Research 1

School (2/1/21 – 5/15/21)

Tasks

Mon	2/1/21	Advisor Interaction	00:52	0.87	Talked and sent slack messages to profs asking about topic ideas,
Tue	2/2/21	Advisor Interaction	00:45	0.75	Watched some videos about TLA+, a recommended topic from Dr. Nurk
Wed	2/3/21	Advisor Interaction	00:53	0.88	Slacked Hibschman about envisage problem, Slacked Nurk about TLA+, looked into a few TLA+ papers
Fri	2/5/21	Advisor Interaction	00:44	0.73	Asked Dr. Nurk if he'd advise, described the Envisage idea
Sun	2/7/21	Advisor Interaction	00:33	0.55	A0
Mon	2/8/21	Class Time	00:51	0.85	Discussed research topics and advisors, introduced literature survey milestones. Dr. Denning mentioned room scheduling and dating website algorithms as potentially relevant
Tue	2/9/21	Literature Survey	01:37	1.62	Talked to Tim about the problem
Wed	2/10/21	Literature Survey	00:36	0.60	Found a few different potential algorithmic approaches including the calendar scheduler algorithm and simulated annealing
Fri	2/12/21	Literature Survey	01:51	1.85	Got lost in ACM IEEE websites and paid money, Started surveying the landscape found 2 related articles, started looking at an abandoned patent
Sun	2/14/21	Literature Survey	04:32	4.53	Found a few papers, including a very close one, Got references for A, A1 Lit Survey milestone 0
Mon	2/15/21	Class Time	00:46	0.77	Research vs project discussion

		Literature Survey	00:32	0.53	
Wed	2/17/21	Literature Survey	02:16	2.27	Reading TAROT, Finished reading tarot, Reading Algrotihmic Decision…
		Advisor Interaction	00:54	0.90	Talked about papers I read, constraints, paper reading process
Fri	2/19/21	Literature Survey	01:27	1.45	Looked into paper and Zotero, decided not to use them for now,
Sat	2/20/21	Literature Survey	03:15	3.25	Started survey, Started survey, skimmed more articles
Sun	2/21/21	Literature Survey	02:54	2.90	Rough draft survey
Wed	2/24/21	Literature Survey	00:45	0.75	Persepolis
		Advisor Interaction	00:30	0.50	Caught Dr. Nurk up on time tabling, hybrid genetic algorithm, DSL
Fri	2/26/21	Literature Survey	00:51	0.85	Percepolis
Sun	2/28/21	Literature Survey	06:01	6.02	persepolis, Updated lit survey, skimmed all the rest of the papers, journaled them
Wed	3/3/21	Advisor Interaction	00:56	0.93	
Fri	3/5/21	Literature Survey	05:48	5.80	, Finished rough draft of survey
Sat	3/6/21	Literature Survey	00:32	0.53	15 sources, reviewed first half
Mon	3/8/21	Literature Survey	00:49	0.82	
		Misc Research	00:47	0.78	Made spring timeline
		Preferences	00:05	0.08	Created question for preference “survey”

Wed	3/10/21	Advisor Interaction	01:01	1.02	Went over timeline
Mon	3/15/21	Class Time	00:43	0.72	Went over timelines
Tue	3/16/21	Preferences	00:38	0.63	
Wed	3/17/21	Preferences	01:56	1.93	Preference organization, Structuring preferences
Sat	3/20/21	Inputs/Outputs	00:58	0.97	
Sun	3/21/21	Inputs/Outputs	05:31	5.52	Catalog inputs, Finished schema
Wed	3/24/21	Advisor Interaction	00:58	0.97	
Tue	4/6/21	Constraint Logic	00:53	0.88	
Sat	4/24/21	Constraint Logic	00:32	0.53	Looked into JPL,
Mon	4/26/21	Constraint Logic	01:15	1.25	Diagramming time constraints
		Inputs/Outputs	00:21	0.35	Started java objects,
		Java Objects	01:07	1.12	Java getopt, Gson. Reading files, but need to figure out how to create objects so they can be serialized
Fri	4/30/21	Archive Paper	01:05	1.08	
Sat	5/1/21	Java Objects	01:29	1.48	Json serializing and deserializing
Sun	5/2/21	Archive Paper	02:09	2.15	Started writing, Topic, experiment, input output
Wed	5/5/21	Misc Research	01:37	1.62	Researched temporal reasoning, and computation tree logic
Fri	5/7/21	Archive Paper	00:06	0.10	
Mon	5/10/21	Archive Paper	01:12	1.20	
Tue	5/11/21	Archive Paper	05:26	5.43	Added prototype documentation, started learnings, , Finished Archive Paper
		Archive Paper	**09:58**	**9.97**	
		Java Objects	**02:36**	**2.60**	
		Misc Research	**02:24**	**2.40**	

Class Time	**02:20**	**2.33**
Advisor Interaction	**08:06**	**8.10**
Constraint Logic	**02:40**	**2.67**
Inputs/Outputs	**06:50**	**6.83**
Preferences	**02:39**	**2.65**
Literature Survey	**33:46**	**33.77**
	71:19	**71.32**

Language Structures

School (2/1/21 – 5/15/21)

Tasks

Mon	2/22/21	DSL	01:17	1.28	Talked to Dr. Denning, understood problem
Wed	2/24/21	DSL	01:01	1.02	Setup
Sun	2/28/21	DSL	01:12	1.20	
Mon	3/1/21	DSL	03:20	3.33	Input & output, Milesone first, selection
Tue	3/9/21	DSL	00:46	0.77	
Wed	3/10/21	DSL	01:19	1.32	ANTLR grammar and constraint grammar
Sat	3/13/21	DSL	02:19	2.32	Milestone 1&2
Mon	3/22/21	DSL	00:48	0.80	
Tue	3/23/21	DSL	01:00	1.00	Started helpers.py
Wed	3/24/21	DSL	00:53	0.88	validation
Sun	3/28/21	DSL	05:39	5.65	Added more helpers and context generators, objectivication
Sat	4/3/21	DSL	02:10	2.17	Preference scoring
Thu	4/8/21	DSL	03:20	3.33	Started python generation API
Sat	4/10/21	DSL	02:02	2.03	Worked on python generation API
Sun	4/11/21	DSL	00:54	0.90	
Mon	4/12/21	DSL	00:19	0.32	
Wed	4/14/21	DSL	04:45	4.75	Symmetric functions, , Symmetric functions
Mon	4/19/21	DSL	05:13	5.22	Implementing API, , bounds, Operator Constraints,
Tue	4/20/21	DSL	01:48	1.80	If, when blocks

Fri	4/23/21	DSL	01:06	1.10	Not constraints
Mon	4/26/21	DSL	01:04	1.07	readme
Thu	4/29/21	DSL	00:23	0.38	
Mon	5/3/21	DSL	02:56	2.93	Documentation, , Documentation
Thu	5/6/21	DSL	02:31	2.52	
Fri	5/7/21	DSL	03:51	3.85	Finished project
Tue	5/11/21	DSL	00:31	0.52	Practice presentation
			52:27	**52.45**	

	Total	**123:46**	**123.77**	

Spring Journal

Feb 1: Research 1

1. Find advisor
2. Find achievable research topic
    1. The answer should bring more questions
3. Survey becomes related work section of paper
    1. 10-20 relevant papers
    2. 2-4 strongly related papers
    3. 5-10 pages?
    4. Note where other papers are published, when the conferences deadline is, whether they accept undergrads
4. 10 hours a week

Advisors

- Meet once a week, discuss what papers you have read, direction of research

Questions:

- When, why

Not:

- What, how

March 15: Articles

- A Hybrid Genetic Algorithm for Course Scheduling and Teaching Workload Management

Skimmed Papers

Relevant

- A Hybrid Genetic Algorithm for Course Scheduling and Teaching Workload Management
  - Genetic algorithm that offers various optimizations
- School Timetabling in Theory and Practice
  - Compares Simulated Annealing and Tabu Search
  - Simulated Annealing arrived at better solutions, and in less time than Tabu Search
  - Includes helpful pseudo code for both
- Solving the Course Scheduling Problem Using Simulate Annealing
  - Decreasing threshold/temperature for bad steps

- How a data-driven course planning tool affects college students' GPA: Evidence from two field experiments

> Our two studies offer consistent evidence that knowledge of the distribution of prior students' academic experiences causes changes in behavior that, in the aggregate, produce lower earned grades.

- A Guided Search Genetic Algorithm for the University Course Timetabling Problem
  - Guide the GA by noting which parts of a chromosome have no penalty, and encouraging the passing on of those parts of the chromosome
- Comparison Using Particle Swarm Optimization And Genetic Algorithm For Timetable Scheduling

    - PSO had less penalty after 500 iterations
- A survey of metaheuristic-based techniques for University Timetabling problems
- The complexity of timetable construction problems
    - Specification for time tables
- A Survey of Automated Timetabling
- Using Tabu Search for Solving a High School Timetabling Problem
    - Accounts for course time
- Solving the Post Enrolment Course Timetabling Problem by Ant Colony Optimization

Less Relevant

- CourseRank: A Social System for Course Planning
    - A closed-community, hybrid-system (official and crowd sourced) course ranking and planning site
- A Json-based Self-advising System
    - Specifies JSON for degree requirements
- Empirical Support for Winnow and Weighted-Majority Algorithms: Results on a Calendar Scheduling Domain
    - Super advanced calendar scheduling algorithms using machine learning methods
- Taking DCOP to the Real World: Effifficient Complete Solutions for Distributed Event Scheduling
    - Distributed constraint optimization
- Automatic Generation Of University Timetables: An Evolutionary Approach
    - $\frac{1}{1+\left(\sum_{C} weight*cost\right)^2}$
- A Genetic Algorithm for the Teacher Assignment Problem for a University in Indonesia

- Timetable Scheduling Using Particle Swarm Optimization
    - Particles updated according to other particles (learn PSO)
- Solving University Course Timetabling Problems Using Constriction Particle Swarm Optimization with Local Search
    - Evaluates “inertia weight version” and “constriction version” methods
- A Fast Genetic Algorithm for Solving University Scheduling Problem

A Course Planning Application for Undergraduate Students Using Genetic Algorithm

Considerations

1. Number of courses after 4 year
2. Semesters over 22 credits
3. Prerequisites violation
4. Difference from a plan

Strengths

- Can test a large variation of plans
- Arbitrary cost function doesn’t mess up search space

Weaknesses

- No sense of course offering times
- Searches mostly invalid plan space
- 30-60 minutes

Algorithm

- Create a set of strings encoded as a chromosome string
- Iteratively apply three processes probabilistically according to fitness function
    1. mutation
    2. crossover
    3. Selection

Papers to look at

- 7

Tarot: A Course Advising System for the Future

Considerations

- Double majors
- Study-abroad
- Course-overrides
- Early graduations
- Transfer Credits
- Questions about present: courses taken, GPA, credits earned
- Questions about a possible future: what courses to take, which semester, how do things if you don't want to take more than two major courses a semester
- Questions about all possible futures: when to study abroad to minimize graduation delay, what grade needed for certain GPA
- Questions about rules: overlap between majors, how many teacher s are needed to cover all possible schedules

Related Works:

- Minimum credit hour/semester, courses per semester, required electives

Strengths

- Can generate all possible scenarios
- Can work with arbitrary constraints as specified as Prolog facts
- Avoids evaluating invalid plans
- Integrates with Ellucian Degree Works

Weaknesses

- Devalues Gen Eds
- No sense of course offering times

Algorithm

- Creates blank schedule with free slots
- Recursily fills free slots with courses

Papers to look at

- 7
- 2 Bayes Net
- 6

A Genetic Algorithm Solving a Weekly Course-Timetabling Problem

Considerations

- Strengths
- Intelligently avoids illegal timetables

Algorithm

- C & Prolog
- Distinguishes requirements from preferences
- Prolog procedure creates initial and valid population without regard to preferences
- Squished cost function:
  - $eval(f) = \frac{1}{1+cost(f)}$ ← range is $0 < eval(f) < 1$
  - $cost(f) = \sum_{i=1}^{k} w_i . n_i(f)$
  - e_i ← the importance of the soft constraint
  - $n_i(f)$ ← quantity of violation
- Mutations only produce feasible solutions (somehow) so $eval(f)$ disregards requirements

PERCEPOLIS: Algorithmic Decision Support for Personalized Education

Considerations:

- degree requirements
- student interests
  - SSM
  - historical performance of other students in classes to determine optimal semesters (not so sure this is a good idea)
- time to degree
- number of descendents in prereq chain (schedules them earlier) also called oppurtunity value
- before starting, verifies it's feasable for classes to fit in given semesters
  - also worth noting: longest prereq chain is the minimum number of semester required

Strengths:

- good for online courses which have not scheduling conflicts to worry about
- data heterotenety intelligence using **summary schemas model** (SSM) to connect similar termenology to connect relevant courses
    - across institutions
    - from a user interest

Weaknesses:

- limits semster load by course number
- no sense of course offering

Algorithm:

1. creates graph of learning requirements
    - includes required courses, prereqs, whether a requirement has been met
2. adds electives according to interests (uses SSM)

Papers to look at:

- 5
- 9
- 11
- 12
- 13

March 17: Preferences

The Question

> I'm doing research and trying to create a system that can create university plans based off of different requirements and preferences of a student. Right now I'm trying to get a sense for what kinds of considerations people take when creating a four year plan, and what kinds of things they'd like an automated system to consider if it was scheduling their college courses. Examples might include:
>
> - I prefer morning classes
> - I prefer to spread general education classes evenly over the semesters
> - I prefer classes with Dr. X over Dr. Y
> - I am interested in classes related to topic "x"
>
> If you have time, could you please tell me some of the considerations you use when planning?

Tim

- Space left (for fun classes)
- Ability to front-load (or really just general distribution patterns)
- Has x course done by x time (to get an internship)
- Does x course ASAP
- Has a difficult COS course each semester
- Is flexible, has good alternatives (hard to rate), not bad worst case scenarios
- Front-loads courses that aren't guaranteed to be offered again
- Has overlap with friends' schedules
- (within a semester) sections are consecutive (with a lunch break between 11-2)
- (within a semester) the day starts later

Tessa

- Having a lunch break
- Consecutive classes in same building
- Some Weekdays having no/less/lab classes
- Taking two classes with prof
- Prof history
- Profs with experience in class

Madison

- lunch
- Same building
- Classmates
- Earlier but not 8
- Days of the week
- Preferring two classes not in same semester

Daniel

- One thing that would be nice would be to be able to specify a preference in the order you take classes even if they aren't prerequisites. For example, I would like to take class X after class Y even if Y isn't a prereq for X

Liz

- Lighter semester when taking class
- Not too many major classes in semester
- Contiguous classes vs breaks
- Max contiguous classes
- Later classes
- Holding empty hours for extra curriculars
- Spreading out major and gen ed courses

Step 1: Conglomerate

- [x] Leave space for fun classes
- [x] Front load types of courses
- [x] Complete course by a certain time, or just sooner
- [x] Evenly distribute difficult classes
- [x] Evenly distribute major and gen ed classes
- [x] Has variations also of high performance
- [x] Do uncertain classes sooner (in case they aren't offered again)
- [x] Overlapping with friends schedules (2)
- [x] Prefer Consecutive classes, or prefer breaks
- [x] lunch break (3)
- [x] Late start / Early start
- [x] Consecutive classes in same building (2)

- [x] Weekdays without classes, or just labs (2)
- [x] Take two classes with same prof
- [x] Prefer prof with longer class history
- [x] Don’t have two certain classes at same semester
- [x] Class before even if not prerequisites
- [x] Lighter semester when taking class
- [x] Limit number of major classes
- [x] Hold empty hours for extra curriculars

Step 2: Structure

- [ ] Descriptions of **whole plan**
    - [x] Specify distribution patterns for particular context
        1. Patterns
            1. Front load
            2. Back load
            3. Evenly distribute
        2. Contexts
            1. Global
            2. List of courses
            3. What degree they fulfill (get ed vs major)
            4. Professor
            5. Difficulty?

- [x] Evaluate overlap with another person's plan
    1. Simpler: given their plan
    2. Harder: given their preferences (which also include cyclical mutual preference) generate both plans

- [ ] Of a particular **course**
    - [x] Complete by particular time
    - [x] Complete ASAP
    - [ ] Prefer higher certainty offerings
    - [x] Prefer prof with more experience
        1. In general
        2. In this course
    - [ ] Conditional on another course
        1. Not same semester / same semester
        2. After/before other class
        3. Prefer prof on this class if other class taken with prof

- [ ] Of a particular **day**
    - [ ] Consecutive classes
        1. Prefer consecutive class (aka fewest chunks)
        2. Prefer breaks in between classes (aka most chunks)
        3. Consecutive classes in same building
            1. Harder: or close buildings
    - [ ] Prefer lunch break at certain time range

        - [ ] Late/Early start/end to day
        - [ ] Depending on day of week
            1. Prefer fewer classes
            2. Prefer just labs
        - [ ] Reserve empty hour (eg for extracurriculars)
    - [ ] Of a particular **semester**
        - [ ] If taking class, prefer less credits
    - [ ] Plan in relation to similar plans
        - [ ] High scoring alternatives of uncertain constraints (eg. Tentative schedule)
        - [ ] High scoring worst case senario

Step 3:

Terminology

- **Course**: Meets requirements
- **Offering**: A particular scheduled instance of a course. Specific to one semester, with a set of meetings, instructor, room number, enrolled, certainty
- **Meeting**: Specific to one day in one semester, the start and end times of the class on that specific day
- Time Contexts

Limits the events to the specified time frames

- Semesters
- Days

Examples:

- “*When taking ICS, OS, or Arch*, prefer less classes” ← semester
- "*On Tuesdays and Thursdays*, prefer lighter days” ← days

Course Preferences

Makes choices between different offerings of courses

- Complete by a particular time
- Complete ASAP
- Prefer prof with more experience
    - In general
    - In this course
-

Global Preferences

- Specify whether trying to graduate as early as possible, or in *x* semesters
- Specify distribution
    - Of credits
    - Of courses with particular condition
    - Of particular property of courses (eg difficulty)

General Context Preferences

- Specify max, min, and preferred number of credits per semester

In Relation to Other Plans

- Overlap between two plans (eg prefer sharing classes with someone)

March 17: Preferences (2)

Definitions

- **Scope**
  - Global
  - Semester
  - Day
- **Context**: It is one of the following, specifying
  - The global context
  - A semester context: A set of semesters
  - A day context: A set of semesters, each with a set of days
- **Conditions**: Returns true or false depending on conditions appropriate to scope of current context
  - Global scope eg:
    - `taking "COS 120"`
    - `taking "COS 120" before "COS 121"`
    - `credits > 120`

    -
  - Semester scope eg
    - `taking "COS 120"`
    - `credits > 14`
    - `taking instructor 'Dr. X'`
  - Day scope eg
    - `taking "COS 120"`
    - `Monday`
    - `credits > 4`
    - `minutes > 240`
    - `taking instructor 'Dr. X'`
    -
- **Context Filter**: Given a context, returns a subset of that context
- **Preferences**: Can apply continuous or boolean constraints, applies weight to evaluate score
- **Requirements**: Can only boolean constraints, invalidates plan and adds to violation list when false
- **Boolean Constraints**: Returns true requirement is met, and false otherwise
- **Continuous Constrains**: Return a number between 0 and 1, with 0 implying lowest constraint satisfaction, and 1 implying highest constraint satisfaction

April 3: Implemented Preferences

- Course name in plan
- N course names in plan
- Left before right
- Total credits

- Average start time
- Credits over certain course number
- Courses over certain course number

Future Work

- Support electives
    - Maybe by labeling courses, then creating a context from labels

March 28: Inputs & Outputs

Summary

Input

- Course Catalogs: `catalog.json`
    - Contains information about courses that are not specific to any specific offering
        - Identifier: Department, Course
        - Name
        - Description
        - Offering patterns?
            - Eg. Offered odd springs
            - Dependability? ← implicit in history of catalogs
    - Changes less often

    - eg
    - catalogs
- Course Offerings (aka sections?) `offerings.json`
    - Contains information about individual offerings of courses
        - Identifier: Department, Course, Section, term
        - Professor
        - Enrollment?
        - Max Enrollment
        - Start Date
        - End Date
        - Meeting Days
        - Start, end meeting times
        - Credits
        - Location
    - Eg | eg all
- Requirements & Preferences `preferences.psl`
    - Include degree requirements
        - Required courses
        - Credit counts
        - Prerequesites
    - Specify personal preferences and requirements
- Can include historical data to predict future times and determine confidence

Output:

- Generated Plans: `plans.json`
  - Selection of offerings that fulfill the requirements
  - Required information:
    - List of (Department, Course, Section) that are scheduled in the plan
    - Plan Score
  - Maybe include
    - List of preferences and their satisfaction
    - Info answering "*What if preferences or requirements were different?*"

In: `catalog.json`

Example source of data

- catalog-year
- departments
  - name
  - prefix
  - description
  - courses
    - name
    - number
    - description
    - offering-pattern

- *(prerequisites)*
- *(tags)*

`test-catalog.json`

```
{
    "$schema": "./catalog-schema.json",
    "catalog-year": "2021-2022",
    "departments": [
        {
            "name": "Computer Science",
            "prefix": "COS",
            "description": "The best department at computers",
            "courses": [
                {
                    "name": "Information Technology Concepts",
                    "number": 101,
                    "description": "The ... students."
                },
                {
                    "name": "Introduction to Computational Problem Solving",
                    "number": 120,
                    "description": "Approaches ... requirement.",
                    "offering-pattern": "fall"
                },
                {
                    "name": "Foundations of Computer Science",
                    "number": 121,
```

```
                    "description": "including ... 120.",
                    "prerequisites": ["COS 120"],
                    "offering-pattern": "always"
                }
            ]
        }
    ]
}
```

In: `offerings.json`

- Catalog Year
- Term
    - Department: Course Number: Tag
        - [
            - Section
            - Car
            - Type
            - Credits
            - Professor
            - Location
                - building
                - Room-number
            - NumEnrolled

        - MaxEnrolled
        - *(Start Date)*
        - *(End Date)*
        - Meeting Times
            - [
                - Weekdays
                - Start Time
                - End Time
            -]
    -]

`test-offerings.json`

```
{
    "$schema": "./offerings-schema.json",
    "catalog-year": "2021-2022",
    "winter" : {
        "COS 120" : [
            {
                "section": 1,
                "type": "lecture",
                "credits": 3,
                "professors": ["Dr. X", "Dr. Y"],
                "location": {
                    "building": "Euler",
                    "room-number": "217"
```

```
                },
                "num-enrolled": 0,
                "max-enrollment": 0,
                "start-date": "2021-03-22",
                "meetings": [
                    {
                        "days": ["Monday", "Wednesday", "Friday"],
                        "start-time": "12:00 PM",
                        "end-time": "12:50 PM"
                    }
                ]
            }
        ]
    }
}
```


Out: `plans.json`

- [
  - score
  - comments
  - terms
    - [
      - term

        - year
        - Sections
            - [
                - CRN
            -]
    -]
-]

`test-plan.json`

```
    "$schema": "./schema-plan.json",
    "plans": [
        {
            "score": 100,
            "comments": "None at all",
            "terms": [
                {
                    "term": "spring",
                    "year": 2021,
                    "sections": [ 11013, 23412, 234123, 12313, 1231, 1231 ]
                }
            ]
        }
    ]
}
```

plan.json

```
"$schema": "./schema-plan.json",
    "plans": [
        {
            "score": 100,
            "comments": "None at all",
            "terms": [
                {
                    "term": "spring",
                    "year": 2021,
                    "sections": [
                       "COS 120" : [
            {
                "section": 1,
                "type": "lecture",
                "credits": 3,
                "professors": ["Dr. X", "Dr. Y"],
                "location": {
                    "building": "Euler",
                    "room-number": "217"
                },
                "num-enrolled": 0,
                "max-enrollment": 0,
                "start-date": "2021-03-22",
```

```
                "meetings": [
                    {
                        "days": ["Monday", "Wednesday", "Friday"],
                        "start-time": "12:00 PM",
                        "end-time": "12:50 PM"
                    }
                ]
            }
        ]
                    ]
                }
            ]
        }
    ]
}
```

Spring Timeline

Spring Goals

- Lit Survey
- Understand and implement preferences
- **Implement at least 1 rudimentary plan generation algorithm**

Weeks 1 - 5

(Feb 1-Mar 8)

- [x] Choose Topic
- [x] Literature Survey

Week 6

(Mar 15)

- [x] Create Timeline
- [x] Preference collection
  - [x] Perform rough survey of self, people, and literature
  - [x] Collect and organize any potential preference considerations
- [x] *Preference specification language: Grammar*

Week 7

(Mar 22)

- [x] Gather input/output expectations
  - [x] Design JSON format
  - [x] Create JSON Schema
  - [x] Create example JSON inputs

- [x] Create example JSON outputs

Week 8

(Mar 29)

- [x] System language choice: Java
- [x] Implement preferences

Week 9

(Apr 5)

- [x] *Preference specification language: Parser*
- [ ] Generate conflict-less event sequences data structure

Week 10

(Apr 12)

- [ ] Generate prerequisite tree

Week 11

(Apr 19)

- [ ] Generate search space based on event times, prerequisites, and any possible requirements

Week 12

(Apr 26)

- [x] *Preference specification language: Translator*
- [ ] Brute-force genetic algorithm, backtracking

Week 13

(May 3)

- [x] Semester archive paper

Week 14

(May 10)

- [x] Semester archive paper

Nurk Notes

- Musts and wants
- Multidimensional optimization, simplex method, **temporal reasoning**, **temporal logic**

- Constraint satisfaction, constraint propagation (crossword example)
- Iterative deepening?
- JSON schemas
- Prefixes should belong to course
- Objectify generated python
- Use `with` for contexts

Ask Nurk

-
  -

Is there a python library “python logic programming”

- Temporal reasoning, logic
- [Lecture 12 Linear temporal logic - YouTube]

Computation tree logic

Temporal Reasoning

Terms

- **Linear Temporal Logic (LTL)**: wikipedia
- **Computational Tree Logic (CTL)**: wikipedia
- CTL*
- **Trace:** a single sequence of events
- **Planning Domain Definition Language** (PDDL):

Links

- MIT Advanced 6. Planning with Temporal Logic
    - Temporal logic allows goals to be satisfied by a sequence of states
    - Always eventually be met (eg. Always need a future gas station visit)
    - **Linear time** reasons about one path
        - Can reason about the future, or can flip to reason about the past
    - **Branching time**, all possible future possibilities are considered
    - Video continues with linear time for planning purporses
    - $f := true | p_i | f_i \wedge f_j | \neg f_i | X f_i | f_i \cup f_j$
    - $X \rightarrow$ next
    - $U \rightarrow$ until
    - $F \rightarrow$ future/eventually
    - $G \rightarrow$ globally (true for all)
    - $R \rightarrow$ release (like until, but must overlap for one point)

- Combinations
  - Infinitely often: GFp
  - Eventually forever: FGp
- Planning
  - Using PDDL3 for goal description
  - Buchi Automata