**UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS**

Logotipo

Descripción generada automáticamente

**TÓPICOS EN CIENCIAS DE LA COMPUTACIÓN**

**Sección CC82**

**PROFESOR**

Canaval Sanchez, Luis Martín

**INFORME DE TRABAJO PARCIAL**

|  |  |
| --- | --- |
| Integrantes del equipo: | |
| Código | Nombres y apellidos |
| U202116247 | Rodriguez Calla, Jair Stephano |
| U202111299 | Zuñiga Lovera, Angel Ruben |
| U202111197 | Adauto Angulo, Mauro Imanol Obermeyer |
| U202112116 | Niño Suárez, David Joaquín |

**Agosto, 2024**

**Contenido**

[**1.** **Problemática y Motivación** 3](#_Toc181660743)

[**1.1.** **Problemática** 3](#_Toc181660744)

[**1.2.** **Motivación** 4](#_Toc181660745)

[**2.** **Objetivos** 5](#_Toc181660746)

[**2.1.** **Objetivo General** 5](#_Toc181660747)

[**2.2.** **Objetivos Específicos** 5](#_Toc181660748)

[**3.** **Diseño de Solución** 6](#_Toc181660749)

[**3.1.** **Modelado del Problema con CSP** 6](#_Toc181660750)

[**3.2.** **Implementación en Python usando python-constraint** 6](#_Toc181660751)

[**3.3.** **Optimización de la Solución** 7](#_Toc181660752)

[**3.4.** **Validación y Pruebas** 7](#_Toc181660753)

[**3.5.** **Resultados Esperados** 7](#_Toc181660754)

[**4.** **Desarrollo de Solución** 8](#_Toc181660755)

[**4.1.** **Definición del Problema y Variables** 8](#_Toc181660756)

[**4.2.** **Definición de Restricciones** 8](#_Toc181660757)

[**4.3.** **Búsqueda de Soluciones y Selección de la Mejor Opción** 9](#_Toc181660758)

[**5.** **Diseño de Solución Actualizada** 10](#_Toc181660759)

[**6.** **Plan de Validación** 12](#_Toc181660760)

[**7.** **Conclusión** 13](#_Toc181660761)

# **Problemática y Motivación**

# **Problemática**

En el entorno educativo, una de las tareas más complejas y críticas es la planificación y asignación de horarios para los diferentes cursos que se imparten. Esta planificación implica coordinar a profesores, estudiantes y aulas disponibles, asegurando que no existan conflictos de horario y que se cumplan con todas las restricciones operativas de la institución. La complejidad de esta tarea aumenta proporcionalmente con el número de cursos, profesores y aulas disponibles, y cualquier error en la asignación puede tener consecuencias significativas, como:

1. **Conflictos de Horarios:**
   * Un profesor asignado a dos cursos en el mismo horario.
   * Un aula asignada a más de un curso al mismo tiempo.
   * Estudiantes que deben asistir a dos clases diferentes en el mismo horario.
2. **Subutilización de Recursos:**
   * Aulas vacías durante gran parte del horario lectivo.
   * Profesores con cargas horarias desbalanceadas, con algunos sobrecargados y otros subutilizados.
3. **Sobrecapacidad en Aulas:**
   * Asignación de un aula pequeña a un curso con muchos estudiantes, superando la capacidad física y logística del espacio.
4. **Dificultad para Ajustar Cambios:**
   * Cualquier cambio, como la adición de un nuevo curso o la modificación de la disponibilidad de un profesor, puede requerir una reestructuración significativa de todo el horario.

Estas dificultades, además de afectar la operatividad diaria de la institución, impactan negativamente en la experiencia educativa de estudiantes y docentes. La tarea de asignación de horarios se suele realizar manualmente o utilizando herramientas básicas como hojas de cálculo, lo cual es ineficiente y propenso a errores, especialmente en instituciones con un gran número de cursos y restricciones.

# **Motivación**

La asignación de horarios es un problema típico de optimización que puede ser abordado mediante técnicas de programación con restricciones (Constraint Programming, CSP), ya que estas permiten modelar de manera explícita todas las restricciones del problema y buscar una solución que las satisfaga de manera óptima.

**Razones para utilizar CSP en la asignación de horarios:**

1. **Automatización de la Asignación:**
   * Utilizar CSP para la asignación de horarios permite automatizar un proceso que de otra manera sería tedioso y propenso a errores. Con la definición adecuada de las restricciones, el sistema puede generar un horario completo y consistente en cuestión de segundos.
2. **Optimización de Recursos:**
   * El uso de CSP permite no solo cumplir con todas las restricciones, sino también optimizar el uso de recursos, como asegurar que no haya aulas vacías en horarios pico o balancear la carga horaria entre los profesores.
3. **Flexibilidad y Adaptabilidad:**
   * Un modelo CSP bien diseñado permite realizar cambios en las restricciones de manera sencilla. Por ejemplo, si un profesor cambia su disponibilidad o se añade un nuevo curso, el sistema puede reajustar el horario completo sin necesidad de rehacerlo desde cero.
4. **Reducción de Conflictos:**
   * Al modelar explícitamente todas las restricciones, el sistema asegura que no haya conflictos de horarios entre profesores, estudiantes y aulas. Esto reduce significativamente los problemas operativos que suelen surgir al inicio de cada ciclo lectivo.
5. **Mejora en la Calidad Educativa:**
   * Al evitar problemas de sobrecarga de profesores y aulas, y al asegurar que los estudiantes puedan asistir a todas sus clases sin conflictos, se mejora la calidad del entorno educativo, permitiendo una mejor experiencia tanto para estudiantes como para profesores.

# **Objetivos**

# **Objetivo General**

Desarrollar un modelo de programación con restricciones (CSP) para la asignación óptima de horarios en una escuela, asegurando que todas las restricciones de disponibilidad de profesores, capacidad de aulas y conflictos de horarios entre cursos sean respetadas, y que se minimicen los conflictos y la subutilización de recursos.

# **Objetivos Específicos**

1. **Modelar las restricciones del problema en un modelo CSP:**
   * Definir las variables, dominios y restricciones necesarias para representar el problema de asignación de horarios, asegurando que se contemplen la disponibilidad de profesores, capacidad de aulas y evitar conflictos de horarios.
2. **Implementar el modelo CSP en Python utilizando la biblioteca python-constraint:**
   * Desarrollar el modelo en código, integrando todas las restricciones identificadas y generando soluciones válidas para la asignación de horarios que respeten todas las condiciones del problema.
3. **Validar y optimizar las soluciones generadas:**
   * Probar el modelo CSP con datos de prueba, verificar que no existan conflictos en las soluciones obtenidas y optimizar el uso de recursos (aulas y profesores), buscando el balance óptimo entre las restricciones y la eficacia del horario generado.

# **Diseño de Solución**

El diseño de solución se centra en la implementación de un modelo de Programación con Restricciones (CSP) para la asignación óptima de horarios en una institución educativa. El objetivo principal es asegurar que todas las restricciones de disponibilidad de profesores, capacidad de aulas y conflictos de horarios entre cursos sean respetadas. A continuación, se detallan los pasos y componentes clave del diseño de solución:

# **Modelado del Problema con CSP**

* **Definición de Variables:**
  + Se definen variables para cada curso, las cuales representan la asignación de un aula, un horario y un día específico.
* **Dominios:**
  + Cada variable tiene un dominio que incluye todas las combinaciones posibles de aulas, horarios y días en los que se puede asignar el curso.
* **Restricciones:**
  + Se modelan varias restricciones para garantizar que las soluciones generadas sean válidas y óptimas:
    - **Capacidad de Aulas:** Asegura que la cantidad de estudiantes no exceda la capacidad máxima del aula asignada.
    - **Conflictos de Horarios:** Evita que más de un curso se asigne al mismo aula, horario y día.
    - **Distribución Equitativa:** Garantiza que se utilicen todas las aulas disponibles y que no se concentren todos los cursos en un solo día o aula.
    - **Disponibilidad de Profesores:** Asegura que los profesores no se asignen a más de un curso al mismo tiempo.

# **Implementación en Python usando python-constraint**

* **Configuración del Modelo:**
  + Se utiliza la biblioteca python-constraint para modelar el problema con todas las restricciones identificadas.
* **Definición del Problema:**
  + Se crea un objeto Problem() y se añaden variables, dominios y restricciones mediante funciones addVariable() y addConstraint().
* **Validación de Soluciones:**
  + Se utiliza el método getSolutions() para obtener todas las soluciones posibles que cumplan con las restricciones definidas.

# **Optimización de la Solución**

* **Evaluación de Soluciones:**
  + Se aplica un criterio de evaluación para seleccionar la mejor solución en función de la distribución equilibrada de aulas y horarios.
* **Penalización y Selección:**
  + Se utiliza una función de puntaje para penalizar las soluciones que no cumplen con la distribución ideal de horarios y aulas. La solución con el puntaje más bajo es seleccionada como la mejor.

# **Validación y Pruebas**

* **Pruebas de Consistencia:**
  + Se prueban diferentes combinaciones de cursos, aulas y horarios para asegurar que el modelo puede manejar casos complejos y garantizar la validez de las soluciones generadas.
* **Evaluación de Escenarios de Cambio:**
  + Se simulan cambios en la disponibilidad de profesores o la adición de nuevos cursos para evaluar la adaptabilidad del modelo CSP.
* **Pruebas de Consistencia de Agentes**:
  + Explica que se realizan pruebas unitarias para asegurar que cada agente (Aula, Horario, Optimización) cumpla con su función específica.
* **Pruebas de Integración de Agentes**:
  + Describe cómo los agentes interactúan y se integran para asegurar que el sistema multiagente funcione de manera coordinada, resolviendo las asignaciones sin conflictos.
* **Pruebas de Rendimiento**:
  + Menciona que pruebas de carga y rendimiento se realizarán para verificar que el sistema multiagente puede manejar aumentos en el número de cursos y restricciones.

# **Resultados Esperados**

* **Optimización el Uso de Recursos:**
  + Se espera una utilización óptima de las aulas y una distribución balanceada de la carga horaria de los profesores.
* **Reducción de Conflictos:**
  + Al modelar explícitamente todas las restricciones, se minimizan los conflictos de horario y se evita la sobrecarga o subutilización de recursos.
* **Adaptabilidad a Cambios:**
  + El modelo puede ajustarse fácilmente a cambios en las restricciones o a la adición de nuevas variables sin necesidad de reestructurar completamente el horario.

# **Desarrollo de Solución**

Se definieron tres aulas con capacidades variables, y se generaron aleatoriamente la cantidad de estudiantes para cada curso, asegurando que el total de estudiantes se imprimiera en consola.

Luego, se establecieron restricciones para garantizar que ningún curso excediera la capacidad de su aula asignada y que no se usara un aula por más de un curso a la vez en el mismo horario y día.

Después de buscar todas las soluciones posibles, se calculó el uso de aulas en cada asignación y se identificó la mejor solución, que minimiza la cantidad de aulas utilizadas, presentando finalmente la asignación óptima de cursos junto con sus respectivas aulas, horarios y días.

*Código del desarrollo:* <https://github.com/mauroadauto6/TopicosCC-TP-TF.git>

# **Definición del Problema y Variables**

En esta etapa, se definieron las variables principales del problema: cursos, aulas, horarios y días. Además, se especificaron las capacidades de cada aula y se generó aleatoriamente la cantidad de estudiantes para cada curso.Texto

Descripción generada automáticamente

Este código establece los elementos básicos para la asignación de horarios y aulas, asegurando que cada curso tenga su respectiva cantidad de estudiantes y que las aulas cuenten con límites de capacidad. Estas variables son fundamentales para modelar el problema.

# **Definición de Restricciones**

Las restricciones del problema fueron modeladas a través de agentes que aseguran la validez de las asignaciones:

**Restricción de capacidad de aulas:**

Cada curso debe ser asignado a un aula con capacidad suficiente para los estudiantes inscritos.

Texto

Descripción generada automáticamente

**Restricción de conflictos de horarios:**

No se permite asignar el mismo aula en el mismo horario y día a más de un curso.

Texto

Descripción generada automáticamente

Con estas restricciones, el sistema asegura que las soluciones respeten tanto la capacidad física de las aulas como la distribución de horarios.

# **Búsqueda de Soluciones y Selección de la Mejor Opción**

Para buscar y seleccionar las mejores soluciones, se utilizó un agente de optimización. Este agente evalúa las soluciones generadas y selecciona aquella que maximiza el uso de diferentes aulas.

**Implementación del agente de optimización:**

**Texto

Descripción generada automáticamente**

**Integración y ejecución del sistema:**

**Texto

Descripción generada automáticamente**

# **Validación y Pruebas**

Para garantizar el correcto funcionamiento del sistema, se diseñaron pruebas específicas para cada agente y su interacción conjunta. Estas pruebas permiten verificar que las restricciones se cumplan y que el sistema genere soluciones válidas.

**Generación del dominio de variables:**

Se utilizó una función para generar el dominio de combinaciones posibles de aulas, horarios y días, lo que permite asignar todas las combinaciones a las variables del problema.

Texto, Escala de tiempo

Descripción generada automáticamente

**Pruebas unitarias para agentes:**

**Prueba del Agente de Aula:** Evalúa que los cursos sean asignados a aulas con capacidad suficiente.

**Prueba del Agente de Horario:** Verifica que no existan conflictos de horario entre los cursos asignados.

Texto

Descripción generada automáticamente

**Prueba de integración:**

* Evalúa cómo los agentes trabajan en conjunto para generar soluciones válidas. Se asegura que las restricciones de capacidad y horarios sean respetadas.

Texto

Descripción generada automáticamente

# **Diseño de Solución Actualizada**

Con el objetivo de mejorar la eficacia y flexibilidad del sistema de asignación de horarios, se ha implementado un sistema multiagente que permite dividir el problema en tareas específicas manejadas por agentes independientes. Un sistema multiagente es una arquitectura compuesta por múltiples agentes, cada uno responsable de cumplir con una función específica, colaborando para lograr una solución óptima de forma conjunta. En esta solución, cada agente gestiona una parte del proceso de asignación de horarios, mejorando la adaptabilidad y permitiendo una resolución eficiente de restricciones.

**5.1. Descripción de los Agentes**

1. **Agente de Aula:**
   * Este agente es responsable de asignar aulas a los cursos, asegurando que cada aula tenga la capacidad suficiente para los estudiantes inscritos en el curso correspondiente.
   * Al gestionar la capacidad de las aulas, este agente ayuda a evitar problemas de sobrecarga y garantiza que los recursos físicos sean usados de manera eficiente.
2. **Agente de Horario:**
   * Este agente se encarga de coordinar los horarios de los cursos. Su función es asegurar que no existan conflictos de horarios para los cursos asignados a la misma aula o profesor.
   * Este control evita situaciones en las que un recurso (aula o profesor) esté doblemente asignado en el mismo horario.
3. **Agente de Optimización:**
   * Este agente evalúa las soluciones generadas por los otros agentes y selecciona la mejor alternativa según criterios de optimización.
   * Los criterios incluyen el uso eficiente de las aulas, la distribución equilibrada de la carga horaria y el cumplimiento de todas las restricciones de capacidad y disponibilidad.

**5.2. Interacción entre los Agentes**

Los agentes en este sistema multiagente colaboran entre sí para asegurar que la solución final sea válida y eficiente. La interacción entre los agentes se realiza de la siguiente manera:

1. **Agente de Aula:**
   * Este agente valida que las aulas asignadas a los cursos cumplan con las restricciones de capacidad.
   * Proporciona información al Agente de Horario sobre las aulas disponibles para que este pueda coordinar las asignaciones de manera eficiente.
2. **Agente de Horario:**
   * Este agente se asegura de que no existan conflictos de horarios en la asignación de aulas y cursos.
   * Una vez generadas las posibles soluciones, comunica al Agente de Optimización las combinaciones que cumplen con las restricciones de horarios y capacidad.
3. **Agente de Optimización:**
   * Este agente evalúa todas las soluciones válidas proporcionadas por los agentes anteriores y selecciona la mejor alternativa.
   * La solución seleccionada optimiza el uso de los recursos disponibles, garantizando una distribución equilibrada de aulas y horarios.

**5.3. Ventajas del Diseño Actualizado**

Este diseño presenta las siguientes ventajas:

* **Eficiencia:** Divide el problema en tareas específicas, lo que permite una resolución modular y optimizada.
* **Flexibilidad:** Facilita la implementación de cambios en las restricciones (por ejemplo, agregar nuevos cursos o modificar horarios).
* **Escalabilidad:** El sistema puede manejar un mayor número de variables y restricciones sin pérdida significativa de rendimiento.
* **Colaboración:** La interacción entre los agentes asegura que todas las restricciones sean respetadas y que la solución final sea óptima.

# **Plan de Validación**

Para asegurar que el sistema multiagente cumpla con sus objetivos y funcione correctamente, se han diseñado varias pruebas que permiten validar el comportamiento de cada agente de manera individual, su interacción conjunta y el rendimiento del sistema bajo diferentes condiciones. A continuación, se describen las pruebas realizadas y los criterios de éxito definidos.

**6.1. Definición de Pruebas**

1. **Pruebas Unitarias para Agentes:**
   * Estas pruebas tienen como objetivo validar el correcto funcionamiento de cada agente de forma independiente:
     + **Agente de Aula:** Verifica que todos los cursos asignados a un aula cumplen con las restricciones de capacidad.
     + **Agente de Horario:** Evalúa que no se produzcan conflictos de horarios en las aulas y que cada curso esté asignado a un horario único, sin superposiciones.
     + **Agente de Optimización:** Prueba que las soluciones seleccionadas optimizan el uso de diferentes aulas y garantizan una distribución equilibrada de los recursos disponibles.
2. **Pruebas de Integración de Agentes:**
   * Estas pruebas verifican cómo interactúan los agentes para resolver el problema de manera conjunta:
     + **Interacción entre Agente de Aula y Agente de Horario:** Se asegura que las aulas asignadas respeten las restricciones de capacidad y no generen conflictos de horarios.
     + **Interacción con el Agente de Optimización:** Valida que las soluciones generadas por los agentes de Aula y Horario sean correctamente evaluadas y optimizadas por el Agente de Optimización para seleccionar la mejor alternativa.
3. **Pruebas de Rendimiento:**
   * Estas pruebas evalúan la capacidad del sistema para manejar volúmenes mayores de datos y restricciones:
     + **Escalabilidad:** Incremento del número de cursos, aulas y horarios para observar cómo el sistema gestiona el aumento de la complejidad.
     + **Adaptabilidad:** Pruebas de carga para verificar que el sistema pueda adaptarse a cambios en las restricciones o en el número de cursos, manteniendo tiempos de respuesta aceptables.

**6.2. Criterios de Éxito**

Para garantizar que el sistema multiagente cumple con los objetivos propuestos, se establecieron los siguientes criterios de éxito:

1. **Soluciones sin conflictos:**
   * Todas las asignaciones de horarios y aulas deben cumplir con las restricciones definidas, asegurando que no existan conflictos de capacidad, horarios o uso de recursos.
2. **Distribución eficiente de recursos:**
   * La solución final debe maximizar el uso de diferentes aulas, distribuyendo de manera equitativa la carga horaria entre las asignaciones.
3. **Adaptabilidad a cambios:**
   * El sistema debe ser capaz de ajustarse rápidamente a modificaciones en las restricciones (por ejemplo, añadir nuevos cursos, horarios o aulas) sin necesidad de reestructurar todo el sistema.
4. **Rendimiento óptimo:**
   * El sistema debe generar soluciones válidas en tiempos razonables, incluso al manejar un número elevado de variables y restricciones.

# **Conclusión**

El diseño de solución propuesto garantiza la asignación eficiente de horarios mediante la implementación de un sistema multiagente basado en técnicas de programación con restricciones (CSP). Este enfoque permite dividir el problema en tareas específicas manejadas por agentes independientes, lo que mejora la modularidad y la colaboración en la resolución del problema. La implementación en Python, junto con bibliotecas especializadas, facilita la validación y optimización de las soluciones, asegurando un modelo flexible, escalable y robusto para manejar las complejidades típicas del entorno educativo.

La programación con restricciones (CSP) ha demostrado ser una herramienta eficaz para resolver problemas complejos de asignación, como la distribución de cursos en aulas. A través de la definición precisa de variables, dominios y restricciones, se logran soluciones válidas que maximizan el uso de los recursos disponibles, minimizan los conflictos y optimizan la planificación académica. Además, la integración de un sistema multiagente ha permitido abordar el problema de manera más eficiente, asegurando que las soluciones finales sean válidas y que cumplan con los objetivos de optimización definidos.