1-1-2010

# A Scientific Workflow System For Genomic Data Analysis

Jamal Ali Musleh Alhiyafi
*Wayne State University*

# A SCIENTIFIC WORKFLOW SYSTEM FOR GENOMIC DATA ANALYSIS

by

**JAMAL ALI MUSLEH ALHIYAFI**

**DISSERTATION**

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

**DOCTOR OF PHILOSOPHY**

2010

MAJOR: COMPUTER SCIENCE

Approved by:

Advisor Date

## DEDICATION

*This dissertation is dedicated to very special people in my life:*

My late father, Ali Musleh Alhiyafi

My mother, Husn Mohamed

My wife, Izdehar Alhiyafi

My father-in-law, Mohamed Mohamed

My mother-in-law, Dolah Mohamed

My children: Jihad, Maymoonah, Laith, and Waleed

*With all my love.*

# ACKNOWLEDGMENTS

First of all, I thank Allah for guiding me and making it easy for me to complete my education. I thank Him alone for blessing and protecting me and my family.

I would like to express special thanks to my advisor Dr. Shiyong Lu, whose efforts and thoughtful supervision were extraordinary throughout my PhD studies. Had it not been for Dr. Lu’s insightful comments and suggestions, this dissertation would not have been completed. As my advisor, colleague, and friend, Dr. Lu has been outstanding in providing me with his remarkable guidance, knowledge, and experience so that this research would bear fruit. I also express my thanks and appreciation to Dr. Jeffrey L. Ram for his support and excellent advice that he gave throughout the research period. I appreciate his precious encouragement and suggestions that contributed a great deal to this project. I am very grateful to Dr. Shiyong Lu, Dr. Chandan Reddy, Dr. Jing Hua, and Dr. Jeffrey L. Ram for serving on my dissertation committee and providing constructive suggestions on the direction of my project and wonderful recommendation letters.

I would like to thank and acknowledge all my co-authors during my PhD period specially Cavitha Sabesan for her collaboration during the Analysis of Intragenomic Gene Conversion phase of this research.

I would like to thank my fellow students for their encouragement and friendship: Dr. Artem Chebotko, Dr. Mustafa Atay, Dr. Yi Lu, Cavitha Sabesan, Chunhyeok Lim, Cui Lin, Dong Ruan, Xubo Fei, and the list continue to my memory. I would like to thank my friends: Dr. John Paul Walter, Jeff Ginn and Ahmed Muaydh for their helpful discussion and suggestions during this project. I would also like to thank the following staff for the great service and support in these past few years: Judy Lechvar, Alfred Glenn, and Aragorn Steiger.

My *special thanks and appreciation goes* to my mother, Husn Mohamed, whose prayers have been crucial for my accomplishments. She stood behind me and was fully supportive to my decisions since I was 13 years when my father passed away. Having a mother like her while growing up, was the greatest gift and biggest advantage anyone could ever have given me. Thank you for your unwavering support throughout my journey. Whatever I do or say, I will not be able to pay her back for a minute she spent, and still is, praying for me.

I extend my grateful thanks to my prodigious wife, Izdehar Alhiyafi, my real and continuing love, and my best friend for her help, support, and tolerance throughout my studies. She provided the comfort, affection, and love that were motivating factors toward the accomplishment of my social and academic goals. I would also extend my love and thanks to my daughters Jihad and Maymoonah and my sons, Laith and Waleed.

I also extend my thanks to my Father-in-Law, Mohamed Mohamed, and Mother-in-Law, Dolah Mohamed, for their prayers, love, and support for me, my wife, and my kids.

I also would like to extend my appreciation to my brothers: Abdulwahab, Abdulhakim, and Abu Baker, and my sisters: Tuhrah, Seham, Assma, Maryam, and Shfae for their prayers, love, affectionate concern, encouragement, and constant supports throughout my life. I would like to thank my brother-in-laws: Abdo, Khaled, Waleed, Hamza, and Yousaf, and sister-in-laws: Nabila, Yemen, Bushra, and Hella. Thanks for always being there for me.

My thanks go to special friends for always being there for me: Abdullah Hamood, Ali Jaber, Badr Albaadani, Fari Alshaibani, and Rafiq Talabah for their continuous encouragements, concerns, and wishes. Many of my relatives and friends supported me during the last few years with their encouragements and prayers, and I cannot mention them all but I would like to extend my thanks to the following relatives and friends: Uncle Abdo, My cousins, Noman, Fahd, and Abdulraqib, My friends: Amin Alsaidi, Dima Khalidi, Ibrahim Mohamed, Ibrahim Saleh, Ishraq Thabet, Mahdi Ali, Mansour Sharha, Mohamed Mohamed, Mohamed Nasser, Dr. Mustafa Hashem, Omari Bayi, and Saleh Almansoob. Also, my thanks to my friends overseas who kept asking about my PhD status and provided their encouragements for me to continue achieving my goal: Abdulrahman Alwaheeb, Abdulwahab Mayas, Bashir Aziz, Dr. Fahd Alharbi, Faisal Alyasin, Dr. Ghazi Alotaibi, Dr. Sultan Almeser, Waleed Alsabahi, Yasser Alburaihi and the everlasting list continues to my memory. Thanks for your concerns and encouragements.

Finally, I send my prayers to my father who passed away when I was in the 7th grade (21 years ago to this March). I wish he was here with me during this time to celebrate an accomplishment that he would have been proud of. I also send my prayers to my oldest brother, Hassan, who passed away 6 months ago. His encouragement and support were missed dearly. May Allah put mercy on both of them.

## TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER 1 : INTRODUCTION

In this chapter, the concept of Scientific Workflows will be introduced along with our research motivation, challenges, goals, and contributions. Finally, an organization of the rest of the dissertation is outlined.

## *1.1 Motivation, Challenges, and Approach*

Recently, scientific workflows have become increasingly popular as a new method for scientists to develop and design complex and distributed scientific processes to enable and accelerate many scientific discoveries [25, 27]. Scientific workflows are becoming an efficient way to model and automate these complex computations and automating such complex scientific experiments. Scientific workflows are rapidly becoming recognized as an important unifying mechanism to combine scientific data management, analysis, simulation, and visualization tasks [19]. A scientific workflow is a formal specification of a scientific process, which represents, streamlines, and automates the analytical and computational steps that a scientist needs to go through from dataset selection and integration, computation and analysis, to ultimately data products presentation and visualization [57, 68]. In scientific workflows, it is very important to seamlessly access and integrate various heterogeneous and distributed datasets and to integrate and reuse various third-party analysis tools. The design of a scientific workflow system often focuses on "data flows," i.e., how the input data is streamlined into various analyses using data channels to produce multiple intermediate data products and ultimately final workflow output data products.

Many scientific problems involve processing huge amounts of data and other complex computations which may require the use of many other tools to analyze data and execute accordingly.  Typically, scientists spend tremendous amounts of effort to handle manual work and computations.  The results from such computations are shared among the interested scientists for further analysis or modification. Scientific workflows can provide such a supportive and creative environment for scientists in their research by integrating complex computations and automating complex scientific analysis.

Workflow design is often concerned with the automation of procedures whereby files and data are passed between participants according to a defined set of rules to achieve an overall goal [43]. Automation is a major benefit of the scientific workflow approach. A scientist may want to execute the same data analysis pipeline repeatedly with many different input datasets and/or the same dataset using different parameter settings. Another benefit is the sharing and the reusing of computational components and/or whole workflows. Analysis and results of scientific workflows can be easily shared among collaborators. Workflow systems can be designed to easily implement additional computational steps that are eventually associated by data links to existing workflows.

Several scientific workflow systems have been developed (e.g. Taverna [68], Swift [84], Kepler [57], VIEW [53], Pegasus [28], Triana [87]).  Some are visualization-based system and others are script-based systems. Most of the systems have their proprietary language representations.  Based on a scientific problem that is being researched, a scientist may need to manually combine the results and analyses from multiple systems which can be quite time consuming.

Based on an evaluation conducted on five scientific workflow management systems (SWFMS) using the reference architecture for SWFMSs [53], none of the systems meets all the requirements. Pegasus and Swift provide weak user interaction support while Taverna, Kepler, and Triana provide better user interaction support [53]. Almost all systems have poor support for user interface customizability. Taverna, Kepler, and Triana have partial support for the integration of heterogeneous service and software tools, while Pegasus and Swift focus only on Grid-based applications [53]. Pegasus and Swift have better support to high-end computing, while other systems are being enhanced in such support. Taverna and Kepler provide custom tasks to communicate with the Grid environment, while Triana uses the GAT interface to access Grid jobs [53]. No system provides the ability to reuse existing workflows implemented by other systems.

The problems noted with existing scientific workflow management systems provide the rationale for the development of a new system for the efficient use of heterogeneous scientific workflow systems and the utilization of Grid computing. When developing a scientific workflow, some functionality may be missing in one workflow system, yet available by another, so with a system that supports heterogeneous scientific workflows, the user can develop the targeted scientific workflow. Also, scientific workflows can be shared among users. To illustrate, a user can implement a Taverna workflow and pass it to another user who does not know how to use Taverna; however he can add the workflow to his GENOMEFLOW workflow using a user friendly interface.

By analyzing the results of our research and processing time when running the Recombination Simulation Scientific Workflow [3] and Intragenomic Gene Conversion

[6], we realized the importance of a scientific workflow system that supports the execution of heterogeneous workflows using various services.

## *1.2 Contributions*

Although several scientific workflow management systems (SWFMSs) have been developed, there is a great need for an integrated scientific workflow system that enables the design and execution of higher-level scientific workflows, which integrate heterogeneous scientific workflows enacted by existing SWFMSs. On one hand, science is becoming increasingly collaborative today, requiring an integrated solution that combines the features and capabilities of different SWFMSs, which are typically developed and optimized towards one single discipline. One the other hand, such an integrated environment can immediately leverage existing and emerging techniques and strengths of various SWFMSs and their supported execution environments, such as Cluster, Grid, and Cloud.

The dissertation goal is to address the design of scientific workflow systems and how they can be utilized to serve the bioinformatic community both from the theoretical and practical perspective.

The dissertation makes the following research contributions:

- We present a scientific workflow system that can support the design, development, and execution of heterogeneous scientific workflow systems. We implemented the GENOMEFLOW scientific workflow system to design, develop, and execute heterogeneous tasks in heterogeneous environments. We present a GENOMEFLOW scientific workflow application to demonstrate the capabilities of GENOMEFLOW in support of user interaction of intensive scientific workflows in a heterogeneous and distributed computing environment [2].

- We propose a scientific workflow scheduling algorithm t o enable the parallel execution of such heterogeneous scientific workflows in their native heterogeneous environments [2].
- A scientific workflow to simulate the DNA recombination process was developed. History of the recombinant events is saved for further comparison and analysis. The known history of recombination occurring in the simulation was compared with the output of putative recombinations detected by a well known highly ranked recombination detection program (GENECONV). The results show that the recombination detection software fails to identify more than 50% of recombination events, designated as "cryptic recombinations."
- We implemented GENOMEFLOW towards the life science community and developed several GENOMEFLOW scientific workflows to demonstrate the capabilities of our system for genome data analysis applications.
- A method for generating initial populations of DNA sequences of any given Average Pairwise Difference (APD) is described [3].
- The effect of varying the recombination rate according to the pairwise difference between the potentially recombining sequences was examined [3]. Two effects of varying the recombination rate were:

a. Under some conditions of initial pairwise differences and steepness of the pairwise difference:recombination rate relationship, multiple populations could arise, illustrating a mechanism that could underlie sympatric speciation

b. Decreasing recombination rate according to the pairwise difference between potentially recombining sequences decreased the percentage of simulated recombinations that could be detected by GENECONV. Thus, the problem of 'cryptic recombination', identified by Alhiyafi et al. [4] using constant recombination rates, is exacerbated by the effect of pairwise differences on recombination rate.

- We conclude that variation in recombination rate owing to sequence mismatches should be taken into account when estimating recombination rates and when designing recombination detection experiments.
- Compare Intragenomic Gene Conversions (IGC) in the many additional *Escherichia* and *Shigella* genomes now available to test the hypothesis about the relationship of IGC frequency to pathogenicity more critically. A scientific workflow system for automating IGC analysis in multiple genomes is developed [6]. Furthermore, since sequences flanking both ends of the potential conversion sites have been shown to affect the detectability of gene conversions [4], IGCs were also analyzed with neighboring sequences included, a procedure that allowed the detection of IGCs larger and in greater numbers than had previously been described. Finally, the types of genes exhibiting IGCs were characterized.

- Considering the amount of time it takes to complete a single run-through of all currently available bacterial genomes lead us to put these principles into a high performance computing environment. Since each genome is analyzed individually, these processes are ideal for incorporating the workflow into a parallel processing grid.
- We have developed a realistic method to do bioinformatic analysis on large numbers of whole genomes, designing processors that can be used for other bioinformatic tasks and also establishing a structure into which additional bioinformatic analyses can be incorporated.

## *1.3 Organization*

The remaining chapters of the dissertation are organized as follows. Chapter 2 reviews the research on scientific workflow management systems, using a high performance environment when executing scientific workflows, scientific workflow scheduling algorithms and intragenomic recombination. Chapter 3 presents our scientific workflow system, GENOMEFLOW, and our scheduling algorithm. Chapter 4 presents the use of our scientific workflow system to simulate genomic recombination and detectability of recombination. Chapter 5 introduces our Intragenomic Gene Conversion analysis. Finally, Chapter 6 concludes the dissertation and lists some of the remaining interesting research problems.

# CHAPTER 2 : RELATED WORK

Significant research has been done in the area of scientific workflow management system and scientific workflows scheduling algorithms. In this chapter, we limit ourselves to reviewing the research that is most closely related to the work we have done here: Section 2.1 reviews research in Scientific Workflow Management Systems, Section 2.2 presents the use of Grid for scientific workflow systems, Section 2.3 introduces some of the existing Scientific Workflow Scheduling Algorithms, and Section 2.4 summarizes related work in Intragenomic Recombination. Finally, Section 2.5 gives a comparison between GENOMEFLOW and other systems.

## *2.1 Scientific Workflow Management Systems*

Many scientific workflow management systems are available today that are open source and specifically designed for different disciplines such as biology, astronomy, ecology, chemistry, engineering, and medical imaging. Scientific workflow management systems (SWFMSs) provide an environment to model, develop and run scientific workflows efficiently. Here are some examples with brief information of existing scientific workflow management systems:

**Taverna**. Taverna is a scientific workflow management environment developed by $^{my}$Grid, a UK project [68]. In Taverna, a workflow can be described as a set of processors and the relations between those processors used to define a complex process. A processor is the smallest reusable component, which performs some well-defined function within a process. Data links transfer information from workflow input, or an output of one processor, to the input of another processor, or to a workflow output. A control link enables you to set dependencies to be set between services in a workflow that do not directly share data (i.e. that are not otherwise linked by passing data from one to the other directly or indirectly). A control link allows delaying the invocation of a service until another has finished.

**VIEW.** VIsual SciEntific Workflow Management system (VIEW) has been developed by a group of researchers in the Scientific Workflow Research Laboratory at Wayne State University. VIEW  [22, 53] is a service-oriented scientific workflow management system. VIEW comprises a workbench [22] to visually design workflows, a workflow engine [33] to execute workflows, a task manager to manage the execution of workflow tasks, a provenance manager [21] to store and query workflow provenance, and a data product manager to store and manage data products. VIEW is the first system that features a service-oriented architecture conforming to the reference architecture for scientific workflow management systems [53]. Also, VIEW is the first system that supports dataflow-based MapReduce-style scientific workflows for data-intensive scientific applications [33], and it supports an innovative task abstraction and mapping technique that uniquely addresses the type-II shimming problem, which occurs due to the incompatibility between the ports of a task and the inputs/outputs of its internal task component [54].

**Swift [84].** Swift is a system for scheduling large scale scientific projects. It provides a scripting language that allows the users to express operations on datasets in terms of their local organization [102]. Swift supports the parallel and distributed execution of computationally demanding and data intensive scientific computations [83]. With Swift, the performance gains depend primarily on the parallelism that the workflow exhibits [83].

**Kepler [57].** Kepler is an open source workflow management system that is available for download at the Kepler website [48]. Kepler is based on Ptolemy II, developed at UC Berkeley, and provides a platform for building and executing workflows. Kepler is an Actor-oriented workflow system designed specifically to improve component reusability.

**Triana [87].** Triana is an open source graphical problem solving environment available for download at the Triana website [88]. Triana enables creation and execution of scientific applications, especially signal, text and image processing tasks. Triana is not only a powerful visual programming tool but also provides Grid technology as a means to providing an "easy to use" environment to scientists who may not be interested in the complex details of its implementation.

**VLE-WFBus [103].** VLE-WFBus is a scientific workflow management system developed to support workflow interoperability. Legacy SWFMSs are wrapped as federated components and are loosely coupled as one workflow system. The supported workflow systems are called subworkflows.  The VLE-WFBus support is limited to certain workflow systems.

**Pegasus [26]** (Planning for Execution in Grids) is a workflow mapping engine developed and used as part of several projects in physics, astronomy, gravitational-wave science, earthquake science, and others. Pegasus bridges the scientific domain and the execution by automatically mapping the high-level workflow descriptions onto distributed resources such as the TeraGrid, the Open Science Grid, and others.

## *2.2 Use of Grid for scientific workflow systems*

Along with the scientific workflow systems, scientific communities are utilizing Grids to share, manage and process large data sets [99]. In order to support complex scientific experiments, distributed resources such as computational devices, data, applications, and scientific instruments need to be orchestrated while managing the application workflow operations within Grid environments [61].

Grids [34] have emerged as a global cyber-infrastructure for the next-generation of e-Science applications by integrating large-scale, distributed and heterogeneous resources. Imposing the workflow paradigm for application composition on Grids offers several advantages [82] such as utilization of resources that are located in a particular domain to increase throughput or reduce execution costs, and execution spanning multiple administrative domains to obtain specific processing capabilities.

Some of the existing scientific workflow management systems have tried to exploit the advantages of using Grid technology with scientific workflows. A scientific workflow as a service [85] was implemented based on the Taverna workflow engine and gRAVI (Grid Remote Application Virtualization Interface) as the wrapping tool to improve execution performance. Scientific communities are utilizing Grids to share, manage and process large data sets [99]. In order to support complex scientific experiments, distributed resources such as computational devices, data, applications, and scientific instruments need to be orchestrated while managing the application workflow operations within Grid environments [61].

Some of the features of scientific workflow in Grid environment are: (1) resources are highly distributed, (2) scientific workflows often contain many tasks and involve large data sets which requires intensive computation, so it will be easy to use on the Grid environment [86], and (3) many computational tasks can be processed in parallel.

## *2.3 Scientific Workflow Scheduling Algorithms*

Scheduling of workflows is a problem of finding a correct execution sequence for the workflow tasks [78]. Scientific workflow scheduling in high performance computing environments usually focuses on the optimization of performance when executing workflows. In most cases, the scheduler is required to predict the performance of tasks on the various resources in order to guarantee performance [1].

The ultimate goal of the schedule is to minimize the scientific workflow execution time and maximize resource utilization or throughput. To implement a scientific workflow schedule, the following challenges will be presented [11]: (1) Precedence constraints or control dependencies where outputs from one task in the workflow will serve as inputs to other tasks. In this case, the task that produces this output should be executed as early as possible, (2) Data transfer overhead or data dependencies where all data needed for any task should be available and ready to be transferred to the compute node where the workflow task is to be executed, (3) the ability for the workflow tasks to access the required Grid resources [59], (4) Efficient selection of resources for the components in order to achieve good performance [59], and (5) satisfying all dependencies and automating the Grid execution of the entire workflow [59].

A scientific workflow is a collection of tasks organized in a way to achieve a specific target. Scientific workflow tasks can be categorized into a Simple task or a synchronization task [101]. A synchronization task is a task that has more than one parent or child task. Synchronization Task Scheduling (STS) [101] only considers one task to decide the service for executing that task. If there is only one simple task in a branch (Branch Task Scheduling (BTS)) then the solution for BTS is the same as STS. However, if there are multiple tasks, the scheduler needs to make a decision on which service to execute each task after the completion of its parent task [101]. Simple or Static Scheduler is based on well-known advance reservation based co-allocation techniques. Static scheduler performs sub-deadline re-calculation and re-negotiation if the initial co-allocation request fails, whereas the Simple scheduler does not. Dynamic Scheduler is where the tasks of an application are scheduled Just-in-time. Scheduling of scientific workflows on the Grid is a complex optimization problem which may require consideration of different scheduling criteria. The most important criteria are the expected execution time and the cost of running an activity on a machine [71].

**Min-Min [56, 59].** The Min-Min heuristic algorithm makes decisions based on a set of parallel independent tasks. It assigns priority and schedules the task based on its Expected Completion Time (ECT) for the task over all available resources. It produces good results but is computationally expensive [32]. For each component, the resource having the minimum estimated completion time (ECT) is found. The component having the minimum ECT value is chosen to be scheduled next. This is done iteratively until all the components have been mapped. The concept behind Min-Min is to consider all unmapped independent tasks during each mapping decision. [73].

**Max-Min.** The first step is exactly same as in the Min-Min heuristic. Then the resource having the maximum estimated completion time (ECT) is found and the corresponding component is mapped instead of choosing the minimum. The intuition behind this heuristic is that by giving preference to longer jobs, there is a hope that the shorter jobs can be overlapped with the longer job on other resources [59]. It is similar to Min-Min except that in each iterative step, a task having the maximum ECT is chosen to be scheduled on the resource that is expected to complete the task at the earliest time. Once the machine that provides the earliest completion time is found for every task, the task that has the maximum earliest completion time is determined and then assigned to the corresponding machine [58]. Intuitively, Max-Min attempts to minimize the total workflow execution time by assigning longer tasks to comparatively better resources. Both Min-Min and Max-Min have been used for scheduling workflow tasks in Pegasus [59, 73]. The Max-Min is likely to do better than the Min-Min heuristic in the cases where we have many more shorter tasks than long tasks [58].

**Sufferage.** The first step in the Sufferage algorithm [59] is to find both the minimum and second best minimum ECT values. The difference between these two values is defined as the sufferage value. In the second step, the component having the maximum sufferage value is chosen to be scheduled next. The intuition behind this is that jobs are prioritized on relative affinities. The job having a high sufferage value suggests that if it is not assigned to the resource for which it has minimum ECT, it may have an adverse effect because the next best ECT value is far from the minimum ECT value. A high sufferage value job is chosen to be scheduled next in order to minimize the penalty of not assigning it to its best resource [59].

**Bi-criteria Scheduling.** Bi-criteria Scheduling [71] restricts the user to certain criterion pairs. It requires the user to identify preferences either as (1) weights assigned each criterion or as (2) fixed constraints defined for one criterion. The first approach has the drawback that combining multiple criteria into a single objective function is not always intuitive to the end-user, while the second requires a priori knowledge about the result of the first criterion scheduling result.

**Dynamic Constraint Algorithm (DCA).** DCA addresses the optimization problem of the two independent criteria: execution time and economic cost tradeoff. DCA is based on dynamic programming [71]. The user is expected to identify primary and secondary criteria, and a sliding constraint. DCA algorithm consists of two phases: (1) primary scheduling for optimizing for the primary criterion only; and (2) secondary scheduling for optimizing for the secondary criterion while keeping the primary criterion cost within the defined sliding constraint.

**Heterogeneous Earliest Finish Time (HEFT).** HEFT algorithm makes decisions based on a critical path of the tasks. This strategy selects the task with the highest upward rank value at each step. This is the length of the critical path from a task to the exit task, including the computation cost of this task. It assigns higher priority to the workflow task having higher rank value. It calculates rank value based on the average execution time for each task and average communication time between resources of two successive tasks, where the tasks in the 'critical path' get comparatively higher rank values. The selected task is then assigned to the processor that minimizes its earliest finish time. The algorithm is designed for scheduling DAG (Directed Acyclic Graph) so it is not so efficient for scheduling a huge number of concurrent workflow instances [56]. The advantage of using this technique over Min-Min or Max-Min is that while assigning priorities to the tasks, it considers the entire workflow rather than focusing on only unmapped independent tasks at each step [73].

**Greedy Randomized Adaptive Search Procedure (GRASP).** GRASP is an iterative randomized search technique. In GRASP, a number of iterations are conducted to search a possible optimal solution for mapping tasks on resources. A solution is generated at each iterative step and the best solution is kept as the final schedule. This searching procedure terminates when the specified termination criterion, such as the completion of a certain number of iterations, is satisfied. GRASP can generate better schedules than the other scheduling techniques stated previously as it searches the whole solution space considering entire workflow and available resources [18, 73].

**Portable Batch System (PBS) [38, 67].** The Portable Batch System is a management and batch job scheduling system. It schedules and distributes various types of application runs (serial, parallel, distributed memory, etc.). It supports parallel programming libraries such MPI and openMP. It also can be used as a scheduler for scientific workflow systems such as Swift or can serve as a front end to Globus, permitting the user to submit jobs requesting Globus resources using the normal PBS Pro commands.

**In conclusion**, Min-Min, Max-Min, Sufferage, Greedy Randomized Adaptive Search Procedure (GRASP), and Heterogeneous-Earliest-Finish-Time (HEFT) algorithm only attempt to minimize workflow execution time and do not consider users' budget constraints [100]. Several tools, including Kepler, Triana, and Taverna, provide interfaces and tools to specify and execute scientific workflows. The emphasis of these tools is on formalizing and constructing workflows and providing access to heterogeneous data and distributed web services [20].

## *2.4 Intragenomic Recombination*

DNA consists of many genes and provides genetic information to regulate and reproduce cells. Information from each gene encodes a unique protein, which performs necessary tasks for the cell to function. Bacteria are very small single cell organisms that are found almost everywhere, in the air, water, soil, food, and human body. Bacteria are prokaryotes, which indicate that they contain a single cell that does not contain a nucleus. Instead, their genetic information is within a single circular chain of DNA. Even though they are small organisms, many live in groups and can multiply quickly by cell division, by which a single cell splits into two new daughter cells both with the same genetic material, and can conjugate (~ have sex) with each other to exchange and insert genetic material from one cell to another by homologous recombination. Recombination of bacterial genomes is widespread, occurring in soil bacteria [91] and numerous pathogens, such as *Neisseria meningitides* [39]. Recombination is the transfer, insertion or replacement of a length of DNA into a genome from another source and results in the exchange of genetic information between organisms [70]. The seminal discovery of recombination by bacterial conjugation was made by Lederberg [49]. During conjugation, a single strand of replicated DNA is transferred sequentially through a bridge connecting the donor to the recipient cell. Genes or fragments of genes can integrate into the recipient's chromosome by homologous recombination, in which portions of the recipient cell's chromosome are replaced by similar or identical sequences from the donor. The result of this one-way transfer of DNA sequence is known as gene conversion.

Gene conversion can occur between two cells or between different genomic regions of the same cell. Intragenomic Gene Conversion (IGC), which is an outcome of a recombination event, is defined as the non-reciprocal transfer of genetic information from one gene to another related gene elsewhere in the genome. IGCs play an important role in the evolution of multigene families of bacteria and the generation of antigenic variations, where some pathogenic bacterial strains are avoiding the host immune system [68]. More details on the occurrence of IGCs are given below.

Computer modeling of genomic data is a powerful tool for simulating mechanisms maintaining diversity and mediating evolution of organisms and for testing methods measuring such mechanisms. For example, bacterial populations are highly diverse. Whittam estimates average pairwise differences over several genes of *E. coli* of about 2% [96]. Average pairwise differences reported by Alhiyafi et al. [4] in the *fimH* and *uidA* genes in natural populations of *E. coli* were 2.1% and 2.0%, respectively. While point mutations can certainly create pairwise differences between bacterial strains, genetic change can also come about through horizontal gene transfer by recombination. A central problem in understanding the diversity and evolution of organisms, therefore, is knowing how much genetic change has come about through recombination. This problem is complicated by the fact that recombination is both affected by sequence diversity and is a mechanism that helps maintain it. Evidence that recombination frequency between strains is decreased by higher levels of DNA sequence mismatches has been found in many studies. Among recombinations of five bacterial strains (two strains of *E. coli*, and one each of *Shigella flexneri*, *E. fergusoni*, and *Salmonella typhimurium*, each reciprocally crossed with *E. coli* K12) recombination frequency decreased exponentially with amount of DNA sequence divergence [92]. 10% divergence of DNA sequences decreased recombination frequency by approximately three orders of magnitude. Similarly, recombinations between *E. coli* strains from Lenski's long-term closed cultures [50, 51] decreased in frequency approximately 0.5 log units with a 0.2% sequence divergence [93]. Extrapolating their data suggests a decrease of 5 orders of magnitude for a 2% sequence divergence. In studies of recombination between homologous DNA sequences in plasmids and phage lambda in *E. coli* (K12) hosts, Watt et al. [94] observed a 2- to 4-fold decrease in recombination frequency with a single mismatched base out of 53, and Shen and Huang [79] observed up to a 300-fold decrease in recombination with a 16% DNA sequence mismatch. Fraser et al. [35] also summarizes data in *Bacillus subtilis*, *Bacillus mojavensis*, *Streptococcus pneumoniae*, and *E. coli* that suggest an average 1000-fold decrease in recombination with a 16% DNA sequence mismatch. While considerable variations in the quantitative estimates are apparent, there is no disagreement with the general observation that recombination frequency decreases with greater pairwise difference between potentially recombining genomes.

Although recombination is usually thought of as occurring between two different cells, IGCs can also occur in which recombination occurs between genes in gene families, i.e., genes with similar sequence that can be found in the same genome. For example, the concerted evolution of multiple copies of genes coding for ribosomal RNA may depend on intragenomic recombination between the homologous copies to keep all copies more or less “in synch” [52]. In addition to the concerted evolution of rRNA genes, intragenomic recombination has been demonstrated experimentally between the *tufA* and *tufB* genes in *Salmonella typhimurium* [9, 46]  and the *gadA* and *gadB* genes in *Escherichia coli* [15]. Since the main requirement for this type of genetic exchange to occur is similarity of sequence, families of genes sharing similar sequence within a genome may be fertile ground for IGCs. IGC in bacteria has been suggested to occur more frequently in pathogenic species or strains than in non-pathogenic strains [63, 64, 75]. A computational analysis comparing the genomes of a non-pathogenic strain of *E. coli* (K12) and three pathogenic strains (CFT073 & O157:H7 Sakai and EDL933) concluded that IGCs were more frequent and require less sequence similarity in the pathogenic strains than in K12 [63]. One of the major issues considered in this dissertation is whether the differences in IGC frequency between pathogenic and non-pathogenic *E. coli* are representative of a general difference between pathogenic and non-pathogenic bacteria. However, the conclusion of Morris and Drouin [63] that “gene conversions are more frequent and much less dependent on sequence similarity in pathogenic strains than in K-12" depended on a comparison of only one non-pathogenic strain and a select subset of pathogenic *E. coli* strains. In contrast, a subsequent study of the same four genomes by Morris and Drouin found no difference in the Intragenomic Gene Conversion rates when only "backbone" genes were considered, i.e., only those genes found in common among the four strains [65]. A larger number of whole genome sequences of other pathogenic and non-pathogenic strains of *E. coli* are now available for determining the representativeness of their conclusions. In addition, other bacterial species in which such pathogen:non-pathogen comparisons can be made are also available.

In order to detect whether recombination or gene conversion has occurred in the past, divergent sequences can be analyzed for vestiges of previous genetic transfers. Many methods have been developed during the last 15 years to detect the presence of recombination in sequence alignments. Methods work by identifying discontinuities in sequence similarities or genetic distance (e.g., GENECONV [76, 77] and MAXCHI [70, 80] or by phylogenetic methods that, for example, identify incongruous tree topologies (e.g., RECPARS[42]). Many methods have been evaluated [69, 70, 97], and new ones continue to be developed. GENECONV is among the most highly ranked methods and is included in the RDP2 analysis suite [60]. In a comparison of 14 different gene recombination detection methods, GENECONV was consistently among the best [69, 70]. GENECONV has been applied successfully to various biological data sets to analyze recombination among alleles of homologous genes [10, 15, 31, 89]. GENECONV was also used to demonstrate IGCs between genes in gene clusters in both yeast and *E. coli* genomes[29, 63] and was the method used to compare pathogenic and non-pathogenic strains of *E. coli*. The detection of recombination from DNA sequences by such methods is therefore relevant to the understanding of evolutionary and molecular genetics in bacteria.

## *2.5 A comparison between GENOMEFLOW and related systems*

The problems noted with existing scientific workflow management systems and scheduling algorithms provide the rationale for the development of a new system for the efficient use of heterogeneous scientific workflow systems and the utilization of Grid computing. In this dissertation, we propose a scientific workflow system, GENOMEFLOW, with a scheduling algorithm to execute heterogeneous scientific workflows.

GENOMEFLOW scientific workflow system has the following specifications and advantages compared to other scientific workflow systems:

- GENOMEFLOW is a scientific workflow system that is designed and implemented by following the Reference Architecture for Scientific Workflow Management Systems.
- GENOMEFLOW has a user friendly interface, workbench, which gives the user the ability to design scientific workflows by clicking on existing tasks and adding them to the workflow design window. No need to learn a scripting language (e.g. Swift scripting language) in order to develop the workflow.

- GENOMEFLOW has a user friendly interface, workbench, which gives the user the ability to design scientific workflows by clicking on existing tasks and adding them to the workflow design window. No need to learn a scripting language (e.g. Swift scripting language) in order to develop the workflow.
- GENOMEFLOW scientific workflow system supports the design, development, and execution of heterogeneous scientific workflow systems. When developing a scientific workflow, a user may need to use a function that is missing from the workflow system that he is using, yet available by another, so with a system that supports heterogeneous scientific workflows, the user can develop the targeted scientific workflow.
- Scientific workflows can be shared among users. To illustrate, a user can implement a Taverna workflow and pass it to another user who does not know how to use Taverna; however he can add the workflow to his GENOMEFLOW workflow using a user friendly interface.
- GENOMEFLOW scientific workflow system supports the execution of tasks in a heterogeneous environment.
- GENOMEFLOW executes tasks using its own scheduling algorithms. It executes heterogeneous scientific workflows in parallel processing in a high performance computing environment. GENOMEFLOW scheduling algorithms optimize the process and execution time of the GENOMEFLOW workflow. Most of the existing scheduling algorithms depend on knowing in advance how

workflow tasks work and setting criteria or priority weights that will help make scheduling decision.

- Using GENOMEFLOW will give the user the advantages of using two levels of execution: local and global. The GENOMEFLOW scheduling algorithm looks at the workflow as a whole and also focuses on GENOMEFLOW individual tasks. It then uses the local scheduler (e.g. PBS) to minimize the scientific workflow execution time and maximize resource utilization or throughput.

# CHAPTER 3 : GENOMEFLOW Scientific Workflow System

In this chapter, we introduce our GENOMEFLOW Scientific Workflow System and propose our scientific workflow scheduling algorithm. The rest of the chapter is organized as follows. Section 3.1 discusses motivation and challenges for our research. Section 3.2 presents the service-oriented architecture for GENOMEFLOW. Section 3.3 introduces the GENOMEFLOW Specification Language (GSL). Section 3.4 defines the scheduling algorithm. Section 3.5 presents the system execution and evaluation. Finally, Section 3.6 summarizes the chapter.

## *3.1 Motivation*

Scientific workflows have become increasingly popular as a new computing paradigm for scientists to design and execute complex and distributed scientific processes to enable and accelerate many scientific discoveries. Although several scientific workflow management systems (SWFMSs) have been developed, there is a great need for an integrated scientific workflow system that enables the design and execution of higher-level scientific workflows, which integrate heterogeneous scientific workflows enacted by existing SWFMSs. On one hand, science is becoming increasingly collaborative today, requiring an integrated solution that combines the features and capabilities of different SWFMSs, which are typically developed and optimized towards one single discipline.  One the other hand, such an integrated environment can immediately leverage existing and emerging techniques and strengths of various SWFMSs and their supported execution environments, such as Cluster, Grid, and Cloud.

Based on an evaluation conducted on five scientific workflow management systems (SWFMS) using the reference architecture for SWFMSs [53], none of the systems meets all the requirements. Also, by analyzing the results of our research and processing time when running a recombination simulation scientific workflow [4] and another scientific workflow to study the Intragenomic Gene Conversion [5, 6], we realized the importance of a scientific workflow system that supports the execution of heterogeneous workflows using various services. The advantages of high-performance Grid-based computing for scientific workflows, and the problems noted with existing scientific workflow management systems (SWFMSs), provide the rationale for the development of a new scientific workflow system for the efficient use of heterogeneous scientific workflow systems and utilizing Grid computing.

## *3.2 Service-Oriented Architecture for GENOMEFLOW*

The system was designed and implemented by following the Reference Architecture for Scientific Workflow Management Systems [53]. The system consists of the following major sub-systems (Figure 3.1A): Interface and workflow design subsystem, Workflow Engine, Workflow Monitor, Task Manager, Data Product Manager, and the Provenance Manager. The interface and design subsystem is represented by a workbench that will give the user the ability to visually design the scientific workflow.

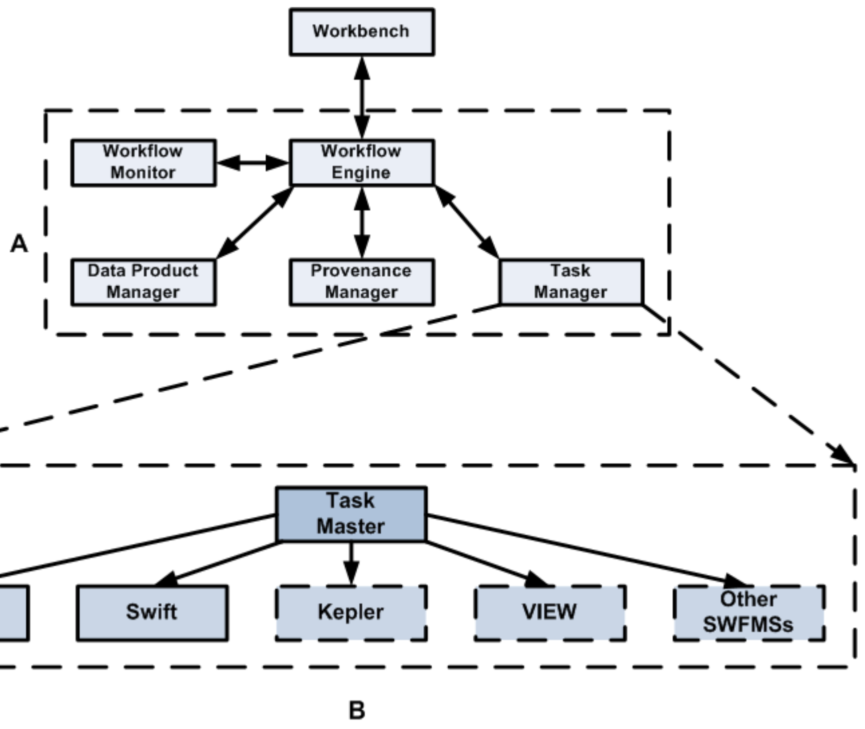


Figure 3.1: (a) Service-oriented architecture for the GENOMEFLOW system; (b) GENOMEFLOW Task Manager for the execution of heterogeneous tasks.

A list of available built-in tasks is available. If a task is not available, the user can create a new task which will be then available for future usage as well (Figure 3.2). A workflow consists of one or more tasks and each task has input/output ports. Each task represents a workflow that was built by a scientific workflow developed by a third party SWFMS. The user will have to enter data for the input ports and/or create data links between input ports and output ports of another task which indicate a dependency step.

The Workflow Monitor monitors the workflow and task execution status. The Task Manager will execute workflow tasks. The Data Product Manager will store, query, and manage data products, and the Provenance Manager stores and queries provenance. Database storage, along with a file repository, holds the task workflows and implemented workflows. The results from running the workflows will also be saved in the storage based on the name of the workflow. Inputs and outputs are transferred back and forth between the local machine and the remote machines when necessary; if the actual execution is occurring remotely.

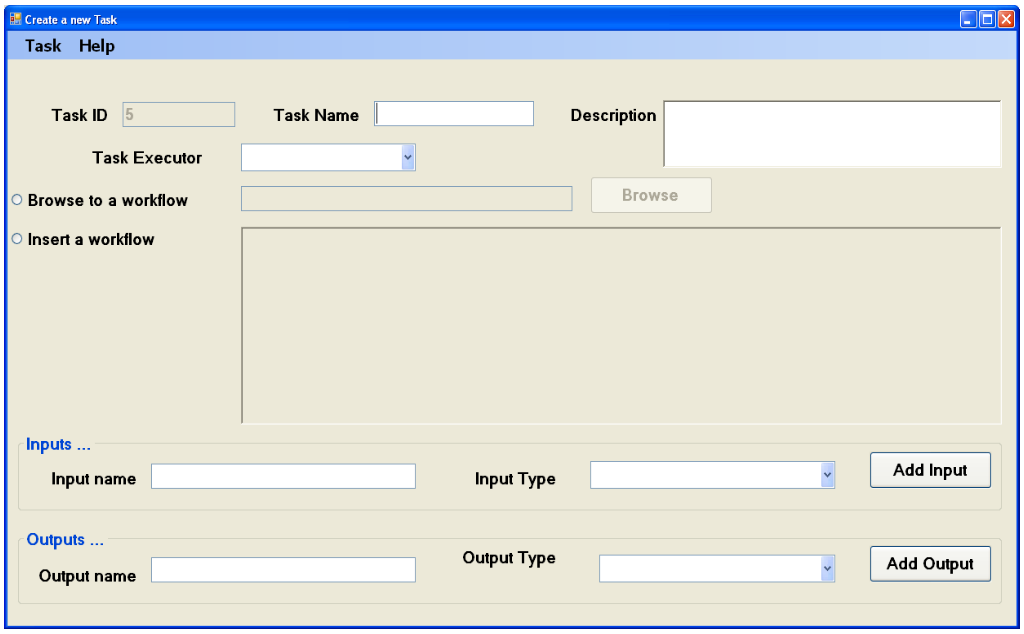


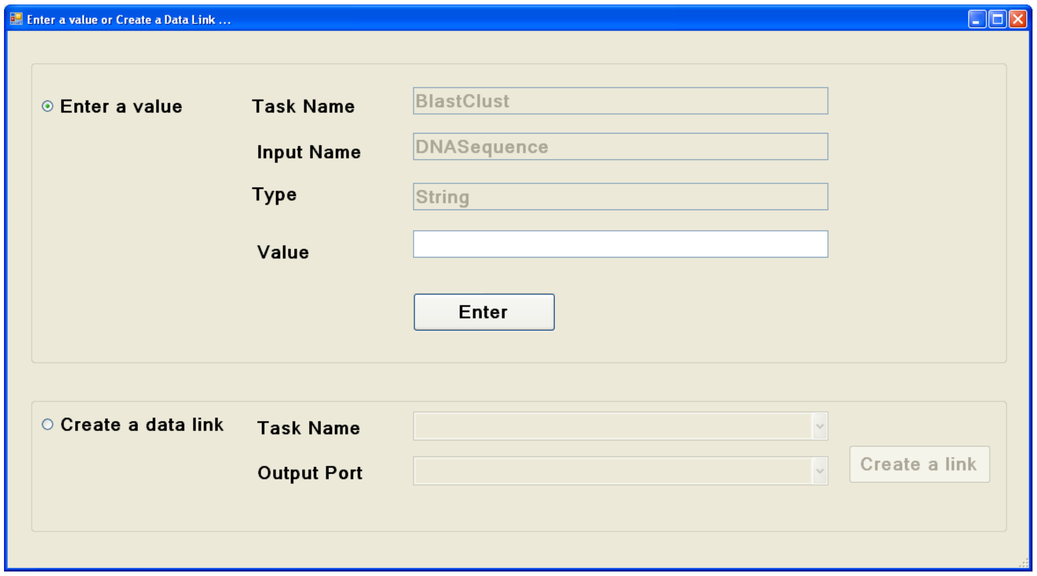
Figure 3.2: Create a GENOMEFLOW workflow task.

Figure 3.1B represents a service-oriented architecture for the Task Manager. The Task Manager consists of heterogeneous scientific workflows and their executable programs with other software tools, including high-end computing environments. The execution of the workflow starts by transferring the GENOMEFLOW Specification Language (GSL) file (.xml) that represents the workflow and internal tasks to the remote machine along with job script execution file. The job script file is called to start the execution and once it finishes, the resultant data is transferred back to the local machine.

When implementing GENOMEFLOW, we applied the key architectural requirements for a SWFMS [53]. The system has user interface and support user interaction (Requirement 1). The end user can develop scientific workflows by using the user friendly workbench. One of the user interfaces is to create control or data links between the workflow tasks (Figure 3.3). The system also meets the second requirement that it supports reproducibility of results by re-executing the workflow. Provenance data will be stored to support this requirement. The core of this system is to support heterogeneous and distributed services and tools (Requirement 3). GENOMEFLOW gives the end user the ability to develop workflows that contain tasks implemented by heterogonous scientific workflow systems and tools. As shown in Figure 3.1B, the Task Manager can execute tasks that represent Taverna workflow, Swift workflow, etc. The system also supports heterogeneous and distributed data product (Requirement 4). GENOMEFLOW also supports high-end computing by executing tasks and subworkflows in parallel mode on the Grid when needed.

## *3.3 GENOMEFLOW Specification Language (GSL)*

Many systems were developed to execute scientific workflows as described in section 2.1 and each one has its own language to represent its workflow. Each system only supports workflows designed and implemented by itself. However, GENOMEFLOW supports heterogeneous workflows that were implemented by other systems and will be connected and represented via GSL. In this section, I will introduce the GENOMEFLOW Specification Language (GSL) that will represent a scientific workflow consisting of one or more tasks. Each task represents a workflow implemented by any one of the other systems. Two tasks are represented in the sample GSL (Figure 3.4); one for a Swift workflow and the other one for a Taverna workflow. The language will also show the data links between tasks where an output of a task can be an input for another one. Figure 3.5 displays a schema representation for the GSL file. It shows that a workflow can contains one or more tasks. Each task contains inputs and outputs. The workflow also can have data links to represent data dependency or control dependency.


Enter a value or Create a Data Link ...

[✓ Enter a value]

| | |
|---|---|
| Task Name | BlastClust |
| Input Name | DNASequence |
| Type | String |
| Value | |

Create a data link

| | |
|---|---|
| Task Name | |
| Output Port | |

Figure 3.3: GUI to enter input data for a task or create a data link between two tasks.

```
< Workflow>
<Workflow_Name>IGC-DownloadFiles</Workflow_Name>
<Workflow_ID>10</Workflow_ID>
<Workflow_Description>test</Workflow_Description>
<Tasks>
 <Task>
  <Task_Name>CreateGenomeFiles</Task_Name>
  <Task_ID>20</Task_ID>
  <Task_Description>Create Genome Files</Task_Description>
  <Task_Executor>Swift</Task_Executor>
  <Task_Workflow_Content>
     type file{};     type messagefile {};
      ……..
  </Task_Workflow_Content>
  <Inputs><Number_of_Inputs>1</Number_of_Inputs>
    <Input>  <Input_Name>GenomeList</Input_Name>
    <Input_Type>File</Input_Type>
    </Input>
  </Inputs>
  <Outputs> <Number_of_Outputs>1</Number_of_Outputs>
    <Output>
    <Output_Name>Input</Output_Name>
    <Output_Type>Files</Output_Type>
    </Output>
  </Outputs>
</Task>
<Task><Task_Name>GetGenomeInfo</Task_Name>
  <Task_ID>11</Task_ID>
  <Task_Description>……</Task_Description>
  <Task_Executor>Taverna</Task_Executor>
  <Task_Workflow_Content>
 <s:scufl xmlns:s="http://org.embl.ebi.escience/xscufl/0.1alpha"  version="0.2" log="0">
   ……..
  </Task_Workflow_Content>
  <Inputs> <Number_of_Inputs>1</Number_of_Inputs>
    <Input> <Input_Name>Bacteria</Input_Name>
    <Input_Type>String</Input_Type>
    </Input>
  </Inputs>
  <Outputs> <Number_of_Outputs>4</Number_of_Outputs>
    <Output>  <Output_Name>NCBIRef</Output_Name>
    <Output_Type>String</Output_Type>
    </Output>
     ……
  </Outputs>
</Task>
</Tasks>
<Data_Links><Number_of_DataLinks>20</Number_of_DataLinks>
   <Data_Link>
    <Source><Task_Name>CreateGenomeFiles</Task_Name>
      <Port_Name>Input</Port_Name>
      <Port_Type>Files</Port_Type>
    </Source>
    <Destination><Task_Name>GetGenomeInfo</Task_Name>
      <Port_Name>Bacteria</Port_Name>
      <Port_Type>String</Port_Type>
    </Destination>
   </Data_Link>
    ...
</Data_Links>
</Workflow>
```

Figure 3.4:  Example of a GENOMEFLOW workflow specification.

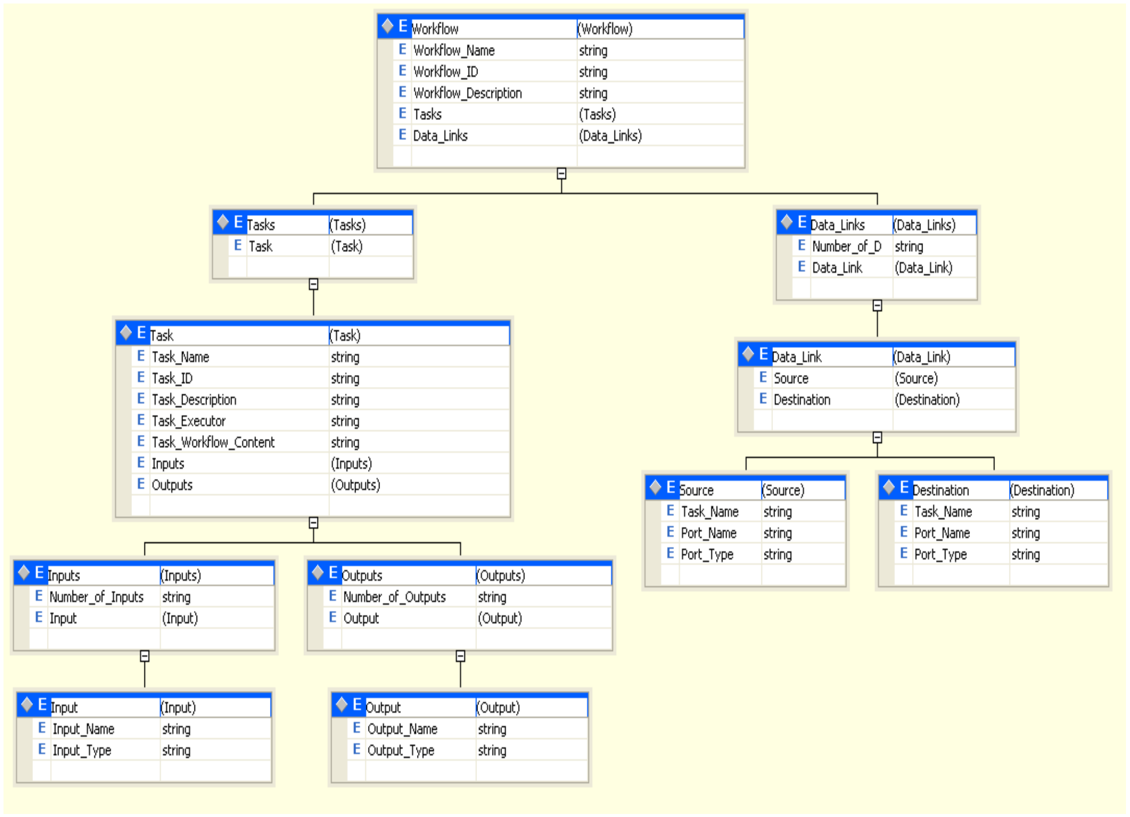


Figure 3.5: A schema representation for a GSL file.

## *3.4 GENOMEFLOW Scheduling Algorithm (GSA)*

In this section, we propose a scheduling algorithm to execute scientific workflows in order to optimize the process and execution time of the GENOMEFLOW workflow.

Scheduling of workflows is a problem of finding a correct execution sequence for the workflow tasks [78]. One of the features that distinguishes various scientific workflow systems is the scheduling algorithm that it uses. The goal is to optimize the execution of the scientific workflow task while utilizing the available resources. Most of the existing scheduling algorithms depend on knowing in advance how workflow tasks work and setting criteria or prioritizing weights that will help make scheduling decisions. Some systems use either task scheduling, which only looks at the tasks within the workflow, and others look at the workflow as a whole. One of the main features for GENOMEFLOW is that it supports the design of scientific workflows from heterogeneous scientific workflow systems. Since each GENOMEFLOW task is representing a scientific workflow system, it automatically uses the scheduling algorithms integrated with the task executor. In addition to this, we propose our GENOMEFLOW Scheduling Algorithm (GSA) that will look at the whole workflow. We schedule the execution of each task based on control and data dependency and based on the available resources.

Algorithm *GSA*, given in Figure 3.6 corresponds to the scheduling algorithm implemented in GENOMEFLOW. The end user will design a scientific workflow using the GENOMEFLOW workbench. Once it is saved, the system will follow the algorithm to prepare the execution job script files. The input for the algorithm will be an XML file representing the GENOMEFLOW workflow designed in the previous step.

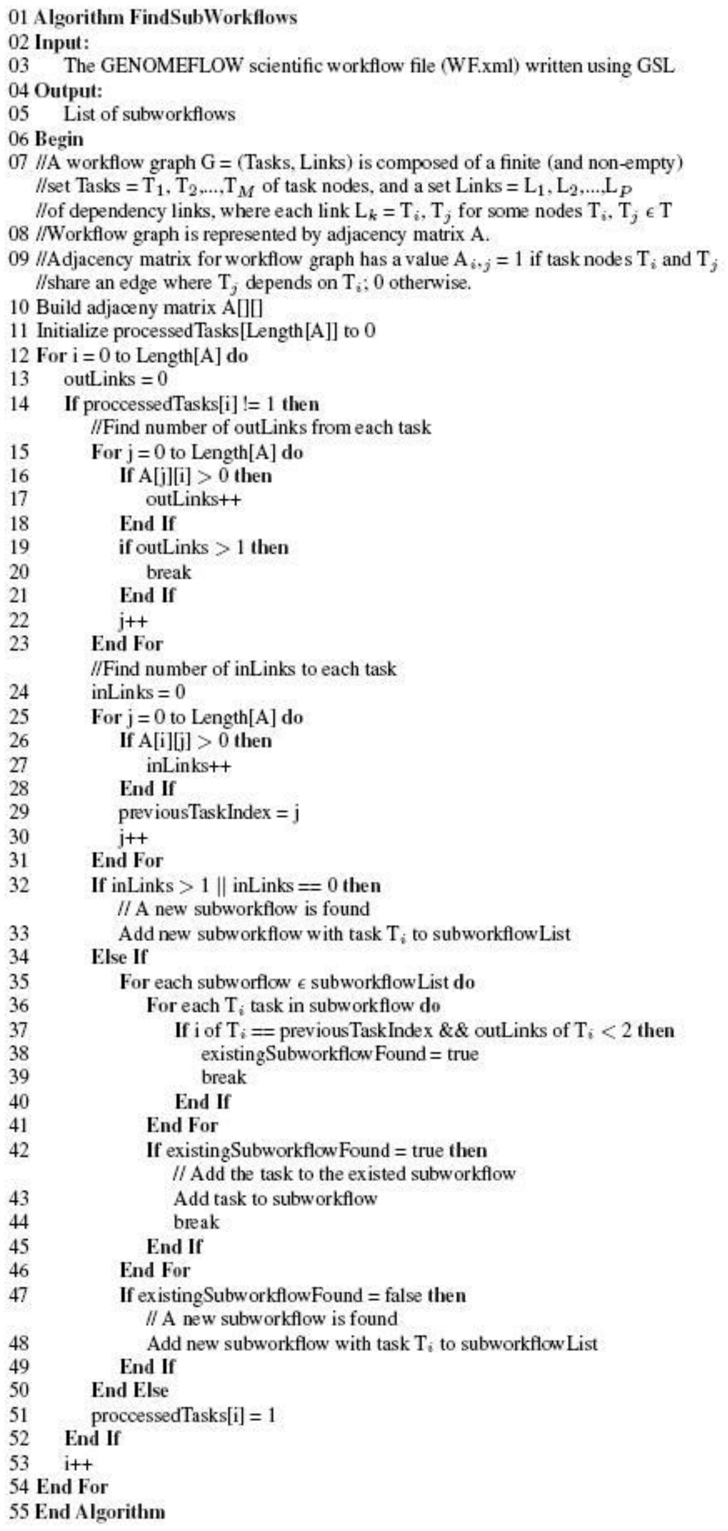
The XML will contain information about the tasks involved in the workflow, inputs/outputs for each task, and the task executor. A task executor is the program that will understand and execute the workflow language representation integrated in each task. The output will be the results from executing each task. In line 8, the system will parse the XML file and capture a list of all tasks with inputs (name, type) associated with each task. In line 9, based on the dependency of each task, a list of subworkflows will be created by applying algorithm *FindSubWorkflows* (Figure 3.8). In line 10, a dependency list will be created based on parsing the workflow file and the data link section of the workflow file. In line 11, corresponding job scripts to execute each subworkflow will be prepared (waiting list). In line 14, the algorithm starts going through each job script in the waiting list.

```
01 **Algorithm GSA**
02 **Input:**
03        The GENOMEFLOW scientific workflow file (WF.xml) written using GSL
04        The input data for the workflow
05 **Output:**
06        The output after running the workflow
07 **Begin**
08    WF = ({Task$_1$, Task$_2$,...., Task$_N$}, inputs) // Parse the WF.xml to identify tasks
09    WFList = [W$_1$, W$_2$,...., W$_N$] // Partition WF based on dependency to subworkflows
10    DependencyList = [D$_1$, D$_2$,...., D$_N$] // List of parent suborkflow for each subworkflow in WFList
11    WaitingJobs = [J$_2$, J$_2$,...., J$_N$] // Create Job script files. Each Job will execute a corresponding subworkflow
12    RunningJobs = $\phi$
13    CompletedJobs = $\phi$
14    **For** each $J_i \in$ WaitingJobs **do**
15        **If** subworkflow $\in J_i$ depends on NULL **Then**
16            Get list of available resources on the Grid
17            Submit the job $J_i$ to start the execution
18            RunningJobs = RunningJobs $\cup J_i$
19            WaitingJobs = WaitingJobs - $J_i$
20        **End If**
21    **End For**
22    **Do**
23        **For** each $J_i \in$ RunningJobs **do**
24            **If** $J_i$ is done **Then**
25                Collect output resulted from executing $J_i$
26                RunningJobs = RunningJobs - $J_i$
27                CompletedJobs = CompletedJobs $\cup J_i$
28            **End If**
29        **End For**
30        **For** each $J_k \in$ WaitingJobs **do**
31            **If** $J_k$ depends on jobs $\in$ CompletedJobs **Then**
32                Get list of available resources on the Grid
33                Submit the job $J_k$ to start the execution
34                RunningJobs = RunningJobs $\cup J_k$
35                WaitingJobs = WaitingJobs - $J_k$
36            **End If**
37        **End For**
38    **While** WaitingJobs is not NULL
39 **End Algorithm**
```

Figure 3.6: Algorithm *GSA*.

If a subworkflow does not depend on any previous tasks then the job script file that contains this subworkflow will be executed after allocating resources necessary to execute it. All the jobs that were executed will be added to the running jobs list (line 18) and deleted from the waiting list (line 19). Once going through all the jobs for the first round, the algorithm parses through all the jobs that are running.  For any job that was done, its output files are collected (line 25), it is deleted from the running jobs (line 26), and the completed task is added to the completed jobs list (line 27). Then the algorithm goes through all the jobs in the waiting list (lines 30-37) as follows: For every job *J* in the waiting list, if the subworkflow that it contains depends on another subworkflow that was executed and completed already, job *J* is executed after allocating the necessary resources. *J* is also added to the running list and deleted from the waiting list. The last steps (lines 23-37) are repeated until all the jobs in the waiting list have been executed and the waiting list doesn't contain any jobs to run.

Algorithm *FindSubWorkflows,* given in Figure 3.7 corresponds to the algorithm implemented to find all the subworkflows within a GENOMEFLOW workflow and it represents line 9 of algorithm *GSA* (Figure 3.6).



```
01 Algorithm FindSubWorkflows
02 Input:
03     The GENOMEFLOW scientific workflow file (WF.xml) written using GSL
04 Output:
05     List of subworkflows
06 Begin
07 //A workflow graph G = (Tasks, Links) is composed of a finite (and non-empty)
   //set Tasks = $T_1$, $T_2$,...,$T_M$ of task nodes, and a set Links = $L_1$, $L_2$,...,$L_P$
   //of dependency links, where each link $L_k$ = $T_i$, $T_j$ for some nodes $T_i$, $T_j$ $\epsilon$ T
08 //Workflow graph is represented by adjacency matrix A.
09 //Adjacency matrix for workflow graph has a value $A_{i,j}$ = 1 if task nodes $T_i$ and $T_j$
   //share an edge where $T_j$ depends on $T_i$; 0 otherwise.
10 Build adjaceny matrix A[][]
11 Initialize processedTasks[Length[A]] to 0
12 For i = 0 to Length[A] do
13     outLinks = 0
14     If proccessedTasks[i] != 1 then
           //Find number of outLinks from each task
15         For j = 0 to Length[A] do
16             If A[j][i] $>$ 0 then
17                 outLinks++
18             End If
19             if outLinks $>$ 1 then
20                 break
21             End If
22             j++
23         End For
           //Find number of inLinks to each task
24         inLinks = 0
25         For j = 0 to Length[A] do
26             If A[i][j] $>$ 0 then
27                 inLinks++
28             End If
29             previousTaskIndex = j
30             j++
31         End For
32         If inLinks $>$ 1 || inLinks == 0 then
               // A new subworkflow is found
33             Add new subworkflow with task $T_i$ to subworkflowList
34         Else If
35             For each subworflow $\epsilon$ subworkflowList do
36                 For each $T_i$ task in subworkflow do
37                     If i of $T_i$ == previousTaskIndex && outLinks of $T_i$ $<$ 2 then
38                         existingSubworkflowFound = true
39                         break
40                     End If
41                 End For
42                 If existingSubworkflowFound = true then
                       // Add the task to the existed subworkflow
43                     Add task to subworkflow
44                     break
45                 End If
46             End For
47             If existingSubworkflowFound = false then
                   // A new subworkflow is found
48                 Add new subworkflow with task $T_i$ to subworkflowList
49             End If
50         End Else
51         proccessedTasks[i] = 1
52     End If
53     i++
54 End For
55 End Algorithm
```

Figure 3.7: Algorithm *FindSubWorkflows*.

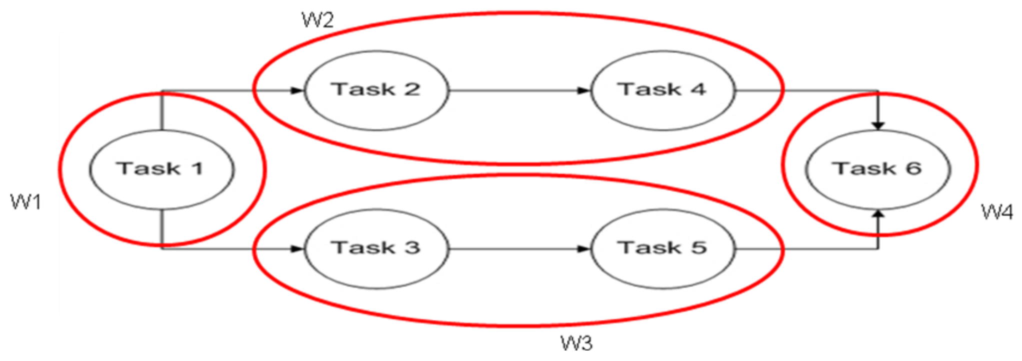


Figure 3.8: Example to illustrate the scheduling and portioning algorithms.

Figure 3.8 illustrates both algorithms: *GSA* and *FindSubWorkflows*. For a workflow that consists of 6 tasks, the arrows represent the flow of tasks and dependency. Based on both algorithms, the following tasks and lists are generated:

Line 8 of *GSA*: WF = ({$Task_1$, $Task_2$, ..,$Task_6$}, …..) Equation 3.1

Line 9 of *GSA* which is the outcome of running *FindSubWorkflows*:

WFList = [$W_1$($Task_1$), $W_2$($Task_2$, $Task_4$), $W_3$($Task_3$, $Task_5$),$W_4$($Task_6$)] Equation 3.2

Line 10 of *GSA*:

DependencyList = [$D_1$(Null), $D_2$($W_1$), $D_3$($W_1$), $D_4$($W_2$,$W_3$)] Equation 3.3

Line 11 of *GSA*: WaitingJobs = [$J_1$, $J_2$, $J_3$, $J_4$] Equation 3.4

Hence, equation 3.1 represents all the tasks, input ports, output ports, and data links in the GENOMEFLOW workflow GSL file. Equation 3.2 represents the subworkflows after partitioning the main workflow based on the dependency found between the tasks. The output of the partitioning step shows that subworkflow $W_1$ contains one task, subworkflows $W_2$ and $W_3$ contains two tasks each, and subworkflow $W_4$ contains one task. Equation 3.3 represents the dependency among the subworkflows found in the previous step. Subworkflow $W_1$ depends on nothing; however, $W_2$ and $W_3$ each depend on the output from $W_1$. $W_4$ depends on output from subworkflows $W_2$ and $W_3$. Equation 3.4 corresponds to each subworkflow listed in equation 3.2.

After partitioning the GENOEMFLOW workflow into subworkflows and preparing the Grid job script files, a list of available free nodes on the Grid will be gathered where its CPU usage is 0%. The job script will be submitted to the Grid to start the execution. For the above example (Figure 3.8), subworkflow $W_1$ will be executed first. Then subworkflows $W_2$ and $W_3$ will be executed in parallel after $W_1$ is finished. Lastly, $W_4$ will be executed.

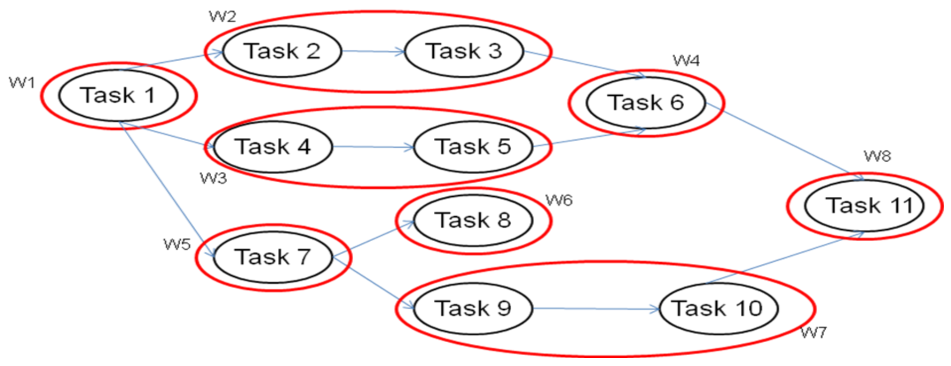


Figure 3.9: Example 2 to illustrate the scheduling and portioning algorithm.

Another example is showing in Figure 3.9:

WF = ({$Task_1$, $Task_2$, ..,$Task_6$}, …..)

WFList = [$W_1(Task_1)$, $W_2(Task_2, Task_4)$, $W_3(Task_3, Task_5)$,$W_4(Task_6)$,

$W_5(Task_7)$, $W_6(Task_8)$,$W_7(Task_9 , Task_{10})$, $W_8(Task_{11})$ ]

DependencyList = [$D_1(Null)$, $D_2(W_1)$, $D_3(W_1)$, $D_4(W_2,W_3)$, $D_5(W_1)$, $D_6(W_5)$,

$D_7(W_5)$, $D_8(W_4,W_7)$]

WaitingJobs = [$J_1$, $J_2$, $J_3$, $J_4$, $J_5$, $J_6$, $J_7$ , $J_8$]

## *3.5 System Execution and Evaluation*

This section demonstrates the implementation of a scientific workflow application in GENOMEFLOW to analyze Intragenomic Gene Conversions (IGC). Previously [6], a methodology to analyze the complete bacterial genomes for intragenomic recombination was identified. Intragenomic recombination, genes which share sequence similarity and therefore exchange or transfer genetic material through recombination, is analyzed by identifying groups of genes having sequence similarity in a genome and then analyzing each group for gene conversions.

Recombination is the transfer, insertion, or replacement of a length of DNA into a genome from another source. This can occur between two cells or different regions of the same cell. The usual requirement for recombination to occur is a similarity of sequence between recombining regions. Gene conversion, which is an outcome of a recombination event, is the non- reciprocal transfer of genetic information from one gene to another.

Recombination and gene conversion can occur between separate genes with related sequences within the same genome, a process known as Intragenomic Gene Conversion (IGC). IGC plays an important role in the evolution of multigene families of bacteria and the generation of antigenic variations [75]. Genome-wide analysis is the key to identifying sequences likely to have resulted from IGC events.

Despite the public availability of the microbial genome sequences and various sequence analysis tools, current IGC analysis relies on a manual and error-prone procedure. The procedures include downloading multiple datasets from public databases, integrating protein and genome data, modifying the format of the output of one analysis tool, and then transferring it into another analysis tool, and so on. IGC analysis and other genomic analysis procedures typically involve over 50 steps of human or computational tasks, and require an inordinate amount of time and labor for analysis.

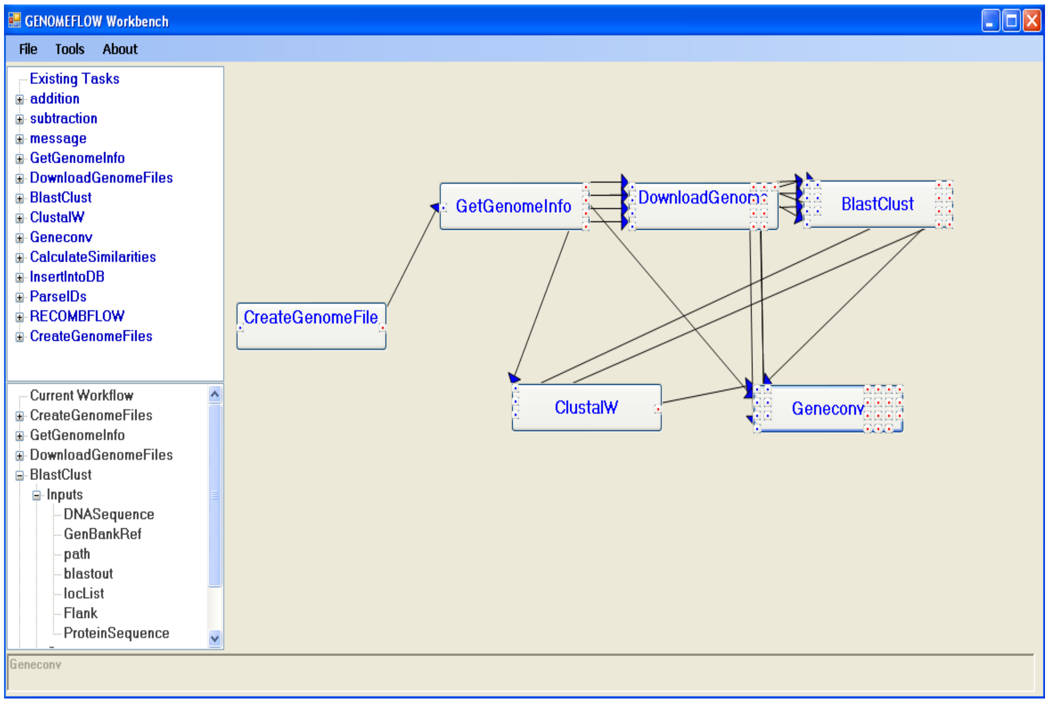


Figure 3.10: GENOMEFLOW scientific workflow workbench.

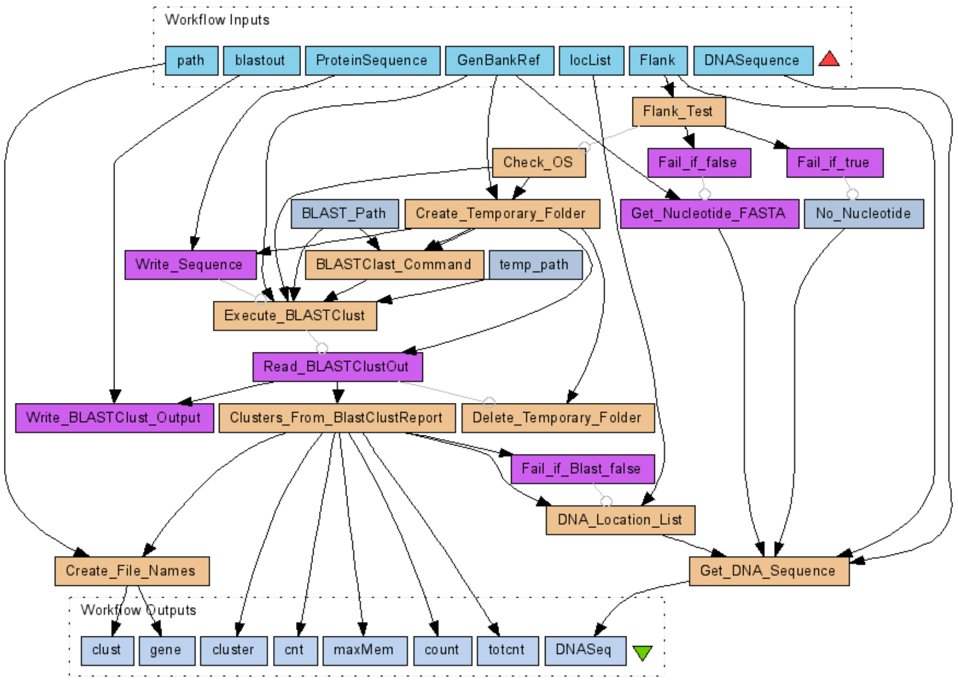


Figure 3.11: A Taverna workflow

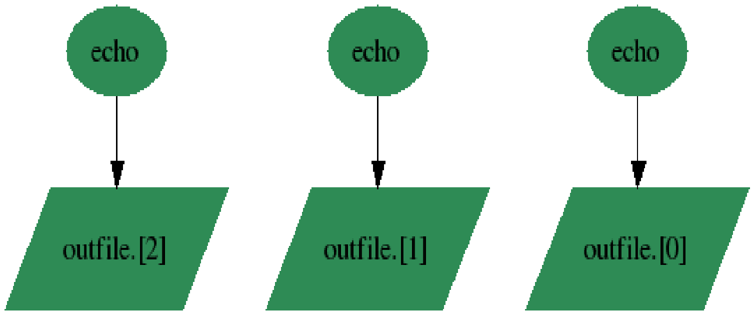


Figure 3.12: A Swift workflow

Taking into consideration the importance of IGC, a methodology to analyze the occurrence of IGC in bacterial genomes has been developed. This application performs genome wide analysis to identify gene conversions found among multigene family members of entire microbial genomes. To accomplish this task, complete genomes are retrieved from GenBank, and then BLASTClust, ClustalW, GENECONV, and various parsing and statistical computations are applied to the genomic data.

We thought of implementing the tool as a scientific workflow, considering the importance and usefulness of scientific workflow concepts and the important role it can have in the bioinformatics field where a large amount of data is available. Biological data processing involves several time consuming and error prone procedures such as downloading from several databases, copying and pasting from one web-based tool to another, annotating data manually, etc. So an effective way of automation is necessary to perform the complex scientific computations using the large amount of biological data and tools.

Figure 3.10 shows the scientific workflow that processes IGC. It consists of several heterogeneous tasks that were implemented by Swift and Taverna. Figure 3.11 shows a Taverna workflow and Figure 3.12 shows a Swift workflow. These tasks are reusable and can be used again when designing a scientific workflow application.

The experiments reported in this section utilized two environments: a frontend system and a backend system. Inputs are entered via a user interface implemented in C# on a windows PC with 2.8 GHz and 1 GB memory. The Windows client machine connects to the backend via a secure channel. The backend is the Wayne State University Grid with AMD and Intel machines. They have 8-16 GB RAM and 2-2.3 GHz operated by Linux machines. Once the user designs and executes the workflow, it passes the necessary data to the Grid for processing after applying the scheduling algorithm mentioned in section 3.4. After the execution is complete, the output data are transferred back to the user's machine.  Output files are organized by the workflow name and other related provenance data.

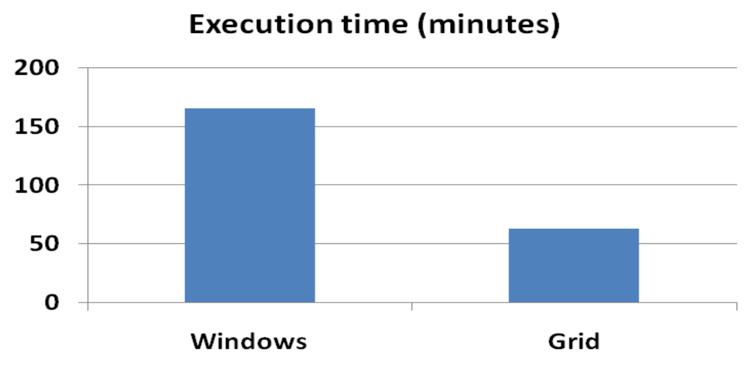


Figure 3.13: Execution time when using Taverna workbench to process IGC vs GENOMEFLOW.

Preliminary results show that the time it took to process 19 genomes individually for IGCs in Windows using the Taverna GUI system is about two hours and 46 minutes. Using GENOMEFLOW to process two genomes at the same time in the Grid environment resulted in processing all 19 genomes in one hour and three minutes (Figure 3.13).

## *3.6 Summary and Future Work*

Scientific workflows are emerging as an important technology for solving complex scientific problems and thereby contributing to scientific development. Many scientific discoveries are achieved through complex and distributed computations.  The advantages of high-performance Grid-based computing for scientific workflows, and the problems noted with existing scientific workflow management systems, provide the rationale for the development of our GENOMEFLOW system for efficient use of heterogeneous scientific workflow systems and utilizing Grid computing. When some functionality is missing in one workflow system, it might be available by another, so with a system that supports heterogeneous scientific workflows, the user can integrate the targeted scientific workflows. Also, scientific workflows can be shared among users. To illustrate, a user can implement a Taverna workflow and pass it to another user who doesn't know how to use Taverna but can add the workflow to his GENOMEFLOW workflow using a user friendly interface. By analyzing the results of our previous research [3] and [6], we realized the importance of a scientific workflow system that supports the execution of heterogeneous workflows using various services.

In this chapter, we presented a scientific workflow system that allows users to execute heterogeneous tasks where each task represent a third party SWFMS. We implemented an application to validate the feasibility of GENOMEFLOW. Ongoing work includes the extension of our system to support more scientific workflow systems and optimizing data movements between heterogeneous environments.

# CHAPTER 4 : SIMULATION OF GENOMIC RECOMBINATION AND DETECTABILITY OF RECOMBINATION

The detection of recombination from DNA sequences is relevant to the understanding of evolutionary and molecular genetics. While programs such as GENECONV have been identified as detecting recombination more reliably than others, previous studies have not analyzed how many recombinations they fail to detect. In this chapter, we will introduce a scientific workflow approach to simulate the genomic recombination and the detectability of recombination. The rest of the chapter is organized as follows: Section 4.1 presents the motivation and challenges for this research. Section 4.2 presents the DNA simulation model and how the initial population is created for the simulation. Section 4.3 introduces the DNA sequence data used in the simulation. Section 4.4 describes the GENECONV detection method. Section 4.5 presents the experimental parameters, and Section 4.6 introduces Preliminary GENECONV Analysis while using fixed recombination rate. Section 4.7 shows the experimental results while using variable recombination rate. Finally, Section 4.8 summarizes the chapter.

## *4.1 Motivation and challenges*

The genomic era has produced a plethora of DNA sequences that provide opportunities for computational biologists to discover meaningful information about biological processes. DNA sequences have information with great relevance to the evolution of organisms; to ongoing genetic processes that mediate antibiotic resistance, genetic diseases, and adaptation; and to bioengineering applications for both beneficial (e.g., pharmaceuticals) and nefarious (e.g., bioterror) purposes. Among the most important processes that may play a role in all of these phenomena is horizontal gene transfer, in which a new section of DNA sequence appears in the DNA sequence of an organism, often from exogenous sources. Similarly, new combinations of DNA sequence are produced by sex, which causes an intentional mixing (or “recombination”) of genes from both parents to create a novel daughter sequence. As will be described, computational analysis of DNA sequences can sometimes detect where such “exceptional” insertions of DNA sequences have occurred.

DNA recombination can be categorized into two kinds of processes: homologous and heterogeneous recombination. Homologous recombination occurs between two homologous DNA molecules and can itself be divided into two kinds: gene conversion (replacement), in which one DNA donates part of its genetic information to another DNA (Figure 4.1A), or crossing over (exchange), in which both parental DNAs exchange part of their genetic information (Figure 4.1B). Gene conversion includes donating larger or smaller pieces of DNA to create a daughter sequence that has portions of sequence from both parental sequences. Gene conversion can occur due to a double-crossing-over event, as occurs during recombination in normal meiosis. Heterogeneous recombination occurs where a completely unrelated sequence is inserted into a sequence from a non-homologous region of DNA. The present chapter focuses on the detection of gene conversion events.

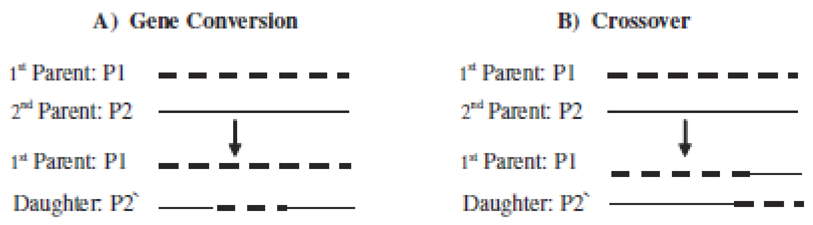


Figure 4.1: Different forms of Homologous Recombination.

The evidence for recombination as a potential mechanism for genetic change lead computational biologists to develop statistical methods to compare existing DNA sequences to identify evidence and locations of prior recombination. Methods work by identifying discontinuities in sequence similarities or genetic distance (e.g., GENECONV [76, 77] and MAXCHI [70, 80]) or by phylogenetic methods that, for example, identify incongruous tree topologies (e.g., RECPARS [41]). Many methods have been evaluated [69, 70, 97] and new ones continue to be developed. In comparisons of recombination analysis software, GENECONV was among the most highly ranked [70] and is included in the RDP2 analysis suite [60].

How well do recombination detection programs work? Since recombination detection programs detect putative recombinant fragments based on identifying significant differences in sequence, and recombination occurs at least as frequently between similar sequences as between divergent ones, how frequently do programs, such as GENECONV, fail to identify recombinations that are known to have occurred? This chapter tests the hypothesis that such programs fail to identify a significant number of recombination events and characterizes how the pairwise differences between the parental sequences affect the recombination detection success rate. Pairwise Difference

(PD) is the total number of base pairs different between the alleles of a particular gene of two individuals. The method described here involves simulating recombination, so that an explicit history of recombination in a set of DNA sequences is known, and then testing the resultant DNA sequences with GENECONV to see how frequently it failed to detect recombination events known to have occurred to produce the simulated sequences. As a result, this chapter identifies the occurrence of "cryptic recombination," i.e., recombinant events that are known to have occurred but were not identified by the recombination detection program.

## *4.2 Methods*

### 4.2.1 Simulation Model: Scientific Workflow Method

An allele is one of the variant forms of a gene sequence. Recombination rate (RR) is the probability that the alleles of two arbitrary individuals will recombine. In the first phase of our experiments, RR was chosen to have a fixed value ($RR_{fixed}$) that resulted in the number of alleles surviving after 1500 generations being close to the initial number of alleles at the start of the simulation. Variable rates of recombination are also considered. Our scientific workflow simulates recombination, replication, and selection through many generations. One of the key features of this algorithm is that a history of all of the recombination events that occurred in each generation is recorded.

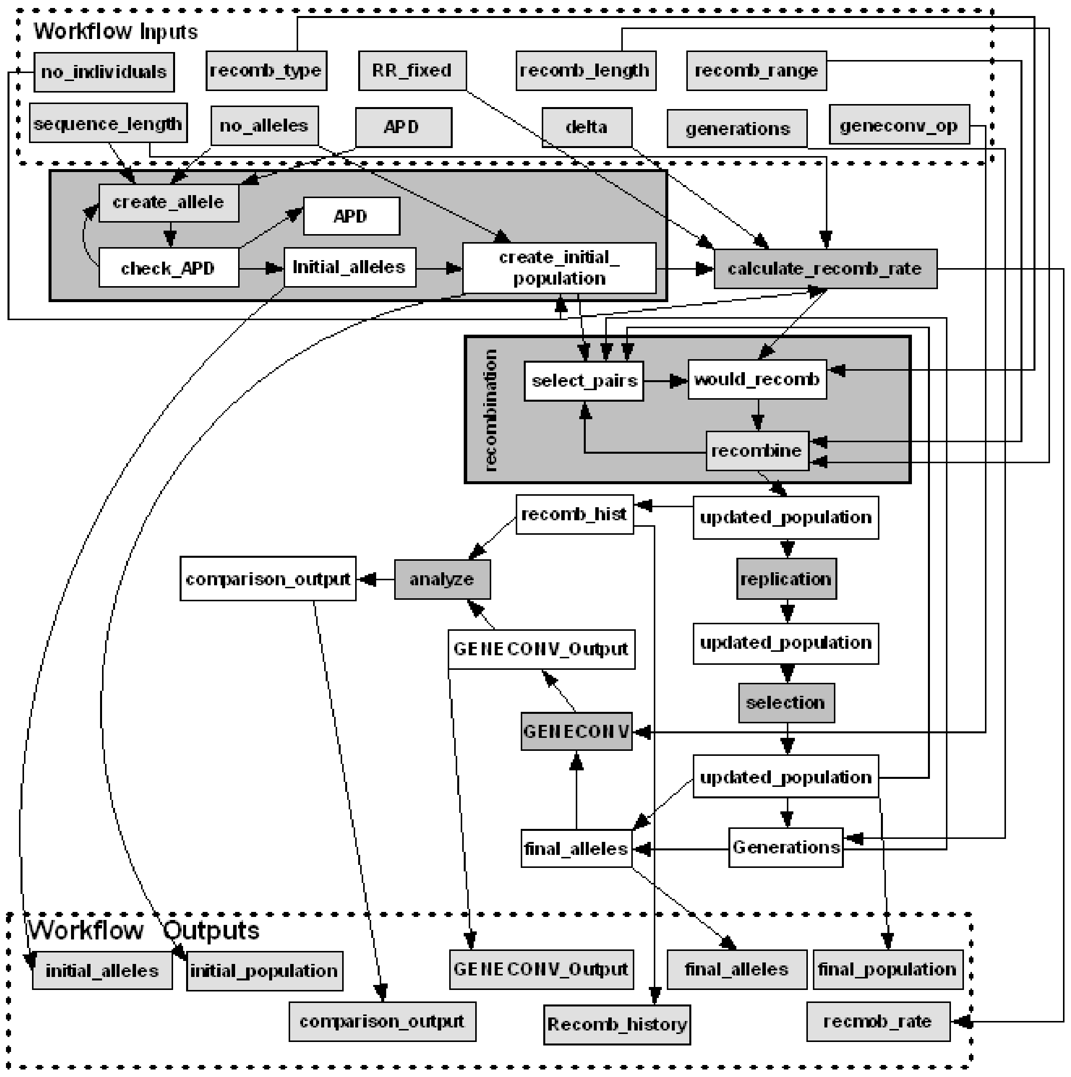


Figure 4.2: Processors, links, and workflow inputs and outputs of the recombination simulation scientific workflow.

Simulated populations of sequences were generated and analyzed by a scientific workflow (Figure 4.2) that employs a workflow system method in a grid environment and simulates recombination, replication, and selection through many generations. We used the scientific workflow method to automate the process, to rerun the analysis multiple times, and to follow the workflow execution. Figure 4.3 shows a GENOMEFLOW scientific workflow to simulate a population of sequences and analyze the detectability of recombination. Each task in the workflow is a scientific workflow implemented by Taverna. Figure 4.4 is a Taverna workflow that is integrated behind the first task of the GENOMEFLOW workflow showing in Figure 4.3. It simulates the DNA recombination and outputs a FASTA file representing all the surviving allele in the last generation, and a history of all the recombination events that occurred during the simulation. Theses outputs are passed to other tasks for further processing.

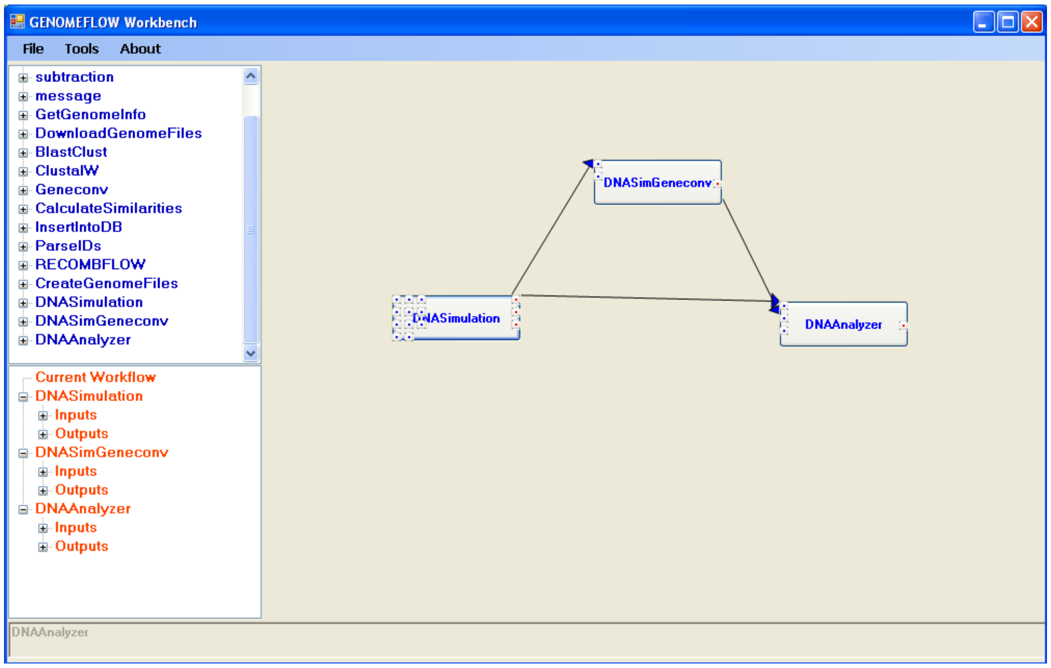


Figure 4.3: A simple GENOMEFLOW scientific workflow to simulate a population of sequences and analyze the detectability of recombination.

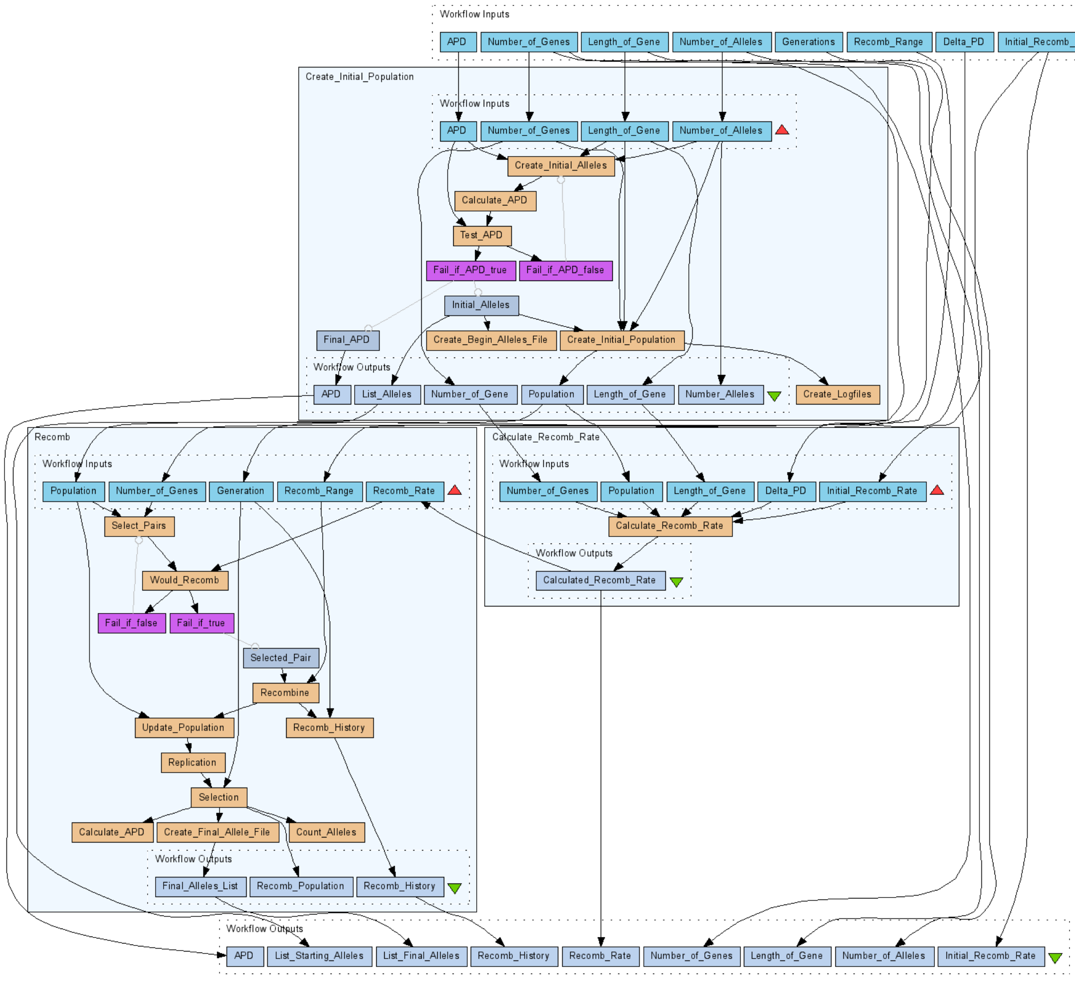


Figure 4.4: A Taverna representation for the DNA Simulation task.

For simplicity, the model does not include point mutations since we wanted to test the influence of pairwise difference on recombination on population structure and detection of recombination when recombination was the only mechanism causing changes in sequence. By initializing simulations with populations of any given pairwise difference, the method did not depend on a random mutation model to generate the diversity being analyzed. Another feature is that this model utilizes a neutral model of population regulation, without selection. As in a previous stochastic model of bacterial recombination [35], the use here of a neutral model is not a denial of selection, but rather an attempt at exploring the diversity of sequences in a null model, upon which selection can be subsequently imposed.

### 4.2.2 Creating the initial population

Creating the initial population consists of two steps. The first step is creating the initial alleles of the population as we describe below, and the second step is making *N/K* copies of each allele, where *N* is the size of the population and *K* is the number of alleles created.

**Creating the initial alleles.** An algorithm was developed to create initial alleles of sequences of any given average percentage pairwise difference (*APD*). The inputs for this process are desired *APD*, sequence length (*L*), and number of alleles (*K*) in the initial population. Each sequence consists of characters from a given character set (for genetic sequences: A, G, C, and T). The goal is to generate a uniformly distributed family containing *K* sequences, each of length *L*, having average number of differences *AD* between all sequences where $AD = APD * L$. The problem is non-trivial since each additional sequence added to the population must be considered with respect to its pairwise differences from every previously existing allele, a complex combinatorial challenge. The method uses a calculation of the estimated average number of changes expected at a particular position in a set of sequences that we derived as follows:

**Calculating the changes per column: Stepping across the rows.** Assume a given family of *K* sequences each of length *L* has the desired *APD*. After aligning all sequences, one above the other, the letters at each sequence position represent a column of values. From these *K* sequences, the average number of differences per column *(differences per column)* is calculated. The total number of sequence comparisons to calculate the *APD* of this family would be: *K(K-1)/2.* It follows that the total number of character comparisons will be *L*K(K-1)/2.* The total number of character differences found across the length of all the sequence comparisons will be

AD * K (K - 1) / 2 = total number of character differences. Equation 4.1

The total number of character differences can also be determined by finding the average number of character differences in each column, then multiplying by the *L* rows. That is *(differences per column) * L = total number of character differences*

This gives:

(differences per column) * L = AD * K (K - 1) / 2. Equation 4.2

Substituting AD = *APD * L* and dividing both sides by *L* gives:

(differences per column) = APD * K (K - 1) / 2. Equation 4.3

**Stepping down the columns.** Next, a worst-case scenario and best-case scenario for the average number of differences per column (differences per column) is calculated by looking at individual columns. Consider a column of length *K* and a base character chosen randomly from a given character set. *C* positions in the column are chosen to contain a random character from the character set that is not the same as the base character. The remaining *K-C* positions contain the base character. The range in the number of character differences in this column is estimated as follows:

**Case 1 – "Best" Case**: Suppose the same character is chosen for each of the *C* positions. The number of character differences in this column is:

(differences per column) = $(K - C)\, C$. Equation 4.4

**Case 2 – "Worst" Case**: Suppose a different character is chosen for each of the *C* positions, so that none of the *C* positions contain the same character. This column then contains *(K-C)C* differences between the base characters and the set of *C* characters. Within the set of *C* characters, since all characters are different, an additional *C(C-1)/2* differences are introduced:

(differences per column) = $(K - C)\, C + C\, (C - 1) / 2$. Equation 4.5

Equation (4.4) and (4.5) indicate the range of differences that results due to changes made to a column. Equation (4.4) represents the fewest number of differences that may result when changing *C* characters in a column, and equation (4.5) represents the largest number of differences from changing *C* characters in a column. Combining these equations with equation (4.3) gives the average number of difference per column for a sequence family with *K* sequences and a given *APD*, as follows:

$APD * K\, (K - 1) / 2 = (K - C)\, C$, Equation 4.6

and

$APD * K\, (K - 1) / 2 = (K - C)\, C + C\, (C - 1) / 2$, Equation 4.7

corresponding to equation (4.4) and (4.5), respectively, which can be rearranged into the following quadratics in terms of C, respectively:

$C^2 - (K * C) + (APD * (K^2 - K) / 2) = 0$ Equation 4.8

$C^2 + ((1 - 2 * K) * C) + (APD * K (K - 1)) = 0.$ Equation 4.9

Solving the quadratic equation for *C* gives bounds on the average number of changes per column. For each quadratic, only one solution for *C* fits the scenario of the problem. Averaging each of the proper solutions gives an approximate number of changes per column to use in the Creating Initial Population Process algorithm.

**Creating Initial Population Process.** Algorithm *CreateInitialPopulation*, given in Figure 4.5, is the first step in the simulation task. In line 10 of the algorithm, the loop starts with the $i^{th}$ character of each sequence. In line 11, the average number of changes per column based on equation (4.8) and (4.9) above is calculated. In line 12, a random base char (A, G, C or T, corresponding to the four DNA nucleotides) is selected from the character set. In line 13, the loop to generate the $i^{th}$ character of each sequence begins. In each iteration, a random number between one and the average number of changes per column is generated. If the random number is one, then the $i^{th}$ character of the current sequence is assigned randomly to a character from the character set. Otherwise, the $i^{th}$ character of the current sequence is set as the base character selected in line 12. Then the algorithm moves to the next sequence and generate its $i^{th}$ character. Once the $i^{th}$ character is generated in all of the *K* sequences, the program proceeds to the next $i^{th}$ character and repeats steps in lines 11 through 19. The above steps are repeated until we generate *L* characters for each sequence. Once all alleles are generated, the initial population consisting of *N/K* copies of each allele is generated.

The above model was used to generate populations with initial APD varying from 1% to 10%. Although bacterial "species" have APDs typically of just a few percent (e.g.,

~2% in *E. coli*; Alhiyafi et al. 2007), larger differences are considered because they are relevant to recombination between species and between sequences in gene families [6].

```
Algorithm CreateInitialPopulation
Input:
    APD: targeted average pairwise difference
    K: number of alleles
    N: number of individuals in the population
    L: length of an allele
Output:
    The initial population that contains N individual and K alleles
Begin
    For i = 1 to L do
        Average number of changes per column = K / ((Equation 4.8 + Equation 4.9) / 2)
        Select a random base character from the character set
        For each of the K sequences do
            Get a random number between 1 and the average number of changes per column
            If random number = 1
                Set $i^{th}$ char of the current sequence to a random character from the character set
            Else
                Set $i^{th}$ char of the current sequence to be the base character
            End If
        Next sequence
    Next i
Duplicate N/K copies of each allele
End Algorithm
```

Figure 4.5: Algorithm *CreateInitialPopulation*.

### 4.2.3 Recombination Rate Processes

Recombination rate (*RR*) is the probability that the alleles of two arbitrary individuals will recombine. The model described here varies RR as a function of pairwise differences between the potentially recombining alleles. Previously used constant recombination rates (i.e. not varying according to pairwise differences, as in Alhiyafi et al. [4] had been chosen to result in approximately the same number of alleles in the population after simulation of 1500 generations. The current study used a similar average recombination rate for the entire population of sequences, taking into account that more closely related sequences would recombine more frequently whereas more divergent pairs would have a lower rate. This enabled the comparison of populations that began with nearly identical initial average recombination rates, despite differences in individual pairwise rates.

The constant recombination rate to which the average recombination rate in the variable rate model is made equivalent in the current study is called the fixed recombination rate ($RR_{fixed}$). The effect of mismatches on recombination rate was incorporated into the calculations with the following equation, which yields an exponential decrease in recombination rate ($RR_{ik}$), dependent on the pairwise difference ($PD_{ik}$) between the potentially recombining sequences, i and k:

$$RR_{ik} = RR_0 * (10^{((\Delta - PD_{ik}) / \Delta)}) \qquad \text{Equation 4.10}$$

*Delta* ($\Delta$) determines the steepness of the relationship. Since empirical estimates of $\Delta$ have varied over a broad range, we tested values of $\Delta$ over a broad range (1.67 – 55). $RR_0$ is a constant determined prior to the simulation to adjust the *initial* average recombination rate for the population to be the same as simulations with a fixed recombination rate, $RR_{fixed}$, with which results are being compared. In that way, populations can start out with the same average recombination rate. $RR_0$ is calculated as:

$$RR_0 = RR_{fixed} / [(\sum_{i=1}^{N} \sum_{k=i+1}^{N} (10^{((\Delta - PD_{ik}) / \Delta)}))/(N(N - 1)/2)] \qquad \text{Equation 4.11}$$

$\Delta$ ranged from 1.67 (a very steep decrease in recombination rate with decreases in recombination rate of about 6 orders of magnitude for each 1% difference in sequence) to

55 (produces a 1000-fold decrease in recombination rate for each 16% increase in pairwise differences, similar to rates reviewed by Fraser et al. [35]). As previously noted, initial average pairwise differences in simulated populations varied up to 10%.

**Calculate recombination rate process.** Prior to running the simulation, recombination rates for all possible pairwise differences are calculated ("calculate_recomb_rate" process in Figure 4.2). During the multigenerational simulation, the "recombine" process uses these pre-calculated rates.

### 4.2.4 Simulation of multiple generations

**Recombination Process.** The third component of the workflow runs the main recombination process. This process is identical to Algorithm *Recombination* shown in Figure 4.6 except that in line 15, the recombination rate for the two potentially recombining sequences is calculated based on the pairwise difference between the two sequences (by lookup in the table generated by the calculate_recomb_rate process) rather than being equal to $RR_{fixed}$. Algorithm *Recombination* determines probabilistically for all possible pairs of sequences whether a recombination event has occurred and if it has, then employs a roulette wheel algorithm to determine where the starting position is located for recombining a fragment of length $L$ from "parent" one into the sequence of "parent" two to produce a new daughter sequence for the next generation.

After setting various parameters (initial number and diversity of alleles, sequence length, population size, recombination fragment size, recombination rate (see above regarding variation of recombinant rate according to pairwise differences), the model is run for a number of generations, $M$.

In *M* generations, recombination events, replication, and selection processes occur. Algorithm *Recombination* given in Figure 4.6 corresponds to the recombination process during each generation. In line 11, the first potential partner gene $g_i$ of the recombination is chosen and checked to see if it has not previously recombined in that generation. If gene $g_i$ is in the recombination set, then another $g_i$ is chosen; otherwise, in line 14, the second partner gene $g_j$ is randomly chosen from the partner set. In line 15, the recombination rate for these two genes is calculated. In the case illustrated, a fixed recombination rate $RR_{fixed}$ is used for all recombination events (an alternative program that uses variable recombination rates modifies this step). In line 16, a random floating point number between 0 and 1 is compared to the recombination rate, $RR_{fixed}$. If this random number is less than or equal to the recombination rate then both genes $g_i$ and $g_j$ will recombine. If gene $g_j$ does not recombine with $g_i$, the program jumps to line 23 where $g_j$ is removed from the partner set and the program returns to line 14 to choose another gene $g_j$ from the partner set.

```
01 Algorithm Recombination
02 Input:
03       population[N], N number of individuals
04       RL: recombination length
05       L: length of individual
06 Output:
07       population [N] after recombination
08 Begin
      //Record individuals that have engaged in a recombination process
09    RecombinationSet = $\phi$
10    For i = 0 to N do
11       If ($g_i \epsilon$ RecombinedSet) then continue;
12       PartnerSet = genes - $\{g_i\}$ - RecombinedSet //candidate partner set
13       While PartnerSet $\neq \phi$ do
14          choose a random gene $g_j$ from PartnetSet
15          $RR_{g_i,g_j} = RR_{fixed}$
16          If randf(0.0,1.0) $\leq RR_{g_i,g_j}$ then
17             startpos = randi(-RL, L)
18             endpos = min(L-1, startpos+RL)
19             Replace substring $g_j$(startpos, endpos] by $g_i$[startpos, endpos]
20             RecombinedSet = RecombinedSet $\cup \{g_i, g_j\}$
21             break
22          Else
23             PartnerSet = PartnerSet - $\{g_j\}$
24             continue;
25          End If
26       End While
27    End For
28 End Algorithm
```

Figure 4.6: Algorithm *Recombination*.

This iteration continues until a recombination has occurred or until all genes in the partner set have been given a chance to recombine. If recombination occurs between $g_i$ and $g_j$, the DNA fragment that is moved from one sequence to another for recombination has a length RL, and the following steps occur: in line 17 the starting position of the gene where recombination will happen is randomly chosen. The starting position is a random integer between -(*RL*) (recombination length) and (*L*) (length of the sequence).

In line 18, the end position of the recombinant fragment is calculated as min (*L-1*, *startpos*+*RL*). In line 19, the substring $g_j$ (*startpos*, *endpos*] is replaced by the substring $g_i$[*startpos*, *endpos*]. In line 20, both genes are added to the recombination set for that generation.

After all recombinations for one generation are determined probabilistically, the resultant population is replicated (process: **replication**), simulating cell division to a potential population size of *2 * N*. However, to keep population size constant, as would be the case for a stable biological population, *N* individuals are randomly chosen from the total *2 * N* population for the next generation (process: **selection**). By randomly choosing which individual sequences to eliminate, this process simulates neutral selection; however, this step could be modified in future versions to yield non-neutral selection.

One series of the processes of **recombination**, **replication**, and **selection** count as one generation and is repeated for *M* generations. A history of all recombination events is recorded for future comparisons and analysis.

Once the process of simulating DNA recombination is finished, the system produces a FASTA file containing the surviving alleles. This FASTA file is one of the inputs for the GENECONV detection program.

## 4.3 DNA Sequence Data

The choices of L and RL and the range of APD used for the simulations were based on biological data from *Escherichia coli*. For the gene for beta-glucuronidase, 525 bases (bases 331 to 855) were sequenced in 1323 strains of *E. coli* 148 alleles were identified, of which 76 occurred in the population at least twice. The APD was 10.5 bases, or approximately 2%. GENECONV identified significant recombination in the 76 multiply occurring alleles, identifying an average gene conversion fragment size of 243 bp. For the fimH gene, 531 bases (bases 80 to 610) were sequenced in 52 strains known to have different beta-glucuronidase alleles. 40 alleles were identified. The APD was 11.0 bases, or approximately 2.1%. GENECONV identified significant recombination among the 40 alleles, identifying an average gene conversion fragment size of 241 bp.

*E. coli* sequence data has also been analyzed with GENECONV for ectopic recombination, for which it is hypothesized that intragenomic recombination takes place between genes coding for similar proteins identified by BLASTCLUST [63]. We repeated the analysis to identify additional variables not previously reported. For 4 different *E. coli* genomes (U00096, AE014075, BA000007, AE005174), the average lengths of aligned DNA analyzed by GENECONV sequences, weighted according to the number of sequences in each cluster, were 965, 828, 871, and 919, respectively. The APD within each cluster, weighted according to the number of sequences in the clusters, averaged 13%, 14%, 10%, and 10%, respectively. The average lengths of gene conversion fragments identified by GENECONV in these clusters were 440, 198, 271, and 315, for an overall average of approximately 300 bp.

Populations simulated in this study had APD values ranging from 1% to 10%, which overlaps with the APD range actually observed and analyzed with GENECONV in the above biological experiments. Similarly, sequence lengths, L, were 531 or 1,000 bp, which overlaps with the biological data, and the models simulated gene conversion fragment lengths of 261 and 300 bp, also in the range of the biological data.

## *4.4 GENECONV Detection Method*

To determine the detectability of simulated recombination events, the FASTA file of DNA allele sequences present at generation 1500 of each simulation was submitted to GENECONV, which generated lists of pairs of sequences identified as having putative gene conversions. To estimate how many known recombination events were not detected (*cryptic recombination*) by GENECONV, the **analyze** process compares the record of simulated recombination events that generated the population being analyzed to the list of putative gene conversions identified as significant pairwise inner fragments by GENECONV. As in our previous study of simulated recombination with fixed recombination rates [4], this comparison was done for only the recombination products known to have been generated in the last ten generations of the simulation since multiple recombination events could have obscured the recombinant products of earlier generations. The effects of pairwise differences of the parent sequences and of Δ, the variable that determines the steepness of the relationship between pairwise differences and the recombination rate, on the detectability of the resultant gene conversion events by GENECONV were analyzed.

As in previous studies by Alhiyafi et al. [4], with constant recombination rates, the overall statistical significance of the pairwise inner fragments identified by GENECONV was determined by comparison to the number of pairwise inner fragments generated with the "randomize Sites" parameter turned on. Significance was determined from the Z value, where Z is calculated as follows:

Z = (avg. observed data – avg. randomized data) / (sqrt (avg. randomized data))

Values of Z greater than ~2.5 indicate that the number of gene conversion fragments identified in the submitted data is significantly greater than would have occurred by chance.

To identify cryptic recombination, the saved history of recombination events generated by the simulation was compared to the list of putative gene conversions identified by GENECONV. Since multiple recombination events could obscure the recombinant products of earlier generations, only the recombination products known to have been generated in the last ten generations of the simulation were compared to the GENECONV output list. The effects of pairwise differences of the parent sequences, the start position of the recombined fragment, the pairwise difference of the recombined fragments from the replaced fragments, and the pairwise difference of the non-transferred segments of the parent sequences were analyzed for their effect on the detectability of the resultant gene conversion events by GENECONV.

## *4.5 Experimental Parameters*

Two sets of parameters were used in the simulation. One set used length of sequence (L) equal to 531 bp with 263 bp for the recombination length (RL). The other set used 1000 bp for the length of sequence (L) with 300 bp for the recombination length. Each individual run of the simulation used 1500 generations (*M*), 1000 individuals in the population (*N*), $3.2 \times 10^{-6}$ as the fixed recombination rate ($RR_{fixed}$), and 40 alleles (*K*) at the beginning of the simulation. As explained above, delta ($\Delta$) in equation 4.10 determines the steepness of the relationship. The simulation was run with various delta values (1.67, 5, 10, 15, 20, 25, and 55) and various *APD* values (1%, 2%, 5%, and 10%).

## *4.6 GENECONV Analysis while using fixed recombination rate*

After analysis of the final set of alleles by GENECONV, Z values were always much larger than 2.5, indicating that the number of gene conversion fragments identified in the submitted data was significantly greater than would have occurred by chance. For example, the value of Z for 49 alleles resulting from running the simulation for 1500 generations with L = 1000 and RL = 300 bp, was 32.42. Another run of GENECONV with 44 alleles resulted in Z = 25.05.

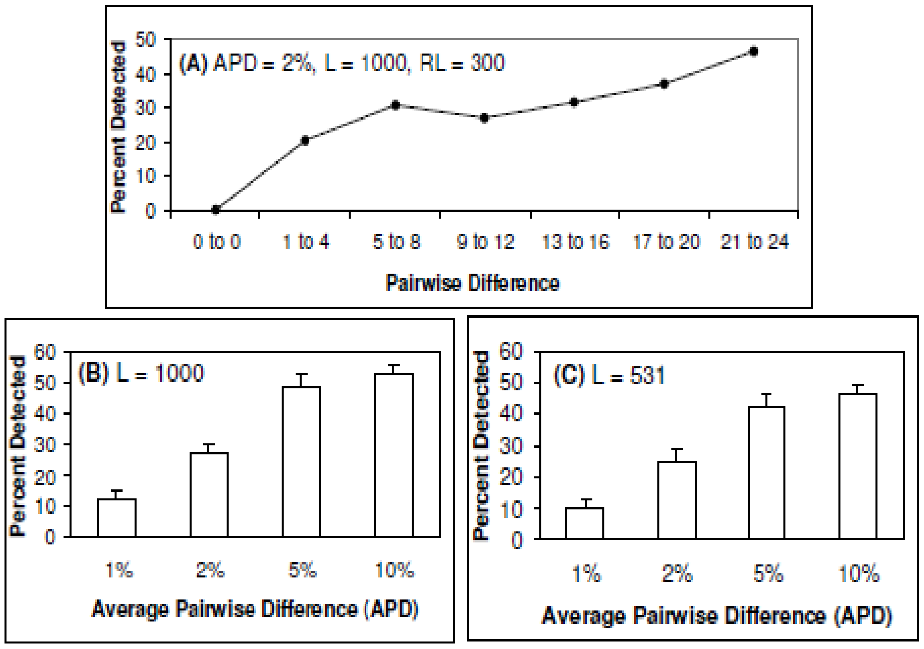


Figure 4.7: Effect of APD % on recombinations detected by GENECONV.

If GENECONV is ideally sensitive, then all recombinations known to have occurred in the last 10 generations should have been identified; however, the program often failed to identify the majority of them. For example, with 2% APD, in the last 10 generations of one run, 41 recombinations occurred and 49 alleles were given to GENECONV. GENECONV detected 257 recombinations, but only 11 of the 41 that actually occurred in the last 10 generations of the simulation process. In another example, where 5% APD was used and 52 alleles were submitted to GENECONV, 39 recombination events occurred in the last 10 generations, and GENECONV detected only 22 of them. Figure 4.7A shows that low parental PD reduces recombination detectability. For the last 10 generations of the simulation, the total number detected for various PD were divided by the total number of occurrences in the 10 runs. Figure 4.7B, C summarize how APD affects whether GENECONV detects known recombinations at different APDs. We used ANOVA in Figure 4.7B & C where $p < 0.001$.

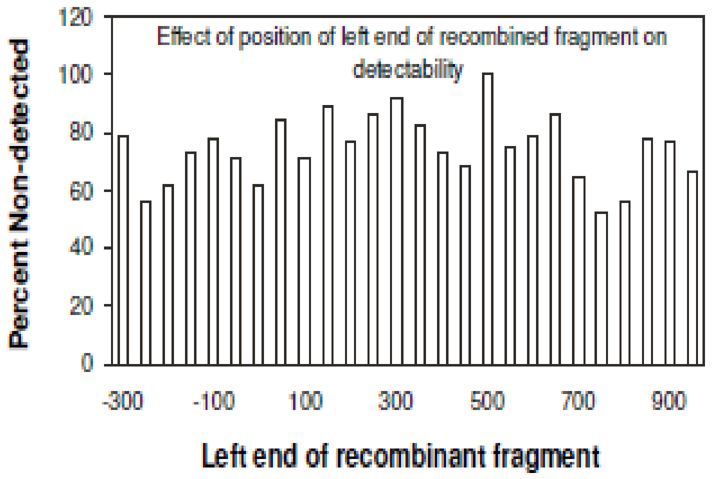


Figure 4.8: Lack of effect of position of recombined fragment on detectability.

The number of non-detected recombinations, i.e., “cryptic recombinations,” decreases as the APD increases and is a more significant problem when APD is low. Subsequent experiments analyzed whether the percent detected could be related to specific characteristics of the recombination event, such as the position of the recombined fragment and the pairwise difference present in the transferred or nontransferred fragments.

Figure 4.8 show that detectability was virtually the same regardless of the position of the recombined segment. The example shown is for sequences analyzed from simulations with initial APD of 2%. Populations with other APD values also failed to show any consistent differences with fragment position, showing the same differences of detectability at all positions as the overall detectability varied between different APD groups.

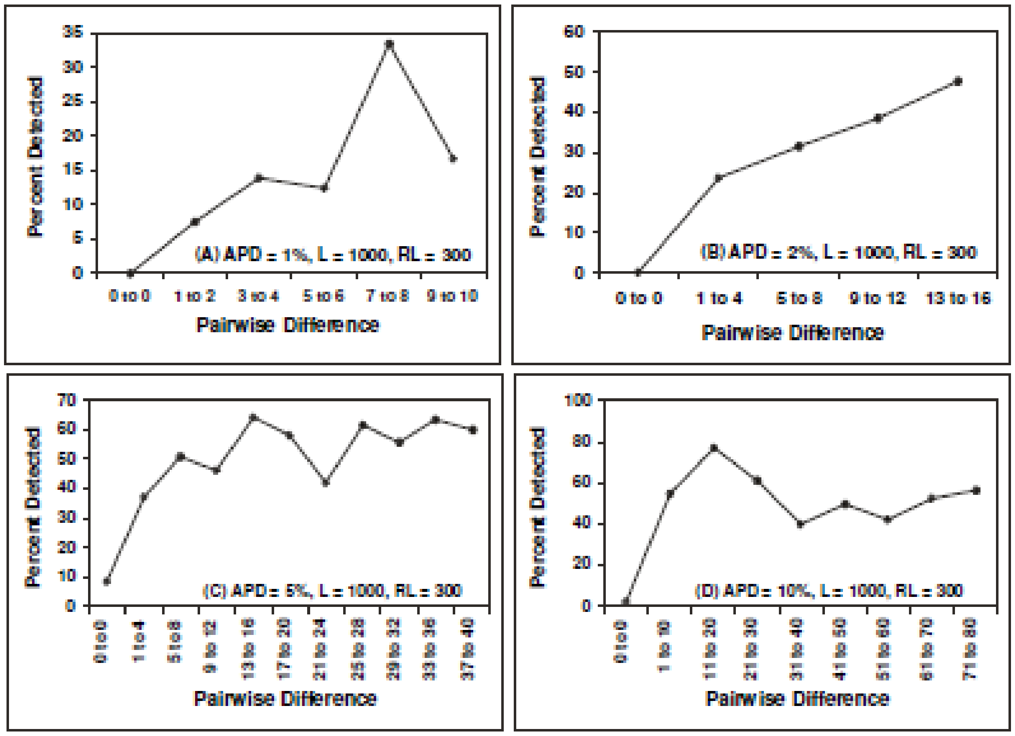


Figure 4.9: Effect of pairwise differences of the non-transferred segment of parental sequences on detection of recombinations by GENECONV.

Analysis of whether the pairwise difference of the recombined fragment affected its detectability also showed no difference across a wide range of values. However, as illustrated in Figure 4.9, the pairwise difference of the nontransferred segment has a significant influence on the detectability of the recombination event. For 10 runs of each initial APD %, the total number of recombinations detected in the recombination history of the last 10 generations was divided by the total number of recombinations in the same period for various pairwise differences of the non-transferred segments. At 1% and 2%

parental APD, the percent detected increased over the entire range of the pairwise differences of the non-transferred segments, and for 5% and 10% parental APD percent detected increased to a plateau of around 60% for pairwise differences of the non-transferred segment greater than approximately 2.5% of the non-transferred segment.

## *4.7 Experimental Results while using variable recombination rate*

### 4.7.1 Changes in population structure

Although all simulations began with 40 alleles, the final number of alleles after 1500 generations increases or decreases depending on the initial *APD*. For example, Table 4.1 shows the number of surviving alleles for various *APD* and Δ set to 15. With 1%, 2%, 5%, and 10% *APD*, the average number of alleles after 1500 generations was 29.8 $\underline{+}$ 1.7, 52.1 $\underline{+}$ 7.1, 4.8 $\underline{+}$ 0.6, and 2.9 $\underline{+}$ 0.3 (mean $\underline{+}$ sem, n = 12, 12, 6, and 9), respectively. Number of replicates varied from 6 to 12. Regardless of the initial *APD*, after 1500 generations, the *APD* of the final population decreased to about 50% of the initial population *APD* (Figure 4.10). The number of recombinations occurring in 1500 generations with the above mentioned parameters was approximately 4,750 and 14,500 recombination events when using fixed and variable recombination rates, respectively, an average of 3 and 9 recombinations/generation. The final set of alleles is the input for the GENECONV program to detect recombination events that will be compared to the recombination history recorded during the simulation process. In the last 10 generations that were compared with GENECONV output, 35-50 recombination events occurred when using a fixed recombination rate, and more than 135 recombination events occurred when using variable recombination rate. However, we discuss here first the effects of *APD* and Δ on population structure, and their effect on GENECONV performance in a later section.

Table 4.1: Number of surviving alleles after running the simulation.

| APD (%) | Fixed recombination rate | | Variable recombination rate | |
|---|---|---|---|---|
| | Initial Population | Final Population | Initial population | Final population |
| 1 | 40 | 30 | 40 | 29.8 |
| 2 | 40 | 34.2 | 40 | 52.1 |
| 5 | 40 | 42.3 | 40 | 4.8 |
| 10 | 40 | 37.8 | 40 | 2.9 |

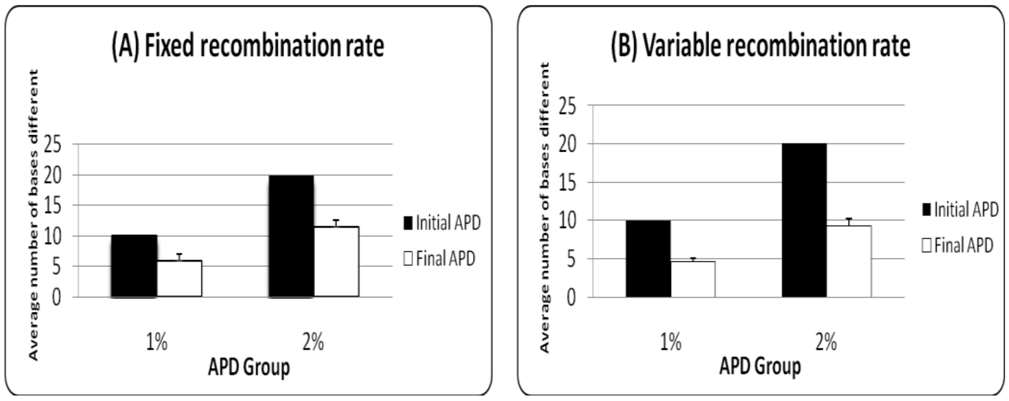


Figure 4.10: Decrease in PDs between initial and final populations.

Simulated populations appear to change in structure over many generations. Depending on the values of *APD* and Δ, populations sometimes separate into multiple groups or clusters some of which appear to become extinct with passing generations. This is illustrated in representative population history diagrams in Figure 4.11, in which the number of pairs with a given pairwise difference in each generation is quantified as different colors, over 1500 generations. Figure 4.11 show the representative changes in population structure over the course of 1500 generations for initial Average Pairwise Differences (APDs) of 1% and 2% and steepness of the recombination rate/pairwise difference relationship (Δ) equal to 15. The graphs illustrate the numbers of pairs with a various numbers of Pairwise Differences (PD) in each generation (up to 1500 generations), represented by color according to the $\log_{10}$(number of pairs having that difference). Zero representatives for a particular PD is plotted as Log(0.1) = –1 which is the darkest blue color.

In Figure 4.11A, recombination of an initial population with an initial average pairwise difference of 10 (i.e. 1% of the 1000 bp sequence) and Δ of 5 (a relatively steep decline in the probability of recombination with pairwise difference) almost immediately lost its most extreme pairwise differences and then slowly drifted towards populations with smaller pairwise distances. Lower Δ means a steeper fall-off of recombination rate with pairwise difference.

In Figure 4.11B and C, with initial average pairwise distances of 2% and 5%, respectively, and Δ of 10 and 20, respectively, populations similarly narrowed initially, but one now sees that various subpopulations appear and disappear over time. In both examples, sequences with smaller pairwise differences than the initial population arise due to recombination. In some cases, particularly in Figure 4.11C, a specific pairwise difference would appear in the population for some generations and then disappear, presumably because the **Selection** process that maintains a constant population size had by chance eliminated them. Hence, subgroups in the population become extinct.

“Extinction” also occurred in Figure 4.11B for many subgroups, including alleles with pairwise differences of 31 which briefly broadened around generation 600 only to become extinct before generation 800. On the other hand, it’s perhaps no surprise that with recombinations between near relatives favored, that a population with pairwise differences of less than 10 arose in Figure 4.11C and continued until the end of the experiment. Another notable feature of Figures Figure 4.11B and C is that by the end of 1500 generations the population seems to have broken up into several distinct subgroups, at least three in Figure 4.11C, and at least two distinct populations in Figure 4.11B (and perhaps three groups, depending on how one might interpret the smaller numbers (less reddish color) with pairwise differences between 5 and 10).

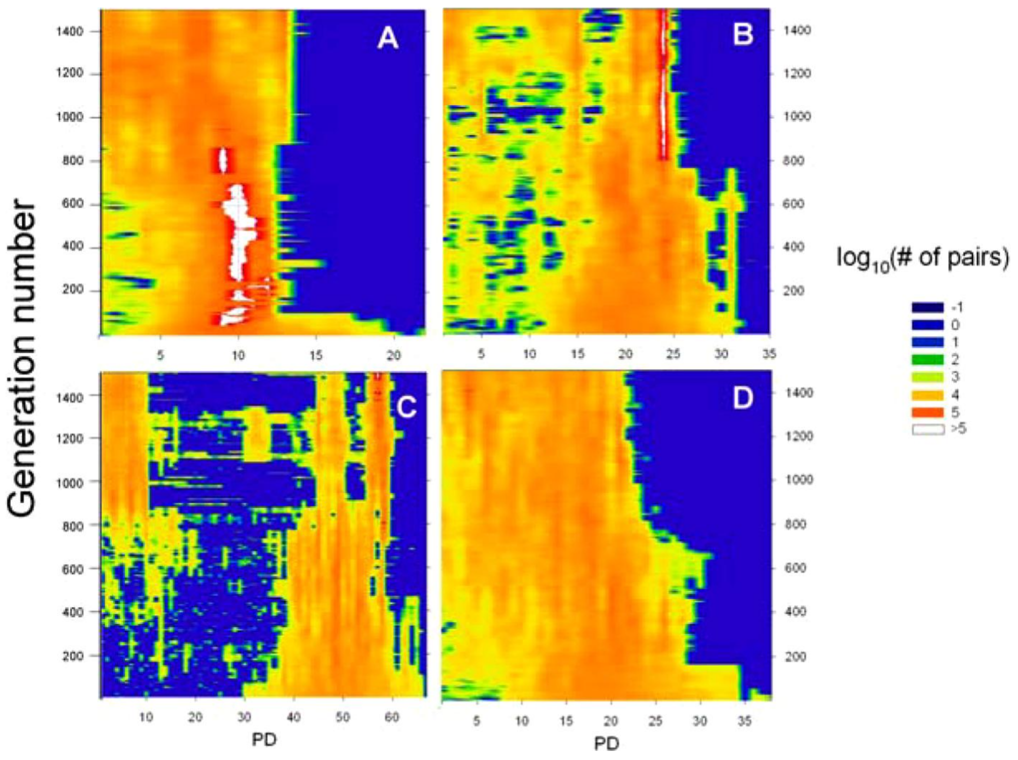


Figure 4.11: Representative changes in population structure over the course of 1500 generations.

In Figure 4.11D, with a value of Δ of 55, the largest considered in this study, a population starting with an initial average pairwise distance of 20 (i.e., 2%), the population history somewhat resembles that shown in Figure 4.11A, with much smaller Δ and APD. Relatively small numbers of the population with large PD disappear relatively quickly, and the average pairwise difference in the population drifts towards lower values as the experiment proceeds. A similar population history to Figure 4.11D occurs when the simulations are done with a constant recombination rate that does not vary with pairwise difference (data not shown).

The occurrence and relative stability of separate populations at the end of some of these 1500 generation simulations (e.g., Figure 4.11B and C) is a remarkably different outcome compared to a single population that slowly drifts towards lower average pairwise differences in other experiments (Figure 4.11A and D). The presence of multiple populations at generation 1500 was analyzed for a large number of simulations over a range of various values of initial APD and Δ. Multiple populations resulted from only a few sets of paramaeter combinations and never occurred with a constant recombination rate. More than 50% of the runs had multiple populations with APD = 5% and Δ = 20 or 25. Multiple populations at generation 1500 were also occasionally seen with APD = 1%, Δ = 5 and APD = 2%, Δ = 10. All other combinations of APD and Δ rarely (no more than once) or never resulted in multiple populations, as tested with 8 to 17 runs of each APD, Δ combination.

### 4.7.2 Detectability of Recombination

Sequences at generation 1500 were analyzed for evidence of recombination using GENECONV, and the identified related pairs were compared to the actual history of recombination recorded by the workflow system. The number of identified gene conversion fragments identified by GENECONV was always much greater than would have occurred by chance. For example, for the four representative populations illustrated in Figure 4.11A – D, the numbers of gene conversion fragments identified in the populations at generation 1500 were 32, 68, 243, and 295, for which values of Z were 11.7, 23.4, 59.1, and 67.3, respectively (values of Z > than 2.5 indicates significance at $p < 0.05$).

However, despite detecting a significant number of gene conversions, when the gene conversions detected by GENECONV were compared to those that had actually occurred in the simulation, it was apparent that GENECONV missed a large proportion of them. A typical result is illustrated in Figure 4.12, for populations that had been initiated with APD = 2%. Compared to the number of recombinations that actually occurred in the last 10 generations (generations 1491 to 1500) GENECONV detected only 8.1% $\pm$ 0.9% of them. The percent detected improved somewhat in simulations with higher values of $\Delta$, but was still only a little higher than 20% for $\Delta = 25$. In comparison, with an equivalent fixed recombination rate, 30.0% $\pm$ 1.6% were detected. Results followed similar trends for populations with other initial values of APD: as delta increased, the percent of recombinations detected increased. Bars in Figure 4.12 represent the percentage of allele pairs known to have had gene conversions in the last ten generations of the simulation that were also identified by GENECONV as having gene conversion fragments. Also shown for comparison, is the result for simulations with APD = 2% when the recombination rate had a fixed value.

A specific example, for the population illustrated in Figure 4.11B (2% *APD* and Δ = 10) is the following: At the end of 1500 generations, 25 alleles remained and were passed on to GENECONV. GENECONV detected 68 recombinations, but these included only 14 of the 255 gene conversions that actually occurred in the last 10 generations of the simulation process. The remainder of the 68 gene conversion fragments may represent recombinations that occurred in prior generations; however, this would still only account for a small fraction of the actual number of simulated recombinations, since in 1500 generations for this run, a total of 28,888 recombinations had occurred in the simulation, far more than the total detected by GENECONV.

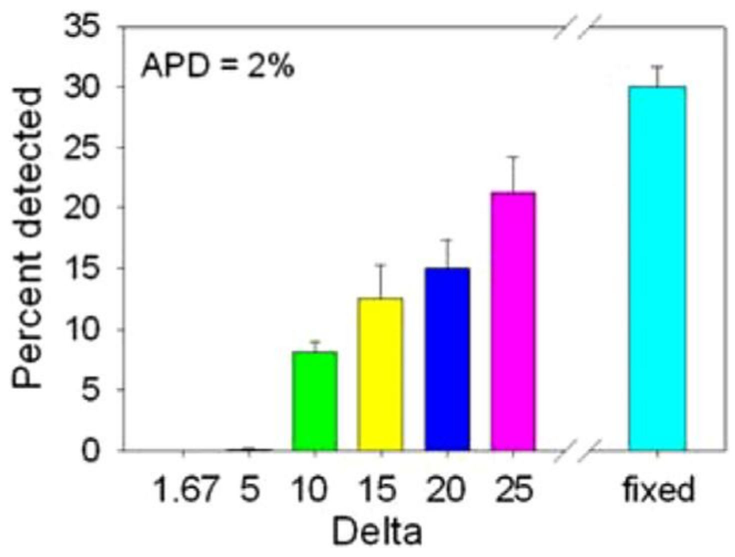


Figure 4.12: Effect of Δ on the percentage of simulated gene conversions detected by GENECONV for simulated populations with initial APD = 2%.

## *4.8 Summary and Future Work*

DNA recombination detection systems underestimate the amount of recombination since they are unable to detect the most frequent recombinations that occur between similar sequences. For DNA sequences with average pairwise differences of 1% to 2%, more than 70% of recombinations that are known to have occurred failed to be detected. These figures are based only on comparing sequences from the most recent 10 generations of simulation; however, even considering the entire 1500 generations, the total set of gene conversions identified by GENECONV is often much less than the 6,000 such events known to have been simulated in 1500 generations; viz., the representative example cited below Figure 4.7. Potentially, by knowing the APD of a population being analyzed, the number of cryptic recombinations can be estimated from the quantitative analysis provided here. These results would yield a higher recombination rate than previous analyses have suggested.

The causes of the low detectability of some recombinations does not seem to be related to the position of the recombined piece (Figure 4.8) nor the number of bases different in the transferred piece of DNA. Since the pairwise difference of the parents appears to affect detectability (Figure 4.9) but the pairwise difference of the transferred fragment does not, then a key variable mediating detectability may be the pairwise difference of the non-transferred segment, an observation supported by results illustrated in Figure 4.9. Future studies should further investigate the causes of non-detectability in order to reduce this problem or estimate its magnitude in future analyses.

The present study demonstrates that the population APD is a key variable in affecting recombination detectability. Pairwise differences could also affect quantitative estimates of recombination in another way: biological studies have determined that recombination is more likely to occur between similar sequences (i.e., those with a lower pairwise difference) than between more distant sequences [79, 92]. Thus, the recombinations that are hardest to detect, according to the present study, also occur most frequently. The stochastic simulation model developed in the present model is an ideal platform for analyzing the impact of pairwise differences on estimates of recombination rates. Preliminary results with simulations in which recombination rates decrease with increasing pairwise differences indicate that an even higher proportion of recombinations fail to be detected than is demonstrated here using a fixed rate of recombination.

This chapter also demonstrates that distance-modulated rates of recombination affect population structure and the detectability of recombination. The project also illustrates the use of a grid-based scientific workflow system for efficient simulation and analysis of the DNA recombination process and several useful algorithms in its implementation. The initial average pairwise differences in the population and the magnitude of the pairwise difference effect on recombination both affected whether populations remained unitary or broke up into distantly related subgroups.

The present model shows that changes in population structure akin to sympatric speciation can occur due simply to the effect of pairwise differences on recombination rate. Only certain combinations of pairwise differences and steepness of the pairwise difference:recombination rate relationship appear to favor such population clustering.

The value of Δ seems key. Empirical estimates of the effect of pairwise differences on recombination rate vary widely. The observations of Vulić et al. [93] suggest a decrease of 5 orders of magnitude for a 2% sequence divergence, which corresponds in our calculations to a value of Δ of approximately 4, while the rate effect reported by Fraser et al. [35] corresponds to our Δ = 55. Part of the explanation for this difference of the effect of sequence divergence on recombination rate is that the experiments of Vulić et al. [93] were carried out in strains in which the mismatch repair mechanism was defective, possibly exaggerating factors that affect recombination. Vulić et al. [93] have suggested that variations in the mismatch repair mechanisms may promote speciation. This suggestion is supported in the present study by the fact that multiple populations emerged here with Δ within this range of possible values. Moreover, since the present recombination analysis model is run without simulated point mutations or non-neutral selection, the clustering of subgroups produced under some conditions could be enhanced where selection and point mutations are allowed. Recent empirical data indicates that sympatric speciation can occur in nature [12, 16, 36]. This study indicates how the pairwise effect of sequence divergence on recombination rate may play a role in speciation.

Sequence divergence effects on recombination rate also exaggerated the difficulty of detecting gene conversion events. The present study shows that variation of recombination rates with pairwise differences needs to be taken into account when estimating recombination rates and when designing recombination detection experiments. GENECONV often failed to identify more than 80% of recombination events, an error rate that increased as the mismatch effect became larger (i.e., lower Δ). In order to make a more realistic estimate of the rate of recombination in any given population, it may be necessary not only to ascertain the average pairwise difference by sequencing portions of representative strains in the population but also to do experiments to determine the value of Δ in any given population. These results thus confirmed our hypothesis that the pairwise difference effect on recombination frequency decreases the efficacy of recombination detection programs in detecting recombination.

# CHAPTER 5 : INTRAGENOMIC GENE CONVERSION ANALYSIS

Intragenomic Gene Conversion (IGC) is important in the evolution of bacteria but has only been analyzed computationally in a few strains of *Escherichia coli*. This chapter describes a scientific workflow approach to analyze IGC in all NCBI bacterial genomes. We analyze for the first time the large variation of IGC in the pathogen *Streptococcus pyogenes*, and also in non-pathogenic bacteria. Also, Intragenomic Gene Conversion (IGC) has been suggested to mediate concerted evolution of genes. Previous studies on four *E. coli* strains suggested that IGC occurs more frequently in pathogenic strains. These hypotheses are investigated by (a) analyzing IGC in seven *E. coli* and six *Shigella* genomes, and (b) identifying specific bacterial genes in which IGC has occurred. The workflow system approach enables organizing large-scale computational analyses of multiple genomes and will facilitate future comparative studies of genome organization.

The rest of the chapter is organized as follows. Section 5.1 discusses the motivation and challenges. Section 5.2 presents an introduction on intragenomic gene conversions. Sections 5.3 introduce the intragenomic recombination analysis pipeline. Section 5.4 describes the implementation of the workflow. Section 5.5 presents the genome sequences used in this research. Section 5.6 presents the computational methods used. Section 5.7 displays the experimental results. Section 5.8 presents the biological results. Finally, Section 5.9 summarizes the chapter.

## 5.1 *Motivation and Challenges*

Recombination is the transfer, insertion, or replacement of a length of DNA into a genome from another source. This can occur between two cells or different regions of the same cell. The usual requirement for recombination to occur is a similarity of sequence between recombining regions. Gene conversion, which is an outcome of a recombination event, is the non-reciprocal transfer of genetic information from one gene to another. Recombination and gene conversion can occur between separate genes with related sequences within the same genome, a process known as *Intragenomic Gene Conversion* (*IGC*). IGC plays an important role in the evolution of multigene families of bacteria and the generation of antigenic variations [75]. Genome-wide analysis is the key to identifying sequences likely to have resulted from IGC events.

Despite the public availability of the microbial genome sequences and various sequence analysis tools, current IGC analysis relies on a manual and error-prone procedure. The procedures include downloading multiple datasets from public databases, integrating protein and genome data, modifying the format of the output of one analysis tool, and then transferring it into another analysis tool, and so on. IGC analysis and other genomic analysis procedures typically involve over 50 steps of human or computational tasks, and require an inordinate amount of time and labor for analysis.

Taking into consideration the importance of IGC, a methodology to analyze the occurrence of IGC in bacterial genomes has been developed. This application performs genome wide analysis to identify gene conversions found among multigene family members of entire microbial genomes. To accomplish this task, complete genomes are retrieved from GenBank, and then BLASTClust, ClustalW, GENECONV, and various parsing and statistical computations are applied to the genomic data. These procedures have previously been applied manually to four genomes of *Escherichia coli [63]*; however, the massive amount of work that is involved in testing the results with different control parameters and extending the analysis to all bacterial genomes necessitated automating the analysis by developing a scientific workflow system.

In this dissertation, a scientific workflow approach is demonstrated for analyzing IGC in complete microbial genomes. We have developed a system to automate this complex procedure, including protein and genomic data retrieval from the web and the invocation of various wrapped local protein and genome sequence analysis tools. We used the scientific workflow system to efficiently analyze the microbial genomes available from the National Center for Biotechnology Information (NCBI) via the internet.

## *5.2 Intragenomic Gene Conversions*

DNA consists of many genes and provides genetic information to regulate and reproduce cells. Information from each gene encodes a unique protein, which performs necessary tasks for the cell to function. Bacteria are very small single cell organisms that are found almost everywhere, in the air, water, soil, food, and human body. Bacteria are prokaryotes, which indicate that they contain a single cell that does not contain a nucleus. Instead, their genetic information is within a single circular chain of DNA. Even though they are small organisms, many live in groups and can multiply quickly by cell division, by which a single cell splits into two new daughter cells both with the same genetic material, and can conjugate (~ have sex) with each other to exchange and insert genetic material from one cell to another by homologous recombination.

Homologous recombination, which can occur between two similar DNA sequences, is the exchange or replacement (conversion) of genetic information in one DNA sequence by a homologous DNA sequence from the other sequence (Figure 5.1). This is an important factor for the survival and evolution of the cells. Although recombination is usually thought of as occurring between two different cells, intragenomic recombination, including IGC, can also occur in which recombination occurs between genes in gene families, i.e., genes with similar sequence that can be found in the same microbial genome. Where concerted evolution of related sequences occurs, e.g., the highly related multiple copies of ribosomal RNA genes, the similarity of the multiple copies may be maintained by intragenomic recombination [52]. Coevolution of the *tufA* and *tufB* genes of *Salmonella typhimurium* by IGC is supported by experimental analysis of recombination products [9, 46]. In addition, a bioinformatic approach has been applied by Morris and Drouin [63] to several strains of *Escherichia coli*, in which gene families were identified and then their members were compared for evidence of previous gene conversions using GENECONV [76], software that identifies non-random similarities between sequences as evidence for conversion events. In comparison with the K12 laboratory strain of *E. coli*, for which comparatively few intragenomic events were detected, numerous instances of IGC were identified in the genomes of three pathogenic strains of *E. coli* [63]. In order to understand the evolutionary role of IGC and also to analyze whether high IGC levels are especially characteristic of pathogenic bacteria, we used the complete set of bacterial genome sequences available in NCBI's GenBank to perform the analysis of IGC among members of identified multigene families.

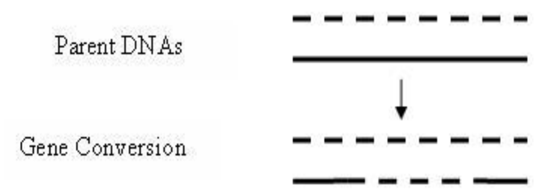


Figure 5.1: Gene Conversion. A Segment of one DNA sequence is replaced by a segment from the other sequence.

## *5.3 Intragenomic recombination analysis pipeline*

Analysis of Intragenomic Gene Conversion was begun in collaboration with M.S. student Cavitha Sabesan. She implemented the initial code of this workflow. I then added the Analysis of protein identifications (PIDs) of genes as a process to the workflow also another process to check similarities between genes. Modifications to the workflow were added to maintain the changes that were occurring in the NCBI website where we parse and download necessary data for the analysis. I also ran the workflow to analyze IGC in all current NCBI bacterial genome. The description here is similar and describes extensions to that collaborative work.

Figure 5.2 illustrates a pipelined view of the analysis of IGC as a set of processes and data flows among those processes. Except as noted, the processes and parameters follow the methodology applied previously to four *E. coli* genomes [63]. The protein sequence of a complete genome was obtained from GenBank and all multi-gene families of that genome were identified using the BLASTClust program [17]. BLASTClust identifies the multi-gene families and produces an output file that lists one family per line such that each line contains all members of that family. Each family member is indicated by the Protein Identifiers for genes, separated by spaces. Parameters were set so that two protein sequences need to be 60% identical over at least 50% of the area covering their length to be included in a family.

The corresponding DNA sequence for each member in the families was obtained and aligned using ClustalW [24]. These aligned DNA sequences were then processed using GENECONV [37] to identify gene conversions. Converted genes were identified as genes for which global inner fragments with $p<0.05$ were identified. G-scale, which is the mismatch penalty between different sequences and allows for point mutations to have occurred since the most recent recombination, was set to 2 to obtain more significant fragments. When GENECONV identifies the global inner fragments in a given sequence it lists each and every conversion identified in each pair of sequences. When considering a multi-gene family, this can result in a list of duplicate conversions if gene duplication had occurred after the recombination rather than before. In order not to count a single conversion multiple times, duplicate conversions were removed and counted only once, resulting in a conservative estimate of the number of gene conversions.

When considering the gene conversions identified by GENECONV, some converted fragments start at the beginning or end of the gene sequences being analyzed. This implies that if the analysis to detect gene conversions were to be carried out in genes with additional sequence flanking the region of interest, then the actual beginning or end of the fragment in the flanking region could be identified, rather than be limited by the ends of the sequences being considered. Also, by providing more sequence with which to identify gene conversions, converted sequences that previously were too short to be detected might be identified. Thus, the workflow pipeline incorporated methods of finding and adding on the flanking DNA sequences to all members of the gene families being considered prior to application of GENECONV. Since GENECONV also identified some conversions purely in the added flanking region and those conversions are fully outside of the gene clusters being compared, fragments completely contained within the flanking regions were eliminated in the final count. Size of the flanking region that was added to the DNA sequences prior to GENECONV analysis was an input parameter, which we generally set at 0 (“no flank” method) or 600 (“flank 600” method), which pilot experiments determined would reduce the number of conversions that started at the beginning or end of the sequences being analyzed to be less than 5% of the total number of identified gene conversions.

The final process in the workflow pipeline analyzed a number of statistical features regarding the conversions obtained, including numbers of multigene families, numbers of members in each family, maximum number of genes identified in those families, numbers of gene conversions and sizes of gene conversions.

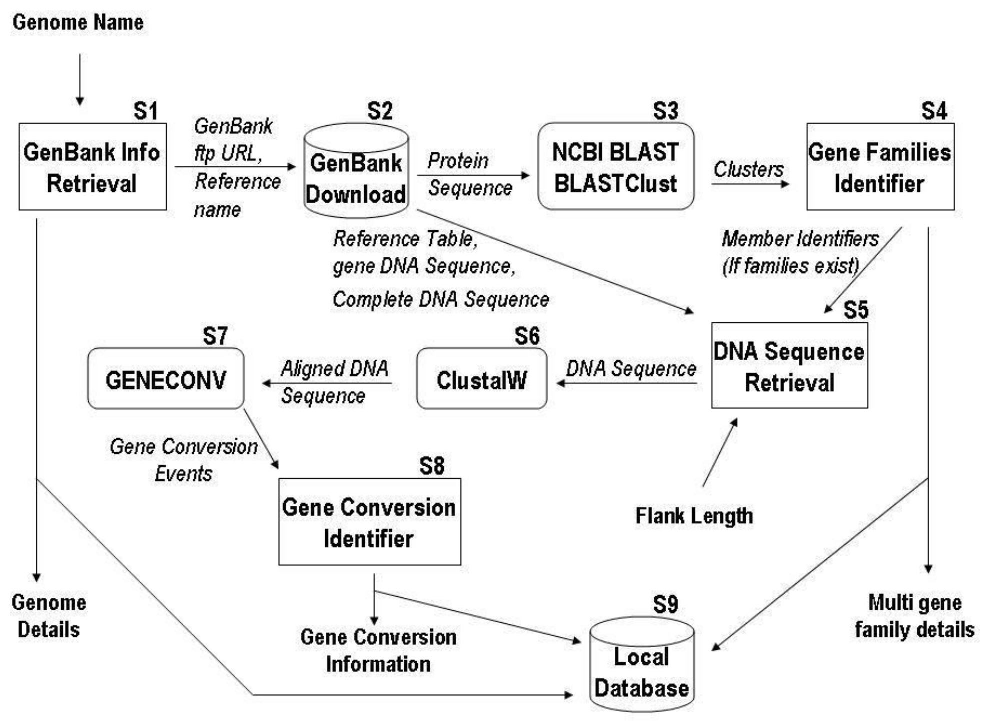


Figure 5.2: Pipelined view of the intragenomic recombination analysis.

## *5.4 Implementation of the workflow*

A scientific workflow system [6] was built to analyze IGC in complete bacterial genomes using both local and remote resources. The intragenomic recombination analysis workflow (Figure 5.3), can be considered not only a complete processor, but it can easily be shared, modified, and reused within other more comprehensive bioinformatics workflow systems. The various processors that are used in the workflow were implemented using Taverna scientific workflow system. Then, the workflow was implemented using GENOMEFLOW (Section 3.5) in order to incorporate the use of a high performance environment and to process more than one genome at the same time.

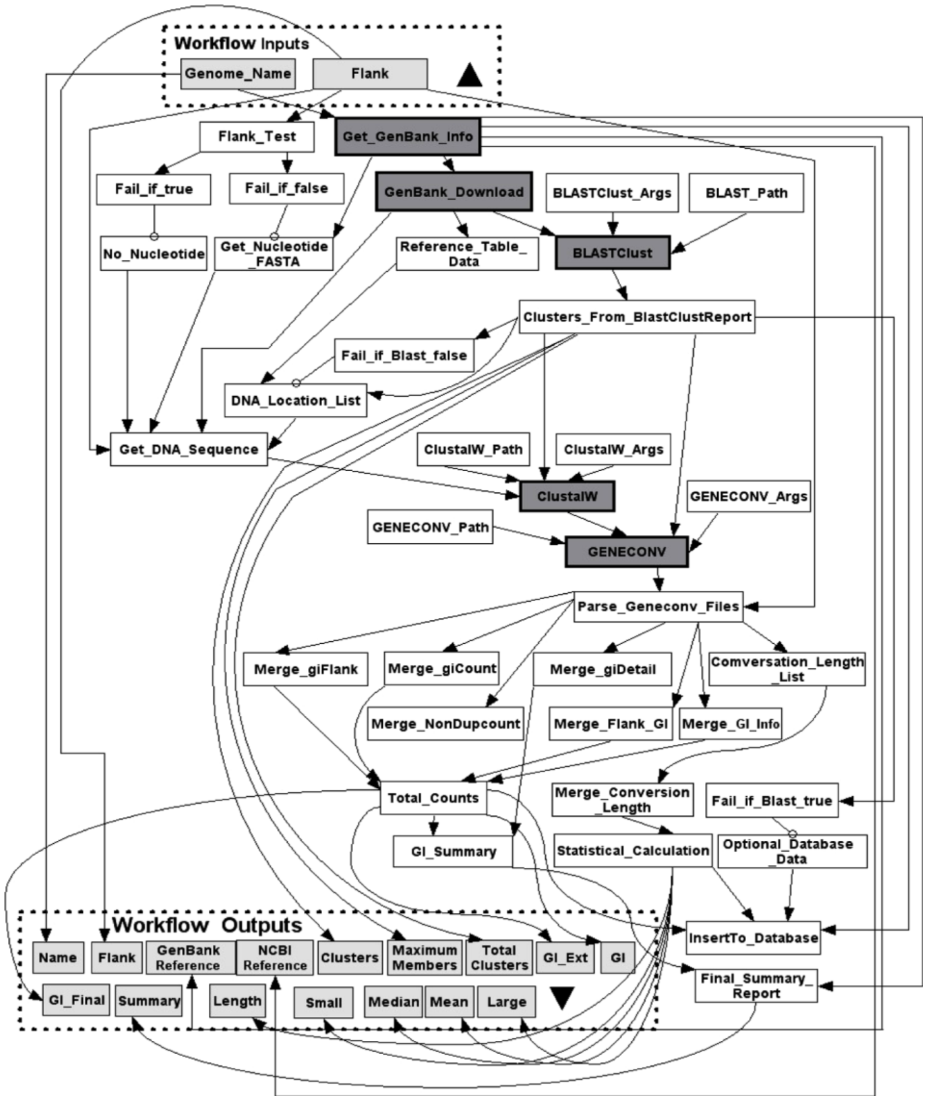


Figure 5.3: A scientific workflow for Intragenomic Recombination Analysis.

## 5.5 *Genome Sequences*

Complete bacterial genomes were obtained from the GenBank repository that is available from the NCBI ftp site (ftp://ftp.ncbi.nih.gov/genomes/Bacteria). The protein sequence (.faa), DNA sequence for proteins (.ffn), complete DNA sequence (.fna) and the reference table for proteins (.ptt) files were retrieved for each genome in the analysis.

The genomes analyzed in the present study were Escherichia_coli_K12_MG1655 (U00096), Escherichia_coli_CFT073 (AE014075), Escherichia_coli_O157H7_EDL933 (AE005174), Escherichia_coli_O157:H7_Sakai (BA000007), Escherichia_coli_536 (CP000247), Escherichia_coli_UTI89 (CP000243), Escherichia_coli_W3110 (AP009048), Shigella_boydii_Sb227 (CP000036), Shigella_dysenteriae_Sd197 (CP000034), Shigella_flexneri_2a_301 (AE005674), Shigella_flexneri_2a_2457T (AE014073), Shigella_flexneri_5_8401 (CP000266), Shigella_sonnei_Ss046 (CP000038). The labels in parentheses are the Genbank accession numbers of the core nucleotide sequence of each strain. The first four genomes are the same as studied by Morris and Drouin [63], except that their download was in 2002 and several updates of the sequences have been made in the intervening period (data in the present research are based on sequences downloaded in March, 2007). Strain 536 is an O6 serotype UPEC strain obtained from the Institut fur Hygiene und Mikrobiologie, Universitat Wurzburg, Germany [14] whose complete genome was reported by Hochhut et al. [42]. UTI89 is a strain provided by Langerman from a patient having acute bladder infection [66] and completely sequenced by Chen et al. [23]. Strain W3110 is an ancestral K12 strain whose complete genome was reported by Hayashi et al. [40]. *Shigella* strains Sb227, Sd197, and Ss046 were all isolated during epidemics in China in the 1950s and obtained from the Institute of Epidemiology and Microbiology, Chinese Academy of Preventive Medicine, for complete sequencing by Yang et al. [98]. *S. flexneri* variants are from epidemic strains that were sequenced by various authors: strain 301 was isolated from a patient with severe shigellosis in Beijing in 1984 [47]; strain 2457T 2a was obtained from the

Walter Reed Army Institute of Research and sequenced by the Blattner group [95]; and strain 8401 is from an epidemic in China and provided by the National Institute for Communicable Disease Control and Prevention, Chinese Centre for Disease Control and Prevention [47].

## *5.6 Computational methods*

Various software applications were incorporated as processes within the workflow system. The initial methods reproduced and verified the procedures of Morris and Drouin [63], which were then modified as described below. BLASTClust, used to identify multi-gene family members in the genome, was obtained from the NCBI ftp site (ftp://ftp.ncbi.nih.gov/blast). ClustalW, used to align sequences, was obtained from EBI (http://www.ebi.ac.uk/clustalw). GENECONV was obtained from Sawyer's web site (http://www.math.wustl.edu/~sawyer/geneconv) and used to identify gene conversion events between pairs of aligned DNA sequences. Duplicate gene pairs were removed from the GENECONV output with self-written software by a method described below. A graphical view of the IGC analysis scientific workflow is shown in Figure 5.3.

Processes and parameters used in the analysis were as follows:

Protein sequences within a genome were classified as members of multigene families with BLASTClust using 60% identity over at least 50% of their lengths as the criterion.

Sequences of family members were then aligned with ClustalW. Initially, Morris and Drouin's results were verified using their procedure of aligning the protein sequences (a procedure that inserts gaps when needed according to the amino acid sequence), followed by aligning the corresponding DNA sequences in the same register. However, since recombination takes place at the level of homologous DNA sequences rather than the protein sequence, it seemed more logical to align the DNA sequences of family members directly with ClustalW as this would more closely mimic the alignment presumed to occur during recombination. Therefore, after verification of previous results, subsequent alignments applied ClustalW directly to the DNA sequences corresponding to protein sequence family members identified as output from BLASTClust. Results will show that both methods produce comparable although not exactly identical values.

Aligned DNA sequences were processed using GENECONV to identify the gene conversions. Global inner fragments with p-values less than 0.05 were counted. The g-scale value, which sets the mismatch penalty for single nucleotide differences in the converted region, to allow for the possibility that point mutations had occurred after gene conversion, was set equal to 2, the same value used by Morris and Drouin [63].

With large clusters, pairs of global inner fragments identified by GENECONV often included duplicate fragments between sets of different pairs, which might have been due to a single gene conversion event followed by subsequent genomic duplications and divergence. In order to not count a single event multiple times, all but one representative of each set of duplicates were removed before counting up the gene conversion events. Since similar gene conversion events could have occurred multiple times, this duplicate removal procedure provides a conservative estimate of the number of Intragenomic Gene Conversion events.

In the course of running the above analyses and comparing to previous results, it became apparent that a significant number of gene conversion fragments actually started in the middle of the gene and ran all the way to the end. In other words, very likely the converted fragment was longer than the initial analysis revealed but the actual length was not detected because only the coding sequence was subjected to GENECONV analysis. To remedy this and to obtain more accurate estimates of the lengths of converted fragments, sequences of the DNA flanking each member of the families were appended to the sequences prior to alignment and subjecting sequences to GENECONV analysis. As will be described, various lengths of flanking DNA sequences were tested, and longer lengths of some previously very short fragments were revealed. In the count of conversion events in each cluster, we count only those fragments having at least one end beginning within the protein-coding region; additional conversion events that were wholly contained within the flanking sequences are not counted in the results presented, although these numbers are also available.

In summary, the computational methods initially duplicated that of Morris and Drouin [63] but were subsequently modified by applying ClustalW directly to DNA sequences of family members and by adding flanking DNA sequences to the DNA sequences prior to alignment and GENECONV analysis.

Subsequently, gene functions were identified with a new process added to the scientific workflow system to extract the annotations of genes in gene families from .ptt files downloaded from NCBI. Subsets of these families exhibiting IGCs were subsequently analyzed.

## 5.7 *Experimental Results*

### 5.7.1 Workflow performance

During the process, we recorded the start and end times for analyzing all the genomes. It took ~4.5 days to process >400 genomes, using both “no flank” and “flank 600” methods. It took about 1.8 days to process the genomes with the “no flank” method (median time per genome, 3.4 minutes) and about 2.7 days for genomes with the “flank 600” method (median time per genome, 4.6 minutes; significantly longer than “no flank”, $p<0.001$, signed rank test).

In addition to the effect of analysis method (flank v. no flank) on processing time, biological factors, such as genome size and numbers of gene families, also influence processing time. Figure 5.4 shows process time generally rising with genome size, for both “no flank” and “flank 600” methods. The relationships can be fit well ($r^2 = 0.735$, no flank; 0.534, flank 600; $p<0.001$ for both) with power equations. The processing time also increased with number of gene families identified in the genome (data not shown).

Process times varied greatly, even among genomes that are approximately the same size, and the longest processing times were not obtained with the largest genomes. Instead, the longest processing times were obtained from several genomes of intermediate size. Particularly notable in the graph for the “flank 600” output are 11 genomes that have processing times >50 min (circled in Figure 5.4, right). They include five different strains of *Shigella*, two of *Xanthomanas oryzae*, and several other moderately sized genomes. These genomes are among those in which the largest numbers of IGC were identified, and will be discussed in the section on biological aspects of the output. A similar group of points is present in the "no flank" graph of Figure 5.4.

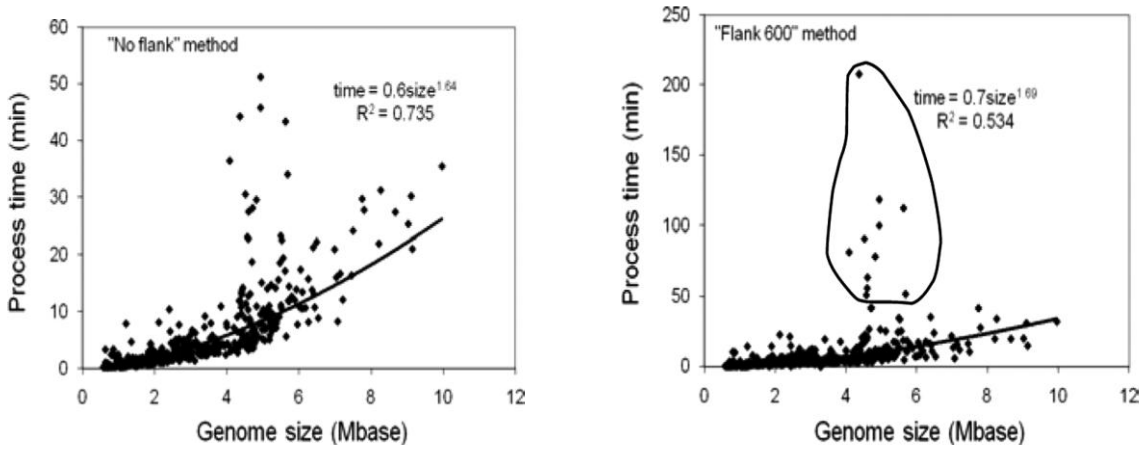


Figure 5.4: Processing time vs. genome size.

## 5.8 *Biological results*

### 5.8.1 Variation of IGC between and within species

The numbers of intragenomic gene conversions in ~430 bacterial genomes varied greatly both between species as well as within species. Analysis of the largest and smallest of the genomes and illustrate the species-to-species variation, even among strains that are similar in size to one another. The 15 largest genomes (Table 5.1), varying from about 7 megabases to nearly 10 megabases, ranged in numbers of intragenomic gene conversions from a low of 3 (no flank) or 10 (flank 600) conversions in *Pseudomonas fluorescens* Pf-5 to as many as 153 (no flank) or 449 (flank 600) conversions in *Trichodesmium erythraeum*. Similarly, among the smallest genomes analyzed, although most species show no intragenomic conversions (e.g., *Ureaplasma* *urealyticum*, AF222894, 751,719 bp; *Neorickettsia sennetsu*, CP000237, 859,006 bp), three of the smallest strains have large numbers of IGC (Aster yellows witches-broom phytoplasma AYWB, CP000061, 706,569 bp, 21 IGCs no flank, 113 IGCs with flank; *Mycoplasma mobile* 163K, AE017308, 777,079 bp, 19 IGCs no flank, 20 IGCs with flank; and *Mycoplasma pneumoniae*, U00089, 816,394 bp, 78 IGCs no flank, 203 IGCs with flank).

IGC also varies within species. This principle is demonstrated here to be a general property of bacterial genomes, by examining all groups of genomes for species for which >3 whole genomes have been reported (Table 5.2). The variation is particularly large among 8 strains of *E. coli*, ranging from 15 intragenomic gene conversions (no flank method) in the laboratory strain K12 to as many as 193, in the pathogenic strain O157:H7 Sakai, a 12.9-fold range, and is similar to the variation previously reported by Morris & Drouin, [63]for four *E. coli* strains. The data show here that *Rhodopseudomonas palustris* has a similarly large variation from a low number of 8 conversions (no flank method) in the BisB5 strain to as high as 102 conversions in the BisA53 strain, a 12.8-fold range. An even larger fold-range of intragenomic gene conversions, if not as large in absolute terms, is found in the pathogen *Streptococcus pyogenes*, for which strains with as few as 4 gene conversions (flank 600 method) to as many as 80 gene conversions have been sequenced. Because of the importance of *S. pyogenes* as a pathogen we look here at this species in greater detail.

Table 5.1: Intragenomic gene conversions among the 15 largest microbial genomes.

| Genome Name | GenBank Reference | Size | Gene Conversions | |
|---|---|---|---|---|
| | | | no flank | with flank |
| *Mesorhizobium loti* | BA000012 | 7,036,071 | 25 | 61 |
| *Pseudomonas fluorescens* Pf-5 | CP000076 | 7,074,893 | 3 | 10 |
| *Pirellula sp* | BX119912 | 7,145,576 | 10 | 81 |
| *Hahella chejuensis* KCTC 2396 | CP000155 | 7,215,267 | 22 | 51 |
| *Bradyrhizobium* ORS278 | CU234118 | 7,456,587 | 34 | 98 |
| *Frankia alni* ACN14a | CT573213 | 7,497,934 | 17 | 126 |
| *Trichodesmium erythraeum* IMS101 | CP000393 | 7,750,108 | 153 | 449 |
| *Rhodococcus* RHA1 | CP000431 | 7,804,765 | 55 | 118 |
| *Saccharopolyspora erythraea* NRRL 2338 | AM420293 | 8,212,805 | 30 | 138 |
| *Bradyrhizobium* BTAi1 | CP000494 | 8,264,687 | 50 | 142 |
| *Streptomyces coelicolor* | AL645882 | 8,667,507 | 21 | 43 |
| *Streptomyces avermitilis* | BA000030 | 9,025,608 | 29 | 62 |
| *Bradyrhizobium japonicum* | BA000040 | 9,105,828 | 53 | 164 |
| *Myxococcus xanthus* DK 1622 | CP000113 | 9,139,763 | 13 | 25 |
| *Solibacter usitatus* Ellin6076 | CP000473 | 9,965,640 | 26 | 184 |

### 5.8.2 Intragenomic recombination in *Streptococcus pyogenes*

*S. pyogenes*, also known as Group A Streptococci (GAS), are common human pathogens that cause various throat and skin infections, some of which may be life-threatening. Among the diseases caused by *S. pyogenes* are strep throat, scarlet fever, and rheumatic fever. Invasive GAS infections cause necrotizing fasciatis ("flesh-eating bacteria") and streptococcal toxic shock syndrome, which have mortality rates >20%. GAS are categorized into specific serotypes based on cell surface protein antigens, of which M protein is the most important (Table 5.3). The serotype, genome size, and the GenBank reference are listed for twelve strains of *S. pyogenes* whose sequences were available for Intragenomic Gene Conversion analysis.

Table 5.2: Variation of number of intragenomic gene conversions among species for which >3 genomes have been sequenced.

| Species name | genomes sequenced | Number of gene conversions (no flank, flank 600) | | |
|---|---|---|---|---|
| | | Least | Most | Median |
| *Chlamydophila pneumoniae* | 4 | 0, 0 | 3, 8 | 1.5, 3.5 |
| *Haemophilus influenzae* | 4 | 0, 0 | 3, 7 | 0, 0 |
| *Legionella pneumophila* | 4 | 2, 8 | 17, 23 | 4.5, 15.5 |
| *Mycobacterium tuberculosis* | 4 | 18, 70 | 32, 88 | 20.5, 82.5 |
| *Francisella tularensis* | 5 | 0, 3 | 4, 119 | 0, 34 |
| *Rhodopseudomonas palustris* | 5 | 8, 26 | 102, 280 | 40, 75 |
| *Yersinia pestis* | 6 | 5, 35 | 20, 101 | 12, 70 |
| *Escherichia coli* | 8 | 15, 38 | 193, 529 | 32, 60 |
| *Prochlorococcus marinus* | 10 | 0, 0 | 36, 219 | 4.5, 10.5 |
| *Staphylococcus aureus* | 10 | 7, 10 | 46, 69 | 31, 53 |
| *Streptococcus pyogenes* | 12 | 0, 4 | 51, 78 | 6.5, 13.5 |

The analysis of IGCs begins with identification of gene families having more than 2 members, which shows considerable variation among the 12 *S. pyogenes* genomes (Table 5.4). The largest family, with 13 members, occurs in the M18 strain MGAS8232. MGAS8232 and the M3 strains SSI-1 and MGAS315 have the largest number of families with more than two members, 15, 18, and 17, respectively. A family with 10 members occurs in the M6 strain MGAS10394. Both the M3 strains SSI-1 and MGAS315 contain 5 members in their largest family. Only a few multigene families are identified in other strains, and of those most have 5 or fewer members.

Table 5.3: *Streptococcus pyogenes* strains.

| Strain Name | GenBank Ref. | Serotype | Size (bp) | IGC fragment size (no flank, flank 600) | |
|---|---|---|---|---|---|
| | | | | Median | Maximum |
| MGAS5005 | CP000017 | M1 | 1,838,554 | 46, 808 | 46, 1342 |
| M1 GAS | AE004092 | M1 | 1,852,441 | 155.5, 155.5 | 813, 919 |
| MGAS10270 | CP000260 | M2 | 1,928,252 | 50, 322 | 50, 1429 |
| SSI-1 | BA000034 | M3 | 1,894,275 | 72, 88 | 497, 1499 |
| MGAS315 | AE014074 | M3 | 1,900,521 | 91, 109 | 1880, 2483 |
| MGAS10750 | CP000262 | M4 | 1,937,111 | 0, 862 | 0, 1813 |
| Manfredo | AM295007 | M5 | 1,841,271 | 47, 203.5 | 752, 2937 |
| MGAS10394 | CP000003 | M6 | 1,899,877 | 95, 146 | 593, 2111 |
| MGAS9429 | CP000259 | M12 | 1,836,467 | 44, 164 | 665, 1321 |
| MGAS2096 | CP000261 | M12 | 1,860,355 | 0, 254 | 0, 1152 |
| MGAS8232 | AE009949 | M18 | 1,895,017 | 90, 123.5 | 531, 1408 |
| MGAS6180 | CP000056 | M28 | 1,897,573 | 0, 797.5 | 0, 815 |
| | Averages | | 1,881,810 | 77, 336 | 647, 1602 |

The genomes with the largest number of gene families and family members also tended to have the higher numbers of IGCs (Figure 5.5). The highest number of IGCs was identified in strain MGAS8232, with 51 (no flank) or 78 (flank 600) conversions identified. Both the M3 strains SSI-1 and MGAS315 and the M6 strain MGAS10394 also have large numbers of conversions. All other strains have fewer than 13 IGCs. As noted earlier, addition of the flanking sequences enables the full length of converted fragments to be identified, as well as identifying fragments that were formerly too short to be statistically significant. Thus, not only are more converted fragments identified, but the sizes of the fragments identified are increased as well, as shown in Table 5.3.

Table 5.4: Number of members of gene families of *Streptococcus pyogenes*.

| Strain Name | Serotype | Number of members of indicated family size[1] | | | | | | | | | | | Total # of families of >2 members |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | |
| MGAS5005 | M1 | 2 | 1 | 1 | 1 | - | - | 1 | - | - | - | - | 6 |
| M1 GAS | M1 | 3 | 1 | - | - | - | - | - | - | - | - | - | 4 |
| MGAS10270 | M2 | 5 | - | 1 | - | - | - | - | - | - | - | - | 6 |
| SSI-1 | M3 | 11 | 5 | 2 | - | - | - | - | - | - | - | - | 18 |
| MGAS315 | M3 | 10 | 3 | 4 | - | - | - | - | - | - | - | - | 17 |
| MGAS10750 | M4 | 4 | 1 | 4 | 1 | - | - | - | - | - | - | - | 10 |
| Manfredo | M5 | 4 | 5 | - | - | - | - | - | - | - | - | - | 9 |
| MGAS10394 | M6 | 2 | 2 | 5 | - | - | - | - | 1 | - | - | - | 10 |
| MGAS9429 | M12 | 4 | 1 | - | - | - | 1 | - | - | - | - | - | 6 |
| MGAS2096 | M12 | 3 | 1 | 1 | - | 1 | - | - | - | - | - | - | 6 |
| MGAS8232 | M18 | 5 | 8 | 1 | - | - | - | - | - | - | - | 1 | 15 |
| MGAS6180 | M28 | 1 | 1 | 1 | - | - | - | - | - | - | - | - | 3 |

[1]a "-" means no gene families of the indicated size were found

### 5.8.3 Species with largest number of IGCs

A graph of genome size versus number IGCs (Figure 5.6) indicates a general upward trend of IGCs with genome size. 430 genomes were analyzed using the 'no flank' method (Figure 5.6A) and the 'flank 600' method (Figure 5.6B). Although there is a slight trend upward as genome size increases, the relationship is not significant. On Figure 5.6B, 17 points for which the number of IGCs equal to or greater than 370 appear in a 'cloud' of points above the main distribution are circled and described in more detail in Table 5.5. Above the main trend lies a distribution of strains of intermediate genome size that have much higher numbers of conversions than most others.

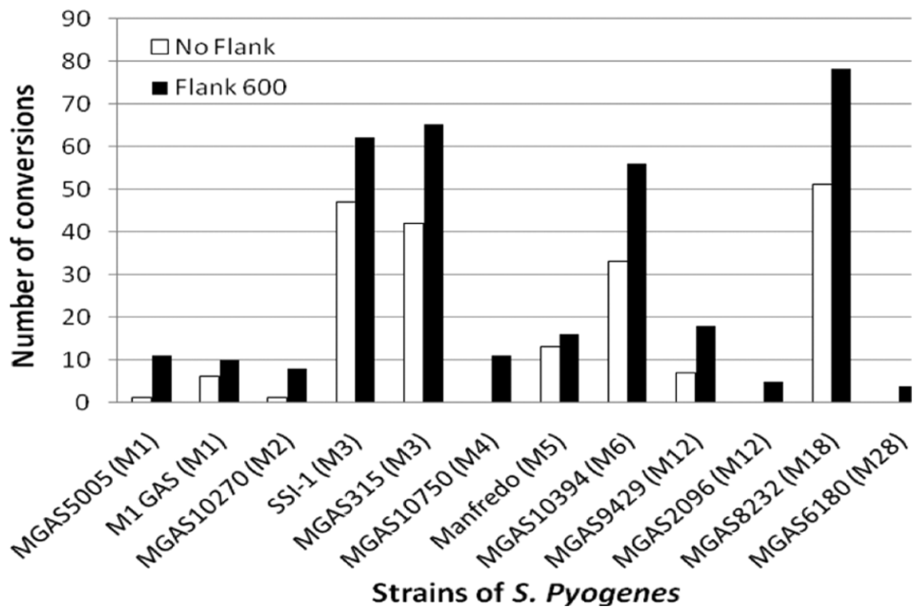


Figure 5.5: Number of IGCs found in *Streptococcus pyogene*.

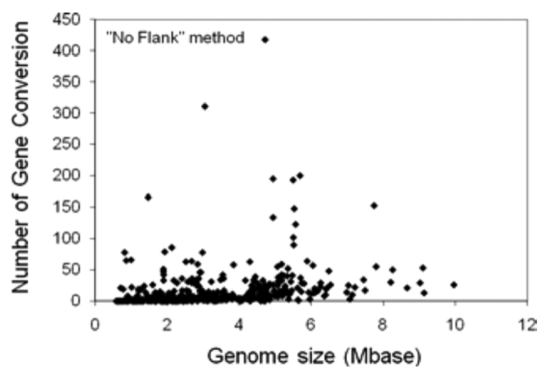
Table 5.5 lists the 17 genomes with largest number of IGCs (circled in Figure 5.6B). Seven of them are pathogenic variants of *E. coli*, including 2 strains of *E. coli* O157:H7 (AE005174 and BA000007) and 5 of *Shigella* (and AE014073, CP000266, AE005674, CP000036, CP000034), which are considered derivatives of *E. coli* [72]. Two strains from *Xanthomonas oryzae* are on the list, as well as two strains of cyanobacteria from a Yellowstone hotspring. *Xanthomonas oryzae* causes bacterial blight of rice [45]; whereas, the cyanobacteria are species of *Synechococcus* and are not known pathogens [7]. The rest are a miscellaneous collection of pathogenic and non-pathogenic bacteria.

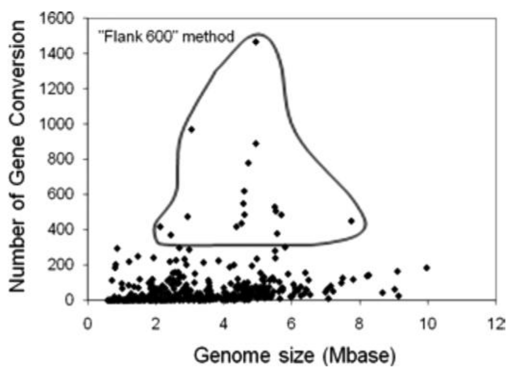
Table 5.5: Details of large number of conversions identified

| Genome Name | GenBank Ref | Size (Kb) | Fam >2* | Max mem/fam* | Number of IGCs | |
|---|---|---|---|---|---|---|
| | | | | | no flank | flank 600 |
| *Xanthomonas oryzae* MAFF 311018 | AP008229 | 4.9 | 39 | 76 | 195 | 1466 |
| *Cyanobacteria bacterium* Y B-Prime | CP000240 | 3 | 12 | 38 | 311 | 969 |
| *Xanthomonas oryzae* KACC10331 | AE013598 | 4.9 | 44 | 108 | 134 | 889 |
| *Magnetococcus* MC-1 | CP000471 | 4.7 | 94 | 20 | 418 | 779 |
| *Shigella flexneri* 2a 2457T | AE014073 | 4.6 | 36 | 104 | 15 | 619 |
| *Shigella flexneri* 5 8401 | CP000266 | 4.6 | 36 | 107 | 24 | 548 |
| *Escherichia coli* O157H7 | BA000007 | 5.5 | 90 | 18 | 193 | 529 |
| *Escherichia coli* O157H7 EDL933 | AE005174 | 5.5 | 93 | 21 | 148 | 505 |
| *Shigella flexneri* 2a | AE005674 | 4.6 | 37 | 109 | 21 | 485 |
| *Photorhabdus luminescens* | BX470251 | 5.7 | 84 | 30 | 200 | 484 |
| *Cyanobacteria bacterium* Y A-Prime | CP000239 | 2.9 | 12 | 20 | 46 | 474 |
| *Trichodesmium erythraeum* IMS101 | CP000393 | 7.8 | 69 | 27 | 153 | 449 |
| *Shigella boydii* Sb227 | CP000036 | 4.5 | 25 | 170 | 28 | 436 |
| *Shigella dysenteriae* | CP000034 | 4.4 | 20 | 314 | 14 | 417 |
| *Orientia tsutsugamushi* Boryong | AM494475 | 2.1 | 38 | 27 | 86 | 417 |
| *Verminephrobacter eiseniae* EF01-2 | CP000542 | 5.6 | 35 | 22 | 123 | 378 |
| *Lactococcus lactis cremoris* SK11 | CP000425 | 2.4 | 12 | 50 | 0 | 370 |

*"Fam>2" is number of families with >2 members. "Max mem/fam" is the number of members in the largest family.

The origins of the large number of IGCs in this group of bacteria are probably varied. Some have very large numbers of gene families (e.g., *E. coli* O157:H7 and *Magnetococcus*), while others have fewer families but exceptionally large numbers of members in those families (e.g., various strains of *Shigella* and *Xanthomonas oryzae*).





(A) (B)

Figure 5.6: Number of IGCs vs. genome size.

### 5.8.4 Number of Gene Families, Genes, and Gene Conversions

As shown in Table 5.6, analysis of EDL933, Sakai, CFT073, and K12 (MG1655) strains of *E. coli* with BLASTClust, followed by ClustalW alignment of amino acid sequences, conversion to corresponding DNA sequences, GENECONV analysis, and elimination of duplicates gave results similar to Morris and Drouin [63]. Thus, the EDL933 and Sakai O157:H7 strains had more multigene families (297 and 241, respectively, identical to Morris and Drouin [63]) than CFT073 (195 gene families in our analysis; 196 in Morris and Drouin's), which in turn was more than the MG1655 strain (107 here, versus 104 for Morris and Drouin). Comparably, the current analysis confirmed that families with only two members were greater in the O157:H7 (204 and 151) and CFT073 (142) pathogenic strains than in the non-pathogenic MG1655 strain (81). Multigene families with more than two members were, respectively, 93, 90, 53, and 26. Inspection of Table 5.6 confirms that other intermediate values in the analysis are similar, if not always identical to Morris and Drouin [63]. Finally, the important result, with respect to gene conversions, is that multigene families with more than two members in the EDL933, Sakai, CFT073, and MG1655 genomes contained 142, 231, 52, and 16 gene conversions, respectively. This result confirmed the higher rates of intragenomic gene conversions, as determined by GENECONV in this set of pathogens, compared to this particular non-pathogenic strain.

Before continuing the analysis with other *E. coli* genomes, we made the adjustment, as explained in Section 5.4 and 5.5, of doing the ClustalW alignments on the DNA sequences corresponding to the members of the clusters, instead of aligning the amino acid sequences before converting to DNA sequences. Results with this modification (see top 4 lines of data in Table 5.7) are close to, but not identical, to the original method (Table 5.6). For example, with the revised method the numbers of gene conversions contained in multigene families with more than two members in the EDL933, Sakai, CFT073, and MG1655 genomes was 148, 193, 41, and 17, respectively. The important result of higher numbers of gene conversions for the O157:H7 strains than the UPEC strain, and in turn more than the non-pathogenic strain is confirmed. It's also clear that the changed method produces results that can be either slightly higher or lower than the original method (i.e., no systematic bias is introduced).

Next, the method was applied to genomes of additional *E. coli* strains that had become available since 2002 and also to genomes of *Shigella* spp. The results are summarized in lines 5 through 13 of Table 5.7.

Table 5.6: Verification of method by comparison to Morris & Drouin (M&D), 2004.

| Strain name | Genbank accession | Genome size (bp) | Multi gene families | | | | | | | | Gene conversions | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Total number | | 2 members | | >2 members | | max. members | | Total number | | Smallest (bp) | | Largest (bp) | |
| | | | M&D | This research | M&D | This research | M&D | This research | M&D | This research | M&D | This research | M&D | This research | M&D | This research |
| Escherichia_coli_K12_MG1655 | U00096 | 4,639,675 | 104 | 107 | 81 | 81 | 23 | 26 | 11 | 11 | 17 | 16 | 12 | 7 | 3422 | 3422 |
| Escherichia_coli_O157H7_EDL933 | AE005174 | 5,528,445 | 297 | 297 | 204 | 204 | 93 | 93 | 21 | 21 | 150 | 142 | 4 | 4 | 1479 | 1479 |
| Escherichia_coli_O157H7(Sakai) | BA000007 | 5,498,450 | 241 | 241 | 151 | 151 | 90 | 90 | 18 | 18 | 230 | 231 | 9 | 8 | 3704 | 3704 |
| Escherichia_coli_UTI_CFT073 | AE014075 | 5,231,428 | 196 | 195 | 143 | 142 | 53 | 53 | 19 | 19 | 52 | 52 | 6 | 6 | 917 | 917 |

First, another K12 strain, W3110, has been sequenced. The number of gene conversions in gene families with two or more members in W3110 is 15, compared to 17 for the MG1655 strain, confirming the relatively low number of gene conversions in these non-pathogens.

Second, genomes from several additional UPEC strains have been sequenced. Gene conversions in gene families with two or more members in strains 536 and UTI89 were both 24, which is closer to the values (~16) for the K12 strains and only ~60% of the value of 41 observed for the previously studied UPEC strain, CFT073.

Finally, analysis of *Shigella* genomes revealed only relatively low numbers of intragenomic gene conversions, all in the range of the newer UPEC genomes and the K12 strains. For example, only 14 gene conversions were detected by GENECONV in the genome of *Shigella dysenteriae*, the lowest value seen yet in an *Escherichia*-related genome. Values of other *Shigella* species and strains were 28, 21, 15, 24, and 24 (see Table 5.7 for identification of strains).

Table 5.7: Gene conversion numbers and sizes in *E. coli* and *Shigella* spp.

| Strain name | Genbank accession | Genome size (bp) | Multi gene families | | | | Gene conversions, flank = 0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Total | 2 mem | >2 mem | Max. mem | Total number | Smallest (bp) | Largest (bp) | Median (bp) | Mean (bp) |
| Escherichia_coli_K12_MG1655 | U00096 | 4,639,675 | 107 | 81 | 26 | 11 | 17 | 10 | 3422 | 45 | 439 |
| Escherichia_coli_K12_W3110 | AP009048 | 4,646,332 | 107 | 83 | 24 | 18 | 15 | 10 | 3422 | 58 | 495 |
| Escherichia_coli_O157H7_EDL933 | AE005174 | 5,528,445 | 297 | 204 | 93 | 21 | 148 | 4 | 1491 | 172 | 275 |
| Escherichia_coli_O157H7(Sakai) | BA000007 | 5,498,450 | 241 | 151 | 90 | 18 | 193 | 11 | 3704 | 181 | 310 |
| Escherichia_coli_UTI_536 | CP000247 | 4,938,920 | 133 | 103 | 30 | 12 | 24 | 12 | 393 | 66[a] | 107 |
| Escherichia_coli_UTI89 | CP000243 | 5,065,741 | 154 | 124 | 30 | 7 | 24 | 10 | 902 | 82[a] | 133 |
| Escherichia_coli_UTI_CFT073 | AE014075 | 5,231,428 | 195 | 142 | 53 | 19 | 41 | 6 | 917 | 156 | 198 |
| Shigella_Boydii_Sb227 | CP000036 | 4,519,823 | 79 | 54 | 25 | 170 | 28 | 9 | 2773 | 552[b] | 630 |
| Shigella_Dysenteriae | CP000034 | 4,369,232 | 62 | 42 | 20 | 314 | 14 | 11 | 1052 | 190 | 319 |
| Shigella_Flexneri_2a_301 | AE005674 | 4,607,203 | 124 | 87 | 37 | 109 | 21 | 10 | 885 | 246 | 277 |
| Shigella_Flexneri_2a_2457T | AE014073 | 4,599,354 | 95 | 59 | 36 | 104 | 15 | 23 | 1183 | 175 | 272 |
| Shigella_Flexneri_5_8401 | CP000266 | 4,574,284 | 115 | 79 | 36 | 107 | 24 | 6 | 1354 | 222 | 345 |
| Shigella_Sonnei_Ss046 | CP000038 | 4,825,265 | 103 | 68 | 35 | 172 | 24 | 7 | 1846 | 269 | 395 |

[a,b]Values marked with the same superscripts are not significantly different from each other, but do differ significantly from values marked with a different superscript (Kruskal-Wallis One Way Analysis of Variance on Ranks, $p<0.001$; pairwise comparisons by Dunn's method, $p < 0.05$). No other pairwise comparisons are significant.

As noted in the first paragraph of this section, the *E. coli* strains with the most gene conversions were also the ones with the most gene families. Therefore, one could suppose that the numbers of gene conversions in *Shigella* might simply be a function of the numbers of gene families. Indeed, there is a significant positive correlation of numbers of gene families with numbers of gene clusters if all 13 genomes are analyzed as a group (Figure 5.7, $r^2 = 0.895$; $p < 0.001$). However, this significant relationship is strictly a result of the extreme values due to the EDL933, Sakai, CFT073 strains. Leaving out those three strains still leaves 10 strains with values for both independent and dependent variables varying over almost a two-fold range and showing no significant correlation at all (Figure 5.7, $r^2 = 0.098$; $p = 0.378$). In Figure 5.7, the two squares are values for O157:H7 strains and the circle is for the UTI *E. coli* strain CFT073. The dashed line shows the linear regression for all points, for which r2=0.985, while the solid line is the non-significant linear regression for all points excluding the O157:H7 and CFT073 strains.

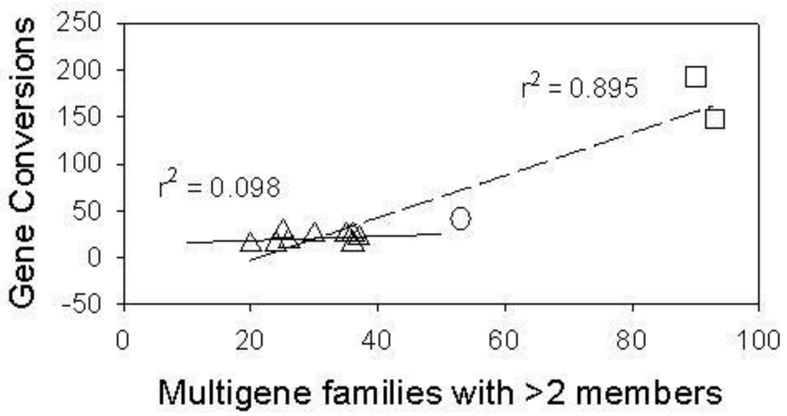


Figure 5.7: Relationship between number of IGCs identified in various *E. coli* and *Shigella* strains.

Another factor that might be expected to account for the low numbers of gene conversions in *Shigella* is the number of members of their multigene families. Indeed, for the four genomes studied by Morris and Drouin, a significant positive correlation was found between the size of a multigene family and the number of conversions [63]. According to this reasoning, *Shigella* would be expected to have small gene families; however, this is clearly not the case with *Shigella*. A remarkable difference of all of the *Shigella* species from those of *E. coli* is the consistently larger numbers of members of the multigene families in *Shigella.* As illustrated in Figure 5.8, the largest gene families in *Shigella* species ranged from 104 to 314 members; whereas, the largest multigene family in the *E. coli* genomes was 21 in EDL933. Moreover, even though Figure 5.8 highlights only the largest families, it is not only the largest family that stands out in these genomes. For example, the ten largest gene families of *Shigella boydii* Sb227 have 170, 125, 41 (3 families), 40, 34, 29, and 24 (2 families) members. The average *Shigella* genome had 7 ± 1 (mean ± sem) families with >21 members in them; whereas, none of the *Escherichia* strains had families with >21 members.

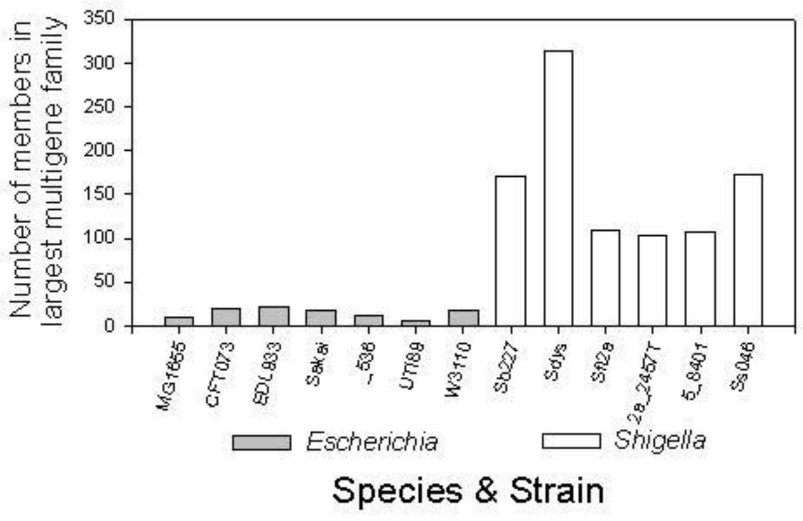


Figure 5.8: Number of members in largest multigene families in *E. coli* and *Shigella* genomes.

### 5.8.5 Lengths of gene conversion sequences

The sizes of the gene conversions detected by the above methods (Table 5.6 and Table **5.7**) for EDL933, Sakai, CFT073, and K12 (MG1655) strains of *E. coli* were practically identical to that reported by Morris and Drouin [63]. A particular gene family that will be discussed in the Discussion is the Rhs family which in the EDL933 O157:H7 genome had IGC fragments of lengths 1216, 1212, 564, 357, 257, 196, 98, 97, and 96 (median = 257; average = 455). Neither the size ranges nor the mean converted lengths among these four strains differ significantly from one another. When extended to the analysis of the newer *E. coli* genomes and the *Shigella* genomes (Table 5.7), the conversion lengths for most genomes are not significantly different from one another; however, the genomes at the extreme ends of the distribution do differ significantly from each other: *Shigella boydii*, with a median converted length of 552 differs significantly from the *E. coli* UPEC strains 536 (median length = 66) and UTI89 (median length = 82) (Kruskal-Wallis One-Way ANOVA on Ranks, $p<0.001$; Dunn's multiple pairwise comparison procedure, $p<0.05$; for all other comparisons, $p>0.05$). For all strains and sequences considered together, the median length of converted fragments by these methods was 175 nucelotides. The average size was 323 $\pm$ 40 (mean $\pm$ sem of mean sizes for each strain).

However, upon examining these converted sequences in detail, we noticed that in many cases the ends of the identified converted sequence were coincident with the start or end of the clustered sequence (i.e., either began at position 1 of the sequences being tested or ended at a base number equal to the length of the sequence). This suggested the possibility that the transferred sequence actually extended beyond the end of the sequence tested, which, by the method used, was limited to just the clustered gene. In addition, we suspected that additional converted sequences that began in the clustered sequence but were too short to achieve statistical significance without testing their full length might also be present. To test this, we extended the sequence further along the genome both upstream and downstream from the clustered sequence, in order to determine the full length of the converted fragment. In pilot tests, extensions of various lengths (100, 200, 300, etc.) were tested to see how long the extension should be in order that no more than 5% of the identified conversions that had at least one end in the clustered sequence should start or end at the end of the tested sequence (which now included 100 or more additional upstream and downstream nucleotides). The pilot test determined that extensions of 600 nucleotides at either end achieved that goal for EDL933, Sakai, CFT073, and K12 (MG1655) strains of *E. coli*.

Table 5.8: Gene conversions identified with flanking region of 600 bp included in the calculation.

| Strain name | Gene conversions, flank = 600 | | | | |
|---|---|---|---|---|---|
| | Total number | Smallest (bp) | Largest (bp) | Median (bp) | Mean (bp) |
| Escherichia_coli_K12_MG1655 | 76 | 8 | 3836 | 718[a,c] | 757 |
| Escherichia_coli_K12_W3110 | 68 | 8 | 3836 | 855[a] | 935 |
| Escherichia_coli_O157H7_EDL933 | 683 | 15 | 3107 | 460[d] | 569 |
| Escherichia_coli_O157H7(Sakai) | 774 | 9 | 3882 | 428[d] | 571 |
| Escherichia_coli_UTI_536 | 144 | 6 | 1464 | 232[b,c,d] | 486 |
| Escherichia_coli_UTI_UTI89 | 87 | 6 | 1155 | 126[b] | 303 |
| Escherichia_coli_UTI_CFT073 | 308 | 10 | 1500 | 234[b] | 393 |
| | | | | | |
| Shigella_Boydii_Sb227 | 452 | 9 | 3417 | 668[a] | 676 |
| Shigella_Dysenteriae | 432 | 17 | 3912 | 715[a] | 711 |
| Shigella_Flexneri_2a_301 | 514 | 7 | 1858 | 762[a] | 771 |
| Shigella_Flexneri_2a_2457T | 644 | 4 | 1817 | 668[a] | 682 |
| Shigella_Flexneri_5_8401 | 592 | 8 | 2015 | 700[a] | 741 |
| Shigella_Sonnei_Ss046 | 217 | 15 | 2762 | 648[a] | 721 |

IGC fragments wholly contained within the flanking regions have been excluded.
*Values marked with the same superscripts are not significantly different from each other, but do differ significantly from values marked with no superscripts in common (Kruskal-Wallis One Way Analysis of Variance on Ranks, $p<0.001$; pairwise comparisons by Dunn's method, $p < 0.05$).

Table 5.8 shows the lengths of converted sequences when a flanking extension of 600 nucleotides from the genomic sequence was added onto both ends of each sequence. In determining lengths of conversion fragments, we excluded all conversions that started and ended only in the flanking region, and thus included only those conversions that were either fully contained or had at least one end within the clustered sequence. The result is that the average size of converted fragments was much larger than detected with the previous method. For all strains and sequences considered together, the median length of converted fragments by these methods was 668 nucleotides. The average size was 640 ± 47 (mean ± sem of mean sizes for each strain). The closer agreement between mean and median for these data than for the analysis without the flanking regions added also suggests a "better behaved" data set. The Rhs family in EDL933 is somewhat of an exception, in that the number of IGCs increased to 17 IGCs but with a range of lengths from 38 to 1886 nucleotides, so that the median length actually decreased to 98 while the average length IGC decreased to 390.

Also, with the length analysis accomplished with flanking sequences included in the calculation, significant differences by the previous method are still significant and some trends in the original data set now become significant: As before, median converted lengths for the UPEC strains 536 (median = 232) and UTI89 (median = 126) had significantly smaller average lengths than *Shigella boydii* (median = 668). In addition, other comparisons are significant. Converted fragments of all of the UPEC strains (536, UTI89, and CFT073 (median = 234)) are significantly smaller than for not only *Shigella boydii*, but also compared to all of the other *Shigella* genomes, as well as compared to the W3110 strain of *E. coli* (see statistics summarized in Table 5.8). Similarly, both O157:H7 strains (EDL933 and Sakai) also have significantly smaller converted lengths than all of the *Shigella* strains, as well as W3110 *E. coli*. The MG1655 *E. coli* strain had average converted fragment lengths (718) near to the overall median and hence was only significantly different from two of the UPEC strains, UTI89 and CFT073. Overall, one can conclude that the converted lengths of the *Shigella* species were significantly longer than most, while the pathogenic *E. coli* strains, especially the UPEC strains, were shorter than most.

### 5.8.6 Higher numbers of conversions when full conversion lengths are analyzed

As previously noted, extending the DNA sequence of the clustered sequences might be expected not only to reveal the full length of converted sequences but also to allow the detection of sequences that in the core sequence were too short to have achieved statistical significance in the GENECONV analysis. In fact, the number of gene conversions detected increased an average of approximately 3-fold for *E. coli* strains (Table 5.8) and approximately 25-fold for *Shigella* spp. over the numbers detected without the addition of the flanking sequences.

As summarized in Figure 5.9, with flanking sequences in the analysis the numbers of apparent gene conversions in all *Shigella* spp. except for *Shigella sonnei* is now comparable to the numbers of conversions in the O157:H7 strains, about 500 conversions. However, two of the UPEC strains, 536 and UTI89, have only about 50 conversions, which is comparable to the numbers in the K12 strains, while *Shigella sonnei* and the UPEC strain CFT073 are intermediate, with about 200 conversions. The count of conversions does not include any that are fully contained within the flanking sequences (i.e., at least one end must be in the gene being analyzed).

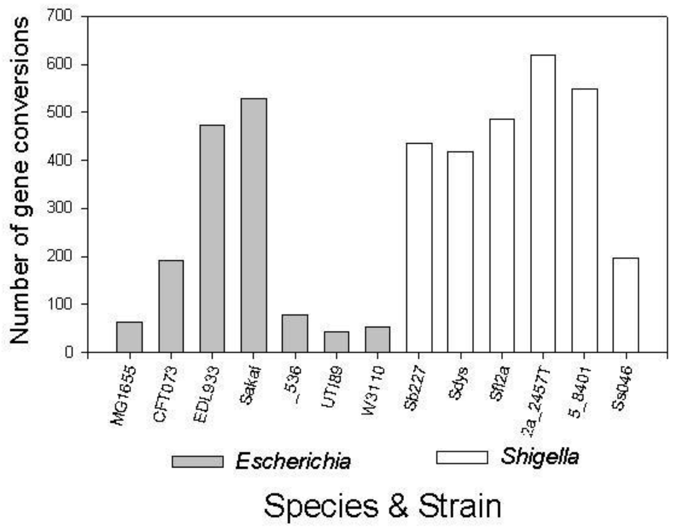


Figure 5.9: Number of IGCs in *E. coli* and *Shigella* strains determined with flanking sequences 600 bp long including in the analysis.

Without the flanking sequences, the very large multigene families in *Shigella* had not contributed much to the overall numbers of gene conversions. The even larger increase in numbers of conversions in *Shigella* spp. than in *E. coli* strains when flanking sequences were taken into account appeared in many cases to be associated with conversions identified in the largest gene families. For example, in *Shigella boydii* sb227, the multigene family with 170 members had 49 global inner fragment pairs identified as having evidence of gene conversions. It therefore became of interest to identify the types of genes in these large families, as well as to consider how they may differ from other genes in which IGC fragments were identified.

### 5.8.7 Genes exhibiting evidence of gene conversion

Analysis of protein identifications (PIDs) of genes showing gene conversion reveals that these genes generally fall into four categories: (1) enzymes and other functional proteins coded for by multiple genes, (2) toxin-antitoxin pairs, (3) prophage proteins, and (4) proteins in insertion sequences and associated transposases.

Only six families of functional bacterial proteins showed evidence of Intragenomic Gene Conversion in four or more of the strains analyzed (Table 5.9). These families are multidrug efflux system and related efflux pumps, L-serine deaminases and dehydratases, FeS binding subunits of oxidoreductases and glutamate synthase, porins and related outer membrane proteins, proteins of rhs elements, and L-ribulose-5-phosphate 4-epimerase and related enzymes. In contrast to the overall higher numbers of IGCs in the O157:H7 strains, compared to the K-12 strains, the number of families exhibiting these IGCs in the *Shigella* species is significantly lower than in the K-12 strains ($p<0.02$; t test). The total number of IGCs in these six genes averaged 14.5 for the K12 strains, 16 for the O157:H7 strains, 6.7 for the UTI strains, and 4.5 for the *Shigella* strains.

Among other types of proteins exhibiting IGCs, proteins of toxin-antitoxin pairs include, for example, the antitoxin of the YeeV-YeeU system found in both K12 strains and similar proteins identified in UTI strains. For the strains that showed an enormous increase in family sizes (*Shigella* spp.) and the highest total number of IGCs (O157:H7 strains), these increases came primarily from the presence of insertion sequences and families of prophage proteins, respectively. For example, of the 37 families identified as having IGCs in the EDL933 O157:H7 strain (using the non-flank method), 33 families were identified as being proteins from various variants of prophage CP-933. Even for K12 strains, prophages accounted for several of the IGCs: for the MG1655 K12 strain, out of the 6 families exhibiting IGCs, three families are CP4 prophage protein families.

The PID annotations for the W3110 K12 strains have similar protein names to these prophage families but do not designate them as prophage proteins. As noted earlier, relatively few IGCs were identified in *Shigella* species and strains unless the analysis was done with flanking sequences included. However, with flanking sequences included, the majority of the families and almost all of the IGCs were in insertion sequences (ISs) or associated transposases. For example, *Shigella sonnei* had only 24 identified IGCs (of which 4 were IS proteins) in 11 families without the flanking sequences. When flanking sequences were used in the analysis, 28 families exhibited 217 IGCs, of which 138 in 12 families were annotated as ORFs in ISs, and 14 in 3 families were in transposases.

Table 5.9: Intragenomic gene conversions identified in functional bacterial protein families with >2 members[1]

| Strain name | Genbank accession | drug efflux systems[2] | serine catabolism[3] | FeS subunits[4] | outer membrane pores[5] | rhs core proteins[6] | ribulose-phosphate-epimerase[7] | Families with IGCs |
|---|---|---|---|---|---|---|---|---|
| Escherichia_coli_K12_MG1655 | U00096 | + | ++ | + | ++ | ++ | + | 6 |
| Escherichia_coli_K12_W3110 | AP009048 | + | ++ | + | - | ++ | + | 5 |
| Escherichia_coli_O157H7_EDL933 | AE005174 | ++ | ++ | ++ | + | ++ | - | 5 |
| Escherichia_coli_O157H7(Sakai) | BA000007 | ++ | ++ | + | + | ++ | - | 5 |
| Escherichia_coli_UTI_536 | CP000247 | + | - | + | ++ | - | + | 4 |
| Escherichia_coli_UTI89 | CP000243 | + | ++ | ++ | ++ | - | + | 5 |
| Escherichia_coli_UTI_CFT073 | AE014075 | - | ++ | + | ++ | - | - | 3 |
| | | | | | | | | |
| Shigella_Boydii_Sb227 | CP000036 | ++ | ++ | - | - | ++ | - | 3 |
| Shigella_Dysenteriae | CP000034 | ++ | - | - | - | + | - | 2 |
| Shigella_Flexneri_2a_301 | AE005674 | + | ++ | - | + | - | - | 3 |
| Shigella_Flexneri_2a_2457T | AE014073 | - | ++ | - | - | - | - | 1 |
| Shigella_Flexneri_5_8401 | CP000266 | ++ | ++ | - | - | - | + | 3 |
| Shigella_Sonnei_Ss046 | CP000038 | ++ | ++ | + | - | ++ | - | 4 |

[1]Listed proteins had IGCs in >4 of the 13 genomes analyzed. ++, IGCs detected with and without flanking sequences included in the calculation; +, IGCs detected only with flanking sequence (600 bp) used in the calculation; -, no IGCs detected by either method.
[2]"drug efflux systems" family PIDs typically include acridine efflux pump, permease, acriflavine resistance protein, aminoglycoside pump, integral membrane protein, multidrug efflux, etc.
[3]"serine catabolism" family PIDs include L-serine deaminases and L-serine deydratases.
[4]"FeS subunits" family PIDs include the small FeS containing subunits of putative or predicted oxidoreductase and glutamate synthase.
[5]"outer membrane pores" family PIDs typically include several of the following descriptions: porin, phosphoporin, outer membrane pore protein, outer membrane protein, etc.
[6]"rhs core proteins" family PIDs typically include several of the following descriptions: rhsA(or B or C or G) protein in the rhs element, rhs core protein, putative protein in rhs element, etc.
[7]"ribulose phosphate epimerase" family PIDs include L-ribulose-5-phosphate 4-epimerase, putative epimerase/aldolase, and probable sugar isomerase.

## 5.9 *Summary and Future Work*

In this chapter, we demonstrated the implementation of a scientific workflow environment to analyze IGCs in microbial genomes. A methodology to identify IGCs is described and that analysis is carried out on hundreds of complete bacterial genomes available from NCBI. This analytical tool not only automates the process of identifying IGCs, but also in the process of developing this tool we have created generic processors for accessing and downloading web-based data sets and for parsing and reformatting outputs of genetic analysis tools to be able to function as inputs of other tools. The result is a developed tool that can be simply understood, shared, modified, and reused in other bioinformatic applications.

The biological results demonstrate the generality of IGCs across a large number of species of bacteria. Several principles emerge:

(1) The number of IGCs varies greatly both between and within species. Variation within a species was previously demonstrated for *E. coli* by Morris and Drouin [63]. This is now shown to be a general principle, as illustrated by at least 8 of the 11 species listed in Table 5.2, and further highlighted by detailed results for *S. pyogenes* (Figure 5.5).

(2) While pathogens frequently appear among the bacteria with the most IGCs, some species of non-pathogenic bacteria also have a large number of IGCs. Moreover, pathogens are also among the species that have the lowest numbers of IGCs. Thus, both *Neorickettsia sennetsu* and *Ureaplasma urealyticum*, which have no IGCs, are considered to be pathogens [44, 74]. *Prochlorococcus marinus*, which is not considered a pathogen,

has strains that vary from no IGCs to many IGCs (Table 5.2). The frequent occurrence of pathogens in the output of these analyses may partially reflect the fact that genomic sequencing thus far has mostly focused on bacteria that can cause harm to humans or livestock, and thus most of the genomes so far completed are pathogens. While the relationship IGCs to pathogenicity is discussed further below, there clearly is no necessary relationship.

(3) The addition of flanking sequences to the core family sequence enables the full length and more significant instances of IGC to be detected (e.g., Table 5.1 and Table 5.3). In a previous publication [4] we have discussed the phenomenon of "cryptic recombinations," that is, recombinations that have occurred but cannot be detected by sequence analysis either because the converted sequence is too similar to the original sequence (hence, not enough differences to be detected statistically) or because there is not enough neighboring reference sequence to be able to differentiate the converted sequence from its unchanged context. In that study, simulations of the gene conversion and detection process showed that longer neighboring sequences enabled greater detection of these previously "cryptic" events. This conclusion is supported by the results of adding the flanking regions in order to detect these otherwise cryptic recombinations.

(4) *E. coli*, for which 4 genomes had previously been analyzed for IGCs [63], appears to be somewhat of a special case, given that the numbers of IGCs in several *E. coli* strains and their *Shigella* relatives are at the extreme end of the distribution, with respect to high numbers of IGCs and the numbers or sizes of gene families in which they

are observed.  These may be interesting genomes to investigate the mechanisms involved in the extremes of these phenomena; however, a species like *S. pyogenes* might be considered more "typical."

The data on *S. pyogenes* may be interesting from a clinical perspective.  The strains with the highest numbers of IGCs are in serotypes M3, M6, and M18.  M3 organisms are said to cause a disproportionate number of invasive disease cases, including necrotizing fasciitis, bacteremia, and streptococcal toxic shock syndrome and can exhibit epidemic behavior  [13]. In the U.S., serotype M18 has been associated with acute rheumatic fever outbreaks [81].  An interesting question therefore is whether the high level of DNA mobility suggested by the high IGC count in these strains may facilitate their pathogenic potential.  This is not to suggest that IGC is necessary for pathogenicity since, for example, a study in Canada [90] found that the most frequent serotypes of *S. pyogenes* infections in the blood, brain and cerebrospinal fluid were M1 (28.2%), M28 (9.2%), and M12 (9.1%), all of which have low IGC levels in the present study.

We extended the IGC analysis of Morris and Drouin [63-65] but also contradicted the hypothesis that pathogenic strains have higher IGC levels than non-pathogenic strains.  By using a scientific workflow system to extend the analysis to include flanking sequences, larger numbers of IGCs can be detected and the full length of IGCs that formerly appeared to terminate at the end of the open reading frame of the genes being analyzed can be determined.  Specific bacterial proteins whose sequences may have undergone concerted evolution by exchanging genetic material have been identified.

With respect to the hypothesis that pathogenic strains have higher IGC levels, this chapter shows that the hypothesis is incorrect when additional pathogenic strains are taken into account. We first verified the previous observation [63] that the O157:H7 pathogenic strains of *E. coli* and the CFT073 UTI strain have higher IGC frequencies than does the K12 MG1655. However, with additional data, the hypothesis is contradicted on several accounts: (1) analysis of two additional UTI strains had much lower IGC levels, closer to the number of IGCs in the two non-pathogenic K12 strains than to the CFT073 UTI strain previously analyzed; (2) when analyzed by the original method without adding flanking sequences to the analysis, the *Shigella* species and strains, all of which are pathogenic derivatives of *E. coli*, had low numbers of IGCs, including *Shigella dysenteriae* in which the IGC counts were below that of the K12 strains; and (3) when considering the IGCs in only the functional bacterial genes (i.e., those genes listed in Table 5.9), the *Shigella* species and strains actually had lower number of genes with IGCs than the non-pathogenic K12 strains. In this regard, the three pathogenic strains upon which the original analysis of Morris and Drouin [63] was based appear to be special cases of exceptionally high IGCs. This is particularly evident in Figure 5.7, where it is clear that these three pathogenic strains appear distant from the main cluster of strains, and also that the significant increase in IGCs with family size is strictly dependent on the exceptional positions of the two O157:H7 strains. The increase in IGCs is not a property associated with pathogenicity in general but rather a special property, particularly of the O157:H7 strains. Alhiyafi et al [6] made a similar point with respect to pathogenicity of various strains of *Streptococcus pyogenes*, in which it was concluded that the most frequent serotypes of *S. pyogenes* in infections in the blood, brain and cerebrospinal fluid actually had the lower IGC levels out of 9 *S. pyogenes* serotypes considered in that study.

Also, we identify IGCs in six gene families of >3 members that code for bacterial proteins (as distinct from IS or prophage sequences that also have IGCs). A recent report by Morris and Drouin [65] similarly listed "backbone" genes identified as having IGCs. While several genes, discussed below, are the same as those in the present report, many details differ significantly. The differences may be due to the facts that (a) the present chapter analyzes a more recently updated set of genomes; (b) this chapter includes additional *E. coli* and *Shigella* species; (c) the analysis here encompasses neighboring sequences in order to allow identification of the full length of IGCs; and (d) the methods used here align nucleotide sequences with ClustalW prior to running GENECONV, which differs from Morris and Drouin's procedure of aligning the protein sequences with ClustalW first, a seemingly minor difference that inserts gaps in the resultant nucleotide sequences in slightly different places and does result in small quantitative differences in results. The justification for our modification is that recombination depends on alignment of similar DNA sequences, and hence computational alignment should reflect the biological mechanism.

*Rhs* gene families are interesting in that they served, in effect, as a positive control in identifying computationally a set of genes that had previously been shown to undergo intragenomic recombination experimentally. The *rhs* genes were originally identified as "recombination hot spots" (hence, the *rhs* designation) with a measured rate of ectopic recombination of 2 x $10^{-4}$ between *rhsA* and *rhsB* in K-12 [55]. Eight *Rhs* regions in the *E. coli* genome are said to contain a core region that has been maintained among the various *Rhs* cores by ectopic recombination. *RhsA* protein is associated with outer membrane proteins and mutations of it affect polysaccharide biosynthesis [62]. Although Morris and Drouin [65] listed gene conversion of *rhs* genes as "specific to the K12 genome," in our analysis GENECONV found numerous conversions among these families of genes not only in K-12, but also in O157:H7, and several *Shigella* genomes (Table 5.9). The discrepancy may, in part, be due to the fact that several of the named genes in K12, *RhsB* and *RhsD*, do not appear to have a clearly annotated homolog in O157:H7, where several of the genes in the *Rhs* family of O157:H7 are simply designated as "unknown" or "hypthetical" proteins associated with an *Rhs* element. In any case, the evidence that these genes actually do exhibit IGC experimentally corroborates identifying IGCs in the *Rhs* families computationally. The present research analyzed IGCs only in families having more than two members. Empirical demonstrations of gene conversions between gene pairs *tufA/tufB [9, 46]* and *gadA/gadB* [15] are similarly supported by GENECONV identification of IGCs in these two member families [65].

Among the other functional bacterial genes that exhibit IGCs, the L-serine hydratase/deaminase family is of some interest because of the possibly important roles of serine accumulation and catabolism in mediating colonization by *E. coli*. Mutation of L-serine deaminase genes in CFT073 results in a competitive defect of these strains in colonization of murine urinary tract [8]. Perhaps a further indication of the importance of serine in colonization is the fact that differences in serine chemotaxis relative to aspartate chemotaxis are present *E. coli* strains from different host animals [30]. Thus, the presence of redundancy of serine hydratases/deaminases and their concerted evolution maintained by IGC may be a positive fitness trait.

Finally, we consider the large increase in identification of IGCs in *Shigella* when the flanking sequences are included in the computation and the high levels of IGCs in general in the O157:H7 strains. These high levels of IGCs are primarily due to their identification in IS and prophage sequences. For *Shigella*, the IGC detection method practically functioned as an "IS discovery method," and one may wonder whether some of the "hypothetical proteins" that have IGCs according to GENECONV may represent as yet undescribed IS regions. Although computationally equivalent to the identification of IGCs in "backbone" genes of the bacteria, a fair question to ask is whether apparent recombination among such sequences really represents a bacterial mechanism. For lysogenic phage, which are really a type of viral infection of the genome, it is certainly possible that the multiple copies and versions of the prophages may represent pre-existing variation and/or recombination prior to the time of infection. Hence, the evidence of IGCs among these sequences may not represent bacterial mechanisms at all or at least not a mechanism mediated by their current hosts. Similarly, the mobility of IS sequences may mean that gene conversions among these sequences represent unique mechanisms that occur when the IS sequences are moving, rather than chromosomal mechanisms that would be required by genes such as serine deaminases or the *rhs* genes.

The number and identity of genes exhibiting intragenomic recombination varies widely among pathogenic and non-pathogenic strains of *Escherichia coli* and its derivative *Shigella* strains, indicating no consistent association with pathogenicity of the strains. Six bacterial gene families determined computationally to exhibit intragenomic recombination include the *Rhs* gene family, previously shown empirically to undergo intragenomic recombination, and the serine deaminase/dehydratase family, for which concerted evolution via intragenomic recombination may help maintain serine utlization as a positive fitness trait.

# CHAPTER 6 : CONCLUSIONS AND FUTURE WORK

Scientific workflows have become increasingly popular as a new method for scientists to develop and design complex and distributed scientific processes to enable and accelerate many scientific discoveries. The advantages of high-performance Grid-based computing for scientific workflows, and the problems noted with existing scientific workflow management systems (SWFMSs), provide the rationale for the development of a new scientific workflow system for the efficient use of heterogeneous scientific workflow systems and utilizing Grid computing. In this dissertation, we addressed how SWFMSs can be utilized to serve the bioinformatic community. In summary, our main contributions are:

- We propose a scientific workflow system to design, develop, and execute scientific workflows from heterogeneous scientific workflows.
- We propose a scheduling technique for the parallel execution of heterogeneous scientific workflows.
- We present GENOMEFLOW-based scientific workflow applications to showcase the capability of our system.
- We developed a recombination simulation scientific workflow for simulating recombination and using GENECONV to test the effect of pairwise differences in a diverse population on the detectability of recombination. A program to simulate the DNA recombination process was developed. History of the recombinant events is saved for further comparison and analysis. The known history of recombination occurring in the simulation was compared with the output of

putative recombinations detected by a well known highly ranked recombination detection program (GENECONV). The results show that the recombination detection software fails to identify more than 50% of recombination events, designated by this dissertation as “cryptic recombinations.”

- Decreases in recombination rate owing to pairwise differences resulted in population clusters analogous to sympatric speciation under specific conditions and decreases in detectability of recombination, a phenomenon that we call ‘cryptic recombination’. This computational method demonstrated the value of scientific workflow methods for analyzing a complex process and data driven problem.
- Intragenomic Gene Conversion (IGC) is important in the evolution of bacteria but has only been analyzed computationally in a few strains of *Escherichia coli*. Results show that IGC varies greatly, both between different species and among multiple genomes of the same species. We analyze for the first time the large variation of IGC in the pathogen *Streptococcus pyogenes*, and also in non-pathogenic bacteria.

In the following, we list some interesting research directions for future work:

- Extended GENOMEFLOW scientific workflow system to support more scientific workflow systems.
- Extended GENOMEFLOW scientific workflow system to support more heterogeneous environment.
- Optimize data movements between heterogeneous environments.

# REFERENCES

[1] Afzal, A., Darlington, J., and Mcgough, A.S. QoS-constrained stochastic workflow scheduling in enterprise and scientific Grids. *The Seventh IEEE/ACM International Conference on Grid Computing*, 2006: p. 1-8.

[2] Alhiyafi, J. and Lu, S. Developing Scientific Workflows from Heterogeneous Scientific Workflows. *The Eighth International Conference on Web Services*, 2010. submitted.

[3] Alhiyafi, J., Lu, S., and Ram, J.L. Simulation of genomic recombination and the influence of DNA sequence diversity on cluster formation and detectability of recombination: a scientific workflow method. *International Journal of Computational Biology and Drug Design*, 2009. 2(1): p. 81-99.

[4] Alhiyafi, J., Sabesan, C., Lu, S., Harriott, M., Jurczyszyn, D., Ritchie, R.P., Gonzales, F.S., and Ram, J.L. Computational methods for analysis of cryptic recombination in the performance of genomic recombination detection software. *Proceedings of IEEE International Symposium on Bioinformatics and Life Science Computing (BLSC'07)*, 2007. AINAW07: p. 707-712.

[5] Alhiyafi, J., Sabesan, C., Lu, S., and Ram, J.L. Association analysis between Intragenomic Gene Conversions and pathogenicity in genomes of *Escherichia coli* and *Shigella* spp.: a scientific workflow approach. *International Journal of Functional Informatics and Personalised Medicine*, 2009. 2(1): p. 57-76.

[6] Alhiyafi, J., Sabesan, C., Lu, S., and Ram, J.L. RECOMBFLOW: a scientific workflow environment for Intragenomic Gene Conversion analysis in bacterial

genomes, including the pathogen *Streptococcus pyogenes. International Journal of Bioinformatics Research and Applications*, 2009. 5(1): p. 1-19.

[7] Allewalt, J.P., Bateson, M.M., Revsbech, N.P., Slack, K., and Ward, D.M. Effect of temperature and light on growth of and photosynthesis by Synechococcus isolates typical of those predominating in the Octopus Spring microbial mat community of Yellowstone National Park. *Applied and Environmental Microbiology*, 2006. 72(1): p. 544-550.

[8] Anfora, A.T., Haugen, B.J., Roesch, P., Redford, P., and Welch, R.A. Roles of serine accumulation and catabolism in the colonization of the murine urinary tract by *Escherichia coli* CFT073. *Infection and Immunity*, 2007. 75(11): p. 5298-5304.

[9] Arwidsson, O. and Hughes, D. Evidence against reciprocal recombination as the basis for tuf gene conversion in Salmonella enterica serovar typhimirium. *Journal of Molecular Biology*, 2004. 338: p. 463-467.

[10] Baldo, L., Bordenstein, S., Wernegreen, J.J., and Werren, J.H. Widespread recombination throughout Wolbachia genomes. *Molecular Biology and Evolution*, 2006. 23(2): p. 437-499.

[11] Barga, R.S., Fay, D., Guo, D., Newhouse, S., Simmhan, Y., and Szalay, A. Efficient scheduling of scientific workflows in a high performance computing cluster. *The Sixth International Workshop on Challenges of Large Applications in Distributed Environments*, 2008: p. 63-68.

[12] Bearhop, S., Fieldler, W., Furness, R.W., Votier, S.C., Waldron, S., Newton, J., Bowen, G.J., Berthold, P., and Farnsworth, K. Assortative mating as a mechanism for rapid evolution of a migratory divide. *Science*, 2005. 310(5747): p. 502-504.

[13] Beres, S.B., Richter, E.W., Nagiec, M.J., Sumby, P., Porcella, S.F., DeLeo, F.R., and Musser, J.M. Molecular genetic anatomy of inter- and intraserotype variation in the human bacterial pathogen group A Streptococcu. *Proceedings of the National Academy of Sciences of the United States of America*, 2006. 103(18): p. 7059-7064.

[14] Berger, H., Hacker, J., Juarez, A., Hughes, C., and Goebel, W. Cloning of the chromosomal determinants encoding haemolysin production and mannose resistant haemagglutination in *Escherichia coli*. *Journal of Bacteriology*, 1982. 152: p. 1241-1247.

[15] Bergholz, T.M., Tarr, C.L., Christensen, L.M., Betting, D.J., and Whittam, T.S. Recent gene conversions between duplicated glutamate decarboxylase genes (*gadA and gadB*) in pathogenic *Escherichia coli*. *Molecular Biology and Evolution*, 2007. 24(10): p. 2323-2333.

[16] Bethenod, M.T., Thomas, Y., Rousset, F., Frérot, B., Pélozuelo, L., Genestier, G., and Bourguet, D. Genetic isolation between two sympatric host plant races of the European corn borer, Ostrinia nubilalis Hubner. II: assortative mating and host-plant preferences for oviposition. *Heredity*, 2005. 94(2): p. 264-270.

[17] BLAST. Available from: http://www.ncbi.nlm.nih.gov/.

[18] Blythe, J., Jain, S., Deelman, E., Gil, Y., Vahi, K., Mandal, A., and Kennedy, K. Task scheduling strategies for workflow-based applications in Grids. *IEEE International Symposium on Cluster Computing and the Grid* 2005. 2: p. 759-767.

[19] Bowers, S. and Ludäscher, B. Actor-oriented design of scientific workflows. *Lecture Notes in Computer Science*, 2005. 3716: p. 369-384.

[20] Bright, L. and Maier, D. Efficient scheduling and execution of scientific workflow tasks.

[21] Chebotko, A., Fei, X., Lin, C., Lu, S., and Fotouhi, F. Storing and querying scientific workflow provenance metadata using an RDBMS. *The Third IEEE International Conference on e-Science and Grid Computing* 2007: p. 611-618.

[22] Chebotko, A., Lin, C., Fei, X., Lai, Z., Lu, S., Hua, J., and Fotouhi, F. VIEW: a VIsual SciEntific Workflow management system. *The First IEEE International Workshop on Scientific Workflows*, 2007: p. 207-208.

[23] Chen, S.L., Hung, C.-S., Xu, J., Reigstad, C.S., Magrini, V., Sabo, A., Blasiar, D., Bieri, T., Meyer, R.R., Ozersky, P., Armstrong, J.R., Fulton, R.S., Latreille, J.P., Spieth, J., Hooton, T.M., Mardis, E.R., Hultgren, S.J., and Gordon, J.I. Identification of genes subject to positive selection in uropathogenic strains of *Escherichia coli*: a comparative genomics approach. *Proceedings of the National Academy of Sciences*, 2006. 103(15): p. 5977-5982.

[24] ClustalW. Available from: http://www.ebi.ac.uk/.

[25] Davidson, S. and Freire, J. Provenance and scientific workflows: challenges and opportunities. *ACM SIGMOD International Conference on Management of Data* 2008: p. 1345-1350.

[26] Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Patil, S., Su, M., Vahi, K., and Livny, M. Pegasus: mapping scientific workflows onto the Grid. *Across Grids Conference*, 2004.

[27] Deelman, E. and Chervenak, A. Data management challenges of data-intensive scientific workflows. *The Eighth IEEE International Symposium on Cluster Computing and the Grid* 2008: p. 687-692.

[28] Deelman, E., Singh, G., Su, M., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G., Good, J., Laity, A., Jacob, J., and Katz, D. Pegasus: a framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 2005. 13(3): p. 219-237.

[29] Drouin, G. Characterization of the gene conversions between the multigene family members of the yeast genome. *Journal of Molecular Evolution*, 2002. 55(1): p. 14-23.

[30] Dzinic, S.H., Luercio, M., and Ram, J.L. Bacterial chemotaxis differences of *Escherichia coli* isolated from different hosts. *Canadian Journal of Microbiology*, 2008. 12(1): p. 1043-1052.

[31] Ezawa, K., Oota, S., and Saitou, N. Genome-wide search of gene conversions in duplicated genes of mouse and rat. *Molecular Biology and Evolution*, 2006. 23(5): p. 927-940.

[32] Falzon, G. and Li, M. Evaluating heuristics for Grid workflow scheduling. *The Fifth International Conference on Natural Computation*, 2009: p. 227-231.

[33] Fei, X., Lu, S., and Lin., C. A MapReduce-enabled scientific workflow composition framework. *IEEE International Conference on Web Services* 2009: p. 663-670.

[34] Foster, I. and Kesselman, C., The Grid: blueprint for a future computing infrastructure. 1999: *Morgan Kaufmann Publishers*.

[35] Fraser, C., Hanage, W.P., and Spratt, B.G. Recombination and the nature of bacterial speciation. *Science* 2007. 315(5811): p. 476-480.

[36] Friesen, M.L., Saxer, G., Travisano, M., and Doebeli, M. Experimental evidence for sympatric ecological diversification due to frequency-dependent competition in *Escherichia coli Evolution*, 2004. 58(2): p. 245-260.

[37] GENECONV. Available from: http://www.math.wustl.edu/~sawyer/geneconv/.

[38] GridWorks. PBS Professional 10.0 User's Guide. *Altair Engineering, Inc.*, 2008.

[39] Hanage, W.P., Fraser, C., and Spratt, B.G. Fuzzy species among recombinogenic bacteria. *BMC Biology* 2005. 3: p. 6-12.

[40] Hayashi, K., Morooka, N., Yamamoto, Y., Fujita, K., Isono, K., Choi, S., Ohtsubo, E., Baba, T., Wanner, B.L., Mori, H., and Horiuchi, T. Highly accurate genome sequences of *Escherichia coli* K-12 strains MG1655 and W3110. *Molecular Systems Biology*, 2006. doi:10.1038/msb4100049: p. 1-5.

[41] Hein, J. Reconstructing evolution of sequences subject to recombination using parsimony. *Mathematical Biosciences*, 1990. 98(2): p. 185-200.

[42] Hochhut, B., Wilde, C., Balling, G., Middendorf, B., Dobrindt, U., Brzuszkiewicz, E., Gottschalk, G., Carniel, E., and Hacker, J. Role of pathogenicity island-associated integrases in the genome plasticity of uropathogenic *Escherichia coli* strain 536. *Molecular Microbiology*, 2006. 61(3): p. 584-595.

[43] Hollinsworth, D. The workflow reference model. *Workflow Management Coalition*, 1994.

[44] Honma, Y., Yada, Y., Takahashi, N., Momoi, M.Y., and Nakamura, Y. Certain type of chronic lung disease of newborns is associated with Ureaplasma urealyticum infection in utero. *Pediatrics International*, 2007. 49(4): p. 479-484.

[45] Hu, J., Zhang, Y., Qian, W., and He, C. Avirulence gene and insertion element-based RFLP as well as RAPD markers reveal high levels of genomic polymorphism in the rice pathogen Xanthomonas oryzae pv. oryzae. *Systematic and Applied Microbiology*, 2007. 30(8): p. 587-600.

[46] Hughes, D. Co-evolution of the tuf genes links gene conversion with the generation of chromosomal inversion. *Journal of Molecular Biology*, 2000. 297: p. 355-364.

[47] Jin, Q., Yuan, Z.H., Xu, J.G., Wang, Y., Shen, Y., Lu, W.C., Wang, J.H., Liu, H., Yang, J., Yang, F., Qu, D., Zhang, X.B., Zhang, J.Y., Yang, G.W., Wu, H.T., Dong, J., Sun, L.L., Xue, Y., Zhao, A.L., Gao, Y.S., Zhu, J.P., Kan, B., Chen, S.X., Yao, Z.J., He, B.K., Chen, R.S., Ma, D.L., Qiang, B.Q., Wen, Y.M., Hou, Y.D., and Yu, J. Genome sequence of *Shigella flexneri* 2a, Insights into

pathogenicity through comparison with genomes of *Escherichia coli* K12 and O157. *Nucleic Acids Research*, 2002. 30: p. 4432-4441.

[48] Kepler. Available from: http://kepler-project.org.

[49] Lederberg, J. and Tatum, E.L. Gene recombination in *Escherichia Coli*. *Nature*, 1946. 158(4016): p. 558-558.

[50] Lenski, R.E. and Travisano, M. Dynamics of adaptation and diversification: A 10,000-generation experiment with bacterial populations. *Proceedings of the National Academy of Sciences of the United States of America*, 1994. 91(15): p. 6808-6814.

[51] Lenski, R.E., Winkworth, C.L., and Riley, M.A. Rates of DNA sequence evolution in experimental populations of *Escherichia coli* during 20,000 generations. *Journal of Molecular Evolution*, 2003. 56(4): p. 498-508.

[52] Liao, D. Gene conversion drives within genic sequences:  concerted evolution of ribosomal RNA genes in bacteria and archaea. *Journal of Molecular Evolution*, 2000. 51: p. 305-317.

[53] Lin, C., Lu, S., Fei, X., Chebotko, A., Lai, Z., Pai, D., Fotouhi, F., and Hua, J. A reference architecture for scientific workflow management systems and the VIEW SOA solution. *IEEE Transactions on Services Computing*, 2009. 2(1): p. 79-92.

[54] Lin, C., Lu, S., Fei, X., Pai, D., and Hua, J. A task abstraction and mapping approach to the shimming problem in scientific workflows. *IEEE International Conference on Services Computing*, 2009: p. 284-291.

[55] Lin, R.J., Capage, M., and Hill, C.W. A repetitive DNA sequence, *rhs*, responsible for duplications within the *Escherichia coli* K-12 chromosome. *Journal of Molecular Biology*, 1984. 177: p. 1-18.

[56] Liu, K., Chen, J., Jin, H., and Yang, Y. A Min-Min average algorithm for scheduling transaction-intensive Grid workflows. *The Seventh Australasian Symposium on Grid Computing and e-Research*, 2009. 99.

[57] Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J., and Zhao, Y. Scientific workflow management and the KEPLER system: research articles. *Concurrency and Computation: Practice & Experience*, 2006. 18(10): p. 1039-1065.

[58] Maheswaran, M., Ali, S., Siegel, H.J., Hensgen, D., and Freund, R. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. *Heterogeneous Computing Workshop*, 1999.

[59] Mandal, A., Kennedy, K., Koelbel, C., Marin, G., Mellor-Crummey, J., Liu, B., and Johnsson, L. Scheduling strategies for mapping application workflows onto the grid. *High Performance Distributed Computing*, 2005: p. 125-134.

[60] Martin, D.P., Williamson, C., and Posada, D. RDP2: recombination detection and analysis from sequences alignments. *Bioinformatics*, 2005. 21: p. 260-262.

[61] Mayers, A., McGough, S., Furmento, N., Lee, W., Gulamali, M., Newhouse, S., and Darlington, J. Workflow expression: comparison of spatial and temporal approaches. *Workflow in Grid Systems Workshop*, 2004.

[62] McNulty, C., Thompson, J., Barrett, B., Lord, L., Andersen, C., and Roberts, I.S. The cell surface expression of group 2 capsular polysaccharides in *Escherichia coli*: the role of KpsD, RhsA, and a multi-protein complex at the pole of the cell. *Molecular Microbiology*, 2006. 59(3): p. 907-922.

[63] Morris, R.T. and Drouin, G. Ectopic gene conversions in four *Escherichia coli* genomes: increased recombination in pathogenic strains. *Journal of Molecular Evolution*, 2004. 58(5): p. 596-605.

[64] Morris, R.T. and Drouin, G. Ectopic gene conversions in bacterial genomes. *Genome*, 2007. 50: p. 975-984.

[65] Morris, R.T. and Drouin, G. Similar ectopic gene conversion frequencies in the backbone genome of pathogenic and nonpathogenic Escherichia coli strains. *Genomics*, 2008. 92: p. 168-172.

[66] Mulvey, M.A., Schilling, J.D., and Hultgren, S.J. Establishment of a persistent *Escherichia coli* reservoir during the acute phase of a bladder infection. *Infection and Immunity*, 2001. 69: p. 4572-4579.

[67] Nitzberg, B., Schopf, J.M., and Jones, J.P. PBS Pro: Grid computing and scheduling attributes. *University of Chicago as Operator of Argonne National Laboratory*.

[68] Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Gover, K., Pocock, M., Wipat, A., and Li, P. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 2004. 20(17): p. 3045-3054.

[69] Posada, D. Evaluation of methods for detecting recombination from DNA sequences: empirical data. *Molecular Biology Evolution*, 2001. 19(5): p. 708-717.

[70] Posada, D. and Crandall, K.A. Performance of methods for detecting recombination from DNA sequences: Computer simulations. *Proceedings of the National Academy of Science of the United States of America*, 2001. 98: p. 13757-13762.

[71] Prodan, R. and Wieczorek, M. Bi-criteria scheduling of scientific Grid workflows. *IEEE Transactions on Automation Science and Engineering*, 2009.

[72] Pupo, G.M., Lan, R., and Reeves, P.R. Multiple independent origins of Shigella clones of Escherichia coli and convergent evolution of many of their characterisitcs. *Proceedings of the National Academy of Sciences of the United States of America*, 2000. 97(19): p. 10567-10572.

[73] Rahman, M., Venugopal, S., and Buyya, R. A dynamic critical path algorithm for scheduling scientific workflow applications on global Grids. *IEEE International Conference on e-Science and Grid Computing*, 2007: p. 35-42.

[74] Rikihisa, Y., Zhang, C., Kanter, M., Cheng, Z., Ohashi, N., and Fukuda, T.A. Analysis of p51, groESL, and the major antigen P51 in various species of Neorickettsia, an obligatory intracellular bacterium that infects trematodes and mammals. *Journal of Clinical Microbiology*, 2004. 42(8): p. 3823-3826.

[75] Santoyo, G. and Romero, D. Gene conversion and concerted evolution in bacterial genomes. *FEMS Microbiology Reviews*, 2005. 29(2): p. 169-183.

[76] Sawyer, S. Statistical tests for detecting gene conversion. *Molecular Biology and Evolution*, 1989. 6(5): p. 526-538.

[77] Sawyer, S.A. GENECONV: a computer package for the statistical detection of gene conversion. Distributed by the author. *Department of Mathematics, Washington University in St. Louis*, 1999.

[78] Senkul, P. and Toroslu, I.H. An architecture for workflow scheduling under resource allocation constraints. *Information Systems* 2005. 30(5): p. 399-422.

[79] Shen, P. and Huang, H.V. Homologous recombination in *Escherichia coli*: dependence on substrate length and homology. *Genetics*, 1986. 112(3): p. 441-457.

[80] Smith, J.M. Analyzing the mosaic structure of genes. *Journal of Molecular Evolution*, 1992. 34: p. 126-129.

[81] Smoot, J.C., Barbian, K.D., Gompel, J.J.V., Smoot, L.M., Chaussee, M.S., Sylva, G.L., Sturdevant, D.E., Ricklefs, S.M., Porcella, S.F., Parkins, L.D., Beres, S.B., Campbell, D.S., Smith, T.M., Zhang, Q., Kapur, V., Daly, J.A., Veasy, L.G., and Musser, J.M. Genome sequence and comparative microarray analysis of serotype M18 group A Streptococcus strains associated with acute rheumatic fever outbreaks. *Proceedings of the National Academy of Sciences of the United States of America*, 2002. 99(7): p. 4668-4673.

[82] Spooner, D., Cao, J., Jarvis, S., He, L., and Nudd, G. Performance-aware Workflow Management for Grid Computing. *The Computer Journal, Oxford University Press*, 2004.

[83] Stef-Praun, T., Clifford, B., Foster, I., Hasson, U., Hategan, M., Small, S.L., Wilde, M., and Zhao, Y. Accelerating medical research using the Swift workflow system. *Studies in Health Technology and Informatics*, 2007. 126: p. 207-216.

[84] Swift. Available from: http://www.ci.uchicago.edu/swift/index.php/.

[85] Tan, W., Chard, K., Sulakhe, D., Madduri, R., Foster, I., Soiland-Reyes, S., and Goble, C. Scientific workflow as services in caGrid: a Taverna and gRAVI approach. *IEEE International Conference on Web Services*, 2009: p. 413-420.

[86] Tan, W., Missier, P., Madduri, R., and Foster, I. Building scientific workflow with Taverna and BPEL: a comparative study in caGrid. *The Fourth International Workshop on Engineering Service-Oriented Applications*, 2008: p. 118-129.

[87] Taylor, I., Shields, M., Wang, I., and Harrison, A. Visual Grid workflow in Triana. *Journal of Grid Computing*, 2006. 3: p. 153-169.

[88] Triana. Available from: http://www.trianacode.org.

[89] Tsaousis, A.D., Martin, D.P., Ladoukakis, E.D., Posada, D., and Zouros, E. Widespread recombination in published animal mtDNA sequences. *Molecular Biology and Evolution*, 2005. 22(4): p. 925-933.

[90] Tyrrell, G.J., Lovgren, M., Forwick, B., Hoe, N.P., Musser, J.M., and Talbot, J.A. M types of group a streptococcal isolates submitted to the National Centre for Streptococcus (Canada) from 1993 to 1999. *Journal of Clinical Microbiology*, 2002. 40(12): p. 4466-4471.

[91] Vilas-Boas, G., Sanchis, V., Lereclus, D., Lemos, M.V.F., and Bourguet, D. Genetic differentiation between sympatric populations of *Bacillus cereus* and

*Bacillus thuringiensis. Applied and Environmental Microbiology*, 2002. 68(3): p. 1414-1424.

[92] Vulić, M., Dionisio, F., Taddei, F., and Radman, M. Molecular keys to speciation: DNA polymorphism and the control of genetic exchange in enterobacteria. *Proceedings of the National Academy of Science of the United States of America*, 1997. 94(18): p. 9763-9767.

[93] Vulić, M., Lenski, R.E., and Radman, M. Mutation, recombination, and incipient speciation of bacteria in the laboratory. *Proceedings of the National Academy of Science USA*, 1999. 96(13): p. 7348-7351.

[94] Watt, V.M., Ingles, C.J., Urdea, M.S., and Rutter, W.J. Homology requirements for recombination in *Escherichia coli. Proceedings of the National Academy of Science of the United States of America*, 1985. 82(14): p. 4768-4772.

[95] Wei, J., Goldberg, M.B., Burland, V., Venkatesan, M.M., Deng, W., Fournier, G., Mayhew, G.F., Plunkett, G.I., Rose, D.J., Darling, A., Schwartz, D.C., and Blattner, F.R. Complete genome sequence and comparative genomics of *Shigella flexneri* serotype 2a strain 2457T. *Infect Immun.*, 2003. 71(5): p. 2775-2786.

[96] Whittam, T.S. Genetic variation and evolutionary processes in natural populations of *Escherichia coli*. In: Neidhardt FC (ed.) *Escherichia coli* and *Salmonella*: cellular and molecular biology. *Washington, DC: American Society for Microbiology*, 1996. 2: p. 2708-2720.

[97] Wiuf, C., Christensen, T., and Hein, J. A simulation study of the reliability of recombination detection methods. *Molecular Biology and Evolution*, 2001. 18(10): p. 1929-1939.

[98] Yang, F., Yang, J., Zhang, X., Chen, L., Jiang, Y., Yan, Y., Tang, X., Wang, J., Xiong, Z., Dong, J., Xue, Y., Zhu, Y., Xu, X., Sun, L., Chen, S., Nie, H., Peng, J., Xu, J., Wang, Y., Yuan, Z., Wen, Y., Yao, Z., Shen, Y., Qiang, B., Hou, Y., Yu, J., and Jin, Q. Genome dynamics and diversity of *Shigella* species, the etiologic agents of bacillary dysentery. *Nucleic Acids Research*, 2005. 33(19): p. 6445-6458.

[99] Yu, J. and Buyya, R. A taxonomy of workflow management systems for Grid computing. *Journal of Grid Computing*, 2005. 3(3-4): p. 171-200.

[100] Yu, J. and Buyya, R. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. *Scientific Programming* 2006. 14(3,4): p. 217-230.

[101] Yu, J., Buyya, R., and Tham, C.K. Cost-based scheduling of scientific workflow applications on utility Grids. *The First International Conference on e-Science and Grid Computing* 2005: p. 140-147.

[102] Zhao, Y., Hategan, M., Clifford, B., Foster, I., Laszewski, G.v., Raicu, I., Stef-Praun, T., and Wilde, M. Swift: fast, reliable, loosely coupled parallel computation. *IEEE Congress on Services*, 2007: p. 190-206.

[103] Zhao, Z., Booms, S., Belloum, A., Laat, C.d., and Hertzberger, B. VLE-WFBus: a scientific workflow bus for multi e-Science domains. *The Second IEEE International Conference on e-Science and Grid Computing*, 2006: p. 11-11.

# ABSTRACT

## A SCIENTIFIC WORKFLOW SYSTEM FOR GENOMIC DATA ANALYSIS

by

**JAMAL ALI MUSLEH ALHIYAFI**

**May 2010**

**Advisor:** Dr. Shiyong Lu

**Major:** Computer Science

**Degree:** Doctor of Philosophy

Scientific workflows have become increasingly popular as a new computing paradigm for scientists to design and execute complex and distributed scientific processes to enable and accelerate many scientific discoveries. Although several scientific workflow management systems (SWFMSs) have been developed, there is a great need for an integrated scientific workflow system that enables the design and execution of higher-level scientific workflows, which integrate heterogeneous scientific workflows enacted by existing SWFMSs. On one hand, science is becoming increasingly collaborative today, requiring an integrated solution that combines the features and capabilities of different SWFMSs, which are typically developed and optimized towards one single discipline. One the other hand, such an integrated environment can immediately leverage existing and emerging techniques and strengths of various

SWFMSs and their supported execution environments, such as Cluster, Grid, and Cloud. The main contributions of this dissertation are: 1) We propose a scientific workflow system, called GENOMEFLOW, to design, develop, and execute higher-level scientific workflows, whose workflow tasks are themselves scientific workflows enacted by existing SWFMSs; 2) We propose a workflow scheduling algorithm, called GSA, to enable the parallel execution of such heterogeneous scientific workflows in their native heterogeneous environments; and 3) We implemented GENOMEFLOW towards the life science community and developed several GENOMEFLOW scientific workflows to demonstrate the capabilities of our system for genome data analysis applications.

# AUTOBIOGRAPHICAL STATEMENT

## JAMAL ALI MUSLEH ALHIYAFI

Jamal Alhiyafi received his Ph.D. degree in Computer Science from Wayne State University (USA) in 2010. He received his M.S. in Software Engineering and B.S. in Computer Science from University of Michigan-Dearborn (USA) in 2002 and 2000, respectively. He also received a Graduate Certificate in Scientific Computing from Wayne State University (USA) in 2006.

His current research interest is in the field of scientific workflow models, high performance computing, scientific workflow scheduling, and their usage in the area of bioinformatics. His research has resulted in three journal and one conference/workshop publications.

As a Ph.D. student at Wayne State University, Jamal Alhiyafi was a recipient of the National Science Foundation – IGERT Fellowship and the Ford Motor Company Computer Science Award.