Mejoras

En este fichero, puesto que muchas de las mejoras implementadas ya se han comentado en el fichero “EDA”, nos centraremos sobre todo en realizar una crítica sobre el estado actual del código y algoritmos, y comentaremos también los posibles caminos a seguir para la futura mejora del proyecto.

Algoritmo de resolución

En general, hay que apreciar que el algoritmo no es simple, y quizás estandarizamos e intentamos hacer más legible el algoritmo, pero es una tarea difícil que sin la ayuda de los tutores va a ser muy complicada.
En general nuestro algoritmo tiene muchos snippets que son simétricos para vertical y horizontal, y a veces desearíamos que esto quedará más visual.
Dentro de nuestras posibilidades vemos mucho más asequibles una serie de mejoras teóricas y prácticas que ahora explicaremos.
La más notable es añadir filtros polinómicos que reduzcan la cantidad de permutaciones que visitamos en el *backtrackingFiltroSequencias*. Estas serían comprobaciones que tienen tiempo lineal respecto de la secuencia y que simplemente se aseguran de que se cumplen ciertas condiciones necesarias (pero no suficientes) para que existan permutaciones de esa secuencia. Esto reduce la cantidad de secuencias “insatisfacibles” que es el objetivo de eficiencia.
En segundo lugar hay que quizas reestructurar la funcion de inicializacion del kakuro para que sea mas rapida, pero dudamos que podamos encontrar mejoras fáciles en este aspecto.
Aparte de esto estamos estudiando algoritmos similares al que hemos diseñado, porque muchas de las ideas de “pruning” de los algoritmos por backtrack han sido portables en forma de reglas o deducciones en nuestro formato iterativo. Con esto en mente quizás intentamos “adaptar ideas” en el futuro, pero al no tener muchas referencias sobre algoritmos con nuestro enfoque ya hemos implementado muchas de las mejoras de dominio que fueron surgiendo.
Por supuesto el estudio de qué estructuras de datos son más eficientes sigue, y a pesar de que ya están muy asentadas en el algoritmo, sigue habiendo variables locales que tenemos que investigar comparativamente (x vs y) que estructura de datos es mejor.

Algoritmo de generación

En cuanto a las mejoras del algoritmo de generación, la situación todavía no está donde nos gustaría. Actualmente disponemos de dos algoritmos para la fase de proponer los valores de las casillas. Una de ellas parte de una idea más compleja, similar en parte al algoritmo de resolución, considerando las posibles combinaciones posibles en cada momento. Este algoritmo, debido a su gran complejidad, a día de hoy aún no se ha conseguido que funcione adecuadamente. Tras probar ciertas simplificaciones sobre este algoritmo, optamos por desarrollar en paralelo un algoritmo mucho más sencillo basado en backtracking, puesto que el anterior no dio los resultados que deseábamos. Esto no ha sido una solución ideal, pero debido a las restricciones de tiempo, hemos preferido ofrecer un algoritmo temporal mucho más sencillo y funcional, a arriesgarnos a no conseguir arreglar el otro a tiempo. Para la próxima entrega deberemos volver a estudiar donde falla nuestro algoritmo y considerar si sale más a cuenta intentar que un algoritmo conceptualmente mejor pero mucho más complejo funcione, o hacer que el algoritmo basado en backtracking sea más sofisticado y nos ofrezca las soluciones que buscamos a coste de la eficiencia.

No solamente esto, sino que en la fase de construcción del tablero nos hemos puesto también unas restricciones bastante severas a la hora de definir los tableros que podemos crear, a saber, a momento de hoy tan solo podemos crear kakuros simples en los que la primera fila y columna contiene todas las casillas negras con suma, y por lo tanto, el tamaño máximo del tablero es de 10x10; y por otro lado, también hemos permitido que se puedan crear tableros en los que todas las casillas blancas se hallan en las diagonal principal y las inmediatamente superior y superior. Como limitación, estos últimos tan solo funcionan cuando el tablero es cuadrado. Cuando es rectangular no se hallará solución.

Por ahora estas decisiones se han tomado para simplificar un proceso que debimos hacer incremental desde un buen principio. Haber intentado implementar una versión compleja desde un buen principio ha tenido su coste, tal y como se puede observar. Cabe mencionar que esto también es debido a todo el tiempo dedicado a intentar que el algoritmo de resolución siguiese la idea que buscábamos, y hay que remarcar que los resultados obtenidos han sido muy satisfactorios.

Queda pues como trabajo futuro la gran mejora de este algoritmo para poder generar tableros con mayor versatilidad y eficiencia.