GENETIK ALGORITIMLAR

Abdusattorova M.V.[1], Ma'murova M.R.[2] , Ismoilov A.I.[3]

[1] Farg'ona davlat unversiteti 3-kurs talabasi

mohigulabdusattorova50@gmail.com

[2]Farg'ona davlat universiteti 3-kurs talabasi

mamurovmansurbek05@gmail.com

[3]Farg'ona davlatunversiteti

amaliy matematika va informatika kafedrasi katta o'qituvchisi

Annotasiya. Genetik algoritmlarning matematik va algoritmik asoslari, zamonaviy informatika hamda amaliy matematikadagi o'rni yoritilgan. Evolyutsion hisoblashning asosiy operatorlari – seleksiya, krossingover va mutatsiya – tahlil qilinib, ularning optimallashtiruv samaradorligiga ta'siri ko'rsatiladi. An'anaviy optimallashtirish usullari bilan qiyosiy tahlil qilingan, real muhandislik va iqtisodiy masalalarda qo'llash imkoniyatlari muhokama qilingan.

Kalit so'zlar. genetik algoritm, evolyutsion hisoblash, moslashuvchanlik funksiyasi, seleksiya, krossingover, mutatsiya, kombinatorial optimallashtiruv, metaevristika, sun'iy intellekt.

Kirish.

Insoniyat doim eng yaxshi yechimni topishni xohlaydi. Matematika va informatika rivojlanishi davomida optimallashtiruv masalalariga qiziqish hech qachon kamaymaydi. Real hayotdagi muammolarning aksariyati – xoh bu korxona resurslarini taqsimlash bo'lsin, xoh transport marshrutini qisqartirish qandaydir maqsad funksiyasining eng kichik yoki eng katta qiymatini topish masalasiga olib keladi. Klassik yondashuvlar, jumladan, simpleks metod, gradient tushish yoki Nyuton usullari, aniq shartlar ostida juda samarali ishlaydi. Funksiya uzluksiz bo'lishi, differensiallanishi, cheklovlar chiziqli bo'lishi kerak. Amaliyotda esa bu shartlar ko'pincha bajarilmaydi murakkab fizik jarayonlar, diskret o'zgaruvchilar, shovqinli maqsad funksiyalari klassik usullarni ojiz qoldiradi.

Aynan shu bo‘shliqni to‘ldirish uchun tabiatning o‘zidan ilhom olingan hisoblash usullari metaevristikalar paydo bo‘ldi. Ularning eng yorqin vakillaridan biri genetik algoritmlardir. Bu algoritmlar Darvin evolyutsiyasining oddiy, ammo dahshat darajada kuchli g‘oyasiga tayanadi: yashash uchun kurashda kuchli individlar omon qoladi, zaiflar yo‘qoladi va har bir yangi avlod oldingisidan biroz yaxshiroq bo‘ladi. Bu jarayonni matematik tilga ko‘chirish natijasida, murakkab masalalarni yechish uchun nihoyatda moslashuvchan va kuchli vosita qo‘lga kiritildi.

Genetik algoritmlar bo‘yicha tizimli nazariyani birinchi bo‘lib Jon Holland 1975 yilda shakllantirgan. O‘tgan yarim asr ichida bu soha ulkan yo‘lni bosib o‘tdi. Bugungi kunda muhandislik konstruksiyalaridan tortib sun’iy intellekt modellarigacha bo‘lgan keng doirada genetik algoritmlarsiz tasavvur qilib bo‘lmaydi. Genetik algoritmning asosiy tushunchalari: Genetik algoritmni tushunish uchun dastlab uning biologik ildizlariga qisqacha nazar tashlash foydali. Tabiatda populyatsiya bir turga mansub individlar to‘plami. Har bir individ o‘z genetik kodiga ega. Atrof-muhitga ko‘proq moslashgan individlar ko‘proq nasl qoldiradi, moslasha olmaganlar esa yo‘q bo‘lib ketadi. Natijada populyatsiya avloddan-avlodga o‘sha muhitga tobora yaxshiroq moslashib boradi.

Matematik nuqtai nazardan, bizda qandaydir $f(x)$ funksiyani minimallashtirish (yoki maksimallashtirish) masalasi bor. An’anaviy usullar bitta yechim olib, uni asta-sekin yaxshilashga urinsa, genetik algoritm bir vaqtning o‘zida yechimlar to‘plami populyatsiya bilan ishlaydi. Ana shu parallel qidiruv xususiyati algoritmga bir vaqtning o‘zida qidiruv fazosining turli qismlarini tadqiq qilish imkonini beradi va mahalliy minimumga tushib qolish xavfini keskin kamaytiradi. **Xromosoma** bitta yechimni kodlash shakli. Klassik variantda u ikkilik satr, ya’ni $x \in \{0,1\}n$ ko‘rinishida ifodalanadi. Masalan, $n = 8$ bo‘lsa, “10110100” xromosomasi qidiruv fazosidagi aniq bir nuqtaga ishora qiladi. Agar masala uzluksiz o‘zgaruvchilar bilan ishlasa, xromosoma $x \in R^n$ ko‘rinishiga ega bo‘ladi, bunda har bir gen bitta haqiqiy sonni anglatadi. **Moslashuvchanlik funksiyasi** har bir xromosomaning “yaxshiligini” baholovchi mezon. Agar biz minimallashtirish masalasini yechayotgan bo‘lsak, moslashuvchanlik sifatida to‘g‘ridan-to‘g‘ri maqsad funksiyaning

o‘zini yoki unga bog‘liq monoton almashtirishni olish mumkin. Maksimallashtirish masalasini minimallashtirish ko‘rinishiga o‘tkazish uchun esa,

$$F(x) = -f(x)$$

almashtirish kifoya. Moslashuvchanlik funksiyasini to‘g‘ri tanlash butun algoritm samaradorligini belgilaydi , bu funksiya masalaning fizik mohiyatini qanchalik aniq aks ettirsa, algoritm shunchalik tez va ishonchli yechim topadi.

Asosiy operatorlar va ularning mexanizmi. Har qanday genetik algoritmning yuragini uchta operator tashkil qiladi. Seleksiya, krossingover va mutatsiya. Ularning to‘g‘ri nisbati va parametrlari qidiruv jarayonining muvaffaqiyatini hal qiladi.

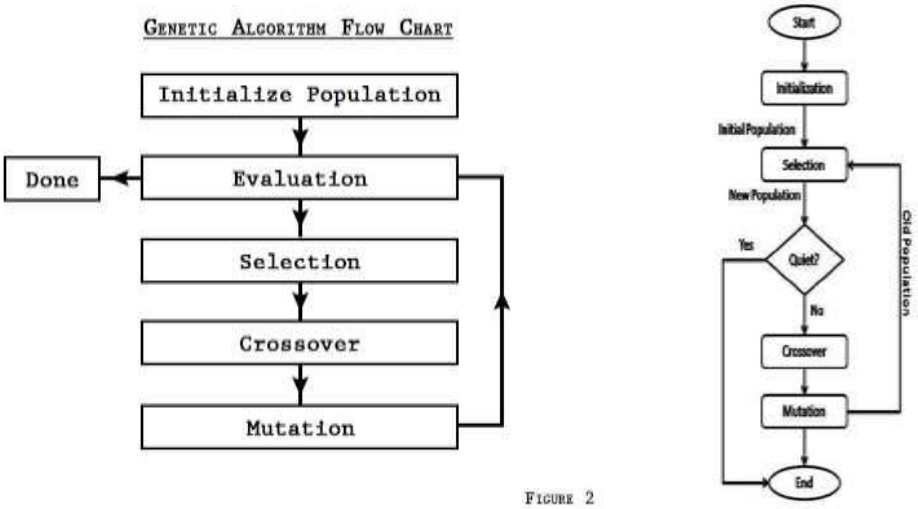


Figure 2

Seleksiya keyingi avlod uchun ota-ona yechimlarni tanlash jarayoni. Eng ko'p ishlatiladigan usullar,

Ruletka seleksiyasi har bir individuaning tanlanish ehtimolligi uning moslashuvchanlik qiymatiga proporsional:

$$P(x_i) = \frac{f(x_j)}{\sum f(x_j)}$$

Oddiy, lekin yuqori moslashuvchanlikli individlar populyatsiyani tez egallash xavfi mavjud.

Turnir seleksiyasi tasodifiy tanlangan k ta individualdan eng yaxshisi tanlanadi. k = 2 ikkilik turnir amalda keng tarqalgan. Bu usul seleksiya bosimini boshqarishda qulay hisoblanadi.

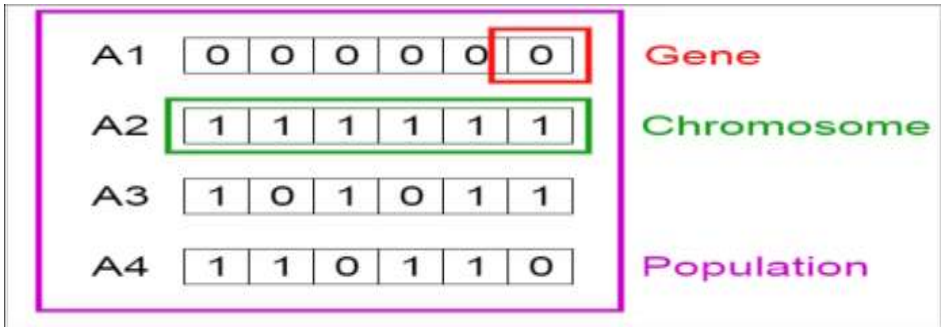
Rang bo‘yicha seleksiya individlar moslashuvchanlik qiymatiga qarab tartiblanadilar va rang asosida ehtimollik tayinlanadi. Moslashuvchanlik qiymatlarining katta farqlarida samarali ishlaydi.

Krossingover bu ikkita ota-ona xromosomasidan yangi avlodni yaratuvchi operator. Biologiyadagi meoz bo‘linishiga o‘xshab, ota-onalar genetik materialining bir qismini almashtiradi. Bir nuqtali krossingoverda tasodifiy pozitsiya tanlanadi va shu nuqtadan keyingi barcha genlar almashinadi.

Ota-ona 1: **1 0 1 1** | 0 1 0 0

Ota-ona 2: **0 1 0 0** | 1 1 0 1

Avlod: **1 0 1 1** 1 1 0 1



Ikki nuqtali krossingover esa yanada keng qidiruv imkonini beradi va amaliyotda ko‘proq tavsiya etiladi. Krossingover ehtimolligi odatda 0,6 dan 0,9 gacha tanlanadi. Juda past qiymat populyatsiyaning yetarlicha yangilanishiga yo‘l qo‘ymaydi, juda yuqori qiymat esa yaxshi yechimlarning “buzilishi”ga sabab bo‘ladi. **Mutatsiya** bu populyatsiyaga kichik tasodifiy o‘zgarishlar kiritish. Ikkilik kodlashda bu biror bitni teskarisiga aylantirishdir. Mutatsiya ehtimolligi juda kichik odatda 0,001 dan 0,05 oralig‘ida tanlanadi. Mutatsiyaning vazifasi qidiruv fazosining hali o‘rganilmagan hududlariga chiqish imkonini yaratishdir. Agar mutatsiya darajasi keragidan ortiq bo‘lsa, algoritm oddiy tasodifiy qidiruvga aylanib qoladi va evolyutsion mexanizm o‘z kuchini yo‘qotadi. Aksincha, u juda kichik bo‘lsa, populyatsiya tezda bir xillashadi va mahalliy optimumdan chiqa olmay qoladi. Mana shu nozik muvozanatni topish har bir aniq masala uchun algoritmni sozlashning eng muhim qismidir. Algoritmning umumiy ishlash tartibi. Genetik algoritmning to‘liq sikli quyidagicha amalga oshiriladi. Dastlab, populyatsiya hajmi belgilanadi. Odatda 50 dan 300 gacha individ. Bu individlar qidiruv fazosida

tasodifiy tarqatiladi. Har bir individ uchun moslashuvchanlik funksiyasining qiymati hisoblanadi. Keyin seleksiya operatori yordamida ota-onalar juftligi tanlanadi. Krossingover va mutatsiya operatorlari qo‘llanib, yangi avlod yaratiladi. Ushbu yangi avlod eski populyatsiyani almashtiradi va jarayon takrorlanadi. Algoritmning to‘xtash sharti turlicha bo‘lishi mumkin: avlodlar sonining oldindan berilgan chegarasi, moslashuvchanlik funksiyasining maqsad qiymatga erishganligi yoki so‘nggi bir necha avlod davomida sezilarli yaxshilanish kuzatilmaganligi. Ayniqsa oxirgi variant stagnatsiyani aniqlash amaliyotda eng ko‘p ishlatiladi, chunki u algoritmni ortiqcha hisoblashdan saqlaydi.

Elitizm strategiyasi ham alohida e‘tiborga loyiq. Bunda har bir yangi avlodga eski populyatsiyaning eng yaxshi 2–5 foizi hech qanday o‘zgarishsiz o‘tadi. Bu oddiy qoida algoritmga “topgan eng yaxshi yechimini yo‘qotmaslik” kafolatini beradi. Elitizmsiz ishlaydigan algoritmlar, empirik kuzatuvlarga ko‘ra, natijaga erishish uchun o‘rtacha 15–30 foiz ko‘proq avlod talab qiladi. Qo‘llanilish sohalari genetik algoritmlarning amaliy qo‘llanilish doirasi kengdir. Quyida eng muhim yo‘nalishlar haqida to‘xtalamiz.

Kombinatorial optimallashtiruv. Bu sohaning eng klassik namunasi – sayohatchi sotuvchi masalasidir. Berilgan n ta shaharni bir marta borib, eng qisqa yo‘l bilan boshlang‘ich nuqtaga qaytish. $n = 50$ bo‘lganda ham to‘liq saralash usuli amalda imkonsiz bu masala NP-qiyin sinfiga kiradi. Genetik algoritm bu yerda maxsus krossingover operatorlari – tartibli krossingover (OX) yoki qisman akslantirilgan krossingover (PMX) yordamida optimal yechimga juda yaqin natija bera oladi, ko‘p hollarda optimaldan 3–5 foiz farq bilan.

Muhandislik loyihalash. Konstruksiyalarning geometrik parametrlarini tanlash, elektron sxemalar sintezi, mexanizm va agregatlar optimallashtirish bularning barchasida maqsad funksiyasi murakkab fizik hisob-kitoblarga asoslanadi va gradientni aniqlash imkoni bo‘lmaydi. Aynan shunday vaziyatlarda genetik algoritmlar o‘zining universalligi tufayli eng ishonchli vositalardan biri bo‘lib qoladi. Ko‘plab muhandislik tadqiqotlarida genetik algoritm boshqa lokal usullar bilan birga gibrid shaklda qo‘llanilib, ancha yuqori aniqlikka erishilmoqda.

Mashinali o‘rganish. Neyron tarmoqlarining giperparametrlarini yashirin qatlamlar soni, neyronlar zichligi, aktivatsiya funksiyasi turi, o‘qitish tezligi qo‘lda tanlash juda katta tajriba talab etadi. Genetik algoritm bu jarayonni avtomatlashtirish imkonini beradi. Zamonaviy AutoML tizimlarining asosida ham ko‘p hollarda evolyutsion qidiruv yotadi.

Iqtisodiy modellashtirish. Investitsiya portfelini shakllantirish, yetkazib berish zanjirlarini rejalashtirish, moliyaviy risklarni baholash kabi masalalarda genetik algoritm an’anaviy usullarga nisbatan nolinear va murakkab cheklovli sharoitlarda ancha ishonchli natija ko‘rsatadi.

Afzalliklari, kamchiliklari va boshqa usullar bilan, har qanday optimallashtirish usulini ob’ektiv baholash uchun uning kuchli va zaif tomonlarini bir xilda ko‘rib chiqish zarur. Genetik algoritmning asosiy afzalligi yuqori darajadagi moslashuvchanlik. U maqsad funksiyasidan differensiallanish, uzluksizlik, qavariqlik kabi xususiyatlarni talab qilmaydi. Parallel qidiruv tufayli mahalliy minimumlardan nisbatan erkin o‘tish imkoniyati mavjud. Ikkilik, haqiqiy sonli va kombinatorial kodlashlarga deyarli teng darajada moslasha oladi. Biroq kamchiliklar ham yetarli. Birinchidan, moslashuvchanlik funksiyasi ko‘p marta hisoblanadi bu esa hisoblash resurslarini talab qiladi. Ikkinchidan, natija deterministik emas, ya’ni bir xil boshlang‘ich parametrlarda algoritmning ikkita alohida ishga tushirilishi turli natijalar berishi mumkin. Uchinchidan, populyatsiya hajmi, mutatsiya ehtimolligi kabi parametrlarni to‘g‘ri sozlash muayyan tajribani talab qiladi. Taqqoslash uchun boshqa metaevristikalarni eslab o‘tish foydali. Zarralar to‘dasi optimizatsiyasi (PSO) uzluksiz masalalarda ko‘pincha genetik algoritmdan tezroq yaqinlashadi, chunki har bir zarraning tezlik vektori qidiruvni yo‘naltiradi. Simulyatsiyalangan temperlash esa parametrlash jihatidan soddaroq, lekin u parallel qidiruv imkoniga ega emas. Mashhur “No Free Lunch” teoremasi barcha optimallashtiruv algoritmlari o‘rtacha hisobda teng samaradorlikka egaligini qat’iy isbotlab beradi. Demak, yuqori natijaga erishishning yagona yo‘li hal qilinayotgan masalaning ichki tabiatini chuqur anglash va shunga mos strategiya tanlashdan iborat. Zamonaviy rivojlanish yo‘nalishlari. Genetik algoritmlar hozirgi kunda ham jadal rivojlanishda davom etmoqda. Quyidagi yo‘nalishlar alohida diqqatga sazovor.

Parallel genetik algoritmlar. Hisoblash yuki bir necha protsessor yoki GPU larga taqsimlanadi. Orol modeli deb ataluvchi yondashuvda bir necha mustaqil populyatsiya parallel rivojlanadi va vaqti-vaqti bilan ular o‘rtasida individlar almashinuvi sodir bo‘ladi. Bu xilma-xillikni saqlab qolishga va umumiy konvergensiyani tezlashtirishga xizmat qiladi.

Moslashuvchan parametrli algoritmlar. An’anaviy yondashuvda mutatsiya ehtimolligi va krossingover koeffitsienti butun ishlash davomida qat’iy belgilangan bo‘ladi. Yangi avlod algoritmlarida esa bu parametrlar qidiruv jarayonining holatiga qarab dinamik o‘zgartiriladi, agar populyatsiya juda tez bir xillashib qolsa, mutatsiya oshiriladi; agar qidiruv barqaror yaxshilanayotgan bo‘lsa, parametrlar kamaytiriladi.

Genetik algoritmni lokal optimallashtirish usullari bilan birlashtirish memetik algoritmlar ko‘plab sohalarda sof genetik algoritmdan sezilarli ustunlik ko‘rsatmoqda. Odatda genetik algoritm avvalo global optimalga yaqin hududni aniqlab beradi, so‘ngra gradient yoki simpleks kabi lokal usul bu yechimni tezda aniqlashtiradi.

```
import random
from datetime import datetime, timedelta
print("\n===== AVTOMATIK DARS JADVALI =====\n")
n = int(input("Nechta fan kiritasiz? : "))
courses = []
for i in range(n):
    print(f"\n{i+1}-fan")
    subject = input("Fan nomi: ")
    teacher = input("Ustoz ismi: ")
    courses.append((subject, teacher))
days = [
    "Dushanba",
    "Seshanba",
    "Chorshanba",
    "Payshanba",
```

```
    "Juma"
]
rooms = [
    "1-XONA",
    "2-XONA",
    "3-XONA",
    "4-XONA"
]
def generate_times():
    start_time = datetime.strptime("08:00", "%H:%M")
    lesson_times = []
    current = start_time
    total_lessons = 5
    for i in range(total_lessons):
        lesson_start = current
        lesson_end = lesson_start + timedelta(minutes=45)
        lesson_times.append(
            f"{lesson_start.strftime('%H:%M')} - "
            f"{lesson_end.strftime('%H:%M')}"
        )
        if i == 2:
            current = lesson_end + timedelta(minutes=10)
        else:
            current = lesson_end + timedelta(minutes=5)
    return lesson_times
times = generate_times()
def create_individual():
    individual = []
    teacher_schedule = set()
```

```
    room_schedule = set()
    for subject, teacher in courses:
        while True:
            day = random.choice(days)
            time = random.choice(times)
            room = random.choice(rooms)
            teacher_key = (teacher, day, time)
            room_key = (room, day, time)
            if (
                teacher_key not in teacher_schedule
                and room_key not in room_schedule
            ):
                teacher_schedule.add(teacher_key)
                room_schedule.add(room_key)
                individual.append(
                    (subject, teacher, day, time, room)
                )
                break
    return individual
def fitness(individual):
    conflicts = 0
    teacher_schedule = set()
    room_schedule = set()
    for subject, teacher, day, time, room in individual:
        teacher_key = (teacher, day, time)
        if teacher_key in teacher_schedule:
            conflicts += 1
        else:
            teacher_schedule.add(teacher_key)
```

```
            room_key = (room, day, time)
            if room_key in room_schedule:
                conflicts += 1
            else:
                room_schedule.add(room_key)
        return max(0, 100 - conflicts * 25)
    def create_population(size):
        return [
            create_individual()
            for _ in range(size)
        ]
    def selection(population):
        population.sort(
            key=fitness,
            reverse=True
        )
        return population[:len(population)//2]
    def crossover(parents, size):
        children = []
        while len(children) < size:
            p1 = random.choice(parents)
            p2 = random.choice(parents)
            if n == 1:
                child = p1[:]
            else:
                point = random.randint(1, n - 1)
                child = (
                    p1[:point] +
                    p2[point:]
```

```
            )
        children.append(child)
    return children
def mutation(population, rate=0.1):
    for individual in population:
        for i in range(len(individual)):
            if random.random() < rate:
                subject, teacher, _, _, _ = individual[i]
                individual[i] = (
                    subject,
                    teacher,
                    random.choice(days),
                    random.choice(times),
                    random.choice(rooms)
                )
    return population
def genetic_algorithm(
    generations=200,
    pop_size=100
):
    population = create_population(pop_size)
    best_overall = None
    for gen in range(generations):
        population.sort(
            key=fitness,
            reverse=True
        )
        best = population[0]
        if (
```

```
            best_overall is None
            or
            fitness(best) > fitness(best_overall)
        ):
            best_overall = best
        print(
            f"Avlod {gen+1:<3} "
            f"| Fitness = {fitness(best)}"
        )
        if fitness(best) == 100:
            print("\nIDEAL JADVAL TOPILDI!\n")
            return best
        parents = selection(population)
        children = crossover(
            parents,
            pop_size - len(parents)
        )
        population = parents + children
        population = mutation(population)
    return best_overall
best_schedule = genetic_algorithm()
best_schedule.sort(
    key=lambda x: (
        days.index(x[2]),
        times.index(x[3])
    )
)
print("\n==================================================
====================")
```

```
    print("                    ENG YAXSHI JADVAL")
    print("=========================================================
==================\n")
    print(
       f"{'KUN':<15}"
       f"{'VAQT':<20}"
       f"{'FAN':<20}"
       f"{'USTOZ':<20}"
       f"{'XONA'}"
    )
    print("-" * 90)
    for subject, teacher, day, time, room in best_schedule:
       print(
          f"{day:<15}"
          f"{time:<20}"
          f"{subject:<20}"
          f"{teacher:<20}"
          f"{room}"
       )
    print("\n=========================================================
====================")
    print(f"Jadval sifati: {fitness(best_schedule)}%")
    print("=========================================================
==================")
```

Ushbu dastur genetik algoritm yordamida avtomatik dars jadvalini yaratish uchun ishlab chiqildi. Dastur Python dasturlash tilida yozilgan bo‘lib, fanlar, ustozlar, vaqt va auditoriyalarni hisobga olib optimal jadval hosil qiladi. Dastur foydalanuvchi tomonidan kiritilgan fan va ustoz ma’lumotlari asosida jadval yaratadi. Jadval tuzishda bir ustozning

bir vaqtning o‘zida ikki darsga tushib qolmasligi hamda bir auditoriyaning bir vaqtda band bo‘lmasligi nazorat qilinadi. Dasturda genetik algoritmning asosiy bosqichlari qo‘llanilgan

1. seleksiya;
2. krossover;
3. mutatsiya.

Fitness funksiyasi orqali har bir jadval sifati baholanadi. Konfliktlar mavjud bo‘lsa fitness qiymati kamayadi, konfliktlarsiz jadval esa maksimal 100% natija beradi. Algoritm bir necha avlod davomida ishlaydi va eng yaxshi jadvalni tanlaydi. Dastur afzalliklari

1. dars jadvalini tez yaratadi;
2. inson xatolarini kamaytiradi;
3. ustoz va auditoriya konfliktlarini oldini oladi;
4. optimal jadval hosil qiladi;
5. katta hajmdagi ma’lumotlar bilan ishlay oladi;
6. avtomatlashtirilgan tizim hisoblanadi.

Loyihaning yutuqlari genetik algoritm amaliy masalada qo‘llanildi, avtomatik jadval yaratish tizimi ishlab chiqildi, konfliktlarni minimallashtirishga erishildi, Python yordamida optimallashtirish modeli yaratildi, sun’iy intellekt elementlari qo‘llanildi. Ushbu loyiha dars jadvalini avtomatik yaratish jarayonini soddalashtiradi va tezlashtiradi. Genetik algoritm yordamida optimal va konfliktlarsiz jadval hosil qilinadi. Dastur ta’lim muassasalarida vaqtni tejash va samaradorlikni oshirishga xizmat qiladi.

Xulosa.

Genetik algoritmlarning asosiy kuchi biron-bir qat’iy formulada emas, balki universalligi va moslashuvchanligidadir. Bu usul muammoni yechishning tayyor retseptini emas, balki yaxshi yechimni tanib olishning samarali mexanizmini taqdim etadi. Har qanday uslub singari, genetik algoritmlar ham hamma masala uchun universal javob bo‘la olmaydi. Masala chiziqli dasturlash yoki dinamik dasturlash kabi ixtisoslashgan usullarga bo‘ysunadigan bo‘lsa, ular ancha tez va aniq ishlaydi. Ammo an’anaviy usullar ojizlik qiladigan murakkab, ko‘p o‘lchovli va gradient hisoblab bo‘lmaydigan masalalar borki, aynan o‘sha hududda genetik algoritmlar o‘zining haqiqiy qudratini namoyon etadi.

Bugungi kunda parallel hisoblash infratuzilmasining kengayishi va amaliy masalalarning tobora murakkablashib borishi genetik algoritmlarning ahamiyatini yanada oshirmoqda. O‘zbekiston ilmiy muhitida ham evolyutsion hisoblashga qiziqish ortib bormoqda, bu esa kelgusida mazkur sohaning yanada rivojlanishiga mustahkam zamin yaratadi.

Foydalanilgan adabiyotlar

1. Saidov A.S., Toshmatov N.R. Diskret matematika va algoritmlar nazariyasi. - Toshkent: “Fan va texnologiya”, 2018.
2. Ibrohimov B.I., Nishonov X.N. Hisoblash usullari va algoritmlar. - Toshkent: TDTU nashriyoti, 2019.
3. Razzaqov A.R. Algoritmlar va ma’lumotlar tuzilmalari: o‘quv qo‘llanma. - Toshkent: “Yangi asr avlodi”, 2017.
4. Yusupov R.M. Optimallashtirish usullari. - Samarqand: SamDU nashriyoti, 2020.
5. Karimov J.X., Xoliqov M.T. Amaliy matematika: darslik. -Toshkent: “O‘qituvchi”, 2016.
6. Tojiboyev M.M., Mamatov N.A. Evolyutsion algoritmlar: o‘quv qo‘llanma. - Toshkent: TATU nashriyoti, 2021.
7. Sobirov S.R. Diskret optimallashtirish va kombinatorial masalalar. - Toshkent: “Fan”, 2019.