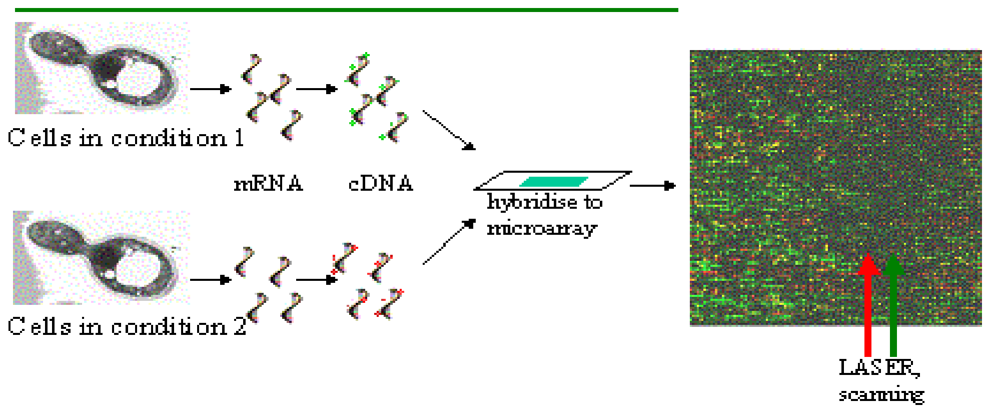


Fig 3.3 Comparing the same cell type in a healthy and diseased state

3.3.2 Microarray Fabrication and Experiment

DNA microarrays are also referred to as DNA arrays, microarrays, DNA chips, biochips or GeneChips. Microarrays exploit the preferential binding of complementary single-stranded nucleic acid sequences. A microarray is typically a glass (or some other material) slide, on to which DNA molecules are attached at fixed locations.

According to Young (Young, 2000), a DNA microarray is an orderly arrangement of samples, which provide a medium for marching known and unknown DNA samples based on base-pairing rules (A-T and C-G) and thus automating the process to identifying the unknowns. Several steps are required to conduct a microarray experiment. Firstly, the probe samples are to be synthesized. A probe is a subsequence of a gene coding sequence, which means a part of chromosome that is related to protein synthesis, which can represent the whole gene. Next, the synthesized probes of thousands of gene sequences are automatically dotted on the chip to make an array. Upon setting up of the array, both dyed samples of test issues and control tissues are added on the probe dots. Since the samples constitute of RNAs complimentary to all gene sequences, they will react to the relative probes on the array. From measuring the reaction strength on each dot, the respective RNA amount in the samples can be determined, and thus the researcher can find which gene sequence is more active. From this method, the relationship of gene sequence and specified disease can be found, and many other biomedical problems can be solved using similar techniques with the microarray (DeRisi, 1997; Ren, 2000) Fig 3.4 shows a general overview of the DNA microarray experiment.

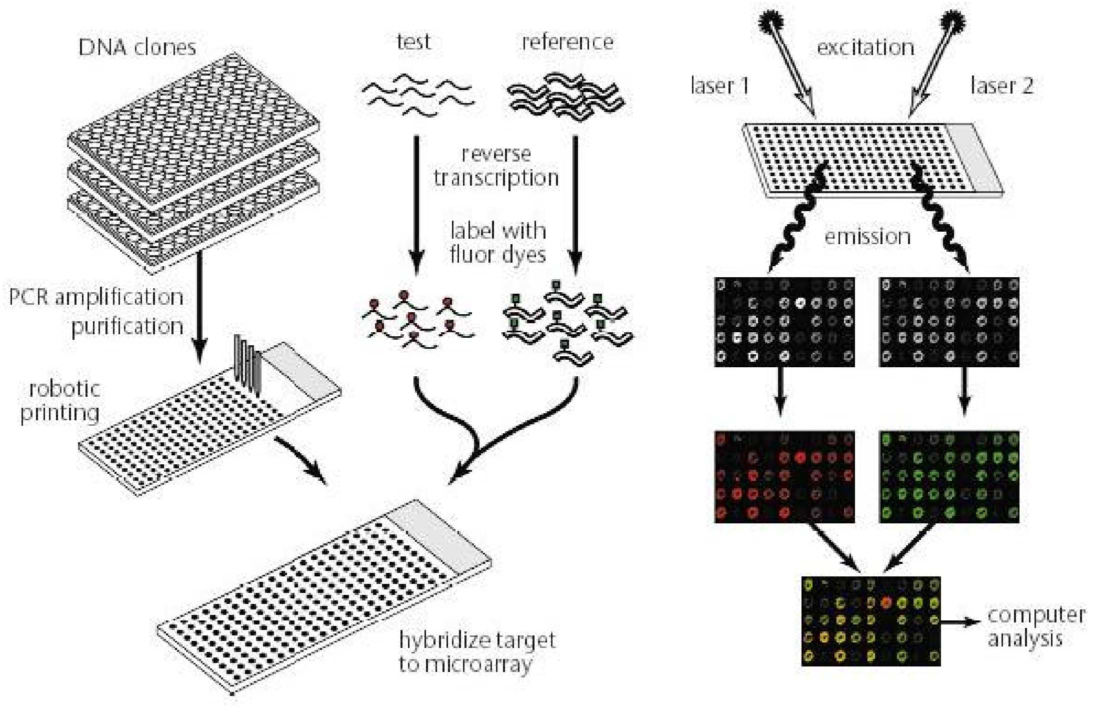


Fig 3.4 A general overview of the DNA microarray experiment

3.3.3 Preparation for the Probes

Many current DNA microarray protocols utilize double-stranded PCR (Polymerase Chain Reaction) products spanning the entire gene sequences as DNA probes immobilized onto glass slides (Bosch, 2000). Fig 3.4 above illustrates the steps needed to generate PCR-based microarrays. PCR is a technique to produce copies of a DNA strand, using the original DNA as a template. To generate full-length gene sequences, PCR requires the complete complementary DNA (cDNA) library as a template to amplify. The amplification of all genes in a genome can tale a long time, e.g. 4 months to amplify more than 6000 genes by a small group (DeRisi, 1997). Furthermore, the PCR products may fail to be verified, with a failure rate of 5-10% (DeRisi, 1997).

For gene expression studies, RNA is typically reverse transcribed to give complementary DNA. Upon denaturation of both RNA and the immobilized DNA, the mixture is allowed to hybridize. After hybridization and washing, microarrays may be monitored as temperature is increased. The temperature at which 50% of a single stranded DNA annealed with its complement to form a perfect duplex is defined as melting temperature.

3.3.4 Criteria in Searching Probes

Based on the requirements in using the microarray, a set of criteria should be fulfilled when searching the probes. The detailed criteria can be varied with different application. In the Paladin-DES package proposed, constraints can always be added as required very easily. For this current work, three very essential criteria are suggested. They are:

(1) Uniqueness criterion: each probe should identify one and only one gene.

(2) Melting temperature criterion: the melting temperature of the probe should be within a range to perform hybridization. Among various measuring methods of melting temperature, the following formula proposed by Breslauer (Breslauer, 1986) is more accurate.

$$T_m(x) = \frac{H(x)}{S(x) + R * \ln(C/4)} + 16.6 * \log \frac{[K+]}{1 + 0.7[K+]} - 273.15$$

where

$T_m(x)$: melting temperature of given probe x

H(x): Enthalpy for helix information of x

S(x): Entropy for helix information of x

R: molar gas constant (1.987 cal/^{o}C mol)

C: concentration of the probe (set as 250pmol)

[K+]: salt concentration (set as 50mmol)

(3) Non self-folding criterion: the qualified probe should avoid self-folding.

All of the three criteria will be discussed in details in the next chapter.

3.4 Conclusion

In this chapter firstly the basic concept of bioinformatics has been presented in details. This project used the Paladin-DES package to search the malaria parasite, *Plasmodium Falciparum's* coding sequence file for the qualified oligo sets (probes) to print on the microarray. Therefore knowing the genetic information transfer from original DNA to the final pre-translation coding sequence is very important. The two steps involved in genetic information transfer from DNA to protein-transcription and translation- are reviewed. Finally the latest biotechnology, microarray technology was also discussed and three criteria were suggested in searching the probes.

Chapter 4

Case Study: Searching Oligo Sets of Malaria Parasite, *Plasmodium Falciparum*

4.1 Introduction

In previous two chapters the distributed computational intelligence technique, Paladin-DES package, and some basic bioinformatics have been discussed. This Paladin-DES package has been applied to a complex bioinformatics problem: searching the oligo sets of malaria parasite, *Plasmodium Falciparum*. Simulation results show that the Paladin-DES package performs better than some existing searching techniques and demonstrates its capability. This chapter describes the problem formulation in details, including the *plasmodium* species, ES fitness functions and the three important searching criteria.

4.2 Problem Formulation

4.2.1 Malaria Parasite *Plasmodium Falciparum*

Approximately 40% of the world's population lives in areas where malaria is transmitted. Each year there are an estimated 300–500 million new malaria infections and 1–3 million deaths caused by the disease (Hoffman, 2002). The mortality levels are greatest in sub-Saharan Africa, where children under 5 years of age account for 90% of all deaths due to malaria (Breman, 2001). Four species of Plasmodium infect humans and cause malaria. All species are vector borne diseases, being spread by anopheline mosquitoes, and the disease is distributed throughout much of the world. Fig 4.1 shows the distribution.

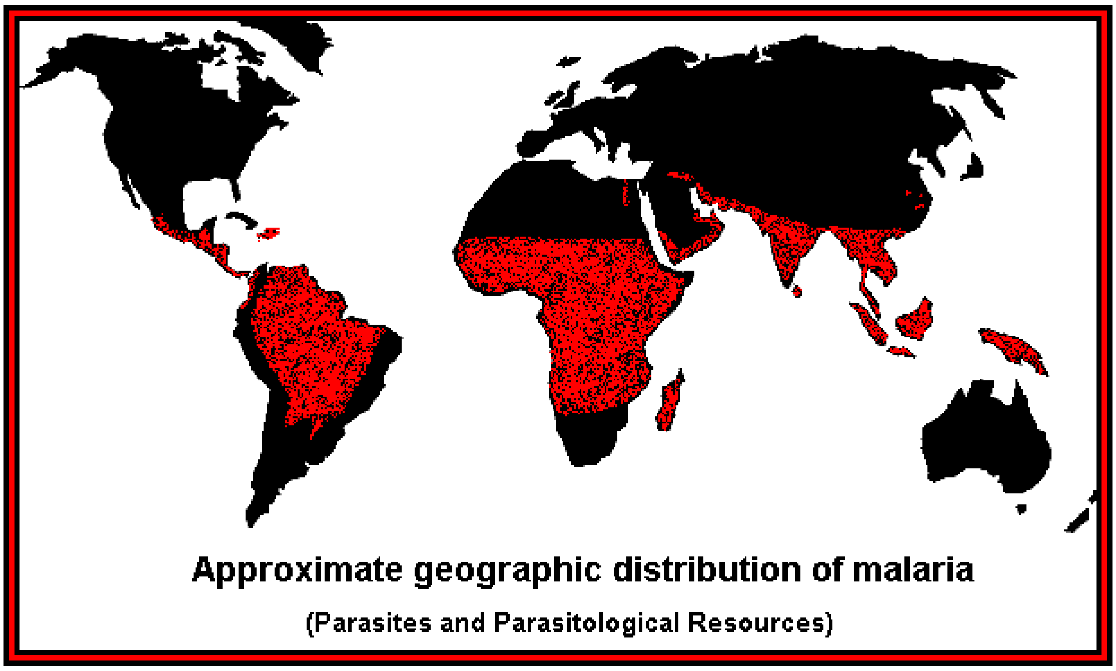

Approximate geographic distribution of malaria
(Parasites and Parasitological Resources)

Fig 4.1 Approximate geographic distribution of malaria

There are four species of *Plasmodium*: *Plasmodium vivax*, *Plasmodium falciparum, Plasmodium ovale* and *Plasmodium malariae. Plasmodium vivax* is the most extensively distributed and causes much debilitating disease. *Plasmodium falciparum*, which is also widely spread, results in the most severe infections and is responsible for nearly all malaria-related deaths. *Plasmodium ovale* which is mainly confined to Africa is less prevalent, while *Plasmodium malariae*, which causes the least severe but most persistent infections, also occurs widely.

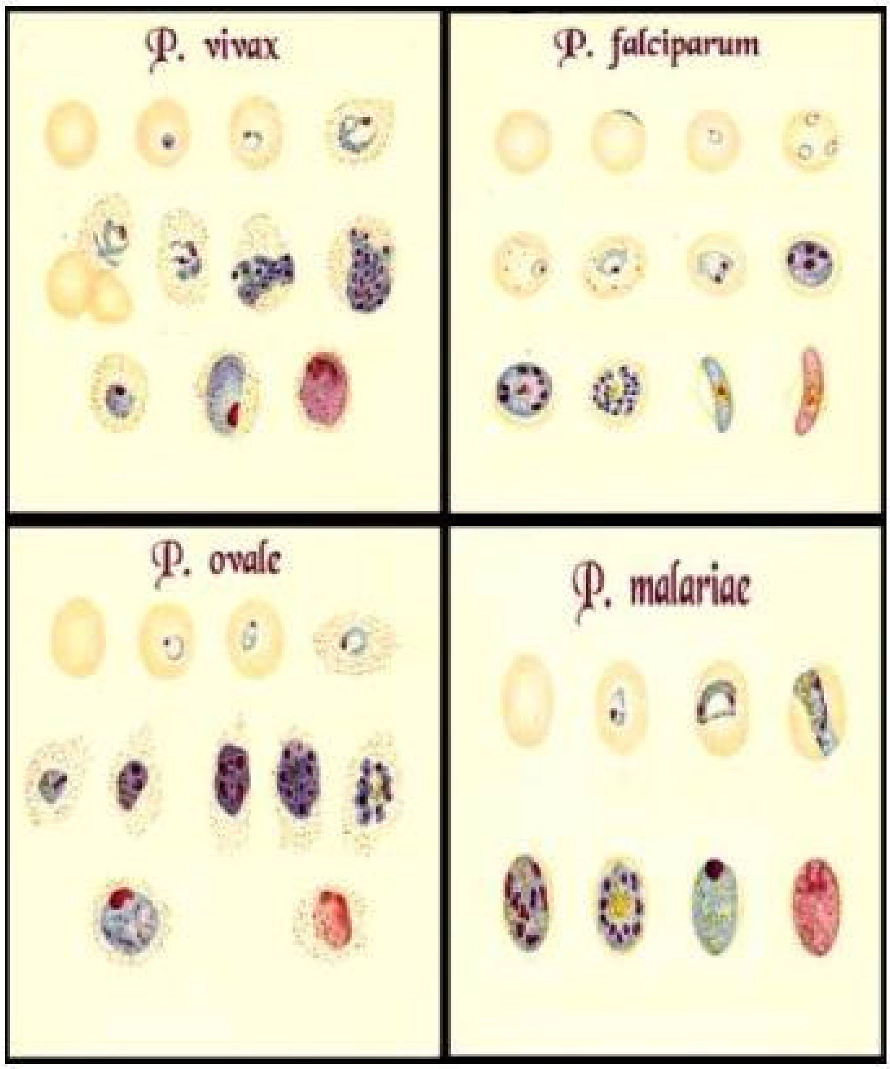


Fig 4.2 Four species of *Plasmodium*

Of the four species of *Plasmodium* that infect humans, *Plasmodium falciparum* is the most lethal. It is responsible for more than 95% of all malaria deaths. Resistance to anti-malarial drugs and insecticides, the decay of public health infrastructure, population movements, political unrest, and environmental changes are contributing to the spread of malaria (Greenwood, 2002). In countries with endemic malaria, the annual economic growth rates over a 25-year period were 1.5% lower than in other countries. This implies that the cumulative effect of the lower annual economic output in a malaria-endemic country was a 50% reduction in the per capita GDP compared to a non-malarious country (Gallup, 2001). Recent studies suggest that the number of malaria cases may double in 20 years if new methods of control are not devised and implemented (Breman, 2001).

The effort to sequence the *Plasmodium Falciparum* genome starts from 1996. It is an international collaboration, mainly includes laboratories from USA, UK and Australia. Altogether the 23-megabase nuclear genome consists of 14 chromosomes and encodes about 5300 genes. The genome sequencing work was announced completed by *Nature* in year 2002 (Gardner et al, 2002). An official website containing all the sequence information was open to all.

For microarray technology, ideally qualified probes of every single gene should be found and printed. From the gene coding sequencing file (cds file) downloaded from the malaria genome sequencing official website, the Paladin-DES package has been applied to search for all the probes for altogether 5409 sequences. When searching, the probes which fulfill all the following three criteria are defined as qualified probes.

4.2.2 Criteria for Probes Search

When applying evolutionary strategies, a fitness function is required to evaluate the performance of an individual. The ones with higher fitness will beat others in the selection scheme and will be chosen as the parents for the next generation. While designing the fitness function of this probe-searching problem, the basic considerations are specificity and sensitivity.

Specificity means that a probe should avoid cross- hybridization with other genome. It should hybridize primarily with its target. To ensure this, the probe should be a unique sequence that only appears in the specified gene and nowhere else. Good sensitivity requires favorable thermodynamics of probe-target hybridization and avoiding self-hybridization.

With these two considerations in mind, there are three essential criteria for a qualified sequence:

1) Uniqueness criterion. This will ensure the probe will not appear elsewhere in the whole genome sequence, which is the specificity consideration.
2) Melting temperature criterion. This criterion ensures the probe will have favorable thermodynamics performance.
3) Non self-folding criterion. This criterion ensures that the probe found will not perform self-hybridization.

Three functions are defined to represent the three criteria respectively.

1) $f_{\text{uni}}(x)$	$= 0.8 - \textit{length}(x) / 10000$	if the probe x is unique
	$= 0$	if the probe x is not unique
2) $f_{\text{tm}}(x)$	$= 0.1$	if the probes' melting temperature is in the desired range
	$= 0$	if the melting temperature is not in the desired range
3) $f_{\text{sf}}(x)$	$= 0.1$	if the probe has no self complementary sequence
	$= 0$	if the probe has self complementary sequence

The fitness function is defined as the summation of the three functions above.

$$F(x) = f_{uni}(x) + f_{tm}(x) + f_{sf}(x)$$

As the uniqueness test is the most important part, it has the largest weight of about 0.8. The other two tests equally share the remaining 0.2. In probes searching, shorter probes are preferred. The reason is that the longer the probe is, the more difficulties exist for the probe's hybridization on the microarray. Therefore the length of the probe has a negative effect on its final fitness value. One qualified probe will have a fitness value very close to 1. For example, consider a unique qualified probe with a length of 20 amino acids, which is 60 in nucleotide length, if it also fulfills the other 2 criteria, then it will have a fitness value of 0.8-60/10000+0.1+0.1=0.9940.

4.2.2.1 Uniqueness Criterion

First of all, the qualified probe should not appear in any other genes. If the sequence appearing in more than one gene was chosen as a probe, when doing the hybridization, it will act with the first gene coming to it instead of its designate gene. In this case the microarray experiment result will not be accurate. Uniqueness criterion is the most crucial criterion among the three and hence it takes the largest weight in the fitness evaluation function.

The computational cost of the uniqueness test is rather high. Since a probe is randomly selected from the gene and then compared with the whole 23-Megabyte genome sequence file, it is a computationally expensive task. It is this characteristic which makes traditional methods requiring extremely long time to find one probe. Moreover, the feasible region of the sequences that satisfy the uniqueness criterion is highly nonlinear. The non-linearity comes from the fact that some genes are homology, i.e. they evolve from a common ancestor, therefore they share a degree of conservation (Duret, 2000)

4.2.2.2 Melting Temperature Criterion

The stability and association between complementary DNA molecules critically depends on the melting temperature (T_m). The melting temperature of an oligonucleotide refers to the temperature at which the oligonucleotide is annealed to 50% of its exact complement. In DNA double-helix structure, 4 bases form pairs as base A from one strain always pairs with T of the opposing strain in the same location.

GC are paired together in the same way. GC pairs are held together with 3 hydrogen bonds while AT pairs have 2 bonds, therefore GC pairs need more energy to break all the hydrogen bonds, and hence the melting temperature of GC-rich sequences will be higher than AT- rich sequences. In a typical microarray experiment, thousands of DNA spots on the microarray interact with a very complex mixture of labeled DNA under one single condition. Therefore, optimal hybridization condition is necessary to obtain the best result. One way to accomplish optimal hybridization is to control the melting temperature of the immobilized DNA on the microarray.

A number of methods are present for calculating T_m. One of the more accurate equations for Tm is the Nearest Neighbor Method (Breslauer, 1986 and Santalucia, 1996).

$$T_m(x) = \frac{H(x)}{S(x) + R * \ln(C/4)} + 16.6 * \log \frac{[K+]}{1 + 0.7[K+]} - 273.15$$

where

$T_m(x)$: melting temperature of given probe x

H(x): Enthalpy for helix information of x

S(x): Entropy for helix information of x

R: molar gas constant (1.987 cal/^{o}C mol)

C: concentration of the probe (set as 250pmol)

[K+]: salt concentration (set as 50mmol)

The table of *H* and *S* values can be found in Table 4.1 and 4.2 respectively (Breslauer, 1986 and Santalucia, 1996).

2nd Nucleotide / 1st Nucleotide	A	C	G	T
A	9.1	6.5	7.8	8.6
C	5.8	11.0	11.9	7.8
G	5.6	11.1	11.0	6.5
T	5.0	5.6	5.8	9.1

Table 4.1 Enthalpy H values of a neighbor nucleotide (in -kcal/mol)

2nd Nucleotide / 1st Nucleotide	A	C	G	T
A	24.0	17.3	20.8	23.9
C	12.9	26.6	27.8	20.8
G	13.5	26.7	26.6	17.3
T	16.9	13.5	12.9	24.0

Table 4.2 Entropy S values of a neighbor nucleotide (in -cal/K.mol)

Example calculation of enthalpy H and entropy S:

H (GATC) = H (GA) + H (AT) + H (TC) = - (5.6+8.6+5.6) kcal/mol

S (GATC) = S (GA) + S (AT) + S (TC) = - (13.5+23.9+13.5) cal/K.mol

In this work the suitable value for T_m is chosen in the range of 65°C to 80°C.

4.2.2.3 Non Self-Folding Criterion

A qualified probe should not have complementary pair. If one section of a probe is the same as the complement of another section in the reverse direction, it is a complementary pair. For example, a probe has a sequence GTTGAC and another section GTCAAC. Reverse the second section, the resulting sequence CAACTG is the complementary base pairs of the first section. They form a complement pair (hairpin pattern) as illustrate in Fig 4.3. If the length of the complement pair is too long, it will cause self-hybridization; hence the probe will be inactive in microarray test. In this project, the length of the complementary pair is set to seven.

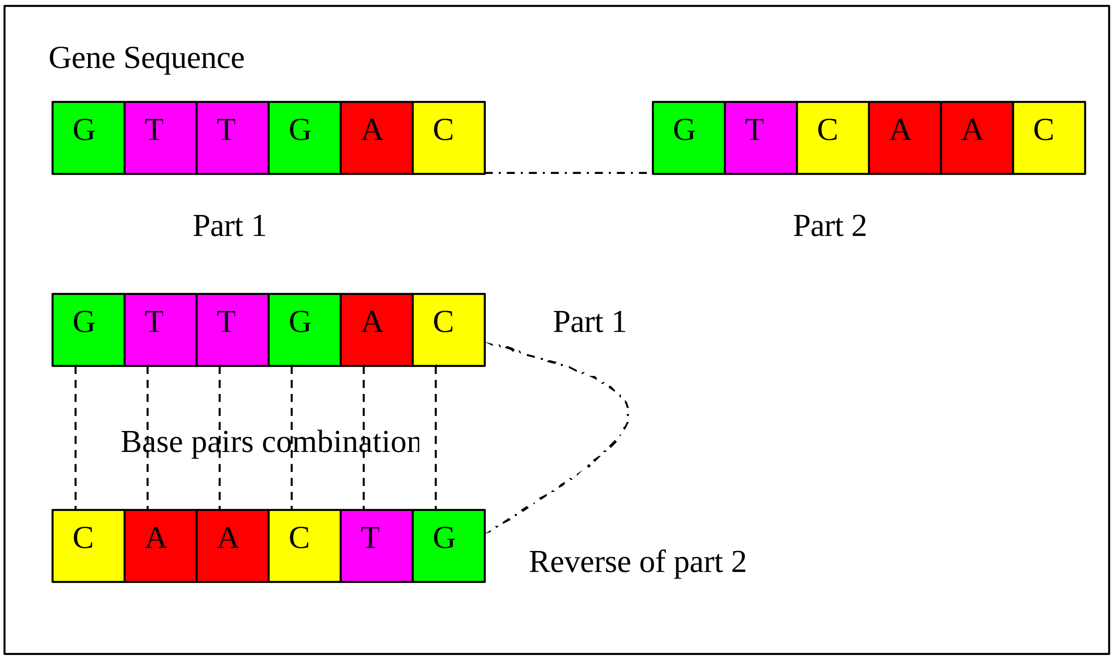


Fig 4.3 Self-folding illustration

In this work, only three criteria are set for the qualified probes. These are the three most fundamental ones. Other criteria can be included if required.

4.3 Conclusion

In this chapter the malaria parasite, *Plasmodium Falciparum's* probes searching problem has been presented. Among four kinds of *plasmodium*, the *Plasmodium Falciparum* is the most lethal. When applying the Paladin-DES package to search for the probes, three criteria have to be fulfilled. They are: uniqueness criterion, melting temperature criterion and non self-folding criterion. In the next chapter the simulation results of the distributed package and comparison with other searching techniques will be presented.

Chapter 5

Results and Discussions

5.1 Introduction

After gathering enough knowledge on the Paladin-DES package and the real world bioinformatics case study: searching oligo sets for the malaria parasite, *Plasmodium Falciparum* from previous three chapters, in this chapter we will present the simulation results of the distributed computational intelligence technique and some discussion about its performance comparison with other searching methods. It will be shown that the Paladin-DES package is a good choice in searching the probes both effectively and efficiently.

5.2 Competing Criteria

As described in chapter 4, the qualified probes should fulfill three requirements: uniqueness criterion, melting temperature criterion and non self-folding criterion. On the base of these three searching criteria, there are two measuring criteria: effectiveness and efficiency. Effectiveness refers to the quantity aspect, which means whether the program can locate all the probes in the genome. Efficiency refers to quality part, which is the time required to locate one qualified probe. In the following comparison between the Paladin-DES package and other searching techniques, attention will be paid on these two aspects.

5.3 Simulation Setup

When setting up the simulation for Paladin-DES package to search the probes, some general settings for normal evolutionary computation are applied. Table 5.1 shows the ES parameters used in searching the *Plasmodium Falciparum* genome.

Parameter Type	Parameter value
Generation size	500
Total population size	200
Mutation rate	0.1
Selection type	Tournament Selection
Migration rate	0.02
Migration interval	0.1

Table 5.1 ES parameter in *Plasmodium Falciparum* case

In the simulation to find all the probes for the 5000 more genes of *Plasmodium Falciparum*, maximally 10 computers in the university LAN are used for distributing the job simultaneously. All of them have different processing unit frequency and memory, so the computation power also varies from each other largely from one computer to another and hence the performance of each peer has a large difference. For allocating one qualified probe, the fastest peer needs only 5 seconds, while the slowest needs more than 20 seconds. Fig 5.1 shows how long it takes one peer computer to obtain a qualified probe.

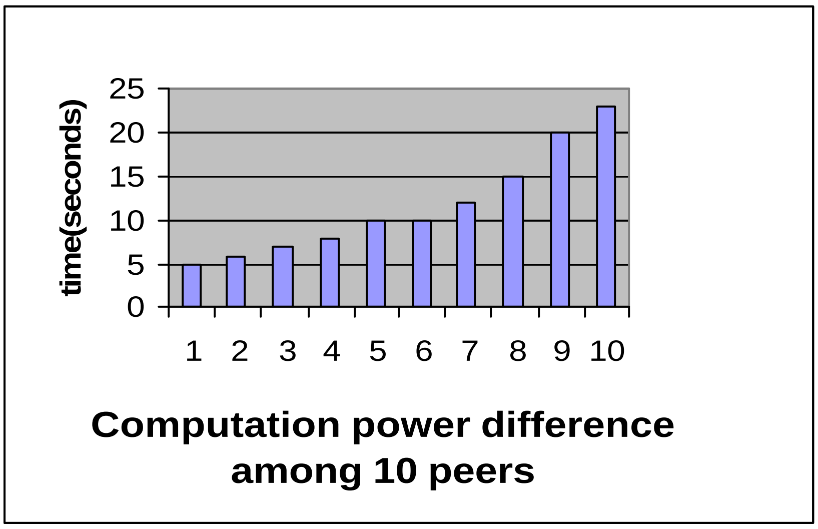


Fig 5.1 Peer computers' computation difference

During the process of the simulation, the 10 computers are grouped into two categories according to their computation power. Later in this chapter the group with the 5 faster computers will be referred to as faster group, while the other will be referred to as slower group.

5.4 Simulation Results

The Paladin-DES was first applied on two small organisms *Buchnera sp. APS* and *Chlamydia pneumoniae*, which have 575 and 1054 genes respectively, to test whether it is capable of finding gene probes. The testing results showed that the percentages of probes being found of both species are more than 99%. After getting this inspiring result, the package was applied to the *Plasmodium Falciparum* case, whose genome is much larger, more than 5000 genes. Table 5.2 shows the searching results for the three different organisms.

Method	Paladin-DES (5 peers, faster group)	Paladin-DES (5 peers, faster group)	Paladin-DES (10 peers)
Applied species	*Buchnera sp. APS*	*Chlamydia pneumoniae*	*Plasmodium Falciparum*
Number of non-found probes	2	0	54
Total number of genes tested	575	1054	5409

Table 5.2 Simulation results of DES applied to three different organisms

Fig 5.2 shows the sample for the found probes'location within each gene. From this figure it is clearly shown that the location of the probes are extremely distributed, or uncertain. For some genes, it may appear just from the beginning while others may exist at the end of the gene.

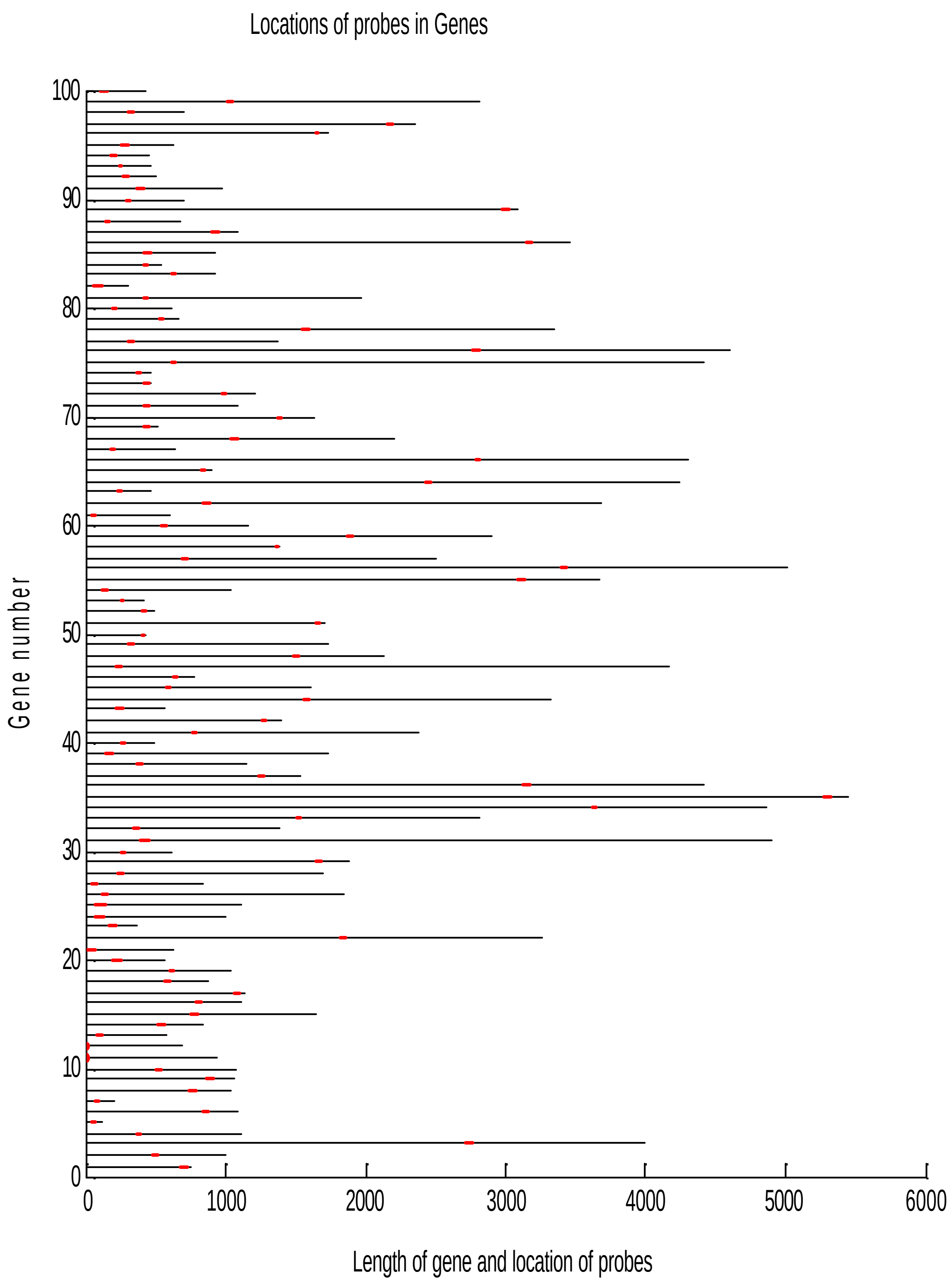


Fig 5.2 Sample found probes locations in gene

5.5 Comparison

Existing techniques for searching of these probes are not really available; a standard approach one could think of is to select a probe from a sequence and comparing it with all other sequences within the genome. It becomes computationally intensive when applied to more complicated genomes. In this section two most frequently used methods: enumerating method and ES with BLAST method are discussed and their searching results are compared with the results from Paladin-DES package.

5.5.1 Enumerating Method

A most straightforward way one could think of for finding unique probes is the enumerating method. A probe is first selected from a sequence and compared with all other sequences within the genome. Once a probe is found, it is then tested whether it meets the other 2 criteria. One would expect such a thorough search to be computationally intensive due to its large search space. The number of sub-sequence of a gene (sequence) with length n is $(n(n-1)/2)$. For a typical gene in the malaria parasite *Plasmodium Falciparum* with length 1000, there will be 500,000 sub-sequences to be tested to find a qualified probe. It is clear that as the length of the genes gets longer, it becomes more computationally intensive in the search process.

In this project, the enumerating method is coded in JAVA language which is the same as the Paladin-DES package.

```
import java.io.*;

public class enumerating_method{

//read each gene from the original cds file
public static void readGeneFile(){
… .
}

public static void main(String[] args){

//loop until all the genes in the cds files are read, save the gene presently as gene
while(gene=in.readLine()!=null){

//select from the beginning of the gene, a certain length of nucleotide as testing
//sequence
x = gene.chosen(length)

//first check the uniqueness criterion
if (checkUniqueness(x))

        //if passed, check self folding criterion
        if (checkSelf_folding(x))

                //if passed, check the melting temperature criterion
                if(checkMT(x))

                        //if all the three criteria passed, print to a result file
                        out= FileWriter("enumeration_out.dat ");

}//end of while
}//end of main
}//end of class
```

5.5.2 ES with BLAST method

ES, because of its powerful optimization searching scheme, is a good candidate of finding the probes. ES combined with a biological software tool called BLAST has been previously applied in searching for the oligo sets of human chromosome 12 (Tay, 2002) so as to speed up the whole searching process.

Basic Local Alignment Search Tool (BLAST) is a powerful method that shows good overall search speed and puts database searching on a firm statistical foundation in local alignment, both for protein and DNA. Tens or hundreds of genes can be put together and compared with the database to find whether there are any same sub-sequences in these genes. The uniqueness test is the bottleneck and most time-consuming part of sequencing comparison algorithms, with BLAST, multiple genes are compared simultaneously and therefore importing BLAST into ES saves a lot of time. In ES with BLAST method, BLAST is used to reduce the computational time of the uniqueness test. The results of the uniqueness test are sent to MATLAB for the other two tests.

As suggested by the name, BLAST searches for local alignments, meaning that given a long query sequence, BLAST will report sequences in the database that significantly match the subsequences of the query sequence. Consequently, non-unique regions in a gene can be identified by feeding the gene as a query sequence to BLAST. Comparing to other searching methods, the main computation task of ES with BLAST falls into the checking of non self-folding criteria instead of uniqueness checking.

There are three basic parameters in BLAST that can be varied to adjust the sensitivity of BLAST search. They are Expected value (E), the threshold value (T) and the word size (w).

The BLAST used in the simulation is the standalone BLASTN version 2.2.6 for windows. The standalone BLAST 2.2.6 version is downloaded from the National Center for Biotechnology Information (NCBI) ftp website. The procedures in evaluating the three criteria of a probe are:

(1) Use `formatdb` command of standalone BLAST to prints the gene sequence being evaluated to a text file `current.txt`.

(2) Set BLAST with the following parameter w = T = S = 15. S and E are related by

$$E = Kmne^{-lS}$$

where m and n are the lengths of the two sequences being compared, K and l are constants. (K=0.711, l =1.37) (Karlin and Altschul, 1990)

(3) Run the BLAST with the following command

```
blastall –p blastn –d db.fasta –i current.txt –o
out.txt –F F –g F –W 15 –f 15 –e evalue
```

The command is running from the main algorithm, which is implemented in MATLAB, by using the in-built DOS interface.

(4) The main algorithm reads the BLAST report in out.ext, creating the list containing the matching subsequences.

(5) Evaluate the uniqueness of each individual, if it passes the uniqueness test, then proceed to MATLAB for the melting temperature and non self-folding test which has no relationship with BLAST any more.

5.5.3 Effectiveness Comparison

The fastest peer computer among the ten candidates, which is a Pentium IV, 1.6G Hz, 512M RAM computer, is used for the enumerating and ES with BLAST simulation. For the Paladin-DES package proposed here, multiple peers can work together to search qualified probes. The results show that the package is more effective and more efficient than the other methods. Table 5.3 shows the effectiveness comparison for the three different searching techniques.

Method	Enumerating Method	ES with BLAST method	Paladin-DES (1 peer)	Paladin-DES (10 peers)
Number of non-found probes	50	76	1616	54
Total number of genes/exons tested	5409	501	5409	5409
Effectiveness	99.1%	84.83%	70.1%	99.0%

Table 5.3 Effectiveness comparison

The Paladin-DES package performs very well in finding the probes. From the table it can be seen that the Paladin-DES package performs quite good in the *Plasmodium Falciparum* case. It can achieve an effectiveness above 99%, which can reach the same level of enumerating method, which is the most thorough searching algorithm. For the ES with BLAST method, because it has the window size limitation, it performs not as well as the other two methods.

However, it has to be pointed out that this good result is the contribution of multiple peers. If only one peer is present, the finding ratio is only about 70%. When multiple clients are available, the migration scheme will transport the good individuals between different sub-populations. This increases the opportunity of higher fitness candidate to be found than using only one single client computer. And hence increases the chance of locating a qualified probe.

5.5.4 Efficiency Comparison

Although the enumerating method is the most thorough technique in searching the probes, it is also the most time-consuming method. In this section of efficiency comparison, the enumerating method is taking the disadvantage.

5.5.4.1 Comparison between Paladin-DES and ES with BLAST

ES with BLAST has the advantage in uniqueness criterion testing. BLAST is a proven technique in sequence comparison, and it is the most powerful and popular tool used for sequence aliasing presently. Fig 5.3 shows the uniqueness test results by using the Paladin-DES package and BLAST.

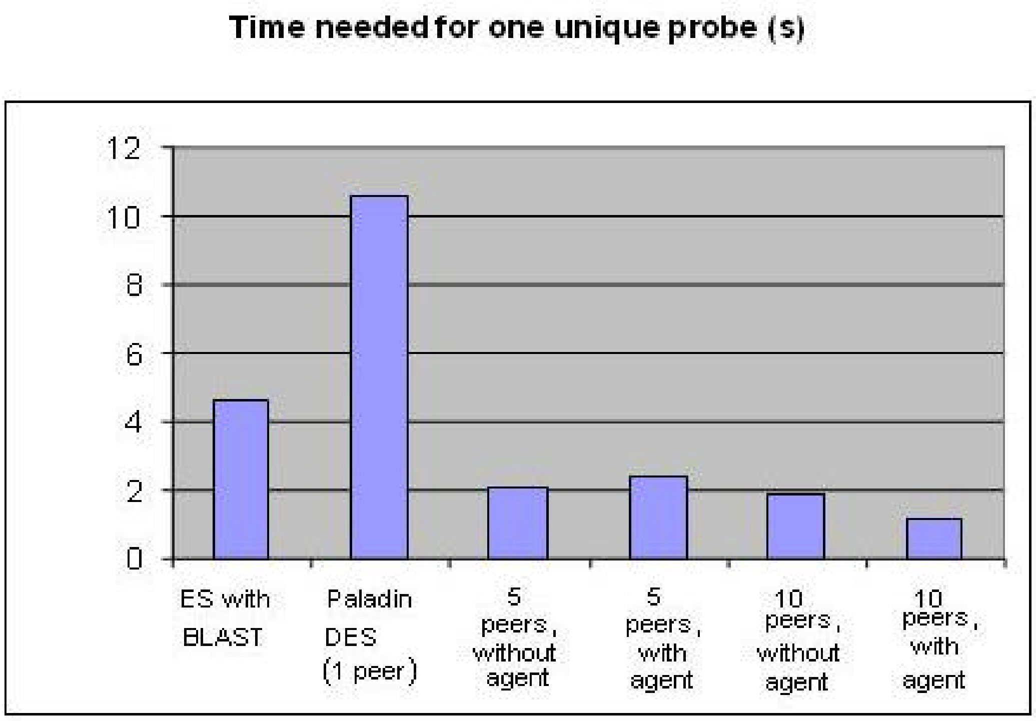


Fig 5.3 Uniqueness comparison between Paladin-DES and ES with BLAST

As expected, with only one computer, ES with BLAST will perform much faster comparing with the proposed DES package. BLAST software only needs 4.59s to find out one unique sequence while DES needs 10.516s. The advantage of the Paladin-DES package can only be shown when multiple peers distribute the job and work simultaneously. From the results it can be seen that when 5 peers logon the distributed system and work together, the package takes about 2 seconds for locating one unique sequence, which saves half the time BLAST needs. The computation power brought in by multiple peer computers is much stronger than the high performance technology like BLAST, and this is the underlying reason why the distributed system is developed.

5.5.4.2 Comparison between Paladin-DES and Enumerating method

Although the enumerating method is the most thorough searching method, it is computationally time consuming. By using only one computer, it finds 5359 probes out of 5409 genes using 295,424 seconds. On average, it requires 55.1 second for one probe. For the Paladin-DES package, with one computer, it only takes 11.6 second to locate one qualified probe averagely. Table 5.4 shows the efficiency difference between the enumerating method and the Paladin-DES package.

Searching Method	Average time needed for one qualified probe (second)
Enumerating Method	55.1
Paladin-DES (1peer, average)	11.6
Paladin-DES (5 peers with agent)	2.3
Paladin-DES (5 peers without agent)	2.1
Paladin-DES (10 peers without agent)	1.9
Paladin-DES (10 peers with agent)	1.1

Table 5.4 Efficiency comparison between Paladin-DES and enumerating method

Table 5.4 also demonstrates the effect of adding more resources (computers) in solving the problem. The 10 computers involved in this simulation are of different computational power. They are divided into 2 categories according to their computational capabilities. Results shown in row 4 and 5 of Table 5.4 are from the faster group.

From Table 5.4, it is obvious that 5 peers need 1/5 of the time required for 1 PC. While from 5 peers to 10 peers, the time reduced is not very large. The reason is that in the scheduling procedure, faster-funning PCs will have to share some work from slower ones and hence the time saved will not be as impressive as adding the first a few PCs.

Another observation is the effect of assigning an agent. From table 5.4 it could be noticed that with five peers present in the system, assigning an agent does not make the system function faster. However for the 10-peer case, agent-mode does improve the whole performance. As has been mentioned above, instead of talking to server directly, in the agent mode, clients can transfer data to the agent and it is agent that directly communicates to the server. When only 5 peers logged on, the communication overhead is not very heavy, assigning an agent will reduce the total computation power of the system; as more peers comes in, assigning an agent to manage data transfer does help the system work faster.

5.6 Missing Probes

Even with the most thorough searching technique, the enumerating methods, there are still 50 missing probes. This is a quite strange phenomenon. After checking the genome sequence file again, it is found that all these genes have failed to pass the most important criteria, uniqueness test. For example, gene 1190 and gene 1751 are identical to each other. Obviously they will fail the uniqueness test. Every partition from gene 1190 will find its clone in gene 1751. Therefore the qualified probes for both of the two genes could not be found. For all the 50 genes whose probes could not be found, each one of them can find its clone pairs inside these 50.

5.7 Conclusion

In this chapter the simulation result of applying the Paladin-DES package to search probes of malaria parasite, *Plasmodium Falciparum*, has been shown. Effectiveness and efficiency, these two competing criteria have been defined. From the result comparison it has been shown that the Paladin-DES package demonstrates good performance in terms of the number of probes found and computational time when comparing with the traditional enumerating methods and other previously developed probe-finding algorithms.

Chapter 6

Conclusions and Future Directions

6.1 Conclusions

In this thesis a distributed computational intelligence technique, Paladin-DES package has been introduced. The Paladin-DES package was developed on the bases of Paladin-DEC package, which exploits the inherent parallelism of evolutionary algorithms by creating an infrastructure necessary to support distributed evolutionary computing using existing Internet and hardware resources.

The Paladin-DES has been applied to a real world bioinformatics problem: to search for unique and optimized probe sets. Probes of the human malaria parasite, *Plasmodium Falciparum*, have been found using the Paladin-DES package and results are compared with other previously developed techniques. Simulation results demonstrate the capability of Paladin-DES.

6.2 Future Directions

This project is the third stage of the distributed computational intelligence research. There are three ways on which the research can be further carried out.

The first way is to further improve the system. Theoretically with a powerful server, the current distributed system could handle as many peer computers as possible. However, practically the network delay confines the number of peers the server can manage to a finite number. More consideration can be given on the system fault tolerance, security, robustness to make the server handle more peers.

Secondly the package can be applied to some genomes which are much more complicated. The case study in this project is human malaria parasite, *Plasmodium Falciparum*, which consists of 5000 more genes. The package can be applied to some larger genomes, for example, plants genomes which normally contain more than a hundred thousand genes.

The third way is to combine the underlying distributed technology with other computational intelligence techniques. Some research has been done by using artificial neural network to handle HIV's multi -drug resistance problem.

Drug resistance is probably the most important factor influencing the failure of present HIV therapies. The emergence of anti-retroviral drug resistance is not unexpected, as drug resistance had been reported for other viruses such as herpes simplex, varicella-zoster, cytomegalovirus, influenza A and rhinovirus. However, the drug resistance problem is far more important in the case of the HIV virus because of the severe final outcome of HIV-related illnesses (Draghici and Potter, 2003).

In the literature, it is discovered that the effectiveness of the contacts between the protease inhibitor drug Saquinavir and the HIV protease gene is related to the amino acid sequence of HIV protease mutants. The prediction is based on a set of HIV protease mutants with reported Saquinavir IC90 values, which were used to classify the resistance of the mutants tested. In this research, a Learning Vector Quantisation (LVQ) network is constructed for the purpose of predicting the HIV resistance of the drug Saquinavir and results generated will be compared with a SOFM (Self-Organizing Features Map) network used by Draghici and Potter.

Further research can be done in this direction since the multi-drug resistance research is very complicated and very helpful in both medical and biology science. With including the different insight from engineering aspect, the problem could be analyzed more specifically.

References

Altschul SF., Gish W., Miller W., Myers EW. and Lipman DJ. Basic Local Alignment Search Tool, Journ. Mol. Biol 1990; vol. 215: 403-410

Arabas, J., Michalewicz Z. and Mulawka J. GAVaPS-A Genetic Algorithm with Varying Population Size. Proceedings of the First Conference on Evolutionary Computation, 1994; vol. 1, 73-74

Back, T., Fogel, D. B., and Michalewicz, Z. (editors). Handbook on Evolutionary Computation, Bristol, UK: Institute of Physics Publishing and New York: Oxford University Press, 1997.

Bahl A et al, PlasmoDB: the Plasmodium genome resource. A database integrating experimental and computational data. Nucleic Acids Res. 2003 Jan 1; vol 31, issue 1: 212-215

Baxevanis AD, and Ouellette BF. Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins, Wiley-Interscience; 2001

Ben Mamoun C et al, Co-ordinated programme of gene expression during asexual intraerythrocytic development of the human malaria parasite Plasmodium falciparum revealed by microarray analysis. Mol Microbiol. 2001 Jan; vol 39, issue 1:26-36

Bezdek JC. On the relationship between neural networks, pattern recognition and intelligence, Int. J. Approximate Reasoning; 1992; vol. 6: 85-107

Bezdek JC. What is computational intelligence? Computational Intelligence Initating Life, IEEE Press; 1994: 1-12

BLAST http://www.ncbi.nlm.nih.gov/blast

Bioinformatics Introduction http://www.library.csi.cuny.edu/~davis/molbiol/lecture_notes/bioinformatics_genomics/bioinformaticsIntro.html

Bosch JT., Seidel C., Batra S., Lam H., Tuason N., Saljoughi S., and Saul R. Validation of sequence-optimized 70 base oligonuclieotides for use on DNA microarrays; 2000; at http://www.operon.com

Bozdech Z, et al, Expression profiling of the schizont and trophozoite stages of Plasmodium falciparum with a long-oligonucleotide microarray. Genome Biol. 2003;4(2):R9. Epub 2003 Jan 31

Breman JG. The ears of the hippopotamus: manifestations, determinants, and estimates of the malaria burden. Am. J. Trop. Med. Hyg 2001; vol. 64:1-11

Breslauer KJ, et al. Predicting DNA duplex stability from the base sequence. Proceeding Natural Academic Science 1986; vol. 83:3746-3750

Cantú-Paz E. A survey of parallel Genetic Algorithms, Calculateurs Paralleles, Reseaux et Systems Repartis, Paris: Hermes, 1998; vol. 10, no. 2: 141-171,

Chait, Y. QFT Loop-Shaping and Minimisation of the High-frequency Gain via Convex Optimisation. Proceedings Symposium on Quantitative Feedback Theory and other Frequency Domain Method and Applications, Glasgow, Scotland, 1997; 13-28

Chen, Y. W., Nakao, Z., and Xue F., “A Parallel Genetic Algorithm Based on the Island Model for Image Restoration”, Proceedings of the 13th International Conference on Pattern Recognition, 1996; vol. 3, 694-698

Chipperfield, A.J. and Fleming PJ. Gas Turbine Engine Controller Design using Multiobjective Genetic Algorithms. In Proceedings of the First IEE/IEEE

International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, ed. by A.M.S. Zalzala, 1995; 214-219

Cristea, V. and Godza, G., “Genetic algorithms and Intrinsic Parallel Characteristics”, Proceedings of the 2000 Congress on Evolutionary Computation, 2000; vol. 1, 431- 436

Dasgupta D, and Michalewicz Z. Evolutionary algorithm- An overview. Evolutionary Algorithms in Engineering Application. Springer 1997; 3-28

Degrave WM, Melville S, Ivens A, Aslett M. Parasite genome initiatives. Int J Parasitol. 2001 May 1; 31(5-6):532-536

Dejong KA. Analysis of the behavior of a class of genetic adaptive systems. Ph.D. Thesis USA: University of Michigan, Ann Arbor, MI; 1975

DeRisi J., et al. Exploring the metabolic and genetic control of gene expression on a genomic scale; Science 1997; vol 278, issue 5338: 680-686

Draghici S. and Potter RB. Predicting HIV drug resistance with neural networks. Oxford University Press, 2003; vol 19, issue1: 98-107

Duggan DJ., et al. Expression profiling using cDNA microarrays. Nature Genetics 1999; vol. 21;supplementary 10-14

Duret L, and Abdeddaim S. Multiple Alignments for Structural, Functional, or Phylogenetic analyses of homologous sequences. Bioinformatics: Sequence, Structure and databases. Oxford press 2000; 51-76

Engelbrecht AP. Computational Intelligence, an Introduction, John Wiley & Sons Ltd, 2002

Fogel DB. An Introduction to Simulated Evolutionary Optimization. Evolutionary Computation, the fossil record. IEEE Press 1998; 1-28

Gallup JL, and Sachs JD. The economic burden of malaria. Am. J. Trop. Med. Hyg 2001; vol. 64:85-96

Ganesan K, Jiang L, Rathod PK. Stochastic versus stable transcriptional differences on Plasmodium falciparum DNA mic roarrays. Int J Parasitol. 2002 Dec 4; vol 32, issue 13:1543-1550

Gardner MJ, et al. Genome Sequence of the Human Malaria parasite Plasmodium Falciparum. Nature 2002; vol. 419:498-511

Gardner MJ, et al. Sequence of Plasmodium falciparum chromosomes 2, 10, 11 and 14. Nature 2002; vol. 419:531-534

Gershon D. Microarray technology: An Array of Opportunities. Nature 2002; vol 416:885 – 891

Goldberg DE. Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley, Massachusetts, 1989a

Goldberg DE. Sizing populations for serial and parallel genetic algorithms, In Schaffer, J. D. (editor). Proceedings of the Third International Conference on Genetic Algorithms. San Mateo, CA: Morgan Kaufmann Publishers Inc., 1989b: 70-79

Greenwood B, and Mutabingwa T. Malaria in 2002. Nature 2002; vol. 415:670-672

Grefenstette JJ, and Baker JE. How Genetic Algorithm Work: A Critical Look at Implicit Parallelism. Genetic Algorithm, IEEE Computer Society press 1992: 12-19

Hall N, et al. Sequence of Plasmodium falc iparum chromosomes 1, 3–9 and 13, Nature 2002; vol. 419:527-531

Haykin S., Neural Networks – A Comprehensive Foundation, Prentice Hall International Inc, Second Edition 1999.

Hayward RE, Derisi JL, Alfadhli S, Kaslow DC, Brown PO, Rathod PK. Shotgun DNA microarrays and stage-specific gene expression in Plasmodium falciparum malaria. Mol Microbiol. 2000 Jan; vol 35, issue 1:6-14.

Higgins D. and Taylor W., Bioinformatics: Sequence, Structure, and Databanks: A Practical Approach, Oxford University Press, 2000

Hiroyasu, T., M. Miki and S. Watanabe. Distributed Genetic Algorithm with a New Sharing Approach in Multiobjective Optimization Problems. IEEE International Conference on Evolutionary Computation, 1999; Vol.1, 69-76

Hoffman SL, et al. Plasmodium, human and Anopheles genomics and malaria. Nature 2002; vol 415:702-709

Hoffmeister F, and Baeck T. Genetic algorithms and evolution strategies: similarity and differences. Technical Report No. SYS-1/92, University of Dortmund, 1992

Holland J. Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor, Mich; 1975

Hughes TR., Mao M., Jones AR., Burchard J., Marton MJ., Shannon KW., Lefkowitz SM., Ziman M., Schelter JM., and Meyer et al. Expression profiling using microarrays fabricated by an ink-jet oligonucleotide synthesizer. Nature Biotechnology, 2001; vol. 19, issue 4: 342-347,

Hyman RW, et al. Sequence of Plasmodium falciparum chromosomes 12. Nature 2002; vol. 419:534 -537

Joe YY, Xu H, Dong ZY, Ng HH, and Tay A. Searching Oligo Sets of Human Chromosome 12 using Evolutionary Strategies, Congress on Evolutionary Computation 2003

Karlin S. and Altschul SF. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. Proceedings of National Academy of Science, U.S.A 1990; vol. 87; 2264-2268

Lin, D.S. and Leou J.J. A Genetic Algorithm Approach to Chinese Handwriting Normalization. IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, 1997; Vol. 27, No. 6, 999-1007.

Lipshutz RJ., Fodor SP., Gingeras TR., and Lockhart DJ., High Density Synthetic Oligonucleotide Arrays. Nature Genetics, 1999; vol. 21, issue 1, supplement: 20-24

Lockhart DJ and Winzeler EA. Genomics, Gene Expression and DNA Arrays. Nature 2000; vol. 405:827- 836

Meta Group Consulting, CORBA VS DCOM: Solutions for Enterprise, 1998

Michalewicz Z. Genetic Algorithms + Data Structure = Evolutionary Programs, Springer-Verlag, Berlin, 2nd Edition, 1994

Mohammadian M, Sarker RA. and Xin Y. Computational Intelligence in Control, Idea Group Publishing, 2003

NCBI Education http://www.ncbi.nlm.nih.gov/Education

Operon http://www.operon.com

Paechter, B. and Back, T. A Distributed Resources Evolutionary Algorithm Machine (DREAM). Proceedings of the 2000 Congress on Evolutionary Computation, 2000; vol. 2: 951-958

Paladin-DES package searching malaria parasite result http://evolab.ece.nus.edu.sg/project_malaria/result/

Pease, A. C., Solas, D., Sullivan, E. J., Cronin, M. T., Holmes, C. P., and Fodor, S. P., "Light Generated Oligonucleotide Arrays for Rapid DNA Sequence Analysis", Proceedings of the National Academy of Sciences of the United States of America, 1994; vol. 91, 5022-5026

Pedrycz W. and Vasilakos A. Computational Intelligence in Telecommunication Networks, CRC Press LLC, 2001

Pedrycz W. and Peters JF. Computational Intelligence in Software Engineering, World Scientific, 1998

Pena-Reyes CA, and Sipper M. Evolutionary computation in medicine: an overview. Artif Intell Med 2000; vol. 19:1-23

Phimister B. Going global. Nature Genetics 1999; vol. 21, pp. 1

Plasmodium Falciparum sequence website http://www.plasmodb.org/restricted/GridddPf.shtml

Rathod PK, Ganesan K, Hayward RE, Bozdech Z, DeRisi JL. DNA microarrays for malaria. Trends Parasitol. 2002 Jan; vol 18, issue 1:39-45

Ren B. et al. Genome-wide location and function of DNA binding proteins. Science 2000; vol. 290:2306-2309

Rivera W. Scalable Parallel Genetic Algorithms. Artificial Intelligence Review 2001; vol. 16:153-168

Subbu, R. and Sanderson, A. C., "Modeling and convergence analysis of distributed co-evolutionary algorithms", Proceedings of the 2000 Congress on Evolutionary Computation, 2000; vol. 2, 1276-1283

Santalucia J, Allawi HT, and Seneviratne PA. Improved Nearest-Neighbor Parameters for Predicting DNA Duplex Stability. Biochemistry1996; vol. 35, issue 11:3555-3562

Schena M, et al. Quantitative Monitoring of Gene Expression Patterns with a Complementary DNA Microarray. Science 1995; vol. 270:467-470

Schwefel, H. P., *Evolution and Optimum Seeking*, New York, NY: John Wiley, 1995.

Sun Microsystems Inc., IBM co. RMI-IIOP Programmer's Guide, 1999.

Sun Microsystems Inc. J2EE tutorial, 2001a.

Sun Microsystems Inc. Java Message Service Tutorial, 2001b.

Sun Microsystems Inc. www.jxta.org, 2002.

Snustas DP, Simmons MJ, Jenkins JB. Principles of Genetics. John Wiley; 1997

Tan KC., Khor EF., Cai J., Heng CM. and Lee TH. Automating the drug scheduling of cancer chemotherapy via evolutionary computation, Artificial Intelligence in Medicine, 2002; vol. 25: 169-185.

Tan KC., Tay A. and Cai J. Design and implementation of a distributed evolutionary computing software. IEEE Transactions on Systems, Man and Cybernetics: Part C (Applications and Reviews), 2003; vol. 33, issue 3: 325-338

Tan KC., Wang ML. and Peng W. A P2P genetic algorithm environment on the Internet. Communications of the ACM, accepted.

Topping BHV, Khan AI, and Sziveri J. Parallel and Distributed Processing for Computational Mechanics: An Introduction. Parallel and Distributed Processing for Computational Mechanics: Systems and Tools. Saxe-Coburg Publication 1999; 1-23

Urben BC, et al. Plasmodium falciparum-infected erythrocytes modulate the maturation of dendritic cells. Nature 1999; vol. 400:73 – 77

Wu Y, Wang X, Liu X, Wang Y, Data-mining approaches reveal hidden families of proteases in the genome of malaria parasite, Genome Res. 2003 Apr; vol 13, issue 4 :601-16.

Xu H, Tay A, Dong ZY, and Ng HH. Searching Probe Set of Yeast Genome: An implementation of Evolutionary Strategy, 4th Asian Control Conference 2002; Sept: 25-27

Yoshida N. and Yasuoka T., "Multi-GAP: Parallel and Distributed Genetic Algorithms in VLSI", IEEE International Conference on Systems, Man, and Cybernetics 1999; vol. 5:571-576.

Young RA. Biomedical discovery with DNA arrays. Cell 2000; vol. 102; 9-15

Zurada JM., Marks II R.J., and Robinson C.J. Computational Intelligence Imitating Life, IEEE Press, 1994

List of Publications

Journal Papers

Tan, K. C., Wang, M. L. and Peng, W., 'A P2P genetic algorithm environment on the Internet', Communications of the ACM, accepted.

Conference Papers

Tan, K. C., Peng, W., Lee, T. H. and Cai, J. (2003). Development of a distributed evolutionary computing package, IEEE Congress on Evolutionary Computation 2003, Canberra, Australia, 8-12 December, pp. 77-84.