7.1 Agile Asymptotic Analysis

Es importante mencionar que nuestro estudio de la complejidad ha sido calculado usando un programa implementado *ad hoc* para este fin. Este, además, nos ha permitido realizar una maquetación personalizada para nuestro libro, lo que nos permite, como se ha podido observar en el capítulo "Prototype", indicar línea a línea cómo va evolucionando esa complejidad, así como también indicar el costo asintótico asociado a cada una de las funciones que hemos implementado en nuestro prototipo. Sin embargo, cabe mencionar que este estudio no es automático, sino que toma la complejidad leyendo usos comentarios o etiquetas que asociamos principalmente a los distintos bloques iterativos (for, while, recursive-if, etc.), presentes en cada una de las funciones. A partir de esta información, somos capaces de asignar a cada función su complejidad máxima siguiendo las reglas de Big O, y extender el análisis asintótico por todo el algoritmo, lo que nos permite hacer un análisis de grano grueso, fácilmente actualizable de la complejidad que presenta todo el algoritmo. Es decir, un análisis muy pesimista y aproximado de cuál es la complejidad de todas las funciones de nuestro prototipo que, además, puede evolucionar fácilmente en caso de que cambiemos algo del código (Agile Asymptotic Analysis).

En este contexto, tenemos que decir que la finalidad principal de este estudio no es analizar cómo de eficiente (o ineficiente) es el algoritmo prototipo, sino, simplemente, poder enmarcar cada procedimiento y/o función dentro del mundo polinómico o exponencial, en tanto y en cuanto se recuerde que, si todas las funciones de un algoritmo presentan unos costos temporales polinómicos ($O(n^k)$), entonces, podemos decir que nuestro algoritmo es polinómico, por muy ineficiente que sea este (por muy grande que sea el grado k de estos polinomios). Es por eso que hemos optado, en este análisis, por intentar ser lo más pesimistas posibles pues, a fin de cuentas, en este contexto, lo que nos interesa es saber si nuestras decisiones de diseño nos han permitido (o no) evitar esa escalada exponencial del espacio y, por ende, del tiempo. En resumen, para nosotros, lo más importante con este estudio no es mostrar cómo de eficiente (o ineficiente) es este algoritmo prototipo, sino cómo lo hemos diseñado. Es decir que nos interesa mucho más que se entienda por qué nuestras decisiones de diseño nos han conducido a diseñar una estructura de datos con la que logramos trasladar esa naturaleza exponencial de nuestros casos de estudio HC y TSP a la dimensión de lo abstracto.

Resumen 7.1.1 *Prototype's Asymptotic Analysis*

- *Hamiltonian Machine:*
  - *Solves: HC (NP-Complete)*
  - *Machine Execution Complexity:* $O(N^{16})$ *then is Polynomial*.*
  - *Reader Path Complexity:* $S * O(N^{13})$*, where* S *is the number of solutions to read, therefor when* $S = 1$*, the reading process is Polynomial*.*
  - *Total HC reduction:* $O(N^{16}) + O(N^{13}) \approx O(A^8)$
- *TSP Machine:*
  - *Solves: Optimization Version of TSP (NP-Hard)*
  - *Machine Execution Complexity:* $O(B * N^{15})$ *where* B *is the maximum length of a solution tour, therefor we can consider that for a huge* B*, we are working with Pseudo-Polynomial* algorithm.*
  - *Reader Path Complexity:* $S * O(N^{13})$*, where* S *is the number of solutions to read, therefor when* $S = 1$*, the reading process is Polynomial*.*
  - *Total TSP-Optimization reduction:* $O(B * N^{15}) + O(N^{13}) \approx O(B * N^{15})$

Es importante mencionar que nuestro prototipo acepta también como instancias grafos dirigidos, por lo que a priori puede responder a las versiones dirigidas de los problemas HC y TSP.

NOTA:
Hay que tener en cuenta que toda consideración rigurosa de que un algoritmo resuelve correctamente todas las instancias de un problema en tiempo polinomial o pseudopolinómico debe estar avalada por una verificación formal o por una demostración matemática que dé soporte formal a tal afirmación. En otras palabras, ante la ausencia de tal demostración, por ejemplo, la afirmación de que el problema HC es resoluble en tiempo polinómico por nuestro algoritmo sería una conclusión al menos poco rigurosa, en la medida en que no podemos afirmar que sea correcto para todas las instancias.

A pesar de que nuestra Hamiltonian Machine, como vimos en la página 201 (véase función "execute!"), para construir el conjunto ϕ solución presentaría un orden de complejidad enorme $O(N^{16}) \approx O(A^8)$, este no sería exponencial. Es decir que, por muy ineficiente que nos pueda parecer, gracias al uso de las

abstracciones exponenciales, podríamos llegar a valorar —por contraintuitivo que parezca— que el proceso de construcción del conjunto de todos los circuitos hamiltonianos (cómputo de ϕ) de una instancia I_{HC} podría estar P.

Esta idea de forma previa a plantearnos que podríamos usar la dimensión de lo abstracto para el diseño de algoritmos nos hubiera parecido algo completamente contraintuitivo, en tanto y en cuanto no tiene sentido que sea más *fácil* construir todo el conjunto ϕ de circuitos hamiltonianos ($SETHC \in P$?) que encontrar un solo circuito ($HC \in NP - complete$). Ahora bien, es importante destacar cómo este planteamiento encaja y es coherente *físicamente* en la medida en que no hemos pretendido eliminar la naturaleza exponencial que inherentemente presenta el problema SETHC, sino que solo la hemos trasladado a una dimensión distinta del espacio y del tiempo (véase 7.1). De hecho, aceptamos esa naturaleza del problema durante nuestro diseño, y le buscamos un acomodo en una estructura de datos exponencial (EDS) cuyos requisitos no funcionales y fin principal consisten, precisamente, en usar la abstracción para evitar la escalada exponencial del espacio y, por ende, del tiempo.

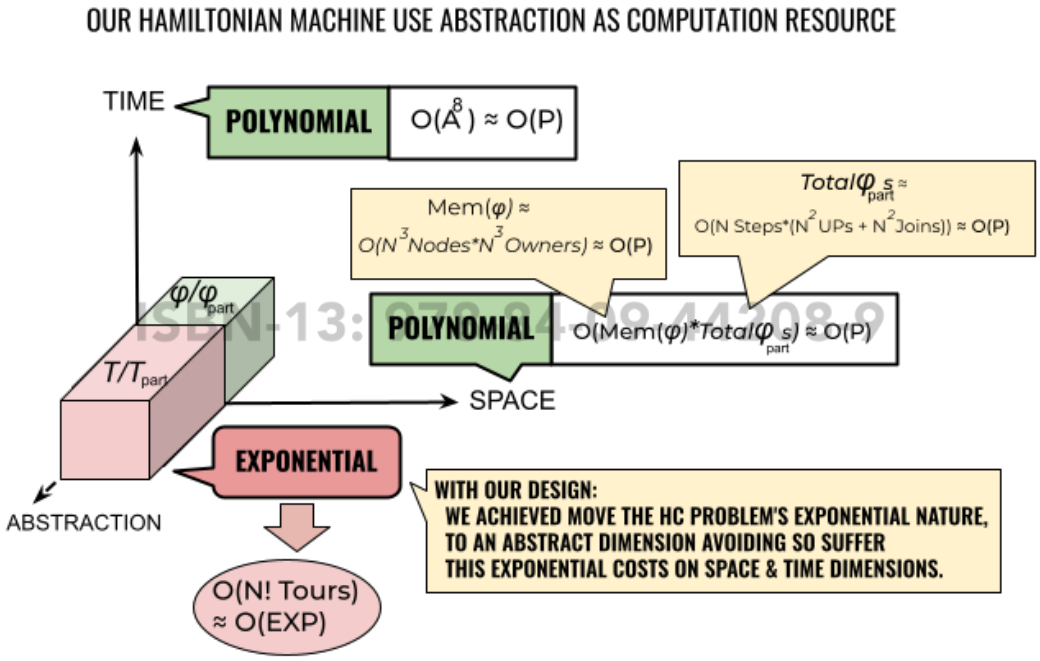

Figura 7.1: Usando la abstracción como un recurso computacional

De hecho, recordemos cómo, aunque sea de una forma informal, a lo largo del diseño de nuestro algoritmo, hemos ido reconociendo esa naturaleza exponen-