Final Exam Scheduler User/Developer Manual

G2. Team Kyz

Kevin Cheek, Yandong Wang, Ziyan Zhou
Department of Computer Science
Rochester Institute of Technology
2/22/2009

Table of Contents

Compiling the Software **3**

Running the Applications **3**

Scripted Method 3

Manual Method 3

Permutation *4*

Simulated Annealing *5*

Genetic Algorithm *7*

Interpreting output **8**

Creating Custom data **9**

Generate Section File 9

Generate Data Set with Custom Problem Size 9

Compiling the Software

In order to compile the software we have included a script for compilation in the **scripts** directory of our project. Simply run "`./compile.bash`" and from the paranoia cluster and the application will compile the project.

The first few lines of the compile script include the path to JDK and other necessary jars. Change can be made to the file to better suit your developing environment.

Running the Applications

Scripted Method

The easiest method for running our application is to change to the scripts directory for our project. Within the scripts direction there is a **benchmark.bash** script. This can be executed by typing "`./benchmark.bash`" when this is done it will run the sequential and parallel versions of the permutation, simulated annealing, and genetic algorithms with from 1 to 16 nodes and on data sets of ranging from 5 to 200. If you want to selectively run tests then use the **run.bash**. This can be executed by calling:

```
./run.bash <algorithm> <K> <N> [JVM-args]
```

The **algorithm** parameter specifies which algorithm you would like to run. The values for it can either be "**Permutation**", "**SA**" for simulated annealing or "**Genetic**" for the genetic algorithm. The **K** parameter is the number of nodes that the program should run on. The value "**0**" indicates that it should run the sequential program. The **N** Parameter is the problem size that the algorithms will run with. The available data sets are in path **data/**. Currently, the valid values are 5, 10, 20, 40, 50, 100, 200, and 2500. **JVM-args** is optional and can be used to define extra parameters for the program. For example, "**-Dfes.ga.gen=100**" determines the number of generations to run in the genetic algorithm. (See detailed parameters in later sections)

All of the outputs from the run or benchmark scripts are created in the performance folder. Each type of algorithm creates its own subfolder in this directory, each corresponding with the algorithm (Permutation, SA, or Genetic) that was run. These subdirectories will then contain a timing file, an output file and generated schedules.

Manual Method

In order to run any of the applications manually data will need to be provided. Either the stock data can be used located in the data/<problem size> directory or custom data can be created. The three files located within each of the problem size subdirectories can then be used as the data files for each of the programs. Before running the application, make sure that the class path has been exported and that it references the parallel java library. Then read the section

pertaining to the application you want to run. In addition to the normal arguments for each programs there are also a number of optional parameters that can be given to the JVM with the `-D<property=<value>>` argument. The following are the common parameters that can be given to all of the algorithms.

Name	Property	Default Value	Description
Weight Variant	fes.v	1.0	Sets the weight variant
Variant Threshold	fes.vt	100.0	Determines if a schedule's slot counter varies too much
Consecutive exam weight	fes.ce	2.0	This determines how much weight is placed by a student on a consecutive exams
Multiple Exam weight	fes.me	5.0	this determines how much weight is place by a student by a student on having more than two exams in a single day.
Friday Exam weight	fes.fe	0.5	Determines how much weight is place by a student on Friday exams.
Simultaneous Exam weight	fes.se	infinity	Determines how much weight is place by a student on having multiple exams at the same time

Permutation

To run the permutation algorithm, follow the directions defined below for each of the versions. The following parameters for the JVM are common to the sequential and parallel versions and can be used by adding `-D<property=<value>>` to the java command.

Name	Property	Default Value	Description
Running time limit	fes.p.time	500	This can be a value between (0, +infinity) in milli-seconds

Permutation Sequential

To run the permutation algorithm sequentially, execute the following command:

```
java FESPermutationSeq <sections-file> <students-file> \
<relationship-file> [schedule-output-file]
```

Section File: This file is required and contains data about class sections. By default these are named “courses.txt”.

Student File: This data file containing the students taking classes. By default this file is named “students.txt”.

Relationship File: This file is required and contains information about what student is taking which course. By default this file is called “relationships.txt”.

Schedule Output File: This is an optional argument to specify the path where generated optimal schedule should be saved. If not specified, the generated schedule will be discarded. This is generally not used when the user is only interested in timing and result statistics.

Permutation Cluster

To run the permutation algorithm in parallel, execute the following command:

```
java -Dpj.np=<K> FESPermutationClu <sections-file> <students-file> \
<relationship-file> [schedule-output-file]
```

Pj.np: This property with the argument K is the number of nodes in the cluster that the program will run on.

Section File: This file is required and contains data about class sections. By default these are named "courses.txt".

Student File: This data file containing the students taking classes. By default this file is named "students.txt".

Relationship File: This file is required and contains information about what student is taking which course. By default this file is called "relationships.txt".

Schedule Output File: This is an optional argument to specify the path where generated optimal schedule should be saved. If not specified, the generated schedule will be discarded. This is generally not used when the user is only interested in timing and result statistics.

In addition to the normal arguments to be given to this application there are some other parameters that can be given to the JVM with the `D<property=<value>>` argument.

Name	Property	Default Value	Description
Scheduler trunk size	fes.p.trunk	1	This can be a value between (0, +infinity)

Simulated Annealing

To run the simulated annealing algorithm, follow the directions defined below for each of the versions. The following parameters for the JVM are common to the sequential and parallel versions and can be used by adding `D<property=<value>>` to the java command.

Name	Property	Default Value	Description
Initial Temperature	fes.sa.it	.93	This can be a value between (0, +infinity)

Freezing Temperature	fes.sa.ft	2^{30}	This can be a value between (0, +infinity)
Temperature decreasing rate	fes.sa.phi	.95	This can be a value between (0, 1)
Peturb rate	fes.sa.p	.1	This can be a value between (0, +infinity)

Simulated Annealing Sequential

To run the simulated annealing sequentially, execute the following command:

```
java FESSASeq <sections-file> <students-file> <relationship-file> \
[schedule-output-file]
```

Section File: This file is required and contains data about class sections. By default these are named "courses.txt".

Student File: This data file containing the students taking classes. By default this file is named "students.txt".

Relationship File: This file is required and contains information about what student is taking which course. By default this file is called "relationships.txt".

Schedule Output File: This is an optional argument to specify the path where generated optimal schedule should be saved. If not specified, the generated schedule will be discarded. This is generally not used when the user is only interested in timing and result statistics.

Simulated Annealing Cluster

To run the simulated annealing in parallel, execute the following command:

```
java -Dpj.np=<K> FESSAClu <sections-file> <students-file> \
<relationship-file> [schedule-output-file]
```

Pj.np: This property with the argument k is the number of nodes in the cluster that the program will run on.

Section File: This file is required and contains data about class sections. By default these are named "courses.txt".

Student File: This data file containing the students taking classes. By default this file is named "students.txt".

Relationship File: This file is required and contains information about what student is taking which course. By default this file is called "relationships.txt".

Schedule Output File: This is an optional argument to specify the path where generated optimal schedule should be saved. If not specified, the generated schedule will be discarded. This is generally not used when the user is only interested in timing and result statistics.

Genetic Algorithm

To run the Genetic algorithm, follow the directions defined below for each of the versions. The following parameters for the JVM are common to the sequential and parallel versions and can be used by adding `D<property=<value>>` to the java command.

Name	Property	Default Value	Description
Number of Generations	fes.ga.gen	500	As a result of the need to benchmark the sequential and parallel algorithms these programs do not end on a convergence condition instead they run for a fixed number of iterations.
Crossover Rate	fes.ga.cr	.75	The crossover rate is the percentage of the population that will breed each generation.
Mutation Rate	fes.ga.mr	.25	The mutation rate is the percentage of the children from the crossover that will have a mutation introduced in their genes.
Top-Breeding Rate	fes.ga.tbr	.2	The top-breeding rate is the upper percentage of the individuals that are automatically entered into the breeding pool of individuals.
Fresh blood Rate	fes.ga.fb	.1	In order to obtain accurate benchmarks the sequential and parallel algorithms needed to remain as similar as possible. As a result of this migration was not introduced into the parallel version and instead new individuals are introduced at the end of each generation to simulate the effects of a migration as close as possible. The rate at which individuals are replaced is the fresh blood rate.

To run the genetic algorithm sequentially, execute the following command:

```
java FESGeneticSeq <sections-file> <students-file> <relationship-file> \
[schedule-output-file]
```

Section File: This file is required and contains data about class sections. By default these are named “courses.txt”.

Student File: This data file containing the students taking classes. By default this file is named “students.txt”.

Relationship File: This file is required and contains information about what student is taking which course. By default this file is called “relationships.txt”.

Schedule Output File: This is an optional argument to specify the path where generated optimal schedule should be saved. If not specified, the generated schedule will be discarded. This is generally not used when the user is only interested in timing and result statistics.

Genetic Cluster

To run the genetic algorithm in parallel, execute the following command:

```
java -Dpj.np=<K> FESGeneticClu <sections-file> <students-file> \
<relationship-file> [schedule-output-file]
```

Pj.np: This property with the argument k is the number of nodes in the cluster that the program will run on.

Section File: This file is required and contains data about class sections. By default these are named "courses.txt".

Student File: This data file containing the students taking classes. By default this file is named "students.txt".

Relationship File: This file is required and contains information about what student is taking which course. By default this file is called "relationships.txt".

Schedule Output File: This is an optional argument to specify the path where generated optimal schedule should be saved. If not specified, the generated schedule will be discarded. This is generally not used when the user is only interested in timing and result statistics.

Interpreting Output

When the program has completed its execution it will print out a variety of numbers on standard output, which are in a specific format. The following is the format for the output.

```
[running time] [bestRank] [SE_Counter] [CE_Counter] [ME_Counter] [FE_Counter]
```

Running time: This is the amount of time the program took to execute.

Best Rank: This is the lowest rank that could be found.

SE Counter: The number of exam conflicts (multiple exams at the same time).

CE Counter: The number of consecutive exams.

ME Counter: The number of multiple exam problems (3 or more per day) that exist in the schedule.

FE Counter: The number of Friday exams that occur in a schedule.

If an output file was specified then the specific details of the schedule will be printed there. The output file consists of a line describing all of the counters then it will list all of the courses and their exam time.

Creating Custom data

Generate Section File

In order to create a custom problem size a section (or courses) file must be created. This was obtained from RIT's Student information system using a web crawler that uses a combination of bash and php. In order to run the script, it must be run on a unix system and have php installed.

To execute the crawler and place the output in a file run the crawler as follows:

```
./crawler.bash | php crawler.php > courses.txt
```

To execute the crawler and have the output placed in a mysql database call it in the following manner:

```
./crawler.bash | php crawler.php | php upload.php <host> <username> \
<password> <dbname>
```

Note that in order to use the mysql database option, you'll need to create a mysql database using the db.sql provided in the scripts/crawler/ directory.

After this is done the section file will be created and called courses.txt. This section file can then be used for generating data of a custom size.

Generate Data Set with Custom Problem Size

In order to generate custom random test data of a specific size you can call Generate data in the following way:

```
java FESGenerateData <sections-file> <students-file> <relationship-file>
<#-of-students> <student-load>
```

Sections file: The path to the file contains all of the courses. This was generated by our crawler script.

Students File: The path to store the student's file in.

Relationships File: the Path to store the relationship file.

Number of students: The number of students to be generated.

Student load: The average number of courses each student will be taking.