Algorithmic Perspective in the UPLB Student Registration Process

Joseph Anthony C. Hermocilla
(jachermocilla@uplb.edu.ph)

The Registration Process

- Two perspectives
  - Student
    - What subjects to take to graduate on time, or earlier?
    - Obtain a schedule that is most “convenient”
  - Administration
    - How to minimize resources (teachers,classrooms, time, monetary cost) required
    - What subjects to offer and how many sections to open per offered subject

What subjects to take

- Let students select the subjects to take
  - Guided by the curriculum
    - A regular student can blindly follow what is prescribed in the curriculum
    - Not all students are regular
    - Students might not know what subjects to prioritize

Obtain convenient schedule

- Dependent on the course offering
- Dependent on the selected subject
- Dependent on the sections chosen
  - Schedule conflicts may arise

Minimize Resources

- Minimize the number of sections to offer per subject
  - Less instructor, less rooms, less time
- Can be done at the department/institute level
- Requires estimate of the demand for each subject
  - Requires grades from previous semester
- Each section must have at at least the minimum number of students

Problem 1

Given a set **C** of subjects based on the curriculum, and a set **T** subset of C where each t in T has been taken by a student, create a list **A** with elements from subset of **C-T** subject to the following:

- All element **a** of A must be offered next semester
- All prerequisites of element a of A must be satisfied
- All elements a of A must not delay the student from graduating

Determining Critical Subjects

- Uses Critical Path Method (CPM)
  - A tool in project management for scheduling a set of activities
  - Inputs
    - Work Breakdown Structure (WBS)
      - List of activities required to complete a project
    - Duration of each activity
    - Dependencies between the activities
  - Calculates the longest path (critical path) of planned activities

Determining Critical Subjects

- Uses Critical Path Method (CPM)
  - For each activity, the following information will be derived (function of dependencies)
    - Earliest start
    - Earliest finish
    - Latest start
    - Latest finish
    - Float time/slack time
  - If float time is zero, activity is a critical activity

Implementation of CPM

- Start by creating a task network
- Basically a graph with vertices and edges
  - Vertices represent events(completion of a task)
  - Edges represent tasks
- Special vertices: START and END
- Apply graph traversal algorithms to update the information for each vertex

Example

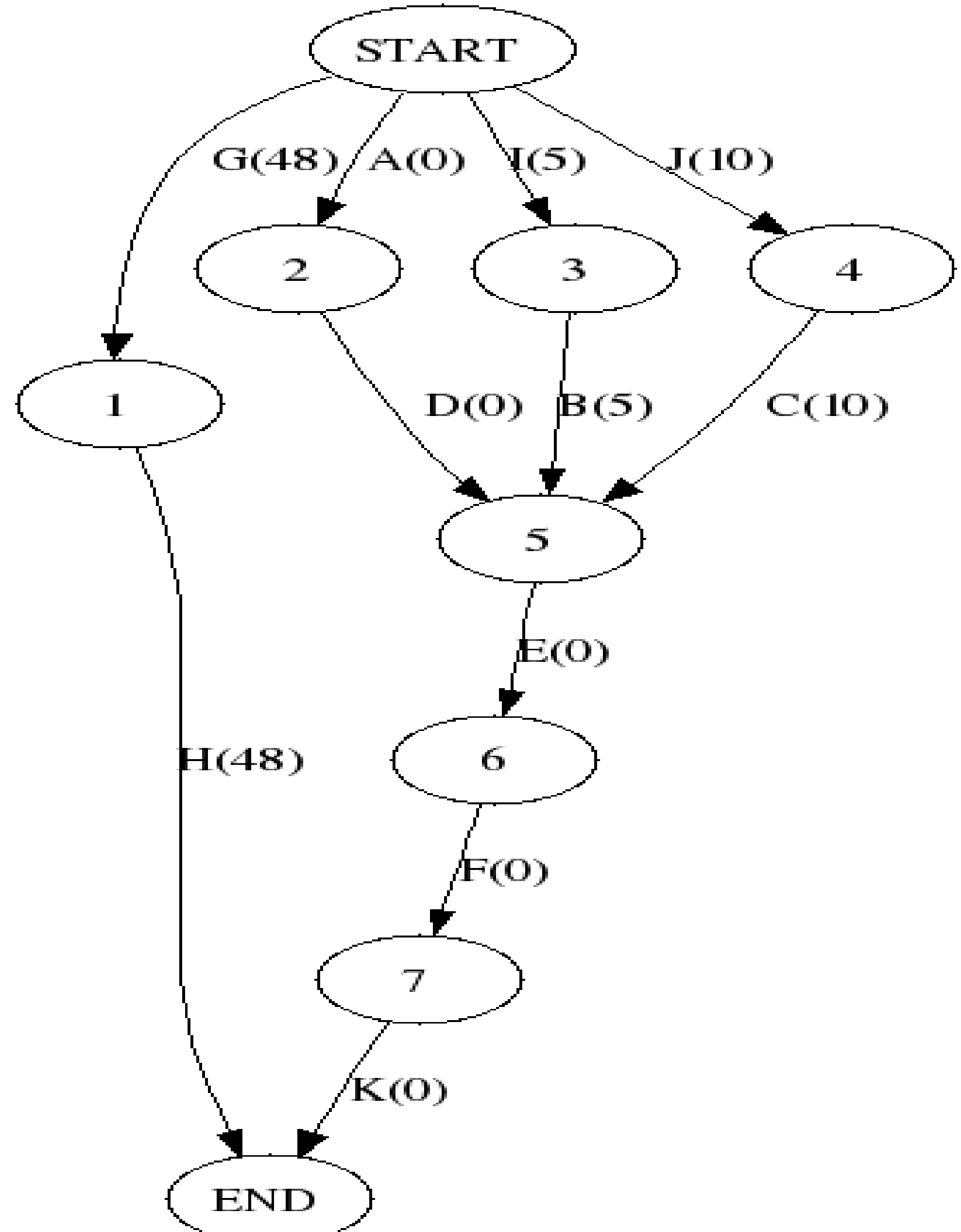
- Game development
- Activities or Tasks
  - A,14,Graphics Engine
  - B,5,Sound Engine
  - C,5,Music Engine
  - D,10,Input Engine
  - E,31,Gameplay/General Programming
  - F,7,Physics
  - G,14,2D Artwork
  - H,21,3D Artwork
  - I,14,Sound Effects
  - J,9,Music Recording
  - K,21,Level Design

Example

- Dependencies
  - 1,START,G
  - 2,START,A
  - 3,START,I
  - 4,START,J
  - 5,2,D
  - 5,3,B
  - 5,4,C
  - 6,5,E
  - 7,6,F
  - END,7,K
  - END,1,H

Game Development

- Result of CPM



Mapping

- WBS maps to curriculum
- Subjects in curriculum map to activities
  - Duration may be constant or a rank value
- Edges represent subjects
- Nodes represent completion of subject
- Dependencies map to prerequisites
- (See sample graph for 1996 BSCS curriculum)

Algorithm 1

Define two functions: **is_offered**(x) and **prereq_satisfied**(x)

Let **D** = set of subjects on the **critical path**

Let **A** = empty

Let **B** = **C** – **T**

For each **d** *in* **D**

    If ((**d** in **B**) *and* (**is_offered**(**d**)) *and* (**prereq_satisfied**(**d**))) *then*

        add **d** to **A**

    End If

End For

For each **e** *in* **(B** - **D)**

    If (**is_offered**(**e**)) *and* (**prereq_satisfied**(**e**)) *then*

        add **e** to **A**

    End If

End For

Demand Prediction

```
Let S be the set of students
For semester from 1 to N
    For each student s in S
        Retrieve student record of s
        Predict courses to be registered by s for semester (Algorithm 1)
        Generate random grades for s
        Update student record of s
        Accumulate statistics
    End For
    Summarize statistics
End For
```

Problem 2

Given a set of students **S** and a set of classes **T,** |S| < |T| , define a function **f**: **s** -> **t**, that maps each element of S to an element t of T. Let **U** be the set of the mappings generated by f. Let $\mathbf{V_i}$ be a subset of U such that for each element **(x,y)** of V_i, x = i. For all V_i, i=1 to |S|, the following properties must hold true.

Problem 2

Given a set of students and the subject they need to register together with the course offering, assign the students to sections so such that

- There are no conflicts among the sections assigned to a student
- Capacity of each section is not exceeded

Assumptions

- Capacity of each section of a subject is sufficient to accommodate all students taking the subject
- All sections of a subject needed by a student are not in conflict with all the section of another course

REGIST

Let **S** be the set of students

Let **T** be the list of subjects

Sort **T** according to **rank** of subject

Foreach subject **t** in T

    Let **U** be the list, subset of S, who needs to take t

10: Let **V** be the set of sections for subject t

    Foreach **u** in U

        Let **W** be a list, subset of V, so that each **w** element of W is not in conflict with each **x** of the sked set **X** of u

        Sort W according to *desirability*

    Sort U based on the cardinality of W for each u

    Foreach u in U and **not_full**(V)

        Add w to X of u such that *desirability* of w is high

        if **full**(w) remove w from V and goto 10

REGIST

- Implemented in Fortran 95
- Currently used in UPLB
- Attempt to implement the algorithm in Java
  - Results must be validated with results of original implementation
    - Some heuristics were not implemented, may lead to different result
  - Advantages: Maintainability, Easy integration with information systems (SystemOne)

REGIST

- Is everyone(student and administration) happy?
- My opinion: YES!
- However, there's always room for improvement
- Attempt to improve REGIST to make everyone HAPPIER
- More of an intellectual exercise

Attempts to Improve REGIST

- Considered problem as Optimization Problem
- An objective function is defined, attempt to minimize or maximize the objective function
- Uses variety of techniques
  - Metaheuristics

Guinto (2004)

- MS Thesis
- Used Tabu Search
  - Local search technique, relies on memory-based prohibition
- Efficient solution representation
- Objective function
  - APL=(Sum of % of units given to each student) / (number of students)
- Output of REGIST used as initial solution

Marapao (2004)

- MS Thesis
- Used Genetic Algorithms
  - Search algorithms based on the natural behavior of organisms
- With GUI
- Objective function
  - APL=(Sum of % of units given to each student) / (number of students)
  - Fitness Function: $1/(1+APL^2)$
- Output of REGIST used as initial solution

Sandig (2004)

- MS Thesis
- Used Simulated Annealing
  - Based on the physical process of annealing, cooling of a collection of hot vibrating atoms
- Uses temperature parameters
- Objective function
  - APL=(Sum of % of units given to each student) / (number of students)
- Output of REGIST used as initial solution

Other Possibilities

- Ant Colony Optimization
- Great Deluge
- Swarm Optimization
- Multiagent Systems

Observations

- So far, REGIST is still the winner!
- Comparative analysis can be done to evaluate the different improvement attempts
  - Requires a standard data set
- One drawback of current solutions: Does not consider student preference in schedule

Multiagent Systems

- Subfield of Distributed Artificial Intelligence
- A system composed of agents, distributed
- Each agent is an autonomous entity, acting on behalf of an owner or user, to carry out a specific task, in an environment
- May interact with other agents to achieve a goal
  - Coordinate, Negotiate

Multiagent Systems

- Multiagent Resource Allocation
  - <A, R, U>
    - A is a set of agents
    - R is a set of resources
    - U is a set of utility functions, maps a subset of resources to a real number
  - Find an allocation that assigns an agent to a subset of resources

Multiagent Systems

- In the context of registration (a work in progress)
  - An agent is assigned to a student
  - Resources are the subjects and sections
  - Utility refers to the “desirability”
- Ideally, we let the agents interact with each other, and hopefully, they will come up with the allocation
- It is more complex however

Multiagent Systems

- Negotiation
  - Agents must negotiate to obtain desired subset of resources
    - An agent may have the resource desired by another agent
      - Willing to exchange resource for something better ('swapping' in registration terms)
  - What are some protocols for negotiation?
    - Auctions
      - Regist allocates slots per subject, agents can place bids
    - Monotonic Concession Protocol
      - Bilateral negotiation, for 'swapping'

Other Problems

- REGIST relies on two assumptions
  1. Capacity of each section of a subject is sufficient to accommodate all students taking the subject
  2. All sections of a subject needed by a student are not in conflict with all the section of another course
- How can we guarantee that these assumptions are met, specifically assumption 2?
  - Generation of course offering

References

- R. Carino. *Automation of Student Preregistration*. 1995
- D. Guinto.*University Timetabling Using Tabu Search.* 2004
- R. Marapao. *A Genetic Algorithm Approach for Solving the University Timetabling Problem.* 2004
- J. Sandig. *An Automated Timetabling Using Simulated Annealing*. 2004

Thank you for listening!

Questions?