Propuestas de Trabajos Finales – Programación.

Este documento reúne una serie de propuestas de trabajos finales. Cada propuesta incluye una descripción detallada y referencias iniciales para orientar la investigación.

Movimiento Colectivo (Simulación Multiagente)

Bandadas de Pájaros (y similares)

Este proyecto propone simular el movimiento colectivo de animales como bandadas de pájaros o bancos de peces mediante el modelo de Vicsek. El objetivo es comprender cómo reglas simples de interacción local (alinearse con vecinos, evitar colisiones, mantener cohesión) generan patrones globales coordinados. La simulación permitirá ajustar parámetros como el número de individuos, velocidad, radio de interacción o nivel de ruido, y observar cómo emergen comportamientos colectivos diferentes. Este trabajo ayuda a introducirse en la programación de sistemas multiagente y en el estudio de fenómenos emergentes.

Referencias:

- - Vicsek, T. et al. (1995). Novel type of phase transition in a system of self-driven particles. PRL.
- - Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. SIGGRAPH.
- - Wikipedia: Vicsek model

Estrategias de Evacuación de Recintos

Este trabajo consiste en modelar y simular la evacuación de un recinto lleno de personas (estadio, sala de conciertos, avión, etc.) en situaciones de emergencia. Usando modelos de multitudes basados en agentes o partículas, se pueden estudiar cómo influyen factores como el número de salidas, su tamaño, la densidad de ocupación o el comportamiento de los individuos. Se podrán probar distintas configuraciones y estrategias de evacuación (disposición de salidas, limitación de cuellos de botella, control del flujo) para analizar cuál resulta más eficiente.

Referencias:

- - Helbing, D. et al. (2000). Simulating dynamical features of escape panic. Nature.
- - Kirchner, A. & Schadschneider, A. (2002). Simulation of evacuation processes. Physica A.
- - Wikipedia: Evacuation dynamics

Apocalipsis Zombie

Este proyecto es una variación más "lúdica" del movimiento colectivo. La idea es simular una población de zombies y supervivientes, donde los primeros intentan propagar la infección y los segundos escapar o resistir. Se pueden implementar reglas como persecución, evasión, contagio, inmunidad o curación, lo que conecta con los modelos epidemiológicos clásicos (SIR). Además de ser divertido, permite reflexionar sobre dinámicas de contagio, propagación en poblaciones y comportamientos emergentes bajo presión. La visualización mostrará cómo evoluciona la población con el tiempo, registrando estadísticas de supervivientes y zombies.

Referencias:

- - Alemi, A. A. et al. (2015). You can run, you can hide: The epidemiology and statistical mechanics of zombies. PRE.
- - Munz, P. et al. (2009). When zombies attack!: Mathematical modelling of an outbreak of zombie infection. Nova Science.
- - Wikipedia: Mathematical model of zombie apocalypse

Juegos y Simulación

Juego de la Vida (Conway)

El Juego de la Vida es un autómata celular inventado por John Conway en 1970. Consiste en un tablero bidimensional de celdas que pueden estar vivas o muertas y que evolucionan siguiendo reglas muy simples (supervivencia, muerte por soledad o sobrepoblación, nacimiento). A pesar de su sencillez, genera patrones complejos como osciladores, naves espaciales y estructuras que parecen "vivas". El trabajo consiste en implementarlo en Python, con visualización gráfica, y explorar distintos estados iniciales y patrones.

Referencias:

- - Gardner, M. (1970). Scientific American (Juego de la Vida).
- - Wikipedia: Conway's Game of Life
- - Wolfram, S. (2002). A New Kind of Science.

Problema de las N-Reinas

El clásico problema de las N-Reinas consiste en ubicar N reinas en un tablero de ajedrez de tamaño NxN de forma que ninguna se ataque entre sí (en filas, columnas o diagonales). Es un excelente ejemplo para practicar algoritmos de búsqueda, backtracking y optimización. El proyecto incluye programar un algoritmo que encuentre soluciones para distintos valores de N, representarlas gráficamente y comparar métodos de resolución.

Referencias:

- - Bell, J., & Stevens, B. (2009). A survey of known results and research areas for n-queens. Discrete Mathematics.
- - Wikipedia: N-Queens problem.
- Norvig, P. (1999). *Artificial Intelligence: A Modern Approach* (ejemplos de backtracking).
- https://www.geeksforgeeks.org/dsa/n-queen-problem-backtracking-3/

Juego de la Serpiente Automático (Snake)

Este proyecto propone implementar el clásico videojuego Snake, pero con un modo automático en el que la serpiente se mueve siguiendo un algoritmo programado en lugar de un jugador humano. Se pueden explorar distintas estrategias de inteligencia artificial para maximizar la supervivencia y la puntuación (por ejemplo, búsqueda de caminos, algoritmos heurísticos o simples reglas de comportamiento).

Referencias:

- - Wikipedia: Snake (video game)
- - Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach.

Sudoku y otros Juegos Lógicos

En este trabajo se pueden desarrollar algoritmos para resolver el famoso puzzle Sudoku (y, si se desea, extender a otros juegos lógicos como Kakuro o Nonogramas). Resolver Sudokus implica aplicar técnicas de restricción, backtracking y búsqueda heurística. Posible extensión: generador de Sudokus con distintos niveles de dificultad.

Referencias:

- - Simonis, H. (2005). Sudoku as a constraint problem.
- - Norvig, P. (2006). Solving Every Sudoku Puzzle.
- - Wikipedia: Sudoku solving algorithms

Juego del Wordle Automático

Wordle es un juego de adivinanza de palabras en el que el jugador tiene que descubrir una palabra de cinco letras en seis intentos, recibiendo pistas de colores tras cada intento. El reto consiste en programar una versión automática en Python que pueda jugar y resolver el Wordle de forma eficiente. Se pueden explorar diferentes estrategias: desde fuerza bruta hasta algoritmos heurísticos o evolutivos. Además, se puede incluir un modo jugador humano y un modo automático.

Referencias:

- - Wikipedia: Wordle
- - Knuth, D. (1976). The computer as master mind.
- Implementaciones en Python disponibles en GitHub (buscar “Wordle solver”).

Algoritmo MiniMax (IA en Juegos)

El algoritmo Minimax es un enfoque clásico de inteligencia artificial para juegos de dos jugadores de suma cero (como 3 en raya, Conecta 4, ajedrez). El objetivo es implementar este algoritmo en Python y aplicarlo a un juego sencillo como 3 en raya o Conecta 4, dotando al programa de una IA básica capaz de tomar decisiones óptimas. Se puede incluir la técnica de poda alfa–beta para mejorar la eficiencia.

Referencias:

- - Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach.
- - Knuth, D., & Moore, R. W. (1975). An analysis of alpha-beta pruning.
- - Wikipedia: Minimax
- https://www.geeksforgeeks.org/dsa/minimax-algorithm-in-game-theory-set-1-introduction/

Inteligencia Artificial y Optimización

Algoritmo Genético

Los algoritmos genéticos son una técnica de optimización inspirada en la evolución biológica. Representan posibles soluciones a un problema como individuos en una población, que evolucionan generación tras generación mediante operadores como selección, cruce y mutación. El objetivo de este proyecto es implementar un algoritmo genético en Python y aplicarlo a un problema concreto: optimización de una función matemática, selección de un conjunto óptimo de elementos o resolución de un puzzle. Observar el efecto de variar diferentes parámetros (tamaño de población, tasa de mutación, número de generaciones) y comparar resultados. Además, se pueden visualizar las soluciones y su evolución para entender mejor el proceso evolutivo.

Referencias:

- - Holland, J. H. (1975). Adaptation in Natural and Artificial Systems.
- - Mitchell, M. (1996). An Introduction to Genetic Algorithms.
- - Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning.
- https://towardsdatascience.com/genetic-algorithm-6aefd897f1ac/

Colonia de Hormigas (ACO)

El algoritmo de colonia de hormigas es una técnica de optimización inspirada en el comportamiento de las hormigas cuando buscan comida. Basado en el uso de feromonas y la cooperación indirecta, este algoritmo permite resolver problemas de optimización difíciles, como el famoso problema del viajante de comercio (TSP) o el enrutamiento en redes. El objetivo del trabajo es implementar un ACO en Python y aplicarlo a un problema concreto (p. ej. encontrar el camino más corto en un grafo). Variar el valor de parámetros como el

número de hormigas, la evaporación de feromonas o la probabilidad de exploración, y analizar cómo estos influyen en la calidad de las soluciones. La visualización gráfica de los caminos recorridos por las hormigas ayudará a comprender la dinámica del algoritmo.

Referencias:

- - Dorigo, M., & Stützle, T. (2004). Ant Colony Optimization. MIT Press.
- - Wikipedia: Ant colony optimization algorithms
- https://induraj2020.medium.com/implementation-of-ant-colony-optimization-using-python-solve-traveling-salesman-problem-9c14d3114475

Matemáticas y Física Computacional

Fractales y Juego del Caos

Los fractales son estructuras geométricas que se caracterizan por su autosimilitud y complejidad infinita. En este trabajo se propone implementar distintos fractales clásicos como el conjunto de Mandelbrot, el copo de Koch o el triángulo de Sierpiński. Además, se plantea programar el **Juego del Caos**, un método iterativo basado en reglas simples y aleatoriedad para generar estructuras fractales. Este proyecto permitirá explorar visualmente propiedades matemáticas como la dimensión fractal, los atractores y la emergencia de orden a partir de reglas sencillas. Se recomienda experimentar con parámetros, generar imágenes y analizar cómo pequeños cambios producen resultados muy distintos.

Referencias:

- - Mandelbrot, B. (1982). The Fractal Geometry of Nature.
- - Barnsley, M. (1988). Fractals Everywhere.
- - Wikipedia: Fractal
- - Wikipedia: Chaos game

Órbitas de Kepler (Integración Numérica)

Este trabajo propone simular las órbitas planetarias resolviendo numéricamente las ecuaciones diferenciales que rigen el movimiento según las leyes de Kepler y la gravitación de Newton. Se pueden aplicar métodos sencillos como el método de Euler y comparar su precisión con métodos más avanzados como Runge-Kutta. Comparar precisión, estabilidad y errores numéricos.

Referencias:

- - Press, W. H. et al. (2007). Numerical Recipes.
- - Butcher, J. C. (2016). *Numerical Methods for Ordinary Differential Equations*. Wiley
- - Wikipedia: Euler method
- - Wikipedia: Kepler problem

Modelos de Dinámica de Poblaciones y Epidemiología (EDOs)

Introducción a la modelización matemática con sistemas de EDOs:

- Lotka–Volterra (presa–depredador**):** estudio de ciclos poblacionales, estabilidad y retratos de fase.
- SIR epidemiológico**:** evolución de una epidemia mediante compartimentos Susceptibles–Infectados–Recuperados, análisis de R_0 , medidas de control.

Referencias:

- - Murray, J. D. (2002). Mathematical Biology I. Springer.
- - Kermack, W. O. & McKendrick, A. G. (1927). A contribution to the mathematical theory of epidemics.
- - Munz, P. et al. (2009). When zombies attack!: Mathematical modelling of an outbreak of zombie infection.

Música y Arte Computacional

Generación de Música

Componer melodías y armonías mediante algoritmos de generación: reglas, cadenas de Markov, búsqueda, algoritmos genéticos. Posible extensión: exportar a MIDI y visualizar partituras o espectrogramas.

Referencias:

- - Roads, C. (1996). The Computer Music Tutorial. MIT Press.
- - Miranda, E. R. (2001). Composing Music with Computers. Focal Press.
- - Wikipedia: Generative music

Percepción Musical y Acústica

Simulación de fenómenos acústicos que afectan a la percepción musical: consonancia, disonancia, fundamental ausente, reverberación. Permite analizar señales con Fourier y visualizar espectros.

Referencias:

- - Sethares, W. A. (2005). Tuning, Timbre, Spectrum, Scale. Springer.
- - Zwicker, E., & Fastl, H. (1999). Psychoacoustics: Facts and Models. Springer.
- - Wikipedia: Missing fundamental
- - Wikipedia: Reverberation

Normas Generales para los Trabajos Finales

Presentación oral

- Cada grupo dispondrá de 10 minutos para presentar su trabajo, seguidos de 5 minutos de preguntas por parte del profesor y/o los compañeros.
- La presentación debe ser clara, estructurada y apoyarse en material visual (diapositivas, demostración en vivo, etc.).

Entrega del cuaderno de trabajo

Se deberá entregar un Jupyter Notebook (o equivalente) que contenga:

- El código completo y funcional.
- Celdas Markdown que expliquen los objetivos, metodología, resultados y conclusiones.
- Gráficos, visualizaciones y, en su caso, animaciones que apoyen la explicación.

Uso de librerías gráficas

- Es muy recomendable utilizar al menos una librería de visualización o interfaz gráfica (Matplotlib, Pygame, Tkinter, etc.) para ilustrar de forma clara el funcionamiento del programa.

Defensa del trabajo

- Todos los integrantes del grupo deben ser capaces de explicar cualquier parte del código, el modelo utilizado y las decisiones de diseño tomadas.
- Durante las preguntas, el profesor podrá dirigirse a cualquier miembro del grupo.

Composición de grupos

- Los trabajos se realizarán en grupos de **4 o 5 estudiantes** (salvo casos excepcionales justificados).
- Se espera una **participación equilibrada** entre todos los miembros.

Originalidad y buenas prácticas

- El código debe ser **original** (no copiado directamente de Internet) y estar correctamente comentado.
- Se valorará especialmente la claridad, la modularidad (funciones bien definidas), y el uso de convenciones de estilo (PEP8 en Python).

Resultados y conclusiones

- El trabajo debe incluir un apartado de **análisis de resultados** (por ejemplo: métricas, gráficas comparativas, discusión de limitaciones).
- Debe presentarse también una **conclusión final** donde el grupo reflexione sobre lo aprendido y posibles mejoras.

Criterios de evaluación

- Precisión y corrección técnica de la implementación.
- Claridad y calidad del código.
- Documentación y explicaciones en el cuaderno.
- Calidad de la presentación oral.
- Capacidad de defensa individual de los miembros del grupo.
- **La calificación será la misma para todos los miembros del grupo. Es responsabilidad compartida que cada integrante conozca y comprenda el trabajo realizado, de modo que todos puedan responder de forma adecuada a las preguntas planteadas durante la defensa.**