# 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 робустне 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 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 податке, врши опоравак од грешака, итд.

* *MATLAB* *библиотеке за паралелизацију*. *MATLAB* нам нуди један од најлакших начина за паралелизацију. Међутим, овај софтвер није бесплатан.
* *R Parallel package*

*R* је бесплатан софтвер, који је дизајниран за статистичка рачунања. Постоје разне библиотеке 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.

* *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– поглавље познато и под кружни или прстен),
* *Diffusion PPSO*– поглавље,
* *Broadcast PPSO*– поглавље.

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 информације о глобалном оптимуму 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 најбољу 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 оптимумима надређеном и глобални оптимум се мења.
7. Ове кораке понављамо док не буде задовољен критеријум заустављања.

### 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 паралелно дођу до својих персоналних и глобалних оптимума.
3. Најбоља честица се замени са најгором честицом у комшијском под-роју. Током сваке комуникације међу под-ројевима се и глобални оптимум изме- њује.
4. Под-ројевима се измени позиција и коефицијент убрзанја са новим персоналним и глобалним оптимумима.
5. Понављамо корак 3.
6. 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 имају левог и десног комшију, али и горњег и доњег. Под-ројеви су 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