*Black box* оптимизација *PSO* алгоритмом

Душан Бркић, Филип Живанац, Ласло Сабади Барањи

Департман за рачунарство и информатику

Факултет техничких наука, Универзитет у Новом Саду

Нови Сад, Србија

***Апстракт*—Овај рад представља примену *PSO* (*Particle Swarm Optimization*)алгоритма као решења *black-box* оптимизационих проблема. У раду се разматра и унапређење претраживачке моћи PSO алгоритма путем разних приступа поставци вредности параметара, као и хибридизацијом са другим алгоритмима. Анализирано је решење путем обичног PSO алгоритма, као и *PSO-GA* хибрида у решавању оптимизационих проблема са ограничењима. Дискутовани су начини модификације параметара алгоритма, као и могућност паралелизације. Проблем оптимизације се често јавља у индустрији и економији. Конкретно, у еконимији, функције које оптимизујемо, поред много непознатих делова имају и много локалних оптимума. *PSO* је по својој природи прилагођен за овакву врсту проблема, а могуће га је и модификовати сходно задатом проблему, без великих губитака у брзини извршавања. При томе се брзина извршавања може додатно побољшати паралелизацијом.**

***Kључне речи—PSO; Particle Swarm Optimisation; PSO-GA hybrid; inertia-weight; PPSO; прерана конвергенција; MPSO; паралелизација;***

# Увод

За сврху решавања комплексних *black-box* оптимизационих проблема се користе еволутивни алгоритми, међу које спада и PSO(*Particle Swarm Optimization*)*.* *Black-box* је нама непо*з*ната функција, чији маскимум или минимум тражимо.

PSO алгоритам нуди јединствен начин да се балансира однос између робусности и перформанси. Мана овог алгоритма је што при лошем одабиру параметара може доћи до преране конвергенције и веома дугог извршавања алгоритма. Да би се ово избегло, алгоритам је потребно прилагодити домену проблема. Адекватно прилагођавање параметара остварује бољи однос између експлорације и експлоатације.

Паралелизација алгоритма је такође битна у случајевима када проблем има пуно димензија. Паралелизација омогућава расподелу задатака на више нити у циљу бржег извршавања.

Одличан начин за додатно унапређење одлика алгоритма јесте и хибридизација, односно спајање основног алгоритма са неком другом нелинеарном методом претраге. На тај начин се могу комбиновати предности једног и другог алгоритма и знатно унапредити моћ претраживања. У раду је описана хибридизација овог алгоритма генетским алгоритмом [5], односно *PSO-GA* [3], као и решавање вишедимензионалног оптимизационог проблема са ограничењима [7].

У поглављу II ће бити реч о основном PSO алгоритму*.* Поглавље 3 садржи опис разних приступа модификације овог алгоритма. Поглавље 4 се фокусира на могућности паралелизације PSO алгоритма. У поглављу 5 се упознајемо са *PSO-GA* хибридним алгоритмом. Коначно, поглавље VI закључује овај рад.

# Основни *PSO*

PSO[1] алгоритам је заснован на имитацији понашања животињских скупина, односно, јединки у тим скупинама (јата птица и риба, ројеви инсеката итд.). PSOје еволутивна, популациона техника. Скуп тачака (потенцијалних решења) посматрамо као честице, чије промене положаја посматрамо као померање позиције усред претраге.

## Иницијализација

Пре почетка итерација алгоритма, потребно је извршити иницијализацију честица. Она се извршава тако што се свакој честици додељује насумична позиција која задовољава ограничења проблема и иницијална брзина честице у насумичном правцу. Алгоритму се такође прослеђује и број димензија оптимизационог проблема, толеранција критеријума заустављања као и функција за евалуацију.

## Рачунање позиције честице

Свака честица памти:

* своју текућу позицију (потенцијално решење),
* најбољу позицију икад достигнуту (најближу решењу) и
* текућу брзину.

Рој, као целина, памти своју најбољу позицију икада постигнуту.

Итеративно, за сваку честицу *k* рачуна се њена брзина, након чега долази до промене позиције према следећим формулама:

где су:

* *v* – брзина, *p* –најбоља позиција честице *k*, *x* – тренутна позиција честице *k*, *g* – глобална најбоља позиција.
* *w* – параметар инерције (инерциони фактор). Овај параметар се на почетку поставља на 1, а кроз алгоритам се смањује. Односно, како се приближавамо оптимуму, честице се крећу све спорије.
* *cp* и *cg* – променљиви фактори убрзања. Служе да смањује шансе да се алгоритам заглави у локалном оптимуму. *cp* је на почетку 2,5 и смањује се, док је *cg* на почетку 0,5 и кроз итерације се повећава.
* *rp* и *rg* – насумични фактори. Представљају бројеве од 0 до 1 који служе да разбију монотоност алгоритма.

У зависности од имплементације алгоритма, неки од ових параметра се могу и проследити зарад ручног подешавања основног алгоритма.

## Заустављање

Заустављање PSO алгоритма се врши када се достигне максимални број итерација који је прослеђен алгоритму. Алтернативно, алгоритам се зауставља када је разлика између резултата најбоље честице у тренутној итерацији и најбоље честице у претходној итерацији мања од прослеђене вредности.

## Анализа и тестирање

Нашу конфигурацију алгоритма смо тестирали над Химелблауовим оптимизационим проблемом [7], који је описан у поглављу 5, а приказан на слици 1. Наши резултати тестирања се налазе у табели 1, у којој приме- ћујемо да наша конфигурација има слабију могућност претраге у области глобалног оптимума.

# Modified Particle Swarm Optimisation (MPSO)

MPSO (*Modified Particle Swarm Optimisation*) се односи на све алгоритме који представљају модификације основног PSO алгоритма. У овом поглављу ћемо се посветити неким особинама различитих модификација алгоритма које су предложене у [2]. Изложене модификације ће се односити на модификације параметара инерције (поглавље A) и модификације ... (поглавље B).

## Модификације параметра инерције

Параметар инерције је параметар који значајно утиче на на перформансе PSO алгоритма. У наредним потпоглављима ћемо описати како стратегије његове промене утичу на баланс експлорације и експлоатације алгоритма.

### Линеарна промена параметра инерције

За PSO алгоритам је критично да се изврше локална и глобална претрага. У ранијим истраживањима је доказано да константа вредност параметра инерције (0.4) не успева да нађе баланс између експлорације и експлоатације. Уколико је његова вредност велика, фаворизује се глобална претрага. У супротном, фаворизује се локална претрага.

Због тога су научници у [2] дошли на идеју да параметар инерције линеарно смањују у свакој итерацији почевши од задатог максимума (0.9) у првој итерацији, па све до задатог минимума (0.4) у последњој итерацији. Оваква стратегија данас је једна од најраспрострањенијих.

Са друге стране, стратегија линеарног повећавања параметра инерције углавном са око 0.4 до око 0.9. Ова стратегија значајно побољшава перформансе алгоритма јер се даје значај на бржој конвергенцији.

### Нелинеарна промена параметра инерције

Ослањајући се на идеју линеарног смањивања вредности параметра инерције, разматрана је могућност о његовом експоненцијалном смањењу. Овакво подешавање омогућава алгоритму да у ранијим фазама извршења брже конвергира, што резултује бољим перформансама не смањујући робусност.

Модификације параметра инерције у којима он узима вредности из конкавне и конвексне опадајуће функције поређене су са линераном опадајућом функцијом. Симулације показују да, уколико параметар инерције узима вредности из конвексне функције, перформансе алгоритма бивају лошије него кад узима вредности из линеарно опадајуће функције, а боље кад узима вредности из конкавне функције.

Значајна побољшања у конвергенцији дала је и стратегија у којој параметар инерције узима вредности из растуће сигмоидне функције. Уколико параметар инерције узима вредности на основу ове стратегије, PSO алгоритам агресивно конвергира ка глобалном оптимуму у каснијим итерацијама, а тиме брже смањује претраживани простор док га не сузи до простора око оптимума. Ова стратегија фаворизује боље истраживање простора око оптимума и бржу конвергенцију.

### Остале промене параметра инерције

Још један начин да се боље истражи иницијални простор даје и насумични одабир параметра инерције. У овој стратегији, вредности параметра инерције узимају вредности из униформне расподеле [0.5, 1]. Ова примена користи се кад се истражују функције које имају више неправилности у себи и када не знамо да одредимо баланс између експлорације и експлоатације.

Због насумичне природе овог параметра добијају се и поларизовнији резултати, док што се тиче перформанси брза конвергенција је присутна у ранијим фазама извршавања. Насумичност овог параметра често уме да резултује добрим балансом између локалне и глобалне претраге.

## Модификације осталих параметара алгоритма

### Иницијализација заснована на хаосу

Значајну улогу у проналаску оптимума игра иницијализација. Циљ иницијализације је да униформно распореди честице у простору који претражујемо. Много научних радова бавило се проблемом иницијализације где су сва добијена решења показала боље резултате него насумична иницијализација. Конкретно, рад [2] описује иницијализацију базирану на логистичкој мапи.

У начину иницијализације приказаном у [2] честице се иницијализују насумично. Затим се за одређени број итерација извршава мапирање честица. Након генерисања честица, оне се пуштају да се крећу по простору. Уколико честица упадне у локални оптимум она се тада мапира на начин предложен у раду [2], а у супротном, њена вредност се мења на начин предложен у истом раду.

Ова стратегија је добра за спречавање преране конвергенције алгоритма јер у је у фази иницијализације спречен утицај глобалног резултата и честицама се даје одређена доза хаоса тј. могућности да претраже своју околину.

### Formulated sigmoid-like inertia weight

*Formulated sigmoid-like inertia weight* је стратегија одабира параметра инерције која је заснована на комбиновању линерне и нелинеарне функције и узима вредности из интервала [0.4, 0.9] или [0.4, 0.95]. За задат проценат, параметар ће узимати вредности горње границе интервала првих максималног максималног броја итерација. Након тога, параметар ће узимати вредности из опадајуће сигмоидне фунцкије од дела у коме она почиње нагло да опада. Оваква функција изгледа налик сигмоидној. Она омогућава боље претраживање простора на почетку због своје велике вредности у свом линеарном делу. За разлику од обичне сигмоидне функције, она у каснијим итерацијама узима мање вредности. На овај начин омогућава се боља претрага простора око дотадашњег глобалног оптимума јер се спречава брза конвергенција у већ претражен простор.

### Остале стратегије

Стратегија ексклузивног ажурирања је стратегија која се фокусира на томе да загарантује конвергенцију алгоритма. То се постиже тако што се параметри честице која има најбољи резултат ажурирају по алгоритму предложеном у [2]. Параметри ове честице се ажурирају све док она не дође до локалног оптимума.

Максимално растојање фокуса је стратегија базирана на степену агрегације једне честице. Што је степен агрегације већи то се честица налази у насељенијем комшилуку. Када се рачуна максимално растојање фокуса у [2], узима се у обзир степен агрегације. На основу добијеног резултата одлучује се о даљој стратегији честице. Над неким честицама се могу извршити мутације које ће променити одређене димензије положаја честице или се у крајњем случају честица реиницијализује по логистичкој мапи.

# Паралелизација

Оптимизациони проблеми су често комплексни и захтевају обраду огромног броја података. Пошто је PSO алгоритам склон да конвергира у локални оптимум, јако је битно ојачати ефикасност разним стратегијама попут паралелизације. Сви алгоритми базирани на разматрању популације решења могу бити паралелизовани тако да се појединачни чланови популације процесирају у паралели. У *PSO* алгоритму све се честице крећу независно од осталих честица у роју и једино зависе од дотад најбољег глобалног решења. Стога је PSO лако паралелизовати. У наредним поглављима биће описане стратегије за PPSO (паралелни PSO) алгоритам.

Када се паралелизује неки алгоритам, треба имати у обзир и циљану компоненту која врши паралелизацију. На пример, паралелизацију је могуће постићи коришћењем једног или више рачунара. Паралелизација се на једном рачунару може вршити на процесору са неколико језгара или на графичкој картици са више хиљада језгара. Описане стратегије груписане су по компоненти која врши паралелизацију.

## Паралелизација базирана на процесору

У овој стратегији паралелизације користимо процесоре са више физичких и виртуелних језгара. Имамо избор да користимо неку од следећих имплементација [4] :

* *Hadoop MapReduce.* Овај модел је развила *Google* компанија за обраду огромног броја података. Модел је у стању да самостално подели податаке, врши опоравак од грешака, итд. За PSO имплементацију, довољно је да дефинишемо неке основне операције. Захтева се и један пар кључа и вредности коју ће библиотека да нам обради
* *MATLAB* *библиотеке за паралелизацију*. *MATLAB* нам нуди један од најлакших начина за паралелизацију. Међутим, овај софтвер није бесплатан. За PSO имплементацију, довољно да дефинишемо паралелну *for* петљу.
* *R Parallel package*

*R* је бесплатан софтвер, који је дизајниран за статистичка рачунања. Постоје библиотеке као *foreach и doParallel*, које нам омогућују паралелизацију и у овом језику.

* *Julia: Parallel for and MapReduce*

*Julia* је модеран и функционалан језик изграђен баш за сврху јаке паралелизације. Овде имамо *@parallel for* и *MapReduce* као опције.

* *Python библиотеке*

*Python* је флексибилан интерпретирани језик. Постоје разне библиотеке за паралелизацију као на пример Joblib.

* *OpenMP са C++*

*С++* је компајлирани језик познат по својој брзини и робустности. *OpenMP* је једна од најлакших библиотека које можемо да користимо за наше сврхе у овом језику.

* *MPI*

Служи за паралелизацију у системима са више рачунара.

## Паралелизација базирана на графичкој картици

Прошлих година паралелизација коришћењем графичке картице постаје све популарнија. Она може да има више хиљада језгара. Све претходно поменуте имплементације се могу и овде користити, али постоје и имплементације које су направљене само за графичке картице, од којих су најпопуларнији:

* *CUDA*

Овај модел је изградила компанија *nVIDIA*, за коришћење на њиховим компатибилним картицама. Корисник мора да дефинише функције које ће картица да изврши, затим да алоцира меморију за променљиве.

* *OpenACC*

Има архитектуру *OpenMP,* код се гради тако што се серијском коду додају неке кључне речи. Код може да се пребаци и на процесор, чак и на њихову мешавину

## Конвенционални PPSO алгоритми

Када се паралелизује неки алгоритам, један од највећих проблема нам представља синхронизација, то јест, комуникација међу задацима. У овом случају задаци су нам под-ројеви или саме честице. Четири најкоришћенија алгоритама за комуникацију су:

* Star *PPSO* (звезда – поглавље ),
* *Migration PPSO* (миграција познато и под кружни или прстен),
* *Diffusion PPSO*,
* *Broadcast PPSO*.

Илустратитвни примери комуникације међу под-ројевима може да се нађе у [4].

### Звезда

Овај алгоритам има *master-slave* топологију, што значи да имамо један под-рој или једну честицу која је надређена, и која шаље свим осталим подређеним под-ројевима. Не постоји директна комуникација међу подређеним под-ројевима. Кораци у овом алгоритму су:

1. Надређени одлучи какве параметре за алгоритам ће користити, и подели их са подређенима. Ови параметри су углавном број итерација, тежина инерције, период комуникације, величина популације и коефицијенти убрзања
2. Изврши се померанје роја, сваки под-рој паралелно мења или добија своју досад најбољу вредност, и глобалну најболју вредност.
3. Кад је свако извршио своје померање, подређени шаљу надређеном своју досад најбољу вредност.
4. Надређени из свих досад примлљених вредности бира најбољу.
5. Свима се промени коефицијент убрзања и позиција.
6. Опет се шаљу информације о персоналним оптимума надређеном и опет се мења глобални оптимум.
7. Опет се шаљу информације о персоналним оптимума надређеном и опет се мења глобални оптимум.

### Миграција

У овом алгоритму су под-ројеви повезани у једном кругу, и под-ројеви могу само да комуницирају са комшијама. Један под-рој може да комуницира само са под-ројем који је са његове леве или десне стране. Кораци овог алгоритма су:

1. Сви параметри су познати одмах на почетку.
2. Сви подројеви се померају у паралели, и паралел-
3. но дођу до својих персоналних и глобалних оптимума.
4. Најбоља честица се замени са најгором честицом у комшијском под-роју. Током сваке комуникације међу под-ројевима се и глобални оптимум изме- њује.
5. Под-ројевима се измени позиција и коефицијент убрзанја са новим персоналним и глобалним оптимумима.
6. Понављамо корак 3.
7. Ове кораке понављамо док не буде задовољен критеријум заустављања.

### Broadcast

Принцип овог алгоритма је да сваки под-рој може да комуницира са сваким другим под-ројем. Сви под-ројеви се извршавају у паралели, и све информације шаљу свим осталим под-ројевима. Први и други корак овог алгоритма су идентични миграционом, а остали су:

1. Сви под-ројеви шаљу своју дотад најбољу позицију да би се сазнало која је сада најбоља позиција.
2. После измене оптимума, под-ројеви ажурирају своје позиције и убрзања.
3. Понављамо корак 1.
4. Ове кораке понављамо док не буде задовољен критеријум заустављања.

### Diffusion

Овај алгоритам је јако сличан миграционом алгоритму, са разликом да сада сваки под-рој има 4 комшије. Под-ројеви има левог и десног комшију, али и горњег и доњег, па су под-ројеви распоређени налик неке матрице, са разликом да су под-ројеви на угловима, где не би имали 4 комшије избачени. Кораци који треба да се имплементирају су исти као кораци из миграционог алгоритма

# *PSO-GA* хибридни алгоритам

У овом поглављу предложен је хибридни алгоритам [3] добијен коришћењем *particle swarm* оптимизације и генет- ског алгоритма, који је даље коришћен за решавање

оптимизационих проблема са ограничењима, а његова решења су анализирана и упоређена са решењима других аутора, добијених њиховим верзијама еволутивних алго- ритама, као и са решењем нашег, основног *PSO* алгоритма.

## Генетски алгоритам

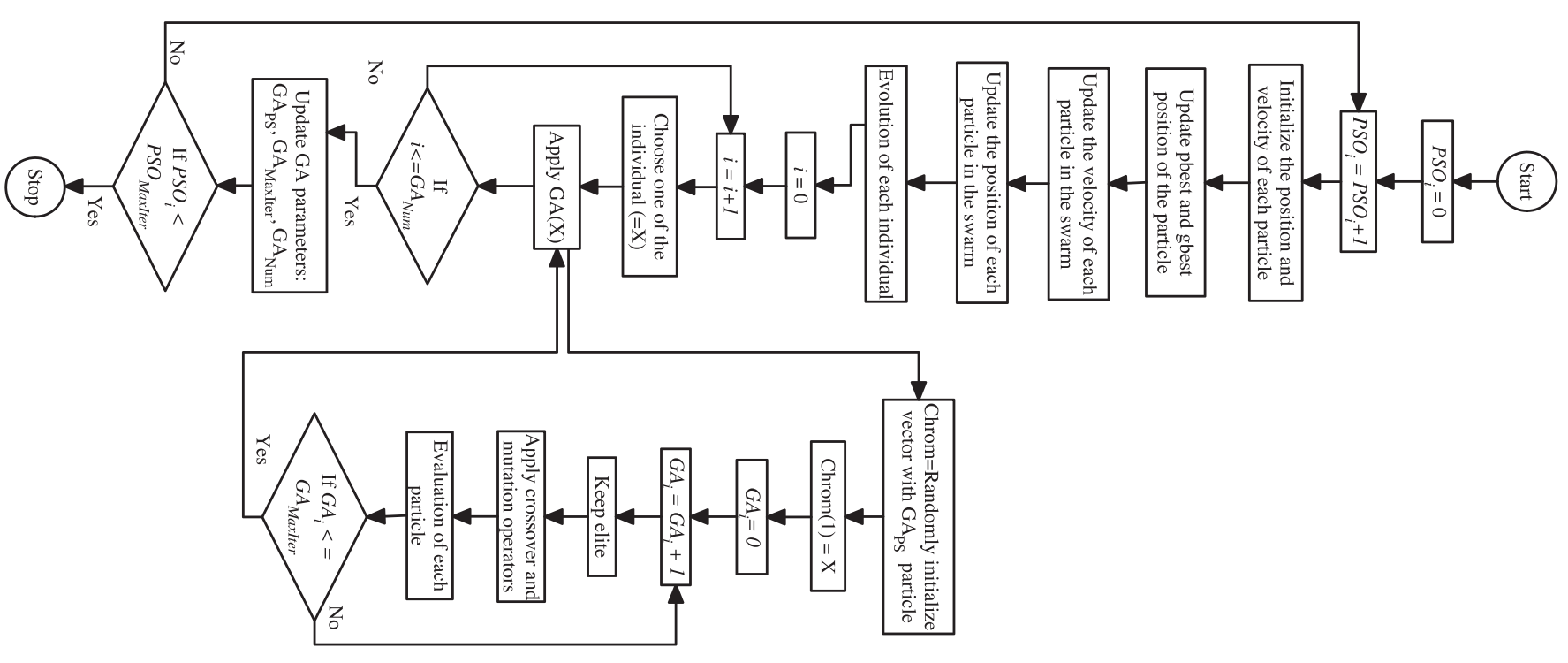
Генетски алгоритам [5] је еволутивни алгоритам прет- раге заснован на хеуристици и природној селекцији. Први пут је предложен 1960. од стране Тома Холанда, и до сада је широко испитиван и коришћен у разним инжењерским дисциплинама. Фундаментални концепт алгоритма је базиран на тези „опстанка најприлагођенијих“ Чарлса Дарвина. У алгоритму, претпостављени скуп решења (популација), пролази кроз селекцију која омогућава варијабилност, а користи технике инспирисане природном селекцијом, као што су мутација и рекомбинација (кросовер). Свако појединачно решење (индивидуа) је оцењено његовом вредношћу у функцији претраге (фитнес), од ког зависи да ли ће решење учесвовати у креирању нове итерације популације (генерације).

## Опис хибридног алгоритма

Мотивација иза креирања *PSO-GA* хибрида је свакако идеја да се споје предности генетског алгоритма и particle swarm оптимизације. Оба алгоритма имају своје предности и мане. У генетском алгоритму, уколико индивидуа није селектована, информације које је она носила се губе заувек, што значи да постоје веће шансе да се заглави у локалном оптимуму, односно слабије је робусности. *PSO* не уништава честице које се покажу лоше у тражењу решења, што га чини робуснијим, мада оне расипају доста ресураса, што успорава конвергенцију. Дакле, основна идеја комбинације ова два алгоритма јесте спајање могућности друштвеног мишљења у *PSO*, са предностима локалне претраге у *GA*. Пошто су и један и други базирани на популацији, додатно се олакшава њихово комбиновање.

Алгоритам почиње из фазе иницијализације, у којој се иницијализују честице и њихове брзине насумично преко простора претраге, односно свака честица *xi* насумично узима позицију из униформне расподеле U(x\_min,x\_max), у рангу [x\_min,x\_max], где x\_min и x\_max представљају доње и горње ограничење. Вектор брзине (1) се састоји из два фактора - личног и друштвеног, односно заснован је на знању сваке честице - најбољем положају у којем је честица била (*personal best - pbest*), као и на целокупном знању читавог јата, односно најбољој позицији у којем се јато налазило (*global best - gbest*). У таквој конфигурацији свака честица узима у обзир своје лично искуство, као и искуства њених суседа. Након рачунања брзине, свака честица мења своју позицију пратећи једначину (2).

Када се оправи нова генерација честица, са унапређеним положајима сваке честице, одређени број честица се селектује и над сваком честицом се примењује *GA* засебно. Након што се из *GA* популације изабере најбоља честица, *GA* има задатак да направи нову популацију смењујући тачке у тренутној популацији бољим тачкама користећи генетске принципе, и то приме- њујући операторе селекције, мутације и рекомбинације. Селекција је примењена методом точка рулета (*roulette wheel selection*), а рекомбинација једном тачком раздвајања (*one point crossover*). Након селекције, мутације и реком- бинације, примењена је и форма елитизма, за очување најбољих решења у популацији. Након евалуације нове популације, величина популације и максимални број итерација се ажурира узимајући у обзир тренутну итерацију *PSO* алгоритма. Кроз понављање процеса репродукције популације, популација се води ка глобалном оптимуму. Репрезентација алгоритма је приказана на слици 1.



Слика 1 - *PSO-GA*

## Оптимизациони проблеми

Представљени оптимизациони проблеми јесу нелинеарни оптимизациони проблеми са ограничењима. Главни задатак у решавању ових проблема јесте баратање са ограничењима. Најуобич -30486.181541 ајенији начин за баратање ограничења у заједници која се бави еволутивним алгоритмима је увођење пенала. У том случају, уколико

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Методе | Променљиве | | | | | Решење | Ограничења | | |
| x1 | x2 | x3 | x4 | x5 | g1(X) | g2(X) | g3(X) |
| *GA* [8] | 80.39 | 35.07 | 32.05 | 40.33 | 33.34 | −30005.700 | 91.65619 | 99.53690 | 20.02553 |
| *Himmelblau* [7] | NA | NA | NA | NA | NA | −30373.949 | NA | NA | NA |
| Наш *PSO* | 79.06 | 34.05 | 30.58 | 43.8 | 35.53 | -30486.181 | 91.98643 | 95.166912 | 20.0000 |
| *cuckoo* [11] | 78.00 | 33.00 | 29.99616 | 45.00 | 36.77605 | −30665.233 | 91.99996 | 98.84067 | 20.0003 |
| *harmony* [9] | 78.00 | 33.00 | 29.995 | 45.00 | 36.776 | −30665.500 | 92.00004 | 98.84051 | 19.99994 |
| *PSO* [10] | 78.00 | 33.00 | 29.995256 | 45.00 | 36.7758129 | −30665.539 | 93.28536 | 100.40478 | 20.00000 |
| *PSOa*, *PSOstr* [13] | 78.00 | 33.00 | 29.995256 | 45.00 | 36.775813 | −30665.54 | 92.00000 | 98.84050 | 20.00000 |
| *simplex* [12] | 78.00 | 33.00 | 29.995256 | 45.00 | 36.775813 | −30665.538 | NA | NA | NA |
| *PSO* [10] | 78.00 | 33.00 | 29.995256 | 45.00 | 36.7758129 | −30665.539 | 93.28536 | 100.40478 | 20.00000 |
| *GA*[6] | NA | NA | NA | NA | NA | −30665.539 | NA | NA | NA |
| *PSO-GA* [3] | 78.00 | 33.00 | 29.9951741 | 45.00 | 36.7757340 | −30665.56614 | 91.99999 | 98.84047 | 20.000 |
| Табела 1 – Решења Химелблауовог оптимизационог проблема | | | | | | | | | |

пронађено решење не задовољава ограничења, решење се „казни“ тако да не учествује у претрази. Упркос великој популарности, увођење пенала има многе недостатке, од којих је главни то што има превише параметара, и тражење одговарајуће комбинације ограничења није лак посао, а такође и узима превише времена. Решење је коришћење пеналних функција без параметара [6].

### Химелблауов нелинеарни оптимизациони проблем

Овај проблем је оригинално предложио Химелблау [7], и он је широко коришћен за упоређивање ефикасности различитих еволутивних алгоритама. Проблем је дефинисан као петодимензионални, са шест нелинеарних ограничеања типа неједнакости и десет граничних услова. Дефинисан је у фигури 1. У табели 1 се може видети како решење *PSO-GA* хибрида стоји у поређењу са решењима пронађеним другим алгоритмима. У табели су приказана решења разних метода као што су: *GA* [8, 6], *harmony search* [9], *PSO* [10], кукавичја претрага (*cuckoo search*) [11], симплекс (*simplex search*) [12] и *PSOa*, *PSOstr* [13]. Такође се да приметити да решење (линк) не задовољава ограничење *g1*, тако да није валидно. Примећује се да препоручена метода проналази најбоље решење X=[78.00,33.00,29.99517417,45.00,36.7757340], а вредност функције F(X)=-30665.56614.

На основу изнетих података да се закључити да је *PSO-GA* препоручена метода по природи врло робусна и има најбољи квалитет претраге. Штавише, најгори резултат добијен препорученом методом је и даље бољи од било ког решења добијених осталим методама претраге.

Minimize f(X)= 5.3578547(x3)^2 + 0.8356891x1x5 + 37.293239x1 - 40792.141 s.t.

0 ≤ g1(X) ≤ 92

90 ≤ g2(X) ≤ 110

20 ≤ g3(X) ≤ 25

where g1(X) = 85.334407 + 0.0056858x2x5 + 0.0006262x1x4 - 0.0022053x3x5

g2(X) = 80.51249 + 0.0071317x2x5 + 0.0029955x1x2 - 0.0021813(x3)^2

g3(X) = 9.300961 + 0.0047026x3x5 + 0.0012547x1x3 + 0.0019085x3x4

78 ≤ x1 ≤ 102

33 ≤ x2 ≤ 45

27 ≤ x3,x4,x5 ≤ 45

Фигура 1 - Химелблауов оптимизациони проблем

# Закључак

Проблем непознатих функција често се јавља у индустрији, поготово у области економије. Тај проблем смо решавали модификацијом основног *PSO* алгоритма, његовом паралелизацијом као и хибридизацијом другим алгоритмима. Применом различитих алгоритама базираних на *PSO* добили смо иницијалну представу о томе како се функција понаша и сходно томе бисмо прилагодили *PSO* задатом проблему.

У области модификације параметара алгоритма развој постаје спорији због ограничених могућности измене. Паралелизовање *PSO* алгоритма је условљено развојем хардверских компоненти. Показано је на примеру да се слабости алгоритма могу анулирати неким другим еволутивним алгоритмом, који би их својим одликама поништио. Битно је увек тежити ка неконвенционалним решењима и истраживати нове идеје за решавање нелинеарних оптимизационих проблема.

# Литература

1. Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. Proceedings of ICNN’95 - International Conference on Neural Networks.
2. Tian, D., & Shi, Z. (2018). MPSO: Modified particle swarm optimization and its applications. Swarm and Evolutionary Computation, 41, 49–68.
3. Garg, H. (2016). A hybrid PSO-GA algorithm for constrained optimization problems. Applied Mathematics and Computation, 274, 292–305.
4. A Survey on Parallel Particle Swarm Optimization Algorithms Soniya Lalwani · Harish Sharma · Suresh Chandra Satapathy · Kusum Deep· Jagdish Chand Bansal © King Fahd University of Petroleum & Minerals 2019.
5. D. E. Goldberg, Genetic Algorithm in Search, Optimization and Machine Learning, MA: Addison-Wesley (1989).
6. K. Deb, An efficient constraint handling method for genetic algorithms, Comput. Methods Appl. Mech. Eng. 186 (2000) 311–338.
7. D.M. Himmelblau, Applied nonlinear programming, McGraw-Hill, New York, 1972.
8. A. Homaifar, S.H.Y. Lai, X. Qi, Constrained optimization via genetic algorithms, Simulation 62 (4) (1994) 242–254.
9. K.S. Lee, Z.W. Geem, A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice, Comput. Methods Appl. Mech. Eng. 194 (2005) 3902–3933.
10. S. He, E. Prempain, Q.H. Wu, An improved particle swarm optimizer for mechanical design optimization problems, Eng. Optim. 36 (5) (2004) 585–605.
11. A. Gandomi, X.S. Yang, A. Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems, Eng. Comput. (2011a) 1–19
12. V.K. Mehta, B. Dasgupta, A constrained optimization algorithm based on the simplex search method, Eng. Optim. 44 (5) (2012) 537–550.
13. G.G. Dimopoulos, Mixed-variable engineering optimization based on evolutionary and social metaphors, Comput. Methods Appl. Mech. Eng. 196 (4-6) (2007) 803–817.