# *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 унапређење претраживачке моћи PSO алгоритма путем разних приступа постављања вредности параметара, хибридизацијом са генетским алгоритмом, као и његовом паралелизацијом. Анализирано је решење добијеног путем обичног PSO алгоритма, као и PSO-GA хибрида у решавању Химелблауовог оптимизационог проблема. Дискутоване су стратегије модификовања параметара алгоритма, као и могућност паралелизације. Проблем *black-box* оптимизације се често јавља у индустрији и економији. Конкретно, у економији, функције које оптимизујемо су непознате и имају много локалних оптимума. PSO је по својој природи прилагођен за овакву врсту проблема. Покретањем основног алгоритма, стиче се иницијална представа о природи оптимизационог проблема. Према томе се може одредити даља стратегија промене алгоритма, као што су модификација параметара и хибридизација са другим еволутивним алгоритмима, сходно задатом проблему, без великих губитака у брзини извршавања. При томе се брзина извршавања може додатно побољшати паралелизацијом.**

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

## I. Увод

У сврху решавања комплексних *black-box* оптимизационих проблема се користе еволутивни алгоритми, међу које спада и PSO (*Particle Swarm Optimization*) [1]. *Black-box* је нама непозната функција, чији максимум или минимум тражимо.

PSO алгоритам нуди јединствен начин да се балансира однос између робусности и перформанси [1]. 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, односно између претраге простора и одабира најбољих резултата. У раду је дискутовано побољшање алгоритма модификацијом параметра инерције, с обзиром на домен проблема.

Паралелизација алгоритма је такође битна у случајевима вишедимензионалних функција. 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 III садржи опис разних приступа модификације овог алгоритма. Поглавље IV се фокусира на могућности паралелизације PSO алгоритма. У поглављу V се упознајемо са PSO-GA хибридним алгоритмом. Поглавље VI служи евалуацији конфигурације нашег алгоритма над Химелблауовим оптимизационим проблемом, као и упоређивање решења са другим ауторима. Коначно, поглавље VII закључује овај рад.

## II. Основни *PSO*

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

Алгоритму се прослеђује:

- број димензија оптимизационог проблема,
- толеранција критеријума заустављања,
- функција за евалуацију,
- број честица,
- максимални број итерација.

Као резултат алгоритам враћа:

- позицију оптимума,
- вредност функције у оптимуму,
- време извршавања.

Кораци наведеног алгоритма су иницијализација честица, рачунање њихових нових позиција, као и критеријум заустављања. У наредним потпоглављима ће бити објашњени наведени кораци.

### A. *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.

### B. *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:

$$v[k] = w[k] \cdot v[k-1] + cp[k] \cdot rp[k] \cdot (p[k] - x[k]) + cg[k] \cdot rg[k] \cdot (g[k] - x[k])$$

$$x[k+1] = x[k] + v[k]$$

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 смањују шансе да се алгоритам заглави у локалном оптимуму. $cp$ је на почетку 2,5 и смањује се до 0,5, док је $cg$ на почетку 0,5 и кроз итерације се повећава до 2,5 [1].
- $rp$ 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.

### C. *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.

## III. MODIFIED PARTICLE SWARM OPTIMISATION (MPSO)

MPSO (*Modified Particle Swarm Optimization*) 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 (A) и стратегије модификовања (B).

### A. *Модификације параметра инерције*

Параметар инерције је параметар који значајно утиче на перформансе, првенствено на брзину извршавања, PSO алгоритма [2]. У наредним потпоглављима ћемо описати како стратегије његове промене утичу на баланс експлорације и експлоатације алгоритма.

#### 1) *Линеарна промена параметра инерције*

За PSO алгоритам је критично да се изврше локална и глобална претрага [2]. Локална претрага је претрага простора у околини честице, односно у околини места на ком је иницијализована свака честица. Глобална претрага је претрага целокупног простора проблема. У ранијим истраживањима је доказано да константа вредност параметра инерције (0.4) не успева да нађе баланс између експлорације и експлоатације [1]. Уколико је његова вредност велика, фаворизује се глобална претрага. У супротном, фаворизује се локална претрага.

Због тога су научници у [2] дошли до закључка 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, линеарно повећавање параметра инерције углавном се врши повећавањем са 0.4 на 0.9. Ова стратегија значајно побољшава перформансе алгоритма јер се даје значај на бржој конвергенцији [2].

#### 2) *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 [2]. Симулације показују да, уколико параметар инерције узима вредности из конвексне функције, перформансе алгоритма бивају лошије него кад узима вредности из линеарно опадајуће функције. Насупрот томе када параметар инерције узима вредности из конкавне функције перформансе бивају боље.

Значајна побољшања у конвергенцији дала је и стратегија у којој параметар инерције узима вредности из растуће сигмоидне функције [2]. 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.

#### 3) *Остале промене параметра инерције*

Још један начин да се боље истражи иницијални простор даје и насумични одабир параметра инерције [2]. У овој стратегији, вредности параметра инерције узимају вредности из униформне расподеле [0.5, 1]. Ова примена се користи 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 поларизованији 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.

### B. *Стратегије за MPSO*

#### 1) *Иницијализација заснована на хаосу*

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

У овом начину иницијализације, све честице се иницијализују насумично у интервалу [0, 1], под претпоставком да радимо са једном димензијом, уколико радимо са више, исти алгоритам важи за сваку димензију. Затим се за произвољан број итерација n извршава ремапирање честица. Ремапирање честица се врши по следећој формули:

$$x_{n+1} = ux_n(1 - x_n)$$

где је x_n вредност честице у n-тој итерацији а u(коефицијент бифуркације) је предефинисана константа за коју се најчешће узима вредност 4. Када се заврши итерирање, вредности честица се скалирају на простор проблема. Ова стратегија на псеудо-насумичан начин распоређује честице, тако да оне буду боље распоређене него да су насумично иницијализоване. Боља распоређеност честица резултује бољој претрази простора, као што је експериментима доказано у [2] .

#### 2) *Formulated sigmoid-like inertia weight*

*Formulated sigmoid-like inertia weight* је стратегија одабира параметра инерције која је заснована на комбиновању линеарне и нелинеарне функције и узима вредности из интервала [0.4, 0.9] или [0.4, 0.95]. За задат проценат α, параметар ће узимати вредности горње границе интервала првих α максималног броја итерација. Након тога, параметар ће узимати вредности из опадајуће сигмоидне функције 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.

#### 3) *Остале стратегије модификовања PSO*

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.

Стратегија максималног растојања фокуса је стратегија базирана на сконцентрисаности честица око честице са најбољим резултатом.

Након одређеног броја итерација алгоритма, честице чије је растојање веће од просечног у односу на честицу са најбољим резултатом се мењају. Оне се мењају тако што се над њима врши мутација предложена у [2] или се реиницијализују по логистичкој мапи. На овај начин спречава се заглављивање алгоритма у више локалних оптимума.

## IV. 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 имплементирати робустан алгоритам. Проблем са тиме је да су робусне имплементације знатно спорије од основног алгоритма. Из тог разлога је битно убрзати алгоритам 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 се све честице крећу независно од осталих честица у рою 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 узети 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.

### A. *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*
- *MATLAB библиотеке за паралелизацију*
- *R Parallel package*
- *Julia: Parallel for and MapReduce*
- *Python библиотеке*
- *OpenMP са C++*
- *MPI*

### B. *Паралелизација базирана на графичкој картици*

Од пре пар 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 [4]:

- *CUDA*
- *OpenACC*

### *C. 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 најкоришћенијих алгоритама за комуникацију су:

- Star PPSO – 1),
- *Migration* PPSO – 2),
- *Broadcast* PPSO – 3).
- *Diffusion* PPSO – 4),

Илустративни примери комуникације међу под-ројевима могу да се нађу у [4].

#### *1) Звезда*

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

1. Надређени под-рој 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 померање роја, сваки под-рој паралелно мења или добија своју досад најбољу вредност, и глобалну најбољу вредност.
3. Кад је свако извршио своје померање, подређени шаљу надређеном своју досад најбољу вредност.
4. Надређени из свих досад примљених вредности бира најбољу.
5. Свима се промени коефицијент убрзања и позиција.
6. Опет се шаљу информације о персоналним оптимумима надређеном и глобални оптимум се мења.
7. Ове кораке понављамо док не буде задовољен критеријум заустављања.

#### *2) Миграција*

У овом алгоритму су под-ројеви повезани у једном кругу, и под-ројеви могу само да комуницирају са суседним под-ројевима. 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 под-ројеви се померају у паралели, и паралелно дођу до својих персоналних и глобалних оптимума.
3. Најбоља честица се замени са најгором честицом у суседном под-року. Током сваке комуникације међу под-ројевима се и глобални оптимум измењује.
4. Под-ројевима се измени позиција и коефицијент убрзања са новим персоналним и глобалним оптимумима.
5. Понављамо корак 3.
6. Ове кораке понављамо док не буде задовољен критеријум заустављања.

#### *3) 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.

#### *4) Diffusion*

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

## *V.* PSO-GA ХИБРИДНИ АЛГОРИТАМ

У овом поглављу предложен је хибридни алгоритам [3] добијен коришћењем *particle swarm* оптимизације (PSO) и генетског алгоритма (GA), који је даље коришћен за решавање Химелблауовог оптимизационог проблема [7], а његова решења су анализирана и упоређена са решењима других аутора, добијених њиховим верзијама еволутивних алгоритама, као и са решењем нашег, основног PSO алгоритма.

### *A. Генетски алгоритам*

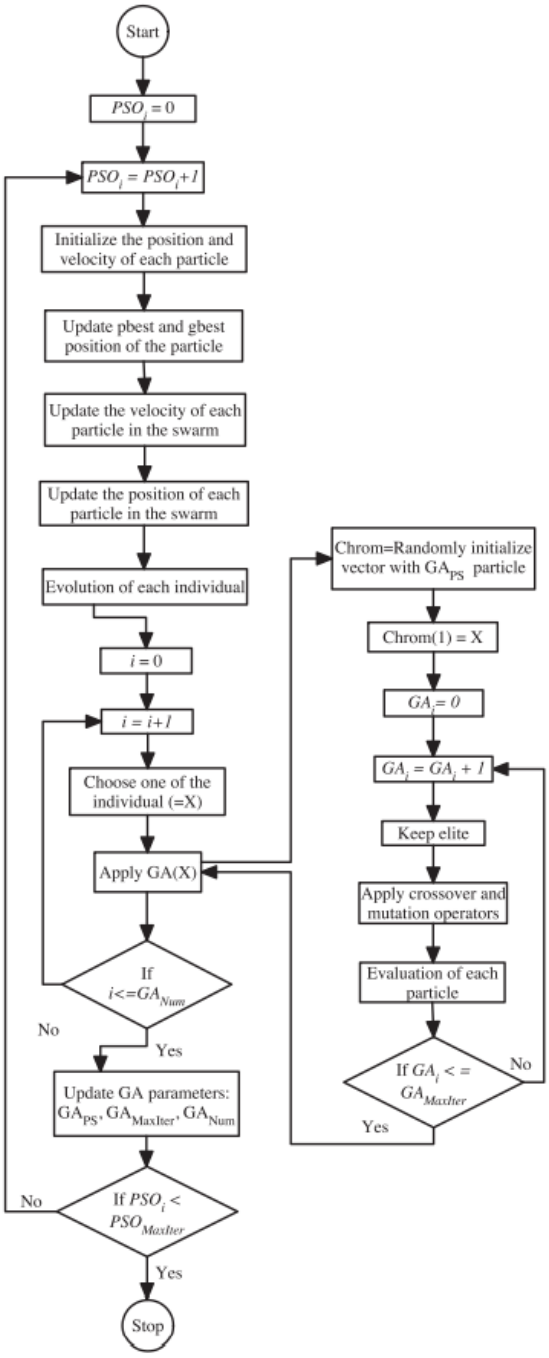
Генетски алгоритам [5] је еволутивни алгоритам претраге заснован на хеуристици и природној селекцији. Први пут је предложен 1960. од стране Тома Холанда, и до сада је широко испитиван и коришћен у разним инжењерским дисциплинама. Фундаментални концепт алгоритма је базиран на тези „опстанка најприлагођенијих“ 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 учествовати у креирању нове итерације популације (генерације).

### *B. 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 ресурса, 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), где x_min и x_max представљају доње и горње ограничење. Вектор брзине (B) 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 (B).

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 примењујући 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 рекомбинације, 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 – Скица PSO-GA*

## VI. Евалуација решења

У овом поглављу ће бити евалуирано наше решење добијено нашом конфигурацијом, основног PSO, над Химелблауовим нелинеарним оптимизационим проблемом, са решењима PSO-GA хибрида [3], као и неких других оптимизационих метода:

- GA [8, 6],
- *harmony search* [9],
- PSO [10],
- кукавичја претрага (*cuckoo search*) [11],
- симплекс (*simplex search*) [12],
- PSOa, PSOstr [13].

Поредиће се искључиво решења алгоритама, како би се утврдило који је најефикаснији у претрази око глобалног оптимума. Дата решења се могу видети у табели 1.

Да се приметити да решење [10] не задовољава ограничење *g1*, тако да није валидно. Такође се примећује да PSO-GA метода проналази најбоље решење X=[78.00, 33.00, 29.99517417, 45.00, 36.7757340], а вредност функције F(X)=-30665.56614.

### A. Химелблауов нелинеарни оптимизациони проблем

Овај проблем је оригинално предложио Химелблау [7], и одабран је јер је широко коришћен за упоређивање ефикасности различитих еволутивних алгоритама. Проблем је дефинисан као петодимензионални, са шест нелинеарних ограничења типа неједнакости и десет граничних услова. Овај проблем садржи доста локалних оптимума у пределу око глобалног, тако да ће се најбоље показати алгоритми који су најефикаснији у локалној претрази.

### B. Конфигурација нашег основног PSO

За потребе тестирања користили смо основни PSO са следећим подешавањима:

- *cp* је на почетку 2.5 и кроз итерације се смањује до 0.5,
- *cg* је на почетку 0.5 и кроз итерације се повећава до 2.5,
- *w* је на почетку 1 и у свакој итерацији се множи са 0.99,
- број итерација 200,
- број честица 1000,
- толеранција $10^{-15}$.

Ограничења смо руковали употребом казнене функције.

| 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* | 78.00 | 33.00 | 29.99 | 44.99 | 36.75 | -30665.190 | 91.99 | 94.915 | 20.0002 |
| *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 |
| *SIMPLEX* [12] | 78.00 | 33.00 | 29.995256 | 45.00 | 36.775813 | −30665.538 | NA | NA | NA |
| *GA*[6] | NA | NA | NA | NA | NA | −30665.539 | NA | NA | NA |
| *PSO* [10] | 78.00 | 33.00 | 29.995256 | 45.00 | 36.7758129 | −30665.539 | 93.28536 | 100.40478 | 20.00000 |
| *PSO*$^{A}$, *PSO*$^{STR}$ [13] | 78.00 | 33.00 | 29.995256 | 45.00 | 36.775813 | −30665.54 | 92.00000 | 98.84050 | 20.00000 |
| *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*

## VII. Закључак

Проблем проналажења оптимума *black-box* функција често се јавља у индустрији, поготово у области економије. Код *black-box* функције једина информација коју можемо да добијемо јесте евалуација у некој одређеној тачки. Такве функције се стога могу решавати ефикасно нелинеарним оптимизационим претрагама. Дискутовали смо решавање тог проблема модификацијом основног *PSO* алгоритма, његовом паралелизацијом као и хибридизацијом другим алгоритмима. Упознавањем са различитим методама модификације и хибридизације алгортима, закључили смо да стратегија иницијализације засноване на хаосу побољшава глобалну претрагу, док хибридизација генетским алгоритмом побољшава претрагу у области оптимума.

У области модификације параметара и иницијализације алгоритма развој је условљен применом нових математичких концепата.

Даље напредовање у паралелизацији PSO алгоритма је условљено развојем хардверских компоненти и прилагођавања алгоритма њима.

Показано је на примеру да се слабости алгоритма могу анулирати неким другим еволутивним алгоритмом, који би их својим одликама поништио. На основу изнетих података из табеле 1 да се закључити да је PSO-GA метода по природи врло робусна и има најбољи квалитет претраге у области глобалног оптимума. Даљи развој хибридизације овог алгоритма је условљен проналажењем нових нелинеарних метода претраге које опонашају природне појаве.

## VIII. 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.