Development of University Timetabling System By Using Evolution Strategies and Simulated Annealing

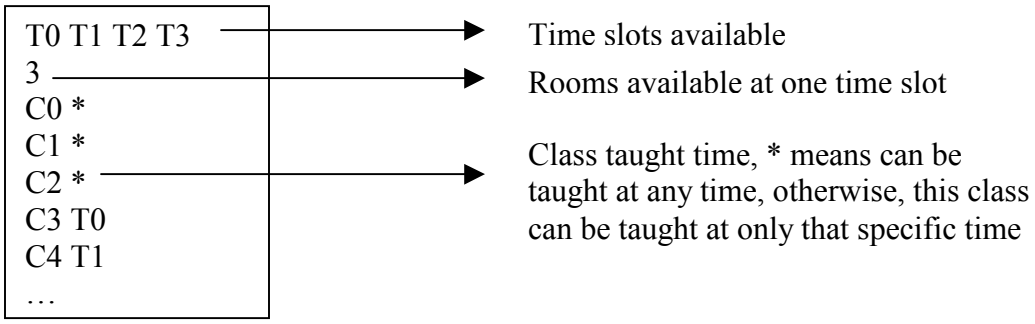
Endi Wu
EDS PLM Solutions
Troy, MI USA
Endi.wu@ugs.com

Submitted to Dr. Chan-Jin Chung, LTU/MCS Department, in partial fulfillment of the requirements for MCS 7033 Collaborative Research Project II class in Fall 2001

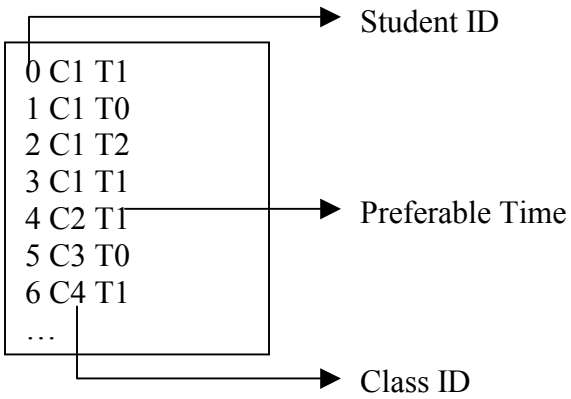
1. System Input Format

There are two files needed to create for scheduling. One is teacher schedule file that includes time slots, room available, and teacher's preference for the class. The other is student survey data that includes time preferred by student for classes. The structure of two files as follows:

Teacher schedule file:



Student survey file:



2. Algorithm

Timetabling is the assignment of time slots to a set of events, subject to constraints on these assignments. The NP-complete classes and time slots that bring highest score is a constraint satisfaction problem after evaluating student preferable time. Here I introduce two algorithms with different kinds of constraints to optimize the score.

2.1 Evolutionary Algorithm, 1 plus 1

Basic algorithm is initialize one schedule, using this schedule to generate another schedule by random method, compare two schedule, higher score schedule will survive and generate next schedule until no generate can be produced. During the reproducing, a legal schedule needs to be found such that no room is expected to accommodate more than one class at a time. The constraints for this problem can be hard (strict) or generous.

2.1.1 Strict rule

Strict constraints are usually constraints that physically cannot be violated. This includes events that must not overlap in time, such as:

- Class must be taught by the specified time appointed by professor
- One class taught by only one time

Algorithm

Schedule(parent)=Initialize schedule (init file)

For gen=1 to Maximum generation
do
    Schedule(child) = generate(Schedule(parent))
While HardConstraint(Schedule(child) return ture
If (Score(Schedule(child)) < Score(Schedule(parent)))
    Schedule(parent) = Schdule(child)
End for

2.1.2 Generous rule

Generous constraints are usually constraints that can be violated in certain range. Because sometime illegal schedule will be adjusted to good result

Algorithm

Schedule(parent) = Initialize schedule (init file)

For gen=1 to Maximum generation

    Schedule(child)=generate(Schedule(parent))

    if (GeneralConstraint(Schedule(child))

```
                If ( Score(Schedule(child) ) < Score(Schedule(parent)) )
                        Schedule(parent) = Schdule(child)
                EndIF
        EndIF
        Else
                If(InRange)
                        Schedule(parent) = Schdule(child)
                EndIF
                Else
                        Continues
EndFor
```

2.2 Simulated Annealing

The simulated annealing (SA) procedure uses the Metropolis Algorithm but varies the temperature parameter T from a high value (system at "melting point") to a low value (system at "freezing point"). The full SA procedure for minimization is then as follows (for maximization set E=-E):

Initialize T

Generate random configuration X_{old}

WHILE $T > T_{min}$ DO
    FOR i = 1 to Nc DO

        generate new configuration, X_{new}

        calculate new energy, E_{new}

        calculate $\nabla E = E_{new} - E_{old}$
        IF $\nabla E < 0$ or random < prob = $e^{-\nabla E/T}$ THEN

            $X_{old} = X_{new}$
            $E_{old} = E_{new}$
        END IF
    END FOR

    reduce T
END WHILE

Where N_c is the number of random changes in configuration at each temperature and is chosen so that the configuration has reached a minimum energy state for the current temperature. The variable *random* is a randomly generated number in the range [0,1].

2.2.1 Strict rule

Algorithm

Initialize T

Generate random configuration X_{old}

WHILE $T > T_{min}$ DO
FOR i = 1 to Nc DO

generate new configuration, X_{new}

IF (! HighContraint(X_{new}))
IF (inRange)
$X_{old} = X_{new}$
ENDIF
ELSE
continues

ENDIF
ELSE

calculate new score, E_{new}

calculate $\nabla E = E_{new} - E_{old}$
IF $\nabla E < 0$ or random < prob = $e^{-\nabla E/T}$ THEN

$X_{old} = X_{new}$
$E_{old} = E_{new}$
END IF
END FOR

reduce T
END WHILE

2.2.2 Generous rule

Algorithm

Initialize T

Generate random configuration X_{old}

WHILE $T > T_{min}$ DO
FOR i = 1 to Nc DO

```
                generate new configuration, X new

                IF ( ! HighContraint(X new ) )
                        Break;
                ENDIF
                ELSE

                        calculate new score, E new

                        calculate ∇E = E new – E old
                        IF  ∇E < 0 or random < prob = e -∇E/T THEN

                                X old = X new
                                E old = E new
                        END IF
        END FOR

        reduce T
END WHILE
```

3. Sample Result

Sample Result Include

Large number of classes
1. EC Strict with 150 classes, 2760 preferable time
2. EC Generous with 150 classes, 2760 preferable time
3. SA Strict with 150 classes, 2760 preferable time
4. SA Generous with 150 classes, 2760 preferable time

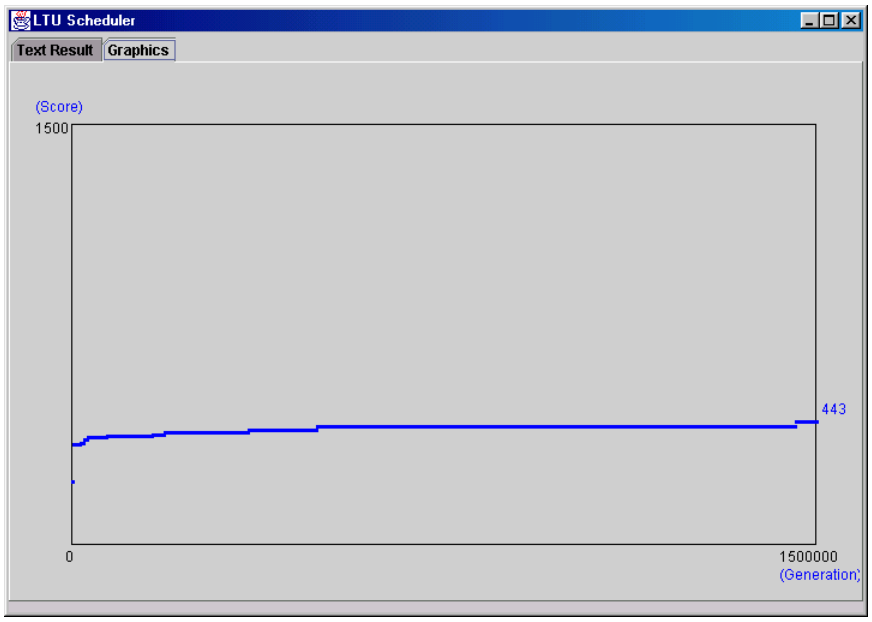
(Picture only)

Medium number of classes
5. SA Strict with 150 classes, 1337 preferable time
6. SA Generous with 150 classes, 1337 preferable time
7. EC Strict with 150 classes, 1337 preferable time
8. EC Generous with 150 classes, 1337 preferable time

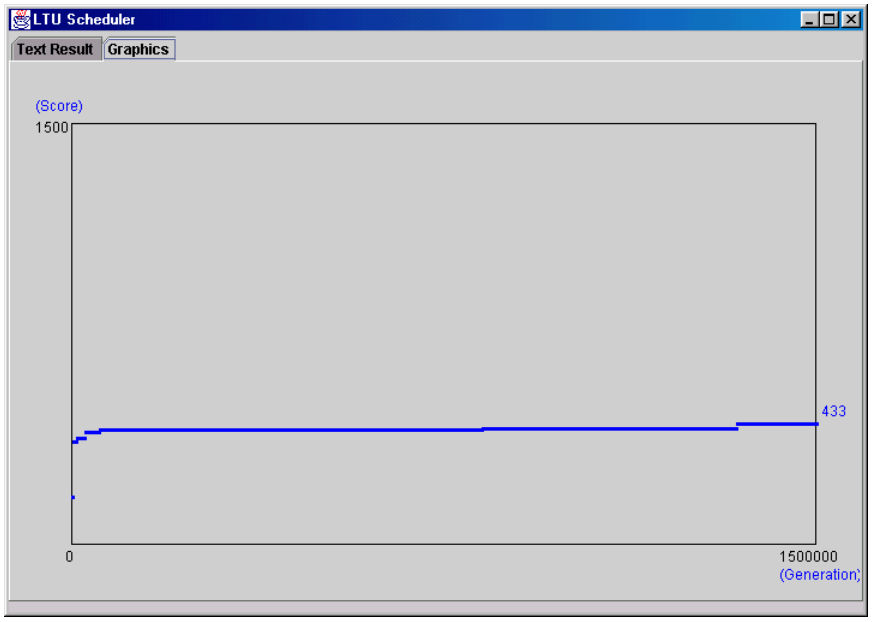
(Picture only)

Small number of classes
9. EC Strict with 2 classes, 43 preferable time
10. EC Generous with 2 classes, 43 preferable time
11. SA Strict with 2 classes, 43 preferable time
12. SA Generous with 2 classes, 43 preferable time

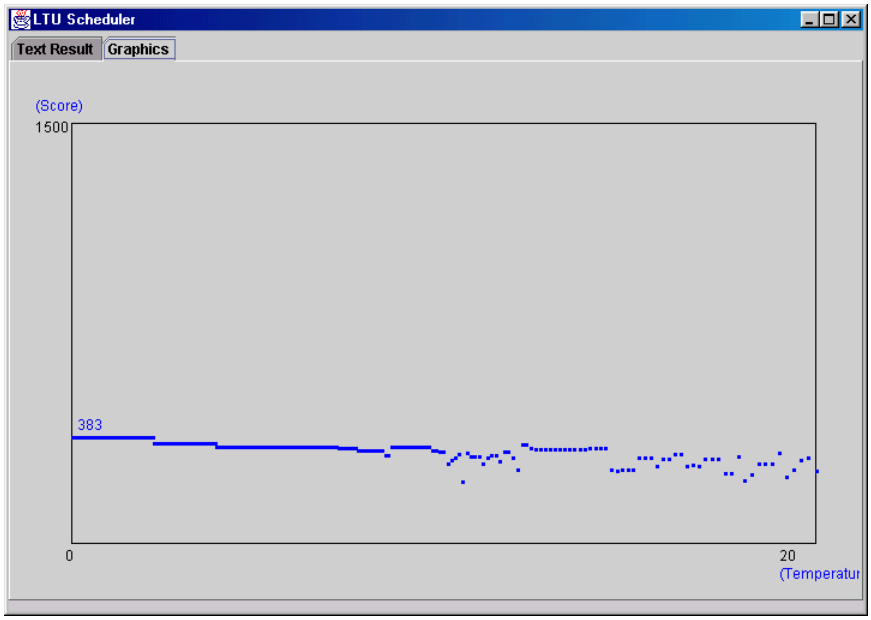
(Picture, final class assignment using program and manpower)



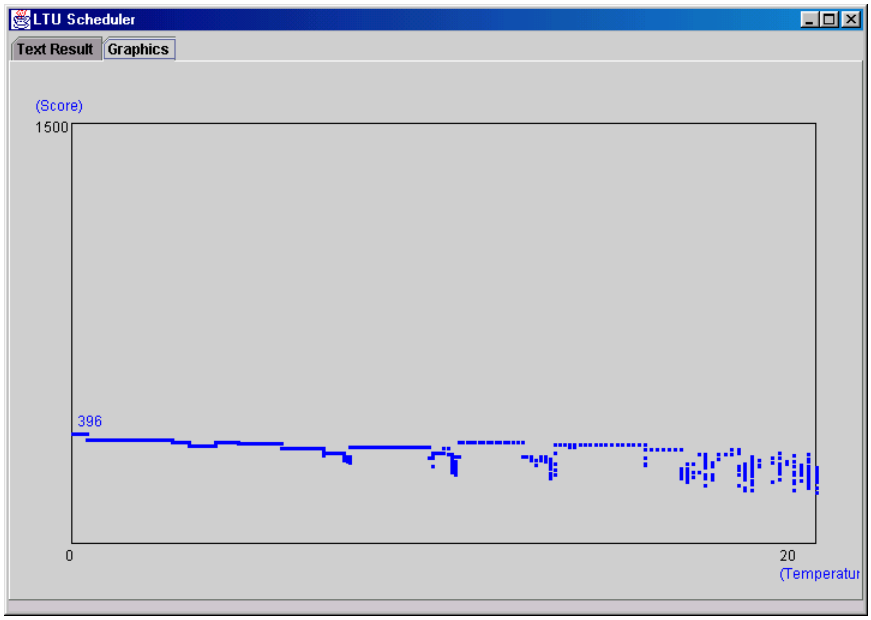
1. Evolution Computation Strict, 150 classes, total score: 443



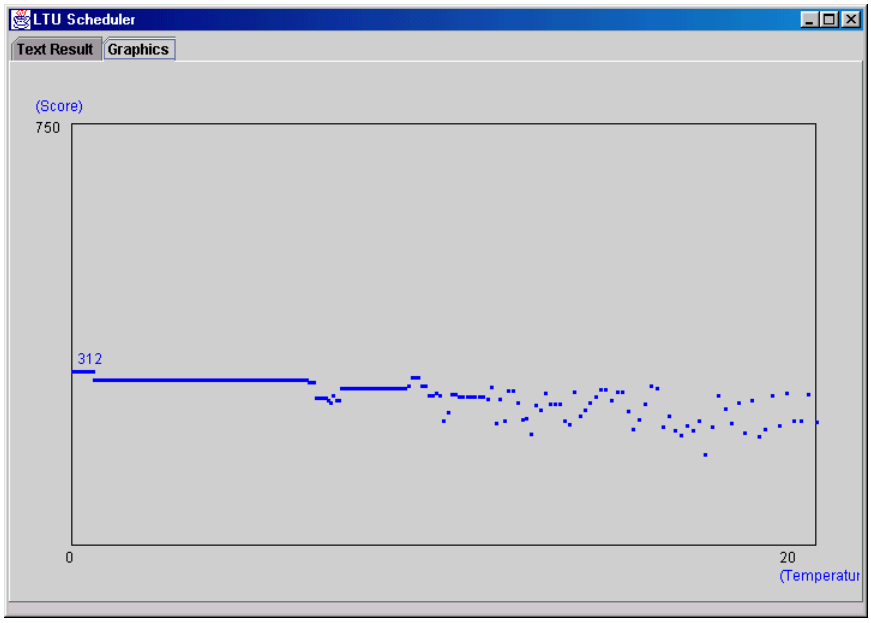
2. Evolution Computation Generous, 150 classes, total score: 433



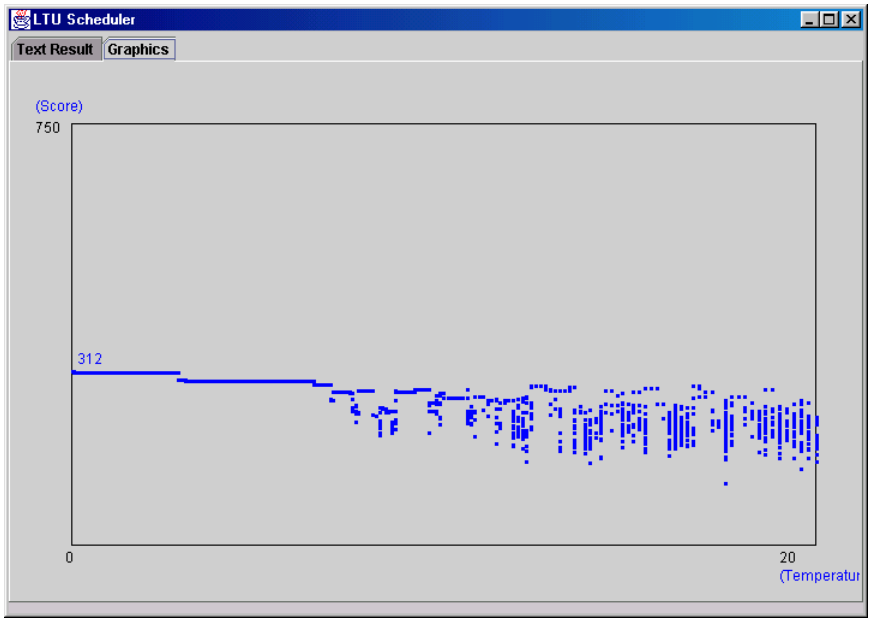
3. Simulated Annealing Strict, 150 classes, total score: 383



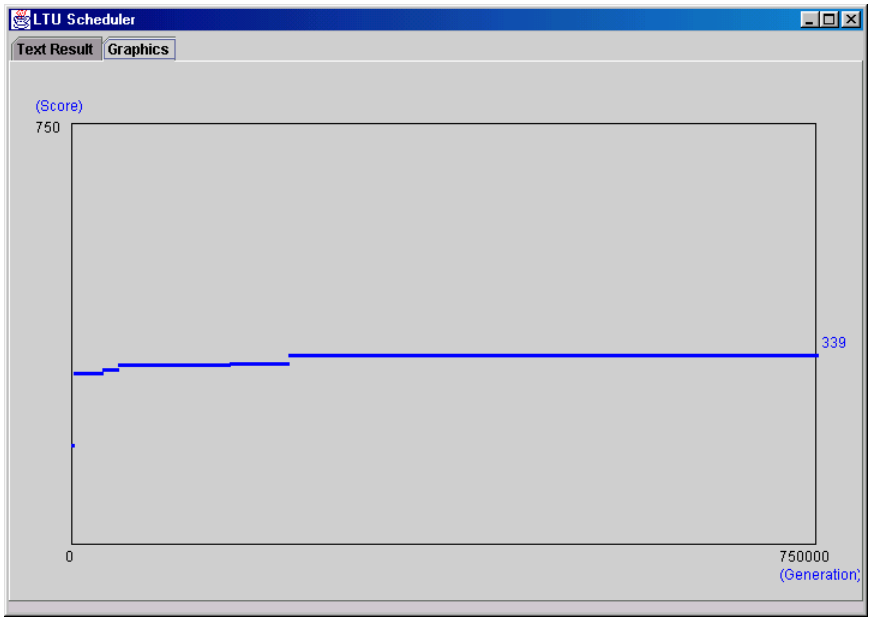
4. Simulated Annealing Generous, 150 classes, total score 396



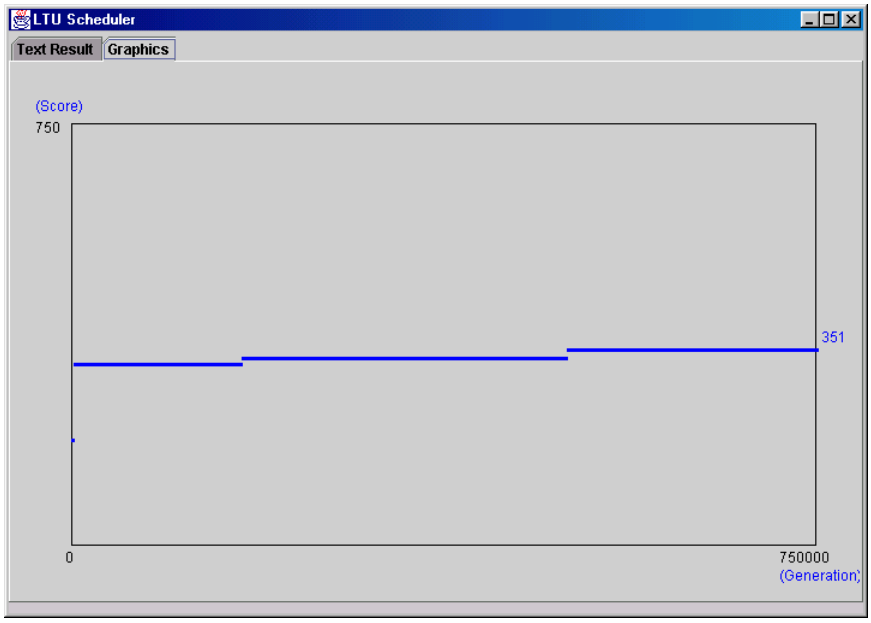
5. Simulated Annealing Strict, 75 classes, total score 312



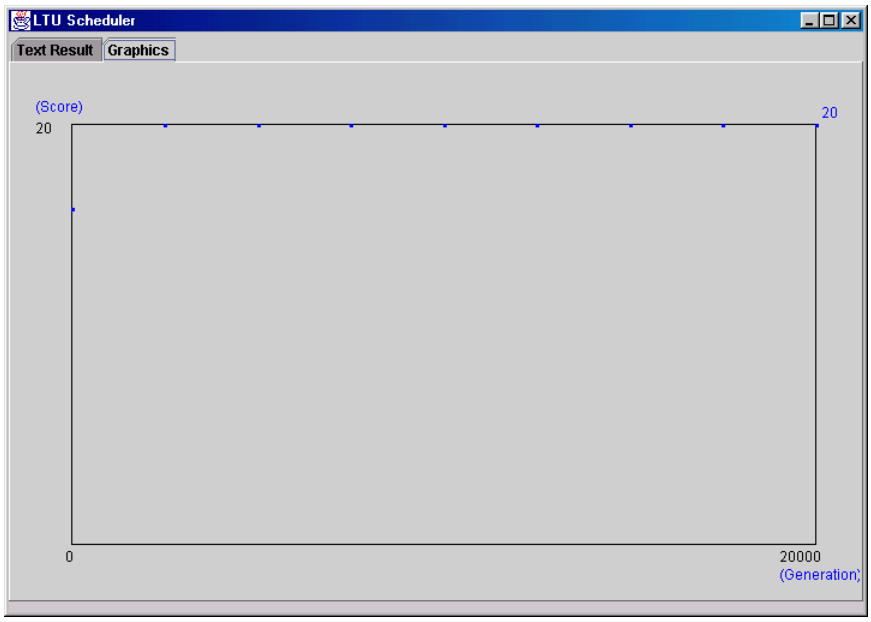
6. Simulated Annealing Generous, 75 classes, total score: 312



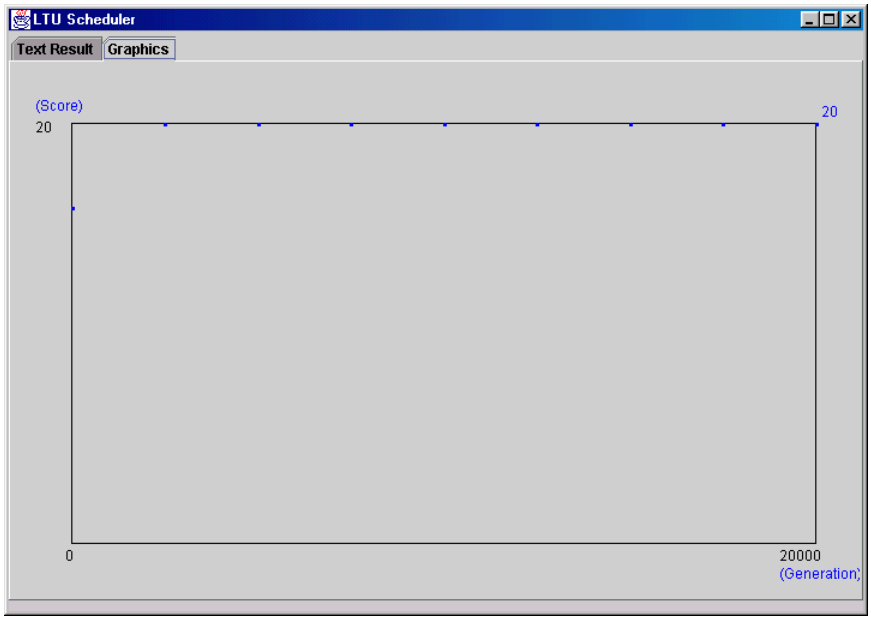
7. Evolution Computation Strict, 75 classes, total score 339



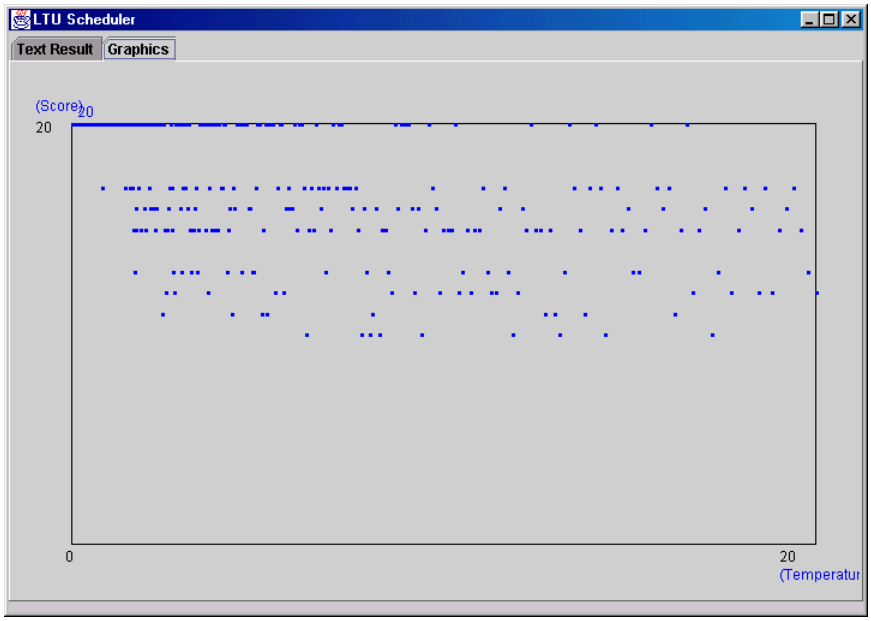
8. Evolution Computation Generous, 75 classes, total score 351



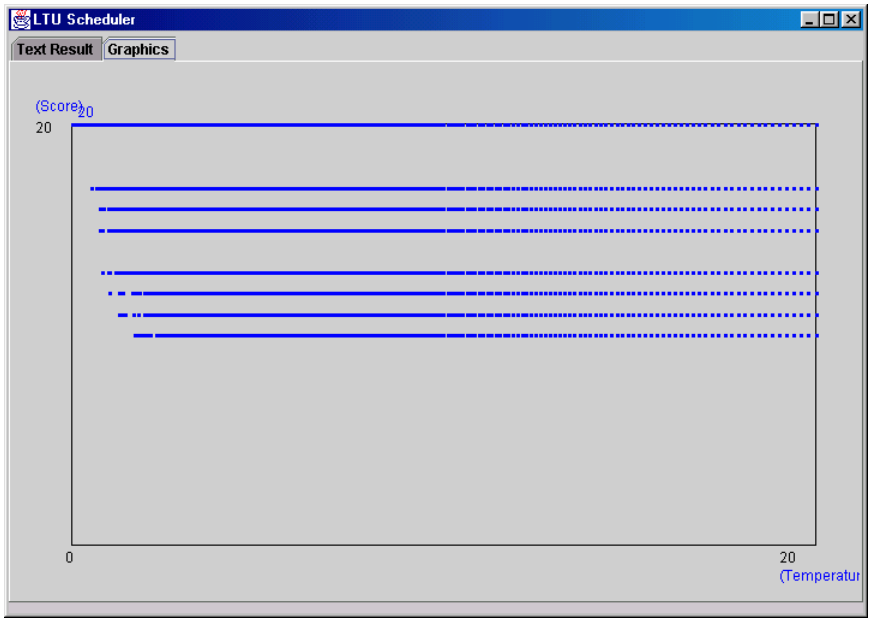
9. Evolution Computation Strict, 2 classes, total score 20



10. Evolution Computation Generous, 2 classes, total score 20



11. Simulated Annealing Strict, 2 classes, total score 20



12. Simulated Annealing Generous, 2 classes, total score 20

Student Survey File

0 C0 T0
1 C0 T1
2 C0 T2
3 C0 T2
4 C0 T2
5 C0 T2
6 C0 T2
7 C0 T2
8 C0 T0
9 C0 T0
10 C0 T2
11 C0 T0
12 C0 T1
13 C0 T2
14 C0 T0
15 C0 T2
16 C0 T0
17 C0 T1
18 C0 T1
19 C0 T2
20 C0 T1
21 C1 T1
22 C1 T1
23 C1 T2
24 C1 T2
25 C1 T0
26 C1 T1
27 C1 T2
28 C1 T0
29 C1 T1
30 C1 T1
31 C1 T2
32 C1 T1
33 C1 T0
34 C1 T1
35 C1 T0
36 C1 T1
37 C1 T0
38 C1 T1
39 C1 T1
40 C1 T2
41 C1 T2
42 C1 T2

Teacher Schedule File

T0 T1 T2
2
C0 *
C1 *

Class Assignment using program:

1. EC Strict with 2 classes, 43 preferable time
Result:

C0 T2
C1 T1
Score: 20

2. EC Generous with 2 classes, 43 preferable time
Result:

C0 T2
C1 T1
Score: 20

3. SA Strict with 2 classes, 43 preferable time
Result:

C0 T2
C1 T1
Score: 20

4. SA Generous with 2 classes, 43 preferable time
Result:

C0 T2
C1 T1
Score: 20

Class assignment using manual calculation

C0	C1	
T0	T0	6+5 =11
T0	T1	6+10=16
T0	T2	6+7=13
T1	T0	5+5=10
T1	T1	5+10=15
T1	T2	5+7 =12
T2	T0	10+5=15
T2	T1	10+10=20
T2	T2	10+7=17

So result:

C0 T2
C1 T1
Maximum Score = 20

4. Summary Results

1. Evolution Strategies (ES) with 1/5 rule performs better than Simulated Annealing (SA) according to the graph result from large to medium size of classes.
2. Generous rule is approach to handle illegal schedules, which works fine with both ES and SA.
3. ES with Generous rule once gave the best result ever found before while I was testing the system. For 150 classes, it gave about 460.