Bessó Digital d'una Xarxa de Trànsit Sistema de Suport a la Decisió

Carles Andreu Torreblanca

Resum– Les xarxes de trànsit defineixen sistemes altament complexos de gestionar que es veuen afectats per múltiples paràmetres. Aquest estudi analitza l'aplicació i compara el rendiment de diferents algoritmes de cerca de rutes, amb pas per diferents punts d'interès, tenint en compte les condicions del trànsit en una xarxa simulada. S'inclouen els algoritmes Dijkstra, Greedy i A*, i es valida l'ús de diferents heurístiques per adaptar les rutes calculades al trànsit de la xarxa. A continuació, es proposa l'ús d'algoritmes genètics per optimitzar el trànsit d'una xarxa, obtenint resultats prometedors. Per dur a terme l'estudi s'utilitza com a base el *framework* de simulació SUMO i les funcionalitats principals desenvolupades es presenten mitjançant una nova interfície gràfica.

Paraules clau– Xarxa de trànsit, rutes, grafs, cerca, optimització, heurístiques, simulació, Greedy, A*, Dijkstra.

Abstract–
Traffic networks define highly complex systems to manage that are affected by multiple parameters. This study analyzes the application and compares the performance of different route search algorithms, passing through various points of interest, taking into account traffic conditions in a simulated network. The algorithms Dijkstra, Greedy, and A* are included, and the use of different heuristics to adapt the calculated routes to network traffic is validated. Furthermore, the use of genetic algorithms to optimize network traffic is proposed, yielding promising results. The SUMO simulation framework serves as the basis for conducting the study, and the main functionalities developed are presented through a new graphical interface.

Keywords– Traffic network, routes, graphs, search, optimization, heuristics, simulation, Greedy, A*, Dijkstra.

1 Introducció - Context del treball

Una xarxa de trànsit, també coneguda com a xarxa de transport o xarxa de mobilitat, és un conjunt d' infraestructures i sistemes de transport interconnectats que permeten el moviment de persones, béns i serveis entre diferents indrets geogràfics. Un operari de xarxes de trànsit, per altra banda, és aquella persona que desenvoluparà la tasca de manteniment, millora i gestió d'aquestes xarxes. Aquesta tasca és una feina força complexa ja que depèn de molts factors referents a la xarxa sobre la que s'està exercint dita tasca. aquests factors poden ser el tipus de trànsit, l'estructura de la xarxa, hora del dia, zona geogràfica, entre molts altres. És per això que la motivació d'aquest document és la de comprendre millor com funcionen les xarxes de trànsit. Amb tot això, també es vol crear un eina capaç de donar suport als operaris de trànsit a desenvolupar la seva feina de forma més senzilla. Aquesta motivació es vol plasmar en tres objectius principals.

- Elaborar un anàlisi d'algorismes de càlcul de ruta òptima tenint en compte el trànsit d'una xarxa
- Desenvolupar una eina per a optimitzar xarxes de trànsit
- Desenvolupar una interfície gràfica perquè un operari pugui interactuar amb les diferents funcionalitats desenvolupades

Com es comenta als punts anteriors primer de tot es vol realitzar un anàlisi d'algorismes per a generar rutes dins d'una xarxa de trànsit en funcionament. Per a realitzar aquesta tasca s'haurà d'investigar sobre el funcionament del trànsit i sobre quins algorismes són els més adequats per a assolir aquest objectiu.

Un cop complert el primer objectiu, es vol focalitzar l'estudi en l'optimització de xarxes, és a dir, modificar una xarxa en concret per a que millori en alguna o vàries de les mètriques que determinin el seu rendiment.

• E-mail de contacte: carlesandreu4@gmail.com
• Menció realitzada: Computació
• Treball tutoritzat per: Lluis Echeverria Rovira (Computer Science)
• Curs 2022/23

Finalment es volen agrupar les funcionalitats desenvolupades en els dos punts anteriors en una interfície gràfica. Aquesta interfície gràfica estarà pensada per a que l'usuari, o operari de xarxes de trànsit, pugui interactuar amb les diferents funcionalitats que s'han dut a terme en el compliment dels dos primers objectius. Aquest objectiu està directament vinculat a la motivació principal del document, és a dir, facilitar la feina de l'operari de xarxes de trànsit.

Un cop descrit el problema a afrontar i esmentats els objectius a complir, es farà un breu resum de l'estructura que obtindrà el present document. Primer es començarà parlant de l'estat de l'art actual i la metodologia utilitzada. Seguit a la metodologia s'explicaran les diferents funcionalitats desenvolupades per al compliment d'objectius, i finalment s'exposaran els resultats i conclusions obtingudes.

2 Estat de l'Art

Els algorismes de cerca de rutes òptimes s'utilitzen en diversos camps, com ara transport i logística, sistemes de navegació, robòtica, xarxes de transport públic i optimització de xarxes de comunicació. Aquests algorismes permeten determinar la millor ruta entre dos punts en una xarxa o graf, tenint en compte diversos factors com el trànsit, les restriccions vials i les preferències de l'usuari. Un exemple és l'aplicació Google Maps, aquesta aplicació calcula rutes entre un punt inicial i un de final. També cap la possibilitat d'afegir fins a 10 parades a la ruta.

En el camp de l'optimització, els algorismes genètics tenen un paper important. Aquest tipus d'algorismes busquen aconseguir una solució propera a la òptima de problemes computacionalment molt complexes. Un exemple pot ser la d'optimitzar la estructura d'una xarxa neuronal o la de buscar els millors paràmetres en un model d'aprenentatge computacional[18].

En l'actualitat es pot trobar una gran quantitat d'articles que parlen sobre les xarxes de trànsit, el propi trànsit i la presa de decisions al moment de decidir una ruta. En l'article *Analysis of the Conflict between Car Commuter's Route Choice Habitual Behavior and Traffic Information Search Behavior*[23] es parla sobre un estudi de la decisió de les rutes. Es centra en el conflicte que hi ha entre escollir una ruta habitual i escollir una ruta segons informació del trànsit. L'article conclueix exposant que a la llarga preval la decisió habitual tot i que en un primer moment segons la informació del trànsit pogués haver canviat. Tot i que l'article tracti més sobre un fenomen que de l'aplicació d'algorismes al càlcul de rutes s'ha trobat força interessant i amb aspectes en comú amb l'estudi del present document, com ara els factors tinguts en compte a l'hora de decidir quina és la millor ruta.

En l'article *Application of Genetic Optimization Algorithm in Financial Portfolio Problem* [19] es parla sobre la utilització d'algorismes genètics per resoldre el model de variància mitja de Markowitz[20]. Aquest model busca trobar una distribució d'inversions en una cartera. La resolució d'aquest problema és força complex i per això s'utilitza aquest tipus d'algorisme. L'article explica que la utilització d'un algorisme genètic funciona molt bé per a trobar una solució al problema, no obstant, també es parla de les limitacions d'aquest tipus d'algorisme. Es comenta a l'article algorismes genètics funcionen molt bé en cerques globals, però que té les capacitats limitades quan es tracta d'una cerca local.

En l'article *Genetic Algorithm for TMS Coil Position Optimization in Stroke Treatment*[21] es parla sobre la tècnica d'estimulació magnètica transcranial (EMT)[22], una tècnica no invasiva per estimular el cervell humà el qual té un error considerable en l'error de col·locació de la bobina. En l'article es busca optimitzar la col·locació de la bobina per a realitzar aquest tractament aconseguint una considerable millora. L'algorisme utilitzat a l'estudi millora un 17,48% la intensitat del camp elèctric amb la posició de la bobina optimitzada.

Per últim es vol destacar un article en el que es busca calcular rutes òptimes utilitzant algorismes genètics. *An Optimal Round-Trip Route Planning Method for Tourism Based on Improved Genetic Algorithm* [24]. L'estudi busca optimitzar rutes turístiques que acabin al inici, és a dir, circulars. Les rutes turístiques acostumen a tenir una gran quantitat de parades pel que fa que la solució òptima sigui complexa d'aconseguir.

Un cop vist quin és l'estat de l'art actual en relació al problema plantejat s'explicarà la metodologia utilitzada per assolir els objectius.

3 Metodologia

En aquesta secció s'explicaran els mètodes i tecnologies utilitzades per a aconseguir els objectius proposats. Donat que es volen tractar objectius amb certa complexitat, que requereixen d'un desenvolupament àgil i flexible, i a més també es vol tocar diferents branques dins de la informàtica, des de un principi es va pensar en el llenguatge de programació Python. Tot i que aquest llenguatge pot tenir algun inconvenient com ara la seva baixa velocitat d'execució comparat amb altres llenguatges, com ara C o Java, s'han tingut en compte altres característiques molt positives i que beneficien molt al desenvolupament del projecte proposat. Algunes d'aquestes característiques per les que s'ha escollit aquest llenguatge són la flexibilitat, és a dir, Python és un llenguatge molt versàtil i és capaç de dur a terme una gran quantitat de diferents tipus de funcionalitats, des de la implementació d'algorismes fins a la creació de serveis web. La rapidesa de desenvolupament també és una característica que s'ha tingut en compte positivament, Python no és un dels llenguatges més ràpids, però si que ho és en quant a l'escriptura de codi, és a dir, generalment una funcionalitat costarà menys temps desenvolupar-la en Python que en altres llenguatges, això es deu principalment a la seva sintaxis. Per a la part d'interfície gràfica s'ha optat per utilitzar tecnologies web basades en Html, Css i Javascript. Aquestes tecnologies serviran exclusivament per a que l'usuari pugui interactuar amb les funcionalitats desenvolupades en Python. S'han escollit aquestes tecnologies ja que estan molt estandarditzades i donen molta flexibilitat a l'hora de crear interfícies gràfiques, de la mateixa forma també donen la possibilitat de realitzar aquesta interfície amb el nivell de complexitat que sigui requerit. Aquest fet és important ja que encara que s'ha volgut tenir desenvolupada una interfície gràfica per a l'usuari, aquesta no era un requisit principal en un principi, per tant, el pes principal del projecte recau sobretot sobre els dos primers objectius.

Per al desenvolupament s'ha utilitzat una metodologia

Agile de desenvolupament, s'han dividit els objectius en petites tasques i aquestes s'han realitzat mitjançant l'ús de *sprints*. Aquesta metodologia s'ha dut terme durant aproximadament 6 mesos, i s'ha dividit en 14 tasques de desenvolupament, és a dir, estimadament unes dues setmanes per *sprint*. Aquesta metodologia ha ajudat a desgranar els objectius i per tant tenir un major control i seguiment del desenvolupament.

3.1 Simulador de Trànsit

Els objectius d'aquest projecte estan fortament vinculats al trànsit, pel que es requereix d'una eina que pugui simular trànsit en una configuració de xarxes viàries. Aquesta eina és necessària degut a que el seu desenvolupament seria altament complex i temporalment inviable, és per això que s'ha optat per la recerca d'una eina que ens proporcioni les següents funcionalitats mínimes:

- Simulació de trànsit dins d'una xarxa
- Manipulació de la xarxa sobre la que es vol realitzar simulacions
- Interacció del simulador amb el llenguatge de programació utilitzat en temps real de simulació

Després de realitzar una investigació per a trobar un simulador que pogués complir com a mínim aquestes tres funcionalitats, es va trobar el simulador SUMO (Simulation of Urban Mobility)[6]. Aquest simulador compleix les característiques mínimes que es buscaven, a més, també ens aporta d'altres interessants. El principal avantatge de SUMO és la fàcil instal·lació i ús, independentment del sistema operatiu que s'estigui utilitzant. A més, també existeix la llibreria *traci* [8] escrita en Python la qual permet la comunicació directa i en temps real amb el simulador. Aquest simulador també proporciona una interfície gràfica per a veure en temps real en quin estat està la simulació en curs. SUMO proporciona diverses eines per a la generació de rutes aleatòries sobre una xarxa, la manipulació de les xarxes mitjançant grafs i fitxers .xml. Finalment, també ens permet la obtenció d'un gran conjunt d'estadístiques referents a cada *step*(procés incremental en una simulació SUMO) de la simulació en curs.

També s'han explorat altres opcions de simuladors. El que ha tingut més pes a l'hora de prendre la decisió ha sigut CityFlow [1], un simulador que també complia amb els requisits exposats. No obstant, el simulador SUMO, com ja s'ha explicat, aportava funcionalitats que es van considerar molt positives per a l'estudi.

Un cop exposades les tecnologies que s'han utilitzat en aquest estudi i feta una breu descripció del simulador que ha donat suport per a poder assolir els diferents objectius del proejcte, es començarà amb l'explicació de les funcionalitats desenvolupades per tal de poder assolir els objectius descrits en el present document.

4 Entorn de Desenvolupament

Per a poder provar els diferents algorismes utilitzant el simulador SUMO primer de tot es requereix configurar un entorn en el qual es tinguin una configuració de xarxa de proves, trànsit generat de forma aleatòria i una forma d'insertar les rutes calculades pels diferents algorismes dins de la simulació. La generació de trànsit aleatori s'ha realitzat utilitzant l'eina del simulador SUMO *randomTrips* [9]. Aquesta eina ens permet generar rutes aleatòries donada una configuració de xarxa SUMO. Un cop s'han generat les rutes només caldrà insertar-les utilitzant la llibreria *traci*. Aquesta última llibreria permet tenir control sobre el simulador, des de la seva inicialització amb o sense interfície gràfica, fins a l'afegit de rutes o l'obtenció de mètriques referents a la simulació en curs.

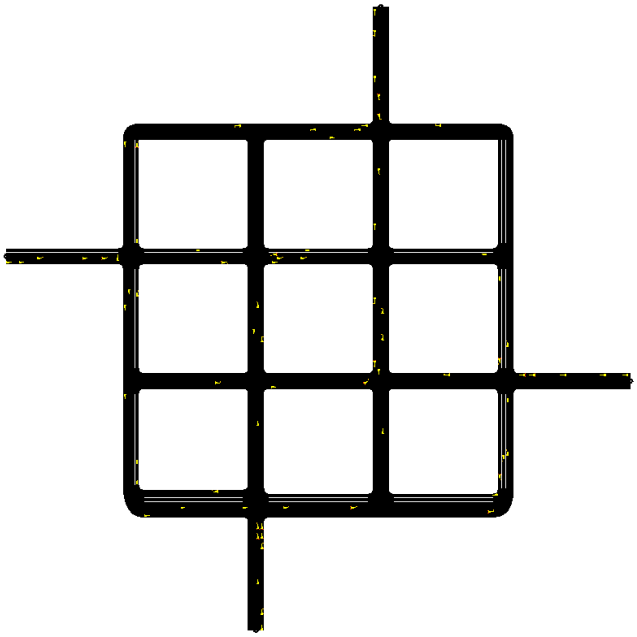

Fig. 1: Xarxa de prova amb trànsit visualitzat per SUMO GUI(Graphical User Interface)

A la figura 1 es pot veure trànsit generat de forma aleatòria amb la eina *randomTrips*, la visualització per altra banda és la de l'entorn d'interfície gràfica SUMO GUI(Graphical User Interface). La configuració que es veu a la figura 1 és una de les que s'ha utilitzat per al desenvolupament. Es tracta d'una xarxa senzilla que permet realitzar proves de les diferents funcionalitats desenvolupades en el projecte.

Finalment per concloure la secció s'han de comentar les consideracions que s'han tingut en compte sobre els escenaris de trànsit. En primera instància, cal recalcar la complexitat d'un fenomen com el trànsit, el trànsit és molt complex no només per la seva aleatorietat, sinó que també per la gran quantitat de variables que conté. Per exemple la gran quantitat de diferents tipus de vehicle i de la seva funció, per exemple, actuarà de diferent forma un autobús a un turisme d'ús individual. Per altra banda també té una infinitat de variables que fan referència a l'estructura de la xarxa, no és exactament el mateix que en una intersecció la prioritat la marqui un semàfor o altres senyals, com per exemple senyals de *stop*. És per aquest conjunt abundant de variables que s'ha optat per a generar un tipus de trànsit simplificat, és a dir, que només hi hagi un tipus de vehicle i que les intesseccions es regeixin per prioritats, sense semàfors ni altres tipus de senyals. Amb tot això, tot i que la simulació s'allunyi de la realitat es redueix molt la complexitat del problema a tractar i el fa viable. Encara que s'utilitzi un model

de trànsit simplificat el simulador SUMO permet afegir un component *alpha* que afegeix aleatorietat al comportament dels vehicles.

Per a poder treballar amb xarxes s'ha decidit per mitjà de la seva representació amb grafs dirigits, això facilita la seva manipulació i aplicació d'algorismes sobre les xarxes. Les arestes del graf representaran els carrers de la xarxa i els nodes les interseccions entre aquests. Les arestes guardaran valors referents al tipus de carrer, per exemple la velocitat màxima o el número de carrils, en canvi, al utilitzar únicament prioritats els nodes només guardaran informació referent a les arestes que connecten. D'aquesta secció en endavant s'utilitzarà també la terminologia dels grafs per a referir-se a xarxes de trànsit. Explicat com serà l'entorn sobre el que s'aplicaran les funcionalitats desenvolupades es pot començar a parlar sobre aquestes.

5 ANÀLISI D'ALGORISMES

Un dels objectius que té aquest treball és l'anàlisi de diferents algorismes per a la generació de rutes en una simulació SUMO en curs, per això que s'han desenvolupat un conjunt d'algorismes per tal de poder realitzar el seu anàlisi i comparació dintre d'una simulació de trànsit generada per SUMO.

Per al càlcul de rutes òptimes amb un únic destí s'ha desenvolupat l'algorisme Dijkstra [3]. Aquest calcula el camí òptim entre dos nodes, per aquest motiu s'ha buscat complexificar el problema. S'ha plantejat que les rutes hagin de tenir parades abans d'arribar el destí, és a dir, calcular el camí òptim (més curt) donats un node d'inici, un node final, i un conjunt de nodes pels quals s'ha de passar abans d'arribar al node final. Aquest problema es coneix com el problema del viatjant[10] i possiblement es tracti d'un dels problemes computacionals més complexes, donat que a mesura que es va augmentant el número de visites que es volen realitzar, la complexitat del problema creix de la forma que es mostra a l'equació 1:

$$O(\frac{(n-1)!}{2}) \tag{1}$$

L'equació 1 equival al número de rutes possibles sent n el número de visites a realitzar.

TAULA 1: CREIXEMENT PROBLEMA VIATJANT

Núm. Visites	Núm. Possibles Rutes
5	12
10	181.440
20	Aprox. 100.000 Bilions

Com es pot veure a la taula 1, el creixement de l'espai de solucions segons el número de visites a realitzar creix de forma exponencial. No obstant, no tots algorismes exploren totes les possibles combinacions de camins. Alguns algorismes no contemplen tot l'espai de solucions, és a dir, que descarten possibles rutes a mesura que l'algorisme es va executant, és per això que la complexitat del problema es redueix a nivells pràctics. D'altres utilitzen estratègies més simples i òptimes que apropen a una solució òptima, però sense garantir-la. A continuació s'explicaran els algorismes que s'han desenvolupat per a l'estudi.

5.1 Dijkstra

L'algorisme Dijkstra[3] és un algorisme principalment utilitzat per a trobar el camí més curt o òptim des d'un node inicial fins a tots els altres nodes d'un graf. Això es duu a terme tenint en compte els pesos assignats a les arestes que uneixen els nodes del graf, els quals no poden ser negatius. En el cas del problema que s'està descrivint els pesos de les arestes equivaldran a la distància euclidiana entre les posicions dels dos nodes als que connecta l'aresta.

L'algorisme comença assignant una distància infinita a tots els nodes a excepció del node inicial, el qual tindrà una distància inicial assignada de zero. El graf s'explora utilitzant una estructura de dades equivalent a una cua de prioritat, d'aquesta manera es seleccionen els nodes amb la distància més curta. A mesura que s'explora el graf es van actualitzant les distàncies als demés nodes fins que tots han sigut visitats.

Aquest algorisme assegura obtenir els camins òptims d'un node a cada node del graf, a l'equació 2 es mostra la complexitat de l'algorisme.

$$O((V+E)*Log(V)) \tag{2}$$

On V Representa el número de vèrtex al graf i E el número d'arestes. La complexitat és polinòmica i per tant molt eficient, d'aquesta manera, tot i que s'augmenti el tamany del graf el temps d'execució creixi de forma logarítmica.

Com s'ha comentat anteriorment al document, es vol buscar el camí òptim donat un conjunt de visites, i com s'acaba d'explicar l'algorisme Dijkstra funciona molt bé en trobar la ruta òptima entre dos nodes, però no funciona si es volen incorporar visites a la ruta. És per això que aquest algorisme no s'utilitzarà directament per a trobar les solucions al problema, sinó que actuarà com un element de suport per als algorismes que si que buscaran trobar rutes òptimes amb visites. Donades un conjunt de visites s'ha de trobar la combinació de camins entre nodes que sigui més òptim, per tant, la part de càlcul de camins entre nodes recaurà sobre l'algorisme Dijkstra.

5.2 Greedy

S'ha desenvolupat un algorisme del tipus Greedy [13], aquest algorisme és més senzill que els demés degut a que es va plantejar com a un bon punt de partida per a l'anàlisi. El problema de l'algorisme Greedy en contraposició del que es vol aconseguir és que no assegura trobar sempre una combinació de parades que sigui òptima. Tot i això, un dels punts més forts d'aquest algorisme és la seva baixa complexitat, aquest fet el converteix en una molt bona opció en situacions en les que no és estrictament necessari trobar una solució òptima, i per tant, es considera un bon resultat trobar simplement una bona solució que s'aproximi a l'òptima. Aquest tipus d'algorisme es tracta d'un algorisme d'aproximació al resultat ja que tot i que no troba la solució òptima pot arribar a apropar-se.

La implementació de l'algorisme es basa principalment en els següents passos:

1. Aplicar l'algorisme Dijkstra al node inicial
2. Afegir al camí final la visita amb la distància més petita

3. Repetir el primer pas fins que no quedin visites
4. Agafar el camí Dijkstra de l'última visita fins al node final

La clau de la baixa complexitat d'aquest tipus d'algorisme és la de no fer passes enrere en la construcció de solució que s'està buscant, és a dir, un cop afegida una parada a la solució final, aquesta no es reemplaçarà per provar si amb un altra possible parada el camí final podria arribar a ser més curt i en definitiva, més òptim. La complexitat del algorisme Greedy utilitzat té la complexitat representada per l'equació 3, On N és el número de visites a realitzar.

$$O(N((V+E)*Log(V))) \tag{3}$$

Amb la consecució dels primers resultats utilitzant aquest algorisme es va començar amb el següent tipus d'algorisme, el qual si que proporciona la solució òptima al problema del viatjant.

5.3 A*

Tot i que per a trobar la solució òptima al problema es podria haver desenvolupat un algorisme del tipus Backtracking s'ha optat per no realitzar aquesta implementació. Aquesta desició s'ha pres donat que aquest tipus d'algorisme són molt ineficients ja que només exploren tot l'espai de possibles combinacions de visites, finalment es queden amb la solució que tingui el recorregut més curt. Aquest estudi no es centra en l'ànalisi de rendiment dels diferents algorismes, per aquest motiu s'ha implementat l'algorisme A*

Els algorismes A* [2], a diferència dels algorismes Backtracking busquen reduir l'espai de solucions, és a dir, són capaços de donar prioritat a unes solucions per davant d'altres, també poden descartar solucions abans d'arribar a explorar-les i finalment poden identificar quan s'ha arribat a una solució òptima sense la necessitat d'explorar tot el conjunt de possibles solucions. Tot i que la complexitat teòrica dels algorismes A* sigui la mateixa que la dels algorismes Backtracking, a la pràctica el rendiment de l'A* millora considerablement el rendiment que pot donar un algorisme del tipus Backtracking. És per aquests fets que s'ha optat per desenvolupar des d'un inici aquest tipus d'algorismes.

Els algorismes *A** prioritzen explorar el conjunt de solucions, aquesta priorització es fa mitjançant heurístiques que determinen quina solució a explorar és més prometedora per arribar a una solució òptima. Ja que les heurístiques determinaran la forma en la que s'explora el conjunt de solucions ha de tenir certes restriccions. Una heurística ha de subestimar el valor de la mètrica sobre la que vol trobar la solució òptima[14], s'ha de subestimar però apropant-se tot el possible al valor real, si no es subestima la mètrica a optimitzar es poden trobar falses solucions òptimes. La utilització d'heurístiques fa que es puguin desenvolupar diferents versions d'aquest algorisme adaptant l'heurística que s'utilitza. A continuació s'explicaran les diferents versions de l'algorisme *A** que s'han desenvolupat per a trobar rutes òptimes tenint en compte el trànsit de la simulació SUMO.

Per a canviar els resultats que es volen obtenir, les heurístiques han d'afectar també a la forma en que es calculen els camins de punt a punt, és a dir, s'ha de modificar l'algorisme Dijkstra per a que accepti costos extres a l'hora de calcular els pesos de les arestes. Aquest cost seran mètriques de la simulació, per exemple el temps d'espera, consum, etc.

5.3.1 Camí més Curt

La primera heurística que s'ha implementat és la més bàsica i simple, aquesta funció heurística és la suma del cost que porta la solució actual més la distància del camí *Dijkstra* fins a la visita final. Aquesta heurística ja permet trobar la solució òptima d'una forma molt més ràpida que amb un algorisme del tipus Backtracking. El problema és que el trànsit de la xarxa no té cap paper i per tant, no es té en compte com està el trànsit a la resta de la xarxa en el moment de calcular la ruta. És per això que les pròximes heurístiques si que tenen en compte el trànsit.

5.3.2 Velocitat Mitjana

La heurística basada en velocitat mitjana ja té en compte el trànsit que hi ha a la xarxa. Això s'aconseguix per mitjà de la obtenció de mètriques referents al trànsit de la simulació en curs, en aquest cas la velocitat mitjana en cada carrer. Tot i que s'utilitzi la velocitat mitjana per obtenir una estimació heurística, no es deixa de tenir en compte la distància mínima, és a dir, si només es té en compte la velocitat mitjana s'obtindran rutes on els carrers per on el trànsit sigui fluït, tenint en compte això, cap la possibilitat de que la ruta obtinguda sigui molt llarga, per tant, aquesta heurística s'ha calculat mitjançant una combinació del camí més curt i la velocitat mitjana. El calcul de l'heurística és representat per l'equació 4:

$$h = cost + dDijkstra + (\alpha * \sum_{i=1}^{n} vMitjana_i) \tag{4}$$

En l'equació 4 la velocitat mitjana s'afegeix a l'heurística sumant totes les velocitats mitjanes dels carrers que conformen per una part, la solució actual i l'estimació de carrers per arribar al node final. Tot això multiplicat per el factor α, el qual permet regular el pes que té la velocitat mitjana en l'obtenció de la ruta òptima.

Aquesta heurística dona pes a la fluïdesa del trànsit, i per tant dona prioritat a rutes per les quals es circula amb més velocitat, això pot desencadenar en resultats de rutes no necessàriament més curtes en quant a distància però en canvi, però que si ho poden ser en quant a temps de recorregut.

5.3.3 Temps d'espera

Aquesta heurística és molt semblant a l'anterior ja que és càlcula de la mateixa manera però en aquest cas utilitzant una altra mètrica de la simulació. Aquesta mètrica és el temps d'espera dels cotxes en cada carrer. Les simulacions SUMO basen el temps en *steps*, unitats de temps que representen una quantitat de segons, el temps d'espera d'un vehicle representa la quantitat de *steps* durant els que aquest vehicle ha tingut una velocitat de $0m/s$, és per això que aquesta heurística premia a les rutes on no hi hagi retenció. A l'equació 5 es mostra la fórmula per a calcular l'heurística:

$$h = cost + dDijkstra + (\alpha * \sum_{i=1}^{n} tEspera_i) \quad (5)$$

5.3.4 Consum de Combustible

S'ha desenvolupat una heurística que té en compte el consum de combustible a cada part de la xarxa de trànsit. L'objectiu de l'heurística, com el seu propi nom indica prioritza les rutes que tenen menys consum, aquest fet pot estar relacionat amb la velocitat mitja i el temps d'espera ja que són mètriques que poden afectar al consum, al final, si un carrer està ple de vehicles parats el consum mig d'aquest carrer serà molt baix. Per altra banda també pot estar relacionat al tipus de vehicles que circulen per cada punt de la xarxa, aquest factor, com s'ha explicat anteriorment s'ha obviat en la simplificació del model de trànsit. A la equació 6 es mostra la fórmula que calcula l'heurística esmentada:

$$h = cost + dDijkstra + (\alpha * \sum_{i=1}^{n} consum_i) \quad (6)$$

5.3.5 Contaminació

Per últim s'ha desenvolupat una heurística enfocada en les diferents emissions dels vehicles. L'objectiu d'aquesta heurística és la de buscar de reduir les contaminacions en el conjunt de la simulació, per tant, és la única heurística que no busca un resultat individual. Les altres heurístiques tenen també una finalitat global, com per exemple la de no afavorir a les retencions, però si que tenen un benefici individual a diferència d'aquesta.

Aquesta heurística té en compte 3 tipus d'emissions, CO2 (diòxid de carboni), NOX[11] i PMX[12] i és calcula amb l'equació 7 forma:

$$h = cost + dDijkstra + (\alpha \sum_{i=1}^{n} \beta * Co2_i + \gamma * Pmx_i + \omega * Nox_i) \quad (7)$$

6 Optimitzador de Xarxes

Un dels objectius d'aquest document és desenvolupar una eina que permeti optimitzar una xarxa de trànsit determinada. El concepte optimització fa referència al procés per a aconseguir la millora d'un rendiment o resultat, i com ja s'ha explicat anteriorment en aquest document, el trànsit i les xarxes de trànsit són sistemes molt complexes, això fa que hi hagi incomptables possibles canvis que podrien o no portar a una xarxa òptima. En un principi es va analitzar quin cost podria tenir l'obtenció d'una solució òptima al problema. Es va arribar a la conclusió que s'adequaria més al problema buscar solucions properes a la òptima. Un cop es va arribar a aquesta conclusió es van pensar maneres de poder millorar l'eficiència d'una xarxa de trànsit de forma progressiva i que no fos necessari esperar a trobar una possible solució òptima al problema. Aquestes millores es medirien[7] utilitzant mètriques referents a les simulacions i serien les que determinarien si una xarxa és millor que una altra. Un cop analitzat el problema es va plantejar l'utilització d'algorismes genètics. Un tipus d'algorisme que donades les seves característiques s'adapta molt bé al problema plantejat.

6.1 Algorismes Genètics

Els algorismes genètics o evolutius són una tècnica d'optimització basada en l'evolució biològica. Aquest tipus d'algorismes busquen imitar el procés de selecció natural per a resoldre problemes complexes d'optimització i per tant, trobar solucions òptimes o sub-òptimes. Un algorisme genètic es pot desglossar en les següents etapes, les quals es repetiran fins arribar a una condició de parada o convergència.

1. **Inicialització**: Es crea una població inicial d'individus de forma aleatòria o utilitzant un coneixement inicial.
2. **Evaluació**: Cada individu de la població és avaluat utilitzant una funció d'aptitud, aquesta funció mesura quant bó és un determinat individu resolent el problema plantejat.
3. **Selecció**: En aquesta etapa els individus de la població ja han sigut avaluats i per tant, es seleccionen el o els que maximitzen la funció d'aptitud per a "reproduirse".
4. **Reproducció**: Es crea una nova població a partir del o dels individus seleccionats a l'etapa anterior afegint canvis aleatoris en cada nou individu.
5. **Reemplaç**: Es substitueix la població antiga per la nova població creada en l'anterior etapa.

Aquestes etapes s'han de traslladar al problema plantejat en l'estudi. Primer de tot, es comença amb una xarxa inicial escollida, aquesta estarà representada mitjançant un graf que s'utilitzarà per a crear la població inicial. La població inicial es crearà copiant el graf n cops i aplicant sobre el mateix modificacions aleatòries, aquestes modificacions seran senzilles i es basaran principalment en eliminar i/o afegir arestes i nodes al graf a més de la possibilitat de modificar el nombre de carrers que conformen les carreteres (arestes). Un cop creada la població inicial s'executaran simulacions de forma paral·lela i amb trànsit aleatori sobre tota la població, quan les simulacions hagin acabat es reculliran les mètriques de cada una i es seleccionarà la millor, un exemple podria ser utilitzar el mínim temps d'espera. Un cop s'ha seleccionat la millor simulació es tornarà a repetir el procés creant una nova població amb aquesta millor simulació. El procés que agrupa els 5 passos de l'algorisme se l'anomena època. Aquest procés estarà parametritzat donat que hi han diverses variables a tenir en compte i que podran modificar els resultats, per exemple variables com la quantitat de modificacions, *steps* per simulació, tamany de la població, número d'èpoques a realitzar entre altres.

Amb aquest procés s'espera obtenir una millora en les mètriques sobre les quals s'està realitzant l'optimització, els resultats però s'analitzaran a la secció 8.

Cal afegir que aquest procés, donat que s'han de realitzar simulacions independents sobre cada població, es paral·lelitzarà, és a dir, les creacions de configuracions i les

simulacions s'executaran de forma paral·lela per a optimitzar l'execució de l'algorisme. Aquest últim fet és important ja que aquest tipus de problemes, al anar-se apropant a una possible solució òptima, quantes més èpoques s'executin, millors haurien de ser els resultats, és per això que la paral·lelització de l'algorisme permetrà buscar uns millors resultats.

7 Interfície Gràfica

Un dels objectius proposats és que les funcionalitats desenvolupades puguin ser accessibles a través d'una interfície gràfica. Aquesta interfície gràfica està feta amb tecnologies web i en funció de l'informació rebuda es mostrarà el contingut. Per a poder comunicar les funcionalitats desenvolupades amb la interfície s'ha desenvolupat un servidor *Rest API* que implementarà un conjunt de *endpoints* els quals seran cridats per la interfície segons l'interacció de l'usuari. La *Rest API* s'ha implementat utilitzant la llibreria de Python FastAPI, per altra banda, per la visualització de la interfície s'han utilitzat les llibreries jQuery[15] i Bootsrap[16].

8 Resultats

En aquesta secció s'exposaran els resultats obtinguts un cop finalitzat el desenvolupament per a aconseguir els objectius proposats. Cal destacar que per a mostrar els resultats de forma visual s'utilitzarà la interfície gràfica desenvolupada.

8.1 Anàlisi d'Algorismes

En la secció 5 s'han desenvolupat un conjunt d'algorismes per a la generació de rutes entre dos punts d'una xarxa amb visites o parades per les que s'ha de passar abans d'arribar al destí. A continuació es mostra una ruta sense visites obtinguda amb l'algorisme Dijkstra:

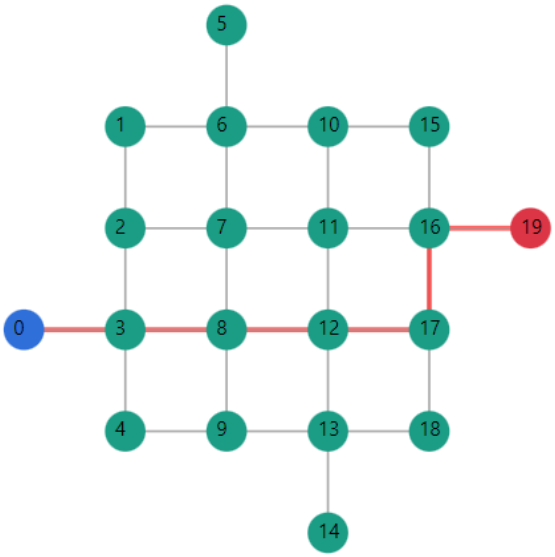

Fig. 2: Ruta Dijkstra

Com es pot veure, les línies vermelles indiquen la ruta des del node 0 fins al node 19.

8.1.1 Greedy vs A*

Com s'ha comentat a la secció 5, els algorismes del tipus Greedy tot i que són molt eficients i ràpids no són capaços de trobar una solució òptima, és per això que s'ha volgut provar aquest fet utilitzant un tipus de xarxa i ruta a calcular específica.

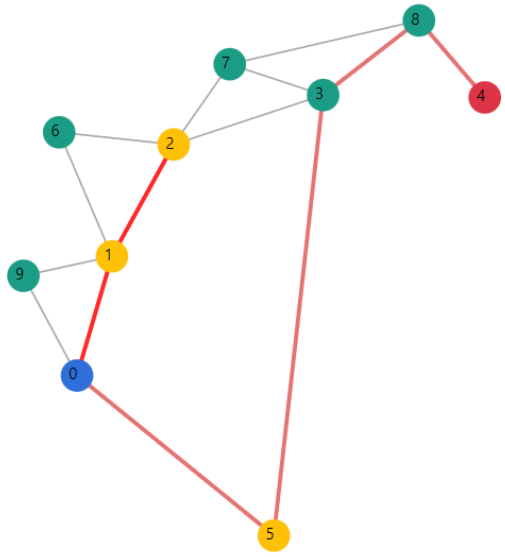

Fig. 3: Ruta no òptima calculada per l'algorisme Greedy

En la figura 3 s'ha realitzat el càlcul d'una ruta utilitzant l'algorisme Greedy desenvolupat. Com es pot veure, la ruta comença al node 0, finalitza al node 4 i ha de passar pels nodes 1, 2 i 5 abans d'arribar al node final. En la ruta calculada primer es passa pels nodes 1 i 2 per després tornar al node 0 i finalitzar la ruta passant pel node 5. També s'ha calculat la mateixa ruta utilitzant l'algorisme A* desenvolupat per provar si la ruta calculada per l'algorisme Greedy és la òptima.

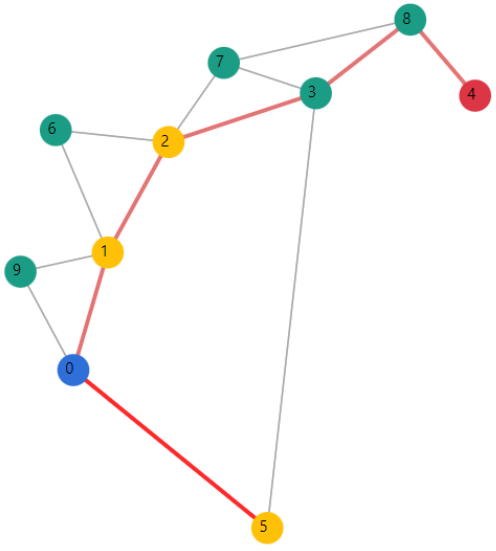

Fig. 4: Ruta òptima calculada per l'algorisme A*

Com es veu a la figura 4, a diferència del que realitza l'algorisme Greedy, en aquest cas es visita primer el node 5 i després afegeix les visites 1 i 2 per arribar després fins a destí de la ruta, això fa que el total de distància recorreguda sigui més curta i per tant, sigui la solució òptima al problema. Tot i que es trobi una ruta òptima en un temps baix s'ha de comentar que anteriorment en aquest document l'algorisme té una alta complexitat comparat amb el Greedy, això produeix que en un cas com l'actual, en el que només hi han 3 visites la ruta òptima es trobi ràpid, no obstant, només que s'afegeixin unes poques visites més en un graf més gran el temps per a trobar la ruta òptima creixerà molt ràpid.

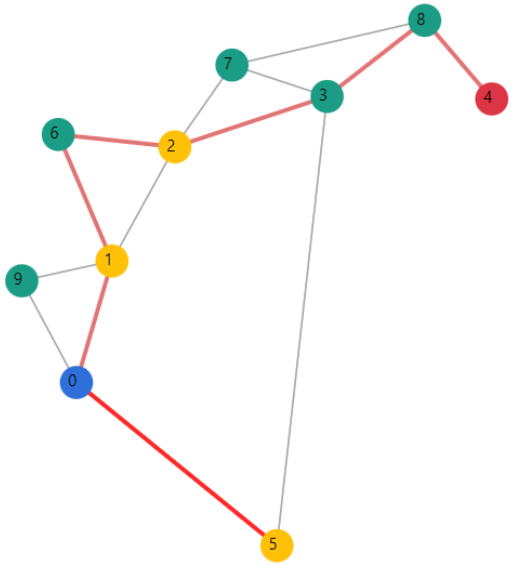

Fig. 6: Ruta òptima tenint en compte el trànsit de la xarxa

8.1.2 A* amb Trànsit

Un dels motius per els qual s'ha desenvolupat aquest algorisme és per poder provar-lo tenint en compte el trànsit d'una simulació. Això s'aconsegueix mitjançant l'ús d'heurístiques que modifiquin l'estimació d'una ruta en funció de mètriques relacionades amb el trànsit en el moment que es vol calcular la ruta, és per això que aquestes heurístiques només es poden utilitzar en el moment d'execució de la simulació. En la figura 5 es pot veure la xarxa que s'ha utilitzat a les figures 3 i 4 durant una simulació.

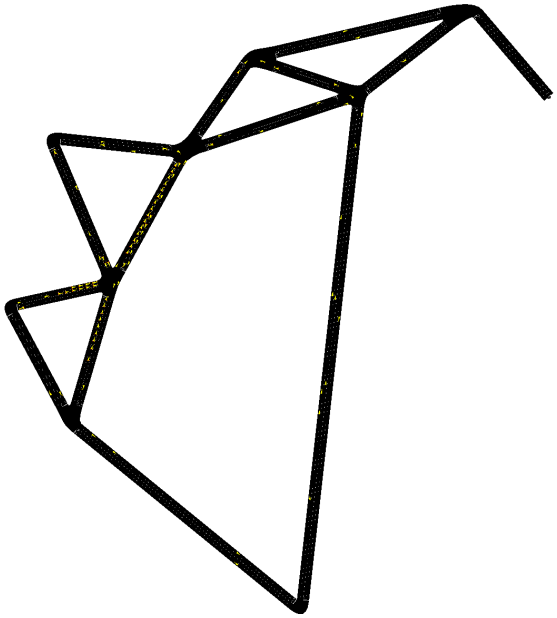

Fig. 5: Xarxa col·lapsada pel trànsit

Com es veu a la figura un dels carrers està saturat per un gran conjunt de cotxes. Al aplicar l'algorisme A* per a buscar la ruta òptima que s'ha obtingut a la figura 4 s'aconsegueix el següent resultat.

A la figura 6 es pot veure com s'aplica l'heurística que té en compte el temps d'espera de cada carrer a l'hora de calcular les rutes. Com es pot apreciar, la ruta òptima ha sigut lleument modificada per evitar la retenció entre els nodes 1 i 2, l'algorisme ho resol passant per el node 6 evitant el trànsit.

Les altres heurístiques explicades en aquest document totes tenen un comportament similar, és a dir, adapten la ruta més curta en funció de les diferents mètriques en cada aresta o carrer que conforma aquest camí.

Cal tenir en compte que tot i que les rutes calculades amb aquestes heurístiques contemplin el trànsit de la xarxa, aquestes ho fan en el moment en el que es calcula i per tant, no assegura que durant el temps en el que el vehicle realitza la ruta el trànsit no hagi canviat i la ruta òptima ja no sigui la inicialment calculada. Una possible millora a tenir en compte seria el calcul de la ruta cada cop que el vehicle que la recorre arribi a una intersecció o node, d'aquesta forma s'aconseguiria un calcul de rutes més proper al temps real.

8.2 Optimitzador de Xarxes

En aquest apartat es comentaran els resultats obtinguts amb l'optimitzador de xarxes desenvolupat. Primer de tot s'ha volgut provar l'optimitzador amb una xarxa senzilla.

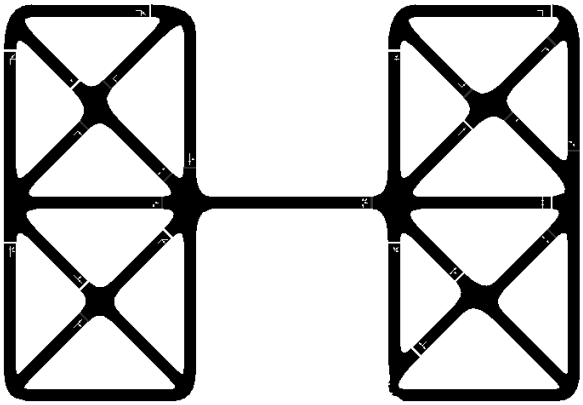

Fig. 7: Xarxa simple a optimitzar

A la figura 7 es pot veure una xarxa senzilla, aquesta xarxa té dues zones diferenciades on hi ha una acumulació de carrers, aquestes dues zones estan únicament conectades per un carrer amb dos carrils, un per cada sentit. Donat que només existeix aquesta connexió entre les dues zones el trànsit té retencions en aquesta única connexió. S'ha provat a optimitzar la xarxa per a provar si l'optimitzador pot millorar la mètrica temps d'espera. A la figura 8 es poden veure els resultats obtinguts.

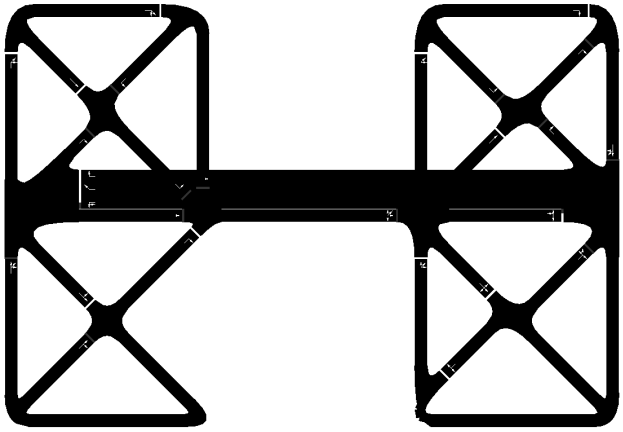

Fig. 8: Xarxa simple optimitzada per l'optimitzador de xarxes

Després de 3 èpoques amb una població de 50 individus, es pot veure que la solució més òptima que ha trobat l'optimitzador és la d'afegir més carrils en la connexió entre les dues zones i treure un carrer de la zona esquerra. En aquesta optimització s'han realitzat 3 èpoques amb una població de 50 individus per època, també s'han limitat els canvis aleatoris a l'afegit o eliminació d'un carrer.

Tot i que visualment el resultat que s'ha obtingut té força sentit, s'ha comprovat que realment s'han reduït les retencions de la xarxa, i per tant el temps d'espera total. En la primera simulació s'obtenia un temps d'espera acumulat de 232453 $steps$, un cop finalitzades les tres èpoques el temps d'espera acumulat ha baixat a 21500 $steps$, és a dir, s'ha millorat en més d'un 90% la mètrica en la simulació.

Aquesta prova, tot i que ajuda a saber si l'optimitzador funciona correctament s'ha volgut realitzar una altra prova per a dotar de més solidesa els resultats obtinguts. S'ha realitzat la optimització de 25 xarxes creades de forma aleatòria, s'han simulat per a veure les seves mètriques inicials i s'han optimitzat una per una per comprovar si milloren el temps d'espera. Per cada simulació s'han executat 5 èpoques amb una població de 24 individus, s'han escollit aquests valors donada l'alt cost computacional de la prova. Amb aquesta prova es busca reduir la variància i veure realment si la mitjana de la mètrica optimitzada de totes les xarxes es menor al final de totes les optimitzacions. A continuació es pot veure una taula amb els resultats obtinguts de les optimitzacions.

TAULA 2: RESULTATS OPTIMITZADOR DE XARXES

Època	**Temps Espera Mig**	**Mitjana de Millora**
Inicial	73131,84 *steps*	-
1	13158,68 *steps*	82,00%
2	10739,56 *steps*	18,38%
3	9789,00 *steps*	8,85%
4	8845,64 *steps*	9,63%
5	7821,40 *steps*	11,57%
Total	-	89,30%

Com es pot veure a la taula 2 s'ha aconseguit millorar molt la mètrica temps d'espera, tot i que cada cop que s'executa una època la millora va decreixent, la millora a l'última època és d'un 89,30%, una xifra molt significativa. Uns resultats com aquests, tot i que són força bons s'han de posar en el contexte sobre el que es treballa, s'està donant una llibertat absoluta a modificar qualsevol carrer, cosa que allunya el problema d'un de real, a demés, com s'ha comentat en aquest document, s'estan realitzant simulacions amb una versió de trànsit molt simplificada, fet que faci que no s'estiguin tenint en compte molts factors que afectarien a les simulacions, no obstant, dintre de l'entorn sobre el que s'ha treballat, s'ha aconseguit l'objectiu principal, millorar el rendiment de les xarxes de trànsit. Hi ha altres factors que poden haver limitat una millora més gran dels resultats. Un d'ells podria ser possibles mínims locals, és a dir, el fet de que cada nou individu pugui modificar tan sols un carrer pot arribar a provocar que el canvi no sigui suficient en alguns casos, això produiria l'estancament de les optimitzacions sense la possibilitat de fer els canvis suficients per a poder canviar la direcció cap a una direcció on l'optimització pogués ser millor. De totes formes, aquest últim fet no és un fet que s'hagi comprovat però que estaria sobre la taula per a futures millores.

9 CONCLUSIONS

En el present document s'han repassat diferents formes d'aplicar algorismes a les xarxes de trànsit, tot això amb motiu de poder crear una eina de suport als operaris d'aquestes xarxes. S'han marcat tres objectius principals els quals s'han pogut assolir. Primer de tot s'ha pogut realitzar un anàlisi de diferents tipus d'algorismes per al càlcul de rutes, i s'han realitzat versions d'aquests algorismes que també han tingut en compte el trànsit a l'hora de calcular les rutes resultants, aquests algorismes han tingut resultats satisfactoris i ajustats als resultats esperats. També s'ha aconseguit el segon objectiu, s'ha desenvolupat un algorisme

genètic capaç d'optimitzar xarxes amb bons resultats. Finalment s'han pogut recollir la majoria d'aquestes funcionalitats desenvolupades en una interfície gràfica, la qual s'ha utilitzat per a donar suport visual al document.

Assolits els objectius es vol remarcar que la principal conclusió que s'ha obtingut d'aquest treball és que el trànsit i les xarxes on és dona són sistemes extremadament complexes en els quals l'aleatorietati milers de factors tenen un gran pes, es per això que s'ha hagut de simplificar l'entorn per a representar aquests elements fent assolibles els objectius amb els recursos que s'han tingut. Per finalitzar recalcar que tot i la complexitat del problema, s'han obtingut i detallat resultats que fan pensar que s'ha pres una direcció correcta a l'hora d'enfocar la resolució dels problemes i els objectius fixats.

Referències

[1] CityFlow. (4 Desembre de 2020). Obtingut de https://cityflow.project.github.io/

[2] Games, R. B. (Febrer de 20 de 2023). Introduction to A*. Obtingut de https://tinyurl.com/2y6pm4w6

[3] Javaid, A. (Gener de 2013). Understanding Dijkstra Algorithm. Obtingut de https://tinyurl.com/34wrsu8n

[4] Mitma Mayta, W. Z. (20 de Novembre de 2019). ESTUDIO DE TRAFICO Y OPTIMIZACIÓN DE LA RED VIAL QUE COMPRENDE EL JR. LIBERTAD, JR. OLÍMPICO Y AV. GANDOLINI DE LA CIUDAD DE LIRCAY - ANGARAES. Obtingut de https://repositorio.unh.edu.pe/items/835b9c5c-194f-40b6-abdc-31e46680a296

[5] SUMO. (09 de Març de 2023). SUMO-GUI. Obtingut de https://sumo.dlr.de/docs/sumo-gui.html

[6] SUMO. (09 de Març de 2023). SUMO-GUI. Obtingut de https://eclipse.dev/sumo/

[7] Joaquín Amat Rodrigo. (Febrer, 2019). Optimización con algoritmo genético y Nelder-Mead. Obtingut de https://tinyurl.com/mrxs3kp8

[8] SUMO. (09 de Març de 2023). Documentació llibreria traci. Obtingut de https://sumo.dlr.de/pydoc/traci.html

[9] SUMO. (09 de Marc de 2023). Documentació randomTrips.py. Obtingut de https://sumo.dlr.de/docs/Tools/Trip.html

[10] Merrill M. Flood, (1956) The Traveling-Salesman Problem. Operations Research 4(1):61-75. https://doi.org/10.1287/opre.4.1.61

[11] Ministerio para la Transición Ecológica y el Reto Demogràfico, Óxidos de Nitrógeno. https://www.miteco.gob.es/es/calidad-y-evaluacion-ambiental/temas/atmosfera-y-calidad-del-aire/calidad-del-aire/salud/oxidos-nitrogeno.aspx

[12] Agencia de Protección Ambiental de Estados Unidos (EPA). (26 juny, 2023). Conceptos básicos sobre el material particulado. https://espanol.epa.gov/espanol/conceptos-basicos-sobre-el-material-particulado-pm-por-sus-siglas-en-ingles

[13] Greedy Algorithms. Brilliant.org. Retrieved 19:16, July 1, 2023, from https://brilliant.org/wiki/greedy-algorithm/

[14] Akbar Karimi, (9 de Novembre de 2022). What Is a Heuritic Function?. Obtingut de https://www.baeldung.com/cs/heuristics#: :text=2.1.-,Definition,the%20solution%20is%20too%20long.

[15] OpenJS Foundation, (2023). jQuery v3.7.0. Obtingut de https://jquery.com/

[16] Bootstrap team, (2023). Bootsrtap v5.3.0. Obtingut de https://getbootstrap.com/

[17] AntsRoute, (18 de Febrer de 2022). Google Maps: ¿Podemos hablar de optimización de rutas?. Obgingut de https://antsroute.com/es/solucion/google-maps-podemos-hablar-de-optimizacion-de-rutas/

[18] Q.J. Wang, Using genetic algorithms to optimise model parameters, Environmental Modelling & Software, Volume 12, Issue 1, 1997, Pages 27-34, ISSN 1364-8152, https://doi.org/10.1016/S1364-8152(96)00030-8. (https://tinyurl.com/2y6pm4w6)

[19] Li H, Shi N. Application of Genetic Optimization Algorithm in Financial Portfolio Problem. Comput Intell Neurosci. 2022 Jul 15;2022:5246309. doi: 10.1155/2022/5246309. PMID: 35875786; PMCID: PMC9307338.

[20] José Francisco López, (1 setembre de 2020). Modelo de Markowitz). Obtingut de https://economipedia.com/definiciones/modelo-de-markowitz.html

[21] Lu S, Jiang H, Li C, Hong B, Zhang P, Liu W. Genetic Algorithm for TMS Coil Position Optimization in Stroke Treatment. Front Public Health. 2022 Mar 11;9:794167. doi: 10.3389/fpubh.2021.794167. PMID: 35360667; PMCID: PMC8962518.

[22] Ibiricu, M.A., & Morales, G.. (2009). Transcranial magnetic stimulation. Anales del Sistema Sanitario de Navarra, 32(Supl. 3), 105-113. Recuperado en 02 de julio de 2023, de https://tinyurl.com/ycyfz5yu

[23] Liu K. Analysis of the Conflict between Car Commuter's Route Choice Habitual Behavior and Traffic Information Search Behavior. Sensors (Basel). 2022 Jun 9;22(12):4382. doi: 10.3390/s22124382. PMID: 35746164; PMCID: PMC9231029.

[24] Cao S. An Optimal Round-Trip Route Planning Method for Tourism Based on Improved Genetic Algorithm. Comput Intell Neurosci. 2022 Aug 28;2022:7665874. doi: 10.1155/2022/7665874. PMID: 36072721; PMCID: PMC9441363.

APÈNDIX

1.1 Recursos Adicionals

El codi font utilitzat durant l'estudi es troba disponible en un repositori públic de GitHub (*BessoDigitalXarxaTransit*, https://github.com/carles444/BessoDigitalXarxaTransit).
El repositori conté totes les implementacions detallades dels algorismes utilitzats per a generar els resultats de l'estudi.