

# Análisis de algoritmos: Practica 1.

# **ALGORITMOS VORACES.**

Daniel Lois Nuevo

Adrián García Oller

1. **Especificación del problema.**

**Precondición:**

Los costes tienen que ser mayores o iguales que cero.

El gasto de traslado tiene que ser mayores o iguales que cero.

Tiene que haber un array de costes por cada sede.

El tamaño de los arrays de costes debe ser igual.

**Postcondición:**

El coste óptimo tiene que ser superior o igual a cero.

**Función objetivo:**

Sumatorio de la suma de los costes de cada sede mas el coste de traslado en caso de que se quiera cambiar de sede respecto al mes anterior

1. **Algoritmos.**

Algoritmo h1, decide la sede en función del menor coste, cuando ya ha elegido la sede suma el coste de traslado si la sede es distinta a la anterior, si es la primera no suma el coste de traslado.

public static int sedesSinCostedeTraslado(int[] c0, int[] c1, int f){

    int coste = 0;

    int anterior = 0;

    if (c0[0] < c1[0]){

        coste = c0[0];

    } else {

        coste = c1[0];

    }

    for (int i = 1; i < c0.length;i++){

        if ((anterior == 0)){

            if (c0[i] < c1[i]){

                coste = coste + c0[i];

                anterior = 0;

            } else if (c0[i] > c1[i]){

                coste = coste + c1[i] + f;

                anterior = 1;

            }

        } else if (anterior == 1) {

            if (c0[i] < c1[i]){

                coste = coste + c0[i] + f;

                anterior = 0;

            } else if (c0[i] > c1[i]){

                coste = coste + c1[i];

                anterior = 1;

            }

        }

    }

    return coste;

}

Algoritmo h2 es igual que el anterior solo que a la hora de elegir la sede tiene en cuenta el coste del traslado en caso de que la sede sea distinta a la anterior.

public static int sedesConCostedeTraslado(int[] c0, int[] c1, int f){

    int coste = 0;

    int anterior = 0;

    if (c0[0] < c1[0]){

        coste = c0[0];

        anterior = 0;

    } else {

        coste = c1[0];

        anterior = 1;

    }

    for (int i = 1; i < c0.length;i++){

        if ((anterior == 0)){

            if (c0[i] < (c1[i] + f)){

                coste = coste + c0[i];

                anterior = 0;

            } else if (c0[i] > (c1[i]+f)){

                coste = coste + c1[i] + f;

                anterior = 1;

            }

        } else if (anterior == 1) {

            if ((c0[i]+ f) < c1[i]){

                coste = coste + c0[i] + f;

                anterior = 0;

            } else if ((c0[i]+ f) > c1[i]){

                coste = coste + c1[i];

                anterior = 1;

            }

        }

    }

    return coste;

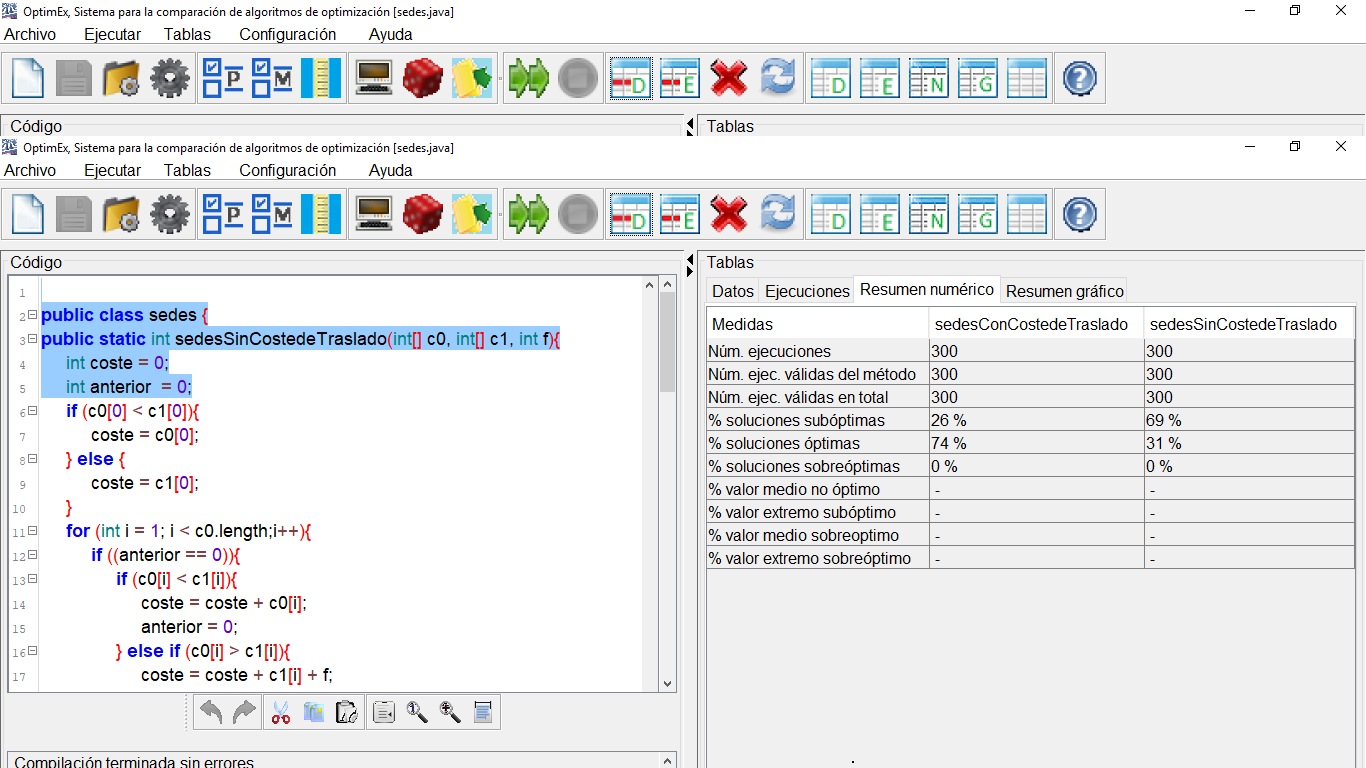
}

}

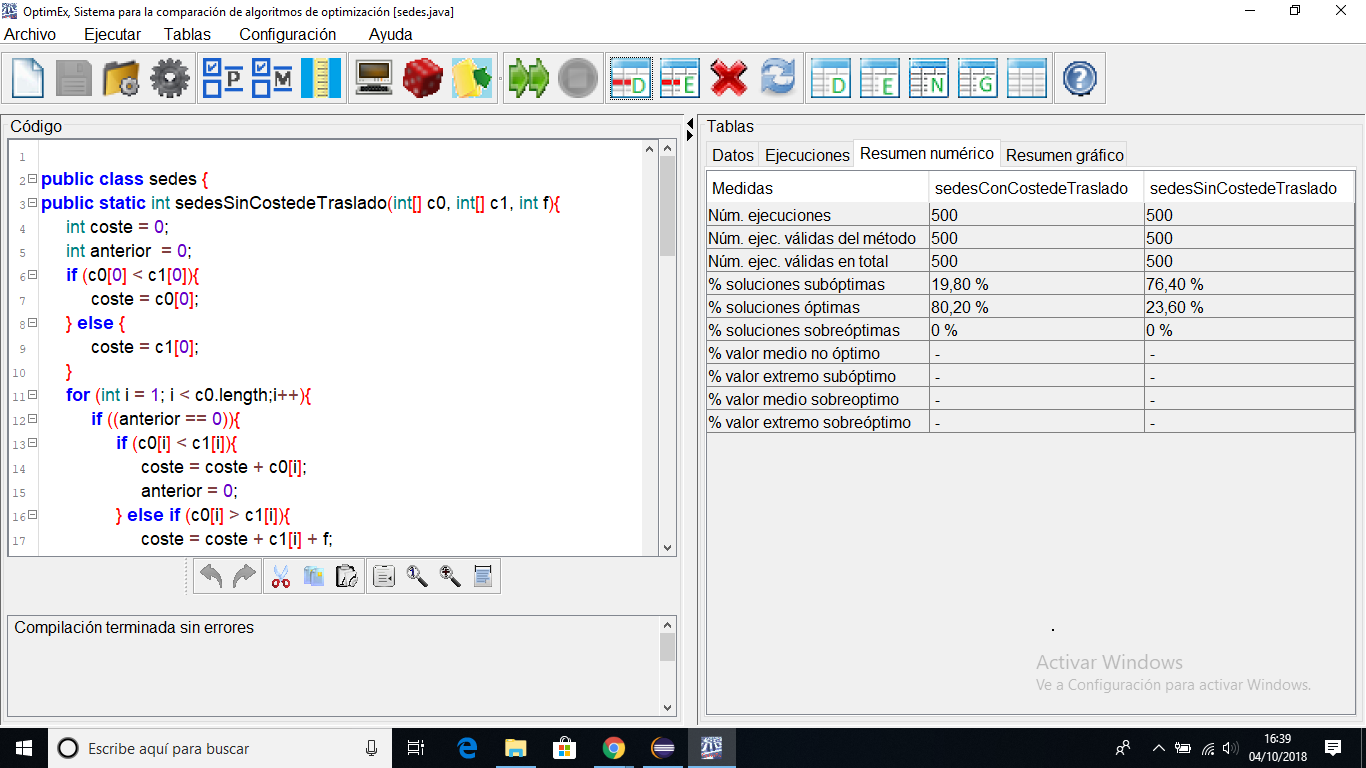
1. **Experimentación con la optimalidad de los algoritmos.**

Al tratarse de dos algoritmos heurísticos ninguno será exacto. Esto se puede observar en que en todas las pruebas de Optimex realizadas nunca se obtenía un 100% de resultados óptimos por parte de ninguno de los dos algoritmos.

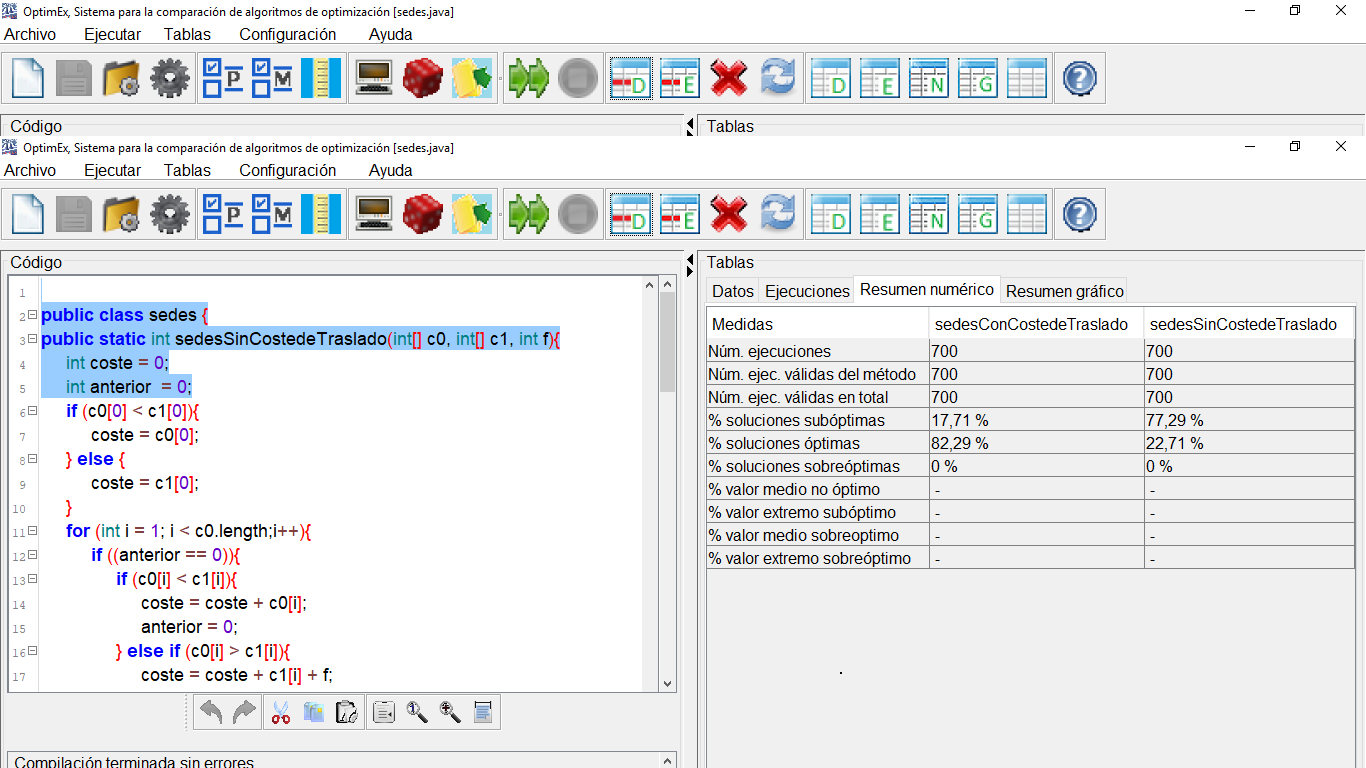
Para la obtención de las pruebas hemos optado por realizar 4 baterías de pruebas, con un tamaño de 300, 500, 700 y 1000 ejecuciones respectivamente.



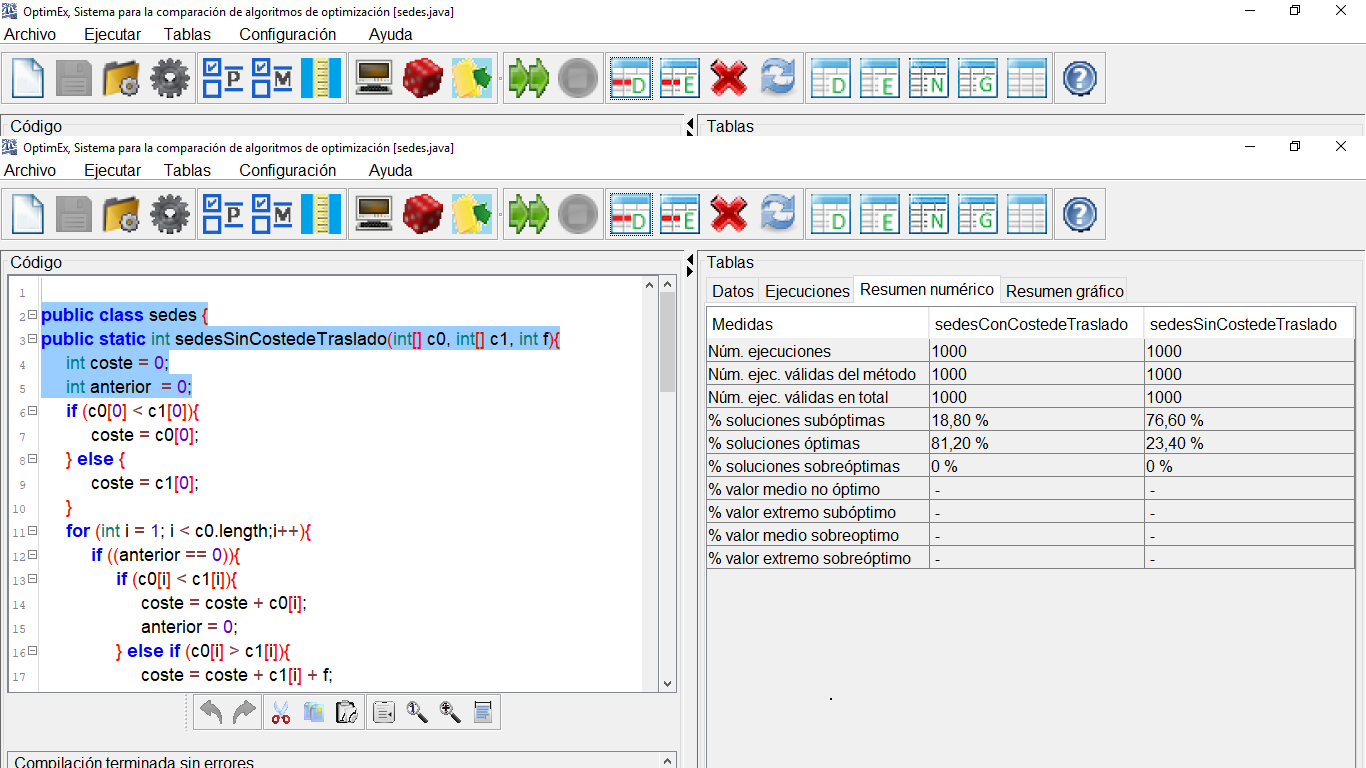
300 ejecuciones. En la imagen se ve que h1 (sedesCosteSinTraslado) da un 31% de resultados subóptimos mientras que h2(sedesCosteConTraslado) da un 74%.



500 ejecuciones. En la imagen se ve que h1 (sedesCosteSinTraslado) da un 23.6% de resultados subóptimos mientras que h2(sedesCosteConTraslado) da un 80.2%.



700 ejecuciones. En la imagen se ve que h1 (sedesCosteSinTraslado) da un 22.71% de resultados subóptimos mientras que h2(sedesCosteConTraslado) da un 82.29%.



1000 ejecuciones. En la imagen se ve que h1 (sedesCosteSinTraslado) da un 23.4% de resultados subóptimos mientras que h2(sedesCosteConTraslado) da un 81.2%.

Observamos en las tablas de resúmenes que el algoritmo heurístico que tiene en cuenta el coste de traslado a la hora de elegir el cambio de sede (H2) obtiene en torno a un 80% de las ejecuciones una solución menor que el que no tiene en cuenta el cambio de sede (H1). Por lo tanto, podemos afirmar que, según los resultados obtenidos, la heurística empleada en H2 resulta ser más precisa que la de H1.

1. **Conclusiones.**

Esta practica ayuda a comprender los algoritmos heurísticos y a practicar con herramientas de estudio de algoritmos. En cuanto a la dificultad de la práctica creemos que tiene una complejidad adecuada para poder aprender a utilizar herramientas de análisis de algoritmos y aprender a estudiar la optimalidad de estos.