Exemple de disseny Orientat a Objectes
L'Algorisme Genètic senzill

Enunciat:

Anem a estudiar en detall un mètode d'optimització anomenat *algorisme genètic*. No l'estudiarem per utilitzar-lo optimitzant res, sino com a exemple de disseny orientat a objectes.

Primer plantejarem el problema en termes concrets, que ens permetin fer ràpidament un primer disseny.

Treballarem amb cadenes de bits d'una determinada mida N, constant. Per exemple '010010100101001' si N=15. El punt de partida serà un conjunt de M cadenes de bits de mida N, triades a l'atzar, que anomenarem *població* inicial $P_0 = \{c_1, c_2, \ldots c_M\}$. Aleshores, partint de P_0 anirem manipulant les cadenes de la població (tal com es detalla més endavant) per obtenir una nova població P_1, i així successivament, P_2, P_3, . . . Quan aturem el procés? L'aturem quan trobem una població P tal que alguna de les seves cadenes de bits sigui l'òptim que busquem. Una restricció que tindrem en compte és que totes les poblacions han de tenir el mateix nombre M de cadenes de N bits.

Per fer-ho senzill, nosaltres decidirem que l'"òptim" és una cadena especial, diguem-ne c*, que triarem inicialment a l'atzar. Així doncs, aturarem el procés quan trobem una població P_n tal que c* pertanyi a P_n.

A cada cadena li podem associar una *mesura de fitness*, és a dir, una quantitat numèrica que ens dirà com de propera és una cadena de bits a la cadena de bits que busquem, l'"òptim" c*. En el nostre cas utilitzarem com a mesura el nombre de bits d'una cadena c que són iguals i estan en la mateixa posició que els bits corresponents a c*. Per exemple, suposem que N=15 i c*='**010010100101001**'. Aleshores la mesura de fitness de la cadena '**0**0**0**0**1**1**1**0**1**1**0**1**1**0**0**' serà: d('**0**0**0**0**1**1**1**0**1**1**0**1**1**0**0**' ,c*) = 10 ja que dels 15 bits n'hi ha 5 de diferents.

Com passem de la població P_n a la població P_{n+1}? Donada una població P_n, calculem la mesura de *fitness* de cada una de les seves M cadenes $d(c_1,c^*)$, $d(c_2,c^*)$, . . . $d(c_M,c^*)$. Un cop fet això podem procedir de diferentes maneres:
a.- Podem mantenir un determinat nombre (M') de cadenes de la població, usualment aquelles que tenen la mesura de *fitness* més alta. Això ens deixa amb una població provisional de M' < M cadenes. Caldrà crear M-M' cadenes addicionals per arribar a P_{n+1}.
b.- Podem sustituir totes les cadenes de P_n per cadenes noves. Així doncs, caldrà crear M cadenes addicionals per arribar a P_{n+1}. Ho farem a partir de les cadenes de P_n. Això seria un cas particular del mencionat en l'apartat anterior on M'=0.
Normalment es considera M'=1, això s'anomena *elitisme* i és un paràmetre de l'algorisme.

En el paràgraf anterior hem parlat de crear M-M' cadenes noves per construir P_{n+1}, a partir de les M cadenes que tenim a P_n. Com ho fem? Ho farem basant-nos en processos

que tenen una lleugera similaritat amb processos que trobem en les cèl.lules vives per combinar els gens (d'aquí el nom dels algorismes). Considerarem essencialment dues maneres de fer-ho:

a.- Recombinació: Donades dues cadenes de bits c_i i c_j, construirem una cadena c nova a partir dels bits de c_i i c_j. Hi ha diverses maneres de fer-ho. Per simplicitat el que farem és que el bit k de la nova cadena de bits c, serà el bit k de c_i amb probabilitat ρ o el bit k de c_j amb probabilitat $1-\rho$. Anomenarem a ρ *llindar de recombinació* i és un paràmetre del sistema. Com triem c_i i c_j? Podem triar a l'atzar un cert nombre (que anomenarem *nombre de tornejos* i és un paràmetre del sistema) de cadenes de P_n i triar d'aquest grup la cadena amb la *fitness* més alta. Això ho farem dues vegades, una per c_i i una altra per c_j.

b.- Mutació: Donada una cadena c de N bits, generarem una cadena nova a partir de c on cada bit canviarà amb una determinada probabilitat τ. A τ l'anomenarem *taxa de mutació* i és un altre paràmetre del sistema.

Quan calgui crear una cadena nova per afegir a P_{n+1}, primer triarem dues cadenes de P_n, c_i i c_j, les recombinarem generant una cadena c_R, i aquesta c_R la mutarem generant la cadena c a afegir a P_{n+1}.

Això no és tot. Penseu que el procés té un fort component d'atzar, així que trobar la solució no està garantit. Per evitar estar buscant indefinidament podem considerar un nombre límit de poblacions generades MAX, de manera que aturarem el procés en arribar a la població P_{MAX} i respondrem la cadena d'aquesta població que sigui més propera a la solució.

Fixem-nos que aquest problema permet abstraure diversos factors que el caracteritzen. A classe estudiarem possibles abstraccions i generalitzacions per fer un algorisme genètic d'un abast més gran.