به نام او که از ما به شما عاشق‌تر است

**تحلیل و طراحی چالش Data Date**

**گروه نرم افزاری پارت**

نفیسه عامری1

1 دانشجو ارشد هوش مصنوعی، دانشکده مهندسی، دانشگاه فردوسی، مشهد

**چکیده**

در این چالش قصد داریم که ابتدا با دریافت تعداد تگ از کاربر و ارسال آن به سرور درخواست می­دهیم که لیستی از آبجکت­های مورد نظر را که آن تگ­های مدنظر کاربر را دارد، به ما برگرداند. سپس با این لیست، عملیات مرتب سازی و صفحه­بندی را با توجه به نیاز کاربر انجام می­دهیم. این برنامه را با زبان node.js نوشته­ایم.

# شرح تکنیکال

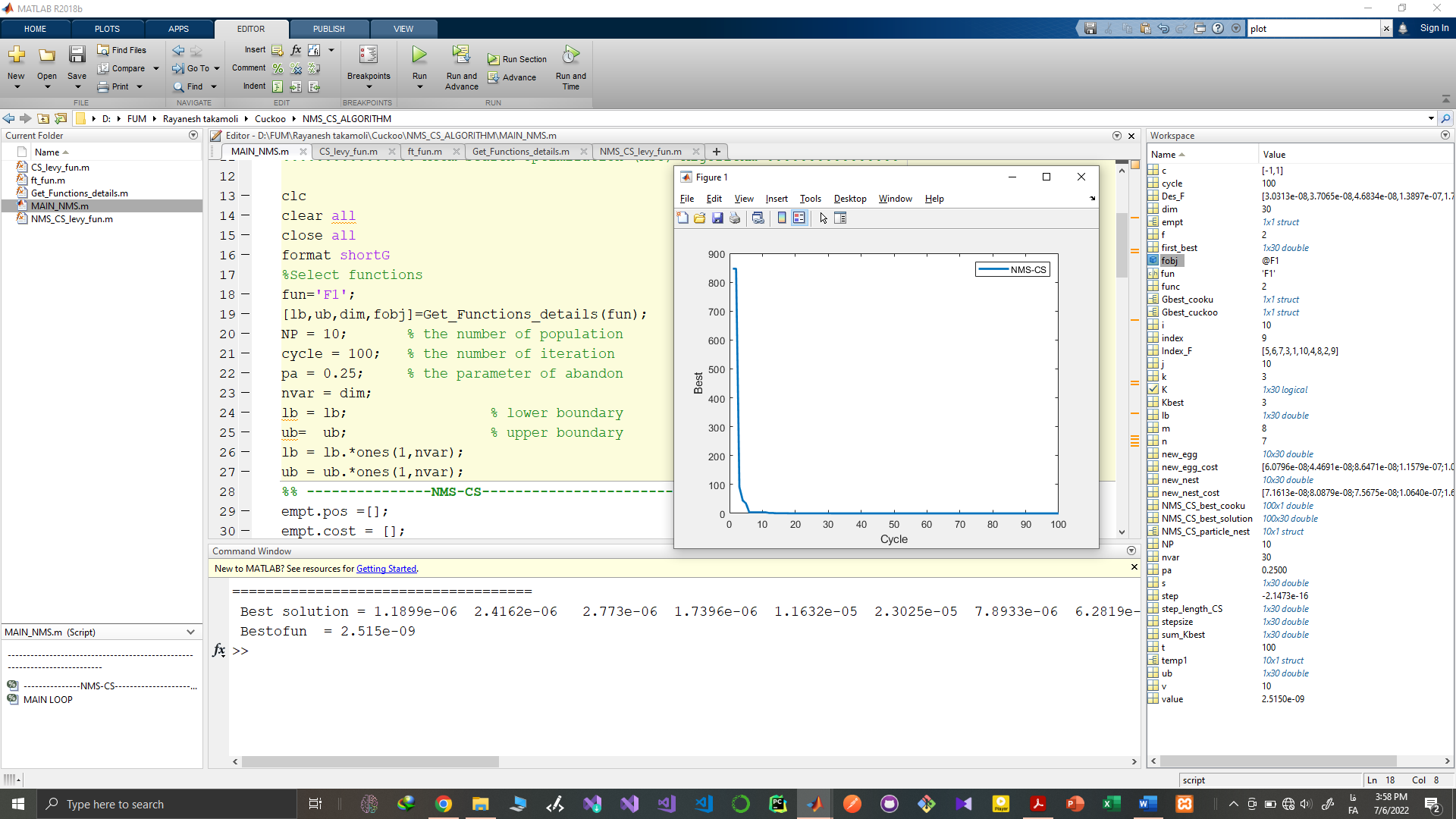
ابتدا با دریافت تعداد تگ از کاربر و ارسال آن به سرور درخواست می­دهیم که لیستی از آبجکت­های مورد نظر را که آن تگ­های مدنظر کاربر را دارد، به ما برگرداند. به دلیل اینکه به سرور دسترسی نداریم، یک شبیه­سازی برای سرور نوشتیم؛ به این صورت که از فایل sample.json که تعدادی آبجکت نمونه وجود دارد، میخوانیم و به درخواست ارسال شده پاسخ می­دهیم.

برای پیاده سازی بخش اصلی برنامه، ابتدا پارامترهای که در تصویر 2 وجود داشت از کاربر دریافت می­کنیم. سپس متغییر Tags را به سرور ارسال کرده و خروجی آن را در DataSet ذخیره می­کنیم.

سپس با این لیست، عملیات مرتب سازی و صفحه­بندی را با توجه به نیاز کاربر انجام می­دهیم.

# شرح نتایج

خروجی نتایج مسئله به شرح زیر است.



شکل ‏2‑1-نمودار بهترین دقت طی یک دوره 100تایی با تابع F1

همانطور که در شکل­های بالا نشان داده شده است، تابع F5 نتوانسته به خوبی همگرا شود و در دوره 13 توانسته جواب بهینه را پیدا کند؛ مانند تابع F3. همچنین تابع F1, F2, F4 همانطور که مشخص است به خوبی همگرا شده و سریعتر به جواب بهینه رسیده است.

همانطور که در شکل­های بالا مشخص است، تابع F8 نتوانسته به خوبی همگرا شود و نتوانسته جواب بهینه را پیدا کند. همچنین تابع F6 همانطور که مشخص است به خوبی همگرا شده و سریعتر به جواب بهینه رسیده است.

**پیوست**

**کد فایل server.js**

*const* http = require('http');  
*const* log4js = require('log4js');  
*const* fs = require('fs');  
  
*const* logger = log4js.getLogger();  
logger.level = 'debug';  
  
*const* port = process.env.port || 5000;  
*const* sample = fs.readFileSync('sample.json').toString();  
  
  
*const* server = http.createServer((req, res) => {  
 *if* (req.url == '/search') {  
 res.writeHead(200);  
 res.write(sample);  
 res.end();  
 }  
}).listen(port);

**کد تابع cs\_levy\_fun**

function [z] = CS\_levy\_fun(n,m,beta)

% This function implements Levy's flight.

% For more information see

%Multiobjective cuckoo search for design optimization Xin-She Yang, Suash Deb

% Coded by Hemanth Manjunatha on Nov 13 2015.

% Input parameters

% n -> Number of steps

% m -> Number of Dimensions

% beta -> Power law index % Note: 1 < beta < 2

% Output

% z -> 'n' levy steps in 'm' dimension

% n = 10;

% m = 30;

% beta = 1.4;

num = gamma(1+beta)\*sin(pi\*beta/2); % used for Numerator

den = gamma((1+beta)/2)\*beta\*2^((beta-1)/2); % used for Denominator

sigma\_u = (num/den)^(1/beta);% Standard deviation

u = random('Normal',0,sigma\_u^2,n,m);

v = random('Normal',0,1,n,m);

z =((u./(abs(v).^(1/beta))));

end