NEW HYPER-HEURISTIC ALGORITHM FOR GENE FRAGMENT ASSEMBLY

MURNIYANTI BINTI MALIK

UNIVERSITI TEKNOLOGI MALAYSIA

PSZ 19:16 (Pind. 1/07)

**UNIVERSITI TEKNOLOGI MALAYSIA**

# UNIVERSITI TEKNOLOGI MALAYSIA

**DECLARATION OF THESIS STATUS**

Author’s full name : MURNIYANTI MALIK

Date of birth : 1st NOVEMBER 1991

Title : NEW HYPER-HEURISTIC ALGORITHM FOR GENE FRAGMENT ASSEMBLY

Academic Session : 2016/2017

I declare that this thesis is classified as:

I acknowledged that Universiti Teknologi Malaysia reserves the right as   
 follows:

1. The thesis is the property of Universiti Teknologi Malaysia.
2. The Library of Universiti Teknologi Malaysia has the right to make copies for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

**SIGNATURE OF STUDENT SIGNATURE OF SUPERVISOR**

**(NEW IC. NO. / PASSPORT NO.) NAME OF SUPERVISOR**

Date: 28TH FEBRUARY 2017 Date: 28TH FEBRUARY 2017

**CONFIDENTIAL** (Contains confidential information under the Official Secret Act 1972)\*

**RESTRICTED** (Contains restricted information as specified by the organization where research was done)\*

✓

**OPEN ACCESS** I agree that my thesis to be published as online open access (full text)

**DR AFNIZANFAIZAL BIN ABDULLAH**



911101-12-6030

**NOTES:** \* If the thesis is CONFIDENTIAL or RESTRICTED, please attach with the letter  
 from the organization with period and reasons for confidentiality or   
 restriction.

“I hereby declare that I have read this thesis and in my

opinion this thesis is sufficient in term of scope and quality for the

award of the degree of Master of Philosophy”

|  |  |  |
| --- | --- | --- |
| Signature | : | ……………………………….. |
| Name of Supervisor | : | Dr. Afnizanfaizal Bin Abdullah |
| Date | : | 28th February 2017 |

NEW HYPER-HEURISTIC ALGORITHM FOR GENE FRAGMENT ASSEMBLY

MURNIYANTI BINTI MALIK

A thesis submitted in fulfillment of the

requirements for the award of the degree of

Master of Philosophy

Faculty of Computing

Universiti Teknologi Malaysia

FEBRUARY 2017

DECLARATION

I declare that this thesis entitled “*New Hyper-Heuristic Algorithm for Gene Fragment Assembly*” is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

|  |  |  |
| --- | --- | --- |
| Signature | : | …………………………….... |
| Name | : | MURNIYANTI BINTI MALIK |
| Date | : | 28th February 2017 |

To my beloved family

ACKNOWLEDGEMENT

First of all, I would like to acknowledge my supervisor, Dr. Afnizanfaizal Abdullah for his guidance and advice throughout the making of this thesis and for his patience while dealing with a student like me. I would like to thanks the panel of examiners involved during my VIVA presentation for evaluating my thesis. I am very grateful for your help. Secondly, I would like to thanks my friends and seniors for their helps, supports, opinions and continuous information shared throughout the semester to help me finish my thesis. Without them, I would be a lost cause. Thirdly, I would like to give a thousand thanks to the staff of Faculty of Computing, Universiti Teknologi Malaysia, Mrs. Nizamra, Ms. Norshahida, Dr Norafida, Dr Zalmiyah, and Deputy Dean (Academic) for helping me throughout the process of submitting my thesis. Last but not least, I would like to express my deepest gratitude to my parent and family for their support and encouragement during my master study.

ABSTRAK

Perhimpunan gen adalah teknik untuk mengenalpasti jujukan gen berdasarkan serpihan gen yang dijana oleh mesin penjujukan. Serpihan gen tersebut adalah pendek dan banyak. Sekiranya bilangan serpihan gen meningkat, kerumitan masalah meningkat, dan situasi ini menjadikan ruang penyelesaian menjadi semakin luas. Untuk menyelesaikan masalah ini, serpihan gen perlu disusun di dalam susunan yang betul. Namun, disebabkan kerumitan dan ruang penyelesaian yang besar, penyelesaian sukar didapati. Melihat dari perspektif pengkomputeran, masalah perhimpunan serpihan gen dianggap sebagai masalah polinomial tidak berketentuan (NP), dimana masalah ini boleh diselesaikan dengan menggunakan algoritma metaheuristik. Algoritma metaheuristik mengoptimumkan masalah dengan mencari penyelesaian yang hampir optimal. Dalam penyelidikan ini, satu algoritma hiper-heuristik dicadangkan untuk menyelesaikan masalah perhimpunan serpihan gen. Penyelidikan ini dibina berdasarkan tiga objektif. Pertama, untuk menganalisa dua algoritma metaheuristik, iaitu Pengoptimuman Reaksi Kimia (CRO) dan Algoritma Kuantum yang diinspirasikan daripada Algoritma Evolusi (QIEA). Kedua, algoritma hiper-heuristik yang baru dibangunkan berdasarkan CRO dan QIEA. Ketiga, penyelesaian yang didapati daripada ketiga-tiga algoritma dinilai menggunakan analisis statistik. Prestasi algoritma-algoritma dinilai dengan menggunakan analisis penumpuan. Persamaan draf gen yang dijana oleh algoritma dianalisis dengan menggunakan Alat Pencarian Penjajaran Tempatan (BLAST). Hasil kajian menunjukkan bahawa QCRO boleh mencari susunan serpihan-serpihan gen dengan betul dan dapat menyelesaikan masalah perhimpunan serpihan gen. Kesimpulannya, penyelidikan ini membentangkan algoritma hiper-heuristik baru untuk menyelesaikan masalah perhimpunan serpihan gen yang dibuat berdasarkan dua algoritma metaheuristik. Algoritma ini boleh mencari susunan serpihan gen yang betul dan menyelesaikan masalah perhimpunan gen.

ABSTRACT

Gene assembly is a technique to construct a gene sequence by referring to gene fragments generated by sequencing machine. The gene fragments are often short and come in large number. As the number of gene fragments increases, the complexity of the problem increases, and this situation produces a wider solution space. To solve the gene assembly problem, the gene fragments need to be arranged in the right order. However, due to the complexity and wide solution space, the accurate solution to this problem is difficult to be found. By looking from the computational perspective, gene assembly problem is considered as nondeterministic-polynomial (NP) problem, where the gene assembly problem can be solved by using metaheuristic algorithms. Metaheuristic algorithms optimize the problem by searching for almost optimal solution. In this research, a hyper-heuristic algorithm is proposed to solve gene assembly problem due to its advantages that overcome the metaheuristic algorithms. This research is conducted based on three objectives. First, to analyze two metaheuristic algorithms, Chemical Reaction Optimization (CRO) and Quantum Inspired Evolutionary Algorithm (QIEA), to solve the problem. Second, a new hyper-heuristic algorithm (QCRO) is developed based on CRO and QIEA. Third, the solutions generated from all three algorithms are evaluated by using statistical analysis. The performance of the algorithms is evaluated by convergence analysis. The similarities of the draft gene sequence generated by the algorithms are analyzed by using Basic Local Alignment Search Tool (BLAST). The findings show that QCRO is competent in finding the right order of the fragments and solving the gene assembly problem. In conclusion, this research presented a new hyper-heuristic algorithm to solve gene fragment assembly problem that is derived from two metaheuristic algorithms. This algorithm is capable of finding the right order of the gene fragments and thus solves the gene assembly problem.

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE** |

[DECLARATION ii](#_Toc475990788)

[ACKNOWLEDGEMENT iv](#_Toc475990789)

[ABSTRAK v](#_Toc475990790)

[ABSTRACT vi](#_Toc475990791)

[TABLE OF CONTENTS vii](#_Toc475990792)

[LIST OF TABLES x](#_Toc475990793)

[LIST OF FIGURES xi](#_Toc475990794)

[LIST OF ABBREVIATIONS xii](#_Toc475990795)

1. [INTRODUCTION 1](#_Toc475990796)

[1.1 Problem Background 1](#_Toc475990797)

[1.2 Problem Statement 3](#_Toc475990798)

[1.3 Research Goal and Objectives 3](#_Toc475990799)

[1.4 Research Scope and Significance 4](#_Toc475990800)

[1.5 Thesis Outline 4](#_Toc475990801)

[1.6 Summary 5](#_Toc475990802)

1. [LITERATURE REVIEW 6](#_Toc475990803)

[2.1 Gene Sequence 6](#_Toc475990804)

[2.2 Overview on Gene Fragment Assembly 8](#_Toc475990805)

[2.3 Sequence Alignment Analysis 9](#_Toc475990806)

[2.4 Metaheuristic Algorithm 11](#_Toc475990807)

[2.4.1 Genetic Algorithm (GA) 12](#_Toc475990808)

[2.4.2 Particle Swarm Optimization (PSO) 13](#_Toc475990809)

[2.4.3 Firefly Algorithm (FA) 14](#_Toc475990810)

[2.4.4 Ant Colony Optimization (ACO) 15](#_Toc475990811)

[2.4.5 Quantum-inspired Evolutionary Algorithm (QIEA) 15](#_Toc475990812)

[2.4.6 Chemical Reaction Optimization Algorithm (CRO) 18](#_Toc475990813)

[2.5 Overview on Several Available Gene Assembly Software 22](#_Toc475990814)

[2.6 Hyper-heuristic Algorithm 26](#_Toc475990815)

[2.7 Research Trends and Directions 28](#_Toc475990816)

[2.8 Summary 30](#_Toc475990817)

1. [RESEARCH METHODOLOGY 31](#_Toc475990818)

[3.1 Introduction 31](#_Toc475990819)

[3.2 Research Framework 31](#_Toc475990820)

[3.3 Dataset for Experiment 34](#_Toc475990821)

[3.4 Problem Formulation 35](#_Toc475990827)

[3.5 Performance Measurement 36](#_Toc475990828)

[3.5.1. Statistical Analysis 36](#_Toc475990829)

[3.5.2. Fitness Convergence 37](#_Toc475990830)

[3.5.3. Validation with BLAST Alignment 37](#_Toc475990831)

[3.6 Summary 38](#_Toc475990832)

1. [DEVELOPMENT OF CRO AND QIEA FOR GENE FRAGMENT ASSEMBLY 39](#_Toc475990833)

[4.1 Overview 39](#_Toc475990834)

[4.2 CRO for Gene Fragment Assembly 39](#_Toc475990835)

[4.3 QIEA for Gene Fragment Assembly 41](#_Toc475990836)

[4.4 Experimental Result 43](#_Toc475990837)

[4.5 Summary 48](#_Toc475990838)

1. [DEVELOPMENT OF HYPER-HEURISTIC 49](#_Toc475990839)

[5.1 Overview 49](#_Toc475990840)

[5.2 Metaheuristic-based Hyper-heuristic Algorithm 51](#_Toc475990841)

[5.3 Convergence Analysis of CRO, QIEA, and QCRO 53](#_Toc475990842)

[5.4 Statistical Analysis 59](#_Toc475990843)

[5.5 Validation with BLAST Analysis 61](#_Toc475990844)

[5.6 Summary 62](#_Toc475990845)

1. [CONCLUSION AND FUTURE WORKS 63](#_Toc475990846)

[6.1 Conclusion 63](#_Toc475990847)

[6.2 Future Works 64](#_Toc475990848)

[REFERENCES 65](#_Toc475990849)

LIST OF TABLES

|  |  |  |
| --- | --- | --- |
| **TABLE NO.** | **TITLE** | **PAGE** |

[**Table 2.1** Elementary reactions and its suggested operators (Lam and Li, 2010a, 2012) 22](#_Toc475993992)

[**Table 2.2** Gap analysis table of available software for gene assembly. 24](#_Toc475993993)

[**Table 3.1** Details of datasets. 35](#_Toc475993994)

[**Table 4.1** Experimental result with CRO. The experiment is tested on various controlled and manipulated variable. The bolded value is the maximum fitness score for each instance. 45](#_Toc475993995)

[**Table 4.2** Experminetal result with QIEA. The experiment is tested on various controlled and manipulated variable. The bolded value is the maximum fitness score for each instance. 46](#_Toc475993996)

[**Table 4.3**  CRO analysis. The CRO algorithm was run for 10 times. Variable *b* is the best fitness score, *w* is the worst fitness score, *m* is the mean of the maximum fitness scores, and *s* is the standard deviation of fitness scores. Parameters: iteration=1000, population size=1000. 47](#_Toc475993997)

[**Table 4.4** QIEA analysis. The QIEA algorithm was run for 10 times. Variable *b* is the best fitness score, *w* is the worst fitness score, *m* is the mean of the maximum fitness scores, and *s* is the standard deviation of fitness scores. Parameters: iteration=1000, population size=1000. 47](#_Toc475993998)

[**Table 5.1** Statistical comparison of CRO, QIEA, and QCRO. The best maximum fitness score of each algorithm is labelled with \*. 60](#_Toc475993999)

**Table 5.2** Best result alignment from CRO, QIEA, and QCRO compared for the similarity with the original sequence. 61

LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **FIGURE** | **TITLE** | **PAGE** |

[**Figure 2.1** A double helix of DNA strands (Zephyris, 2011) 7](#_Toc473794083)

[**Figure 2.2** Global alignment (Mount, 2001). 10](#_Toc473794084)

[**Figure 2.3** Local alignment (Mount, 2001). 11](#_Toc473794085)

[**Figure 2.4** Process flow of QIEA (Han and Kim, 2002) 18](#_Toc473794086)

[**Figure 2.5** Framework of CRO 19](#_Toc473794087)

[**Figure 2.6** Framework of hyper-heuristic (Burke *et al.*, 2003) 27](#_Toc473794088)

[**Figure 2.7** Several framework of hyper-heuristics 29](#_Toc473794089)

[**Figure 3.1**  Resarch framework 33](#_Toc473794090)

[**Figure 3.2** Example of BLAST output. 38](#_Toc473794091)

[**Figure 4.1** Pseudocode of CRO used to solve gene fragment assembly. The parameter involved is population size, iteration, kinetic energy loss rate, initial kinetic energy, alpha, beta, and molecule collision rate, and the value of the parameters is set as 1000, 1000, 0.2, 500, 10, 0.2, respectively. 41](#_Toc473794092)

[**Figure 4.2** Pseudocode of repair algorithm. 43](#_Toc473794093)

[**Figure 5.1** Hyper-heuristic framework modified from FA in Figure 2.7 to fit with algorithm QIEA and CRO criteria 50](#_Toc473794094)

[**Figure 5.2** Framework of hyper-heuristic QCRO 52](#_Toc473794095)

[**Figure 5.3** Convergence analysis graph for instance A 54](#_Toc473794096)

[**Figure 5.4** Convergence analysis graph for instance B 54](#_Toc473794097)

[**Figure 5.5** Convergence analysis graph for instance C 55](#_Toc473794098)

[**Figure 5.**6 Convergence analysis graph for instance D 56](#_Toc473794099)

[**Figure 5.7** Convergence analysis graph for instance E 57](#_Toc473794100)

[**Figure 5.8** Convergence analysis graph for instance F 58](#_Toc473794101)

[**Figure 5.9** Convergence analysis graph for instance G 58](#_Toc473794102)

LIST OF ABBREVIATIONS

|  |  |  |
| --- | --- | --- |
| A | - | Adenine |
| ACO | - | Ant colony optimization |
| C | - | Cytosine |
| CRO | - | Chemical reaction optimization |
| DNA | - | Deoxyribonucleic acid |
| FA | - | Firefly algorithm |
| G | - | Guanine |
| GA | - | Genetic algorithm |
| mRNA | - | Messenger RNA |
| NP | - | Nondeterministic polynomial |
| NWA | - | Needleman Wunsch algorithm |
| OS | - | Operating system |
| PSO | - | Particle swarm optimization |
| QIEA | - | Quantum-inspired evolutionary algorithm |
| RNA | - | Ribonucleic acid |
| SA | - | Simulated annealing |
| SWA | - | Smith-Waterman algorithm |
| T | - | Thymine |
| TSP | - | Travelling salesman problem |
| U | - | Uracil |

**CHAPTER 1**

INTRODUCTION

1. Problem Background

Metabolic engineering is a study of biochemical reactions in biochemical reaction pathways. The fundamental of the metabolic engineering is to analyze cells as the integral units. This study involves pathway synthesis, thermodynamic feasibility, and pathway flux and flux control. One important application of metabolic engineering is to manipulate the yield and productivity of products synthesized by microorganisms ([Keasling, 2010](#_ENREF_20)). To date, various researches that involved the manipulation of metabolic engineering have been carried out extensively.

As the synthetic biology and metabolic engineering area of study partially overlap, a number of studies have presented the used of synthetic biology to facilitate metabolic engineering especially in the applications of industrial biotechnology ([Keasling, 2012](#_ENREF_21)) and industrial microbiology ([Zhang and Nielsen, 2014](#_ENREF_53)). Synthetic biology aims to create new biologically functional parts, modules and systems by utilizing various molecular biology and synthetic deoxyribonucleic acid (DNA) tools together with the use of mathematical methodologies to perform new tasks ([Chandran *et al.*, 2011](#_ENREF_8); [Copeland *et al.*, 2012](#_ENREF_9)). The concepts of synthetic biology follow the concepts of computer engineering hierarchy ([Andrianantoandro *et al.*, 2006](#_ENREF_2); [Chandran *et al.*, 2011](#_ENREF_8); [Kronberger, 2012](#_ENREF_22)).

The hierarchy consists of few layers: physical layer, device layer, and module layer. In this hierarchy, in physical layer, the transistor, capacitors, and resistors represent the DNA, ribonucleic acid (RNA), proteins and metabolites. In the device layer, the biochemical reactions in microorganisms represent the engineered logic gates that perform computations in a computer. In the module layer, a library of biological devices to assemble pathways represented integrated circuits. The modules and their integration into host cells can be modified in a programmatic fashion. Biological devices and modules are often dependable to each other. Hence, when the devices or modules are engineered, it will modify the whole cells itself (Andrianantoandro et al., 2006; Chandran et al., 2011; Kronberger, 2012).

One major focus in synthetic biology is to engineer a complex cellular behavior by assembling and expressing genes that will encodes well – characterized biological components. Each cellular function is carried out by ‘modules’ made up of numerous species of interacting molecules. The modules are separable by function. Insulation of modules allow cell to have many diverse reactions without bring any harm to the cell. Connectivity of modules allows one function to influence another (Ajikumar et al., 2010). Some challenges of metabolic engineering require the use of synthetic biology. For example, metabolic engineering is about designing, engineering and optimizing pathways to produce variety of products. Synthetic biology provides synthetic DNA for the constructed metabolic pathway.

Heuristic refers to experience-based techniques to find or to discover by trial and error of a problem. Metaheuristic method means to find or to discover the problem by using higher level heuristic method and perform better than simple heuristics. Hyper-heuristic method is a search method that includes the integration of machine learning techniques to automate the process of selecting, combining, generating or adapting few heuristics or its components to solve computational search problems.

1. Problem Statement

In genome sequencing, a set of gene fragments is generated from the sequencing machine. The gene fragments itself have several problems. The gene fragments are not in order. The gene fragments order is decided base on the computed overlap score.

Several challenges are identified. The fragments have unknown orientation. The sequence can be read as 5’ to 3’ or 3’ to 5’. If the algorithm is not able to assemble a set of fragments into single contig, the solution is said to have incomplete coverage. The gene fragments may have repeated regions.

The total amount of fragments generated is proportional to the size of original gene. The fragments are not in order, wide solution space, and it is time consuming. Hence, the running time increase with the number of fragments. This problem can be solved by using optimization algorithm.

1. Research Goal and Objectives

The goal of this research is to apply metaheuristic algorithm to solve assembly problem. The following are the objectives of research.

1. To analyze two metaheuristic algorithms, CRO and QIEA to solve gene fragment assembly problem.
2. To develop a new hyper-heuristic algorithm based on CRO and QIEA, called QCRO to solve gene fragment assembly problem.
3. To validate the result from CRO, QIEA, and QCRO using existing gene sequence database.
4. Research Scope and Significance

Several DNA fragment benchmark datasets were used to study the performance of the proposed algorithm for gene assembly problem. The dataset is provided in <http://chac.sis.uia.mx/fragbench/> website by [Mallén-Fullerton *et al.* (2013)](#_ENREF_29). The file format of the dataset is in the form of FASTA and Comma-Separated Values (CSV) for its tabular data. The score of each overlap fragments are presented in matrices form. The method used to optimize the problem is CRO and QIEA. The algorithms were programmed by using programming language Python. Several open source dependencies libraries for Python such as numPy ([Van der Walt *et al.*, 2011](#_ENREF_43)), sciPy, MatPlotLib ([Hunter, 2007](#_ENREF_19)), is used for multidimensional array, numerical routines, and graph plotting purposes, respectively. The experimental result is generated *in silico*.

The significance of the research is addressed as follows. First, new hyper-heuristic algorithm is designed and developed in terms of the computational contribution. The algorithm employs the heuristic operations of CRO and QIEA to increase the search ability in order to find a good quality of overlap order without the being dependent of the problem and parameters tuning. The outcomes of this research may benefit the biotechnology industry. This is due to the contribution of a new approach in solving gene assembly problem. Since the hyper-heuristic algorithm is problem-independent, the algorithm can be reused and applied for another computational problem such as TSP ([Burke *et al.*, 2010](#_ENREF_5); [Burke *et al.*, 2003](#_ENREF_6); [Burke *et al.*, 2013](#_ENREF_7)).

1. Thesis Outline

This thesis is composed of few chapters.

* Chapter 1: This chapter provides the introduction of the research which includes the research background, problem statement, goal and objectives, and scope and significance of the study.
* Chapter 2: This chapter provides the literature review of the research. It starts with the overview of gene sequence. Then, it proceeds with the discussion of gene fragment assembly problem. Next, the chapter continues on the overview of sequence alignment analysis and discussed the potential metaheuristic algorithms should be used on the research. The research trends and directions on the related issues discussed.
* Chapter 3: This chapter presents the overview of the research operational framework, the flow of the thesis development, and the metaheuristic algorithms, programming language and software used in this study.
* Chapter 4: This chapter discusses the design and development of Chemical reaction optimization algorithm and Quantum-inspired evolutionary algorithm to solve gene fragment assembly problem.
* Chapter 5: This chapter discusses the design and development of hyper-heuristic algorithm to solve gene fragment assembly problem.
* Chapter 6: Conclusion of the research. This chapter also includes contribution of the works and future plans

1. Summary

In conclusion, synthetic biology area of study follows the concepts and principle of engineering to produce new biological parts, systems, or devices with improved functionality. The use of computational tools in this area of study provides necessary knowledges to solve problem encounters. One problem can have numerous possible solutions available. To find the best solution out of the possible solution, metaheuristic algorithm is used. The metaheuristic algorithm can come in many forms and the result is varied. The metaheuristic algorithm is also problem dependent. Hence, selecting the correct algorithm is necessary to solve specific problem.

**CHAPTER 2**

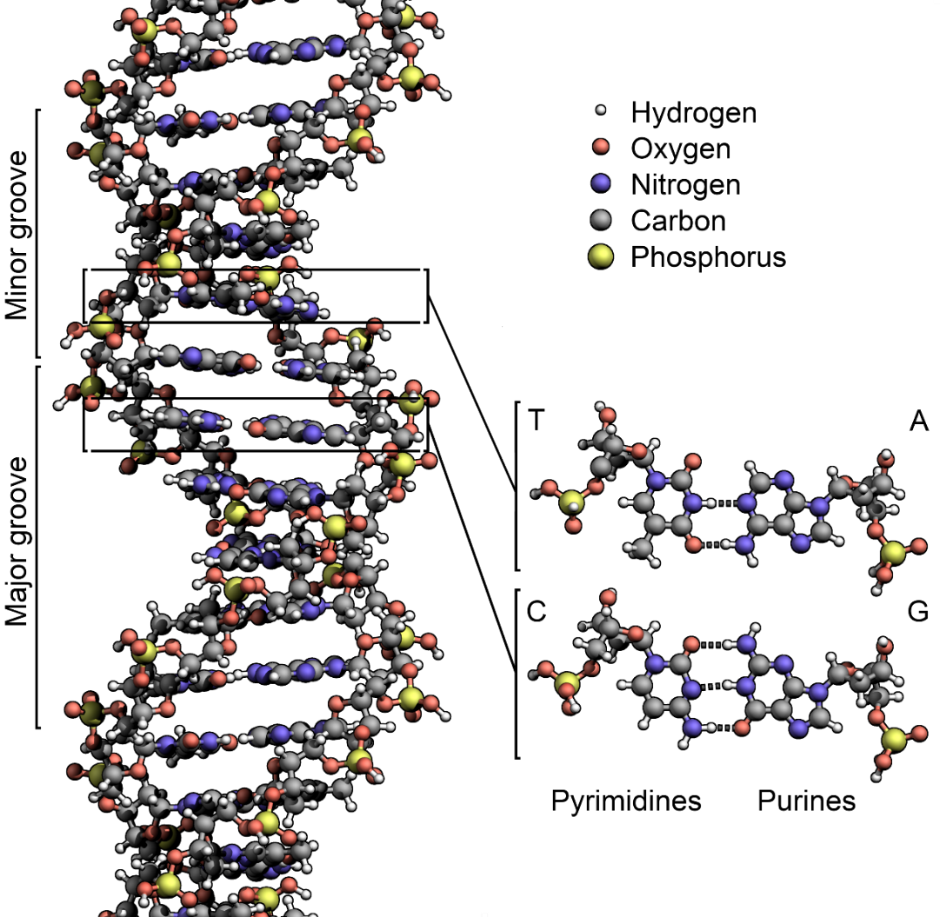
LITERATURE REVIEW

1. Gene Sequence

Deoxyribonucleic acid (DNA) contains genetic instructions that are encoded to make each living organism unique. DNA is found inside the nucleus of a cell. DNA is made up of four chemical bases. These bases are adenine (A), guanine (G), cytosine (C), and thymine (T). Base A paired with base T while base C paired with base G, to form a unit called base pairs. Each base has sugar molecule and phosphate molecule attached to it. The combination of all three molecules called nucleotide.

Figure 2.1 shows a graphic representation of a double helix of DNA strands. The nucleotides are joined together to one another by a chain called phosphodiester bonds in between sugar of previous nucleotide and phosphate of the next nucleotide, then causing an alternating sugar-phosphate backbone. To make double-stranded DNA, the nitrogenous bases bound together by hundred bonds. Each carbon in the DNA’s sugar backbone indicated by numbers. The carbon with phosphate group attached is called 5’ (five prime) and carbon which have a hydroxyl group attached to it is called 3’ (three prime). The DNA molecules attached to each other in 5’-3’ direction.

Genetic information flow from DNA to RNA to protein. This concept is known as central dogma. Several processes are involved. First, the translation process. This process occurs to synthesis DNA. Double stranded DNA template is transcripted into single-stranded messenger RNA (mRNA) molecule. The mRNA is complementary to the DNA strand. Base A, T, G, and C of DNA strand are specified as Uracil (U), A, C, and G, respectively. The process has three phases, which are initiation, elongation, and termination. Second, the translation process. The mRNA produced in transcription process is translated to produce a polypeptide or specific amino acid chain based on the arrangement of the bases in mRNA. The amino acid is specified by codon. The codon is represented by 3 base pairs of DNA. Some codon codes for the same amino acid. The amino acid chain will eventually form a protein.



**Figure 2.1** A double helix of DNA strands (Zephyris, 2011)

1. Overview on Gene Fragment Assembly

Next generation sequencing produces small fragments known as short reads. A large number of small fragments makes the assembly process more complicated. Gene fragment assembly is an important step in gene sequencing. The assembly process influenced the next process, depends on its assembly accuracy. From a DNA sequence, a set of small gene fragments are randomly sliced.

Several basic terminology are required to understand the process. A fragment is a short gene sequence derived from the original gene sequence, with length less than 1000 base pairs. A shotgun data is a set of gene fragments. A prefix is a substring comprising the first *n* characters of a fragment. A suffix is a substring comprising the last n characters of a fragment. An overlap is a similar sequence between the suffix of one fragment and the prefix of another fragment. A layout is an alignment of fragments based on the overlap order. A contig is an arrangement of contiguous overlapping fragments. A consensus is a sequence obtained from the layout.

The distribution of the coverage is calculated to measure the quality of consensus. The coverage measure the redundancy of the fragment data and is defined as the number of fragments at that position. Equation (2.1) shows the calculation of the coverage. Variable *n* denotes the number of fragments.

 (2.1)

The gene sequencing process consists of three phases. First, the overlap phase. In this phase, overlapping fragments is found by finding the best or longest score between the suffix of one sequence and the prefix of another. All possible pairs of fragments is compared to find their similarity and computed as overlap score.

Second, the layout phase. In this phase, fragment order is determine based on the calculated overlap score from the overlap phase. Several problems affects the process in this phase. The fragments can have unknown orientation where it can be read as 5’ to 3’ or 3’ to 5’. Base call error happens due to experimental errors in the electrophoresis procedure. There are three types of errors such as substitution, insertion, and deletion errors. Incomplete coverage happens if the algorithm is not able to assemble a given set of fragments into single contig. Repeated regions appear two or more times in the target gene. Contamination happens because of the incomplete purification of the fragment from the vector DNA.

The third phase is consensus phase. In this phase, gene sequence is derived from the layout. Gene fragment assembly is included as Non-deterministic polynomial (NP) – problem ([Pevzner, 2000](#_ENREF_37)).

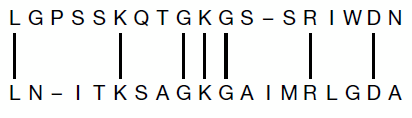
1. Sequence Alignment Analysis

Sequence alignment is a procedure to compare two or more sequences by searching character patterns that are in the same order in the sequences. Dynamic programming is a computational method used to execute sequence alignment analysis by aligning two proteins or gene sequences. The method provides the best alignment between two sequences by comparing all pair of characters in two sequences and generate alignment. This alignment includes matched and mismatched characters, and gaps in two sequences so that the sequences can have matches at maximum possible. The method construct a dot matrix analysis that display any possible sequence alignments as diagonals on the matrix. Simple modification on dynamic programming can be made to construct global and local sequence alignment analysis ([Mount, 2001](#_ENREF_32)).

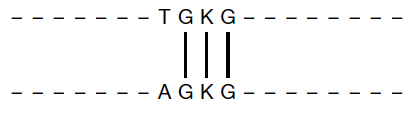
An example of global alignment is Needleman-Wuncsh algorithm (NWA). Figure 2.2 shows how NWA search the sequences for similarity. It search the entire sequence length to include as many characters as possible from the beginning to the end of the sequence. NWA best employed when a pair of sequences have the same size. If the sequences have different size, mismatched ensued at the end of the alignment ([Mount, 2001](#_ENREF_32)).

An example of local alignment is Smith-waterman algorithm (SWA). Figure 2.3 shows how SWA search the sequences for similarity. SWA calculates the overlap matrices of the generated sequence. Unlike global alignment, local alignment stops at the end of region with strong similarity. In Figure 2.3, dashes in the sequences indicates the characters in the sequences that is not included in the alignment.

The parameter settings that is used by SWA is 1 for a match, -3 for a mismatch, and -2 for a gap ([Engle and Burks, 1994](#_ENREF_13); [Mallén-Fullerton *et al.*, 2013](#_ENREF_29)). SWA executes in both orientation 3’ to 5’ and 5’ to 3’, regular and reverse compliment since the proper orientation is unknown. The actual length of overlap is calculated when the maximum score is found. The overlap length is collected and stored into a matrix of overlap score. The task is time-consuming but simple to implement ([Mallén-Fullerton *et al.*, 2013](#_ENREF_29)).



**Figure 2.2** Global alignment ([Mount, 2001](#_ENREF_32)).



**Figure 2.3** Local alignment ([Mount, 2001](#_ENREF_32)).

1. Metaheuristic Algorithm

Metaheuristic means to find new strategies to solve problems. It allows to solve large size problem instances by giving the best solution in reasonable time. A global optimal solution is not guarantee to be found. Metaheuristic algorithms have been used in computational biology field (Banga, 2008). The practice varied from synthetic biology field to metabolic engineering and many others. Several metaheuristic algorithms were analyzed for this research.

One way to solve gene fragment assembly problem is by using overlap-layout-consensus (OLC) method as mentioned in Section 2.2. OLC method consists of three main phases. First, the overlap phase. In the overlap phase, each gene fragment is compare to each other in both forward and reverse orientation. Dynamic programming is used to calculate the overlap score. Several research were known to use Smith-Waterman algorithm to calculate the overlap score ([Engle and Burks, 1994](#_ENREF_13); [Mallén-Fullerton *et al.*, 2013](#_ENREF_29)). The overlap score is collected in a matrix.

Second, the layout phase. In the layout phase, the focus is to find the right order of overlapping gene fragments to form a consensus sequence that resembled the original DNA sequence. Several difficulties are mentioned in section 2.2. Hence, finding the right order is very important to generate a good consensus sequence. To find the right order, metaheuristic algorithm can be applied in this phase to solve the problem ([Luque and Alba, 2005](#_ENREF_27)). Several metaheuristic algorithms have been used to solve the gene assembly problem ([Huang *et al.*, 2014](#_ENREF_18); [Mallén-Fullerton and Fernandez-Anaya, 2013](#_ENREF_28); [Parsons *et al.*, 1995](#_ENREF_36); [Wetcharaporn *et al.*, 2006b](#_ENREF_47)).

The gene assembly problem is a NP-hard problems such as the travelling salesman problem (TSP). To solve NP-hard problem, permutation representation is used. in this case, each gene fragment is assigned with unique integer ID. To get a feasible solution, two condition must be satisfied. First, all fragments must be in the order. Second, the gene fragments should not appears more than once. Feasible solution is a solution that follows all the constraint. An optimal solution is a feasible solution where the objective function reaches its maximum or minimum value. In this case, the optimal solution should reaches a maximum value of objective function. The objective function is stated in section 3.4.

At present, several well-known metaheuristic algorithms are applied to solve gene assembly problem. Two metaheuristic algorithms, CRO and QIEA, are discussed in this section. At present, these two algorithms have not been applied to solve the problem.

1. Genetic Algorithm (GA)

This algorithm has been implemented in various tool in synthesis biology ([Radenbaugh, 2008](#_ENREF_39)). GA concepts follow the process of natural evolution such as inheritance, mutation, selection, and crossover. Biological evolution became the theoretical framework for GA ([Mitchell, 1998](#_ENREF_31)). A population of chromosome moves to a new population by natural selection. Each chromosome contains gene and each gene is an instance of specific allele.

In computational term, the genes can be single bits or blocks of adjacent bits that set a specific element of the candidate solution. Originally, GA use binary representation as the genes. Now, the population of gene is encoded using permutation, trees, and other. GA applied crossover and mutation to choose which chromosome is allowed to reproduce. Crossover operator combines the feature of two parent chromosomes to generate offspring by swapping segments of the chromosome. Mutation operator altered one or more genes of a chromosome according to probability and the mutation rate.

GA has been implemented to gene assembly problem. In a research by [Parsons *et al.* (1995)](#_ENREF_36), GA is tested with several combination of operators such as crossover, swap, inversion, and transposition. The success of these operators influence the approach of solving permutation problems using metaheuristic algorithms. In another research by [Luque and Alba (2005)](#_ENREF_27), compared to other heuristic algorithm, the performance of GA was not very good as the algorithm tend to converge prematurely.

1. Particle Swarm Optimization (PSO)

PSO is a population-based stochastic algorithm that follows swarm intelligence concept. This algorithm follows the movement of organisms in a bird flock or fish school. Unlike evolutionary algorithms, all the particle or individual survive to the end of trial. Computationally, the swarm is the population while candidate solution is the particles. The particle moves around search-space based on three principles. First, the particle change its condition to keeps its inertia. Second, the particles change its condition to keep optimist position. Third, the particles change its condition according to the swarm’s optimist position. Each particle is assigned with randomize velocity and the potential solution. Each particle stays on its coordinates in the search-space according to the best solution that it has achieved so far. In every step, the velocity of each particle changes towards the best solution ([Bai, 2010](#_ENREF_3)).

Three problem-dependent parameters are identified. First, the inertia of a particle. The inertia controls the exploration of the algorithm. Larger values of inertia assist to global behavior. Smaller values of inertia leads to local behavior. The second and third parameter are the ‘trust’ parameters, *c­*1 and *c*2. Parameter *c­*1 shows the confidence of the current particle. Parameter *c­*2 shows the confidence of the current particle has in the swarm ([Eberhart and Shi, 2001](#_ENREF_11)). To prevent premature convergence, some researcher use craziness operator to add randomness to the swarm while other researcher conclude that the additional randomness is not necessary ([Venter and Sobieszczanski-Sobieski, 2003](#_ENREF_44)). A memetic PSO is applied to solve gene fragment assembly in a research by ([Huang *et al.*, 2014](#_ENREF_18)). The algorithm reported to be time consuming.

1. Firefly Algorithm (FA)

FA is follows the behavior of fireflies and their flashing patterns. Basically, FA follows three rules. First, fireflies are unisex. Hence, they are drawn to each other regardless of gender. Second, the attractiveness is proportional to the brightness of the firefly. For example, between two fireflies, the less bright firefly will be attracted to and move to the brighter one. If there is no brighter firefly between these fireflies, the firefly moves randomly. Third, the landscape of the objective function will determine firefly brightness. FA have several advantages compared to other algorithms. First, FA will subdivided into subgroups automatically. Hence, each group can swarm together and find the local optimum. The optimum solution can be found in any of these subgroups. Second, the subdivision allows the fireflies to find all local optima simultaneously. This advantage makes the algorithm suitable for highly non-linear, multimodal optimization problems. Tuning the parameters in FA to control the randomness can affect the speed of the convergence. Hence, fine-tuning FA parameters is a very crucial step. This advantage makes it suitable for several mathematical problem ([Yang, 2010](#_ENREF_49)).

In a work by ([Vidal and Olivera, 2014](#_ENREF_45)), a modified FA is used to solve gene fragment assembly. The experiment is carried out to focus on the challenge of reducing computational time and developing algorithms by using Graphic Processing Units (GPUs).

1. Ant Colony Optimization (ACO)

ACO is based on the foraging behavior of ant species. The objective is to find the shortest path between their nest and food source by secreting the pheromone on the ground while walking. Artificial agents represent individual populations that build solutions by moving on a graph-based representation of the problem ([Dorigo and Birattari, 2010](#_ENREF_10)). The agent’s movement defines the solution components that are added to the solution. Similar to the real ant, artificial agents secrete pheromone that is proportional to the quality of the solution. The artificial pheromone used as probabilistic model for the artificial ants to make decision while constructing a solution ([Dorigo and Birattari, 2010](#_ENREF_10)). In an experiment by [Wetcharaporn *et al.* (2006b)](#_ENREF_47), ACO is used to measure its performance in solving single-contig problem against nearest neighbor heuristic where ACO outperforms the nearest neighbor heuristic search.

1. Quantum-inspired Evolutionary Algorithm (QIEA)

QIEA concepts is based on the principle of quantum computing, such as quantum bit and superposition of states ([Han and Kim, 2002](#_ENREF_16); [Zhang, 2011](#_ENREF_52); [Zhou *et al.*, 2005](#_ENREF_54)). The individual is represented as Q-bit which is defined as the smallest unit of information. Figure 2.4 shows the process flow of QIEA. The following are the solution involves in the process. A Q-bit is defined with a pair of numbers as follows

 (2.2)

where . gives the probability of Q-bit to be found in the “0” state and gives the probabilities of Q-bit to be found in the “1” state. A Q-bit can be in the “1” or “0” state, or in the linear superposition of the two. QIEA have three stages, initialization, iteration, and termination. QIEA maintains its population size. In initialization stage, a Q-bit population is generated at generation *t*.

 (2.3)

where *t*=0, *n* is the population size. Each is defined as

 (2.3)

where *m* is the number of Q-bits, *j* =1,2,…,*n.* To initialize , and are initialize as. Variablerepresents the linear superposition of all possible states with the same probability

 (2.4)

where *Xk* is the *k*th state represented by the binary string . Variableis either 0 or 1 based on the probability of and , respectively.

*Q*(0) is observed to form binary solutions

 (2.5)

at generation *t*=0. , *j*=1,2,…,*n*, is a binary string with length of *m*, based on the probability of and .

Next, each binary solution is evaluated. The best solution is collected and stored into

 (2.6)

For initial generation, is equal to. In the iteration stage, is formed by observing. Each binary solution in is evaluated for the fitness value. Next, Q-bit individuals in is updated by using Q-gate. Generally, the following rotation gate is used as a Q-gate in QIEA ([Han and Kim, 2002](#_ENREF_16); [Zhang, 2011](#_ENREF_52); [Zhou *et al.*, 2005](#_ENREF_54)).

 (2.7)

where, is a rotation angle of each Q-bit. The magnitude of effect the speed of convergence ([Han and Kim, 2002](#_ENREF_16); [Zhang, 2011](#_ENREF_52); [Zhou *et al.*, 2005](#_ENREF_54)).

If the magnitude of is too big, the solution may converge prematurely to a local optima. The recommended value foristo. The magnitude is problem dependent. The best solution among and are selected and stored into. If the best solution in is better than *b,* then the best solution in is stored as best solution *b*. If a migration condition is satisfied, global or local migration occurs among and *b* ([Han and Kim, 2002](#_ENREF_16); [Zhang, 2011](#_ENREF_52); [Zhou *et al.*, 2005](#_ENREF_54))

…

Q-gate

Q-gate

Update

Update

…

…

Observation

Observation

Selection btw. and

Selection btw. and

Local

migration

Local

migration

Local

migration

Selection and global migration









**Figure 2.4** Process flow of QIEA (Han and Kim, 2002)

QIEA is claimed to have a good balance on exploitation and exploration. Hence, QIEA is said to perform quite well even with only having one individual to search for the solution. However, in a study by ([Feng *et al.*, 2006](#_ENREF_14)) to solve a traveling salesman problem (TSP), the algorithm cannot comprehend with huge solution space.

1. Chemical Reaction Optimization Algorithm (CRO)

CRO is constructed based on the behavior of chemical molecule toward chemical reactions to reach a low energy stable state ([Lam and Li, 2010a](#_ENREF_23), [2010b](#_ENREF_24); [Li *et al.*, 2015](#_ENREF_26)). Its main components are molecules and chemical reactions. Figure 2.5 shows the basic framework of CRO. Each molecule have basic attributes such as molecular structure *m*, potential energy, and kinetic energy ([Lam and Li, 2010b](#_ENREF_24); [Yu *et al.*, 2015](#_ENREF_50)). The molecular structure is the possible of a problem. The potential energy *PE* is the objective function value based on the molecular structure. The kinetic energy *KE* is a number that

Intermolecular

collision?

Molecule selection

Decomposition?

Synthesis?

Molecule selection

On-wall collision

Decomposition

Intermolecular ineffective collisions

Synthesis

Check for new minimum point

Stopping criteria

satisfied?

Obtain the global

minimum point

Yes

No

Yes

No

Yes

No

No

Yes

**Figure 2.5** Framework of CRO

measured the tolerance of a system to accept a molecule while having a worse solution compared to other molecules.

 (2.8)

In CRO, four types of chemical reactions takes place to explore the solution space and to redistribute energy of the molecules and energy buffer. CRO consists of three stages: initialization, iteration, and the final stage.

In initialization, a random solution is assigned to molecular structure of a molecule. In iteration stage, chemical reactions takes place to manipulate or search the solution space. The reactions divided into two major actions, single and multi-molecular reactions. The single molecule reaction involved only one molecule and improved by on-wall ineffective reaction collision and decomposition ([Abdullah *et al.*, 2013](#_ENREF_1); [Lam and Li, 2010a](#_ENREF_23), [2010b](#_ENREF_24)). The multi-molecular reactions involved two or more molecules and improved by synthesis and inter-molecular ineffective collision. An on-wall collision occurs when a molecule hit a wall of the container and bounce away as a single unit. The molecular structure *m* of the molecule slightly change into. The *KEm* value change as the energy withdrawn to the surroundings energy. Hence, we get the following *KEm’*.



 (2.9)

where . The *KELossRate* is the system parameter that limit the maximum percentage of KE loss at a time. The lost energy is stored in energy buffer. In decomposition reaction, a molecule with molecular structure *m* hits a wall and breaks into several molecules with molecular structure and , respectively. The generated molecular structure are independent and very different from the existing one. If the original molecule have enough energy to donate, as in

 (2.10)

the change is allowed. New variable *temp1* is assigned to get and .

|  |  |
| --- | --- |
|  | (2.11) |
|  | (2.12) |
|  | (2.13) |

where . In multi-molecule reactions, two or more molecules are involved to be improved by two reaction, synthesis and inter-molecular collision. In synthesis, two molecules with molecular structure, and , respectively, hit each other and combine together, forming new molecule with very different molecular structure from and . Molecule accepted only if

 (2.14)

Kinetic energy for molecule constructed as follows

 (2.15)

is large compared to the and  as is expected to have similar value to and  ([Lam and Li, 2010a](#_ENREF_23), [2012](#_ENREF_25)). Hence, molecule  have the ability to escape local minimum in the next chemical reaction. An inter-molecular ineffective collision involve two molecules, with molecular structure and , respectively, hit each other and bounce away. As the result, the molecules have a new molecular structure and , respectively. The changes is accepted only if

 (2.16)

The kinetic energy for molecular structure and constructed as follows

 (2.17)

 (2.18)

 (2.19)

where . The chemical reactions is represented by specific operator that is suitable to the problem. Lastly, the algorithm terminates and the best solution is found. Table 2.1 shows the four elementary reactions in CRO and its suggested operators that is suggested in (Lam and Li, 2010a, 2012).

**Table 2.1** Elementary reactions and its suggested operators (Lam and Li, 2010a, 2012)

|  |  |
| --- | --- |
| Reactions | Operators |
| Decomposition | Half-total charge, Gaussian perturbation with reflection, circular shift |
| On-wall ineffective collision | Two-exchange |
| Synthesis | Probabilistic select, crossover |
| Inter-molecular ineffective collision | Two-exchange, Gaussian perturbation with reflection |

1. Overview on Several Available Gene Assembly Software

Several software have been developed for gene assembly purpose. Some of the software have the same reconstruction process as OLC method. The process involves generating overlap scores using pairwise alignment, merging as many fragment as possible at layout phase, and finally creating consensus from the merged fragments. Some of the software is shown in Table 2.2.

One of the famous gene assembler is PHRAP. PHRAP uses Smith-Waterman-Gotoh algorithm for pairwise alignment. Then, it aligns the data based on the overlapping score into contigs with the help of PHRED quality score ([Bastide and McCombie, 2007](#_ENREF_4)). PHRED quality score is a quality score that measure the quality of the bases generated by automated DNA sequencing. PHRAP uses the quality score to create high quality contig sequences. PHRED quality score helps to prevent the incorrect overlapping of repeat sequences by showing the unique regions outside of repeat unit that do not match. However, high identity repeat sequences will still manage to be overlapped.

Velvet assembler uses de Bruijn graph to solve gene assembly problem for a very short reads, around 25-50 bp. Velvet remove errors and resolve a large number of repeats. Instead of overlap score, Velvet uses *k-mers­*, the length of all possible subsequences in a read. Velvet encounter the main bottleneck in term if time and memory where the software have to construct a graph that will requires memory for data structures. Velvet uses Tour bus algorithm to remove redundant edge in de Bruijn graph that will raise time complexity. Velvet uses Breadcrumb algorithm to resolve repeats with short read pairs. The computational time cost of the algorithm is depend on the number of nodes as the test is run for every long node ([Zerbino and Birney, 2008](#_ENREF_51)).

**Table 2.2** Gap analysis table of available software for gene assembly.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Overlap | Layout | Consensus | Features | Drawbacks |
| PHRAP | Smith-Waterman-Gotoh. | Based on the overlapping score with help of PHRED quality score. | Based on contigs. | Uses PHRED quality score. | Repeat sequences still overlapped. |
| Velvet | *k-mers*. | De Bruijn graph. | Breadcrumb algorithm. | Remove error, resolve a large number of repeats. | Require huge computational time and memory |
| TIGR | Smith-Waterman, *k-mers*. | Concurrent assembly. | Merge fragments with current consensus sequence. | Overcome repeat region, chimeras, sequencing error | Some nonrepeat region labeled as repeat region. |
| Celera | <6% overlap is repeat-induced. | Unitigger, scaffolds. | Based on consensus metric and base calls. | Repeat resolution with rocks, stones, pebbles. | Undercount base differences in repeat reconstructions, sequence is under annotated as repeat, over counting individual base differences in non-repetitive sequence. |
| Phusion | *k-mers.* | PHRAP for contigs generation. | Based on quality values. | Each clusters treated independently. | Computational time |
| Minimus | Pairwise alignment. | Overlap graph, reduction steps. | Based on overlap graph after reduction steps. | Fewer assembly error. | Not stated. |

TIGR assembler was developed to overcome major problem in assembly project, such as repeat regions, chimeras, and sequencing errors. TIGR compare each fragments with Smith-waterman algorithm for the overlap score. To help speed up the pairwise comparison, TIGR locates all *k-mers* oligonucleotides shared between fragment pairs. The merging of fragments is completed by using a modified Smith-Waterman algorithm to evaluate an overlap between current assembly and a fragment, where the current assembly become the consensus sequence in the end of the process. TIGR identifies repeat region by using clone length constraint to assembly long repeat regions. However, due to the nature of pairwise comparison method, some nonrepeat region fragments are labeled as repeat fragments ([Sutton *et al.*, 1995](#_ENREF_42)).

Celera assembler introduce a concept of solving repeats with rocks, stones and pebbles. In the overlap stage, each fragment is compared. The overlaps that is less than 6% differences is considered repeat-induced. Unitigs is a collections of fragments whose arrangement is uncontested by overlaps. The unitigs then merge into scaffold. Celera assembler solve repeat resolution with the concepts of rocks, stones, and pebbles. the rocks, stones, and pebbles is the repeat reads with different levels of repeat resolution that is going to fill the gaps between scaffolds. Finally, the consensus sequences is generated according to consensus metric and base calls ([Myers *et al.*, 2000](#_ENREF_34)).

For Phusion assembler, the software prepare the data by clipping and screening the reads to remove contaminants. Next, it groups together the reads that share low copy of numbers into clusters. These cluster will assembled its reads independently. Next, PHRAP is used iteratively on each cluster to generate contigs. Then, the contigs are joined together based on the shared reads and sequence overlap. Next, a scaffold of joined contigs is generated based on the read pair information. Finally, the scaffold and contig undergoes contamination screening. This assembler is challenged in term of computational time because of the use of PHRAP iteratively on each cluster ([Mullikin and Ning, 2003](#_ENREF_33)).

Minimus is built from 3 modules of AMOS assembly package, which is hash-overlap, tigger, and make-consensus. Instead of using quality data for overlap, it uses the quality data to trim reads and to compute the consensus from the multiple alignment. MINIMUS have several stages. First, the input stage. The reads are loaded into the AMOS bank. Second, the overlap stage. The overlap between the reads is computed by using pairwise alignment. Third, the unitigger stage. In this stage, an overlap graph is constructed. Three reduction steps are involves. First, the removal of containment edges. If a read is contained within another reads, the read is removed from the graph. Second, transitive reduction. Third, unique-join collapsing where every simple path in the graph are collapsed into single vertex. The vertex represents an individual unitig. Finally, in the consensus stage, the consensus sequences is generated ([Sommer *et al.*, 2007](#_ENREF_41)).

1. Hyper-heuristic Algorithm

Hyper-heuristic methods are used to choose the suitable low level heuristic out of a set of heuristics to conduct the search ([Ozcan *et al.*, 2008](#_ENREF_35)). The decision of selecting suitable heuristic is based on independent measurement. Hyper-heuristic is considered as an alternative to metaheuristic algorithm as hyper-heuristic does not need fine tuning of parameters. Often, a metaheuristic is a problem dependent and certain knowledge of the problem is needed. Unlike metaheuristic, hyper-heuristic does not have any knowledge of the problem domain. It only have knowledge of the set of heuristics used in hyper-heuristic. Hence, applying different problem domains to hyper-heuristic is simple as researcher only have to modify the set of heuristics and evaluation function. An advantage hyper-heuristic provide to researcher is it can be developed by non-specialized programmer with no experience of problem domain. This method can be applied to variety of problem domains ([Ozcan *et al.*, 2008](#_ENREF_35)).

Figure 2.6 shows the general framework of hyper-heuristic. The domain barrier make sure that no knowledge cross the barrier. Hence, hyper-heuristic do not know what is the problem domain. It only know how much low level heuristic it have.

Hyper-heuristic process have two phases: heuristic selection and move acceptance. The combination of heuristic selection and move acceptance mechanisms is varied. However, there is no study that shows the performance comparison of difference mechanisms. In a research by [Ozcan *et al.* (2008)](#_ENREF_35), several heuristic selection methods and acceptance criteria were mentioned and tested to produce comprehensive analysis of its performance. In selection phase, a heuristic is selected and applied to a solution to create a new solution. In the move acceptance phase, the method accepts or rejects the new solution based on acceptance method.

Hyper-heuristic

Domain Barrier

Set of low-level heuristics

…

Evaluation function

Non-domain data flow

Non-domain data flow

**Figure 2.6** Framework of hyper-heuristic ([Burke *et al.*, 2003](#_ENREF_6))

Figure 2.7 shows several frameworks of hyper-heuristic method. Heuristic A and B is the heuristic methods to be selected and applied for heuristic selection phase. In move acceptance/rejection phase, solutions is either accepted or rejected to go through the heuristic selection phase. For figure 2.7, understand that each framework is a loop and heuristic A and B is only an example. In reality, more heuristics can be applied.

Framework FA is a traditional framework of hyper-heuristic. In a set of heuristics, any heuristic can be chosen at each step. In the other frameworks, the heuristics operate separately for a better used of diversity.

In framework FB, if heuristic A is selected and applied, heuristic B will be applied to the current solution. Else, the solution undergoes move acceptance mechanism. This framework guaranteed heuristic B is applied in each step. Let say that heuristic C, D, E, and F exist in framework FB. If heuristic C is selected and applied, heuristic C will be applied as well. If heuristic E is selected and applied, heuristic F will be applied as well.

In framework FC, heuristic C is applied in each step regardless of what heuristic selected in the heuristic selection step. In framework FD, the heuristic selection mechanism selects its heuristic by evaluating the performance of previous heuristic.

Although with all the advantages that hyper-heuristic have, NFL still applied to hyper-heuristics only if it applied to low-level search hierarchy. However, free lunch is also possible if at each level of search hierarchy, the heuristics are evaluated using performance measures that expose the differences at the level immediately below ([Poli and Graff, 2009](#_ENREF_38)).

1. Research Trends and Directions

Gene assembly problem attempt to reconstruct the original gene sequence out of a huge number of gene fragments of the same sequence. A number of well-known software that provide the solution were identified as in Table 2.2. The OLC method is a popular approach to solve gene assembly problem. This method use heuristic and metaheuristic algorithms to solve the difficult part of the method, which is finding the order of gene fragments. This method have been proven to be competent to solve the problem. Some example of the metaheuristic algorithms used for this problem are GA ([Luque and Alba, 2005](#_ENREF_27)), PSO ([Huang *et al.*, 2014](#_ENREF_18)), FA([Vidal and Olivera, 2014](#_ENREF_45)) , and ACO ([Wetcharaporn *et al.*, 2006a](#_ENREF_46), [2006b](#_ENREF_47)).

|  |  |
| --- | --- |
| Select/Apply  Accept/Reject  Heuristic A and B  FA | Select/Apply  Apply Heuristic C  Heuristic A, B  FC  Accept/Reject |
| Select/Apply  Apply Heuristic B  Heuristic A and B  FB  Accept/Reject  Heuristic A applied?  No | Select/Apply  Accept/Reject  Heuristic A, B  FD  Select/Apply  Accept/Reject  Heuristic B, D |

**Figure 2.7** Several framework of hyper-heuristics

1. Summary

Gene assembly is a technique to reconstruct the original gene sequence by referring to a number of gene fragments of the same sequence. The problem can be solve with the help of computational method. Table 2.2 shows some software that uses the computational method to solve gene assembly problem while in some research as mentioned in Chapter 2 uses a simple heuristic to metaheuristic method. All these method is mainly used on the overlap phase of OLC method. These methods are problem dependent. Hence, a new hyper-heuristic algorithm is introduced to avoid the dependency and it does need fine tuning of parameters.

**CHAPTER 3**

RESEARCH METHODOLOGY

1. Introduction

In this chapter, research methodology is discussed. First, the research framework is presented. Next, the dataset used in this research is discussed. Afterward, problem formulation and performance measurement is discussed.

1. Research Framework

Figure 3.1 shows the research framework that consist of six phases. In phase 1, several research on gene fragment assembly was analyzed. The research interest is focused on overlap-layout-consensus method for gene fragment assembly. Based on the previous analysis, several metaheuristic algorithms were compared to choose the suitable algorithm for this research. Since that several metaheuristic algorithms have been used in gene fragment assembly ([Huang *et al.*, 2014](#_ENREF_18); [Luque and Alba, 2005](#_ENREF_27); [Vidal and Olivera, 2014](#_ENREF_45); [Wetcharaporn *et al.*, 2006a](#_ENREF_46), [2006b](#_ENREF_47)), in this research, CRO and QIEA were selected to be applied to gene fragment assembly. The research problems were identified.

In phase 2, a suitable dataset is collected based on the problem. Several research uses the same dataset as it is easy to retrieve and available all the time. In phase 3, CRO and QIEA algorithm is developed to fulfill the first objective. Both algorithms were developed by using programming language Python. CRO algorithm were developed based on a research paper produced by [Lam and Li (2012)](#_ENREF_25) as the processes were explained very details. QIEA algorithm were developed based on a research paper produced by ([Han and Kim, 2002](#_ENREF_16); [Han, 2003](#_ENREF_17)). Once the development is done, several experiments was carried out to apply both of the algorithm to gene fragment assembly. The result was collected for validation in phase 5. The experiment was conducted by using a computer with Mac OS X operating system, with 12GB RAM.

In phase 4, a new hyper-heuristic algorithm that is called QCRO were developed based on CRO and QIEA algorithm to fulfill the second objective. The algorithm was developed to overcome the limitation of CRO and QIEA. The algorithm was developed by using programming language Python. Once the development is done, an experiment was carried out to apply the new hyper-heuristic algorithm QCRO to gene fragment assembly. The result was collected for validation in phase 5. The experiment was conducted by using a 12GB RAM computer with Mac OS X.

In phase 5, a comparative study is conducted. The result from phase 3 and phase 4 is compared to each other. First, the validation include the statistical analysis of result from CRO, QIEA, and QCRO experiment. Second, the fitness convergence.

Third, validation with BLAST analysis. The draft gene assembly after the best result collected were compared with databases in BLAST. The OLC method for gene assembly is famously used similar to TSP problem. The fragments are ordered based on the overlap score to obtain the highest fitness score. Next, the gene fragment sequences are arranged based on the ordered fragments. In this step, a multiple sequence alignment is applied to obtain the draft gene sequences out of the ordered fragments. Next, the consensus sequence is taken to be aligned with non-redundant BLAST database for validation on sequence similarity. In phase 6, the result is documented and reported in this thesis.

**Phase 1: Literature study**

* Analyze previous works on related research interest.
* Analyze and comparison of the metaheuristic to be used in this research.
* Identify the research problems needed to be solved.

**Phase 2: Data collection**

* Data collection and gathering information required for the dataset.

**Phase 3: Algorithm development**

* CRO and QIEA development for the problem to fulfill the first objective.

**Phase 4: A new hyper-heuristic based on CRO and QIEA.**

* **Contribution**: A new hyper-heuristic algorithm based on CRO and QIEA development for the problem to fulfill the second objective.
* **Deliverable:** To develop a new hyper-heuristic algorithm to solve some limitation encountered in previous metaheuristic algorithm.

**Phase 6: report writing**

* Documentation of result analyses
* Writing for thesis and research papers

**Phase 5: Comparative study**

* Result from experiment in Phase 3 and 4 is compared.
* Statistical analysis, fitness convergence.
* Result validation with BLAST.

**Figure 3.1** Research framework

1. Dataset for Experiment

In this research, fragment instances are collected as dataset for the problem. The dataset is a benchmark and standard dataset that was used by several other research to test on gene fragment assembly algorithms. The dataset was retrieved from GenFrag, a program that receive input as nucleotide sequence and generates a set of fragments with specified criteria ([Engle and Burks, 1993](#_ENREF_12); [Engle and Burks, 1994](#_ENREF_13)). The output may differ as the set of fragments is generated randomly. The dataset can be retrieved from http://chac.sis.uia.mx/fragbench/genfrag.php. Table 3.1 shows the details of instances or the sets of fragments used in this research. This set of data used Smith-Waterman alignment for sequence alignment analysis ([Mallén-Fullerton *et al.*, 2013](#_ENREF_29)). The parameter settings are 1 for a match, -3 for a mismatch, and -2 for a gap.

The datasets were generated from several sequences and can be obtained from the National Center for Biotechnology Information (NCBI) website. The sequences were divided by using GenFrag tools ([Engle and Burks, 1994](#_ENREF_13)). GenFrag accept sequences as input and generate overlapping gene fragments. Originally, for gene fragment assembly, a set of 16 instances used for the research. However, due to computational problem, only 7 instances are selected for this research.

The first sequence generated is a cluster of fironectin type III repeats found in the human major histocompatibility complex class III region (ID: X60189) ([Matsumoto *et al.*, 1992](#_ENREF_30)). The length of the sequence is 3835 base pairs. From this sequence, 7 instances were generated, which is A, B, C, and D. Table 3.1 shows the detail of the instances. Instance A have a coverage of 4, and have 39 gene fragments. The mean length is 395. Instance B have a coverage of 5, and have 48 gene fragments. The mean length is 286. Instance C have a coverage of 6, and have 66 gene fragments. The mean length is 343. Instance D have a coverage of 7, and have 68 gene fragments. Its mean length is 387.

The second sequence generated is a human apolipoprotein B gene (ID: M1542). The sequence have a length of 10089. From this sequence, 3 instances were generated. Table 3.1 shows the details of the instances. Instance E, F, and G is derived from this sequence. Instance E have a coverage of 5, and have 127 gene fragments. It have a mean length of 398. Instance F have a coverage of 6, and have 177 gene fragments. It have a mean length of 350. Instance G have a coverage of 7, and have 177 gene fragments, while having the mean length of 383.

**Table 3.1** Details of datasets.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Label | Instances | Coverage | Mean fragment length | Number of fragments | Original sequence length(in bps) |
| A | X60189\_4 | 4 | 395 | 39 | 3835 |
| B | X60189\_5 | 5 | 286 | 48 |
| C | X60189\_6 | 6 | 343 | 66 |
| D | X60189\_7 | 7 | 387 | 68 |
| E | M15421\_5 | 5 | 398 | 127 | 10089 |
| F | M15421\_6 | 6 | 350 | 177 |
| G | M15421\_7 | 7 | 383 | 177 |

1. Problem Formulation

Gene fragment assembly problem is included as Non-deterministic polynomial (NP)-problem ([Pevzner, 2000](#_ENREF_37)). A permutation representation with number encoding is applied. Each fragment in instance is assigned with integer number. These integer numbers is permutated to represents the arrangement of fragments. To maintain a legal solution, several condition must be satisfied. First, all fragments must be represented in the ordering. Second, fragments are not allowed to appear more than once. The solution is evaluated with the following fitness function

 (3.1)

where variable *f* is the fragments, variable *w* is the overlap score between *f*[*i*] and *f*[*i*+1], variable *n* is the number of fragments. Fitness function *F* sums the overlap score in a given solution. The objective is to maximize the scoring value. The best solution will have the highest sum of overlap score (maximum fitness score), and the order of the solution has a strong overlap between adjacent fragments.

1. Performance Measurement

The performance of algorithms is evaluated by using statistical analysis and fitness convergence, and validation with BLAST alignment.

1. Statistical Analysis

After 10 runs of algorithm for each dataset, results are collected to perform evaluation and compared with each algorithm based on its best score, sample mean, and standard deviation. Sample mean is the sum of observed values in a data and divided by the number of observations (Equation (3.2)). In this experiment, the observed values are the result collected from 10 runs of algorithms on one dataset.

 (3.2)

Equation (3.2) shows the calculation of sample mean, where *x* is the observed values, and *n* is the sample size. Standard deviation is used to measure the variability of a set of data.

 (3.3)

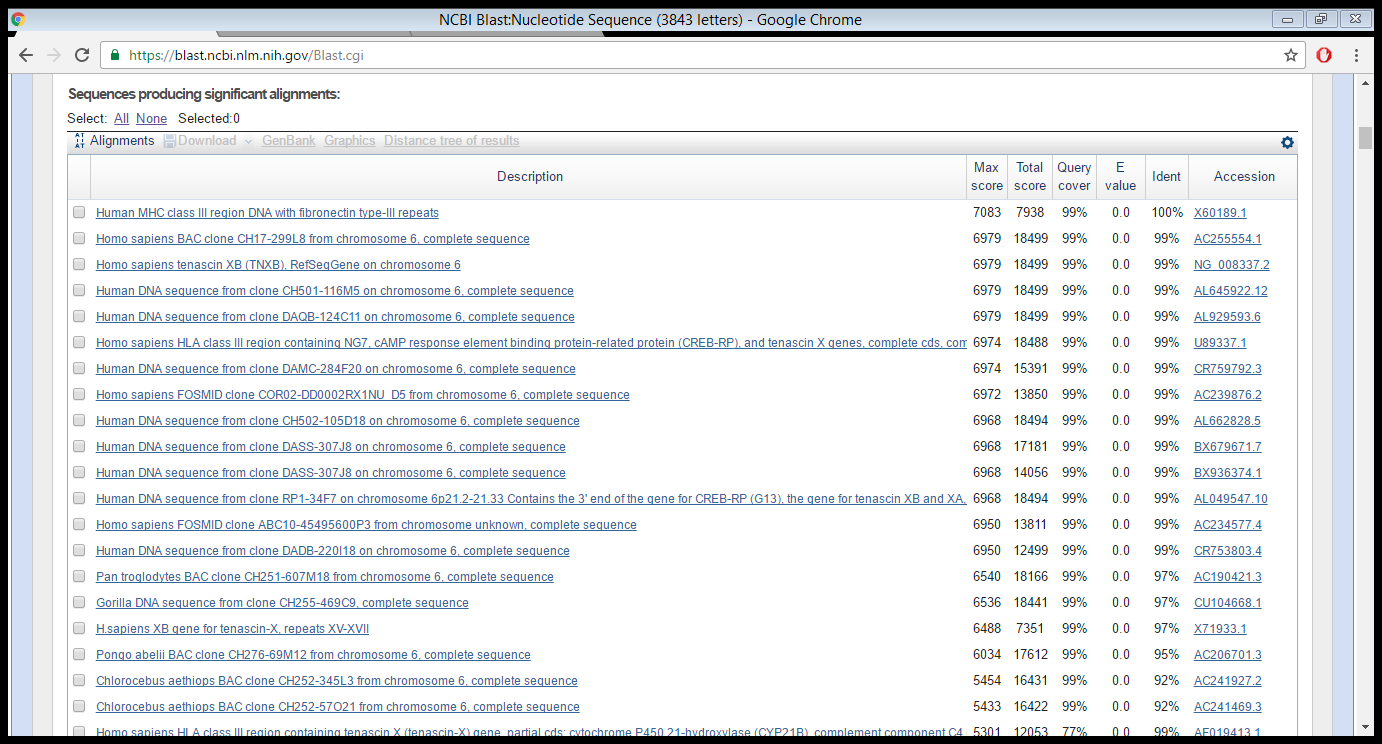
1. Fitness Convergence

The performance of each algorithm is analyzed by using the fitness convergence. In this research, the best-so-far convergence is applied. The best-so-far solution is asked whether it converge or not, as ([Gutjahr, 2010](#_ENREF_15)).

1. Validation with BLAST Alignment

Blast is a Basic Local Alignment Search Tool, a tool to find biological sequence information by comparing the query sequence with database of sequences. BLAST use heuristic method to find the similar sequences. The output included is the description, maximum score, total score, query cover, E value, ident, and accession number. Figure 3.2 shows the example of of BLAST output. The description is the matched database sequence. Maximum score is the highest alignment score from the database sequence. Total score is the total alignment scores for all alignment segments. The query coverage is the percentage of query covered by alignment to the database sequence. The E value is expect value that describe the number of hits one can expect to see by chance when searching a database of a particular size. The lower the E-value, the more significant the match is. Max ident is the max identity of all query-subject alignments.

The result, or the best fragment order in CRO, QIEA, and QCRO is aligned to the database in NCBI using BLAST tool. The similarity of the result to the sequences in database is collected and compared.



Max score

Total score

Query cover

E value

Ident

Description

**Figure 3.2** Example of BLAST output.

1. Summary

A benchmark dataset on gene fragment assembly problem is used to test the performance of proposed algorithm. The highest fitness value of a solution marked as the best solution of an instance. The result from each algorithm collected and compared by using statistical analysis, fitness convergence, and validation by using BLAST.

**CHAPTER 4**

DEVELOPMENT OF CRO AND QIEA FOR GENE FRAGMENT ASSEMBLY

1. Overview

In this research, CRO and QIEA were applied to solve gene fragment assembly. For starter, these algorithm were analyzed to understand its feature. Then, the algorithms is developed using programming language Python and the experiment was conducted by using a 12GB RAM computer with Mac OS X operating system.

1. CRO for Gene Fragment Assembly

In gene fragment assembly problem, the solutions consists of random arrangement of fragments in a dataset. In the initialization stage, the molecular structure in each molecule is initiated as a random permutation of number of fragments. The fitness of the structure is evaluated as *PEm*. In the iteration loop, the molecules are manipulated by the chemical reactions. The chemical reactions are represented by several operators suggested ([Lam and Li, 2010a](#_ENREF_23), [2012](#_ENREF_25)). Gene fragment assembly problem is a permutation problem.

Hence, an appropriate operator need to be assigned to each elementary reaction. Decomposition adopts circular shift operator where the items is solution is shifted to the left or right. This operator is assign to mimic the decomposition reaction where the same molecule have changed it state. Two exchange neighborhood structure operator used for on-wall ineffective collision and inter-molecular ineffective collision. It is a neighborhood search operator for permutation problem ([Lam and Li, 2010a](#_ENREF_23), [2012](#_ENREF_25)). Synthesis adopts edge recombination crossover operator because it is suitable with permutation problem ([Shaj *et al.*, 2016](#_ENREF_40)). At the end of every iteration, elitism is applied to produce the best next generation of molecule for the next iteration.

Figure 4.1 shows the pseudocode of CRO used to solve gene fragment assembly. The input value is the list of gene fragments. Variable *f* is a gene fragment, while variable *k* is the total number of fragments. The output of this experiment is the best solution of ordered fragments. The output is determined by the maximum overlap score. The best solution is represented as *Fmax*. The parameter involved is population size, iteration, kinetic energy loss rate, initial kinetic energy, alpha, beta, and molecule collision rate, and the value of the parameters is set as 1000, 1000, 0.2, 500, 10, 0.2, respectively. The value of kinetic energy loss rate, alpha, beta, and molecule collision rate are set as suggested by ([Lam and Li, 2012](#_ENREF_25)). Variable *F* is a set of permutated gene fragment. Variable *Molecules* contain a set of *F*. Variable *n* is the population size.

In the while loop, the process begin. The value of variable *b* is selected at random between 0 to 1. If the value of *b* is more than molecule collision rate, the process will enter uni-molecular collision. Variable *m* indicate a molecule among the set of *Molecules.* Molecule *m* is selected at random. If the difference of number of hit and minimum hit of the molecule is more than *alpha*, the process will enter decomposition. Else, the process will enter on-wall collision. In decomposition, the molecule will decompose into 2 molecules. In on-wall ineffective collision, the molecule hit the wall of the container and the kinetic energy slightly change. If the value of *b* is less than molecule collision rate, the process will enter inter-molecular collision. Two molecules will be selected at random. If the kinetic energy of the first and second molecules is less than alpha, both of the molecule will enter synthesis. Else, both will enter intermolecular collision. In synthesis, the molecule will merge. In intermolecular collision, the molecules will collide with each other. Once the while loop is done, the best molecule is selected based on the value of fitness score. The best molecule is the one with the higher maximum fitness score.

|  |  |
| --- | --- |
| 1 | Input: Gene fragments=[*f­1, f2, …,fk*], for *k* = number of fragments |
| 2 | Output: Maximum overlap score of the best solution: *Fmax* |
| 3 | Set parameters |
| 4 | *Molecules=*[*F1,F2,…,Fn*], where *n*=*Popsize*, *F*=permutation set of gene fragments |
| 5 | While *i*<=*Iteration*: |
| 6 |  |
| 7 | if *b*> *MoleColl*: |
| 8 |  |
| 9 | if *m.numHit-m.minHit*>*alpha*: |
| 10 | Decomposition(*m*) |
| 11 | else: |
| 12 | On-wall(*m*) |
| 13 | else: |
| 14 | *m1Molecules* |
| 15 | *m2Molecules* |
| 16 | if (*m1.KE*<*beta*) and (*m2.KE*<*beta*): |
| 17 | Synthesis(*m1, m2*) |
| 18 | else: |
| 19 | IntermolecularCollision(*m1,m2*) |
| 20 | *i*=*i*+1 |
| 21 | Best solution=Maximum(*Molecules*) |

**Figure 4.1** Pseudocode of CRO used to solve gene fragment assembly. The parameter involved is population size, iteration, kinetic energy loss rate, initial kinetic energy, alpha, beta, and molecule collision rate, and the value of the parameters is set as 1000, 1000, 0.2, 500, 10, 0.2, respectively.

1. QIEA for Gene Fragment Assembly

In the beginning of the algorithm, parameters such as population size, iteration, rotational angle, and migration conditions are assigned. A set of Q-bit solution is initialized according to equation (4.1). Since gene fragment assembly problem is a permutation problem, Q-bit contain

 (4.1)

where *f* is the *k*th fragments, where *k* is the total number of fragments in a dataset, is the total number of population. In initialization step, andis initialized with complex number. Next,is observed to form , whereis the solution in the form of integer number set where.

Since *P(t)* derived from a list of binary string in *Q(t)*, the solution in may have repeated fragments and the integer value is greater than the total of fragments.

A repair algorithm is employed. Each repeated fragments is replaced with missing integer value in the solution in ascending order. Figure 4.2 shows the pseudocode of the repair algorithm. Variable *S* indicates a set of fragments. Variable *x* indicates a fragment in *S*. since the gene fragment is represented by an integer as unique ID, it is easier to know if the fragment is repeated or missing. Variable *f* indicates the number of fragments. If the value of *x* is more than *f*, assign value *x* at random between 0 to *f.* If *x* is not the first occurrence and is repeated, enter while loop. While *i* is less than *f*, *x* equal to *i*.

The solution is evaluated. Next,is taken as the best set of solutions,. The best solution amongis stored as . Next, the iterations begin. Q-bits setis observed to form. Then,is evaluated. is updated by using rotational Q-gate. The magnitude ofis 0.15. The best solutions inandis stored in. The best solution amongis stored as . Next, if a migration condition satisfied, solutions inmigrates among each other for local migration, or solutionmigrates toglobally.

|  |  |
| --- | --- |
| 1 | For *x* in *S*: |
| 2 | If *x* >= *f*: |
| 3 | Assign |
| 4 | For *x* in *S*: |
| 5 | If *x* is not the first occurrence && repeated: |
| 6 | while *i*<*f*: |
| 7 | *x*=*i* |

**Figure 4.2** Pseudocode of repair algorithm.

1. Experimental Result

Firstly, the experimental result of gene fragment assembly problem with CRO is presented. The first experiment was carried out by controlling and manipulating the variable involved. The variable involved is number of solution or population size, and number of iteration. For each value of controlled variable number of solution, the value of number of iteration is manipulated as 100, 500, and 1000. The maximum fitness score of each experiment for each instance is collected and compared. As observed, the score for instance A reach the maximum fitness score at the number of solution 1000 and number of iteration 1000. For the other instances, as the number of solution and number of iteration increases, the best score increases. The result of this experiment is shown in Table 4.1. Four of the instances which is instance A, B, D, and F, obtain maximum fitness score at the highest value of number of solution and the highest value of number of iteration while the other three instances, C, E, and G obtain the maximum fitness score at number of solution of 500.

Second, the same experiment as above carried out by using QIEA. Table 4.2 shows the results obtained in the experiment with QIEA. All the maximum score is found when the number of solution is the highest value.

Comparing the performance of CRO and QIEA in term of fitness score of all instances involved, CRO produces highest maximum fitness score compared to QIEA. QIEA produces maximum fitness score of all instances that is lower than CRO. Even though with the number of solution is at maximum, the algorithm cannot obtained the results as maximum as in CRO. QIEA obtained lower maximum fitness score compared to CRO.

Since the majority of the best fitness score generated is when population size is 1000 for both algorithms, this values of population size is taken for the next experiment. The value of iteration is 1000. Both of the algorithms was run for 10 times in order to get the best, worst, mean, and standard variation of each instance.

Table 4.3 shows the results obtained for CRO after 10 times run. According to [Luque and Alba (2005)](#_ENREF_27), the optimal value for instance A is 11478. In this experiment, the maximum fitness score of instance A reach the optimal value. Table 4.4 shows the result obtained for QIEA after run for 10 times. From the experiment above, the best result out of the 10 runs from CRO and QIEA is collected to perform convergence analysis. The convergence analysis of CRO and QIEA is presented in Chapter 5 to compare the analysis with the result from QCRO.

**Table 4.1** Experimental result with CRO. The experiment is tested on various controlled and manipulated variable. The bolded value is the maximum fitness score for each instance.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. of solution | Iteration | A | B | C | D | E | F | G |
| 100 | 100 | 10538 | 13050 | 15159 | 16461 | 23904 | 23851 | 27090 |
| 500 | 10712 | 13346 | 16708 | 19980 | 31788 | 33992 | 40243 |
| 1000 | 10653 | 13140 | 16429 | 20134 | 29316 | 35890 | 41908 |
| 500 | 100 | 11248 | 14051 | 17436 | 20323 | 29815 | 30543 | 34728 |
| 500 | 11420 | 14060 | 17884 | 20797 | 34553 | 39381 | 45013 |
| 1000 | 11277 | 14056 | **18327** | 20951 | **37150** | 40915 | **49988** |
| 1000 | 100 | 11464 | 13933 | 17042 | 20765 | 30868 | 32638 | 38376 |
| 500 | 11334 | 14039 | 18133 | **21235** | 35297 | 38619 | 44706 |
| 1000 | **11478** | **14093** | 18142 | 21190 | 36943 | **42140** | 45534 |

**Table 4.2** Experminetal result with QIEA. The experiment is tested on various controlled and manipulated variable. The bolded value is the maximum fitness score for each instance.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. of solution | Iteration | A | B | C | D | E | F | G |
| 100 | 100 | 3876 | 5490 | 5012 | 8001 | 5678 | 7701 | 7009 |
| 500 | 5337 | 5765 | 5419 | 7659 | 5669 | 7201 | 7562 |
| 1000 | 4677 | 5890 | 5210 | 7443 | 5901 | 7439 | 7665 |
| 500 | 100 | 5324 | 4665 | 6431 | 7322 | 6712 | 6509 | 7659 |
| 500 | 4388 | 4977 | 6709 | 7604 | 7044 | 7326 | 8776 |
| 1000 | 4369 | 4761 | 6422 | 8091 | 7534 | 7413 | 8416 |
| 1000 | 100 | 5217 | 5712 | **7001** | **8742** | 7095 | 7500 | 9031 |
| 500 | 5467 | 5931 | 6550 | 8335 | **7811** | 7601 | 9341 |
| 1000 | **5900** | **6021** | 6889 | 8411 | 7012 | **8013** | **9543** |

**Table 4.3**  CRO analysis. The CRO algorithm was run for 10 times. Variable *b* is the best fitness score, *w* is the worst fitness score, *m* is the mean of the maximum fitness scores, and *s* is the standard deviation of fitness scores. Parameters: iteration=1000, population size=1000.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instances | *b* | *w* | *m* | *s* |
| A | 11478 | 11233 | 11375 | 70.21 |
| B | 14137 | 14048 | 14076 | 31.608 |
| C | 18310 | 18012 | 18189 | 98.3 |
| D | 21239 | 20730 | 21070 | 149.16 |
| E | 36975 | 36281 | 36545 | 223.51 |
| F | 44369 | 37558 | 39911 | 2426.3 |
| G | 48777 | 43618 | 46098 | 1967.2 |

**Table 4.4** QIEA analysis. The QIEA algorithm was run for 10 times. Variable *b* is the best fitness score, *w* is the worst fitness score, *m* is the mean of the maximum fitness scores, and *s* is the standard deviation of fitness scores. Parameters: iteration=1000, population size=1000.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instances | *b* | *w* | *m* | *s* |
| A | 5964 | 5177 | 5413 | 217.93 |
| B | 6871 | 6418 | 6594 | 143.3 |
| C | 7429 | 6653 | 7052 | 241.27 |
| D | 8907 | 8207 | 8499 | 201.77 |
| E | 8297 | 7361 | 7688 | 246.22 |
| F | 8319 | 7786 | 7938.1 | 196.62 |
| G | 9931 | 8984 | 9390.5 | 309.51 |

1. Summary

Algorithm CRO and QIEA are designed accordingly to gene fragment assembly problem. The performance of each algorithm is evaluated. The result for QIEA algorithm shows that the algorithm is not suitable to be used to solve gene fragment assembly problem. In the next chapter, a hyper-heuristic algorithm based on CRO and QIEA algorithms is developed to solve the gene fragment assembly problem.

**CHAPTER 5**

DEVELOPMENT OF HYPER-HEURISTIC

1. Overview

Metaheuristic provide general framework to solve specific problem domain. Metaheuristic developments are too problem-specific and required intensive knowledge. In addition, no free lunch theorem showed that all heuristic and metaheuristic had the same average performance ([Wolpert and Macready, 1997](#_ENREF_48)). In this research, a hyper-heuristic algorithm is proposed. Hyper-heuristic have several advantages over metaheuristic. Hyper-heuristic development is easier, as it describe as “knowledge poor” heuristic. Heuristics or metaheuristics are used to choose the most suitable low level heuristic from a set of low level heuristics during the search ([Burke *et al.*, 2003](#_ENREF_6); [Burke *et al.*, 2013](#_ENREF_7); [Ozcan *et al.*, 2008](#_ENREF_35)). Unlike metaheuristic, hyper-heuristic does not required fine tuning of parameter.

Hyper-heuristic process is divided into two phases: heuristic selection and move acceptance. There are seven heuristic selection strategies that has been tested by ([Burke *et al.*, 2013](#_ENREF_7)). For instances, simple random, random descent, random permutation, random permutation descent, choice function, tabu search, and greedy ([Burke *et al.*, 2003](#_ENREF_6); [Burke *et al.*, 2013](#_ENREF_7); [Ozcan *et al.*, 2008](#_ENREF_35)). There are five move acceptance strategies that has been tested by ([Burke *et al.*, 2013](#_ENREF_7)). For instances, all moves, only improving, improving and equal, exponential Monte Carlo with counter, and great deluge ([Burke *et al.*, 2003](#_ENREF_6); [Burke *et al.*, 2013](#_ENREF_7); [Ozcan *et al.*, 2008](#_ENREF_35)).

Several hyper-heuristic frameworks are identified in ([Ozcan *et al.*, 2008](#_ENREF_35)) and mentioned in Figure 2.7 in chapter 2. In this research, hyper-heuristic framework FA in Figure 2.7 is selected and modified to fit with metaheuristic algorithms QEIA and CRO criteria.

Accept/Reject solution

Select/Apply heuristic

4 heuristics selection according to CRO algorithm.

Select/reject solution with QIEA

**Figure 5.1** Hyper-heuristic framework modified from FA in Figure 2.7 to fit with algorithm QIEA and CRO criteria

Figure 5.1 shows the modified framework A of hyper-heuristic. In framework A, the heuristic is selected first before applying it to the solution. After heuristic is decided and applied to the solution, the second method will accept or reject the new solution.

However, in Figure 5.1, the first step is an action of method to accept or reject the solution. Afterward, a heuristic is selected and the accepted solution is processed by the heuristic. Further explanation is given in Section 5.2.

1. Metaheuristic-based Hyper-heuristic Algorithm

In this research, metaheuristic algorithms QIEA and CRO are used to produce the hyper-heuristic algorithm. In Figure 5.2, the framework of QCRO is divided into 2 main area. Area A indicates the move acceptance region. In this region, solutions are update, repair as in QIEA process. Afterward, the solution undergo the move acceptance process. The elitism function accept or reject solutions to go through the next process. For move acceptance phase, only improving strategic is used. Hence, only improving solutions is selected to proceed to the next iterations. By implementing the QIEA algorithm, the individual is represented as Q-bit (equation 4.1).

Since *P(t)* derived from a list of binary string in *Q(t)*, the solution may have repeated fragments and the integer value is greater than the total of fragments. A repair algorithm as in Figure 4.2 is used. Each repeated fragments is replaced with missing integer value in the solution in ascending order. The solution is evaluated. Next,is taken as the best set of solutions,.

The best solution amongis stored as . The beginning of the algorithm is the same as QIEA in Chapter 4. For the iteration cycle, Q-bits setis observed to form. Then,is evaluated. is updated by using rotational Q-gate. The magnitude ofis 0.15. The best solutions inandis stored in. The best solution amongis stored as . The solution undergoes elitism process. In elitism process, the best few solutions are selected to perform the search to find the optimal solution.

Heuristic selection phase shown in Area B in Figure 5.2. Simple random strategic is selected. All four reactions in CRO algorithm are involved. The low level heuristic is selected randomly. The low level heuristics are the chemical reactions in algorithm CRO, which is represented by the same operators as in Chapter 4. The best solution is collected. If the stopping criteria satisfied, the algorithm terminates and the global maximum is obtained.

Update

Update

Update

Repair

Repair

Repair

E=Elitism(X, B)

beta<moleColl

*r*[0,1]<0.5

*r*[0,1]<0.5

On-wall collision

Decomposition

Intermolecular ineffective collisions

Synthesis

Check for new maximum point

Stopping criteria satisfied?

Obtain global maximum

Q(t)

P(t)

Yes

Yes

Yes

No

No

No

**Area A**

**Area B**

**Figure 5.2** Framework of hyper-heuristic QCRO

1. Convergence Analysis of CRO, QIEA, and QCRO

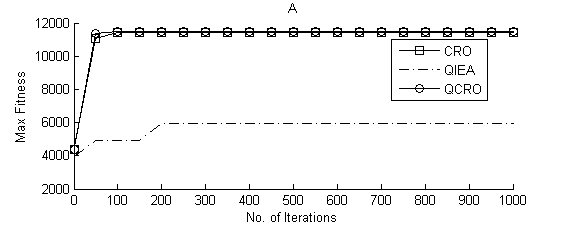
In Chapter 4, an experiment is conducted for both algorithm CRO and QIEA for 10 times to obtain the statistical analysis. From the experiment, the best result is collected for convergence analysis in this section.

In this experiment, several parameter values that are relatable to each algorithm such as iteration, population size, and are set identical. The number of iteration, population size, andis 1000, 1000, and 0.15, respectively. The experiment for each instance is run for 10 times, for algorithm CRO, QIEA, hyper-heuristic QCRO. For each algorithm, the experiment that produce the best score value out of 10 runs is taken to be compared with other algorithm. This comparison is taken to produce a convergence analysis graph to observe the trend of maximum fitness value over number of iterations.

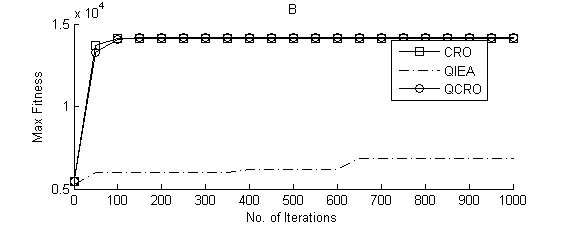
Statistical analysis is taken based on the experiment. For each instance, the best score, worst score, of a solution is taken. The mean, median, and standard variation for 10 runs is taken. These result from algorithm CRO, QIEA, and hyper-heuristic QCRO is compared with each other.

Figure 5.3 shows the convergence analysis graph for instance A. In this figure, CRO and hyper-heuristic QCRO produce far better solutions compared to QIEA from the beginning to end of iteration. For algorithm CRO and hyper-heuristic QCRO, these two algorithms compete each other to produce the best solution. In the end of iteration 1000, algorithm CRO and hyper-heuristic QCRO produce best solution for instance A with maximum fitness score value of 11478. Algorithm QIEA experiences premature convergence and its best score value, 5964, is lower than half of the maximum fitness score for instance A.

Figure 5.4 shows the convergence analysis graph for instance B. In this figure, algorithm CRO and hyper-heuristic QCRO produce solutions with maximum fitness score value better than algorithm QIEA in less than 100 iterations. The maximum fitness score for all algorithms can be view in Table 5.1. Interestingly, at the end of 1000 iterations, hyper-heuristic QCRO produce better solution with maximum score of 14192, higher than algorithm CRO and QIEA. Algorithm CRO produce solution with maximum score of 14137, slightly lower than solution score produced by QCRO. This shows that algorithm CRO and QCRO is competitive. Algorithm QIEA experiences premature convergence and produce solution with score value of 6871.

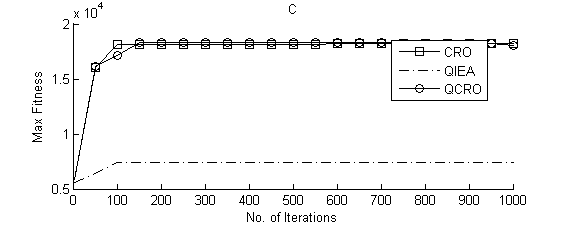


**Figure 5.3** Convergence analysis graph for instance A



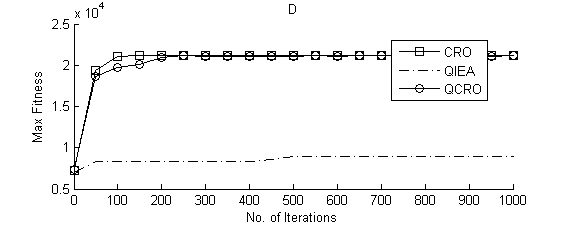
**Figure 5.4** Convergence analysis graph for instance B

Figure 5.5 shows the convergence analysis graph for instance C. Algorithm CRO and hyper-heuristic QCRO produce better solution compared to algorithm QIEA from the beginning to the end of iteration. At the 100th iteration, algorithm CRO produces slightly better solution compared to hyper-heuristic QCRO. Afterward, algorithm CRO and QCRO produces solutions with slightly close score to each other. In the end of iteration, hyper-heuristic QCRO produces solution which is slightly better than one produced by CRO based on their score. Solution produced hyper-heuristic QCRO have maximum score value of 18342 while algorithm CRO produce solution with maximum score value of 18310.



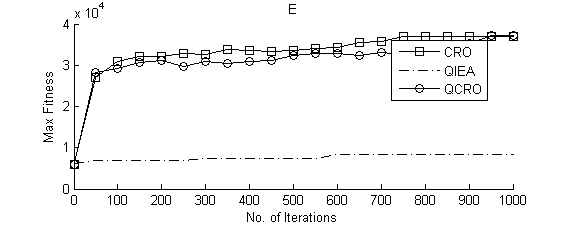
**Figure 5.5** Convergence analysis graph for instance C

Figure 5.6 shows the convergence analysis graph on instance D. from the beginning of the iteration, algorithm CRO and hyper-heuristic QCRO produce solutions far better than algorithm QIEA. Before the iteration 200, hyper-heuristic QCRO produces slightly worst solutions compared to algorithm CRO because of the QIEA characteristic in hyper-heuristic QCRO. As mentioned by ([Feng *et al.*, 2006](#_ENREF_14)) for a permutation problem with a big search space, QIEA shows a worse performance. After the iteration 200, the solutions from both algorithm CRO and hyper-heuristic QCRO have slightly the same maximum score. In the end of iteration, hyper-heuristic QCRO have the higher maximum score compared to algorithm CRO and QIEA. The maximum score taken from hyper-heuristic QCRO is 21262. The maximum score taken from algorithm CRO is 21239. Meanwhile, QIEA experienced premature convergence and have the maximum solution with a maximum score of 8907.



**Figure 5.6** Convergence analysis graph for instance D

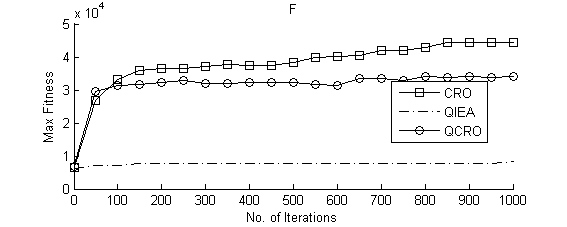
Figure 5.7 shows the convergence analysis graph for instance E. algorithm CRO and hyper-heuristic QCRO search for the maximum solution quickly compared to algorithm QIEA in less than 100 iteration with a huge differences. As the previous instances, algorithm QIEA shows the worst performance compared to the other algorithms. Algorithm CRO and hyper-heuristic QCRO compete with each other to produce solution with maximum score throughout the iteration. In the end of the iteration, hyper-heuristic QCRO produce better solution with a maximum score of 37302. Algorithm CRO follows up with a solution that has maximum score of 36975. Algorithm QIEA produced a worse solution with a maximum score of 8297.



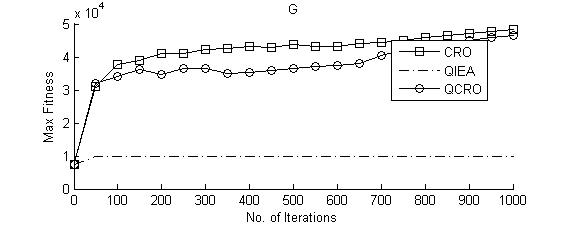
**Figure 5.7** Convergence analysis graph for instance E

Figure 5.8 shows the convergence analysis graph for instance F. Algorithm CRO and hyper-heuristic QCRO search for the maximum solution faster than algorithm QIEA before the 100th iteration. Meanwhile, the performance of algorithm QIEA is worse. Along the iteration, algorithm CRO and hyper-heuristic QCRO compete with each other to find the solution with the maximum score. In the end of the iteration, algorithm CRO produces better solution with a maximum score of 44369. Hyper-heuristic QCRO follows up with a solution that has a maximum score of 35292. As mentioned by ([Feng *et al.*, 2006](#_ENREF_14)), as the search space expand, QIEA started to shows performance badly. The main feature in QIEA, where it cannot handle a big search space started to show in QCRO. Meanwhile, algorithm QIEA continue to produce solutions worse than CRO and QCRO.

Figure 5.9 shows the convergence analysis for instance G. Algorithm CRO and hyper-heuristic QCRO found better solution compared to algorithm QIEA in less than 100 iterations. Throughout the iterations, algorithm CRO produces better solution compared to hyper-heuristic QCRO. QIEA characteristic in hyper-heuristic QCRO leads the algorithm to produces bad solution. However, as the iteration continued, hyper-heuristic QCRO attempt to produces better solution than the previous one. In the end of the iteration, algorithm CRO overpasses hyper-heuristic QCRO with better solution, which has maximum score of 48777. Hyper-heuristic QCRO produces solution with maximum score of 46417 as it cannot no longer handle a problem with big search space. Algorithm QIEA produces solution with the worst maximum score of 9931 for instance G.



**Figure 5.8** Convergence analysis graph for instance F



**Figure 5.9** Convergence analysis graph for instance G

1. Statistical Analysis

The statistical analysis is performed on all instances and shown in Table 5.1. according to ([Luque and Alba, 2005](#_ENREF_27)), the optimal value for instance A is 11478. For instance A, QCRO and CRO get the optimal value of 11478.

For instance B, C, D, and E, QCRO performs better compared to CRO with best fitness score of 14192, 18342, 21262, 37302, against 14137, 18310, 21239, 36975, respectively.

For instance F and G, QCRO performs slightly lower than CRO as it started to inherited QIEA feature where it cannot comprehend with a problem with big search space. QIEA feature in QCRO preventing the algorithm from performing well in instance F and G.

Meanwhile, QIEA experienced premature convergence for all instances since QIEA cannot comprehend with huge search space ([Feng *et al.*, 2006](#_ENREF_14)).

**Table 5.1** Statistical comparison of CRO, QIEA, and QCRO. The best maximum fitness score of each algorithm is labelled with \*.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | CRO | | | | | QIEA | | | | | QCRO | | | | | |
| Best | Worst | Mean | Median | s | Best | Worst | Mean | Median | s | Best | Worst | Mean | Median | s |
| A | 11478\* | 11233 | 11375.7 | 11370 | 70.205 | 5964 | 5177 | 5413.3 | 5354.5 | 217.93 | 11478\* | 11461 | 11474.7 | 11478 | 5.568 |
| B | 14137 | 14048 | 14076 | 14062.5 | 31.61 | 6871 | 6418 | 6594.6 | 6553 | 143.3 | 14192\* | 13974 | 14079.4 | 14059 | 79.475 |
| C | 18310 | 18012 | 18189.2 | 18192.5 | 98.303 | 7429 | 6653 | 7052 | 7082 | 241.267 | 18342\* | 17348 | 18130.4 | 18210 | 275.447 |
| D | 21239 | 20730 | 21070.2 | 21099 | 149.156 | 8907 | 8207 | 8509.4 | 8470.5 | 199.81 | 21262\* | 20955 | 21088.4 | 21085 | 85.121 |
| E | 36975 | 36281 | 36545 | 36520 | 223.51 | 8297 | 7361 | 7688.9 | 7648 | 246.22 | 37302\* | 31952 | 34549.9 | 35336 | 1872.41 |
| F | 44369\* | 37558 | 39911.5 | 38617 | 2426.3 | 8319 | 7670 | 7938.1 | 7849.5 | 196.62 | 35292 | 33029 | 33940.8 | 33639 | 745.32 |
| G | 48777\* | 43618 | 46098.3 | 45010.5 | 1947.19 | 9931 | 9390.5 | 9390.5 | 9312.5 | 309.513 | 46417 | 37873 | 40579 | 40126 | 2341.16 |

1. Validation with BLAST Analysis

The best result from CRO, QIEA and QCRO is collected and undergo multiple sequence alignment to get consensus of the ordered gene fragments. The draft gene sequences are aligned with its original gene sequences, respectively, to recognize how the similarity and how good the draft gene sequences are. Table 5.2 shows the result collected after the alignment with non-redundant BLAST. Query coverage means how many percentage the query sequence match with the original sequence. The E value is a parameter that describes the number of hits one can “expect” to see by chance when searching a database of a particular size. To lower the E value, the more “significant” the match is. Max ident is the max identity of all query-subject alignments.

In table 5.2, all results from all three algorithm shows that the assembly is good as the E value is low. Although QIEA shows bad performance earlier in ordering the gene fragments, the assembly shows that the result from QIEA is as good as the other two algorithms.

**Table 5.2** Best result alignment from CRO, QIEA, and QCRO compared for the similarity with the original sequence.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instances | CRO | | | QIEA | | | QCRO | | |
| Query coverage (%) | E value | Ident (%) | Query coverage (%) | E value | Ident (%) | Query coverage (%) | E value | Ident (%) |
| A | 99 | 0 | 100 | 99 | 0 | 96 | 91 | 0 | 99 |
| B | 100 | 0 | 100 | 97 | 0 | 99 | 97 | 0 | 97 |
| C | 100 | 0 | 99 | 98 | 0 | 94 | 96 | 0 | 99 |
| D | 96 | 0 | 98 | 97 | 0 | 97 | 96 | 0 | 99 |
| E | 98 | 0 | 94 | 96 | 0 | 96 | 98 | 0 | 100 |
| F | 99 | 0 | 97 | 95 | 0 | 97 | 98 | 0 | 97 |
| G | 98 | 0 | 98 | 96 | 0 | 100 | 99 | 0 | 98 |

1. Summary

The experimental result of each algorithm is collected and evaluated. Convergence analysis, statistical analysis are used to evaluate the performance of each algorithm. Validation by similarity of the draft sequence from the algorithms was carried out. The proposed algorithm, hyper-heuristic QCRO shows good performance and is comparable to CRO algorithm.

**CHAPTER 6**

CONCLUSION AND FUTURE WORKS

1. Conclusion

In this study, the focus is to use metaheuristic algorithm to solve bioinformatics problem, which is gene fragment assembly problem. Hence, an algorithm is proposed to solve gene fragment assembly problem. Three objectives were examined. The first objective is to analyze a metaheuristic algorithm to solve gene fragment assembly problem. Several metaheuristic algorithms were investigated for the proposed algorithms. Two metaheuristic algorithms, CRO and QIEA was selected for the purpose.

The second objective is to develop a hyper-heuristic algorithm based on the two metaheuristic algorithm selected in the previous objective. The hyper-heuristic QCRO was developed based on the feature of CRO and QIEA. QCRO was designed to have 4 elementary chemical reactions in CRO in the heuristic selection phase, while having QIEA ways of handling solutions in move acceptance phase, where it update and repair the solutions while maintaining a good balance of exploration and exploitation of solution.

The third objective is the validation of proposed algorithm based on accuracy and optimality of the assembled fragments. The validation includes statistical analysis, convergence analysis, and similarity analysis by using BLAST tool. In comparison, QCRO manage to perform better compared to CRO and QIEA for a small search space. However, QCRO imitated the QIEA feature where it is not able to get better fitness score due to the big search space. For the similarity validation through BLAST tool, the draft gene sequences derived from the gene fragments ordered by QCRO. The similarity validation shows that QCRO is able to solve the gene fragment assembly problem at consensus sequence level with low errors.

1. Future Works

For future work, several improvement is suggested to obtain solutions with better maximum fitness score on gene fragment assembly problem. First, for the proposed hyper-heuristic algorithm, add more variety on heuristic or metaheuristic algorithms. The number of heuristic algorithms a hyper-heuristic is not limited. User control should be introduced where user can choose which algorithms is available and included during the process of searching for the best solution. User can also choose which operator to represent the natural behavior of an algorithm. For example, in CRO, one reaction can be represented by multiple operators. Synthesis reaction in CRO can be represented by variety of crossover operators. However, in this research, synthesis only represented by edge recombination crossover operator, which is suitable for gene fragment assembly problem. For a different problem, user can choose which operator should be used to suit the problem.

In this research, a similarity analysis is conducted to find the similarity of draft gene sequence generated from the arranged fragment order with the gene from the existing database in BLAST. In the future, we can focus on generating better draft gene sequence from the ordered fragments generated by the algorithm. Several improvement can be made by taking example from existing gene fragment assembly software. For example, instead of using Smith-Waterman alignment to generate overlap score, use *k-*mers to get the overlapping bases from two gene fragments. In Celera assembler, if the overlap is less than 6%, that means the gene fragment is a repeat induced. A quality control on the gene fragments could be implemented as in PHRAP assembler.

REFERENCES

Abdullah, A., Deris, S., Mohamad, M. S., and Anwar, S. (2013). An Improved Swarm Optimization for Parameter Estimation and Biological Model Selection. *PLoS ONE*. 8 (4): e61258.

Andrianantoandro, E., Basu, S., Karig, D. K., and Weiss, R. (2006). Synthetic biology: new engineering rules for an emerging discipline. *Mol Syst Biol*. 2 2006.0028.

Bai, Q. (2010). Analysis of particle swarm optimization algorithm. *Computer and information science*. 3 (1): 180-184.

Bastide, M., and McCombie, W. R. (2007). *Assembling genomic DNA sequences with PHRAP*. *Current Protocols in Bioinformatics*. (11.14. 11-11.14. 15).

Burke, E., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., and Woodward, J. (2010). *A Classification of Hyper-heuristic Approaches*. M. Gendreau and J.-Y. Potvin. *Handbook of Metaheuristics*. (449-468). Springer US.

Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., and Schulenburg, S. (2003). *Hyper-heuristics: An emerging direction in modern search technology*. F. Glover and G. A. Kochenberger. *International series in operations research and management science*. (457-474). Springer US.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Oezcan, E.*, et al.* (2013). Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society*. 64 (12): 1695-1724.

Chandran, D., Bergmann, F., Sauro, H., and Densmore, D. (2011). *Computer-Aided Design for Synthetic Biology*. H. Koeppl, G. Setti, M. di Bernardo and D. Densmore. *Design and Analysis of Biomolecular Circuits*. (203-224). Springer New York.

Copeland, W. B., Bartley, B. A., Chandran, D., Galdzicki, M., Kim, K. H., Sleight, S. C.*, et al.* (2012). Computational tools for metabolic engineering. *Metabolic Engineering*. 14 (3): 270-280.

Dorigo, M., and Birattari, M. (2010). *Ant colony optimization*. *Encyclopedia of Machine Learning*. (36-39). Springer US.

Eberhart, R. C., and Shi, Y. (2001). Particle swarm optimization: developments, applications and resources. *Evolutionary Computation, 2001*. COEX, Seoul, Korea, 81-86.

Engle, M. L., and Burks, C. (1993). Artificially Generated Data Sets for Testing DNA Sequence Assembly Algorithms. *Genomics*. 16 (1): 286-288.

Engle, M. L., and Burks, C. (1994). GenFrag 2.1: new features for more robust fragment assembly benchmarks. *Comput Appl Biosci*. 10 (5): 567-568.

Feng, X. Y., Wang, Y., Ge, H. W., Zhou, C. G., and Liang, Y. C. (2006). *Quantum-inspired evolutionary algorithm for Travelling salesman problem*. G. R. Liu, V. B. C. Tan and X. Han. *Computational Methods*. (1363-1367). Dordrecht: Springer Netherlands.

Gutjahr, W. J. (2010). *Convergence Analysis of Metaheuristics*. V. Maniezzo, T. Stützle and S. Voß. *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*. (159-187). Boston, MA: Springer US.

Han, K.-H., and Kim, J.-H. (2002). Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *Evolutionary Computation, IEEE Transactions on*. 6 (6): 580-593.

Han, K. H. (2003). *Quantum-inspired evolutionary algorithm*. Korea Advanced Institute of Science and Technology, KAIST.

Huang, K.-W., Chen, J.-L., Yang, C.-S., and Tsai, C.-W. (2014). A memetic particle swarm optimization algorithm for solving the DNA fragment assembly problem. *Neural Computing and Applications*. 26 (3): 465-506.

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science and Engineering*. 9 (3): 90-95.

Keasling, J. D. (2010). Manufacturing molecules through metabolic engineering. *Science*. 330 (6009): 1355-1358.

Keasling, J. D. (2012). Synthetic biology and the development of tools for metabolic engineering. *Metabolic Engineering*. 14 (3): 189-195.

Kronberger, N. (2012). Synthetic biology: taking a look at a field in the making. *Public Understanding of Science*. 21 (2): 130-133.

Lam, A. Y., and Li, V. O. (2010a). Chemical-reaction-inspired metaheuristic for optimization. *Evolutionary Computation, IEEE Transactions on*. 14 (3): 381-399.

Lam, A. Y., and Li, V. O. (2010b). Chemical reaction optimization for cognitive radio spectrum allocation. *2010 IEEE Global Telecommunications*

*Conference GLOBECOM 2010* 6-10 December 2010. Miami, Florida, USA, 1-5.

Lam, A. Y., and Li, V. O. (2012). Chemical Reaction Optimization: a tutorial. *Memetic Computing*. 4 (1): 3-17.

Li, Z., Li, Z., Nguyen, T. T., and Chen, S. (2015). Orthogonal chemical reaction optimization algorithm for global numerical optimization problems. *Expert Systems with Applications*. 42 (6): 3242-3252.

Luque, G., and Alba, E. (2005). Metaheuristics for the DNA fragment assembly problem. *International Journal of Computational Intelligence Research*. 1 (1): 98-108.

Mallén-Fullerton, G. M., and Fernandez-Anaya, G. (2013). DNA fragment assembly using optimization. *2013 IEEE Congress on Evolutionary Computation*. Cancun, 1570-1577.

Mallén-Fullerton, G. M., Hughes, J. A., Houghten, S., and Fernández-Anaya, G. (2013). Benchmark datasets for the DNA fragment assembly problem. *International Journal of Bio-Inspired Computation*. 5 (6): 384-394.

Matsumoto, K.-i., Arai, M., Ishihara, N., Ando, A., Inoko, H., and Ikemura, T. (1992). Cluster of fibronectin type III repeats found in the human major histocompatibility complex class III region shows the highest homology with the repeats in an extracellular matrix protein, tenascin. *Genomics*. 12 (3): 485-491.

Mitchell, M. (1998). *An introduction to genetic algorithms*. Cambridge, MA, USA: MIT press.

Mount, D. W. (2001). *Bioinformatics: sequence and genome analysis*. Cold spring harbor laboratory press New York.

Mullikin, J. C., and Ning, Z. (2003). The Phusion Assembler. *Genome Research*. 13 (1): 81-90.

Myers, E. W., Sutton, G. G., Delcher, A. L., Dew, I. M., Fasulo, D. P., Flanigan, M. J.*, et al.* (2000). A whole-genome assembly of Drosophila. *Science*. 287 (5461): 2196-2204.

Ozcan, E., Bilgin, B., and Korkmaz, E. E. (2008). A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis*. 12 (1): 3.

Parsons, R. J., Forrest, S., and Burks, C. (1995). Genetic algorithms, operators, and DNA fragment assembly. *Machine Learning*. 21 (1-2): 11-33.

Pevzner, P. (2000). *Computational molecular biology: an algorithmic approach*. Cambridge, Massachusetts: MIT press.

Poli, R., and Graff, M. (2009). *There is a free lunch for hyper-heuristics, genetic programming and computer scientists*. L. Vanneschi, S. Gustafson, A. Moraglio, I. D. Falco and M. Ebner. *Genetic Programming*. (195-207). Berlin, Heidelberg: Springer.

Radenbaugh, A. J. (2008). *Applications of genetic algorithms in bioinformatics*. Master thesis. San Jose University.

Shaj, V., Akhil, P., and Asharaf, S. (2016). Edge-PSO: A recombination operator based PSO algorithm for solving TSP. *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. 21-24 September. Jaipur, India, 35-41.

Sommer, D., Delcher, A., Salzberg, S., and Pop, M. (2007). Minimus: a fast, lightweight genome assembler. *BMC Bioinformatics*. 8 (1): 64.

Sutton, G. G., White, O., Adams, M. D., and Kerlavage, A. R. (1995). TIGR Assembler: A new tool for assembling large shotgun sequencing projects. *Genome Science and Technology*. 1 (1): 9-19.

Van der Walt, S., Colbert, S. C., and Varoquaux, G. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science & Engineering*. 13 (2): 22-30.

Venter, G., and Sobieszczanski-Sobieski, J. (2003). Particle swarm optimization. *AIAA journal*. 41 (8): 1583-1589.

Vidal, P., and Olivera, A. C. (2014). *A Parallel Discrete Firefly Algorithm on GPU for Permutation Combinatorial Optimization Problems*. G. Hernández, C. J. Barrios Hernández, G. Díaz, C. García Garino, S. Nesmachnow, T. Pérez-Acle, M. Storti and M. Vázquez. *High Performance Computing: First HPCLATAM - CLCAR Latin American Joint Conference, CARLA 2014, Valparaiso, Chile, October 20-22, 2014. Proceedings*. (191-205). Berlin, Heidelberg: Springer Berlin Heidelberg.

Wetcharaporn, W., Chaiyaratana, N., and Tongsima, S. (2006a). *DNA Fragment Assembly by Ant Colony and Nearest Neighbour Heuristics*. L. Rutkowski, R. Tadeusiewicz, L. Zadeh and J. Żurada. *Artificial Intelligence and Soft Computing – ICAISC 2006*. (1008-1017). Springer Berlin Heidelberg.

Wetcharaporn, W., Chaiyaratana, N., and Tongsima, S. (2006b). *DNA Fragment Assembly: An Ant Colony System Approach*. F. Rothlauf, J. Branke, S. Cagnoni, E. Costa, C. Cotta, R. Drechsler, E. Lutton, P. Machado, J. Moore, J. Romero, G. Smith, G. Squillero and H. Takagi. *Applications of Evolutionary Computing*. (231-242). Springer Berlin Heidelberg.

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

Yang, X.-S. (2010). *Firefly Algorithm*. *Engineering Optimization*. (221-230). John Wiley & Sons, Inc.

Yu, J. J., Lam, A., and Li, V. O. (2015). Adaptive chemical reaction optimization for global numerical optimization. *2015 IEEE Congress on Evolutionary Computation (CEC)*. 25-28 May 2015. Sendai, Japan, 3192-3199.

Zerbino, D. R., and Birney, E. (2008). Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research*. 18 (5): 821-829.

Zhang, G. (2011). Quantum-inspired evolutionary algorithms: a survey and empirical study. *Journal of Heuristics*. 17 (3): 303-351.

Zhang, W., and Nielsen, D. R. (2014). Synthetic biology applications in industrial microbiology. *Frontiers in Microbiology,* 5 (1), 451-452. Frontiers Media S.A.

Zhou, W., Zhou, C., Huang, Y., and Wang, Y. (2005). *Analysis of gene expression data: application of quantum-inspired evolutionary algorithm to minimum sum-of-squares clustering*. D. Ślęzak, J. Yao, J. F. Peters, W. Ziarko and X. Hu. *Rough sets, fuzzy sets, data mining, and granular computing*. (383-391). Berlin, Heidelberg: Springer.