https://wordlyknowledge.uz/index.php/gpble
Qabul qilingan sana: 07.12.2025
Nashr etilgan sana: 30.01.2026

Inomova Ozodaxon Shavkatjon qizi
Namangan Davlat Universiteti talabasi
E-mail: shavkatjonovna0217@gmail.com

TRANSPORT MASALASI YECHIMINING AVTOMATLASHTIRILGAN MODELI: PYTHON DASTURLASH TILI ASOSIDAGI YONDASHUV

Annotatsiya

Ushbu maqolada chiziqli dasturlashning muhim masalalaridan biri bo‘lgan transport masalasini yechishning avtomatlashtirilgan modeli ko‘rib chiqilgan. Tadqiqotda transport masalasi uchun boshlang‘ich ruxsat etilgan yechimlarni aniqlashda qo‘llaniladigan shimoliy-g‘arbiy burchak usuli va eng kichik elementlar usuli hamda optimal yechimni tekshirish va yaxshilashda ishlatiladigan potensiallar (MODI) usuli tahlil qilingan. Mazkur algoritmlar Python dasturlash tilida amalga oshirilib, foydalanuvchi tomonidan kiritiladigan ma’lumotlar asosida avtomatik hisoblash imkoniyati yaratilgan. Taklif etilgan model hisoblash jarayonini soddalashtiradi, vaqt sarfini kamaytiradi va transport masalalarini yechishda amaliy samaradorlikni oshiradi.

Kalit so‘zlar

Modellashtirish, transport masalasi, shimoliy-g’arbiy burchak usuli, eng kichik elementlar usuli, potensiallar usuli, python dasturlash tili, avtomatlashtirish.

AUTOMATED MODEL FOR SOLVING THE TRANSPORTATION PROBLEM: A PYTHON PROGRAMMING–BASED APPROACH

Abstract

This article considers an automated model for solving the transportation problem, which is one of the fundamental problems of linear programming. The study analyzes the North-West Corner method and the Least Cost method used to obtain an initial feasible solution, as well as the Potential (MODI) method applied

https://wordlyknowledge.uz/index.php/gpble
Qabul qilingan sana: 07.12.2025
Nashr etilgan sana: 30.01.2026

to verify and improve the optimality of the solution. These algorithms are implemented using the Python programming language, enabling automatic computation based on user-defined input data. The proposed automated model simplifies the calculation process, reduces computational time, and increases the practical efficiency of solving transportation problems.

Keywords

Modeling, transportation problem, North-West Corner method, Least Cost method, potential method (MODI), Python programming language, automation.

АВТОМАТИЗИРОВАННАЯ МОДЕЛЬ РЕШЕНИЯ ТРАНСПОРТНОЙ ЗАДАЧИ: ПОДХОД НА ОСНОВЕ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON

Аннотация

В данной статье рассматривается автоматизированная модель решения транспортной задачи, являющейся одной из ключевых задач линейного программирования. В работе проанализированы методы северо-западного угла и минимального элемента, используемые для построения начального допустимого плана, а также метод потенциалов (MODI), применяемый для проверки и улучшения оптимальности решения. Указанные алгоритмы реализованы на языке программирования Python, что позволяет выполнять автоматические вычисления на основе вводимых пользователем данных. Предложенная модель упрощает процесс вычислений, сокращает временные затраты и повышает практическую эффективность решения транспортных задач.

Ключевые слова

Моделирование, транспортная задача, метод северо-западного угла, метод минимального элемента, метод потенциалов, язык программирования Python, автоматизация.

Introduction

https://wordlyknowledge.uz/index.php/gpble
Qabul qilingan sana: 07.12.2025
Nashr etilgan sana: 30.01.2026

Zamonaviy iqtisodiyot, logistika va ishlab chiqarish tizimlarida resurslarni optimal taqsimlash masalasi muhim ahamiyat kasb etadi. Ushbu masalalarning matematik modeli sifatida transport masalasi keng qo‘llaniladi. Transport masalasi yordamida ishlab chiqarish markazlaridan iste’molchilarga mahsulotlarni minimal xarajat bilan yetkazib berish rejalashtiriladi. Shu sababli bu masala amaliy matematika, iqtisodiy modellashtirish va boshqa ko’plab fanlarining asosiy yo‘nalishlaridan biri hisoblanadi.

An’anaviy usullarda transport masalasini qo‘lda yechish katta vaqt va e’tibor talab qiladi, ayniqsa manbalar va iste’molchilar soni oshgan sari hisoblash jarayoni murakkablashadi. Shu bois transport masalasini yechishni avtomatlashtirish dolzarb vazifalardan biri hisoblanadi. Axborot texnologiyalarining rivojlanishi, xususan, dasturlash tillarining keng imkoniyatlari bu masalani samarali hal qilishga zamin yaratmoqda.

Mazkur maqolada transport masalasini yechish uchun klassik algoritmlar asosida avtomatlashtirilgan dasturiy model taklif etiladi. Bunda Python dasturlash tilidan foydalanilgan bo‘lib, u sodda sintaksisi va keng kutubxonalariga ega ekani bilan ajralib turadi. Tadqiqotning asosiy maqsadi transport masalasining boshlang‘ich va optimal yechimlarini topish algoritmlarini tahlil qilish hamda ularni avtomatlashtirish orqali amaliy samaradorlikni oshirishdan iborat.

Methods

Transport masalasining matematik modeli

Transport masalasida m ta ta’minot manbai va n ta iste’molchi mavjud bo‘lib, har bir manbaning zaxirasi va har bir iste’molchining talabi berilgan bo‘ladi. Shuningdek, har bir manbadan har bir iste’molchiga mahsulot tashish xarajati aniqlangan bo‘ladi. Masalaning maqsadi umumiy tashish xarajatlarini minimallashtirishdan iborat.

Mazkur masala chiziqli dasturlash masalasi bo‘lib, cheklovlar tenglamalar ko‘rinishida ifodalanadi. Transport masalasining yopiq turi uchun ta’minotlar yig‘indisi talablar yig‘indisiga teng bo‘lishi talab etiladi.

https://wordlyknowledge.uz/index.php/gpble
Qabul qilingan sana: 07.12.2025
Nashr etilgan sana: 30.01.2026

- Shimoliy-g‘arbiy burchak usuli

Shimoliy-g‘arbiy burchak usuli transport masalasi uchun boshlang‘ich ruxsat etilgan reja tuzishda qo‘llaniladigan eng sodda algoritmlardan biridir. Ushbu usulda xarajatlar jadvalining chap yuqori burchagidan boshlab, ketma-ket ravishda ta’minot va talablar qondiriladi. Usulning asosiy afzalligi hisoblashning soddaligidir, biroq u ko‘pincha optimal bo‘lmagan yechim beradi.

- Eng kichik elementlar usuli

Eng kichik elementlar usuli boshlang‘ich yechimni xarajatlarni hisobga olgan holda shakllantiradi. Bu usulda har bir qadamda eng kichik tashish xarajatiga ega bo‘lgan katak tanlanadi va unga maksimal mumkin bo‘lgan yuk joylashtiriladi. Natijada shimoliy-g‘arbiy burchak usuliga nisbatan yaxshiroq boshlang‘ich reja olinadi.

- Potensiallar usuli (MODI)

Potensiallar usuli transport masalasining optimal yechimga ega yoki ega emasligini aniqlash uchun qo‘llaniladi. Ushbu usulda qator va ustunlar uchun potensiallar aniqlanib, bo‘sh kataklar uchun baholash ko‘rsatkichlari hisoblanadi. Agar barcha baholash ko‘rsatkichlari musbat yoki nolga teng bo‘lsa, yechim optimal deb hisoblanadi.

- Python dasturlash tilida amalga oshirish

Mazkur tadqiqotda barcha algoritmlar Python dasturlash tilida yoziladi. Dastur foydalanuvchi tomonidan kiritiladigan xarajatlar jadvali, ta’minot va talablar asosida avtomatik ravishda hisob-kitoblarni amalga oshiradi. Bu yondashuv inson omilini kamaytiradi va xatoliklar ehtimolini pasaytiradi.

Results

Transport masalasi deganda, mahsulot yetkazib berish xarajatlarini minimallashtirish tushuniladi.

Quyidagi misolni ko’raylik.

Manba	B_1	B_2	B_3	Ta’minot
A_1	6	9	4	20
A_2	5	7	8	25

https://wordlyknowledge.uz/index.php/gpble
Qabul qilingan sana: 07.12.2025
Nashr etilgan sana: 30.01.2026

A_3	9	6	5	15
Talab	15	20	25	60

Jadval 1.1

1. Shimoliy-G'arbiy burchak usuli

Bu usuldagi talablarga birma-bir ta'minotdan yetkaziladi. Natijada jadval quyidagi ko'rinishga keladi:

	B_1	B_2	B_3
A_1	15	5	-
A_2	-	15	10
A_3	-	-	15

Jadval 1.2

Umumiy xarajat: 6·15+9·5+7·15+8·10+5·15=395

2. Eng kichik elementlar usuli

Bu usulda talablarga taminot eng kichik elementi bo'ylab yetkaziladi. Natijada jadval quyidagi ko'rinishga keladi:

	B_1	B_2	B_3
A_1	-	-	20
A_2	15	10	-
A_3	-	10	5

Jadval 1.3

Umumiy xarajat:4·20+5·15+5·5+6·10+7·10=310

3. Potensiallar usuli

Bu usulda yuqoridagi boshlang'ich yechimlarni optimalligini tekshiradi. Agar optimal bo'lmasa, potensiallar usuli orqali optimal yechim topiladi.

$c_{ij}-(u_i+v_j)\geq 0$ shu shart asosida hisoblashlar barchasi musbat chiqadi. Bundan ko'rinadiki, eng kichik elementlar usulidan olingan qiymat optimal hisoblanadi.

4. Pythonda dastur kodi

```
import numpy as np
print("=== TRANSPORT MASALASI (3 USUL + MODI) ===")
m = int(input("Qatorlar soni (ta'minot): "))
n = int(input("Ustunlar soni (talab): "))
C = np.zeros((m, n))
print("\nXarajatlar jadvali:")
```

https://wordlyknowledge.uz/index.php/gpble
Qabul qilingan sana: 07.12.2025
Nashr etilgan sana: 30.01.2026

```
for i in range(m):
    for j in range(n):
        C[i, j] = float(input(f"c[{i+1}][{j+1}] = "))
supply = [float(input(f"a[{i+1}] = ")) for i in range(m)]
demand = [float(input(f"b[{j+1}] = ")) for j in range(n)]
if sum(supply) != sum(demand):
    print(" Masala balanslanmagan")
    exit()
# shimoliy-g’arbiy burchak usuli
s = supply.copy()
d = demand.copy()
X_nw = np.zeros((m, n))
i = j = 0
while i < m and j < n:
    x = min(s[i], d[j])
    X_nw[i, j] = x
    s[i] -= x
    d[j] -= x
    if s[i] == 0: i += 1
    elif d[j] == 0: j += 1
cost_nw = np.sum(X_nw * C)
print("\n--- 1. Shimoliy-g‘arbiy burchak ---")
print("Reja:\n", X_nw)
print("Xarajat =", cost_nw)

# eng kichik elementlar usuli
s = supply.copy()
d = demand.copy()
X_lc = np.zeros((m, n))
```

https://wordlyknowledge.uz/index.php/gpble
Qabul qilingan sana: 07.12.2025
Nashr etilgan sana: 30.01.2026

```
while sum(s) > 0:
    min_cost = np.inf
    pos = (-1, -1)
    for i in range(m):
        for j in range(n):
            if s[i] > 0 and d[j] > 0 and C[i, j] < min_cost:
                min_cost = C[i, j]
                pos = (i, j)
    i, j = pos
    x = min(s[i], d[j])
    X_lc[i, j] = x
    s[i] -= x
    d[j] -= x
cost_lc = np.sum(X_lc * C)
print("\n--- 2. Eng kichik elementlar usuli ---")
print("Reja:\n", X_lc)
print("Xarajat =", cost_lc)
#  potensiallar usuli
def potentials(C, X):
    m, n = C.shape
    u = [None]*m
    v = [None]*n
    u[0] = 0
    basic = [(i,j) for i in range(m) for j in range(n) if X[i,j] > 0]
    while None in u or None in v:
        for i,j in basic:
            if u[i] is not None and v[j] is None:
                v[j] = C[i,j] - u[i]
            elif u[i] is None and v[j] is not None:
```

https://wordlyknowledge.uz/index.php/gpble
Qabul qilingan sana: 07.12.2025
Nashr etilgan sana: 30.01.2026

```
            u[i] = C[i,j] - v[j]
    return u, v
def entering_cell(C, X, u, v):
    best = 0
    pos = None
    for i in range(C.shape[0]):
        for j in range(C.shape[1]):
            if X[i,j] == 0:
                delta = C[i,j] - (u[i] + v[j])
                if delta < best:
                    best = delta
                    pos = (i, j)
    return pos
def find_cycle(X, start):
    m, n = X.shape
    def dfs(path, horizontal):
        i, j = path[-1]
        if horizontal:
            for jj in range(n):
                if jj != j and (X[i,jj] > 0 or (i,jj)==start):
                    nxt = (i,jj)
                    if nxt == start and len(path) >= 4:
                        return path+[nxt]
                    if nxt not in path:
                        r = dfs(path+[nxt], not horizontal)
                        if r: return r
        else:
            for ii in range(m):
                if ii != i and (X[ii,j] > 0 or (ii,j)==start):
```

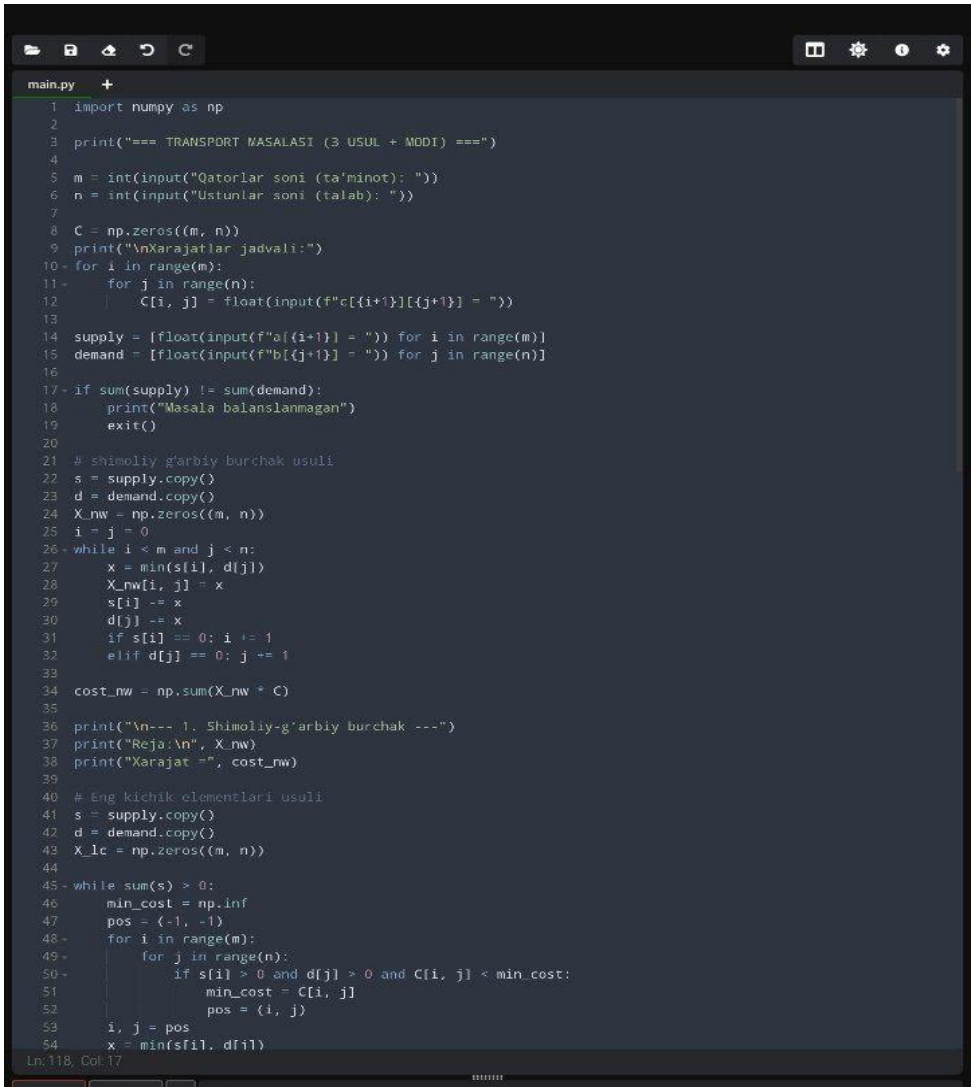
https://wordlyknowledge.uz/index.php/gpble
Qabul qilingan sana: 07.12.2025
Nashr etilgan sana: 30.01.2026

```
                nxt = (ii,j)
                if nxt == start and len(path) >= 4:
                    return path+[nxt]
                if nxt not in path:
                    r = dfs(path+[nxt], not horizontal)
                    if r: return r
        return None
    return dfs([start], True)

#  modi yaxshilash
X = X_lc.copy()
while True:
    u, v = potentials(C, X)
    enter = entering_cell(C, X, u, v)
    if enter is None:
        break
    cycle = find_cycle(X, enter)
    minus = cycle[1::2]
    theta = min(X[i,j] for i,j in minus)
    for k,(i,j) in enumerate(cycle[:-1]):
        if k % 2 == 0: X[i,j] += theta
        else: X[i,j] -= theta
print("\n--- 3. MODI orqali OPTIMAL YECHIM ---")
print("Optimal reja:\n", X)
print("Optimal xarajat =", np.sum(X * C))
```

Bu kod orqali yuqoridagi misol uchun optimal yechimni topamiz.

https://wordlyknowledge.uz/index.php/gpble
Qabul qilingan sana: 07.12.2025
Nashr etilgan sana: 30.01.2026



```python
import numpy as np

print("=== TRANSPORT MASALASI (3 USUL + MODI) ===")

m = int(input("Qatorlar soni (ta'minot): "))
n = int(input("Ustunlar soni (talab): "))

C = np.zeros((m, n))
print("\nXarajatlar jadvali:")
for i in range(m):
    for j in range(n):
        C[i, j] = float(input(f"c[{i+1}][{j+1}] = "))

supply = [float(input(f"a[{i+1}] = ")) for i in range(m)]
demand = [float(input(f"b[{j+1}] = ")) for j in range(n)]

if sum(supply) != sum(demand):
    print("Masala balanslanmagan")
    exit()

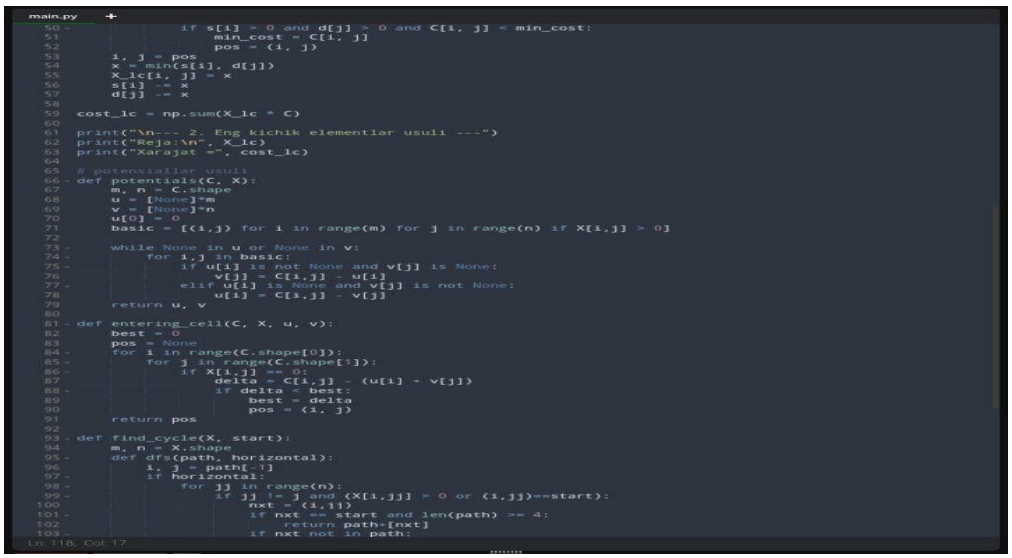
# shimoliy g'arbiy burchak usuli
s = supply.copy()
d = demand.copy()
X_nw = np.zeros((m, n))
i = j = 0
while i < m and j < n:
    x = min(s[i], d[j])
    X_nw[i, j] = x
    s[i] -= x
    d[j] -= x
    if s[i] == 0: i += 1
    elif d[j] == 0: j += 1

cost_nw = np.sum(X_nw * C)

print("\n--- 1. Shimoliy-g'arbiy burchak ---")
print("Reja:\n", X_nw)
print("Xarajat =", cost_nw)

# Eng kichik elementlari usuli
s = supply.copy()
d = demand.copy()
X_lc = np.zeros((m, n))

while sum(s) > 0:
    min_cost = np.inf
    pos = (-1, -1)
    for i in range(m):
        for j in range(n):
            if s[i] > 0 and d[j] > 0 and C[i, j] < min_cost:
                min_cost = C[i, j]
                pos = (i, j)
    i, j = pos
    x = min(s[i], d[i])
```



```python
            if s[i] > 0 and d[j] > 0 and C[i, j] < min_cost:
                min_cost = C[i, j]
                pos = (i, j)
    i, j = pos
    x = min(s[i], d[j])
    X_lc[i, j] = x
    s[i] -= x
    d[j] -= x

cost_lc = np.sum(X_lc * C)

print("\n--- 2. Eng kichik elementlar usuli ---")
print("Reja:\n", X_lc)
print("Xarajat =", cost_lc)

# potensiallar usuli
def potentials(C, X):
    m, n = C.shape
    u = [None]*m
    v = [None]*n
    u[0] = 0
    basic = [(i,j) for i in range(m) for j in range(n) if X[i,j] > 0]

    while None in u or None in v:
        for i,j in basic:
            if u[i] is not None and v[j] is None:
                v[j] = C[i,j] - u[i]
            elif u[i] is None and v[j] is not None:
                u[i] = C[i,j] - v[j]
    return u, v

def entering_cell(C, X, u, v):
    best = 0
    pos = None
    for i in range(C.shape[0]):
        for j in range(C.shape[1]):
            if X[i,j] == 0:
                delta = C[i,j] - (u[i] + v[j])
                if delta < best:
                    best = delta
                    pos = (i, j)
    return pos

def find_cycle(X, start):
    m, n = X.shape
    def dfs(path, horizontal):
        i, j = path[-1]
        if horizontal:
            for jj in range(n):
                if jj != j and (X[i,jj] > 0 or (i,jj)==start):
                    nxt = (i,jj)
                    if nxt == start and len(path) >= 4:
                        return path+[nxt]
                    if nxt not in path:
```

https://wordlyknowledge.uz/index.php/gpble
Qabul qilingan sana: 07.12.2025
Nashr etilgan sana: 30.01.2026

```
main.py +
    for i in range(C.shape[0]):
        for j in range(C.shape[1]):
            if X[i,j] == 0:
                delta = C[i,j] - (u[i] + v[j])
                if delta < best:
                    best = delta
                    pos = (i, j)
    return pos

def find_cycle(X, start):
    m, n = X.shape
    def dfs(path, horizontal):
        i, j = path[-1]
        if horizontal:
            for jj in range(n):
                if jj != j and (X[i,jj] > 0 or (i,jj)==start):
                    nxt = (i,jj)
                    if nxt == start and len(path) >= 4:
                        return path+[nxt]
                    if nxt not in path:
                        r = dfs(path+[nxt], not horizontal)
                        if r: return r
        else:
            for ii in range(m):
                if ii != i and (X[ii,j] > 0 or (ii,j)==start):
                    nxt = (ii,j)
                    if nxt == start and len(path) >= 4:
                        return path+[nxt]
                    if nxt not in path:
                        r = dfs(path+[nxt], not horizontal)
                        if r: return r
        return None
    return dfs([start], True)

# modi yxshilash
X = X_lc.copy()

while True:
    u, v = potentials(C, X)
    enter = entering_cell(C, X, u, v)
    if enter is None:
        break

    cycle = find_cycle(X, enter)
    minus = cycle[1::2]
    theta = min(X[i,j] for i,j in minus)

    for k,(i,j) in enumerate(cycle[:-1]):
        if k % 2 == 0: X[i,j] += theta
        else: X[i,j] -= theta

print("\n--- 3. MODI orqali OPTIMAL YECHIM ---")
print("Optimal reja:\n", X)
print("Optimal xarajat =", np.sum(X * C))
Ln: 118, Col: 17
```

```
Run  Share  $  Command Line Arguments

c[1][2] = 9
c[1][3] = 4
c[2][1] = 5
c[2][2] = 7
c[2][3] = 8
c[3][1] = 9
c[3][2] = 6
c[3][3] = 5
a[1] = 20
a[2] = 25
a[3] = 15
b[1] = 15
b[2] = 20
b[3] = 25

--- 1. Shimoliy-g'arbiy burchak ---
Reja:
 [[15.  5.  0.]
 [ 0. 15. 10.]
 [ 0.  0. 15.]]
Xarajat = 395.0

--- 2. Eng kichik elementlar usuli ---
Reja:
 [[ 0.  0. 20.]
 [15. 10.  0.]
 [ 0. 10.  5.]]
Xarajat = 310.0

--- 3. MODI orqali OPTIMAL YECHIM ---
Optimal reja:
 [[ 0.  0. 20.]
 [15. 10.  0.]
 [ 0. 10.  5.]]
Optimal xarajat = 310.0

** Process exited - Return Code: 0 **
```

https://wordlyknowledge.uz/index.php/gpble
Qabul qilingan sana: 07.12.2025
Nashr etilgan sana: 30.01.2026

Ko’rinib turibdiki, kod natijasi va sonli yechim javoblari bir xil.

Ishlab chiqilgan dasturiy model yordamida turli o‘lchamdagi transport masalalari yechildi. Tajribalar shuni ko‘rsatdiki, shimoliy-g‘arbiy burchak usuli tezkor natija bersa-da, umumiy xarajatlar yuqori bo‘lishi mumkin. Eng kichik elementlar usuli esa boshlang‘ich rejaning sifatini sezilarli darajada yaxshilaydi.

Potensiallar usulini qo‘llash orqali optimal bo‘lmagan rejalar aniqlanib, ularni yaxshilash imkoniyati yaratildi. Natijada umumiy tashish xarajatlari kamaytirildi va optimal reja topildi. Avtomatlashtirilgan yondashuv hisoblash vaqtini bir necha barobarga qisqartirdi.

Discussion

Olingan natijalar shuni ko‘rsatadiki, transport masalasini yechishda boshlang‘ich reja tanlash muhim ahamiyatga ega. Shimoliy-g‘arbiy burchak usuli o‘quv jarayonida qulay bo‘lsa-da, amaliy masalalarda eng kichik elementlar usulidan foydalanish maqsadga muvofiqdir. Potensiallar usuli esa optimal yechimni aniqlashda ishonchli vosita hisoblanadi.

Python dasturlash tilida yaratilgan avtomatlashtirilgan model logistika, ishlab chiqarish va iqtisodiy rejalashtirish masalalarida samarali qo‘llanishi mumkin. Ushbu modelni yanada rivojlantirib, katta hajmdagi ma’lumotlar uchun moslashtirish mumkin.

Conclusion

Mazkur maqolada transport masalasini yechish uchun avtomatlashtirilgan dasturiy model ishlab chiqildi. Shimoliy-g‘arbiy burchak, eng kichik elementlar va potensiallar usullari tahlil qilinib, ularning Python dasturlash tilida amalga oshirilishi ko‘rsatib berildi. Taklif etilgan yondashuv transport masalalarini tez, aniq va samarali yechish imkonini beradi. Kelgusida ushbu modelni kengaytirish va real amaliy tizimlarga joriy etish maqsadga muvofiqdir.

References

1. Qodirov, A., & Rustamov, S. (2018). Operatsion tadqiqot asoslari. Toshkent: O‘zbekiston Milliy Universiteti Nashriyoti.

https://wordlyknowledge.uz/index.php/gpble
Qabul qilingan sana: 07.12.2025
Nashr etilgan sana: 30.01.2026

2. Islomov, B. (2020). Matematik modellashtirish va optimal reja tuzish. Toshkent: Fan va Texnologiya.

3. Abdullayev, J. (2019). Logistika va transport masalalari: nazariya va amaliyot. Toshkent: Iqtisodiyot va boshqaruv.

4. Karimov, D., & Tursunov, N. (2021). Python dasturlash tilida iqtisodiy masalalarni modellashtirish. Toshkent: Axborot texnologiyalari nashriyoti.

5. Sobirov, M. (2017). Chiziqli dasturlash va transport masalasi. Toshkent: O‘zbekiston Davlat Texnika Universiteti Nashriyoti.

6. Shodiev, F. (2022). Amaliy matematika va kompyuter yondashuvlari. Toshkent: Ilm-fan va Ta’lim.

7. https://www.online-python.com