Universidad Nacional de San Agustín
Facultad de Ingeniería de Producción y Servicios
Escuela Profesional Ciencia de la Computación

Trabajo Interdiciplinar III

Sistema Automatizado de Generación de Horarios Académicos mediante Algoritmos Genéticos: DataSet UniTime

INTEGRANTES:

Quispe Huanca, Joselyn Lizeth
Pardavé Espinoza, Christian
Montoya Choque, Leonardo

DOCENTE:

Yessenia Deysi Yari Ramos

AREQUIPA - PERÚ

2025

Índice

1. Conjunto de Datos UniTime

1.1. Introducción

El problema de *University Course Timetabling* consiste en asignar clases a tiempos y aulas disponibles, cumpliendo restricciones de recursos, evitando conflictos entre estudiantes e intentando respetar preferencias de profesores, departamentos y ubicaciones.

En el caso de la Universidad de Purdue, se trata de una institución grande con aproximadamente 39,000 estudiantes, 9,000 clases y 570 espacios de enseñanza, lo que genera alrededor de 259,000 solicitudes individuales de clases. Debido a esta magnitud, el problema se divide en múltiples subproblemas que se resuelven a nivel departamental, manteniendo coordinación central.

1.2. Descomposición del problema

El problema general se divide en tres subproblemas principales:

1.2.1. Large Lecture Room Problem

Corresponde a las clases más grandes de la universidad (alrededor de 800), asignadas a 55 aulas con capacidades de hasta 474 asientos. Estas clases son compartidas por varios departamentos, por lo que este subproblema es muy denso y se resuelve en primer lugar, ya que su solución afecta a los demás.

1.2.2. Departmental Problems

Cada departamento genera y resuelve su propio horario, con aproximadamente 70 subproblemas de entre 10 y 500 clases cada uno. Cada departamento puede crear varias soluciones y luego "*commitear*" (confirmar) una versión final, siempre que no entre en conflicto con horarios previamente comprometidos de otros departamentos. Además, se permite definir matrices de uso compartido de aulas entre departamentos o incluso combinar varios subproblemas en uno mayor.

1.2.3. Computer Laboratory Problem

Incluye aproximadamente 450 clases que se imparten en 36 laboratorios pequeños (20 a 45 asientos). Se resuelve al final del proceso, considerando los horarios ya establecidos de las clases grandes y departamentales.

1.3. Mecanismo de Commit

Cada departamento puede almacenar varias soluciones. Al confirmar una solución (*commit*), el sistema verifica que no existan conflictos con otros horarios ya confirmados. De esta forma, cada problema considera las restricciones impuestas por los problemas ya resueltos, manteniendo la coherencia global. Esto permite una construcción distribuida de horarios con coordinación central.

1.4. Modelo de representación

Purdue utiliza un conjunto de *patrones estándar de reunión* que simplifican la representación del problema:

- Clases de 1 hora tres veces por semana (Lunes-Miércoles-Viernes).

- Clases de 1.5 horas dos veces por semana (Martes-Jueves).
- Clases de 2 o 3 horas una vez por semana en bloques fijos.

Cada clase se modela mediante una única variable, cuyo dominio contiene todas las combinaciones válidas de tiempo, aula e instructor. Cada valor del dominio codifica:

- Patrón de tiempo (por ejemplo, Lunes-Miércoles-Viernes).
- Hora de inicio.
- Aula asignada.
- Instructor.
- Preferencias o penalizaciones asociadas (tiempo, aula, equipamiento, etc.).

1.5. Restricciones del problema

1.5.1. Hard Constraints

- **Restricciones de recursos:** Un profesor o una sala no pueden tener dos clases simultáneamente.
- **Restricciones de grupo:** Determinan relaciones entre clases, como evitar que dos secciones del mismo curso coincidan o requerir que ciertas clases sean consecutivas.
- **Distancias excesivas:** Dos clases consecutivas impartidas por el mismo profesor no pueden estar a más de 200 metros de distancia.

1.5.2. Soft Constraints

- **Preferencias de tiempo y aula:** Se busca maximizar la satisfacción de las preferencias definidas por departamentos o instructores.
- **Conflictos de estudiantes:** Minimizar solapamientos entre clases inscritas por un mismo estudiante.
- **Distancia entre clases consecutivas:**
  - Para profesores: 0 m (sin penalización), 0–50 m (desalentado), 50–200 m (fuertemente desalentado).
  - Para estudiantes: hasta 670 m (aceptable, 10 minutos de cambio), hasta 1000 m si la clase anterior dura 90 minutos.
- **Balance departamental:** Cada departamento debe utilizar horarios de forma equilibrada, evitando concentraciones en la mañana o la noche.
- **Eficiencia de uso de aulas:** Minimizar huecos de menos de una hora entre clases en una misma sala y evitar clases que ocupen menos de dos tercios de la capacidad.

1.6. Asignación de estudiantes (Student Sectioning)

Muchos cursos tienen múltiples secciones (por ejemplo, una clase magistral, recitaciones y laboratorios). Las principales reglas son:

- Cada clase tiene un límite máximo de estudiantes.
- Cada estudiante debe estar inscrito en exactamente una sección de cada subparte del curso.
- Si existen relaciones jerárquicas (por ejemplo, laboratorio dependiente de clase magistral), deben cumplirse.

Inicialmente se realiza una asignación homogénea de estudiantes (*Carter's homogeneous sectioning*) para minimizar conflictos futuros. Posteriormente, puede aplicarse un algoritmo de reasignación (*re-sectioning*) al finalizar la búsqueda para mejorar el resultado.

1.7. Criterio de optimización

El solucionador busca minimizar una función de costo compuesta por:

- Penalizaciones por violaciones de *soft constraints*.
- Penalizaciones por conflictos entre estudiantes.
- Penalizaciones por desequilibrio departamental.

Todas las *hard constraints* deben cumplirse estrictamente. El resultado final es un horario factible y equilibrado que optimiza la satisfacción de preferencias y el uso eficiente de recursos.

2. Estructura del Conjunto de Datos

2.1. Formato de Datos v2.4 del University Course Timetabling

El formato de datos v2.4 del *University Course Timetabling Problem* es casi idéntico al formato v2.3, con algunas modificaciones importantes que afectan la forma en que se representan los estudiantes, las restricciones y la estructura general del archivo XML.

2.1.1. Principales cambios respecto a la versión 2.3

- **Ponderación de estudiantes:** A partir de esta versión, cada estudiante puede tener una `weight` o ponderación asociada dentro de su elemento `offering`. Esto permite modelar diferencias en la proyección de matrícula o tamaño esperado de grupos. En otras palabras, no todos los estudiantes tienen el mismo peso en la optimización: algunos cursos pueden tener mayor prioridad o carga esperada.
- **Nuevos tipos de restricciones grupales:** Se incorporan cuatro nuevas restricciones que permiten mayor control sobre la secuencia y distribución de clases:
  - `NDB_GT_1`: garantiza que un grupo no esté asignado más de una vez en un mismo día.
  - `CH_NOTOVERLAP`: impide que ciertos cursos se superpongan en horario.
  - `FOLLOWING_DAY`: fuerza que dos cursos se dicten en días consecutivos.
  - `EVERY_OTHER_DAY`: exige que dos clases se impartan en días alternos (por ejemplo, lunes, miércoles, viernes).

2.1.2. Estructura general del archivo XML

El archivo tiene la siguiente estructura base:

```
<?xml version="1.0" encoding="UTF-8"?>
<timetable
  version="2.4"
  initiative="puWestLafayetteTrdtn"
  term="2007Fal"
  created="Mon Mar 26 14:42:12 EDT 2007"
  nrDays="7"
  slotsPerDay="288">
```

Los atributos principales del elemento raíz `timetable` son:

- `version`: versión del formato (2.4 en este caso).
- `initiative`: designación del campus o institución.
- `term`: periodo académico (por ejemplo, `2007Fal` para otoño 2007).
- `created`: fecha de creación del archivo.
- `nrDays`: número de días por semana (por defecto 7, de lunes a domingo).
- `slotsPerDay`: número de *time slots* por día (por defecto 288, correspondientes a intervalos de 5 minutos desde medianoche hasta medianoche).

2.1.3. Definición de aulas

Cada aula se define dentro del bloque `<rooms>`, donde cada elemento `<room>` contiene:

- `id`: identificador único de la sala.
- `capacity`: capacidad de asientos.
- `location`: coordenadas $(x,\ y)$ del edificio o sala, utilizadas para calcular distancias en metros:

$$d = 10 \times \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- `constraint`: determina si la sala debe respetar restricciones de horario (por defecto, `true`).
- `discouraged`: si es `true`, se penaliza el uso de esa sala.
- `ignoreTooFar`: indica si deben ignorarse las restricciones de distancia respecto a otras aulas.

Si una sala tiene disponibilidad parcial o se comparte entre departamentos, incluye el elemento `<sharing>`, que define:

- `pattern`: matriz de disponibilidad, codificada como cadena de caracteres.
- `unit`: cantidad de *slots* representados por cada carácter del patrón (por ejemplo, 6 para bloques de 30 minutos).
- `freeForAll`: tiempo disponible para todos los departamentos.
- `notAvailable`: tiempos en que la sala no puede usarse.
- `department`: asignación de códigos de departamento a los valores del patrón.

2.1.4. Definición de instructores

El conjunto de instructores no se define explícitamente. En cambio:

- Las clases que comparten un mismo docente se refieren al mismo identificador de instructor.
- Las restricciones o preferencias de disponibilidad se heredan directamente por las clases asociadas (por ejemplo, si un profesor no trabaja los viernes, sus clases no tendrán ubicaciones de tiempo en viernes).
- Si un instructor ya tiene clases asignadas en otro problema con solución comprometida, estas clases se incluyen en el archivo como *committed classes*.

2.2. Definición de Clases

Cada elemento `<class>` describe una clase específica, sus restricciones, instructores, salas y horarios posibles. Es una de las partes centrales del archivo XML.

- **id**: identificador único de la clase.
- **offering**: identificador de la oferta instruccional a la que pertenece (unidad superior).
- **config**: identificador de la configuración de la oferta (una oferta puede tener múltiples configuraciones).
- **subpart**: identificador de la subparte de programación (por ejemplo, conferencia, laboratorio, recitación).
- **committed**: indica si la clase pertenece a una solución ya existente sobre la que se construye la actual.
- **classLimit**: número máximo de estudiantes que puede tener la clase.
- **minClassLimit, maxClassLimit**: límites mínimo y máximo de estudiantes (si difieren, se aplica la restricción `CLASS_LIMIT`).
- **dates**: cadena binaria que indica los días del semestre en los que puede impartirse la clase.
- **scheduler** / **department**: identificadores del departamento encargado de gestionar o controlar la clase.

Cada clase contiene los siguientes subelementos:

2.2.1. Instructores

- `<instructor id="..."/>`: indica qué instructor está asignado a la clase.
- El atributo `solution="true"` indica la asignación del instructor en la solución actual.

2.2.2. Salas

- `<room id="..."pref="..."/>`: lista las salas válidas para la clase.
- `pref`: preferencia de uso de la sala (valor numérico, menor es mejor).
- `solution="true"` indica la sala asignada en la solución.

2.2.3. Horarios

- `<time days="1010100"start="90"length="12"pref="0.0/>`
- **days**: cadena binaria que representa los días (Lunes–Domingo).
- **start**: índice del intervalo de tiempo en el día (cada slot son 5 minutos).
- **length**: duración de la clase en número de slots.
- **pref**: preferencia del horario (valor negativo indica preferencia fuerte).

Dos horarios se superponen si:

- Las fechas (`dates`) se solapan.
- Los días (`days`) coinciden.
- El tiempo se solapa en el rango (`start + length`).

Las clases con el mismo instructor no pueden superponerse. Si son consecutivas (back-to-back), se evalúa la distancia entre salas:

- $0 < d \leq 50$m: desalentado.
- $50 < d \leq 200$m: fuertemente desalentado.
- $d > 200$m: prohibido.

—

2.3. Restricciones Grupales (Group Constraints)

Las restricciones grupales permiten definir relaciones entre clases. Cada restricción se define como:

```
<constraint id="..." type="..." pref="...">
    <class id="..."/>
</constraint>
```

- **id**: identificador único de la restricción.
- **type**: tipo de restricción (`BTB`, `DIFF_TIME`, `CLASS_LIMIT`, etc.).
- **pref**: nivel de preferencia o requerimiento:
  - R = requerida (hard constraint)
  - P = prohibida (hard constraint)
  - -1 = preferida
  - -2 = fuertemente preferida
  - 1 = desalentada
  - 2 = fuertemente desalentada

La restricción `CLASS_LIMIT` controla que:

- El límite de cada clase sea menor o igual a su capacidad de sala.

- La suma de los límites de clases hijas alcance el mínimo de su clase padre.
- Las clases raíz (sin padre) cumplan con el límite de curso definido por `courseLimit`.

Estas restricciones garantizan coherencia entre jerarquías de clases y balance entre departamentos o subpartes.

—

2.4. Definición de Estudiantes

El bloque `<students>` define la demanda y asignación de estudiantes a clases.

```
<student id="4">
    <offering id="701" weight="1.3200"/>
    <class id="1669"/>
    <prohibited-class id="1702"/>
</student>
```

- **id**: identificador del estudiante.
- **offering**: curso solicitado (por id).
- **weight**: ponderación de demanda (proyección de matrícula; la suma de pesos no debe superar el `classLimit`).
- **class**: clases en las que está inscrito.
- **prohibited-class**: clases en las que no puede inscribirse (por conflictos previos).

Los estudiantes pueden:

- Estar inscritos en clases comprometidas (`committed=true`).
- Tener clases prohibidas por restricciones de horario o capacidad.
- Participar en la sección final (*student sectioning*) del problema de asignación, donde se optimiza minimizar conflictos de horario y maximizar balance.