# Método de optimización ”algo-inspirado”: algoritmo del Homo-Sapiens Tecno-inspirado

Víctor Noriega, Fabián Encinas, Mario Castro

***Abstract*— The intention of this document is to show you a brief introduction to the techno-inspired algorithm that we, three students of computer science, thinked about in the last days. So, hold your seats and keep your head on its place.**

## I. INTRODUCCIÓN

El hombre es uno de los seres vivos más particulares que existen en nuestro planeta. No solo son los padres de novedosos ”inventos”, sino que también lograron adaptarse a estos de manera que les permitiera mejorar condiciones de vida tanto individuales como grupales. Partiendo de que una de las principales y primeras innovaciones del hombre fue formar grupos que tenían un objetivo en común (supervivencia) y con estos grupos dominaban objetivos secundarios (a veces otros grupos) mientras procuraban una mejor herencia (legado), ideamos esta sencilla meta-heurística. A este algoritmo le llamamos ”Algoritmo Homo-Sapiens”.

## II. ¿EL INVENTAR FUE EL PRIMER INVENTO?

A lo largo de su trayectoria en la tierra, el hombre se ha visto superado más de una vez. Incluso en la actualidad, el hombre se ve superado por otros hombres. ¿Fue un momento de inspiración, un golpe de suerte, o una inferencia de conocimientos de alguna Deidad, el que el hombre haya descubierto el fuego? No lo sabemos. Pero sí sabemos que esto permitió al hombre supervivencia de algunas situaciones. La supervivencia es la primer parte de nuestro algoritmo, basada en algún invento de parte de algún subconjunto del grupo. Supongamos que tenemos el problema de las n-Reinas. Seguramente habrá un subconjunto de reinas que tengan menor costo que el total. Diremos que este subconjunto de reinas han ”inventado” algo, y que por haberlo inventado, les permite vivir de mejor manera individualmente que los demás. Sin embargo, estos individuos también son dependientes de los demás, pues si estos mueren, sus probabilidades de sobrevivir disminuyen.

## III. ZAPATERO A TUS ZAPATOS

Supongamos que tenemos ahora cinco reinas en un tablero de ocho por ocho y que el costo es de 0. Estas cinco reinas representan un mejor subconjunto que el total de la comunidad, pues ellos descubrieron ”el fuego” y pueden sobrevivir al invierno ruso. Ahora buscaremos el lugar más adecuado para alguna de las reinas faltantes, tal que no me empeore el costo de mi estado actual. Agregaremos tantas como sea posible hasta que haya alguna que altere mi costo de estado. Es entonces cuando comienza un primer conflicto en nuestra pequeña ciudad ficticia.

## IV. SELECCIONAR NUEVAMENTE

Si en la etapa pasada encontramos un costo igual o menor con una cardinalidad de sub-estado mayor, entonces en nuestra siguiente llamada a función vamos a poder escojer tuplas más grandes para operar, pero si no es así, podríamos incluso regresarnos a la tupla anterior, y así sucesivamente. Cabe destacar que este algoritmo es intuitivamente recursivo, y que debería implementarse de esa manera.

## V. ¿POR QUÉ CONVERGE ASINTÓTICAMENTE A UN MÍNIMO GLOBAL?

Hay varias etapas en las que hacemos uso de la aleatoriedad para convenientemente converger a algún mínimo global. Comenzando con los elementos que seleccionamos aleatoriamente en cada etapa de selección (puede haber varios grupos con costo menor, incluso puede que haya un grupo con un costo menor al que escojimos aleatoriamente). Eventualmente buscamos conflictos en el estado en el que nos ubicamos, esto es, que el costo haya empeorado al agregar un elemento nuevo al sub-estado (este elemento se agrega aleatoriamente; se agregan aleatoriamente y si hay varios elementos que mejoran mi estado, también selecciono aleatoriamente).

## VI. HOMO-PARALELUS

Todo hasta ahora implica buscar un óptimo global a partir de una sola búsqueda. Pero, ¿qué nos impide paralelizar este proceso? Ha sido evidente que ha habido múltiples comunidades de humanos en el mundo a través de la historia. ¿qué vamos a hacer con esto?

Cada hilo es independiente del otro, y vamos a usar algo que he llamado ”progreso y herencia”: el primero será usado por los hilos con peores costos para mejorar su desempeño, mientras el segundo lo usarán los hilos con los costos más pequeños buscando una mejora a futuro. Básicamente el primero consiste en buscar el costo menor de los vecinos inmediatos, y el segundo busca el costo menor de los vecinos de los vecinos. Dado que la ”sociedad” con mejor costo implica que vivan mejor, estarán buscando que las cosas persistan, multiplicando su población y territorio a largo plazo, sacrificando tiempo. Las otras sociedades (piensen en México) necesitan un cambio inmediato para entrar a la carrera del ”primer mundo”.

```
def homo_sapiens(problema, n=0, inventores = [], iter = 10000):
    """
    aqui abajo defino el subestado, que es un
    subconjunto de mi estado.
    generar_num_aleatorio() es un metodo que dependera
    de ciertos parametros (a veces comenzaremos en un
    subestado con la mitad de la cardinalidad del estado
    original, a veces menos, a veces mas.
    @param problema: un objeto clase problema
    @param n: un entero n, que representa el tamano del subproblema
    @param inventores: una lista con los elementos que brindan
    un mejor estado en conjunto.
    @param iter: maximo de iteraciones
    """

    if n is 0:
        n = generar_num_aleatorio(len(inventores))
    if inventores is Empty:
        inventores = seleccionar_subestado_al(problema.estado, n)
        costo_nuevo = problema.costo(inventores)

    #Hay que asegurarse de que seleccionar_subestado_al()
    #nos regrese un subestado con costo menor

    #Adaptacion es un vector del problema con un costo
    #igual a costo_nuevo, pero con cardinalidad mayor o
    #igual a la de inventores

    adaptados = adaptar(inventores, problema.estado,n)

    #Recordemos que adaptar es meter tantos elementos
    #del estado al subestado como sean posibles

    if len(adaptados) is len(inventores):
        #Entrar a este if implica que hubo conflictos
        #bien cabrones; La sociedad no necesita una
        #SmartStove!!!

        homo_sapiens(problema, n-1, inventores, iter-1)
        #Un desempleado mas
    elif len(adaptados) > len(inventores):
        n = len(adaptados)
        homo_sapiens(problema,n,adaptados, iter-1)
```