Solución del problema de optimización

Queremos adaptar el Algoritmo de Recocido Simulado (SA) para resolver el problema de la línea de producción. Como muchos de los algoritmos de optimización, hay tres elementos del SA que tenemos que definir antes de pasar a la codificación:

1. **Estado**

   El estado es el conjunto de condiciones que satisfacen el problema y de los cuales vamos a sacar aquel que nos de la optimización que buscamos. En el algoritmo SA clásico, los estados son las posiciones para las que la función objetivo está definida. Para este problema, los estados son el conjunto de máquinas y el orden en el que procesan los elementos que les corresponde.

2. **Transición**

   La transición es la forma en la que pasamos de un estado a otro en el proceso de búsqueda. Típicamente, estas transiciones son no deterministas, como cuando tomamos un vector aleatorio en la versión elemental del SA. En este caso, para transitar de un estado a otro lo que haremos será tomar una orden aleatoria asignada a una máquina aleatoria, y cambiarla a otra máquina aleatoria. (Por ejemplo, si la orden 2 la tenemos para ser procesada por la máquina 4, cambiarla a la máquina 1).

3. **Evaluar**

   Una vez que definimos el estado y su cambio, debemos de definir la forma en la que vamos a evaluar el estado para saber si es más o menos apropiado que otros por los que hemos pasado. Cuando tenemos una función basta con obtener el valor de la función para las variables dadas. En este caso, vamos a sumar los tiempos de los órdenes de cada máquina, así como sus respectivos tiempos de ajuste, y vamos a sacar el mayor de cada uno. Recordemos que se nos pide obtener el mínimo tiempo de la máquina que se tarda más. Aquí también vale la pena recordar que hay estados que no son válidos (hay órdenes que no pueden ser procesadas en todas las máquinas), por ello tenemos que penalizarlos evaluando con un valor muy grande.

Ya que entendimos estos conceptos, veamos una forma simple en la que pueden ser implementados.

1. **Estado**

   Sea O la cantidad de órdenes y M la cantidad de máquinas. Por simplicidad, consideraremos que las máquinas y órdenes están numeradas de 0 a $M-1$ y de 0 a $O-1$, respectivamente. Se propone usar un arreglo a de tamaño O donde $a[i]=k, 0 \leq k < M$ sea el número de máquina que procesará la orden i. Por ejemplo, el arreglo $a = [0,0,1,2,0,2]$ representa el estado en el que la máquina 0 procesa las órdenes 0, 1 y 4, la máquina 1 procesa la orden 2, y la 2 las órdenes 3 y 5.

2. **Transición**

   Una vez definido el estado, vemos que la transición se puede realizar fácilmente tomando aleatoriamente los enteros $0 \leq j < O$ y $0 \leq p < M$ y realizando la asignación $a[j]=p$. Se sugiere usar una distribución uniforme. Si se desea, se puede validar que

$a[j] \neq p$ para evitar tomar el elemento y cambiarlo a la máquina en la que se encuentra, aunque este paso se podría omitir. También omitimos la validación del estado, pues ya dijimos que implementaremos un sistema de penalización. Estos dos mecanismos podrían mejorar la eficacia del método, así que vale la pena experimentar con ellos.

3. **Evaluar**

   Evaluar es la parte más complicada del código. Una forma simple consiste en hacer M iteraciones sobre nuestro arreglo de estado, una por máquina, sacar la suma y los ajustes que corresponden, y quedarnos con el mayor resultado de esas iteraciones. La complejidad de ese proceso es de $O(MO)$, y si tomamos en cuenta que tenemos que hacer varias iteraciones vemos que podríamos necesitar algo más rápido. Hay una forma de hacerlo en $O(M)$ (una sola iteración sobre el arreglo de estado), pero por ahora omitiremos su explicación. No hay que olvidarnos de implementar la validación en este paso también.