Projecte d'Algorísmia

Mínim Conjunt Dominador d'Influència Positiva

Jia Long Ji Qiu

Shuang Long Ji Qiu

Jiabo Wang

You Wu

Q1 Curs 2021/2022

Grup 11

Índex

1. Descripció del problema

Donat un graf connex i no dirigit G = (V, E), que representa una xarxa social (on V representa el conjunt d'actors, i E representa les relacions entre aquests actors), es vol trobar un conjunt de vèrtexs (actors) D que pertany a V, tal que almenys la meitat dels veïns de cadascun dels vèrtexs de G formen part del conjunt D. Aleshores D es defineix com el conjunt dominador d'influència positiva. L'objectiu és optimitzar aquest conjunt D, tal que la seva cardinalitat sigui la mínima possible.

Aquest és un problema àmpliament conegut, i rep el nom de Mínim Conjunt Dominador d'Influència Positiva, amb acrònim MPIDS, i és un problema NP-hard.

Aquest problema d'optimització es pot resoldre mitjançant diferents algorismes, que segons la seva complexitat proporcionarà una solució amb millor qualitat o pitjor.

A la primera part de la resolució del problema, es proporciona tècniques algorísmiques senzilles, com són un algorisme voraç i una cerca local. En canvi, en la segona es pretén sofisticar la resolució mitjançant tècniques més complexes per tal d'obtenir solucions amb més qualitat.

PRIMERA PART

2. Comprovació de PIDS i conjunt minimal

En aquest primer apartat es demana que donat un graf G = (V, E) i un conjunt D ⊆ V, proporcionar un algorisme, tal que comprovi si D és un conjunt dominador d'influència positiva de G i, en cas de ser-ho, comprovar si aquest és minimal.

Definim un vèrtex dominat com un vèrtex el qual té almenys la meitat de les seves adjacències al conjunt dominador D.

Donat un conjunt D d'un graf G, aquest conjunt serà un conjunt dominador d'influència positiva si tots els vèrtexs de G són vèrtexs dominats.

Si un conjunt dominador d'influència positiva és minimal, implica que si traiem un vèrtex del conjunt dominador, aquest deixaria de ser-ho. Per tant, per tal de comprovar si un conjunt és minimal, bastaria amb comprovar que per cada vèrtex del conjunt D, si el traiem d'aquest, alguna de les seves adjacències deixaria de ser un vèrtex dominat.

3. Algorisme voraç determinista

Un primer algorisme per a intentar resoldre el problema és mitjançant l'ús d'un algorisme voraç. La estratègia plantejada és senzilla: ordenem els vèrtexs de manera ascendent segons el nombre d'adjacències que tingui i, per cada vèrtex, s'afegiran la meitat de les seves adjacències al conjunt dominador, prioritzant aquelles adjacències que tinguin un major nombre d'adjacències.

3.1. Descripció de l'algorisme

L'algorisme consisteix en, primer de tot, llegir els nodes (vèrtexs) i les seves corresponents arestes. Tot seguit, mitjançant l'ús d'una cua de prioritat amb *min-heap*, s'emmagatzema els vèrtexs del graf segons el número d'adjacències que té, de manera que es prioritzen els vèrtexs amb menys adjacències. A continuació, per a cada vèrtex *v* del *min-heap* es fa un recorregut sobre el seu veïnatge: aquells veïns que no formin part del conjunt dominador es guardaran en un *max-heap*, ordenat també segons el nombre d'adjacències i, per tant, prioritzant aquells amb més adjacències. Una vegada hi són tots els veïns al *max-heap*, es van afegint els més prioritaris al conjunt dominador fins que el vèrtex *v* tingui la meitat del seu veïnatge formant part d'aquest conjunt.

Cal esmentar que s'ha decidit començar iterant pels vèrtexs amb menor cardinalitat, ja que considerem que els veïnatges petits tenen una probabilitat més alta de ser imprescindibles dins el conjunt dominador. D'aquests veïnatges, si s'ha d'agafar sí o sí la meitat per cada vèrtex, la opció més segura seria seleccionar els de major cardinalitat, perquè el fet de tenir major cardinalitat implica que més probable és que un altre vèrtex també requereixi que estigui al conjunt dominador.

Podem afirmar que aquest algorisme és correcte ja que, per cada vèrtex, es comprova si tenen la meitat dels seus veïns dins el conjunt solució i, de no complir-se aquesta propietat, s'afegeixen els vèrtexs suficients per assolir la meitat. Aquesta comprovació es fa per tots els vèrtexs, per tant, un cop tractat l'últim vèrtex, s'assegura que tots els vèrtexs tenen com a mínim la meitat dels seus veïns al conjunt dominador.

Analitzant el cost de l'algorisme, siguin *n* el número de nodes i *m* el número d'arestes, un primer cost és $O\ (n\ +\ m)$ per llegir el graf i emmagatzemar-lo. Afegir els n vèrtexs ordenats segons el nombre de veïns de cada vèrtex al *min-heap* té un cost $O(n * log(n))$. Tot seguit, es recorre cada vèrtex del *min-heap* i, per cada vèrtex, es construeix *un max-heap* de tots els seus vèrtexs veïns amb cost $O(m * log(m))$, ja que en el pitjor cas, pot ser adjacent a tots els altres nodes del graf. Un cop construït el *max-heap*, es recorre com a molt primera meitat del *max-heap*, cost $O(m/2)\ =\ O(m)$, per tal de seleccionar fins la meitat dels seus veïns amb més adjacències. Per tant, d'aquest bucle tenim un cost $O(n * (m * log(m) + m))$.

La complexitat temporal és, per tant, $O(n\ +\ m)\ +\ O(n * log(n))\ +\ O(n * (m * log(m) + m))\ =\ O(n * (m * log(m)\ +\ m))$.

3.2. Experimentació

Executant l'algorisme voraç per cada instància, els resultats obtinguts han sigut els següents:

Network	Valor	Temps (segons)
graph_football	75	0.00
graph_jazz	85	0.00
ego-facebook	1984	0.01
graph_actors_dat	3278	0.02
graph_CA-AstroPh	7120	0.04
graph_CA-CondMat	9941	0.02
graph_CA-HepPh	4914	0.02
socfb-Brandeis99	1577	0.03
socfb-Mich67	1502	0.01
soc-gplus	8466	0.01

Tots els conjunts dominadors obtinguts han sigut, efectivament, conjunts dominadors d'influència positiva. No obstant, aquest algorisme voraç no garanteix que siguin minimal. Sobre com minimitzar aquest conjunt, entrarem en més detalls en el següent apartat, on parlarem sobre el plantejament de la cerca local i la estratègia de minimització.

4. Cerca local

La cerca local es una de les diverses tècniques que es poden utilitzar per resoldre problemes d'optimització. En el nostre cas, resoldre un problema que es NP-hard. En general, utilitzarem cerca local si volem minimitzar/maximitzar un determinat criteri.

Un algoritme de cerca local es tracta de buscar la millor solució dins de l'espai de cerca (optima local) que podem explorar-ho mitjançant l'ús operadors partint d'una solució inicial.

Aquests operadors seran funcions que aplicarem sobre la solució inicial per tal de intentar millorar-ho. A més, disposem d'una funció que avalua l'estat resultant un cop aplicat els operadors segons un criteri determinat per veure la seva qualitat.

4.1. Descripció de l'algorisme

Per aquesta part, hem decidit utilitzar l'algorisme de *Hill Climbing* seguint els criteris de *first improvement* i *best improvement*.

Abans de descriure l'algorisme, hem de definir alguns conceptes mencionats anteriorment:

- Espai de solucions: l'espai on ens mourem serà el conjunt de tots els PIDS possibles generats a partir de la solució inicial.
- Solució inicial (estat inicial): la solució inicial serà el nostre punt de partida. En el nostre cas ho generarem amb l'algorisme voraç descrit anteriorment.
- Tal i com es demana en el problema, el nostre objectiu és minimitzar el nombre de vèrtex que pertanyen al PIDS.
- Operadors: tenim com a únic operador treure un vèrtex del PIDS perquè busquem minimitzar la mida del PIDS.

Cal destacar que, tal i com s'ha esmentat anteriorment, l'algorisme voraç que s'ha plantejat no garanteix un PIDS minimal, de manera que és probable que, donat un conjunt dominador inicial, es pot trobar diversos vèrtexs els quals es poden treure d'aquest conjunt. Però això no vol dir que es puguin treure cadascun d'ells i seguir tenint un PIDS com a resultat, sinó que existeix una dependència entre aquests vèrtexs sobrants i, el fet de treure un d'ells, podria provocar que no es pogués treure un altre. És per aquest motiu que ens pot interessar aplicar una cerca local: l'objectiu és definir una estratègia de minimització que ens permeti poder treure el màxim nombre possible de vèrtexs del conjunt dominador.

First Improvement

D'acord amb l'objectiu de minimitzar el nombre de vèrtexs que pertanyen al PIDS, la cerca local amb first improvement consisteix en, de tots els vèrtexs que se'n poden prescindir en el conjunt dominador, anar traient-los a mesura que es van trobant fins que no es puguin treure més. Per tant,

aquesta cerca es pot simplificar com a un recorregut lineal on, per cada vèrtex que forma part del PIDS, es comprova si després de treure-ho segueix sent PIDS. El seu cost serà, per tant, O(n+m).

Best Improvement

En el cas del best improvement, l'estratègia que es planteja consisteix en, per cada conjunt de vèrtexs que es poden treure del conjunt dominador en cada iteració (on cada iteració consisteix en eliminar un dels vèrtexs possibles), es treu del conjunt aquell que menys adjacències tingui.

Aquesta estratègia es basa en què, quant menys adjacències tingui un vèrtex candidat a ser tret, més probable serà que no existeixi cap relació entre aquest vèrtex candidat i un altre, de manera que més probable serà també que aquest altre vèrtex es pugui treure durant les següents iteracions (per exemple, si tenim un conjunt de vèrtexs prescindibles on un d'aquests només té una adjacència i la resta en té moltes, treure el vèrtex amb una única adjacència podria provocar que deixés de poder-se eliminar com a molt un vèrtex, mentre que si trèiem un vèrtex amb moltes adjacències, deixarien de poder-se eliminar fins a tants vèrtexs com adjacències tingui, incloent el vèrtex amb una sola adjacència).

4.2. Experimentació

Per a l'experimentació disposem de diferents instàncies (networks). Estudiarem principalment la mida del PIDS resultant i el temps que ha trigat en acabar l'execució. Aquest experiment ho farem tant per la tècnica de first improvement com pel best improvement.

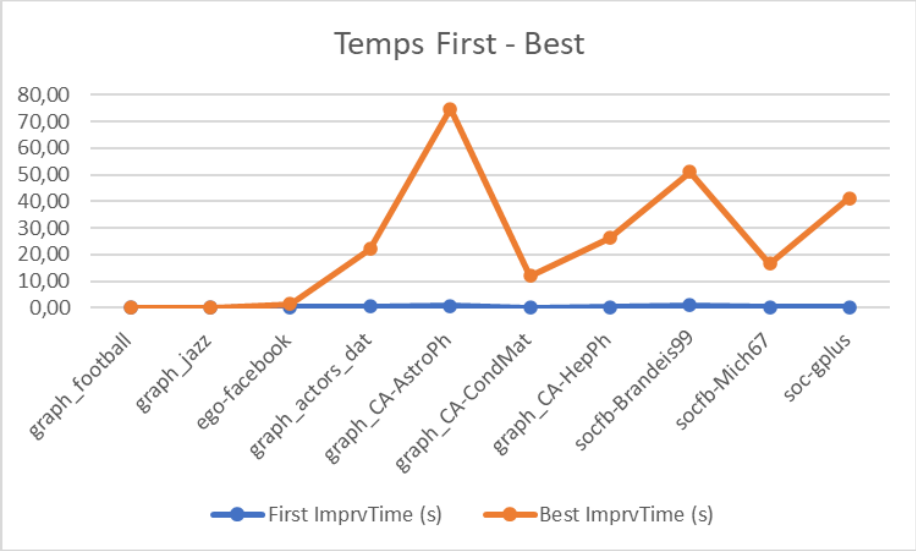
Aplicant cerca local amb *first improvement*, hem obtingut els següents resultats:

Network	Valor	Temps (segons)
graph_football	71	0.00
graph_jazz	83	0.00
ego-facebook	1979	0.22
graph_actors_dat	3203	0.34
graph_CA-AstroPh	6978	0.57
graph_CA-CondMat	9784	0.08
graph_CA-HepPh	4831	0.31
socfb-Brandeis99	1524	0.83
socfb-Mich67	1445	0.27
soc-gplus	8320	0.22

D’altra banda, aplicant *best improvement*, els resultats han sigut:

Network	Valor	Temps (segons)
graph_football	71	0.00
graph_jazz	83	0.00
ego-facebook	1979	1.22
graph_actors_dat	3204	21.82
graph_CA-AstroPh	6974	74.18
graph_CA-CondMat	9783	11.87
graph_CA-HepPh	4827	25.93
socfb-Brandeis99	1517	50.21
socfb-Mich67	1438	16.28
soc-gplus	8301	41.02

Podem observar clarament que hi ha una gran diferencia en quan a temps d’execució. Com mostra la gràfica següent, veiem que en el cas del first improvement, el temps d’execució de la cerca per a les diferents instàncies es quasi constant i pel altre cas, utilitzant best improvement es pot veure que segons quina instancia, assoleix a un temps alt.



Respecte a la mida del PIDS que s’obté utilitzant el first improvement o best improvement, podem apreciar també una certa diferència. Si utilitzem la tècnica de best improvement, obtenim, en general, millors solucions (PIDS amb menor nombre de vèrtex) que si utilitzéssim el first improvement.

Per tant, en termes de temps, fer ús del first improvement implica més rapidesa, però el que guanyem de velocitat el perdem en la qualitat de la solució, ja que, com hem vist, encara que sigui mínima, utilitzar best improvement dona millors resultats.

SEGONA PART

5. Metaheurística

La metaheurística que s'ha implementat ha sigut l'algorisme genètic, ja que donades les característiques del problema de MPIDS, la representació genètica de les solucions és senzilla i permet aplicar fàcilment les operacions necessàries.

5.1. Descripció de l'algorisme

L'algorisme genètic està inspirat en el procés de selecció natural: partim d'una població inicial d'individus i, depenent de la qualitat d'aquests (funció d'aptitud), es fa una selecció aleatòria on aquells amb millor qualitat tenen més probabilitat de ser escollits. A partir de la població seleccionada, els individus s'agrupen en parelles i cadascuna d'aquestes fan un creuament, donant lloc a nous individus, els quals seran els fills. Finalment, cadascun dels fills tenen una mutació i, amb la població resultant, es torna a aplicar el mateix cicle fins que es compleixin unes condicions específiques.

Si aquest concepte evolutiu ho extrapolem al problema de MPIDS, podem definir els individus com cadenes de bits que representen un graf, on cadascun dels bits indica si el vèrtex corresponent pertany al conjunt dominador o no, i és aquesta mateixa cadena la que representa una possible solució al problema.

La funció d'aptitud és, per tant, un indicador de la bondat d'aquesta solució. Aquesta funció es pot plantejar de diverses maneres, i depèn de com es planteja la cerca de solucions, tenint en compte com es genera la població inicial.

Si volem començar amb una població inicial aleatòria o bé on tots els individus no tenen cap vèrtex al conjunt dominador, aquesta funció es podria calcular com el número de vèrtexs que pertanyen al conjunt dominant, amb un factor penalitzador segons si l'individu és un PIDS o no (per tal d'evitar sortir-nos de l'espai de solucions), d'aquesta manera es podria minimitzar els vèrtexs d'aquest conjunt, que és l'objectiu del problema. Una altra opció seria donar com a resultat el número de vèrtexs vàlids, és a dir, vèrtexs que compleixen la propietat de què la meitat del seu veïnatge forma part del conjunt dominador, de manera que quant més vèrtexs compleixen la propietat esmentada, més s'aproparia un individu a ser una solució final.

Si per contra decidim formar una població inicial on tots els individus ja son una solució bastant bona (aplicant un algorisme voraç, per exemple), es podria fixar el PIDS amb menor nombre de vèrtexs pertanyents al conjunt dominador (la millor solució) d'un moment determinat i, respecte a aquest individu, calcular el nombre de vèrtexs pertanyents al conjunt dominador per cada individu i penalitzar aquells que tinguin un nombre major que el mínim actual. D'aquesta manera,

es donaria més prioritat als individus que menys vèrtexs dominadors tenen, però cal tenir en compte que només tenint en compte això no es garanteix que tots els individus estudiats siguin PIDS, per tant, aquells individus que deixen de ser-ho (ja sigui degut a l'encreuament o a la mutació), tindran una aptitud nul·la perquè el que ens interessa seria mantenir en tot moment solucions vàlides dins les poblacions. Aquesta opció consistiria per tant en, en comptes d'anar construint el PIDS, partir directament d'una població on tots els individus son solucions, i anar minimitzant el màxim possible a partir de les operacions evolutives.

Finalment, en relació a les operacions evolutives, aquestes son bastant genèriques dins les diferents variacions d'algorismes genètics: l'encreuament consisteix en agafar un punt aleatori (o múltiples) de la cadena de bits dels dos individus, fer un tall i intercanviar un subconjunt d'una cadena pel subconjunt de l'altre; la mutació s'encarrega de seleccionar un bit aleatori i negar-ho.

En el nostre cas, hem escollit l'opció de partir directament d'una població on tots els individus són PIDS, i tractem de minimitzar-ho seguint l'estratègia esmentada anteriorment. El pseudocodi és el següent:

- Crear un individu a partir de l'algorisme voraç
- Inicialitzar una població de mida N, on cada individu és una còpia d'aquest
- Guardar aquest primer individu com a millor solució
- Inicialitzar l'aptitud de cada individu amb la puntuació màxima
- Mentre el temps d'execució no superi el temps límit i hi ha algun individu PIDS:
    - Selecció dels individus
    - Encreuament
    - Mutació
    - Comprovar si hi ha una solució millor a l'actual
    - Calcular funció d'aptitud

Per calcular la funció d'aptitud i guardar els individus, s'ha implementat un precàlcul de la mida del conjunt dominador i el nombre de vèrtexs vàlids de cada individu. Calcular la mida del conjunt dominador consisteix en comptar quants uns té cada individu, i calcular el nombre de vèrtexs vàlids requereix fer un recorregut en el graf i comprovar per cada vèrtex si la meitat del seu veïnatge forma part del conjunt dominador, segons el conjunt que tingui l'individu a ser estudiat.

La bondat d'una solució la definirem en un interval de [0-100], on 100 serà la millor puntuació.

Així doncs, la funció d'aptitud calcula, per cada individu:

- Si no és PIDS, és a dir, el seu nombre de vèrtexs vàlids no coincideix amb el nombre total de vèrtexs, la seva qualitat serà 0.
- Si és PIDS, inicialment tindrà una puntuació de 100, però s'aplica una penalització de 10 punts per cada unitat que sobrepassa la mida del seu conjunt dominador amb la mida de la millor solució d'aquell moment. Per exemple, si tenim un individu PIDS amb una mida de 10 vèrtexs, i la millor solució en té 8, s'aplica una penalització de $(10 - 8) * 10 = 20$ punts.

Analitzant el cost d'aquest algorisme:

- Cost d'aplicar l'algorisme voraç, on n és el nombre de vèrtexs i m el nombre d'adjacències: $O(n(mlogm + m))$.
- Cost d'inicialitzar una població: $O(n\_població * n) = O(n)$, on *n_població* és una constant, i per tant té cost lineal.
- Cost de calcular el nombre de vèrtexs vàlids de cada individu: consisteix en recórrer tots els vèrtexs i, per cada vèrtex, comprovar si almenys la meitat del seu veïnatge forma part del conjunt dominador, per tant té un cost de $O(n\_població * n * m) = O(n * m)$.
- Cost de calcular la mida del conjunt dominador de cada individu: només requereix de fer un recorregut sobre cada cadena de bits de la població, per tant el cost és $O(n\_població * n) = O(n)$.
- Cost de seleccionar la millor solució del moment: com que ja tenim els valors de vèrtexs vàlids i mida del conjunt dominador, el cost de seleccionar la millor solució és constant $O(n\_població) = O(1)$.
- Cost de calcular la funció d'aptitud: una vegada més, gràcies als valors precalculats, la qualitat de cada solució també es pot calcular amb cost constant $O(n\_població) = O(1)$.
- Cost de fer un encreuament per cada parella d'individus: $O(n)$.
- Cost de fer una mutació per cada individu: $O(1)$.

5.2. Experimentació

Per a la experimentació d'aquesta metaheurística, s'han realitzat 10 aplicacions per cada graf, amb un temps límit d'execució de 600 segons.

Fixant una població de 100 individus, els resultats han sigut els següents:

Network	Valor mitjà
graph_football	63.5
graph_jazz	82.1
ego-facebook	1977.9
graph_actors_dat	3277.1
graph_CA-AstroPh	7119.3
graph_CA-CondMat	9940,3
graph_CA-HepPh	4913,2
socfb-Brandeis99	1573,8
socfb-Mich67	1500.6
soc-gplus	8460.6

Tal i com podem observar, en general, la metaheurística proposada no ha donat bons resultats si ho comparem amb l'experiment anterior de la cerca local. Si bé és cert que en alguns casos ha donat un millor resultat, com ara el graf de "graph-football", en la majoria de casos només s'ha pogut treure un nombre molt petit de vèrtexs, partint de la població inicial donada (recordem que s'obté a partir de l'algorisme voraç).

Creiem que això és degut a que la estratègia plantejada amb l'algorisme genètic s'apropa massa a una cerca aleatòria i, per tant, no només es triga molt en trobar un vèrtex que es pugui treure del conjunt dominador, sinó que també hi ha una probabilitat molt gran de que fent una mutació o un encreuament, els individus deixin de ser PIDS.

Aquest experiment depèn també de la mida prefixada de la població: en el cas de "graph-football", per exemple, és un graf que té 115 vèrtexs, de manera que si utilitzem una població de 100 individus, hi ha una probabilitat molt més gran d'encertar a l'hora d'aplicar els operadors evolutius. En canvi, amb grafs enormes, difícilment es podrà trobar millors solucions.

6. Model ILP executat amb CPLEX

En aquest apartat s'implementa el model ILP del problema MPIDS per a ser executat amb CPLEX. És un model basat en la programació lineal entera.

La programació lineal entera es un mètode d'optimització per resoldre un problema amb restriccions lineals i una funció d'utilitat (o també anomenat d'aptitud) que volem maximitzar o minimitzar. Com ja explica l'enunciat, la nostra funció d'utilitat serà la cardinalitat del PIDS solució que el volem minimitzar i com a restriccions tenim que per a tot vèrtex, almenys la meitat del seu veïnatge pertany al PIDS i que sigui x_i els vèrtex del PIDS, només pot prendre un valor binari.

Usarem el software que ofereix CPLEX per resoldre el problema amb programació lineal.

6.1. Descripció algorisme

L'objectiu del model és minimitzar el número de vèrtexs i les restriccions (constraints) és que tots els vèrtexs tenen com a mínim la meitat del seus veïns dins el conjunt de solució, i les decisions que s'han de fer és incloure o no incloure un vèrtex al conjunt de solució.

6.2. Experimentació

Per dur a terme els experiments, s'ha fixat un temps màxim de 5000 segons, i hem obtingut els següents resultats:

Network	Valor	Temps (segons)	Gap
graph_football*	63	3.12	0
graph_jazz*	79	0.1	0
ego-facebook*	1973	0.63	0
graph_actors_dat	3092	445.85	0.26
graph_CA-AstroPh	6739	1679.03	0.31
graph_CA-CondMat	9585	134.18	0.10
graph_CA-HepPh	4718	7.09	0.07
socfb-Brandeis99	1400	3683.33	1.41
socfb-Mich67	1331	4763.36	1.71
soc-gplus*	8244	0.07	0

(*) Optimalitat assegurada

Després dels experiments es pot observar que per 4 grafs, s'ha pogut trobar l'òptim, és a dir, la solució obtinguda és el MPIDS. És l'algorisme que ha tingut més solucions òptimes, i també s'ha arribat a la millor solució possible dins del termini de temps.

7. Conclusions

Amb aquest projecte hem après sobre les diferents tècniques que es poden aplicar per resoldre un mateix problema.

Respecte l'algorisme voraç, en un principi, la nostra estratègia va ser anar seleccionant els vèrtexs que tenien una cardinalitat més gran fins que sigui un MPIDS, ja que pensàvem que tenia una major probabilitat de pertànyer al conjunt D solució. Un cop fet els experiments, vam observar que no era gens bona, perquè en la majoria dels casos s'acabava seleccionant pràcticament tot el graf. Fent un replantejament d'aquesta estratègia vam pensar en la descrita en aquest informe, la qual a diferència de la estratègia inicial, en aquest, no seleccionem directament el vèrtex si no que ens fixem en les seves adjacències i seleccionem la meitat que té major cardinalitat. Mitjançant els experiments, hem pogut comprovar que aquesta segona estratègia és molt més eficient temporalment que el primer, i la mida del conjunt solució té una cardinalitat molt menor.

Sobre la cerca local, hem vist que, efectivament, partint des de la solució inicial generat amb algorisme voraç que hem pogut generar millors solucions. En quant a les dues tècniques que hem implementat (first improvement/ best improvement), si volem rapidesa, es millor utilitzar la primera estratègia, però, si volem qualitat, és més convenient utilitzar la segona estratègia.

En quant a la metaheurística, tot i que en un principi semblava una bona proposta, hem acabat observant que s'apropava massa a una cerca aleatòria: el problema del MPIDS, tot i que la seva representació fos factible per l'algorisme genètic, els operadors evolutius no eren molt adients per a la resolució d'aquesta.

El model de Programació Lineal Entera (ILP), mitjançant els experiments, es pot veure que es la que proporciona millors resultats en termes d'eficiència temporal i número de nodes, ja que el model ILP intenta buscar la solució òptima dins un interval de temps determinat, cosa que coincideix amb l'objectiu d'aquest problema. En aquest cas la solució òptima que volem assolir seria trobar el Mínim Conjunt Dominador d'Influència Positiva del graf proporcionat.

Implementar programació lineal en aquest problema ha sigut senzill, ja que la funció objectiu era molt clara: minimitzar el número de vèrtexs en el conjunt solució, com també les condicions: que cada vèrtex tingui com a mínim la meitat dels seus veïns formin part del conjunt solució.

Podem concloure finalment que, entre tots els algorismes plantejats, el model de ILP executat amb CPLEX ha sigut el que millor ha resolt el problema del MPIDS. Tot i així, cal destacar també que l'algorisme proposat fent servir l'algorisme voraç amb cerca local implementant *best improvement* també dona resultats bastant bons.

8. Referències

Wikipedia. Metaheuristic. https://en.wikipedia.org/wiki/Metaheuristic. [2021]

Wikipedia. Genetic algorithm. https://en.wikipedia.org/wiki/Genetic_algorithm. [2021]

Brilliant Math & Science Wiki. Linear programming. https://brilliant.org/wiki/linear-programming/. [2021]

IBM Documentation. What is CPLEX? https://www.ibm.com/docs/en/icos/20.1.0?topic=mc-what-is-cplex. [2021]