Facultat d'Informàtica de Barcelona

GIA UPC

Optimització

Implementació de l'algoritme del Simplex Primal

Alumnes :
Abril Risso Matas, 45182567
Irene Pumares Benaiges, 54574822

Tutors :
Mari Paz Linares Herreros

April 12, 2024

Contents

1 Introducció

En aquesta pràctica es demana la implementació de l'algoritme Simplex Primal vist anteriorment a classe.

S'ha escollit utilitzar el llenguatge de programació Python per la seva simplicitat sintàctica, ja que facilita la implementació d'algorismes complexos de manera clara i entenedora. A més, també disposa de la llibreria NumPy, la qual s'ha utilitzat en aquesta pràctica i ens ha pogut proporcionar diferents mètodes per al tractament de matrius i per a la resolució de l'algorisme.

Respecte al càlcul de la solució bàsica factible inicial (SBF) amb la fase I del símplex, es proposaven dues opcions. Finalment, s'ha decidit implementar la primera opció, la qual consisteix a desenvolupar un codi únic que integrés la fase I i formulés i resolgués el problema automàticament a partir dels paràmetres que defineixen el problema original (c, b i A). Un cop identificada una SBF, si aquesta existeix, el codi continuaria amb la fase II.

2 Implementació del codi

La implementació de l'algoritme s'ha organitzat en dos fitxers. El primer s'encarrega de la lectura del fitxer de text que conté els problemes a resoldre, emmagatzemant en tres arrays de NumPy: la funció objectiu "c", les restriccions del problema "A" i els termes independents "b" de cada restricció del problema que l'usuari hagi triat executar. El segon fitxer conté l'algoritme en si mateix, que rep com a entrada les variables processades pel primer fitxer i aplica l'algoritme Simplex al problema escollit. Aquesta estructura permet a l'usuari executar qualsevol problema present al fitxer de text "input" els cops que vulgui i alhora permet una separació clara entre la preparació de dades i l'execució de l'algoritme.

2.1 llegir.py

Aquest fitxer conté una sola funció dissenyada per a poder llegir l'arxiu de text que conté tots els problemes. S'ha decidit poder donar l'opció de resoldre qualsevol dels problemes de qualsevol dels alumnes. La funció s'anomena **llegir_dades** i rep com a paràmetres el nom de l'arxiu (**n_arxiu**), el número d'alumne (**n_alumne**) i el número de problema (**n_problema**) que es vol resoldre.

En primer lloc, s'obre l'arxiu de text en mode lectura i es llegeixen totes les línies, les quals s'emmagatzemen en una llista anomenada 'lines'. A continuació, s'inicialitzen les tres variables que seran retornades al final de la funció i es defineixen les expressions regulars *'alumne_info'* i *'problema_info'* per identificar el número del alumne i del problema, respectivament.

Després d'establir les variables necessàries, es procedeix a recórrer totes les línies del fitxer fins a trobar el número de l'alumne i el número del problema corresponent. Un cop s'han trobat les dades del problema, es busquen les variables *'c'* (vector de coeficients de la funció objectiu), *'A'* (matriu de coeficients de les restriccions) i *'b'* (vector de termes independents de les restriccions) dins del fitxer. Quan troba "c=", "A=" i "b=" al text, els seus continguts s'assignen a les variables corresponents. Un cop es troba i emmagatzema el vector *'b'*, s'assumeix que ja es tenen totes les dades. Per tant, surt del bucle que recorre les línies i la funció retorna les tres variables *'c'*, *'A'* i *'b'*.

2.2 simplex.py

2.2.1 main

Aquesta part del codi és la que s'encarrega de l'execució seqüencial de l'algorisme, és a dir, de cridar cada una de les funcions auxiliars necessàries.

En primer lloc, es defineix el nom de l'arxiu d'entrada '*input.txt*' que conté les dades del problema. A continuació, es demana a l'usuari que ingressi el número d'alumne '*n_alumno*' i el número de problema '*n_problema*' que es vol executar per poder identificar les dades corresponents dins de l'arxiu. Amb aquests paràmetres definits, es crida la funció *llegir_dades* per extreure les dades necessàries, incloent-hi la funció objectiu (**funcio_objectiu**), la matriu de coeficients de les restriccions '*A*', i el vector de termes

independents '*b*'.

Posteriorment, es calculen les següents variables que seran utilitzades més endavant en l'execució de l'algorisme:

- **num_restriccions**: s'utilitza la propietat *shape* de la matriu A per calcular el nombre de restriccions, que correspon al número de files de la matriu.
- **num_variables**: s'utilitza la propietat *shape* de la martiu A per a calcular el nombre de variables, que correspon al número de columnes de la matriu.
- **b**: s'ajusta la forma del vector b per convertir-lo en una matriu columna, ja que aquest format és necessari per a certes operacions posteriors.

Un cop definit el problema, s'inicia l'algorisme del Simplex. S'imprimeix un missatge indicant aquest inici i es crida la funció *fase1()* per iniciar la primera fase de l'algorisme, passant-li la matriu A com a argument. El resultat es guarda en la variable *resultat_fase1*. A continuació, es comprova si aquest resultat és *None*. Si ho és, significa que el problema no és acotat o no és factible, i en aquest cas, el programa s'atura. En cas contrari, es desempaqueten els resultats de la primera fase per preparar les dades necessàries per a la segona fase, i es guarden en les variables corresponents. Aleshores, es crida la funció *fase2()* per iniciar la segona fase, passant-li com a argument les variables retornades per la primera fase, necessàries per continuar amb les iteracions.

Finalment, quan aquesta funció acaba, es surt del programa amb la comanda *exit()* i es dona per finalitzat el problema.

2.2.2 simplex()

Aquesta funció és el nucli de l'algorisme Simplex, dissenyada per trobar la solució òptima que minimitza una funció objectiu donada, si aquesta existeix. Gestiona les dues fases principals de l'algoritme: la Fase 1 i la Fase 2.

Aquesta implementació es pot dividir en diverses parts clau, cadascuna amb un propòsit específic dins del procés de solució.

Fases del Algorisme

La funció comença determinant en quina de les dues fases de l'algorisme es troba: la Fase 1, on s'identifica una solució bàsica factible inicial (SBF), o la Fase 2, on es busca la solució òptima a partir de l'SBF trobada en la Fase 1.

- **Fase 1**: En aquesta etapa es crida a les funcions *afegir_variables_artificials* i *inicialitzacio_fase1* explicades posteriorment.
- **Fase 2**: En aquesta fase es crida a la funció *inicialitzacio_fase2* també explicada posteriorment.

Bucle Principal

Dins del bucle infinit *'while True'*, la funció realitza iteracions per moure's per l'espai de solucions factibles fins trobar la condició d'optimalitat o determinar que el problema no té solució factible o és no acotat.

1. Comprovació d'Optimalitat

Primer, es verifica si la solució actual és factible (tots els elements de *X_ b*, els valors de les variables bàsiques de la base actual, són no negatius).

Si es troba en la Fase 1 i la solució és factible, es comprova si els costos reduïts (representats per la variable *'r'*) indiquen que la solució actual és òptima, cridant a la funció *costos_ reduïts*. Abans de confirmar la optimalitat de la solució, es comprova que el valor de la funció objectiu *'z'* sigui igual a 0. Cal fer aquesta comprovació ja que, en la Fase 1, la z representa la suma de les variables artificials, per tant, un valor de z=0 indica que totes les variables artificials poden ser eliminades del problema (és a dir, tenen valor 0). En aquest cas, s'indicarà que s'ha trobat una solució bàsica factible i es mostrarà la iteració en la qual s'ha trobat aquesta, seguidament es retornaran totes les variables necessàries per a la Fase 2. En el cas que la z sigui diferent de 0, es mostrarà *'El problema no té solució factible'* i la funció retornarà None ja que, almenys, una de les variables artificials és positiva en la solució òptima al problema artificial. Com les variables artificials van ser afegides únicament per facilitar la búsqueda d'una SBF i no tenen significat en el problema original, una variable artificial amb valor indicaria que no es poden satisfer totes les restriccions del problema original.

Si es troba en la Fase 2, directament comprova que la solució sigui òptima amb els costos reduïts ja que anteriorment a la Fase 1 hem pogut comprovar que existeix una solució bàsica factible i, per tant, sabem amb certesa que l'actual també ho és.

2. Selecció de les variables d'Entrada i Sortida

Un cop s'ha comprovat si la solució és òptima, en el cas que no ho sigui, tant si ens trobem a la Fase 1 com a la Fase 2, es procedeix a calcular la variable que ha d'entrar a la base *(variable_ entrada)* cridant a la funció *calcul_ variable_ entrada* que serà explicada posteriorment. Aquesta funció identifica, dins del conjunt de variables no bàsiques, quina és la més adeqüada per ser introduïda a la base i s'emmagatzema a la variable *variable_ entrada*.

Seguidament, es calcula la direcció bàsica factible amb la funció *direccio_ basica_ factible* també explicada en apartats posteriors, la qual retorna la direcció en la qual s'ajustaran les variables bàsiques per permetre que la variable d'entrada s'introdueixi en la base. En la Fase 1, aquest ajust es realitza amb l'objectiu d'arribar a una solució que cumpleixi totes les restriccions del problema, és a dir, trobar factibilitat. En canvi, si ens trobem a la Fase 2, aquest ajust tracta de mantenir la factibilitat de la solució, ja que ja ens trobem en una SBF. Un cop tenim la direcció bàsica factible comprovem que el problema sigui acotat cridant a la funció *es_ acotat* ja que, si la funció indica que el problema no

és acotat, està representant que el valor de la funció objectiu millora indefinidament en la direcció donada com a paràmetre de la funció i, per tant, el problema no tindrà solució òptima finita.

A continuació, es determina quina variable ha de sortir de la base *variable_sortida* bassant-se en el càlcul de **theta**, cridant a la funció *calcul_theta* comentada posteriorment.

3. Actualització de la Solució

Un cop ja tenim les variables d'entrada i sortida identificades, es crida a la funció *actualitzacio*, la qual actualitza les variables necessàries per la següent iteració.

Finalment, després de realitzar les actualitzacions necessàries, d'imprimeix la informació detallada sobre la iteració actual, incloent la variable d'entrada, la de sortida, el valor de theta i el valor de *'z'*.

Si a la Fase 1 s'ha trobat una SBF, l'algoritme està llest per passar a la Fase 2, per tant, la funció retorna les variables necessàries per aquesta.

En la Fase 2, quan surti del bucle principal, s'indicarà que s'ha trobat la solució òptima del problema plantejat.

2.2.3 fase1()

Aquesta funció s'encarrega de **cridar a la funció simplex() per a la Fase 1** de l'algoritme. Rep com a paràmetres la matriu de coeficients de les restriccions A. Dins de la funció, es passa el paràmetre *fase=1*, per indicar que s'està executant la Fase 1, i s'inclouen també els paràmetres A i b que es necessiten per al càlcul.

2.2.4 fase2()

Aquesta funció és similar a l'anterior, ja que també **fa una crida a la funció simplex(), però en aquest cas per a la Fase 2** de l'algoritme. Rep com a paràmetres diferents variables necessàries per continuar amb l'execució de la Fase 2 i poder calcular la següent iteració de l'algoritme.

2.2.5 funcions_auxiliars.py

2.2.5.1 afegir_variables_artificials()

Aquesta funció, tal i com el seu nom indica, **afegeix les variables artificials** al problema. Només és cridada una vegada, just a l'inici de la Fase 1. Pren com a paràmetre la matriu A i crea una matriu identitat de la mida *num_restriccions* per representar les variables artificials, una per a cada restricció. Aleshores concatena horitzontalment aquesta matriu a la matriu original del problema. Finalment retorna la matriu *A_ampliada* amb les variables artificials afegides.

2.2.5.2 inicialitzacio_fase1()

Aquesta funció serveix per a començar la iteració de la fase 1, és a dir, **prepara les dades per a poder començar l'algorisme**. Els paràmetres que necessita per al càlcul i rep la funció són els següents:

- **A**: matriu de coeficients de les restriccions
- **A_ampliada**: matriu A amb les variables artificials afegides
- **b**: vector de termes independents del sistema d'equacions de restriccions

Primer de tot, crea les llistes **basiques** i **no_basiques** que contenen les variables bàsiques i no bàsiques, respectivament. També es crea la llista **basiques_original**, que és una còpia de **basiques**, fem una copia perquè aquesta serà modificada durant les iteracions. La llista de **bàsiques_original** ens serveix per emmagatzemar quines variables són les artificials afegides a la funció *afegir_variables_artificials*. Seguidament, es configura la nova funció objectiu (**c**), que serà un vector de llargada de la suma de les variables originals més les artificials amb coeficients 0 a les originals i 1 a les noves. També s'inicialitza la matriu identitat **B** de grandària de *nombre de restriccions* × *nombre de variables bàsiques*, que representa els coeficients de les restriccions de les variables bàsiques, i la seva inversa (**inversa_B**), que es crea com una còpia de la matriu B ja que, la inversa de la matriu identitat és ella mateixa. Es defineix **A_n**, que representa els coeficients de les restriccions de les variables no bàsiques. Es divideix també el vector de la funció objectiu en dos, un que conté els coeficients per a les variables bàsiques (**C_b**) i un altre per a les no bàsiques (**C_n**). La funció també calcula el vector dels valors de les variables bàsiques **X_b** amb la fòrmula corresponent: $X_b = \text{inversa\_B} \times b$ i crea un un vector de zeros per emmagatzemar els valors de les variables no bàsiques (**X_n**), ja que aquestes tenen valor 0. Finalment, utilitza la funció **calcul_z()** per a calcular el valor inicial de la funció objectiu i retorna tots aquests valors que formen l'estat inicial del problema.

2.2.5.3 inicialitzacio_fase2()

Aquesta funció és molt similar a l'anterior, però **prepara les dades per a l'inici de la Fase 2**. Rep com a paràmetres totes les variables necessàries per al càlcul que són les següents:

- **basiques**: llista de les variables bàsiques en la Fase 1
- **basiques_original**: llista de variables bàsiques original (no s'ha modificat durant l'execució)
- **no_basiques**: llista de les variables no bàsiques en la Fase 1
- **funcio_objectiu**: coeficients de la funció objectiu del problema original
- **b**: vector de termes independents del sistema d'equacions de restriccions
- **inversa_B**: inversa de la matriu de la base trobada a la Fase 1, ja calculada prèviament

En primer lloc, s'eliminen les variables artificals. Per fer-ho, s'itera sobre les variables de la llista *bàsiques_originals* (artificials) i les elimina de la llista **no_basiques**. Tot seguit, es calculen **C_b** i **C_n**, que representen els coeficients de la funció objectiu corresponents a les variables bàsiques i no bàsiques, respectivament. També es calculen els valors de les variables bàsiques (**X_b**) i no bàsiques (**X_n**) així com la matriu dels coeficients de les restriccions de les variables no bàsiques **A_n**. Per últim, es calcula el valor de **z** amb les variables calculades anteriorment i es retornen les variables definides en la funció.

2.2.5.4 calcul_z()

Aquesta funció retorna el valor de la funció objectiu, que es calcula de la següent manera: $z = C_b \cdot X_b + C_n \cdot X_n$

Els paràmetres que rep la funció són els següents:

- **C_b**: coeficients de la funció objectiu de les variables no bàsiques
- **X_b**: els valors de les variables bàsiques de la base actual
- **C_n**: coeficients de la funció objectiu de les variables no bàsiques
- **X_n**: vector dels valors de les variables no bàsiques de la solució factible actual

En aquest cas, com el valor de les variables no bàsiques és 0, la fòrmula quedaria de la següent manera: $z = C_b \cdot X_b$. Tot i així en el nostre programa hem afegit els vectors C_n *i* X_n per assegurar-nos que feiem tots els càlculs correctament.

2.2.5.5 costos_reduits()

Aquesta funció **calcula els costos reduïts de les variables no bàsiques**, rebent com a paràmetres d'entrada:

- **C_n**: vector dels coeficients de la funció objectiu de les variables no bàsiques
- **C_b**: vector dels coeficients de la funció objectiu de les variables bàsiques
- **inversa_b**: inversa de la matriu de base B
- **A_n**: vector dels coeficients de les restriccions de les variables no bàsiques de la solució factible actual

La fòrmula pel càlcul és la següent: $r = C_n - C_b \cdot \text{inversa\_B} \cdot A_n$

2.2.5.6 es_optim()

La funció **verifica si la solució actual és òptima**, és a dir, donats els costos reduïts **r**, mira si tots els elements són més grans o iguals a 0, i retorna un booleà indicant si es compleix (True) o no (False).

2.2.5.7 calcul_variable_entrada()

Aquesta funció **tria la variable no bàsica a introduir a la base en la propera iteració**. Com a paràmetres d'entrada rep:

- **r**: costos reduïts
- **no_basiques**: llista de les variables no bàsiques

La funció recorre el vector r i si el valor del cost reduït és negatiu, s'afegeix l'índex de la variable en una llista anomenada *negatius*. Aleshores, es crea una nova llista, *valors_negatius*, que conté les variables no bàsiques associades als índexs emmagatzemats a *negatius*. S'aplica la **regla de Bland** per a seleccionar la variable, és a dir, si hi ha valors negatius a la llista *valors_negatius*, es **selecciona la variable no bàsica amb l'índex més petit**. Finalment, retorna la variable no bàsica seleccionada com a variable d'entrada a la base. Si no hi ha valors negatius a *valors_negatius*, es retorna *None*, indicant que no hi ha cap variable no bàsica amb cost reduït negatiu i, per tant, no es pot millorar la solució actual.

2.2.5.8 direccio_basica_factible()

La funció **calcula la direcció bàsica factible per la transició cap a una nova solució bàsica**. Rep com a paràmetres els següents:

- **inversa_B**: matriu inversa de B
- **A_n**: matriu dels coeficients de les restriccions de les variables no bàsiques
- **variable_entrada**: variable d'entrada a la base
- **no_basiques**: llista de les variables no bàsiques

Primer de tot, es troba l'índex de la variable d'entrada a la base a la llista de variables no bàsiques. Arribats a aquest punt, ja es pot calcular i retornar la direcció bàsica factible **d**, que es calcula de la següent manera:

$$\mathbf{d} = -\mathbf{B}^{-1} \cdot \mathbf{A}_{\text{entrada}}$$

2.2.5.9 es_acotat()

Aquesta funció **determina si la direcció bàsica factible d que rep com a paràmetre d'entrada manté la solució dins l'espai factible acotat**. Per a comprovar-ho, recorre cada element del vector d i, si algun dels elements és menor o igual a 0, significa que la solució pot sortir de l'espai factible, pero lo que la funció retorna *False*. Si tots els elements són positius, la solució es mantindrà dins l'espai factible i retornarà *True*.

2.2.5.10 calcul_theta()

Aquesta funció **calcula el valor mínim de la longitud de pas en la direcció de la solució bàsica factible**.

La funció rep com a paràmetres:

- **X_b**: valor de les variables bàsiques de la base actual
- **d**: direcció bàsica factible
- **basiques**: llista de les variables bàsiques

Primer de tot, troba tots els possibles valors de θ. Després, troba el valor mínim de θ i aplica la **Regla de Bland** per determinar la variable de sortida corresponent en cas d'empat. Si no hi ha valors possibles de θ, significa que la solució no és acotada, i la funció retorna *None*. Si hi ha valors possibles, aleshores retorna la variable de sortida corresponent i el valor mínim de θ.

2.2.5.11 es_degenerada()

Aquesta funció verifica en la Fase 1, quan s'ha comprovat la optimalitat de la solució i que el valor de la funció objectiu formada per les variables artificials sigui igual a zero, que en el vector dels valors de les variables bàsiques no hi hagi cap zero. En el cas que hi hagues un zero, estariem en una solució degenerada, la qual cosa comportaria cicles infinits.

2.2.5.12 actualitzacio()

Aquesta funció, com bé indica el seu nom, **s'encarrega d'actualitzar els valors de les variables en cada iteració de l'algorisme**, tant per a la fase 1 com per a la fase 2 de l'algorisme.

Els paràmetres d'entrada que rep són els següents:

- **basiques**: llista de les variables bàsiques en la iteració actual
- **no_basiques**: llista de les variables no bàsiques en la iteració actual
- **X_b**: valors de les variables bàsiques de la base actual
- θ: valor mínim de longitud de pas en la direcció de la SBF
- **d**: vector de la direcció bàsica factible per la transició a una nova solució bàsica
- **z_ant**: valor de z en la iteració actual
- **r_ant**: valor dels costos reduïts en la iteració actual
- **C_b**: vector dels coeficients de la funció objectiu de les variables bàsiques
- **C_n**: vector dels coeficients de la funció objectiu de les variables no bàsiques
- **A_n**: matriu que representa els coeficients de les restriccions de les variables no bàsiques del problema actual
- **A_ampliada**: matriu que inclou les variables artificials afegides a la Fase 1 de l'algorisme

- **inversa_B**: matriu inversa de B
- **variable_entrada**: variable no bàsica que entrada a la base en la propera iteració
- **variable_sortida**: variable bàsica ques surt de la base en la propera iteració

Primer de tot, s'actualitzen la llista de variables **bàsiques** i **no bàsiques**, intercanviant les variables d'entrada i sortida. A continuació s'actualitza el vector **X_b**, ajustant el valor de la variable de entrada al valor calculat θ i actualitzant la resta de variables bàsiques d'acord amb la direcció. Es calcula el nou valor de **z** sumant a z_ant el producte de θ pel cost reduït corresponent a la variable d'entrada i es comprova que sigui menor respecte de l'anterior, ja que s'ha d'assegurar que el valor disminueix. Seguidament, es calculen els nous valors per a **C_b** i **C_n**, modificant els valors a les variables d'enrada i sortida, respectivament. També es calcula el nou valor de la matriu **A_n**, que actualitza la columna corresponent a la variable d'entrada amb els valors de la matriu ampliada. Aleshores, s'actualitza la matriu **inversa_b**. Per acabar, la funció retorna els nous valors actualitzats: basiques, no_basiques, X_b, z_nova , C_b, C_n, A_n i inversa_B.

3 Solució problemes

3.1 Alumne nº 1

3.1.1 Problema 1

Table 1: Iteracions de la Fase 1 del Simplex

Iteració	Variable Entrada	Variable Sortida	Theta	Valor Z
1	1	24	0.522	2791.06
2	2	28	2.964	2467.97
3	3	23	0.694	2256.54
4	4	26	4.106	1249.58
5	7	27	0.070	1221.25
6	5	30	2.323	498.99
7	6	25	1.050	335.72
8	8	29	0.049	323.30
9	9	22	0.227	188.31
10	10	6	1.846	87.37
11	12	21	1.456	1.56×10^{-13}

Table 2: Iteracions de la Fase 2 del Simplex

Iteració	Variable Entrada	Variable Sortida	Theta	Valor Z
12	11	1	0.180	-153.97
13	6	8	0.258	-196.43
14	13	9	0.820	-373.02
15	15	6	100.881	-420.67
16	17	15	118.527	-485.60
17	1	10	2.047	-501.78
18	14	4	0.713	-520.87
19	18	14	40.301	-533.91

Table 3: Solució òptima

Variables bàsiques	Valors variables bàsiques	Costs reduïts	Valor z
12	0.93462521	147.686	
13	1.45422339	23.056	
3	2.52146319	73.137	
11	1.42757457	65.943	
1	2.13951296	152.929	-533.91284
18	40.30114207	0.254	
7	1.61477578	0.415	
2	2.18584894	18.283	
17	289.60221892	0.195	
5	3.17603769	0.013	

3.1.2 Problema 2

Table 4: Iteracions de la Fase 1 del Simplex

Iteració	Variable Entrada	Variable Sortida	Theta	Valor Z
1	1	26	0.0357	2509.04
2	2	1	0.5	2500.0
3	3	24	0.0785	2482.88
4	1	28	0.1096	2431.82
5	5	22	0.223	2278.41
6	4	21	0.278	1614.34
7	6	29	0.1469	1578.38
8	7	5	0.366	1240.40
9	8	23	0.2944	1104.08
10	9	7	0.2077	1009.48
11	11	30	2.489	413.63
12	5	3	0.2095	386.06
13	12	25	0.2757	244.46
14	3	5	1.421	202.03
15	7	11	0.5115	113.10
16	5	9	0.1228	95.78
17	11	27	0.675	-1.23×10^{-12}

Table 5: Iteracions de la Fase 2 del Simplex

Iteració	Variable Entrada	Variable Sortida	Theta	Valor Z
18	9	3	0.3515	456.40
19	10	9	0.6423	289.91
20	13	6	0.2135	232.67
21	3	1	0.3326	162.57
22	14	11	0.2809	92.21
23	9	4	0.9884	-13.12
24	15	5	254.7484	-122.66
25	16	2	52.8439	-149.00
26	1	8	0.1866	-156.89
27	17	13	31.7343	-168.57
28	8	1	0.2003	-172.13
29	19	8	95.5878	-247.07
30	2	9	0.0515	-263.17
31	13	16	0.3300	-269.04
32	5	17	1.1629	-269.89

Table 6: Solució òptima

Variables bàsiques	Valors variables bàsiques	Costs reduïts	Valor z
3	1.4417	53.57	
20	41.5183	66.77	
15	16.2597	187.61	
1	0.2017	39.46	
16	61.5069	3.30	-227.3611
13	3.2801	35.67	
8	3.2466	0.20	
7	0.2591	0.76	
11	4.0761	0.44	
10	0.5980	80.58	

3.1.3 Problema 3: No té solució factible

Table 7: Iteracions de la Fase 1 del Simplex

Iteració	Variable Entrada	Variable Sortida	Theta	Valor Z
1	1	24	0.0366	1316.22
2	2	28	0.0477	1299.96
3	3	1	0.1168	1280.03
4	4	2	0.0437	1277.68
5	6	3	0.1139	1272.06
6	8	4	0.1113	1249.78
7	1	6	0.0236	1248.65
8	11	1	0.0352	1241.83
9	13	11	0.0203	1238.15
10	16	26	165.8116	1072.34
11	17	27	216.3459	856.00
12	19	29	17.8084	838.19
13	20	30	48.2483	789.94
14	24	13	1.6792	787.17

3.1.4 Problema 4: No té solució acotada

Table 8: Iteracions de la Fase 1 del Simplex

Iteració	Variable Entrada	Variable Sortida	Theta	Valor Z
1	1	27	9.75	2398.25
2	3	29	1.264	1974.41
3	2	30	2.409	1726.55
4	4	31	1.862	1579.10
5	5	28	2.595	1162.85
6	6	32	3.563	437.34
7	9	26	1.167	223.91
8	8	25	0.436	190.15
9	11	8	0.081	151.83
10	7	33	2.543	15.87
11	8	5	0.300	10.22
12	12	8	0.121	4.47
13	13	34	0.111	2.98×10^{-13}

Table 9: Iteracions de la Fase 2 del Simplex

Iteració	**Variable Entrada**	**Variable Sortida**	**Theta**	**Valor Z**
14	5	11	2.748	-970.07
15	14	13	2.523	-973.83
16	15	7	17.486	-991.76
17	8	3	0.149	-995.26
18	11	6	0.014	-995.47
19	13	11	0.055	-995.77
20	3	14	0.596	-1006.24
21	16	2	104.836	-1043.33
22	6	8	0.140	-1043.54
23	17	5	114.905	-1058.98
24	8	3	0.960	-1063.59
25	18	8	70.703	-1087.00
26	11	4	0.286	-1091.56
27	19	11	14.408	-1101.92
28	3	12	2.524	-1121.63
29	14	3	2.840	-1158.45
30	20	6	221.926	-1433.10
31	2	14	2.458	-1493.55
32	21	13	426.099	-1933.77
33	23	2	201.765	-2161.45
34	24	9	677.40	-5343.80
35	11	1	99.143	-7931.43

3.2 Alumne nº 20

3.2.1 Problema 1

Table 10: Iteracions de la Fase 1 del Simplex

Iteració	Variable Entrada	Variable Sortida	Theta	Valor Z
1	1	27	1.270	2409.382
2	4	24	0.237	2286.030
3	2	1	0.617	2123.165
4	3	29	0.654	1807.380
5	1	2	1.422	1712.549
6	5	21	0.087	1668.951
7	2	23	0.073	1633.579
8	6	1	1.032	1243.468
9	7	28	0.518	1055.106
10	10	30	0.560	873.030
11	8	6	1.045	833.791
12	11	3	0.653	469.130
13	1	4	1.490	414.094
14	12	25	0.141	295.402
15	4	26	0.329	121.861
16	6	8	1.304	26.486
17	9	22	0.295	4.80×10^{-13}

Table 11: Iteracions de la Fase 2 del Simplex

Iteració	Variable Entrada	Variable Sortida	Theta	Valor Z
18	8	6	0.595	245.713
19	13	4	0.297	164.503
20	3	5	1.386	-93.063
21	6	12	0.366	-212.095
22	4	9	0.005	-212.299
23	20	4	1.830	-213.643
24	12	2	0.013	-215.182
25	14	12	0.047	-216.708
26	16	6	47.590	-226.969
27	15	14	16.260	-227.361

Table 12: Solució òptima

Variables bàsiques	Valors variables bàsiques	Costs reduïts	Valor z
3	1.4417	53.57	
20	41.5183	66.77	
15	16.2597	187.61	
1	0.2017	39.46	
16	61.5069	3.30	-227.3611
13	3.2801	35.67	
8	3.2466	0.20	
7	0.2591	0.76	
11	4.0761	0.44	
10	0.5980	80.58	

3.2.2 Problema 2

Table 13: Iteracions de la Fase 1 del Simplex

Iteració	Variable Entrada	Variable Sortida	Theta	Valor Z
1	1	27	3.707	2260.439
2	2	25	4.629	1461.366
3	4	26	0.253	1363.429
4	5	30	0.139	1317.178
5	3	27	1.931	1018.242
6	7	5	0.785	872.785
7	9	21	0.323	699.222
8	6	4	1.133	585.512
9	10	23	0.824	394.000
10	12	29	0.157	302.098
11	4	22	0.230	226.470
12	11	10	1.286	146.621
13	5	1	1.605	132.150
14	13	24	0.285	1.42×10^{-13}

Table 14: Iteracions de la Fase 2 del Simplex

Iteració	Variable Entrada	Variable Sortida	Theta	Valor Z
15	1	5	1.197	-358.769
16	10	11	1.504	-563.429
17	15	13	137.882	-596.654
18	17	15	196.314	-670.003
19	18	4	488.990	-745.291

Table 15: Solució òptima

Variables bàsiques	Valors variables bàsiques	Costs reduïts	Valor z
9	1.4298	183.03	
18	488.9899	121.15	
10	1.5986	137.45	
17	31.2922	178.54	
2	0.3563	218.90	-745.2914
6	2.1250	0.02	
3	5.7801	0.53	
1	2.8438	63.45	
12	1.1157	0.20	
7	0.4677	1.02	

3.2.3 Problema 3: No té solució factible

Table 16: Iteracions de la Fase 1 del Simplex

Iteració	Variable Entrada	Variable Sortida	Theta	Valor Z
1	1	30	0.5	1435.5
2	2	27	0.295	1341.99
3	3	1	0.170	1329.34
4	4	28	0.445	1321.95
5	7	4	0.151	1318.16
6	10	7	0.091	1308.36
7	16	26	251.551	1056.80
8	1	3	0.434	1035.25
9	17	10	24.069	999.25
10	19	29	77.069	922.18

3.2.4 Problema 4: No té solució acotada

Table 17: Iteracions de la Fase 1 del Simplex

Iteració	Variable Entrada	Variable Sortida	Theta	Valor Z
1	1	26	7.316	4036.14
2	2	29	6.725	1871.69
3	3	33	2.967	1218.76
4	4	25	2.377	636.65
5	5	27	0.292	633.05
6	6	5	0.254	547.70
7	8	28	0.328	484.68
8	10	34	1.284	239.04
9	9	8	1.463	144.76
10	5	31	0.010	142.22
11	11	5	0.015	141.24
12	8	4	0.805	69.43
13	12	30	0.072	33.30
14	4	1	3.260	31.58
15	5	32	1.996	-3.55×10^{-14}

Table 18: Iteracions de la Fase 2 del Simplex

Iteració	Variable Entrada	Variable Sortida	Theta	Valor Z
16	1	4	0.612	-856.11
17	13	12	0.508	-896.07
18	15	13	93.340	-913.14
19	16	9	152.668	-991.26
20	13	11	0.395	-995.99
21	17	2	27.849	-1003.57
22	20	15	238.678	-1081.60
23	18	5	131.914	-1117.36
24	21	3	102.882	-1188.80
25	15	8	68.663	-1213.70
26	5	18	0.900	-1263.97
27	3	13	1.201	-1265.38
28	11	6	0.571	-1278.03
29	13	3	0.257	-1283.89
30	19	11	31.959	-1300.57
31	22	17	155.631	-1308.84
32	23	1	485.902	-1735.95
33	17	13	5131.606	-8312.67
34	4	10	1.067	-8361.02
35	18	4	16.000	-8576.67

4 Conclusions

La realització d'aquesta pràctica ha suposat una aplicació dels coneixements rebuts a les classes teòriques sobre l'algorisme del Simplex Primal. A través del desenvolupament del programa d'aquest algorisme amb el llenguatge Python, hem aconseguit entendre amb profunditat el total funcionament del mètode del Simplex i hem comprovat la seva eficàcia en la resolució dels problemes lineals proposats.

La utilització del llenguatge Python ens ha premés la utilització de llibreries com NumPy, la qual ens ha facilitat la creació del programa i el funcionament de l'algorisme.

Aquesta pràctica ens ha ajudat a comprendre millor el funcionament del mètode de Simplex, com per exemple la utilització de les variables artificials en la Fase 1 per trobar una solució bàsica factible. Aquesta etapa és crucial per l'èxit de l'algorisme del Simplex ja que estableix les condicions perquè l'optimització de la funció objectiu sigui efectiva en la Fase 2.