Informe de la pràctica final de l'assignatura d'Intel·ligència Computacional: *Applying genetic algorithms to zone design*

Lluís Bernat Ladaria
Màster MUSI 21-22

Sóller, Març 2022

1 Introducció

A l'article [1] es desgrana una proposta per al disseny de zones electorals. El que es pretén a l'article és exposar un algorisme genètic que permeti trobar propostes de divisió d'una determinada regió composta per n unitats de població agrupant aquestes, en només k zones amb similar població amb dret a vot.

1.1 Funció a optimitzar

Donat que els algorismes genètics requereixen minimitzar o maximitzar una funció de cost que relacioni els paràmetres de les zones amb l'objectiu, el treball en proposa algunes que involucren la desviació entre la població de cada zona i la població objectiva (la mitjana de la població). Altres aspectes que es proposen ponderar són una mesura de la compacitat radial de la zona, definida com el sumatori de totes les distàncies entre el centre de masses de la zona i cada un dels centroids de les unitats que la componen o també una mesura del seu arrodoniment, que és definit com el quocient entre el perímetre al quadrat de la zona i la seva àrea.

Com a exemple del que s'hi exposa, la següent funció a minimitzar té en compte la desviació poblacional, així com la compacitat radial:

$$\sum_j \left(|P_j - \mu| + \sum_{i \in Z_j} d_{ij} \right)$$

On P_j és el valor de la població amb dret a vot de la zona j-èssima, μ és el total de la població amb dret a vot dividit per la quantitat de zones a dissenyar (k) i d_{ij} és la distància entre el centroid de la unitat de població i-èssima i el punt de referència escollit com a *kilòmetre zero* de la zona j-èssima.

1.2 Espai de solucions

També es quantifica quina és la complexitat de l'espai de solucions del problema. De fet la quantitat total de solucions és semblant al número de solucions d'un problema de *clustering*. Per tant la fórmula que ho aproxima és la dels números de *Stirling* de segona espècie:

$$S(n,k) = \frac{1}{k!} \sum_{i=1}^{k} (-1)^i \left(\frac{k}{(k-i)!i!} \right) (k-i)^n$$

Com a exemple, si disposàssim de només 100 unitats poblacionals a agrupar en 8 zones, $S(100,8) \approx 5.052 * 10^{85}$. És a dir un espai de solucions enorme, si bé es pot reduir en certa mesura si tenim en compte que les zones han de ser contínues.

1.3 Codificació

Es proposen dues codificacions (i.e. dos genotips distints) per tal de representar una solució:

1. Assignar a cada una de les n unitats poblacionals un codi numèric enter. I representar les k zones com una cadena (o llista) amb els codis de totes les unitats poblacionals que, al seu torn, constituiran els centres de les futures zones. No es permès que hi figurin codis d'unitat poblacional repetits. El centre geogràfic de la unitat poblacional seleccionada és l'origen de totes les distàncies per a la construcció de la seva zona.

2. Representar els centres geogràfics de les zones per mitjà de punts (x, y) que tenen com a úniques condicions, ésser únics i pertànyer a la regió d'estudi. Aquests punts aleatoris (x, y) són els orígens de les distàncies per a la construcció de la zona.

Per tant en el primer cas el cromosoma consta d'una cadena o llista de k enters dins l'interval $[1, n]$. I en el segon cas és una cadena o llista de duples (x, y) que designen un punt geogràfic al mapa de la regió d'estudi.

En ambdós casos, les zones es construeixen recorrent la llista d'unitats poblacionals i assignant-les al centre més proper.

2 Solució adoptada

En aquest apartat veurem quines són les restriccions que s'han fixat en el nostre particular cas, i assenyalarem les diferències amb les exposades a l'article.

2.1 Restriccions adoptades

El conjunt de restriccions del problema és:

2.1.1 Restriccions *soft*

1. Els valors poblacionals de les zones han de ser propers a la mitja de la població amb dret a vot, és a dir al resultat del quocient entre el total de població amb dret a vot i el número de zones desitjades

2. Les zones han de tenir una certa compacitat. A l'article es xerra només de zones circulars, nosaltres hem decidit experimentar i implementar una nova component a la funció de cost que doni una millor puntuació a zones rectangulars, quadrades i a les que limitin amb la mar

3. Les zones han de ser contínues. Aquesta restricció s'ha implementat com a quasi-*hard*, fent servir un pes molt gran per a penalitzar les zones no contínues dins la funció de cost, tal com descriu l'article

2.1.2 Restriccions *hard*

1. Una unitat poblacional no pot formar part de dues zones distintes

2. No poden existir zones buides

3. La unió de totes les zones ha de constituir la regió d'estudi

2.2 Decisions

Per tal de donar compliment a les restriccions del problema, la nostra implementació s'ha fet adoptant les següents decisions:

- Pel que fa a la codificació, s'ha optat per la segona, és a dir el genotip que codifica la solució és un cromosoma de k duples (x, y) (gens) que representen cada una un punt geogràfic vàlid, és a dir, que assenyalen un punt dins la regió d'estudi.
- La població inicial dels genotips es fa de forma aleatòria seguint una distribució uniforme pel que fa a la selecció de cada coordenada dels punts geogràfics
- La selecció de les parelles de pares es fa per torneig. El número de contrincants és quatre. Hem provat amb 8 com a número de contrincants sense apreciar canvis rellevants en el comportament de l'algorisme.
- L'operador de reproducció s'aplica amb una probabilitat del 95%, tal com a la solució descrita a l'article. Un mateix pare no es pot reproduir dues vegades dins la mateixa generació.
- La reproducció consisteix en la generació de dos fills per cada parella de pares. Per a tal finalitat el fill 1 hereta una part dels gens del cromosoma del pare 1 i la resta de gens del cromosoma del pare 2 fins a completar el seu propi cromosoma. El fill 2 hereta els gens que no ha heretat el seu germà.
- L'operador de mutació s'aplica amb una probabilitat de l'1% per a cada gen de cada cromosoma. La seva mutació consisteix en triar a l'atzar seguint una distribució uniforme, una nova dupla de valors (x, y) vàlids, que reemplaça el gen.
- Com a l'article, per a la selecció de la propera generació, també es segueix un esquema de elitista. En el nostre cas es seleccionen el 40% dels millors pares i per tant el 60% restant es completen amb els millors fills a fi de tenir constant el cardinal de la població. En el cas de que no es generassin el número suficient de fills, es seleccionen els millors pares necessaris per a completar la cardinalitat de la població que és constant per a totes les iteracions.
- El procés de cerca de la solució acaba de forma incondicional al cap de 99999 iteracions o al cap de 5000 iteracions sense que es produeixi millora. Consideram que 5000 iteracions són suficients per a garantir que cap zona queda buida i que la solució proposada complirà en un percentatge alt, les restriccions fixades.
- La funció d'avaluació de cada genotip, funció de cost o *fitness*, inclou tres components, una pel que fa a la desviació del valor poblacional de cada zona proposada respecte del valor que hauria de tenir (el valor mitjà), una segona pel que fa al que hem anomenat en aquest treball, el *cost de connectivitat de la zona*, i finalment una pel que fa a la contigüitat de la zona, a fi de poder penalitzar solucions amb zones no contigües.

El següent apartat el dedicam a definir el *cost de connectivitat de la zona* que feim servir com a segon sumand a la funció de *fitness*.

2.3 Cost de connectivitat de la zona

El concepte que denominam *cost de connectivitat de la zona* és un càlcul que resumeix numèricament dins el rang $(0, 1)$ quin és el cost de *circular* per dins la zona. És a dir, vist de forma intuïtiva, dona una estimació numèrica de com és de fàcil recórrer tota la zona. Zones conformades per unitats que comparteixen gran part dels seus perímetres i que per tant seria més fàcil circular per elles obtindran puntuacions baixes. Pel contrari zones amb unitats que estan unides per costats més petits i que per tant poden tenir parts més estretes, obtindran una puntuació del seu cost més alta i propera a 1. Per

tant s'espera que les zones amb cost baix siguin les rectangulars amb amplària similar al llarg de tot el seu perfil, les de forma quadrada o redona i les que limiten amb la línia de costa. En resum, s'ha optat per innovar amb aquest component de la funció de cost per tal d'experimentar amb un component distint dels proposats a l'article, que afavoreixen exclusivament les zones redones.

Per a poder avaluar aquest cost, quan es carrega el mapa de la regió d'estudi es calcula de forma aproximada quant de perímetre tenen en comú les parelles d'unitats poblacionals veïnes i el total del perímetre que cada unitat comparteix amb totes les unitats adjacents (per a les unitats que no limiten amb la mar, el perímetre total coincideix amb total del perímetre compartit).

Aleshores el *cost de travessar* de la unitat poblacional i a la seva veïna j s'ha definit com:

$$T_{ij} = 1 - \frac{P_{ij}}{\sum_{k \in N_i} P_{ik}}$$

A on P_{ij} és una aproximació als metres lineals que tenen en comú les unitats poblacionals i i j, N_i és el conjunt de unitats poblacionals adjacents (i.e. veïnes) a la unitat i i $\sum_{k \in N_i} P_{ik}$ és el sumatori del perímetre de la unitat i que està en contacte amb altres unitats de la regió d'estudi, inclosa la j.

Tots aquests *costs de travessa* entre unitats són constants al llarg de la vida del problema, doncs depenen de la geometria i ubicació de les unitats. Per tant un cop calculades es guarden dins una estructura del tipus diccionari per tal de facilitar el posterior càlcul dels *costs de connectivitat de cada zona* que es formen a cada nova iteració de l'algorisme. Aquest *cost de connectivitat de la zona* el calculam com la mitja de tots els costs de travessar d'una unitat a una altra dins la mateixa zona. És a dir:

$$C(Z_k) = \frac{1}{M} \sum_{i \in Z_k} \sum_{j \in (N_i \cap Z_k)} T_{ij}$$

A on M és el nombre de sumands i $N_i \cap Z_k$ és el conjunt d'unitats adjacents a la unitat i que també pertanyen a zona k.

2.4 Funció de cost seleccionada

Com ja s'ha esmentat, l'objectiu principal del treball és aconseguir zones geogràficament contínues amb un valor de població amb dret a vot proper a la mitja per zona. Un dels paràmetres que es pot modificar al nostre disseny és si disposam d'un cert marge de tolerància pel que fa a la consigna de població per zona. Per tal d'avaluar la idoneïtat de cada solució proposada s'han dissenyat dues funcions, utilitzant-ne una o l'altre segons si disposam d'aquest marge de tolerància o no.

Si no disposam d'aquest marge, la funció que es fa servir és:

$$\min \sum_{i=1}^{k} \left(\left| \frac{P_i}{\mu} - 1 \right| + C_i + w \cdot U_i \right)$$

A on P_i és el valor de la població de la zona i-èssima; μ és la mitja de la població; C_i és el *cost de connectivitat* calculat per a la zona i-èssima; U_i correspon al número de parts no connectades que formen la zona i-èssima, menys una unitat i finalment w és un pes gran (constant).

Pel cas en que sí tenguem un marge de tolerància, la funció que hem construït per tal d'avaluar la proposta és del tipus paraboloide:

$$\min \sum_{i=1}^{k} \left(R_i \cdot \left(\frac{\frac{P_i}{\mu} - 1}{m} \right)^2 + C_i + w \cdot U_i \right)$$

Amb

$$R_i = \begin{cases} r \in [0,1] & \text{si } \left(\frac{\frac{P_i}{\mu} - 1}{m} \right)^2 < 1 \\ 1 & \text{altrament} \end{cases}$$

A on $m \in (0,1)$ és el marge poblacional del que disposam expressat com a tant per unitat i R_i juga el rol de *reductor* en el cas de que la zona i-èssima assoleixi un valor poblacional dins els marges tolerats.

Per tal de clarificar el paper dels paràmetres R_i i m s'ha representat a les figures 1, 2 i 3 la funció que acabam de descriure per a diferents valors del marge poblacional (m), i el coeficient reductor (r), suposant que la zona i-èssima està constituïda per una sola regió (per tant $U_i = 0$) i que aquesta té un cost de connectivitat ideal (aleshores $C_i = 0$). L'eix x és el valor de $\left(\frac{P_i}{\mu} - 1\right)/m$.

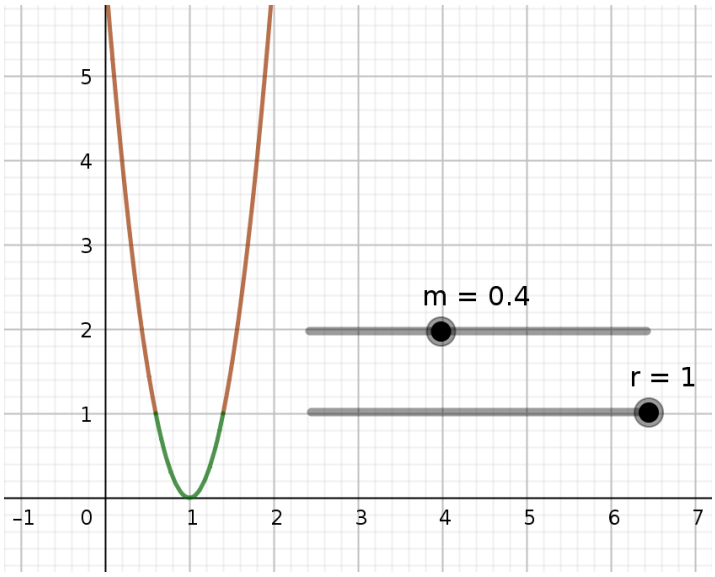

Figura 1: Funció de puntuació de la desviació poblacional per $m = 0.40$ i $r = 1$

Aquesta funció parabòlica té el següent comportament:

- Pel valor $P_i = \mu$, val zero
- Pels valors en que P_i coincideix amb $\mu \pm m \cdot \mu$, val 1
- Pels valors de $P_i \in (\mu - m \cdot \mu, \mu + m \cdot \mu)$, dona un valor entre 0 i 1
- Altrament dona un valor més gran que la unitat i proporcional al quadrat de la distància normalitzada entre P_i i μ

Pels valors en que el coeficient reductor r és menor que la unitat, s'observa l'aparició d'una discontinuïtat de la funció als punts $(1 - m, 1)$ i $(1 + m, 1)$ (figures 2 i 3).

Aquest coeficient reductor ens permet modular indirectament el pes del *cost de connectivitat*. En efecte, atès que el *cost de connectivitat* de la zona (C_i) és un valor dins el rang $(0, 1)$ que es addicionat en el càlcul de la puntuació de la zona, un valor de r proper a zero, prioritzarà zones amb un *cost de connectivitat* baix, pel contrari un valor de r proper a la unitat, donarà prioritat a zones amb valors poblacionals propers al valor consignat.

3 Implementació

La implementació de l'algorisme descrit a l'apartat 2 s'ha desenvolupat en llenguatge *Python 3*. Per tal de facilitar la depuració i reutilització de codi s'ha seguit un disseny descendent i orientat a objectes.

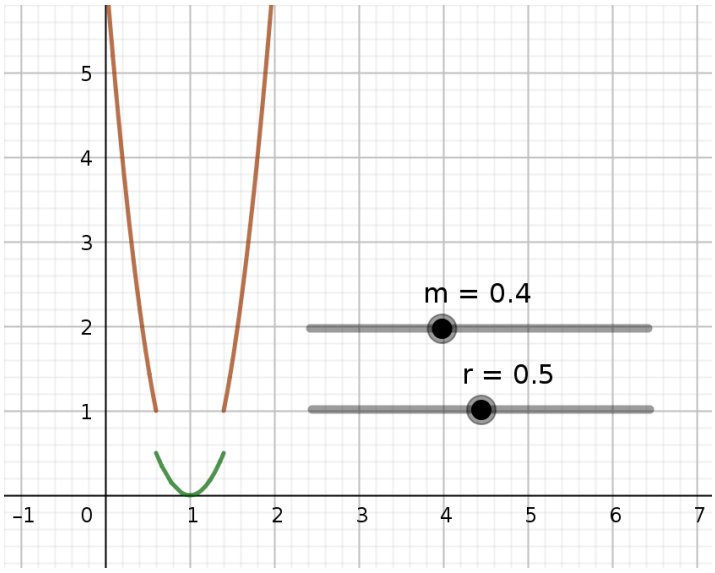

Figura 2: Funció de puntuació de la desviació poblacional per $m = 0.40$ i $r = 0.50$

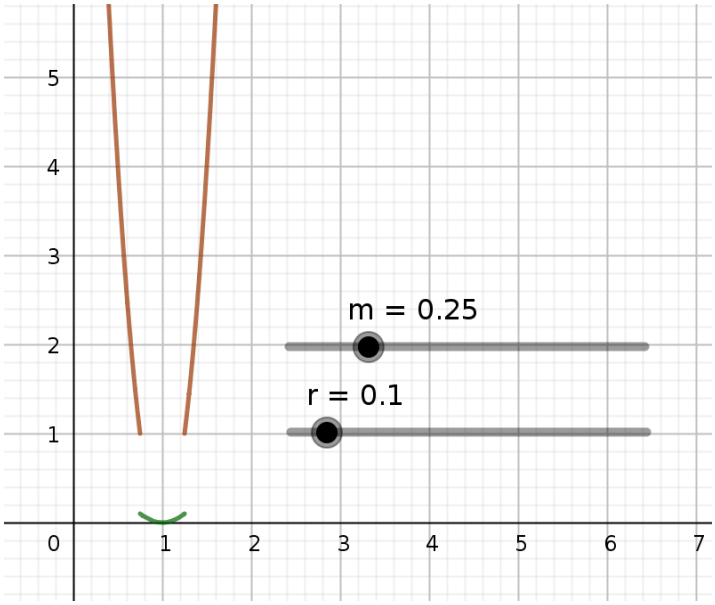

Figura 3: Funció de puntuació de la desviació poblacional per $m = 0.25$ i $r = 0.10$

Totes les funcions, classes, mètodes, i atributs es troben comentats en el propi codi. Tot i això en aquest mateix apartat es farà una descripció dels ítems més rellevants.

Les unitats poblacionals a les que fa referència l'article reben el nom de districtes (*districts* en anglès) a la nostra implementació.

El programari es divideix en 5 mòduls que descrivim a continuació:

- *mt_common.py*: aquest mòdul conté els literals dels missatges informatius i d'error, així com alguns hiper-paràmetres. Aquest són els principals hiper-paràmetres:
  - GA_NONIMPROV_ITERATIONS: valor 5000. Marca el número d'iteracions que han de

transcórrer per tal de donar per acabat el procés de cerca de la solució

- GA_CROSSOVER_PROB: valor 0.95. És la probabilitat de que una parella de pares qualsevol es reprodueixi
- GA_MUTATION_PROB: valor 0.01. Indica la probabilitat de mutació de cada gen. A l'article es fa servir un valor de 0.001
- GA_TOURNAMENT_ADVERSARIES: valor 4. És el número d'adversaris a l'hora de fer el torneig per a seleccionar els futurs pares
- GA_PARENTS_TO_HOLD: valor 0.40. Informa del tant per ú mínim de pares a mantenir entre generacions. A algunes generacions podrien ser més del 40% si el número de fills fos insuficient
- GA_MARGIN_ZONE_VALUE: valor 0.25. És el marge de tolerància del valor de població (m)
- GA_INTO_MARGIN_REDUCTION: valor 0.80. És el valor del coeficient reductor (r)
- GA_UNCONNECTED_ZONE_WEIGHT: valor 10. Pes del terme que informa del número de parts inconnexes a les zones (w)
- GA_1_DISTRICT_ZONE_MEAN_COST: valor 0.80. És el *cost de connectivitat* assignat a les zones de les que no se'ls pot calcular perquè nomes consten d'un districte

Basta canviar el valor de qualsevol d'aquests hiper-paràmetres en aquest mòdul perquè el programa ho tengui en compte i hi adapti el seu comportament a la propera execució.

- *mt_Zone.py*: en aquest mòdul hi ha definida la classe *Zone* per tal d'implementar objectes que representen una zona amb un determinat centre i amb aquestes funcionalitats:

  1. Afegir un districte a la zona
  2. Calcular el seu *cost de connectivitat*
  3. Computar el número de parts no connectades que la formen

  Els mètodes principals de la classe *Zone* són:

  - *add_district*: afegeix un districte a la zona
  - *calc_cost*: calcula el *cost de connectivitat* de la zona i el número de parts no contigües que la formen
  - *get_serialized_zone*: retorna un diccionari amb els codis de districtes que la formen. És utilitzada a l'hora de guardar la solució a disc

- *mt_Partition.py*: aquest mòdul implementa la classe *Partition* que realitza aquestes tasques:

  1. Generar una primera llista de centres de zona (genotip) per tal d'elaborar una proposta de partició (conjunt de zones) inicial
  2. Avaluar la bondat de la partició, aplicant-li la funció de cost (funció de *fitness*)
  3. Per a cada iteració, mutar amb probabilitat GA_MUTATION_PROB cada un dels seus centres (cada gen)

  Les seves principals funcions són:

  - *generate_genotype*: genera un conjunt aleatori de centres de zona vàlids (és a dir dins la regió d'estudi)
  - *compose_partition*: fa la composició de zones afegint cada districte al centre de zona més proper

- *evaluate*: avalua la puntuació del conjunt de zones que conté, calculant el valor de la funció de cost. Com més baix sigui, millor serà la solució
- *mutate*: per a cada gen produeix la seva mutació amb probabilitat GA_MUTATION_PROB
- *get_serialized_partition*: retorna una versió de la partició actual. S'utilitza per gravar la solució *json* a disc

- *mt_PartitionDesigner.py*: aquest mòdul implementa la classe *PartitionDesigner* que realitza aquestes tasques:

  1. Verificar la consistència de les dades alfanumèriques i geogràfiques
  2. Cercar una solució seguint les directrius marcades pels paràmetres de l'algorisme genètic
  3. Aplicar l'operador de reproducció (*crossover*)
  4. Gravar a disc una composició amb el mapa, un diagrama de barres amb les poblacions de cada zona i la gràfica de la puntuació per a cada estadi rellevant del progrés de l'algorisme
  5. Gravar a disc un fitxer *json* amb la solució proposada. El fitxer inclou per a cada zona (enumerades des de la zero fins a la $k-1$, la seva població i una llista dels codis de districte que la composen

  Les principals funcions que la conformen són:

  - *_generate_initial_population*: genera la població inicial, que consisteix en un conjunt d'objectes de la classe *Partition* a cada un dels quals se'ls demana que generin un conjunt de centres, tants com zones a dissenyar
  - *_compose_parents*: demana a tots els objectes *Partition* que composin les seves zones, assignant per cada districte al centre de zona més proper
  - *_evaluate_parents*: calcula la funció de *fitness* per a cada un dels objectes *Partition*
  - *_select_best_partition*: cerca i selecciona la millor solució, la que té la puntuació més baixa
  - *_update_our_score*: actualitza l'historial de puntuacions amb la millor
  - *_generate_parental_couples*: genera dues llistes de pares per tal de conformar les parelles a reproduir. Ho fa utilitzant el mètode del torneig
  - *_apply_crossover_operator*: reprodueix cada una de les parelles de pares utilitzant una probabilitat GA_CROSSOVER_PROB
  - *_apply_mutation_offspring*: demana a cada objecte fill que apliqui amb probabilitat GA_MUTATION_PROB una mutació a cada un dels seus gens
  - *_compose_offspring*: de forma semblant a la funció _compose_parents, demana a tots els objectes fills que composin les seves zones
  - *_evaluate_offspring*: aplica la funció de *fitness* per a cada un dels objectes fills (instàncies de *Partition*)
  - *_select_next_generation*: guarda com a mínim un tant cent dels millors pares (GA_PARENTS_TO_HOLD) i la resta dels millors fills fins a completar el número d'objectes de població necessaris per mantenir la població constant
  - *save_best_map*: estalvia a disc una composició gràfica de la solució actual (l'objecte *Partition* amb la puntuació més baixa)
  - *save_best_solution_file*: guarda en disc un fitxer *json* amb la solució
  - *fit*: fa servir totes les funcions descrites a aquesta llista per tal de trobar una solució i gravar-la a disc. L'usuari només necessita crear una instància de l'objecte d'aquesta classe, i després cridar aquest mètode per a desencadenar l'algorisme genètic i trobar una solució.

- mt_main.py: aquest mòdul conté el bucle principal de la nostra implementació i tres funcions. Els seus rols principals són:

  1. Carregar les dades geomètriques de les unitats poblacionals, i.e. el mapa amb els polígons que representen les unitats poblacionals i el codi d'unitat assignat a cada una.
  2. Carregar el mapa amb el perímetre de la regió d'estudi.
  3. Carregar les dades alfanumèriques que inclouen el codi assignat a cada unitat poblacional i el seu valor de població. Cada un d'aquests codis s'ha de correspondre amb una de les representacions geomètriques per tal de poder dibuixar correctament el mapa de les zones proposades.
  4. Adaptar els noms dels camps de codis i geometries als que esperen els objectes implementats
  5. Canviar la projecció de les capes geomètriques a metres per tal de poder calcular les distàncies de forma correcta
  6. Per a cada unitat poblacional, determinar quines són les seves veïnes i quin és el *cost de travessar* a cada una d'elles (veure 2.3)
  7. I finalment executar la cerca d'una solució per a cada parella de valors de les dues llistes que hi ha definides:
     - NUM_ZONES: és la llista de valors que es faran servir com a quantitat de zones a cercar. Per defecte [2, 3, 8, 10, 20]
     - POPULATION_CARDINALITIES: és una llista de valors que indica el número d'individus a utilitzar. Per defecte [10, 20]

  Les funcions que conté són:

  - *main*: és el bucle principal del programa
  - *make_my_neighbours_lists*: confecciona una dupla de llistes, on la primera conté els codis de les unitats adjacents a una donada (que anomenam *me*) i la segona llista conté l'esmentat *cost de travessar* des de la unitat *me* a cada una de les unitats adjacents
  - *make_dist_conn_dict*: computa un diccionari que conté una entrada per a cada unitat poblacional amb les dades dels seus veïns que li subministra la funció *make_my_neighbours_lists*
  - *prepare_data*: carrega des de disc els dos fitxers de dades geomètriques (unitats i límits de la regió) i els projecta en metres; carrega el fitxer alfanumèric; fa els canvis dels noms dels camps de codi i geometria adients i crida a la funció *make_dist_conn_dict* per tal d'obtenir els veinats de cada unitat i els *costs de travessar* d'una a l'altre

4 Format de les dades d'entrada

Per tal de poder treballar amb aquest programa es necessiten preparar tres fitxers que descrivim tot seguit:

- Fitxer de dades alfanumèriques:

  - Format: *csv* amb capçalera i amb el signe de puntuació punt i coma com a separador de valors
  - Codificació: utf-8
  - Número de registres: un registre de dades per a cada districte que vulguem processar més el primer registre amb la capçalera
  - Camps obligatoris:

        * Camp alfanumèric que identifiqui de forma unívoca el districte
        * Camp numèric amb el total de població amb dret a vot que hi ha censada al districte
    - La resta de camps són ignorats

- Fitxer de capa geogràfica amb la representació de tots els districtes:
    - Format: *geojson*
    - Projecció: pot ser qualsevol, però ha d'estar informada al fitxer, perquè un cop carregada en memòria el programa canvia la projecció a metres utilitzant la EPSG:3857
    - Codificació: *utf-8*
    - Número de registres: un registre de dades per a cada districte que vulguem processar
    - Camps obligatoris:
        * Camp alfanumèric que identifiqui de forma unívoca el districte
        * Camp *geometry* amb el polígon que representa el perfil del districte
    - La resta de camps són ignorats

- Fitxer de capa geogràfica amb la representació del llindar de la regió d'estudi i/o llegendes:
    - Format: *geojson*
    - Projecció: el mateix requeriment que a la projecció del fitxer geogràfic dels districtes
    - Codificació: *utf-8*
    - Número de registres: com a mínim un
    - Camps obligatoris:
        * Camp *geometry* amb el polígon que representa el perfil de la regió d'estudi o les llegendes a representar en el mapa de la solució
    - La resta de camps són ignorats

4.1 Verificacions

Després de la càrrega dels fitxers, l'objecte de la classe *PartitionDesigner* verifica que per a cada districte definit al fitxer alfanumèric, existeix un registre amb el mateix codi al fitxer de la capa geogràfica dels districtes i que el seu camp *geometry* contengui una forma tancada per tal de poder calcular el seu centre geomètric.

Els registres de la capa geogràfica de districtes que no es corresponguin amb cap dels registres alfanumèrics, són ignorats i no es representaran al mapa de la solució.

Per tal de que el càlcul del perímetre que tenen en comú dos districtes sigui acurat i que es pugui computar quins són els seus districtes veïns, no han d'existir *gaps* artificials entre districtes adjacents.

5 Exemples utilitzant Mallorca

Es subministra un joc de tres fitxers relatius a Mallorca, on cada barriada de Palma, juntament amb cada poble, són els distints districtes electorals. Malauradament no hem pogut trobar les dades del cens electoral per barriades de Palma, així que en lloc de subministrar el cens electoral d'un any en concret, hem elaborat com a fitxer alfanumèric, un fitxer del tancament del padró de l'any 2020 utilitzant les dades publicades a [2].

Per a confeccionar els fitxer de dades geogràfiques dels districtes hem combinat dues fonts, una com a base cartogràfica i confecció dels perfils dels pobles [3] i l'altre per a dibuixar dins el recinte de Palma les seves barriades [4].

El fitxer de la capa geogràfica de la línia de costa de Mallorca també s'ha obtingut per manipulació de les dades obtingudes a [3]

El fitxer de dades alfanumèriques, el fitxer de districtes i el de línia de costa es troben respectivament a:

- *habs/PAD2020.csv*
- *maps/products/districts_geometry.geojsonl.json*
- *maps/products/coast_line_geometry.geojsonl.json*

Als apartats que segueixen s'exemplifiquen 5 casos amb diferents graus de dificultat.

5.1 Cas 1: computació de 2 zones

Per a computar aquest cas s'han fet servir els paràmetres següents:

- Número de zones: 2
- Població: 10 particions
- Probabilitat de reproducció: 0.95
- Probabilitat de mutació: 0.01
- Número d'adversaris al torneig de selecció de pares: 4
- Percentatge mínim de millors pares a estalviar: 40%
- Marge del valor poblacional: $\pm 25\%$
- Coeficient reductor r: 0.5
- Pes de les parts no connectades w: 10
- Cost de connectivitat imputat a les zones d'un sol districte: 0.8
- Límit del número d'iteracions no productives: 5000

Comparant les dues darreres instantànies de l'evolució de l'algorisme, figures 4 i 5, es pot apreciar com la component del *cost de connectivitat* a la funció de cost es fa patent, seleccionant com a millor solució, una que conté la zona de color cel amb més línia de costa.

L'exemple complet es troba a la carpeta *outputs/sol-nz-02-pc-10-ok*.

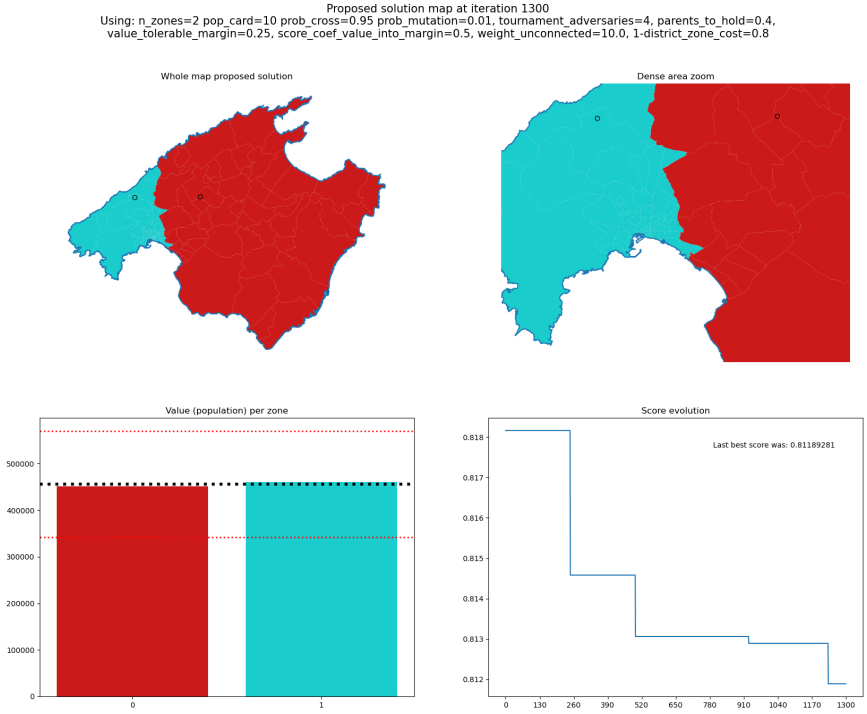

Figura 4: Resum gràfic de les 2 zones just abans de la solució

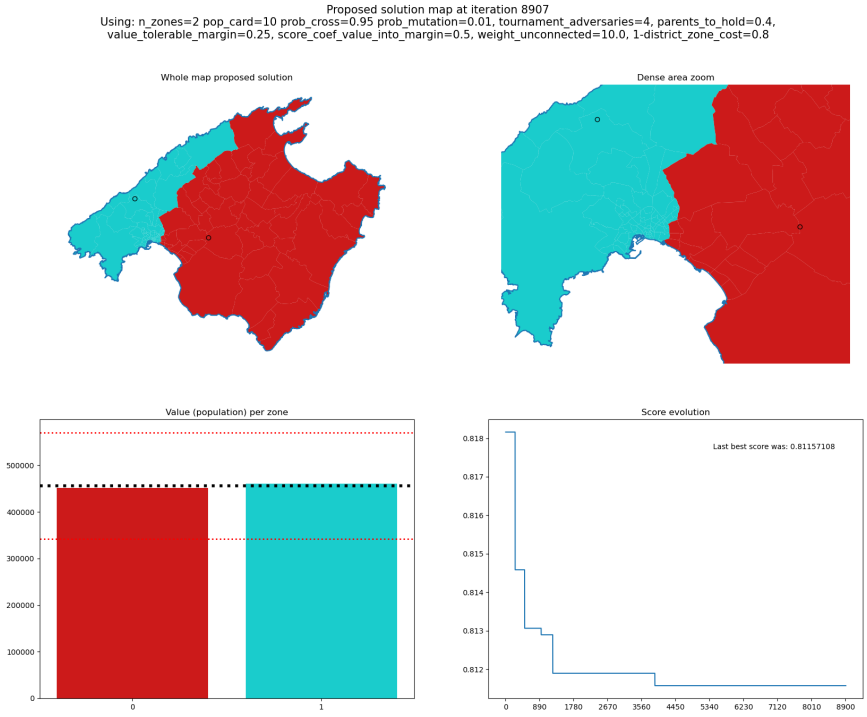

Figura 5: Resum gràfic de la solució trobada per a 2 zones

5.2 Cas 2: computació de 3 zones

Per a calcular aquest cas s'han fet servir els paràmetres següents:

- Número de zones: 3
- Població: 10 particions
- Probabilitat de reproducció: 0.95
- Probabilitat de mutació: 0.01
- Número d'adversaris al torneig de selecció de pares: 4
- Percentatge mínim de millors pares a estalviar: 40%
- Marge del valor poblacional: $\pm 25\%$
- Coeficient reductor r: 0.5
- Pes de les parts no connectades w: 10
- Cost de connectivitat imputat a les zones d'un sol districte: 0.8
- Límit del número d'iteracions no productives: 5000

De forma semblant al cas anterior, comparant les dues darreres instantànies de l'evolució de l'algorisme, figures 6 i 7, es pot apreciar com la funció de cost selecciona una zona amb més línia de costa, malgrat la zona de color vermell surt *perjudicada* i adopta un perfil amb més punta i per tant pitjor connectivitat (és més estreta).

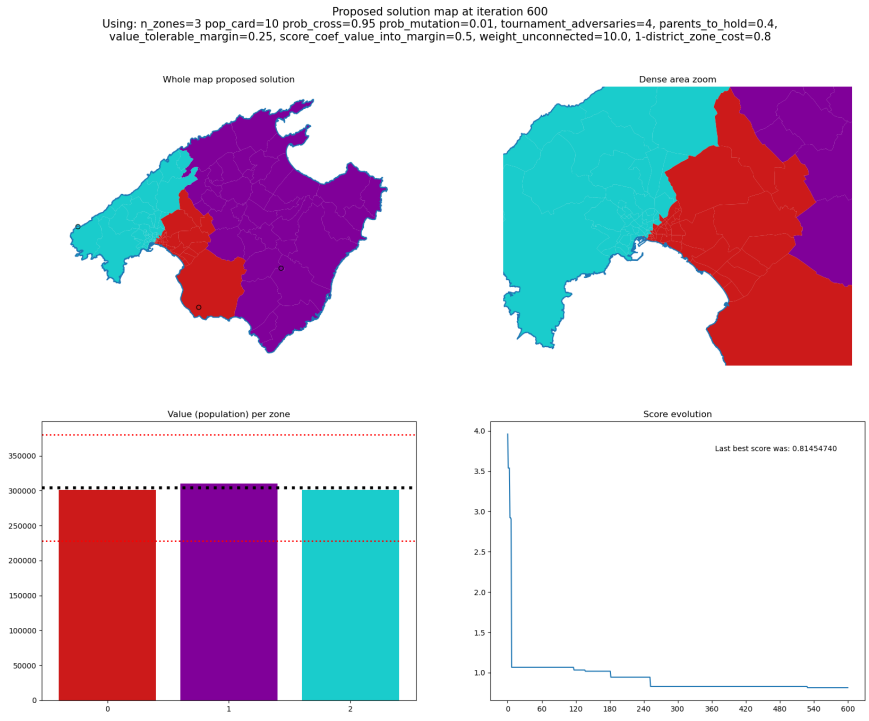

Figura 6: Resum gràfic de les 3 zones just abans de la solució

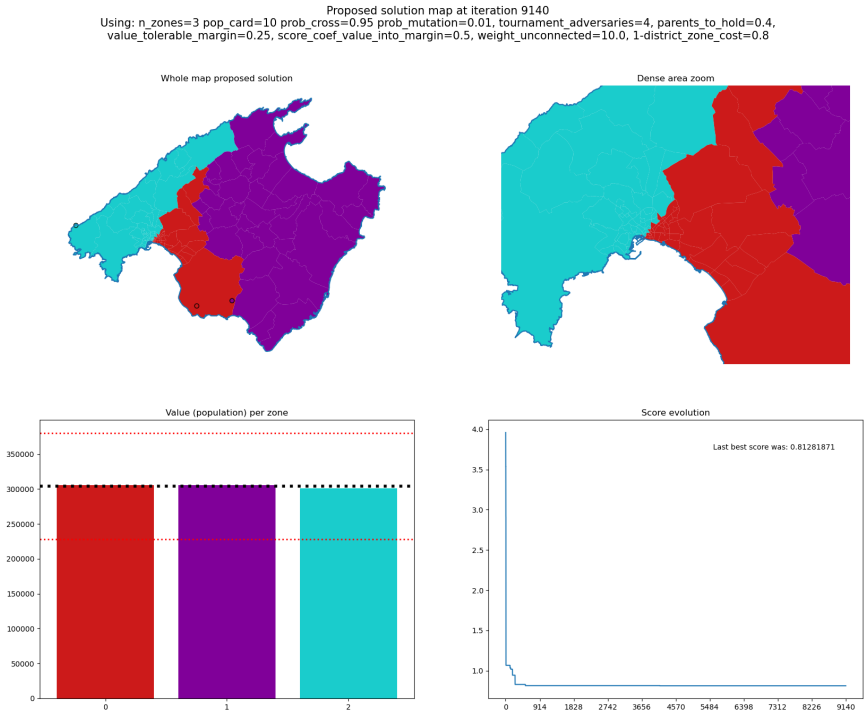

Figura 7: Resum gràfic de la solució trobada per a 3 zones

L'exemple complet es troba a la carpeta *outputs/sol-nz-03-pc-10-ok*.

5.3 Cas 3: computació de 8 zones

Per a aquest cas s'han fet servir els paràmetres següents:

- Número de zones: 8
- Població: 10 particions
- Probabilitat de reproducció: 0.95
- Probabilitat de mutació: 0.01
- Número d'adversaris al torneig de selecció de pares: 4
- Percentatge mínim de millors pares a estalviar: 40%
- Marge del valor poblacional: $\pm 25\%$
- Coeficient reductor r: 0.5
- Pes de les parts no connectades w: 10
- Cost de connectivitat imputat a les zones d'un sol districte: 0.8
- Límit del número d'iteracions no productives: 5000

Comparant les dues imatges, l'anterior a la solució i la de la solució (figures 8 i 9 respectivament), es pot apreciar que la zona nord-est guanya una millor connectivitat, recuperant un dels districtes de la zona veïna verda i configurant-se així amb un perfil més regular.

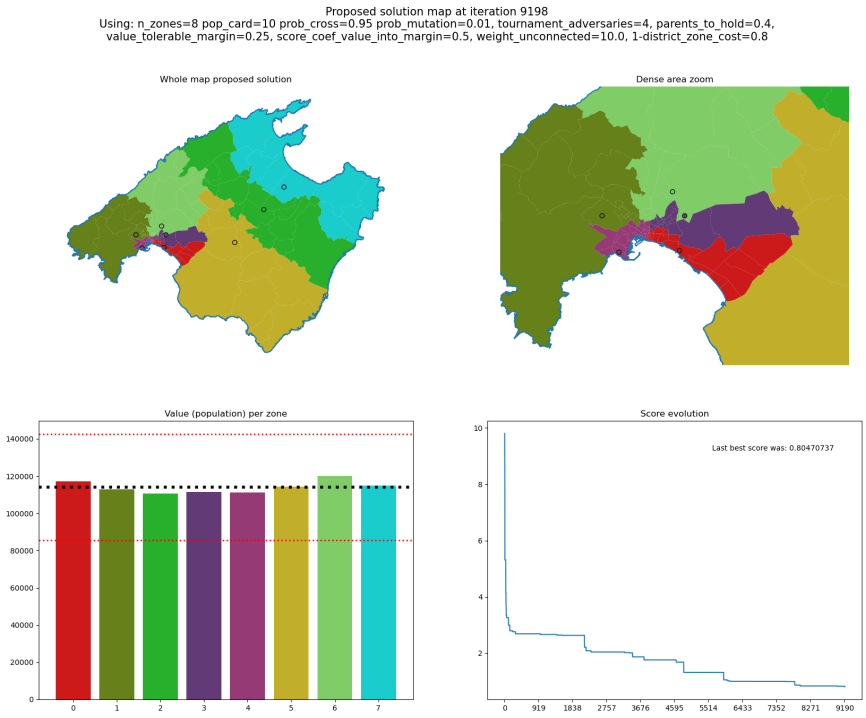

Figura 8: Resum gràfic de les 8 zones just abans de la solució

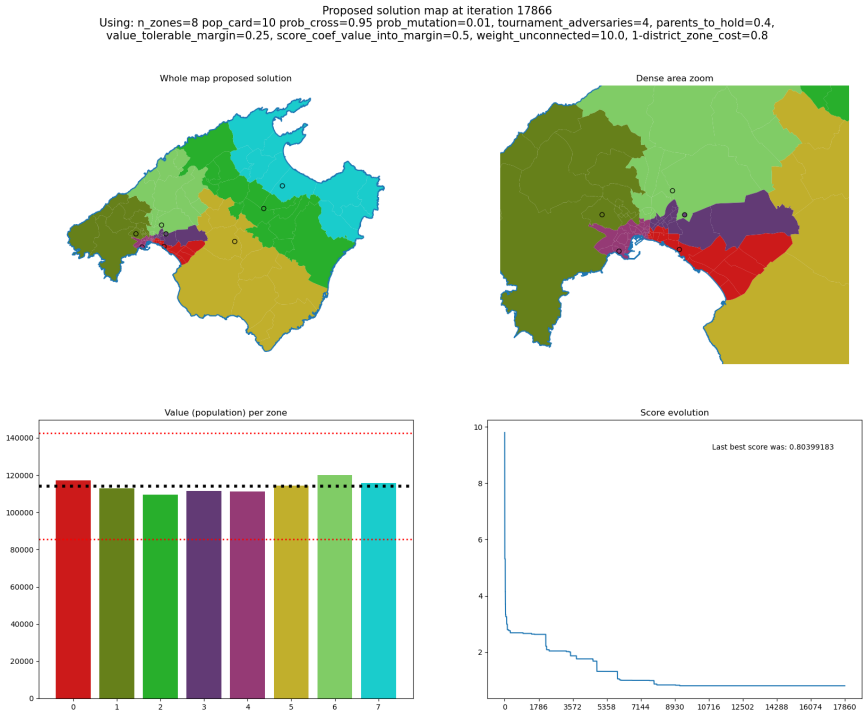

Figura 9: Resum gràfic de la solució trobada per a 8 zones

L'exemple complet es troba a la carpeta *outputs/sol-nz-08-pc-10-ok*.

5.4 Cas 4: computació de 20 zones utilitzant 10 particions

Per a computar les zones d'aquest cas s'han fet servir el paràmetres següents:

- Número de zones: 20
- Població: 10 particions
- Probabilitat de reproducció: 0.95
- Probabilitat de mutació: 0.01
- Número d'adversaris al torneig de selecció de pares: 4
- Percentatge mínim de millors pares a estalviar: 40%
- Marge del valor poblacional: $\pm 25\%$
- Coeficient reductor r: 0.8
- Pes de les parts no connectades w: 10
- Cost de connectivitat imputat a les zones d'un sol districte: 0.8
- Límit del número d'iteracions no productives: 5000

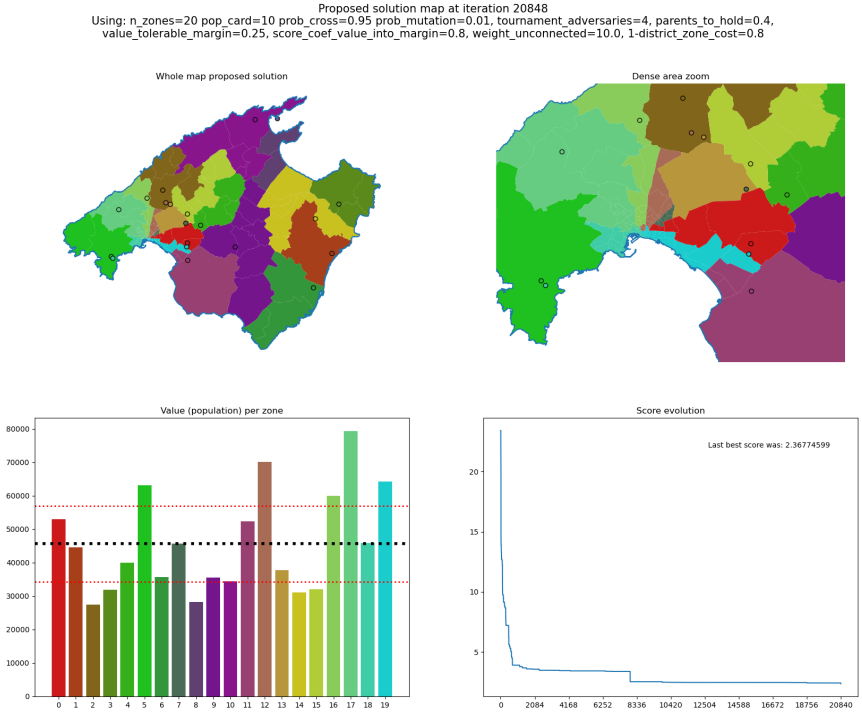

Figura 10: Resum gràfic de les 20 zones just abans de la solució

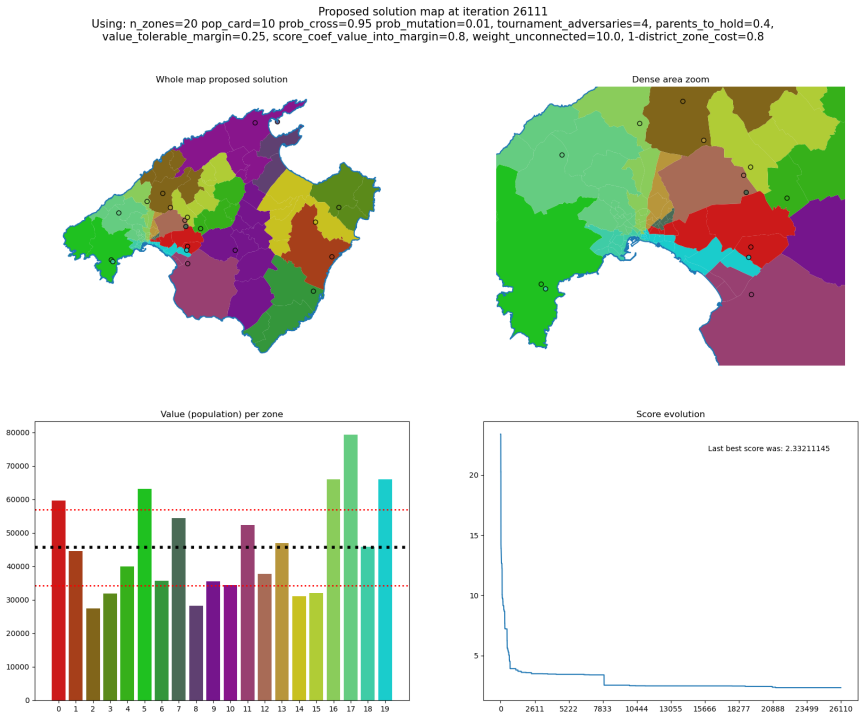

Figura 11: Resum gràfic de la solució trobada per a 20 zones

La meitat de les zones (10 de 20) han quedat fora dels marges de població tolerats (±25%). Per mitigar aquest comportament en aquest cas més difícil, s'opta en el proper cas 5.5 per repetir la cerca utilitzant 20 particions.

L'exemple complet d'aquest cas es troba a la carpeta *outputs/sol-nz-20-pc-10-out10*.

5.5 Cas 5: computació de 20 zones utilitzant 20 particions

Per a aquest darrer cas s'han fet servir el paràmetres següents:

- Número de zones: 20
- Població: 20 particions
- Probabilitat de reproducció: 0.95
- Probabilitat de mutació: 0.01
- Número d'adversaris al torneig de selecció de pares: 4
- Percentatge mínim de millors pares a estalviar: 40%
- Marge del valor poblacional: ±25%
- Coeficient reductor r: 0.8
- Pes de les parts no connectades w: 10
- Cost de connectivitat imputat a les zones d'un sol districte: 0.8
- Límit del número d'iteracions no productives: 5000

L'augment de la població passant de 10 a 20 particions, sembla haver millorat el comportament de l'algorisme, fins al punt de trobar una solució amb només 5 zones fora dels marges poblacionals tolerats. Pensam que un augment del límit del número d'iteracions produiria un millor resultat.

L'evolució d'aquest exemple complet es troba a la carpeta *outputs/sol-nz-20-pc-20-out05*.

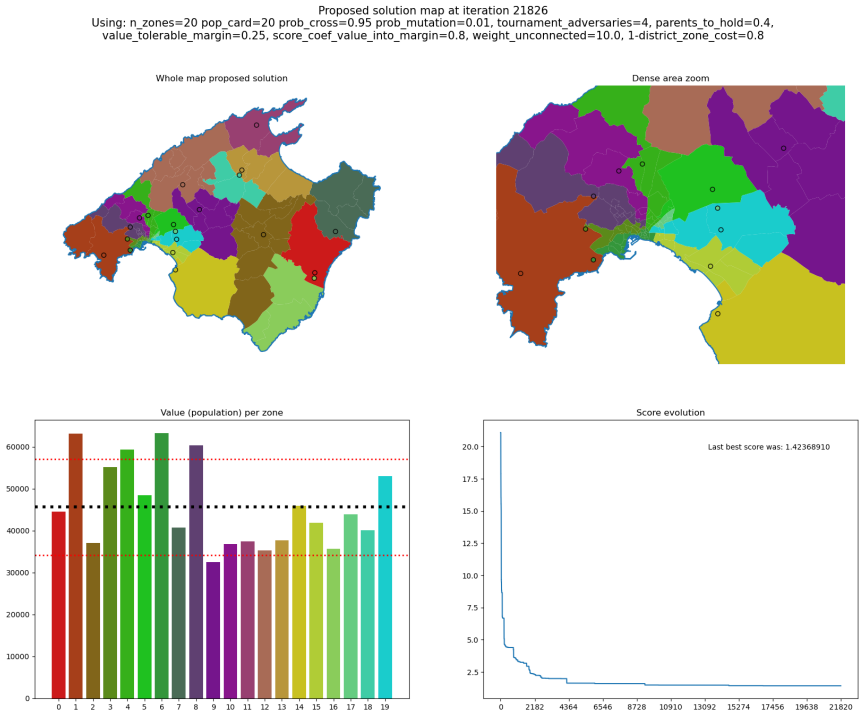

Figura 12: Resum gràfic de la solució trobada per a 20 zones, fent servir 20 particions

5.6 Exemple de fitxer amb solució

A l'acabament del procés de recerca de la solució es grava a disc un fitxer en format *json* que defineix un diccionari amb una entrada per a cada zona, numerades des del zero. Dins cada entrada de zona hi ha niat un segon diccionari amb una entrada per a cada districte i en el seu torn, un tercer nivell de diccionari que inclou tres entrades:

- "1": Valor de població del districte
- "2": Distància en metres des del districte al centre de la zona
- "3": Una llista de tres llistes:
  - "CENTROID": Coordenades del centroide del districte
  - "NEIGHBOURS_CODE_LIST": Llista dels codis dels districtes veïns
  - "NEIGHBOURS_COST_LIST": Llista del *cost de travessar* des del districte a cada un dels seus veïns

En el llistat 1 hi trobareu un exemple de bolcat de proposta de solució.

```
{
  "0": {
    "07033": {
      "1": 44527,
      "2": 10531.89566327439,
      "3": {
        "CENTROID": [360166.2329961238, 4801471.253766101],
        "NEIGHBOURS_CODE_LIST": [
          "07051", "07022", "07041", "07065"
        ],
        "NEIGHBOURS_COST_LIST": [
          0.653422, 0.707929, 0.748865, 0.889784
        ]
      }
    }
  },
  "1": {
    "07005": {
      "1": 11433,
      "2": 11961.135520401374,
      "3": {
        "CENTROID": [267760.62484048156, 4805381.809370395],
        "NEIGHBOURS_CODE_LIST": [
          "07011", "07021"
        ],
        "NEIGHBOURS_COST_LIST": [
          0.134141, 0.865859
        ]
      }
    },
    "07011": {
      "1": 51710,
      "2": 3593.66521419717,
      "3": {
        "CENTROID": [279201.10827797395, 4800924.206823889],
        "NEIGHBOURS_CODE_LIST": [
          "07005", "07045", "07040N18", "07040N36", "07021", "07040N14", "07040N32"
        ],
        "NEIGHBOURS_COST_LIST": [
          0.663661, 0.676051, 0.888152, 0.889767, 0.898778, 0.984714, 0.998877
        ]
      }
    }
  },
  "2": {    },
}
```


Listing 1: Bolcat de les dues primeres zones ("0" i "1") d'una solució

6 Conclusions

Els algorismes genètics són eines molt potents per tal de trobar als problemes amb espais de solucions molt grans, un bon resultat. Requereixen però d'una anàlisis prèvia acurada per tal de determinar el conjunt de restriccions i la millor representació de la solució (genotip). El grau de dificultat de la seva implementació depèn en gran mesura de la complexitat del genotip i de les restriccions a reflectir a la funció de cost (*fitness*).

Com a inconvenient destacable esmentar que per a resoldre el cas que se'ns presenta, i molts d'altres, cal un temps d'execució important que no es pot determinar al manco *a priori*.

Un altre factor important a tenir en compte és l'elecció de la funció de cost, doncs ha de garantir que al llarg de l'execució de les iteracions, la solució vagi convergint cap al punt teòric desitjat. La modificació dels paràmetres de l'algorisme també permet ajudar a aquesta convergència, si bé es tracta d'un ajust amb un important component heurístic.

Pel que fa al nostre particular experiment d'implementar una funció de cost que tengui en compte la regularitat de la forma de la zona, pensam que no ha anat del tot bé, perquè hem observat que en el cas del disseny de 20 zones, les zones que genera a prop de les regions amb gran densitat de població tenen perfils cònics en lloc de rectangulars. Pel cas de 8 o menys zones, sí pensam que el resultat és l'esperat.

Moltes de gràcies per tot! :-)

*** FI ***

Referències

[1] Fernando Bação, Victor Lobo i Marco Painho. "Applying genetic algorithms to zone design". A: *Soft Computing* 9.5 (maig de 2005), pàg. 341 - 348. ISSN: 1433-7479. DOI: 10.1007/s00500-004-0413-4. URL: https://doi.org/10.1007/s00500-004-0413-4.

[2] IBEstat. *Institut d'Estadística de les Illes Balears*. https://ibestat.caib.es/ibestat/estadistiques/poblacio. Accessed on 2022-03-06. Des. de 2020.

[3] Instituto Geográfico Nacional. *Centro de descargas*. http://centrodedescargas.cnig.es/CentroDescargas/buscadorCatalogo.do. Accessed on 2021-11-06. Nov. de 2020.

[4] Ajuntament de Palma. *Les barriades de Palma*. https://llengua.palma.cat/index.php?title=Les_barriades_de_Palma. Accessed on 2021-11-06. Nov. de 2020.