Optimization using Meta-heuristics University Timetabling

Martin Wiboe, Burak Topal, Too Sheng Tack

DTU 2015

May 9, 2015

Overview

Introduction

Every semester universities face the problem of creating good feasible timetable due to many complex constraints that have to be taken into consideration.

- Limited room capacity
- A lecturer can teach more than one courses to be scheduled in different time slots
- A curriculum has more than one courses to be scheduled in different time slots
- Also lecturers and students have preferences

Motivation

- Better utilization of resources
- Atomized planning
- TODO: continue motivation

Problem Description

Meta-heuristics

A high level procedure to find a solution for given optimization problem.

- Efficient and practical
- Do not guaranty the optimal solution

Different types of meta-heuristics:

- Hill Climber
- Simulated Annealing
- TABU

Hill Climber

Incremental local search algorithm.

- Easy to implement
- Traps in local optimum

select inital solution s_0
$s^* = s_0$
repeat
  select $s \in N(s^*)$
  if $f(s) > f(s^*)$ **then**
    $s^* = s$
  end if
until *time limit reached*
return s^*

Algorithm : Hill Climber

Hill Climber - Implementation Details

Stochastic Hill Climber

- Fast average number of **??** iteration per seconds
- Traps local optimum
- Different results for every run
- Traps in local optimum

For each iteration selects the best state from two candidate Neighbours

- candidate state by removing a course in given time slot
- candidate state by adding given course in given time slot

Simulated Annealing

Probabilistic optimization methods that uses the idea of the annealing process in thermodynamic.

- In high temperatures algorithm generally select the proposed action even it worse than the current solution.
- Decreases the temperature for each iteration with given parameter

select inital solution s_0
$T = T_{start}$
$s^* = s_0$
repeat
  select $s \in N(s^*)$
  $\delta = f(s) - f(s^*)$
  if $\delta < 0 orwithprobablityp(\delta, t_i$ **then**
    $s^* = s$
  end if
  $t_{i+1} = t_i * \alpha$
until *time limit reached*
return s^*

Algorithm : Simulated Annealing

Simulated Annealing - Implementation Details

Each iteration algorithm calculates the delta value with remove, assign and swap actions and chooses the best one.

Search(s_0, T_{start}, α)
$T = T_{start}$
$s^* = s_0$
repeat
  repeat
    select day period room randomly
    calculate; new solutions by assign remove abd swap operations
  until *no hard constraint violations*
  selectbestaction $m \in \{Remove, Assign, Swap\}$ *has lowest* $f(s_i \oplus m)$
  $\delta = f(s) - f(s_i \oplus m)$
  if $\delta < 0 orwithprobablityp(\delta, t_i$ **then**
    $s^* = s_i \oplus m$
  end if
  $t_{i+1} = t_i * \alpha$
until *time limit reached*
return s^*

Algorithm : Simulated Annealing - Pseudo Code

TABU

Uses local search paradigm and memory for optimization.

- Generally finds better solution than the other optimization problems
- Contraction of the Tabu list is problem specific

TABU - Neighbourhood Function

The neighbours are the set of the different "*next to*" solutions To generate neighbour program uses three different action:

- Remove: Program goes through all time slots if the current time slots is not empty than it uses the Remove method to generate new solution
- Swap: If current time slot is not empty then program goes through all the time slots and choose another non empty time slot and generate new solution by swapping
- Assign: If the current time slot is empty then program goes through the course list and assign current course in current time slot

TABU - Neighbourhood Function Cont.

Therefore for each iteration program generates;

- $d * p * r$ (max) number of neighbours by removing
- $d * (d - 1) * p * (p - 1)r * (r * 1)$ (max) number of neighbours by swapping
- $d * p * r * c$ (max) number of neighbours by by assigning
- total max $d * (d - 1) * p * (p - 1)r * (r * 1) + d * p * r$ and min $d * p * r * c$ neighbours are generated in each iteration
- d = number of days
- p = number of periods per day
- r = number of rooms
- c = number of courses

TABU - Implementation Details I

Search$(s_0, \mathit{taboLength})$
$s^* = s_0$
repeat
    for each *slot* $t_1 \in \mathit{set}\{\mathit{day}, \mathit{period}, \mathit{room}\}$ **do**
        if t_1 *is not empty* **then**
            $s_n = \mathit{RemoveAt}(t_1$
            if $f(s_n) < f(s^{'})$ *and* $\mathit{RemoveAt}(t)$ *is not tabu* **then**
                $s^{'} = s_n$
            end if
            for each *slot* $t_2 \in \mathit{set}\{\mathit{day}, \mathit{period}, \mathit{room}\}$ **do**
                if t_2 *is not empty* **then**
                    $s_n = \mathit{Swap}(t_1, t_2)$

TABU - Implementation Details II

if $f(s_n) < f(s^{'})$ *and* $Swap(t_1, t_2)$ *is not tabu* **then**
$s^{'} = s_n$
end if
end if
end for
end if
if t_1 *is empty* **then**
for each *courses* $c \in CourseList$ **do**
$s_n = Assign(t_1, c)$
if $f(s_n) < f(s^{'})$ *and* $Assign(t_1, c)$ *is not tabu* **then**
$s^{'} = s_n$
end if
end for

TABU - Implementation Details III

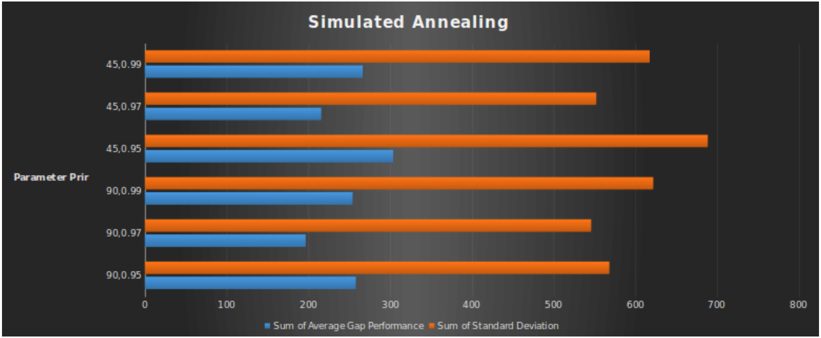
        end if
    end for
    $s = s^{'}$
    $AddTaboList(action)$
    if $f(s') < f(s^*)$ **then**
        $s^* = s^{'}$
    end if
until *time limit reached*
return s^*

Algorithm : TABU - Pseudo Code

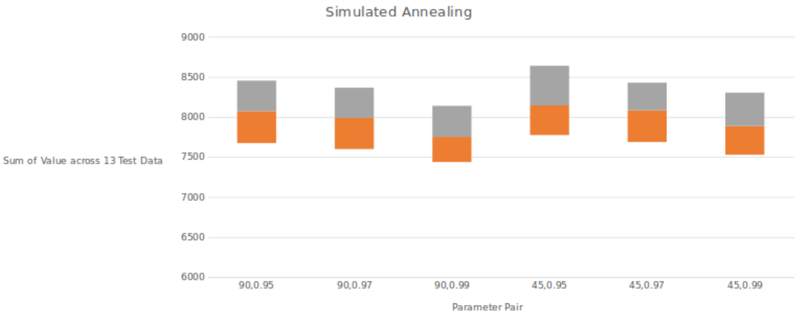
Parameter Tuning

- 13 different test data, 15 runs for each test data
- Simulated Annealing
  - 2 initial temperatures : 90 and 45
  - 3 cooling rate : 0.95, 0.97,0.99
- TABU:
  - 3 taboo length: 5, 10 and 20

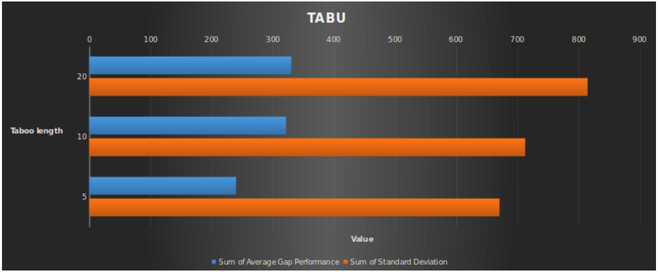
Simulated Annealing



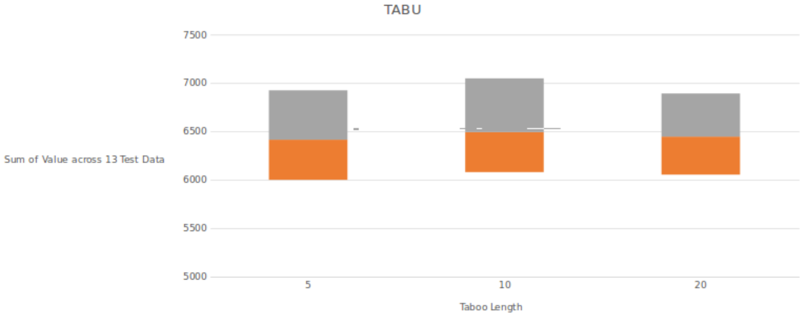
Simulated Annealing



TABU


TABU
0
100
200
300
400
500
600
700
800
900
20
10
5
Taboo length
Value
Sum of Average Gap Performance
Sum of Standard Deviation

TABU


TABU
7500
7000
6500
6000
5500
5000
Sum of Value across 13 Test Data
5
10
20
Taboo Length

Conclusion

Questions