TestChip Technologies, Inc. Optimization

CSE 4395 – Senior Design
Southern Methodist University
Spring Semester 2001
Professor Olinick

Monica Gaston
Niki Holland
Jana Lindley
Davis Peden
Jeff West
I. Executive Summary

As a rapidly growing electrical engineering firm, TestChip Technologies, Inc. felt that it was essential to find a more productive way to analyze and assign tasks to engineers. As client demand and their workforce grow, TestChip must have a program that will assign engineers to tasks systematically, rather than haphazardly. TestChip is also interested in a program that will forecast project length. This senior design project analyzes TestChip's current operations and ultimately provides TestChip with software that will fulfill these goals.

Excel spreadsheets are TestChip's main form of data organization. They currently track all projects, employee information and employee productivity in spreadsheets. Engineer job assignment is random. Although an engineer might not be the best one to perform a job, he/she will be assigned to the job, if he/she is available. As TestChip's workforce grows, they will no longer be able to rely on this nonsystematic assignment.

Another problem arises when TestChip needs to forecast the length of a given project. Currently, TestChip can only use intuition to forecast a project's length. After analyzing these current operations, two approaches were discussed to solve TestChip's problems.

The first approach to optimize Testchip's job assignment schedule was to use an integer linear program. Numerous problems were found when evaluating this approach. The huge number of variables that would be required to solve the problem would cause the program to take hours, possibly days to find an optimal solution. TestChip would also have to purchase a large, expensive supercomputer as well as linear programming software that could solve a problem of this size. Clearly, this approach would not work.

A second approach is a modified job shop model. A program could be written to assign each job of each project to an engineer who is skilled to perform the job. This program could then calculate the expected completion date of any project after assigning all of the jobs to engineers. In this way, TestChip could forecast project lengths. The senior design group formulated two different algorithms: shortest processing first and longest processing first. Both of these algorithms were coded by the Java-gurus of our group and were analyzed and discussed to decide the practicality of each of the programs as well as to analyze the best algorithm.

After analyzing the algorithms, we found that the shortest processing first program provided a slightly superior solution. However, we feel that with the limited amount of project data that was supplied to us, we are unable to thoroughly analyze which algorithm is superior.

It is our conclusion that the optimization program and the Access database structure will undoubtedly benefit TestChip in the future. The following pages provide detailed explanations of the project, the solution, and the results.
II. Project Background and Description

TestChip is a rapidly growing electrical engineering company, with an expanding workforce and rising client demand. Testchip was founded in Dallas, Texas, seven years ago by Mark Harward and Brenda Stoner, two former employees of Texas Instruments. They recently opened a new branch in Austin, Texas, in addition to their headquarters in Plano, Texas. Last October, Testchip was named one of the 500 fastest growing private companies in America. The company is planning to continue this expansion by doubling their workforce in the next year. Currently, the company is capable of handling its project scheduling and employee staffing with basic Microsoft Excel spreadsheets. However, TestChip is concerned that its growth will surpass its current capabilities of data management and analysis. TestChip needs a software system that can track company employee information, productivity, and current projects, as well as project schedules and standings.

Testchip's goal is to provide services to the semi-conductor manufacturing industry, a $200 billion dollar worldwide market. Because of the fast pace of technology and innovation in this industry, companies' biggest challenge is to stay ahead of the competition. Even an advance of one day can be worth millions of dollars. In any kind of innovation or development, test chips are necessary to deliver data about the semi-conductor manufacturing process. Because it is costly and time consuming for companies to create these test chips with internal resources, Testchip exists to allow these companies to outsource this process.

What is a Testchip?

Testchip has 10 to 30 projects in progress concurrently. Each project is composed of multiple modules, with each module made up of different types, such as library, non-library, and copy. These modules are skill-specific. Employees are qualified to work on one or all of the sections of the modules, as indicated in the employee capability spreadsheet. These sections do not have to be completed in any specific order; however, employees must work on only one section of the module at a time. Projects are composed of multiple module types, and there is at least one or more of each module type. For example, one project had 698 total modules, composed of 12 types. Another project had 500 total modules, composed of 5 types.

Testchip looks at skill set and availability when assigning employees to projects and modules. When there is an opening in a project, Testchip will look at the available employees that are capable of performing work on that type of module. From there, they haphazardly select an employee from that list. This random method of assigning workers to projects does not take productivity into account, a problem that will increase as the company and its demand grow.
Management currently uses intuition to forecast lengths. Because the company is relatively small, management is able to use what they know about their employees and rates of productivity to forecast how long it will take them to complete their modules and projects. However, as their workforce size increases, Testchip will no longer be able to rely on intuition as a forecasting method.

Productivity and project status are tracked manually in multiple Excel spreadsheets. They have no way of producing any type of reports or analyzing the data unless by hand. Additionally, data is not housed in one central location. It is difficult to use this data for analysis and forecasting if it is difficult to find and compile. Because of this manual process, there is no way of reporting data to Testchip employees.

TestChip's system must be capable of updating project statuses, indicating the employees staffed on each project. It should display current project progress, estimated project completion dates, and what resources are needed to finish the project on time. It should also be able to forecast when new projects can be started based on resource availability. The system will assign available employees to projects, minimizing costs and efficiently allocating its resources to match employees with assignments of their specified skill level. Project due dates, employee vacations, and unplanned absences must be taken into account when running the program. The system must run and compile data quickly, so that it can be updated at any time to account for changes. As well as project demand, the system must take into account the research and development requirement of the company. TestChip requires that twenty percent of employee hours per year be spent on R&D.

Ideally, TestChip would like to use this system as a forecasting device to estimate project time-lines based on resource availability as well as cost minimization. The system will also serve as an employee database. Employees will be able to access information such as their current project status, productivity levels, job capabilities, absences, and future project agendas. The system must transfer the results from its program to a form that is available for employee viewing.

Our group's assignment is to select the most appropriate technology to fulfill TestChip's requirements. There are several factors that we must consider when deciding what technology to use and when designing the system. The costs of creating and maintaining the system must be within TestChip's budget. Because TestChip's workforce and project assignments are growing quickly, the system must be capable of maintaining its standard pace despite an ever-increasing amount of data and/or variables. In addition, the technical capabilities of the TestChip employees who will be running the system must be taken into account. Our system must minimize the cost of each project while ensuring that the resources are available to complete each project by its due date. We want to maximize productivity rates on employee, project, and company levels. We must identify employee qualification and capabilities, as well as their productivity rates and hourly costs. We also will determine how to assign specific employees to specific projects, based on their qualifications and hourly rates.
III. Analysis of the Situation

The main objective of this project is to develop a scheduling model that will allocate engineers to specific projects based on their skill set and availability. TestChip would prefer for the system to be able to run on the computer hardware that they have available now. They are not interested in spending a lot of money to purchase optimization software that can only run on a large expensive supercomputer to solve their scheduling problem. Therefore, the system must be able to run on a personal computer in a short period of time as well as be easily updated and account for changes quickly. The model will be used to decide whether or not new projects will be able to be completed by their due date. TestChip can use the model to decide whether or not to accept new clients' projects or to renegotiate the due date of a certain project. The scheduling model needs to be able to track and update TestChip's employee information such as vacation and productivity in each skill set, current projects and their due dates, and project schedules and standings. Each project consists of many different module types as well as number of each module type. Each separate module is individually assigned to an employee who is qualified to do the module. The projects' standings can be calculated according to the number of modules that are completed for a given project. The model will use this information to efficiently assign engineers to projects according to their specified skill levels. This scheduling problem can be approached in two different ways.

In order to optimize the available resources, an initial attempt was made to use an integer linear programming model (ILP). A linear program is a linear system of equations that are solved in order to find an optimal solution. A linear program consists of an equation that is to be optimized, an objective function, and a set of other equations that the objective function is subject to. For example, an objective function can minimize cost or maximize productivity while being constrained by availability of workers, productivity levels, or available resources. An integer linear program specifies that each variable must be of an integer value. This type of linear program is ideal when assigning the number of workers to work certain shifts or assigning jobs to employees because you do not want a solution with .765 of a task to be assigned to one person. Hence, an integer linear program is a good choice for solving TestChip's problem.

To understand how modules will be assigned to engineers at Testchip, one must understand what each project consists of. As stated above, each project includes many different types of modules as well as number of each module type. Therefore, the TestChip linear programming model would have to have a variable for each employee, in each project, in each module. An example variable would be the following:

\[ E_{ijk} = 1 \text{ if employee number } i \text{ is assigned to do module } k \text{ of project } j, \text{ 0 otherwise, where,} \]

- \( i \) = the employee number and runs through the number of employees
- \( j \) = the project number and runs through the number of projects
- \( k \) = the module number of project \( j \) and runs through all modules in project \( j \)

The objective function for the TestChip model could maximize the productivity of each employee. An example objective function for employee number 1 in project 1 that includes only 4 modules could be the following:

Max productivity: \( 2.3E_{11} + 5.6E_{12} + 4.5E_{13} + 1.2E_{14} \)
Possible constraints could be availability of employees or project due dates. For example, the availability of a variable for an employee who is not qualified to perform a certain module could be set equal to zero, so that the program will not assign her to that module. This is shown below:

Subject to:
Availability: \( E_{543} = 0 \) (employee number \( i = 5 \) is not qualified for module \( k = 3 \) in project \( j = 4 \))

After development of the integer linear programming model, the feasibility of using such a model was immediately called into question. The number of integer variables that would be required to solve such a system would be too great for a personal computer to solve. In the example program discussed above, if there are 30 engineers, 5 projects and 25 modules per project, this already accounts for 3750 integer variables! This example is only with 5 projects and there are likely to be 10 - 30 projects at one given time. Clearly, the number of variables will become overwhelming and almost impossible to solve if TestChip continues to grow and expand in the next year. Even if TestChip had a large computer that could solve this ILP, they would have to purchase expensive linear programming software in order to solve this program. As mentioned above, TestChip is not interested in spending a large amount of money on this scheduling problem. In addition, a problem of this size would take a long time to solve and therefore could not be reevaluated easily and updating the information is difficult. TestChip would have to hire an employee that was proficient with linear programming software and could easily update and resolve the program. TestChip wants a program that is easily changed and resolved when new employees are hired, an employees' availability changes, or a new project is taken on. This is clearly not the case with an integer linear program and therefore, it is not feasible to use an integer linear program to solve TestChip's problem.

The next approach for solving the TestChip scheduling dilemma is to use a modified job shop model. A program could be written to assign each module of each project to an engineer who is adequately qualified to do the module. Initially the program will transfer project data from Excel spreadsheets to a Microsoft Access Database. The software will analyze the data in the database and assign engineers to projects based on engineer availability and skill sets. Two different job shop algorithms were discussed and coded in an attempt to find the best model.

The first model is the shortest process first (SPF). This model analyzes which employee would take the shortest amount of time to complete a module and assigns that module to him/her. The second model, which is the converse of the shortest process first algorithm, is the longest process first (LPF). This model analyzes which employee would take the longest amount of time to complete a module and assigns that module to him/her. After tasks are assigned to engineers, the software calculates a projected completion date for the project. This will allow TestChip to decide whether or not to accept new clients. Finally, the software returns the output data to the database. TestChip can then print out reports and job assignments that the software calculated. The following section provides an in depth analysis of the functions contained in the program and the two different algorithms.

**Why is this a job shop model?**

**What is the intuition behind the LPF algorithm?**
IV. Technical Description of Model

Summary

This section provides an in-depth examination of the software solution that has been developed to solve the TestChip Technologies Optimization problem. This software solution has four phases: database construction, Java JDBC data extraction, the optimization calculation, and output generation.

Database Construction

TestChip Technologies currently tracks project status using one Microsoft Excel spreadsheet for each project. Each project spreadsheet contains the following fields:

- Project Name
- Due Date

Each row in the project spreadsheet corresponds to a particular module type within the project. Each module type has the following fields associated with it:

- Module Name
- Module Type
- Total Number of Modules
- Area
- Number Specd
- Number Frozen
- Number Laid Out
- Number Documented
- Number TCTVerified
- Number Customer Signoff
- Number LIB
- Number NONLIB
- Number COPY

The Number LIB, Number NONLIB, and Number COPY fields are the most important fields of each module type. These fields break up the total number of modules of a particular type required for the project by the skills that are needed by the TestChip Engineers to complete each module. The Number LIB, Number NONLIB, and Number COPY fields when added together reflect the total number of modules of a particular type required for the project.

As mentioned before, TestChip Technologies currently tracks all of their projects using Microsoft Excel spreadsheets. The use of spreadsheets presented a major problem of how to easily access the data for all projects that are currently being worked on. An initial solution was to have TestChip migrate from using spreadsheets to Microsoft Access to track their data. Citing experience from previous attempts to migrate to another form of project tracking, TestChip stated that they wished for the system to work with the existing spreadsheet mode of project tracking.
To solve the data access problem, we decided to use Microsoft Visual Basic to extract the data from each of the spreadsheets and place the data into corresponding fields in a Microsoft Access database. The Visual Basic script is run from a form within the Access database. The reasons for migrating the data into an Access database are as follows:

1. Using Java DataBase Connectivity (JDBC), we can easily extract data for use in an optimization algorithm.
2. All of the records for each module in each project can be stored in the same table. Having all of the data in one database table is far more efficient than storing the data in multiple spreadsheets.
3. Microsoft Access provides functionality for producing meaningful reports while Excel does not.
4. Having a database backbone will allow for almost seamless migration to using a database to track projects in the future.

In addition to storing project status, the database also contains tables that hold TestChip employee information, employee productivity levels, and the results of the optimization calculations. Forms are provided to allow TestChip management to add, delete, and edit employees as well as to change employee skill sets and productivity levels as needed. All of the employee information requires manual data entry.

**Java JDBC Data Extraction**

We are using a Java program to compute the optimization schedules for the TestChip Optimization Project. In this phase, the employee, employee productivity, and project data are extracted from the Microsoft Access database that was created in the previous phase using Java DataBase Connectivity (JDBC). Three SQL database queries are used to retrieve information from the database. The first query retrieves the employee information from the database. Once the employee information has been retrieved, it is used to generate an employee object for each employee that contains an employee ID, an available flag, a skillset and a worklist.

The next query retrieves the employee productivity information. This information is used to generate an employee skillset, which each employee object possesses. The skillset object contains a vector of skill objects that each contain a moduleId, a jobTypeld, and an associated productivity level.

The final query extracts the project information from the Access Database. Each record in the project database corresponds to a module set within a project. A Project Set is created which contains multiple projects (a project corresponds to one spreadsheet file). A project contains multiple Module Sets. A module set contains multiple modules (a module corresponds to one record of the master table of the Access Database). A module contains elements corresponding to all fields that are contained in the master project database.
The following is a graphical representation of the different object and relationships that exist once all of the queries are complete.
Optimization Calculation

In order to actually perform the optimization calculations, the list of modules must first be sorted by their due date. Once the list of modules is placed in order by their due dates (closest date to furthest date), the optimization calculations can be done using one of two optimization algorithms: modified Shortest Process First (SPF) and modified Longest Process First (LPF). It is necessary to sort all of the modules by their due dates in order to ensure that projects are assigned based on the closest due date first.

Modified Shortest Process First Algorithm (SPF)

In short, the Modified Shortest Process First algorithm assigns a module to the employee who is not already busy, possesses the necessary skills, and who can complete the task in the fastest time. The basic algorithm is this:

```plaintext
for(i=0; i<NumberOfJobs; increment i){
    Retrieve job from project list
    if (employees are NOT available) {
        set all employees to available
    }
    for(x=0; x<NumberOfEmployees; increment x){
        if (Employee has skill and is available) {
            Get the Employee's Productivity
            If (Employee's Productivity is the best) {
                Set employee to busy
                Set the job to assigned
                Add the job to the employee's worklist
                Add the job to the Optimization set
            } //end if Employee has skill
        } //end if Employee's Productivity is the best
    } //end employee for
} //end job for
```
Modified Longest Process First Algorithm (LPF)

The Modified Longest Process First Algorithm assigns a module to the employee who is not already busy, possesses the necessary skills, and who will take the longest time to complete the task out of the available employees. The basic algorithm is this:

```
for(i=0; i<NumberOfJobs; increment i)
    Retrieve job from project list
    if (employees are NOT available)
        set all employees to available
    for(x=0; x<NumberOfEmployees; increment x)
        if (Employee has skill and is available)
            Get the Employee’s Productivity
            If (Employee’s Productivity is the worst)
                Set employee to busy
                Set the job to assigned
                Add the job to the employee’s worklist
                Add the job to the Optimization set
            } //end if Employee has skill
            } //end if emp productivity is the best
        } //end employee for
    } //end jobfor
```

Projected Completion Date Calculation

Once both of the algorithms have run, the resulting optimization sets containing the optimized work lists for each employee are analyzed to compute a projected project due date. The total number of days that it will take each employee to complete all of his or her assigned tasks for a particular project will be recorded. The number of days that it will take to complete the particular project will correspond to the greatest total number of days of all of the employees assigned to a project. A projected finish date is then computed by adding the total number of business days (excluding weekends) that it will take to complete the project to the current date. The projected completion date and the corresponding optimization set are then returned to the database. The algorithm for computing the projected completion date based on a total number of days required to do the project is:

```
For(I=0; I<NumberOfDaysRequiredForProject; increment I) {
    If(CurrentDayOfWeek is Friday) {
        Add 3 days to current date
    } //end Friday
    If(CurrentDayOfWeek is Saturday) {
        Add 2 days to current date
    } //end Saturday
    Else{
        Add 1 day to current date
    } //end all other days
} //end for
```
Output of Results

Once the projected completion date for each project and the optimization set have been produced, it is necessary to return the data to the Access database so that TestChip management can analyze the data. Each of the algorithms in the previous phase will produce its own set of projected completion dates and its own optimization set.

The Access database contains result tables for each of the 2 algorithms: SPFResults and LPFResults. Each time the software package is run, these tables are first cleared to remove any results from a previous execution. Using JDBC and SQL INSERT statements, records are added to the appropriate table that contain the following fields using the projected completion date and data from the optimization set of each of the two algorithms:

- Projected Completion Date
- Project Name
- Employee ID
- Project Due Date
- Module Name
- Module Type
- JobType
- Number of Modules
- Number of Days to Complete a Module

Once the results of the program execution have been added to the appropriate tables in the database, reports are generated to output the projected completion date and the optimized work schedule.
V. Analysis and Managerial Interpretation

Both the shortest processing first and longest processing first algorithms provided a practical solution to TestChip's scheduling problem. With the supplied data, the modified shortest processing first (SPF) algorithm's solution was more optimal than the longest processing first algorithm. We can see this after analyzing the output that is returned to the database. The total number of business days that it takes to complete all three of the test projects under the SPF scheme is 235.69 days. The same value for LPF is 250.87 days. It is obvious that the SPF optimization scheme produces the best result; however, these results are data dependent.

When analyzing the time to complete each individual project using the same data, the SPF scheme proved to be the fastest for project #1 and project #2 while LPF was slightly faster for project 3. The following tables provide a summary of the project results. The average, minimum and maximum values represent number of days per job. A detailed summary of the program output is given in Appendix B.

<table>
<thead>
<tr>
<th>SPF</th>
<th>Project 1</th>
<th>Project 2</th>
<th>Project 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>9.1313675</td>
<td>2.622535</td>
<td>6.19302</td>
</tr>
<tr>
<td>Min</td>
<td>0.076923</td>
<td>0.153846</td>
<td>1.66667</td>
</tr>
<tr>
<td>Max</td>
<td>27.08333</td>
<td>9.769231</td>
<td>12.75</td>
</tr>
<tr>
<td>Total Days</td>
<td>111.7641</td>
<td>68.1859</td>
<td>55.73718</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LPF</th>
<th>Project 1</th>
<th>Project 2</th>
<th>Project 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>10.22885</td>
<td>2.769231</td>
<td>6.235755</td>
</tr>
<tr>
<td>Min</td>
<td>0.153846</td>
<td>1.538462</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>32.5</td>
<td>10.58333</td>
<td>11.76923</td>
</tr>
<tr>
<td>Total Days</td>
<td>122.7462</td>
<td>72</td>
<td>56.12179</td>
</tr>
</tbody>
</table>

After examining the data, we reached the conclusion that the number of days required for a particular project could be affected if one of the modules contained a large number of pieces. In the Appendix B data, Project 1 contains several modules that require over 100 pieces. These modules are being assigned as a whole to one employee. Because only one employee is working on a large module, the total number of days is dramatically increased due to these data points.

As a result of the data analysis, we decided that a better scheme of assigning modules to engineers would be to split the large modules up so that no more than 50 pieces are assigned at a given time. The output from this assignment scheme is shown in Appendix C. The data is summarized below. For SPF, the total time required to complete both projects is 240.65 days. For LPF, the total time required to complete both projects is 245.80 days.

<table>
<thead>
<tr>
<th>SPF</th>
<th>Project 1</th>
<th>Project 2</th>
<th>Project 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>3.336484</td>
<td>2.193093</td>
<td>3.104701</td>
</tr>
<tr>
<td>Min</td>
<td>0.083333</td>
<td>0.153846</td>
<td>0.25</td>
</tr>
<tr>
<td>Max</td>
<td>5</td>
<td>4.16667</td>
<td>4.16667</td>
</tr>
<tr>
<td>Total Days</td>
<td>116.7769</td>
<td>67.9859</td>
<td>55.88462</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LPF</th>
<th>Project 1</th>
<th>Project 2</th>
<th>Project 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>3.368938</td>
<td>2.319892</td>
<td>3.109687</td>
</tr>
<tr>
<td>Min</td>
<td>0.153846</td>
<td>0.230769</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>5</td>
<td>5</td>
<td>4.16667</td>
</tr>
<tr>
<td>Total Days</td>
<td>117.9128</td>
<td>71.91667</td>
<td>55.97436</td>
</tr>
</tbody>
</table>

After comparing the new data to the data produced when the program was run with no limit on the number of pieces that could be assigned to an employee at a given time, we made two conclusions. Using the SPF algorithm, it is better to allow employees to work on unlimited job assignments because the employees with the highest productivity rates are able to work on all the pieces of the module type that they are most skilled at without interruption. With the LPF algorithm, it is advantageous to place a limit on the number of pieces of a specific module that an
employee can work because the least skilled employee is taken off of the module after a maximum of 50 pieces are complete.

The managerial interpretation of this project is that the database and schedule optimization program provide TestChip with a more productive way to organize projects and employee information. Using this system, managers will be able to easily add new and update existing employee information as well as performance metrics using database forms. While the optimization program may not provide the most efficient schedule, it is undoubtedly more efficient than the existing method of assigning employees to projects.

It's too bad that TestChip couldn't give you a sample solution as benchmark.
VI. Conclusion and Critique

Currently, TestChip relies on managers to manually assign jobs to engineers. The designed job shop algorithms provide the managers with an automated method of project job assignment. Therefore, the managers can be more productive with the extra time that they will receive from the use of the program.

Using intuition to forecast job completion dates works well for TestChip now. But, as the company continues to grow and client demand increases, they will want a more accurate and systematic way of forecasting. Using the job shop algorithms, TestChip can easily run the programs when a new project arises to provide a client with a possible completion date.

TestChip will also be able to print meaningful reports from the Access Database tables. They could customize the reports in many different ways for example, to analyze many different project situations or provide average productivity levels per project.

With their current system, we recommend that they use the SPF algorithm because it completes the projects in the shortest amount of time. However, due to the limited data that was provided by TestChip, we are unable to see how it would work on a full size project load. Because of the small number of example projects that were provided, an in-depth analysis was difficult to perform. For this reason, we have provided Testchip with four different algorithm options. Testchip should run each of the four options and analyze which algorithm will work best based on their project load, workforce, and growth.
Appendix A

Optimization Program Code

TestChipOptimize.java
package ms_senior_design;

import java.util.HashSet;

public class TestChipOptimize {

    public static boolean DEBUG = false;

    public TestChipOptimize () {
        if (DEBUG)
            ();
    }

    public static void main(String[] args) {
        DatabaseAccess DBManager = new DatabaseAccess();
        DBManager.DoQueries();
        ProjectSet PL = DBManager.getProjectSet();
        if (DEBUG) System.out.println("Number Of Projects:");
        Project tempP;
        PL.sortByDate();
        ProjectSet SPFProjectSet = new ProjectSet(PL);
        ProjectSet LPFProjectSet = new ProjectSet(PL);
        if (DEBUG) for(int x = 0; x < PL.size(); x++) {
            PL.getProjectByIndex(x).printInfo();
        }
        OptimizeSPF SPF = new OptimizeSPF(ELIST, SPFProjectSet);
        SPF.optimize();
        ProjectSet SPFProjectList = SPF.getOptimizedProjectSet();
        LengthForecaster forecaster = new LengthForecaster(SPFProjectList, ELIST);
        forecaster.forecastDates();
        if (DEBUG) {
            System.out.println("SPF!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");
            for(int x = 0; x < SPFProjectList.size(); x++) {
                System.out.println("SPF Project# " + SPFProjectList.getProjectByIndex(x).projectID() + " StartDate:");
                SPFProjectList.getProjectByIndex(x).getStartDate() + " Jobs:"
            }
        }
        System.out.println("About to upload");
        DBManager.doUpload(ELIST, SPFProjectList, 1);
        System.out.println("Upload complete");
        Employee temp;
    ELIST.resetAvailability();
EList.clearJobLists();

OptimizeLPF LPF=new OptimizeLPF (EList,LPFProjectSet);
LPF.optimize();
if (DEBUG) System.out.println("********************************************");
if (DEBUG) for (int x=0; x<EList.size(); x++) {
    temp=EList.getEmployeeByIdx(x);
    System.out.println("EmployeeID:" +temp.getEmployeeID());
    temp.printJobList();
    System.out.println("----------------------------------------------------------------");
}
ProjectSet LPFProjectList =LPF.getOptimizedProjectSet();

LengthForecaster LPFforecaster = new LengthForecaster (LPFProjectList, EList);
LPFforecaster. forecastDates();
if (DEBUG) System.out.println("********************************************");
if (DEBUG) for (int x=0; x<LPFProjectList.size(); x++) {
    System.out.println("LPF Project#"+(x+1)+" Jobs:"+LPFProjectList.getProjectByIdx(x).getNumberOfJobs());
}

DBManager.doUpload(EList, LPFProjectList, 2);
System.out.println("Upload Complete!");
if (DEBUG) System.out.println(EList.getEmployeeByIdx(EList.size()-1).toString());
Appendix A

Optimization Program Code

OptimizeSPF.java
package ms_senior_design;

/**
 * Title: MS Senior Design
 * Description:
 * Copyright: Copyright (c) 2001
 * Company:
 * @author
 * @version 1.0
 */

public class OptimizeSPF {
    EmployeeSet EList;
    ProjectSet PList;
    OptimizationSet OSet;
    boolean DEBUG=false;
    ProjectSet optimizedProjectSet;

    public OptimizeSPF(EmployeeSet EList, ProjectSet PList) {
        this.EList=EList;
        this.PList=new ProjectSet(PList);
        if (DEBUG) System.out.println("IN SPF!");
        if (DEBUG) PList.printList();
        optimizedProjectSet = new ProjectSet();
    }

    public void optimize() {
        double bestProductivity = -1.0;
        int bestEmployeeID = -1;
        Project currentProject;
        Job theJob;
        JobSet JList=new JobSet();
        Employee theEmployee;
        boolean jobAssigned=false;

        for(int px=0; px<PList.size(); px++) {
            currentProject = PList.getProjectByIndex(px);
            JList=currentProject.getJobSet();
            if (DEBUG) System.out.println("Assigning "+JList.size() + " jobs from Project" + currentProject.projectID());
            while (JList.moreJobs()) {
                for(int jx=0; jx<JList.size(); jx++) {
                    bestProductivity = -1.0;
                    bestEmployeeID = -1;
                    theJob=JList.getJobByIndex(jx);

                    if(!theJob.isAssigned()) {
                        if (DEBUG) System.out.println("Assigning JobProject:" +theJob.getProjectID() + " Job#":" +jx + " JobType: "+theJob.getJobType() + " ModType: " +theJob.getModType() + " Number:" +theJob.getNumberofModules());

                        for(int ex=0; ex<EList.size(); ex++) {
                            if(EList.anyAvailable (theJob.getJobType(), theJob.getModType())) {
                                for(int reset=0; reset< EList.size(); reset++)
                                    EList.getEmployeeByIndex(reset).setAvailable();
                            }
                        }
                    }
                }
            }
        }
    }
}
theEmployee = EList.getEmployeeByIndex (ex);

if (theEmployee.hasSkill (theJob.getJobType (), theJob.getModType ())) {
    if (theEmployee.isAvailable () ) {
        if (DEBUG) System.out.println ("Employee#" + theEmployee.getEmployeeID () + " has the skill, is available, and is better. Productivity:" + theEmployee.getProductivity (theJob.getJobType (), theJob.getModType ())) ;
        double myProductivity = theEmployee.getProductivity (theJob.getJobType (), theJob.getModType ());
        if (bestProductivity < myProductivity) {
            bestProductivity = theEmployee.getProductivity (theJob.getJobType (), theJob.getModType ());
            bestEmployeeID = theEmployee.getEmployeeID ();
        } // end if
    } else {
        if (DEBUG) System.out.println ("Employee#" + theEmployee.getEmployeeID () + " has the skill and is NOT available. ");
    }
}
else {
    if (DEBUG) System.out.println ("Employee#" + theEmployee.getEmployeeID () + " does not have the skill.");
} // end employee for
// end assigned
if (bestEmployeeID != -1) {
    if (DEBUG) System.out.println ("Project:" + (px + 1) + " JOB ASSIGNED:" + jx + " EmployeeID: " + bestEmployeeID);
    (EList.getEmployeeByID (bestEmployeeID)).setBusy ();
    theJob.setAssigned ();
    (EList.getEmployeeByID (bestEmployeeID)).addJob (theJob);
    theJob.setWorkerID (bestEmployeeID);
    optimizedProjectSet .addProjectJob (theJob, px);
} // end if
} // end job for
} // end project for

public ProjectSet getOptimizedProjectSet () {
    if (DEBUG) System.out.println ("Optimized List:" );
    if (DEBUG) optimizedProjectSet .printList ();
    return optimizedProjectSet;
}
Appendix A

Optimization Program Code

OptimizeLPF.java
package ms_senior_design;

/**
 * Title: MS Senior Design
 * Description:
 * Copyright: Copyright (c) 2001
 * Company:
 * @author
 * @version 1.0
 */

class OptimizeLPF {

EmployeeSet EList;
ProjectSet PList;

ProjectSet optimizedProjectSet;

boolean DEBUG = false;

public OptimizeLPF (EmployeeSet EList, ProjectSet PList) {
    this.EList = EList;
    this.PList = new ProjectSet (PList);
    optimizedProjectSet = new ProjectSet ();
}

public void optimize () {
    double bestProductivity = 9999;
    int bestEmployeeID = -1;
    Project currentProject;
    Job theJob;

    JobSet JList = new JobSet();

    JList.clearAssignments();
    Employee theEmployee;

    boolean jobAssigned = false;

    for(int px=0; px<PList.size(); px++) {
        currentProject = PList.getProjectByIndex (px);
        JList = currentProject.getJobSet ();
        JList.clearAssignments ();

        if (DEBUG) System.out.println ("LPF: Assigning +JList.size()+" jobs from Project#" +currentProject.projectID());

        while (JList.moreJobs()) {
            for(int jx=0; jx<JList.size(); jx++) {
                bestProductivity = 9999;
                bestEmployeeID = -1;
                theJob = JList.getJobByIndex (jx);

                if (!theJob.isAssigned ()) {
                    if (DEBUG) System.out.println ("LPF: Attempting Assigning JobProject:" +theJob.getProjectID () +
                                                             "JOB#":" +jx+" JobType: " +theJob.getJobType () +
                                                             " ModType: " +theJob.getModType () +" Number:" +theJob.getNumberOfModules ());

                    for(int ex=0; ex<EList.size(); ex++) {
                        if (!EList.anyAvailable (theJob.getJobType (), theJob.getModType ())) {
                            for(int reset=0; reset<EList.size(); reset++)
                                EList.getEmployeeByIndex (reset).setAvailable ();
                        } else {
                            ex=0;
                        }
                    }

                    jobAssigned = true;
                    break;
                }
            }

            for(int kx=0; kx<JList.size(); kx++) {
                JList.setAvailable (kx, false);    //mark this job as unassigned
            }

            if (DEBUG) System.out.println ("LPF: Reassigning Job Project:" +theJob.getProjectID () +
                                              
                                              "JOB#":" +jx+" JobType: " +theJob.getJobType () +
                                              " ModType: " +theJob.getModType () +" Number:" +theJob.getNumberOfModules ());

            if (DEBUG) System.out.println ("LPF: Assigning +JList.size()+" jobs from Project#" +currentProject.projectID());
        }
    }

    }
theEmployee = EList.getEmployeeByEmailIndex(ex);

if (theEmployee.hasSkill(theJob.getJobType(), theJob.getModType())) {
    if (theEmployee.isAvailable()) {
        // if (DEBUG) System.out.println("Employee\"+theEmployee.getEmployeeID()+\" has the skill, is available, and is better. Productivity:\"+theEmployee.getProductivity(theJob.getJobType(), theJob.getModType())\"\");

        double myProductivity = theEmployee.getProductivity(theJob.getJobType(), theJob.getModType());

        if (bestProductivity > myProductivity) {
            bestProductivity = theEmployee.getProductivity(theJob.getJobType(), theJob.getModType());
            bestEmployeeID = theEmployee.getEmployeeID();
        } // end if
    } else {
        // if (DEBUG) System.out.println("Employee\"+theEmployee.getEmployeeID()+\" has the skill and is NOT available.";
    }
} else {
    // if (DEBUG) System.out.println("Employee\"+theEmployee.getEmployeeID()+\" does not have the skill.";
}
} // end employee for
} // end assigned

if (bestEmployeeID != -1) {
    if (DEBUG) System.out.println("LPF: Project:\"+(px+1)+\" JOB ASSIGNED: \"+jx+\" EmployeeID \"+bestEmployeeID);
    (EList.getEmployeeByID(bestEmployeeID)).setBusy();
    theJob.setAssigned();
    (EList.getEmployeeByID(bestEmployeeID)).addJob(theJob);
    theJob.setWorkerID(bestEmployeeID);
    optimizedProjectSet.addProjectJob(theJob, px);
} // end if
} // end job for
} // end project for

public ProjectSet getOptimizedProjectSet() {
    return optimizedProjectSet;
}
Appendix A

Optimization Program Code

LengthForecaster.java
package ms_senior_design;
import java.util.Calendar;
import java.util.Date;

/**
 * Title: MS Senior Design
 * Description: Copyright (c) 2001
 * Company: 
 * Author: Davis W. Peden
 * Version: 1.0
 */

public class LengthForecaster {

    ProjectSet projectSet = new ProjectSet();
    EmployeeSet employeeSet = new EmployeeSet();
    private boolean DEBUG = false;
    private boolean DEBUG2 = false;

    public LengthForecaster (ProjectSet pSet, EmployeeSet eSet) {
        if (DEBUG) System.out.println("Constructor called ");
        this.projectSet = pSet;
        this.employeeSet = eSet;
    }

    public void forecastDates () {
        int workerID=0;
        double prod;
        int number;
        double days;
        Employee employee;
        Calendar theDate = Calendar.getInstance();
        theDate.setTime(new Date());

        if (DEBUG) System.out.println("IN FORCASTER! ");
        if (DEBUG) projectSet.printList();

        for(int i=0; i<projectSet.size(); i++){
            if (DEBUG) System.out.println("************");
            if (DEBUG) System.out.println("Project Retrieved: " + projectSet.getProjectByIndex(i).projectId());
            double projectLength =0.0;

            for(int z=0; z<projectSet.getProjectByIndex(i).getJobSet().size(); z++){
                //if (DEBUG) System.out.println("Project #: ");
                if (DEBUG) System.out.println("Job #: " + z);
                workerID=projectSet.getProjectByIndex(i).getJobSet().getJobByIndex(z).getWorkerID();

                if (DEBUG) System.out.println("workerID: " + workerID);
                employee=employeeSet.getEmployeeByID(workerID);
                prod=employee.getProductivity (projectSet.getProjectByIndex(i).getJobSet().getJobByIndex(z).getJobType(), projectSet.getProjectByIndex(i).getJobSet().getJobByIndex(z).getModType());

                if (DEBUG) System.out.println("Productivity: " + prod);
                number = projectSet.getProjectByIndex(i).getJobSet().getJobByIndex(z).getNumberOfModules();
                projectLength +=(double)number/prod;

            } //if (DEBUG) System.out.println("Days: ");
            
        } //for
    }
}


if (days > projectLength)
    projectLength = days;
} // end if
"
} // end for

if (DEBUG) System.out.println("|*|************|*|" );
if (DEBUG) System.out.println("ProjectLength: " + projectLength);
if (DEBUG) System.out.println("|*|************|*|" );
projectSet.getProjectByIndex(i).setProjectLength(projectLength);

projectSet.getProjectByIndex(i).setStartDate(theDate.getTime());
theDate.setTime(computeProjectEndDate(projectLength, theDate));
if (DEBUG) System.out.println("End Date: " + theDate.getTime());
projectSet.getProjectByIndex(i).setEndDate(theDate.getTime());
} // end for
} // end forecastDates()

public Date computeProjectEndDate(double numOfDays, Calendar theDate) {
    if (DEBUG) System.out.println("ProjectLength: " + numOfDays);
    if (DEBUG) System.out.println("Input Date: " + theDate.getTime());
    for (int i = 0; i < numOfDays; i++) {
        if (theDate.get(Calendar.DAY_OF_WEEK) == 6) {
            theDate.add(Calendar.DAY_OF_YEAR, 3);
            if (DEBUG) System.out.println("Saturday!");
        } // end if
        else if (theDate.get(Calendar.DAY_OF_WEEK) == 7) {
            theDate.add(Calendar.DAY_OF_YEAR, 2);
            if (DEBUG) System.out.println("Sunday!");
        } // end else if
        else {
            theDate.add(Calendar.DAY_OF_YEAR, 1);
        }
    } // end for
    if (DEBUG) System.out.println("Returning " + theDate.getTime());
    return theDate.getTime();
} // end computeDueDate

// return theDate;
Appendix A

Optimization Program Code

DatabaseAccess.java
public class DatabaseAccess {

public static boolean DEBUG = true; // general debug
public static boolean DEBUG2 = true; // shows object contents

private EmployeeSet empSet;
private ProjectSet projectSet; // set of all records in DB
private Vector jobList = new Vector();
private Vector employees = new Vector();
private HashSet projectFileSet = new HashSet();

public DatabaseAccess () {
    if (DEBUG) System.out.println("Inside DatabaseAccess Constructor");
    projectSet = new ProjectSet();
} // end DatabaseAccess constructor

public void DoQueries () {
    String SqlEmpId = "SELECT EmployeeInfo.EmployeeID FROM EmployeeInfo;";
    if (DEBUG) System.out.println("SqlEmpId: " + SqlEmpId);
    String driverName = "sun.jdbc.odbc.JdbcOdbcDriver";
    if (DEBUG) System.out.println("driverName: " + driverName);
    String URL = "jdbc:odbc:testchip";
    if (DEBUG) System.out.println("URL: " + URL);
    Connection conId = null;
    Statement stmtId = null;
    ResultSet rsId = null;

    try {
        Class.forName(driverName).newInstance();
        conId = DriverManager.getConnection(URL);
        // Do queries...
    } catch (Exception e) {
        System.out.println("Exception: " + e.getMessage());
    } finally {
        if (conId != null) conId.close();
        if (stmtId != null) stmtId.close();
        if (rsId != null) rsId.close();
    }
} // end Doqueries

public void DoOtherQueries () {
    // Do other queries...
}

// Other methods...
} // end class DatabaseAccess
stmtId = conId.createStatement();
rsId=stmtId.executeQuery(SqlEmpId);

int id=0;
String lName=null;
String fName=null;

while(true){
    try{
        rsId.next();
        id=rsId.getInt("EmployeeId");
        if(DEBUG) System.out.println("id="+id);
        Employee employee = new Employee(id);
        employees.add(employee);
    }
    catch(SQLException sqle) {
        break;
    }
}

while (true) {
    try{
        rsId.next();
        id=rsId.getInt("EmployeeId");
        if (DEBUG) System.out.println("id=" + id);
        Employee employee = new Employee(id);
        employees.add(employee);
    }
    catch(SQLException sqle) {
        break;
    }
}

//end try


Connection conSkills = null;
Statement stmtSkills = null;
ResultSet rsSkills = null;
Class.forName(driverName).newInstance();
conSkills = DriverManager.getConnection(URL);
stmtSkills = conId.createStatement();
rsSkills=stmtSkills.executeQuery(SqlSkills);

int modId;
double prod;
int jobTypeID;

while (rsSkills.next()) {
    id=rsSkills.getInt("EmployeeId");
    modId=rsSkills.getInt("ModuleID");
    prod=rsSkills.getDouble("Productivity");
    jobTypeID=rsSkills.getInt("JobID");
    if (DEBUG) System.out.println("id="+id);
    if (DEBUG) System.out.println("modId="+modId);
    if (DEBUG) System.out.println("prod="+prod);
    Skill skill = new Skill(modId, jobTypeID, prod);
    Iterator iter=employees.iterator();
    Iterator iterator = employees.iterator();
    if (DEBUG) System.out.println("emplid = " + emp.getEmployeeID());
    //end if
}

//add skills to employee
for(int i=0; i<employees.size(); i++) {
Employee theEmp = (Employee) employees.get(i);
if (theEmp.getEmployeeID() == id) {
    theEmp.addSkill(skill);
} //end if
} //end for
} //end while

//Print Employees
Iterator iterator2 = employees.iterator();
if (DEBUG2) while (iterator2.hasNext()) {
    Employee emp2 = (Employee) iterator2.next();
    System.out.println("---------------------------------------------");
    System.out.println("EmpID: " + emp2.getEmployeeID());
    System.out.println("Available: " + emp2.isAvailable());
    for (int i = 0; i < emp2.getSkillSet().size(); i++) {
        Skill theSkill = (Skill) (emp2.getSkillSet().get(i));
        System.out.println("Module: " + theSkill.getModuleID() +
            " Productivity: " + theSkill.getProductivity());
    } //end for
} //end while

String sqlMaster = "SELECT Master.ProjectID, Master.ProjectName, Master.Date, Master.ModName,
FROM Master INNER JOIN ModuleNames ON Master.ModType = ModuleNames.ModuleType ORDER BY Master.ProjectID,
Master.ModName;";
Connection conMaster = null;
Statement stmtMaster = null;
ResultSet rsMaster = null;
Class.forName(driverName).newInstance();
conMaster = DriverManager.getConnection(URL);
stmtMaster = conMaster.createStatement();
rsMaster = stmtMaster.executeQuery(sqlMaster);

int projectID;
String projectName;
java.util.Date date;
String modName;
int type;
int totalNum;
double area;
int numSpecd;
int numFrozen;
int numLaidOut;
int numDocd;
int numTCTVerify;
int numCustSignoff;
int numLib;
int numNonLib;
int numCopy;
int jobType;

while (rsMaster.next()) {
    projectID = rsMaster.getInt("ProjectID");
    projectName = rsMaster.getString("ProjectName");
    projectFileSet.add(projectName);
    date = rsMaster.getDate("Date");
    modName = rsMaster.getString("ModName");
    type = rsMaster.getInt("ModuleID");
totalNum = rsMaster.getInt("TotalNum");
area = rsMaster.getDouble("Area");
numSpecd = rsMaster.getInt("NumSpecd");
numFrozen = rsMaster.getInt("NumFrozen");
numLaidOut = rsMaster.getInt("NumLaidOut");
numDocd = rsMaster.getInt("NumDocd");
numTCTVerify = rsMaster.getInt("NumTCTVerify");
numCustSignoff = rsMaster.getInt("NumCustSignoff");
numLib = rsMaster.getInt("NumLIB");
numNonLib = rsMaster.getInt("NumNONLIB");
numCopy = rsMaster.getInt("NumCOPY");

Module newModule = new Module(projectID, projectName, date, modName,
area, numSpecd, numFrozen, numLaidOut, numDocd,
numTCTVerify, numCustSignoff, numLib, numNonLib,
numCopy);

//if (DEBUG) System.out.println(newModule);
projectSet.addProjectModule(newModule, projectID);
} //end while
*/

Iterator jobIterator = jobList.iterator();
if (DEBUG) while (jobIterator.hasNext()) {
    Project job1 = (Project) jobIterator.next;
    System.out.println("---------------------------");
    System.out.println("ProjectName: "+ job1.getProjectName());
    System.out.println("Date: "+ job1.getDate());
    System.out.println("ModName: "+ job1.getModName());
    System.out.println("Type: "+ job1.getModType());
    System.out.println("TotalNum: "+ job1.getTotalNum());
    System.out.println("Area: "+ job1.getArea());
    System.out.println("NumSpecd: "+ job1.getNumSpecd());
    System.out.println("NumFrozen: "+ job1.getNumFrozen());
    System.out.println("NumLaidOut: "+ job1.getNumLaidOut());
    System.out.println("NumDocd: "+ job1.getNumDocd());
    System.out.println("NumTCTVerify: "+ job1.getNumTCTVerify());
    System.out.println("NumCustSignoff: "+ job1.getNumCustSignoff());
    System.out.println("NumLib: "+ job1.getNumLib());
    System.out.println("NumNonLib: "+ job1.getNumNonLib());
    System.out.println("NumCopy: "+ job1.getNumCopy());
    System.out.println("JobType: "+ job1.getJobType());
} //end if/
} //end try

catch (SQLException se) {
    se.printStackTrace();
} //end catch

catch (ClassNotFoundException cnfe) {
    cnfe.printStackTrace();
} //end catch

catch (IllegalAccessException iae) {
    iae.printStackTrace();
} //end catch

catch (InstantiationException ie) {
    ie.printStackTrace();
} //end catch

empSet = new EmployeeSet(employees);
//jobSet = new JobSet(jobList);
public void doUpload (EmployeeSet theList, ProjectSet theProjSet, int uLocation) {
    DEBUG=true;
    EmployeeSet eList = theList;
    ProjectSet pSet = theProjSet;
    int uploadLocation = uLocation;

    Employee emp;
    int projectID=0;
    String projectName;
    int empID = 0;
    java.util.Date theDate;
    String dateString;
    String modName;
    int modType;
    int jobType;
    int number;
    double prod;
    double days=0.0;
    JobSet jSet;
    double projectLength;
    java.util.Date startDate;
    java.util.Date endDate;
    String startDateString =null;
    String endDateString= null;
    int jeffcount=0;

    String driverName="sun.jdbc.odbc.JdbcOdbcDriver";
    String url = "jdbc:odbc:testchip";
    Connection conSPF = null;
    Connection conLPF = null;
    Statement stmtSPF = null;
    Statement stmtLPF = null;
    Statement delStmt = null;
    ResultSet rs = null;
    try{
        Class.forName (driverName ).newInstance ();
        conSPF=DriverManager.getConnection (url);
        conLPF=DriverManager.getConnection (url);
        String sqlDeleteStatement = null;
        if(uploadLocation ==1){
            sqlDeleteStatement = "DELETE * FROM SPFResults;" ;
        }
        else if(uploadLocation ==2){
            sqlDeleteStatement = "DELETE * FROM LPFResults;" ;
        }
        else if(uploadLocation ==3){
            sqlDeleteStatement = "DELETE * FROM CustomResults;" ;
        }
        delStmt=conSPF.createStatement ();
        delStmt.executeUpdate (sqlDeleteStatement );

        for(int x=0;x<eList.size();x++) {
            emp=eList.getEmployeeByIdex (x);
            empID=emp.getEmployeeID () ;
            jSet=emp.getJobList ();
            // if(DEBUG) System.out.println("-----------------------------");
        }
    }catch (SQLException e) {
        e.printStackTrace ();
    }
}
```java
//if(DEBUG) System.out.println("EmpID: "+empID);

for(int i=0;i<jSet.size();i++)
{
    //System.out.println("Putting "+jSet.size()+" records into DB");
    projectID = jSet.getJobByIndex(i).getProjectID();
    projectName = jSet.getJobByIndex(i).getProjectName();
    theDate = jSet.getJobByIndex(i).getDate();
    dateString = theDate.toString();
    modName = jSet.getJobByIndex(i).getModuleName();
    modType = jSet.getJobByIndex(i).getModuleType();
    jobType = jSet.getJobByIndex(i).getJobType();
    number = jSet.getJobByIndex(i).getNumberOfModules();
    prod = emp.getProductivity(jSet.getJobByIndex(i).getJobType(), jSet.getJobByIndex(i).getModuleType());
    days = (double)number/prod;
    projectLength = pSet.getProjectByID(projectID).getProjectLength();
    startDate = pSet.getProjectByID(projectID).getStartDate();
    endDate = pSet.getProjectByID(projectID).getEndDate();

    //DateFormat shortTime = DateFormat.getDateInstance(DateFormat.SHORT);
    if(DEBUG) System.out.println("Start: "+startDate);
    if(DEBUG) System.out.println("Duration: "+days);
    if(DEBUG) System.out.println("Project: "+projectName);
    if(DEBUG) System.out.println("EmployeeID: "+empID);
    if(DEBUG) System.out.println("DueDate: "+dateString);
    if(DEBUG) System.out.println("ModuleName: "+modName);
    if(DEBUG) System.out.println("ModuleType: "+modType);
    if(DEBUG) System.out.println("JobType: "+jobType);
    if(DEBUG) System.out.println("NumberOfModules: "+number);
    if(DEBUG) System.out.println("days: "+days);
    if(DEBUG) System.out.println("ProjectLength: "+projectLength);
    if(DEBUG) System.out.println("StartDate: "+startDate);
    if(DEBUG) System.out.println("EndDate: "+endDate);

    String sqlInsertStatement =null;
    if(uploadLocation ==1)
    { sqlInsertStatement = "INSERT INTO SPFResults (ProjectName, EmployeeID, DueDate, ModuleName, ModuleID, JobID, NumberOfMods, JobLength, ProjectLength, ProjectStartDate, ProjectEndDate) VALUES ('" + projectName +", "+empID+", "+dateString+", "+modName+", "+modType+", "+jobType+", "+number+", "+days+", "+projectLength+", "+startDateString+", "+endDateString+'");";
    }
    else
    { sqlInsertStatement = "INSERT INTO LPFResults (ProjectName, EmployeeID, DueDate, ModuleName, ModuleID, JobID, NumberOfMods, JobLength, ProjectLength, ProjectStartDate, ProjectEndDate) VALUES ('" + projectName +", "+empID+", "+dateString+", "+modName+", "+modType+", "+jobType+", "+number+", "+days+", "+projectLength+", "+startDateString+", "+endDateString+'");";
    }
    if(DEBUG) System.out.println(sqlInsertStatement);
    stmtSPF=conSPF.createStatement();
    stmtLPF=conLPF.createStatement();
    if(uploadLocation ==1)
    {
        //if(DEBUG) System.out.println(sqlInsertStatement);
        stmtSPF.executeUpdate(sqlInsertStatement);
        stmtLPF.executeUpdate(sqlInsertStatement);
    }
}"
```
stmtFP.executeUpdate(sqlInsertStatement);
}
else if(uploadLocation == 2){
stmtLPF.executeUpdate(sqlInsertStatement);
//if(DEBUG) System.out.println("Added job "+(jeffcount++));
}
//end for
}
//end Try
catch(Exception e) {
e.printStackTrace();
}//end catch

public EmployeeSet getEmployeeSet () {
    return empSet;
}

public ProjectSet getProjectSet () {
    return projectSet;
}

public HashSet getProjectFileSet () {
    return projectFileSet;
}

//public static void main(String args[]){
//    DatabaseAccess db = new DatabaseAccess();
//    db.DoQueries();
//} //end main */

}//end databaseAccess class
Appendix A

Optimization Program Code

ProjectSet.java
package ms_senior_design;
import java.lang.System;
//import java.util.*;
import java.util.Vector;
import java.util.Date;

public class ProjectSet {
    protected Vector theVector;
    boolean DEBUG = false;

    public ProjectSet () {
        theVector = new Vector();
    }

    public ProjectSet (ProjectSet PSet) {
        theVector = new Vector(PSet.theVector);
    }

    public void addProjectModule (Module addMe, int projectIDNumber) {
        int index = projectIDNumber - 1;
        if (DEBUG) System.out.println("Adding new Module!");
        Project updateProject;
        if (DEBUG) System.out.println("Vector size:" + theVector.size() + " ProjectID:" + projectIDNumber + " Index:" + index);
        if (theVector.size() <= index) {
            theVector.setSize (projectIDNumber);
            Project imNew = new Project (projectIDNumber);
            theVector.set (index, imNew);
            if (DEBUG) System.out.println("Adding new Project!");
        }
        updateProject = (Project) theVector.elementAt (index);
        if (updateProject == null)
            updateProject = new Project (projectIDNumber);
        updateProject.addModule (addMe);
        theVector.set (index, updateProject);
    }

    public void addProjectJob (Job addMe, int index) {
        int projectIDNumber = addMe.getProjectID();
        //int index = projectIDNumber - 1;
        if (DEBUG) System.out.println("Adding new Job!");
        Project updateProject;
        if (DEBUG) System.out.println("Vector size:" + theVector.size() + " ProjectID:" + projectIDNumber + " Index:" + index);
        if (theVector.size() <= index) {
            theVector.setSize (projectIDNumber);
        }
    }
```java
Project imNew = new Project (projectIdNumber);  
theVector.set (index, imNew);  
if (DEBUG) System.out.println ("Adding new Project!");
}

updateProject = (Project) theVector.elementAt (index);  
if (updateProject == null)  
    updateProject = new Project (projectIdNumber);  
    updateProject.addJob (addMe);  
    theVector.set (index, updateProject);

public void sortByDate () {  
    Vector temp = new Vector();
    Date earliestDate = new Date();  
    Date latestDate = new Date();  
    Date test;
    boolean[] assigned = new boolean[theVector.size()];  
    int numAssigned = 0;
    int index = 0;
    for (int x = 0; x < assigned.length; x++)  
        assigned[x] = false;
    //get latest date
    for (int x = 0; x < theVector.size(); x++) {
        test = ((Project) theVector.elementAt (x)).getDate();
        if (latestDate.before (test)) {
            latestDate = test;
        }
    }
    if (DEBUG) System.out.println ("LATEST:" + latestDate);
    earliestDate = latestDate;
    Project tempP;
    while (numAssigned < theVector.size()) {  
        //get earliest date not yet assigned
        for (int i = 0; i < theVector.size(); i++) {
            test = ((Project) theVector.elementAt (i)).getDate();
            if ((earliestDate.after (test)) || (earliestDate.equals (test)) && !assigned[i]) {
                earliestDate = test;
                index = i;
            }
        }
        tempP = (Project) theVector.elementAt (index);  
        //tempP.setProjectID(index+1);  
        temp.add(tempP);
        assigned[index] = true;
        earliestDate = latestDate;
        numAssigned++;
    }  
    //end while
    theVector = new Vector (temp);
    //printSortedOates();
}

public int getNumberOfJobs () {  
    int number = 0;
    Project temp;
    for (int x = 0; x < theVector.size(); x++) {
```
temp=(Project)theVector.elementAt(x);
number+=temp.getNumberOfJobs();
}
return number;

public void printSortedDates(){
    Project temp;
    for(int x=0;x<theVector.size();x++) {
        temp=(Project)theVector.elementAt(x);
        System.out.println("Index:"+x+" ID:"+temp.projectID()+" Date:"+temp.getDate());
    }
}

public void printList(){
    for(int x=0;x<size();x++) {
        System.out.println("Index:"+x+" ProjectID:"+getProjectByIndex(x).projectID());
    }
}

public Project getProjectByIndex(int index){
    return (Project)theVector.elementAt(index);
}

public Project getProjectByID(int IDNumber){
    for(int x=0;x<size();x++) {
        if(((Project)theVector.elementAt(x)).projectID()==IDNumber) {
            return ((Project)theVector.elementAt(x));
        }
    }
    return null;
}

public void addProject(Project addMe){
    theVector.add(addMe);
}

public int size(){
    return theVector.size();
}
Appendix A

Optimization Program Code

Project.java
package ms_senior_design;

/**
 * Title: MS Senior Design
 * Description: 
 * Copyright: Copyright (c) 2001
 * Company: 
 * @author
 * @version 1.0
 */
import java.util.Date;

public class Project {
    private JobSet projectJobSet;
    private int projectId;
    boolean attributesSet = false;
    private Date projectDate;
    private String projectName;
    private double projectLength;
    private Date startDate;
    private Date endDate;

    public Project() {
        projectJobSet = new JobSet();
    }

    public void setProjectLength(double pLength) {
        projectLength = pLength;
    }

    public double getProjectLength() {
        return projectLength;
    }

    public Date getStartDate() {
        return startDate;
    }

    public void setStartDate(Date sDate) {
        startDate = sDate;
    }

    public Date getEndDate() {
        return endDate;
    }

    public void setEndDate(Date eDate) {
        endDate = eDate;
    }

    public Date getDate() {
        return projectDate;
    }

    public int projectId() {
        return projectId;
    }
}
public void setProjectID (int set) {
    //projectId=set;
}

public String projectName () {
    return projectName;
}

public Project (int projectID) {
    projectJobSet = new JobSet();
    this.projectID = projectID;
}

public JobSet getJobSet () {
    return projectJobSet;
}

public int getNumberOfJobs () {
    return projectJobSet.size();
}

public void addJob (Job addMe) {
    projectJobSet .addJob (addMe);
}

public void addModule (Module addMe) {
    if(!attributesSet) {
        this.projectName = addMe.getProjectName();
        this.projectDate = addMe.getDate();
        attributesSet = true;
    }
    int maxNumberOfJobs = 50;
    int maxJobSize = 50;
    if(addMe.getNumLib() > 0) {
        numberOfMods += addMe.getNumLib();
        while (numberOfMods > maxJobSize) {
            Job newJob = new Job (addMe.getProjectID(), addMe.getProjectName(), addMe.getDate(), addMe.getModuleName(),
            addMe.getModType(), 1, maxJobSize);
            projectJobSet .addJob (newJob);
            numberOfMods -= maxJobSize;
        }
    }
    if(numberOfMods > 0) {
        Job newJob = new Job(addMe.getProjectID(), addMe.getProjectName(), addMe.getDate(), addMe.getModuleName(),
            addMe.getModType(), 1, numberOfMods);
        projectJobSet .addJob (newJob);
    }
    if(addMe.getNumNonLib() > 0) {
        numberOfMods += addMe.getNumNonLib();
        while (numberOfMods > maxJobSize) {
            Job newJob = new Job (addMe.getProjectID(), addMe.getProjectName(), addMe.getDate(), addMe.getModuleName(),
            addMe.getModType(), 2, maxJobSize);
            projectJobSet .addJob (newJob);
            numberOfMods -= maxJobSize;
        }
    }
    if(numberOfMods > 0) {
Job newJob=new Job(addMe.getProjectID(), addMe.getProjectName(), addMe.getDate(), addMe.getModName(),
  addMe.getModType(), 2, numberOfMods);
  projectJobSet.addJob(newJob);
}

if(addMe.getNumCopy()>0){
  numberOfMods = addMe.getNumCopy();
  while(numberOfMods >maxJobSize ) {
    Job newJob=new Job(addMe.getProjectID(), addMe.getProjectName(), addMe.getDate(), addMe.getModName(),
      addMe.getModType(), 3, maxJobSize);
    projectJobSet.addJob(newJob);
    numberOfMods = numberOfMods =maxJobSize;
  }
}
if(numberOfMods >0) {
  Job newJob=new Job(addMe.getProjectID(), addMe.getProjectName(), addMe.getDate(), addMe.getModName(),
    addMe.getModType(), 3, numberOfMods);
  projectJobSet.addJob(newJob);
}

public void printlnfo (){  
  Job tempJob;
  int numberOfJobs =0;
  for(int x=1; x<3; x++){
    for(int y=1; y<4; y++){
      numberOfJobs =0;
      for(int index=0; index<projectJobSet.size(); index++){
        tempJob = projectJobSet.getJobByIndex(index);
        if(tempJob.getModType()==y && tempJob.getJobType()==x) {
          numberOfJobs = numberOfJobs + tempJob.getNumberOfModules();
        }
      }
      System.out.println("ProjectID:" +projectID +" ModType:" +y +" JobType:" +x +" Number:" +numberOfJobs);
    }
  }
}
Appendix A

Optimization Program Code

Module.java
package ms_senior_design;

/**
 * Title: MS Senior Design
 * Description:
 * Copyright: Copyright (c) 2001
 * Company:
 * @author Davis W. Peden
 * @version 1.0
 */

import java.util.Date;

public class Module {
    private String projectName;
    private Date date;
    private String modName;
    private int projectId;
    private int modType;
    private int jobType;
    private int totalNum;
    private double area;
    private int numSpecd;
    private int numFrozen;
    private int numLaidOut;
    private int numDocd;
    private int numTCTVerify;
    private int numCustSignoff;
    private int numLib;
    private int numNonLib;
    private int numCopy;
    private boolean assigned = false;
    private JobSet moduleJobSet;

    public Module(int projectId, String pName, Date dt, String mName, int typ,
                  int tNum, double ara, int nSpecd, int nFrozen, int nLaidOut,
                  int nDocd, int nTCTVerify, int nCustSignoff, int nLib, int nNonLib,
                  int nCopy) {
        moduleJobSet = new JobSet();
        this.projectID = projectId;
        this.projectName = pName;
        this.date = dt;
        this.modName = mName;
        this.modType = typ;
        this.totalNum = tNum;
        this.area = ara;
        this.numSpecd = nSpecd;
        this.numFrozen = nFrozen;
        this.numLaidOut = nLaidOut;
        this.numDocd = nDocd;
        this.numTCTVerify = nTCTVerify;
        this.numCustSignoff = nCustSignoff;
        this.numLib = nLib;
        this.numNonLib = nNonLib;
        this.numCopy = nCopy;
    }

    if (numLib > 0)
Job newJob=new Job (projectName, date, modName, modType, 1, numLib);
  moduleJobSet.addJob(newJob);
}
if (numNonLib>0){
  Job newJob=new Job (projectName, date, modName, modType, 2, numNonLib);
  moduleJobSet.addJob(newJob);
}
if (numCopy>0){
  Job newJob=new Job (projectName, date, modName, modType, 3, numCopy);
  moduleJobSet.addJob(newJob);
}

="/end Job constructor

public String toString () {
  StringBuffer sb=new StringBuffer();
  sb.append("ModuleName: "+getProjectName());
  sb.append("\n");
  sb.append("Date: "+getDate());
  sb.append("\n");
  sb.append("ModName: "+getModName());
  sb.append("\n");
  sb.append("Type: "+getModType());
  sb.append("\n");
  return sb.toString();
}

public int getProjectID () {
  return projectID;
}

public Date getProjectDate () {
  return date;
}

public String getProjectName () {
  return projectName;
}

public Date getDate () {
  return date;
}

public String getModName (){
  return modName;
}

public int getModType (){
  return modType;
}

public int getTotalNum (){
  return totalNum;
}

public double getArea (){
  return area;
public int getNumSpecd()
    return numSpecd;

public int getNumFrozen()
    return numFrozen;

public int getNumLaidOut()
    return numLaidOut;

public int getNumDocd()
    return numDcdd;

public int getNumTCTVerify()
    return numTCTVerify;

public int getNumCustSignoff()
    return numCustSignoff;

public int getNumLib()
    return numLib;

public int getNumNonLib()
    return numNonLib;

public int getNumCopy()
    return numCopy;

public void setAssigned()
    assigned=true;

public int getNumberOfModules()
    switch (jobType) {
        case 1:
            return numLib;
        case 2:
            return numNonLib;
        case 3:
            return numCopy;
    }
    return -1;

public int getJobType()
    return jobType;
} //end Job class
Appendix A

Optimization Program Code

Job.java
package ms_senior_design;

/**
 * Title:        MS Senior Design
 * Description:
 * Copyright:    Copyright (c) 2001
 * Company:
 * @author Davis W. Peden
 * @version 1.0
 */

import java.util.Date;

public class Job {
    private String projectName;
    private Date date;
    private String modName;
    private int modType;
    private int totalNum;
    private double area;
    private int projectId;
    private int numSpecd;
    private int numFrozen;
    private int numLaidOut;
    private int numDocd;
    private int numTCTVerify;
    private int numCustSignoff;
    private int numLib;
    private int numNonLib;
    private int numCopy;
    private int jobType;
    private boolean assigned=false;
    private int numberOfModules;
    private int workerID;

    public Job(int projectId, String theName, Date theDate, String modName,
        int modType, int jobType, int numberOfModules ){
        this.projectId=projectId;
        this.projectName=theName;
        this.date=theDate;
        this.modName=modName;
        this.modType=modType;
        this.jobType=jobType;
        this.numberOfModules=numberOfModules;
    }

    public String toString () {
        StringBuffer sb=new StringBuffer();
        sb.append("Date:	 + getDate());
        sb.append( "
" );
        sb.append("ModName:	 " + getModName());
        sb.append("\n");
        sb.append("Module Type:	 + getModType());
        sb.append("\n");
        sb.append("Job Type:	 + getJobType()");
        sb.append("\n");
    }
sb.append("Number: 	 + getNumberOfModules ()");
sb.append("\n");
return sb.toString();

public int getProjectID (){  
    return projectID;
}

public String getProjectName (){  
    return projectName;
}

public Date getDate (){  
    return date;
}

public String getModuleName (){  
    return modName;
}

public int getModType (){  
    return modType;
}

public boolean isAssigned (){  
    return assigned;
}

public void setAssigned (){  
    assigned=true;
}

public void setUnassigned (){  
    assigned=false;
}

public int getNumberOfModules (){  
    return numberOfModules;
}

public int getJobType (){  
    return jobType;
}

public void setWorkerID (int idNumber){  
    workerID=idNumber;
}

public int getWorkerID (){  
    return workerID;
}

@end Job class
Appendix A

Optimization Program Code

JobSet.java
package ms_senior_design;

import java.util.Vector;
import java.util.Date;

public class JobSet {
    protected Vector theVector;

    public JobSet() {
        theVector = new Vector();
    }

    public JobSet(JobSet JSet) {
        theVector = new Vector(JSet.theVector);
    }

    public Job getJob(int employeeId) {
        for (int x = 0; x < employees.size(); x++) {
            if (((Employee) employees.elementAt(x)).getEmployeeId() == employeeId) {
                return (Employee) employees.elementAt(x);
            }
        }
        System.out.println("Employee ID: " + employeeId + " not found!!");
        return null;
    }

    public void clearAssignments() {
        for (int x = 0; x < size(); x++) {
            getJobByIndex(x).setUnassigned();
        }
    }

    public boolean moreJobs() {
        Job tempJob;
        for (int x = 0; x < size(); x++) {
            if (!this.getJobByIndex(x).isAssigned()) {
                return true;
            }
        }
        return false;
    }

    public void sortByDate() {
        Vector temp = new Vector();
        Date earliestDate = new Date();
        Date latestDate = new Date();
        Date test;
        boolean[] assigned = new boolean[theVector.size()];
```java
int numAssigned = 0;
int index = 0;

for (int x = 0; x < assigned.length; x++)
    assigned[x] = false;
// get latest date
for (int x = 0; x < theVector.size(); x++)
    test = ((Job) theVector.elementAt(x)).getDate();
if (latestDate.before(test))
    latestDate = test;
}

System.out.println("LATEST:" + latestDate);
earliestDate = latestDate;

while (numAssigned < theVector.size()) {
    // get earliest date not yet assigned
    for (int i = 0; i < theVector.size(); i++)
        test = ((Job) theVector.elementAt(i)).getDate();
    if ((earliestDate.after(test) || earliestDate.equals(test)) && ! assigned[i])
        earliestDate = test;
        index = i;
    }
    temp.add(theVector.elementAt(index));
    assigned[index] = true;
    earliestDate = latestDate;
    numAssigned ++;
} // end while
theVector = new Vector(temp);
// printSortedDates();

public void printSortedDates () {
    Job temp;
    for (int x = 0; x < theVector.size(); x++)
        temp = (Job) theVector.elementAt(x);
    System.out.println("Index:" + x + " Date:" + temp.getDate());
}

public void printList () {
}

public Job getJobByIndex(int index) {
    return (Job) theVector.elementAt(index);
}

public void addJob(Job addMe) {
    theVector.add(addMe);
}

public int size () {
    return theVector.size();
}

public void clearJobSet () {
```
theVector.removeAllElements();


Appendix A

Optimization Program Code

EmployeeSet.java
package ms_senior_design;

import java.util.Vector;

/**
 * Title: MS Senior Design
 * Description:
 * Copyright: Copyright (c) 2001
 * Company:
 * @author
 * @version 1.0
 */

public class EmployeeSet {
    private Vector employees;
    // private HashMap employeeMap;

    public EmployeeSet() {
    }

    public EmployeeSet(Vector empList) {
        employees = new Vector(empList);
        //employeeMap=new HashMap();
        for (int x = 0; x < employees.size(); x++) {
            employees.elementAt(x);  
            System.out.println("Employee ID: " + employeeId + " not found!!");
            return null;
        }
    }

    public Employee getEmployeeBylD(Integer employeeId) {
        for (int x = 0; x < employees.size(); x++) {
            if (((Employee) employees.elementAt(x)).getEmployeeID() == employeeId) {
                return (Employee) employees.elementAt(x);
            }
        }
        System.out.println("Employee ID: " + employeeId + " not found!!");
        return null;
    }

    public Employee getEmployeeBylndex(Integer index) {
        return (Employee) employees.elementAt(index);
    }

    public void addEmployee(Employee addMe) {
        employees.add(addMe);
    }

    public int size() {
        return employees.size();
    }

    public void resetAvailability() {
        for (int x = 0; x < employees.size(); x++) {
            getEmployeeBylndex(x).setAvailable(false);
        }
    }

    public void clearJobLists() {
        for (int x = 0; x < employees.size(); x++) {
            System.out.println(employees.elementAt(x));
        }
    }
}
getEmployeeByIndex (x).clearJobList ();
}

public boolean anyAvailable (int JType, int MType){
    for(int x=0;x<employees.size();x++) {
        if(getEmployeeByIndex (x).isAvailable ()&&getEmployeeByIndex (x).hasSkill (JType,MType)){
            return true;
        } //end if
    } //end for
    return false;
} //end anyAvailable

}
Appendix A

Optimization Program Code

Employee.java
package ms_senior_design;

import java.util.Vector;

public class Employee {
    private int employeeId;
    private boolean available;
    private Vector skillSet;
    boolean DEBUG=false;
    private JobSet workList;

    public Employee(int id) {
        this.employeeId = id;
        this.available = true;
        this.skillSet = new Vector();
        this.workList = new JobSet();
    } //end Employee Constructor

    public int getEmployeeID() {
        return employeeId;
    }

    public double get_productivity(int jobType, int modType) {
        if (DEBUG) System.out.println("Getting productivity for: ID:" +getEmployeeID ()+" Job:"+jobType+" Mod:"+modType);
        Skill theSkill;
        if(hasSkill(jobType,modType)) {
            for(int x=0;x<skillSet.size();x++) {
                theSkill=(Skill)skillSet.elementAt (x);
                if(theSkill.getModuleId()==modType &&theSkill.getJobTypeID()==jobType)
                    if (DEBUG) System.out.println("Productivity:" +theSkill.getProductivity ());
                    return theSkill.getProductivity ();
            }
        }
        return -1;
    }

    public boolean hasSkill(int jobType, int modType) {
        Skill theSkill;
        for(int x=0;x<skillSet.size();x++) {
            theSkill=(Skill)skillSet.elementAt (x);
            if(theSkill.getModuleId()==modType &&theSkill.getJobTypeID()==jobType)
                return true;
        }
        return false;
    }

    public String toString () {
    }
}
StringBuffer sb = new StringBuffer("\n");
sb.append("\n");
sb.append("------------------------" );
sb.append("\n");
sb.append("EmpId: " + getEmployeeID());
sb.append("\n");
sb.append("Jobs: " + numberOfJobs());
sb.append("\n");
sb.append("Available: " + isAvailable());
sb.append("\n");
for(int i=0; i<getSkillSet().size(); i++) {
    Skill theSkill = (Skill)(getSkillSet().get(i));
    sb.append(theSkill.toString());
    sb.append("\n");
}
return new String(sb.toString());

public void setBusy() {
    if (DEBUG) System.out.println("I have been assigned! MY ID:" + getEmployeeID());
    available = false;
}

public void printJobList() {
    Job tempJob;
    double days, prod = 1.0;
    int modsToDo;
    for(int x = 0; x<workList.size(); x++){
        tempJob = workList.getJobByIndex(x);
        System.out.println(tempJob);
        prod = getProductivity(tempJob.getJobType(), tempJob.getModType());
        modsToDo = tempJob.getNumberofModules();
        days = (double) modsToDo / prod;
        System.out.println("Days to complete: " + days);
        System.out.println("-------------------------------------" );
    }
}

public void setAvailable() {
    available = true;
}

public boolean isAvailable() {
    return available;
}

public void addJob(Job newJob) {
    workList.addJob(newJob);
}

public int numberOfJobs() {
    return workList.size();
}

public Vector getSkillSet() {
    return skillSet;
public void addSkill(Skill skill) {
    skillSet.add(skill);
}  //end add skillset

public JobSet getJobList() {
    return workList;
}  //end getWorkList

public void clearJobList() {
    workList.clearJobSet();
}

}  //end class Employee
Appendix A

Optimization Program Code

Skill.java
package ms_senior_design;
import java.lang.System;

public class Skill {
    private int moduleID;
    private double productivity;
    private int jobTypeID;

    public Skill(int mid, int jTypeID, double prod) {
        this.moduleID = mid;
        this.productivity = prod;
        this.jobTypeID = jTypeID;
    }

    public String toString() {
        return new String("Module: "+getModuleID()+" Productivity: "+getProductivity());
    }

    public int getModuleID() {
        return moduleID;
    }

    public int getJobTypeID() {
        return jobTypeID;
    }

    public double getProductivity() {
        return productivity;
    }
}
Appendix B

Result Output without Project Breakup
## Shortest Processing Time Optimization Report

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Project Start</th>
<th>Project End</th>
<th>Module Name</th>
<th>Employee Name</th>
<th>Job Type</th>
<th>Component Type</th>
<th>Units per day</th>
<th>Number of Pieces</th>
<th>Days to Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>181</td>
<td>18.1</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>325</td>
<td>32.5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>B. Analog</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>155</td>
<td>15.5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>B. Analog</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>140</td>
<td>14</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>C. Isolation</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>C. Isolation</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>12</td>
<td>1.2</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>C. Isolation</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>31</td>
<td>3.1</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>D. BEOL</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>27</td>
<td>2.7</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>D. BEOL</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>282</td>
<td>28.2</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>E. Reliability</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>204</td>
<td>20.4</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>G. SPICE</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>29</td>
<td>2.9</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>A. Resistors</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>33</td>
<td>3.3</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>A. Resistors</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>19</td>
<td>1.9</td>
</tr>
</tbody>
</table>
Shortest Processing Time Optimization Report

<table>
<thead>
<tr>
<th>Project Nam</th>
<th>Project Star</th>
<th>Project En</th>
<th>Module Nam</th>
<th>Employee Nam</th>
<th>Job Typ</th>
<th>Component Type</th>
<th>Units per day</th>
<th>Number of Pieces</th>
<th>Days to Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>B. Capacitors</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>41</td>
<td>4.1</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>C. PNP</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>7</td>
<td>0.7</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>C. PNP</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>75</td>
<td>7.5</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>D. CMOS</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>88</td>
<td>8.8</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>D. CMOS</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>E. Inductors</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>33</td>
<td>3.3</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>E. Inductors</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>18</td>
<td>1.8</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>E. Inductors</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>4</td>
<td>0.4</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>F. Isolation</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>F. Isolation</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>22</td>
<td>2.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>F. Isolation</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>2</td>
<td>0.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>G. ESD</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>ESD</td>
<td>10</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>G. ESD</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>ESD</td>
<td>10</td>
<td>29</td>
<td>2.9</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>G. ESD</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>ESD</td>
<td>10</td>
<td>11</td>
<td>1.1</td>
</tr>
</tbody>
</table>
## Shortest Processing Time Optimization Report

<table>
<thead>
<tr>
<th>Project Nam</th>
<th>Project Start</th>
<th>Project End</th>
<th>Module Nam</th>
<th>Employee Nam</th>
<th>Job Typ</th>
<th>Component Type</th>
<th>Units per day</th>
<th>Number of Pieces</th>
<th>Days to Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>H. Back End (BEO)</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>7</td>
<td>0.7</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>H. Back End (BEO)</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>46</td>
<td>4.6</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>J. MOS Varactors</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>2</td>
<td>0.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>J. MOS Varactors</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>2</td>
<td>0.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>J. MOS Varactors</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>44</td>
<td>4.4</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>K. Schottky and Va</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>22</td>
<td>2.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>K. Schottky and Va</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>21</td>
<td>2.1</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>M. NPNs</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>24</td>
<td>2.4</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>N. PMC</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>62</td>
<td>6.2</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>B. MOSFET - SPIC</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>140</td>
<td>14</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>B. MOSFET - SPIC</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>20</td>
<td>2</td>
</tr>
</tbody>
</table>
## Shortest Processing Time Optimization Report

<table>
<thead>
<tr>
<th>Project Nam</th>
<th>Project Star</th>
<th>Module Nam</th>
<th>Employee Nam</th>
<th>Job Typ</th>
<th>Component Type</th>
<th>Units per day</th>
<th>Number of Pieces</th>
<th>Days to Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>C. BEOL</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>124</td>
<td>12.4</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>C. BEOL</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>23</td>
<td>2.3</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>D. Reliability</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>27</td>
<td>2.7</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>F. Passives</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>153</td>
<td>15.3</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>F. Passives</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>40</td>
<td>4</td>
</tr>
</tbody>
</table>
## Longest Processing Time Optimization Report

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Project Start</th>
<th>Project End</th>
<th>Module Name</th>
<th>Employee Name</th>
<th>Job Type</th>
<th>Component Type</th>
<th>Units per Day</th>
<th>Number of Pieces</th>
<th>Days to Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>181</td>
<td>18.1</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>325</td>
<td>32.5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>B. Analog</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>155</td>
<td>15.5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>B. Analog</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>140</td>
<td>14</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>C. Isolation</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>C. Isolation</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>12</td>
<td>1.2</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>C. Isolation</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>31</td>
<td>3.1</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>D. BEOL</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>27</td>
<td>2.7</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>D. BEOL</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>282</td>
<td>28.2</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>E. Reliability</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>204</td>
<td>20.4</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>G. SPICE</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>29</td>
<td>2.9</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>A. Resistors</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>33</td>
<td>3.3</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>A. Resistors</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>19</td>
<td>1.9</td>
</tr>
</tbody>
</table>
# Longest Processing Time Optimization Report

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Project Start</th>
<th>Project End</th>
<th>Module Name</th>
<th>Employee Name</th>
<th>Job Type</th>
<th>Component Type</th>
<th>Units per Day</th>
<th>Number of Pieces</th>
<th>Days to Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>B. Capacitors</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>41</td>
<td>4.1</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>C. PNP</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>7</td>
<td>0.7</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>C. PNP</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>75</td>
<td>7.5</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>D. CMOS</td>
<td>West, Jeffrey</td>
<td>STD</td>
<td>STD</td>
<td>10</td>
<td>88</td>
<td>8.8</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>D. CMOS</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>E. Inductors</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>33</td>
<td>3.3</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>E. Inductors</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>18</td>
<td>1.8</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>E. Inductors</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>4</td>
<td>0.4</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>F. Isolation</td>
<td>West, Jeffrey</td>
<td>STD</td>
<td>STD</td>
<td>10</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>F. Isolation</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>22</td>
<td>2.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>F. Isolation</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>2</td>
<td>0.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>G. ESD</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>ESD</td>
<td>10</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>G. ESD</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>ESD</td>
<td>10</td>
<td>29</td>
<td>2.9</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>G. ESD</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>ESD</td>
<td>10</td>
<td>11</td>
<td>1.1</td>
</tr>
</tbody>
</table>
# Longest Processing Time Optimization Report

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Start Date</th>
<th>End Date</th>
<th>Module Name</th>
<th>Employee Name</th>
<th>Job Type</th>
<th>Component Type</th>
<th>Units per day</th>
<th>Pieces</th>
<th>Days to Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>H. Back End (BEO)</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>7</td>
<td>0.7</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>H. Back End (BEO)</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>46</td>
<td>4.6</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>J. MOS Varactors</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>2</td>
<td>0.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>J. MOS Varactors</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>2</td>
<td>0.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>J. MOS Varactors</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>44</td>
<td>4.4</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>K. Schottky and Va</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>22</td>
<td>2.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>K. Schottky and Va</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>21</td>
<td>2.1</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>M. NPNs</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>24</td>
<td>2.4</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>N. PMC</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>62</td>
<td>6.2</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>B. MOSFET - SPIC</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>140</td>
<td>14</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>B. MOSFET - SPIC</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>20</td>
<td>2</td>
</tr>
</tbody>
</table>
## Longest Processing Time Optimization Report

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Project Start</th>
<th>Project End</th>
<th>Module Name</th>
<th>Employee Name</th>
<th>Job Type</th>
<th>Component Type</th>
<th>Units per day</th>
<th>Number of Pieces</th>
<th>Days to Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>C. BEOL</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>124</td>
<td>12.4</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>C. BEOL</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>23</td>
<td>2.3</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>D. Reliability</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>27</td>
<td>2.7</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>F. Passives</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>153</td>
<td>15.3</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>F. Passives</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>40</td>
<td>4</td>
</tr>
</tbody>
</table>
Appendix C

Result Output with Project Breakup
# Shortest Processing Time Optimization Report

<table>
<thead>
<tr>
<th>Project Nam</th>
<th>Project Star</th>
<th>Project En</th>
<th>Module Nam</th>
<th>Employee Nam</th>
<th>Job Typ</th>
<th>Component Type</th>
<th>Units per day</th>
<th>Number of Pieces</th>
<th>Days to Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>25</td>
<td>2.5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>31</td>
<td>3.1</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>B. Analog</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>B. Analog</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
</tbody>
</table>
## Shortest Processing Time Optimization Report

<table>
<thead>
<tr>
<th>Project Nam</th>
<th>Project Star</th>
<th>Project End</th>
<th>Module Nam</th>
<th>Employee Nam</th>
<th>Job Type</th>
<th>Component Type</th>
<th>Units per day</th>
<th>Number of Pieces</th>
<th>Days to Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>B. Analog</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>5</td>
<td>0.5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>B. Analog</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>B. Analog</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>B. Analog</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>C. Isolation</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>12</td>
<td>1.2</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>C. Isolation</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>31</td>
<td>3.1</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>C. Isolation</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>D. BEOL</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>D. BEOL</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>32</td>
<td>3.2</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>D. BEOL</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>D. BEOL</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>D. BEOL</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>27</td>
<td>2.7</td>
</tr>
</tbody>
</table>
## Shortest Processing Time Optimization Report

<table>
<thead>
<tr>
<th>Project Nam</th>
<th>Project Star</th>
<th>Project End</th>
<th>Module Nam</th>
<th>Employee Nam</th>
<th>Job Typ</th>
<th>Component Type</th>
<th>Units per day</th>
<th>Number of Pieces</th>
<th>Days to Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>D. BEOL</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>E. Reliability</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>4</td>
<td>0.4</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>E. Reliability</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>E. Reliability</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>E. Reliability</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>E. Reliability</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>29</td>
<td>2.9</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>A. Resistors</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>33</td>
<td>3.3</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>A. Resistors</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>19</td>
<td>1.9</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>B. Capacitors</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>41</td>
<td>4.1</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>C. PNP</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>C. PNP</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>25</td>
<td>2.5</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>C. PNP</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>7</td>
<td>0.7</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>D. CMOS</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project Name</td>
<td>Project Start</td>
<td>Project End</td>
<td>Module Name</td>
<td>Employee Name</td>
<td>Job Type</td>
<td>Component Type</td>
<td>Units per Day</td>
<td>Number of Pieces</td>
<td>Days to Complete</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>-------------</td>
<td>-------------</td>
<td>--------------</td>
<td>----------</td>
<td>----------------</td>
<td>---------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/1/2001</td>
<td>3/15/2002</td>
<td>D. CMOS</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/1/2001</td>
<td>3/15/2002</td>
<td>D. CMOS</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>38</td>
<td>3.8</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/1/2001</td>
<td>3/15/2002</td>
<td>E. Inductors</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>18</td>
<td>1.8</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/1/2001</td>
<td>3/15/2002</td>
<td>E. Inductors</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>33</td>
<td>3.3</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/1/2001</td>
<td>3/15/2002</td>
<td>E. Inductors</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>4</td>
<td>0.4</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/1/2001</td>
<td>3/15/2002</td>
<td>F. Isolation</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/1/2001</td>
<td>3/15/2002</td>
<td>F. Isolation</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>2</td>
<td>0.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/1/2001</td>
<td>3/15/2002</td>
<td>F. Isolation</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>22</td>
<td>2.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/1/2001</td>
<td>3/15/2002</td>
<td>G. ESD</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>ESD</td>
<td>10</td>
<td>11</td>
<td>1.1</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/1/2001</td>
<td>3/15/2002</td>
<td>G. ESD</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>ESD</td>
<td>10</td>
<td>29</td>
<td>2.9</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/1/2001</td>
<td>3/15/2002</td>
<td>G. ESD</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>ESD</td>
<td>10</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/1/2001</td>
<td>3/15/2002</td>
<td>H. Back End</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>46</td>
<td>4.6</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/1/2001</td>
<td>3/15/2002</td>
<td>H. Back End</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>7</td>
<td>0.7</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/1/2001</td>
<td>3/15/2002</td>
<td>J. MOS Varactors</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>44</td>
<td>4.4</td>
</tr>
</tbody>
</table>
# Shortest Processing Time Optimization Report

<table>
<thead>
<tr>
<th>Project Nam</th>
<th>Project Start</th>
<th>Project End</th>
<th>Module Name</th>
<th>Employee Name</th>
<th>Job Type</th>
<th>Component Type</th>
<th>Units per day</th>
<th>Number of Pieces</th>
<th>Days to Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>J. MOS Varactors</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>2</td>
<td>0.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>J. MOS Varactors</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>2</td>
<td>0.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>K. Schottky and Va</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>21</td>
<td>2.1</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>K. Schottky and Va</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>22</td>
<td>2.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>M. NPNs</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>27</td>
<td>2.7</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>M. NPNs</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>M. NPNs</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>M. NPNs</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>24</td>
<td>2.4</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>M. NPNs</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>12</td>
<td>1.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>M. NPNs</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
</tbody>
</table>
# Shortest Processing Time Optimization Report

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Project Start</th>
<th>Project End</th>
<th>Module Name</th>
<th>Employee Name</th>
<th>Job Type</th>
<th>Component Type</th>
<th>Units per Day</th>
<th>Number of Pieces</th>
<th>Days to Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>B. MOSFET - SPIC</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>40</td>
<td>4</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>B. MOSFET - SPIC</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>B. MOSFET - SPIC</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>C. BEOL</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>C. BEOL</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>C. BEOL</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>24</td>
<td>2.4</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>C. BEOL</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>23</td>
<td>2.3</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>D. Reliability</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>27</td>
<td>2.7</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>F. Passives</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>3</td>
<td>0.3</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>F. Passives</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>40</td>
<td>4</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>F. Passives</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>F. Passives</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>F. Passives</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
</tbody>
</table>
## Longest Processing Time Optimization Report

<table>
<thead>
<tr>
<th>Project Nam</th>
<th>Project Start</th>
<th>Project End</th>
<th>Module Nam</th>
<th>Employee Nam</th>
<th>Job Type</th>
<th>Component Type</th>
<th>Units per day</th>
<th>Number of Pieces</th>
<th>Days to Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>25</td>
<td>2.5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>31</td>
<td>3.1</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>A. Core CMOS</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>B. Analog</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>B. Analog</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
</tbody>
</table>
## Longest Processing Time Optimization Report

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Project Start</th>
<th>Project End</th>
<th>Module Name</th>
<th>Employee Name</th>
<th>Job Type</th>
<th>Component Type</th>
<th>Units per Day</th>
<th>Number of Pieces</th>
<th>Days to Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>B. Analog</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>5</td>
<td>0.5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>B. Analog</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>B. Analog</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>B. Analog</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>40</td>
<td>4</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>B. Analog</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>C. Isolation</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>12</td>
<td>1.2</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>C. Isolation</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>31</td>
<td>3.1</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>C. Isolation</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>D. BEOL</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>D. BEOL</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>32</td>
<td>3.2</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>D. BEOL</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>D. BEOL</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>D. BEOL</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>D. BEOL</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>27</td>
<td>2.7</td>
</tr>
</tbody>
</table>
## Longest Processing Time Optimization Report

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Start Date</th>
<th>End Date</th>
<th>Module</th>
<th>Employee</th>
<th>Job Type</th>
<th>Component Type</th>
<th>Units per Day</th>
<th>Number of Pieces</th>
<th>Days to Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>D. BEOL</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>E. Reliability</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>4</td>
<td>0.4</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>E. Reliability</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>E. Reliability</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>E. Reliability</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 1</td>
<td>5/10/2001</td>
<td>11/21/2001</td>
<td>G. SPICE</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>29</td>
<td>2.9</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>A. Resistors</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>33</td>
<td>3.3</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>A. Resistors</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>19</td>
<td>1.9</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>B. Capacitors</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>41</td>
<td>4.1</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>C. PNP</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>C. PNP</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>25</td>
<td>2.5</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>C. PNP</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>7</td>
<td>0.7</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>D. CMOS</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
</tbody>
</table>
## Longest Processing Time Optimization Report

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Project Start</th>
<th>Project End</th>
<th>Module Name</th>
<th>Employee Name</th>
<th>Job Type</th>
<th>Component Type</th>
<th>Units per Day</th>
<th>Number of Pieces</th>
<th>Days to Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>D. CMOS</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>D. CMOS</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>38</td>
<td>3.8</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>E. Inductors</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>18</td>
<td>1.8</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>E. Inductors</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>33</td>
<td>3.3</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>E. Inductors</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>4</td>
<td>0.4</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>F. Isolation</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>F. Isolation</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>2</td>
<td>0.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>F. Isolation</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>22</td>
<td>2.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>G. ESD</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>ESD</td>
<td>10</td>
<td>11</td>
<td>1.1</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>G. ESD</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>ESD</td>
<td>10</td>
<td>29</td>
<td>2.9</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>G. ESD</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>ESD</td>
<td>10</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>H. Back End (BEO)</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>46</td>
<td>4.6</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>H. Back End (BEO)</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>7</td>
<td>0.7</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>J. MOS Varactors</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>44</td>
<td>4.4</td>
</tr>
<tr>
<td>Project Name</td>
<td>Project Start</td>
<td>Project End</td>
<td>Module Name</td>
<td>Employee Name</td>
<td>Job Type</td>
<td>Component Type</td>
<td>Units per day</td>
<td>Number of Pieces</td>
<td>Days to Complete</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>-------------</td>
<td>-------------------</td>
<td>---------------</td>
<td>----------</td>
<td>----------------</td>
<td>---------------</td>
<td>------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>J. MOS Varactors</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>2</td>
<td>0.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>J. MOS Varactors</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>2</td>
<td>0.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>K. Schottky and Va</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>21</td>
<td>2.1</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>K. Schottky and Va</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>22</td>
<td>2.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>M. NPNs</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>27</td>
<td>2.7</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>M. NPNs</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>M. NPNs</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>M. NPNs</td>
<td>West, Jeffrey</td>
<td>LIB</td>
<td>STD</td>
<td>10</td>
<td>24</td>
<td>2.4</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>N. PMC</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>12</td>
<td>1.2</td>
</tr>
<tr>
<td>Project 2</td>
<td>11/21/2001</td>
<td>3/15/2002</td>
<td>N. PMC</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
</tbody>
</table>
## Longest Processing Time Optimization Report

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Start Date</th>
<th>End Date</th>
<th>Module Name</th>
<th>Employee Name</th>
<th>Job Type</th>
<th>Component Type</th>
<th>Units per Day</th>
<th>Number of Pieces</th>
<th>Days to Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>B. MOSFET - SPIC</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>40</td>
<td>4</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>B. MOSFET - SPIC</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>B. MOSFET - SPIC</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>B. MOSFET - SPIC</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>C. BEOL</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>C. BEOL</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>C. BEOL</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>24</td>
<td>2.4</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>C. BEOL</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>23</td>
<td>2.3</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>D. Reliability</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>27</td>
<td>2.7</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>F. Passives</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>3</td>
<td>0.3</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>F. Passives</td>
<td>West, Jeffrey</td>
<td>COPY</td>
<td>STD</td>
<td>10</td>
<td>40</td>
<td>4</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>F. Passives</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>F. Passives</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Project 3</td>
<td>3/15/2002</td>
<td>6/21/2002</td>
<td>F. Passives</td>
<td>West, Jeffrey</td>
<td>NONLIB</td>
<td>STD</td>
<td>10</td>
<td>50</td>
<td>5</td>
</tr>
</tbody>
</table>
Appendix D

TestChip Technologies, Inc. Optimization

Presentation
TestChip Technologies, Inc. Optimization

CSE 4395 – MS Senior Design
Professor Olinick
Spring 2001 – SMU
10 May 2001

Monica Gaston
Niki Holland
Jana Lindley
Davis Peden
Jeff West

About TestChip Technologies, Inc.

• Locations: Dallas, Texas
  Austin, Texas
• Founded: 1994 by former TI employees
• Customers: Semiconductor Manufacturers
• Market: $200 billion internationally
• Service: Outsource business for semiconductor testchips
Current Operations

- 10-30 projects
- Each project composed of multiple modules requiring different skill sets
- Modules assigned haphazardly to engineers
- Productivity and project status tracked in multiple Excel spreadsheets

Current Operation Problems

- Productivity data tracked manually
  - no central location for data
  - no method of data reporting
- Projecting project length based on intuition
  - doesn’t scale well
- Random assignments to engineers
  - doesn’t take productivity into account
Project Goals

To provide:
• more accurate methods of assigning engineers to projects
• a method of forecasting project length

Solution Approaches

Integer Linear Programming Model (ILP)
 Modified Job Shop
Linear Program Description

Decision Variable: \( E_{ijk} \)

Objective Function
Max Productivity: \( 2.3E_{111} + 5.6E_{112} + 4.5E_{113} + 1.2E_{114} \)

Subject to:
Availability \( E_{543} = 0 \)

ILP Problems

- Too many variables
- Too expensive
- Not easily maintained or modified
Modified Job Shop

2 Algorithms:

- Modified shortest processing time
  - Employee with best productivity gets job

- Modified longest processing time
  - Employee with worst productivity gets job

Program Functionality

- Access table built from Excel spreadsheets using Visual Basic
- Data extracted from tables using Java DataBase Connectivity (JDBC)
- Algorithms process data to create an optimized ProjectSet
- Projected completion dates are computed
- Results are inserted into Access tables
Algorithm Comparison

Unlimited pieces per job

<table>
<thead>
<tr>
<th></th>
<th>Project 1</th>
<th>Project 2</th>
<th>Project 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>9.313875</td>
<td>2.622535</td>
<td>6.19302</td>
</tr>
<tr>
<td>Min</td>
<td>0.076923</td>
<td>0.153846</td>
<td>1.66667</td>
</tr>
<tr>
<td>Max</td>
<td>27.08333</td>
<td>9.769231</td>
<td>12.75</td>
</tr>
<tr>
<td>Total Days</td>
<td>111.7641</td>
<td>68.1859</td>
<td>55.73718</td>
</tr>
<tr>
<td>Overall Days: 235.69</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Project 1</th>
<th>Project 2</th>
<th>Project 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>10.22985</td>
<td>2.789231</td>
<td>6.235750</td>
</tr>
<tr>
<td>Min</td>
<td>0.153846</td>
<td>1.538462</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>32.5</td>
<td>10.58333</td>
<td>11.76923</td>
</tr>
<tr>
<td>Total Days</td>
<td>122.7462</td>
<td>72</td>
<td>56.12179</td>
</tr>
<tr>
<td>Overall Days: 250.87</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SPF is Faster for Unlimited Pieces Per Job

Algorithm Comparison

Pieces limited to 50 per job

<table>
<thead>
<tr>
<th></th>
<th>Project 1</th>
<th>Project 2</th>
<th>Project 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>3.336484</td>
<td>2.193093</td>
<td>3.104701</td>
</tr>
<tr>
<td>Min</td>
<td>0.063333</td>
<td>0.153846</td>
<td>0.29</td>
</tr>
<tr>
<td>Max</td>
<td>5</td>
<td>4.166667</td>
<td>4.166667</td>
</tr>
<tr>
<td>Total Days</td>
<td>116.7769</td>
<td>67.9859</td>
<td>55.88462</td>
</tr>
<tr>
<td>Overall Days: 240.65</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Project 1</th>
<th>Project 2</th>
<th>Project 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>3.368930</td>
<td>2.319892</td>
<td>3.109687</td>
</tr>
<tr>
<td>Min</td>
<td>0.153846</td>
<td>0.230769</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>5</td>
<td>5</td>
<td>4.166667</td>
</tr>
<tr>
<td>Total Days</td>
<td>117.9128</td>
<td>71.91667</td>
<td>55.97436</td>
</tr>
<tr>
<td>Overall Days: 245.80</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SPF is Faster for 50 Pieces Max. Per Job
Algorithm Comparison

But...

Unlimited Pieces Per Job
SPF Overall Days: 235.69
LPF Overall Days: 250.87

50 Pieces Max Per Job
SPF Overall Days: 240.65
LPF Overall Days: 245.80

The limit of 50 pieces hurt SPF but helped LPF

Conclusion

• Reduced manager work load
• A reliable forecasting method
• Meaningful reports can be produced