Haladó algoritmusok féléves feladat

# Rövid összefoglaló:

A projekt működtetéséhez a **Plot projektet kell elindítani** a visual studion belül.  
Csináltam UI-t a könnyebb teszteléshez (az inputoknál nincs hibakezelés, csak számokkal működik), illetve bemutatáshoz. Az NSGA2 metódusnál a kirajzolás néha valamilyen okból kifolyólag egy-két pixelt elcsúszik, sajnos a hiba javítására nem volt időm, ezért előre is elnézést kérek.  
A metódusokat/algoritmusokat a HaladóAlg projekten belül lehet megtalálni. Konzolos megvalósítás csak eleinte tesztelésként volt.

# Utazó ügynök probléma megoldása genetikus algoritmussal:

Ordered crossoverrel történik a populáció népesítéséhez, mivel ez bizonyult messze a legsikeresebbnek, illetve mutációt is használok, amely úgy működik, hogy 50% eséllyel felcserél 1 dns párt a Génben, ha felcserélődik, akkor újabb 50% esélye van a mutációra. Ezzel próbáltam esélyt adni annak, hogy a lokális optimumból kissé jobban tudjon kimászni az algoritmus.  
A fitness igazából egy az egyben a távolságot jelenti. Bal felül egy counter látható a Plot ablakban, ami a generáció számát mutatja-  
Szerintem egészen jól működik olyan 80-100 faluig. Természetesen ha növeljük a populációt, akkor gyorsítjuk a konvergenciát.  
Igyekeztem párhuzamosítani ott ahol tudtam a gyorsítás végett.

# NSGA2:

Próbáltam írni egy viszonylag generális „Person” osztályt aminek jelen esetben 2 tulajdonsága van, de elméletileg (és gyakorlatilag is) sikerült generalizálni, tehát ha átírjuk, hogy 3 vagy 4 tulajdonságuk legyen akkor is jónak tűnt az algoritmus működése.  
Ezt igazából 1 az 1-be a jegyzet pszeudójából implementáltam. Egy kicsit módosításra volt szükség a Non-dominated sortnál, mert ott valamiért nem akart működni a pszeudó alapú implementáció.  
Amikor új populációt csinálok, akkor egy olyan crossover történik, ami veszi az átlagát a két szülő tulajdonságainak és azt adja a gyereknek. Ez a könnyebb tesztelés miatt használtam így(ha jól sejtem így működhetett a példa is). Bemeneti adataink nincsenek, hanem indulásnál randomizáljuk a „Personokat” és az lesz a kiindulásunk.  
Annyi változás történt még, hogy a User interfacen beállítható, hogy egy ciklus alatt hány generáció menjen le, majd ha lemegy egy ciklus akkor történik a kiíratás. Ezt érdemes viszonylag alacsonyra hagyni, mivel úgy jobban látszik a populáció konvergálása. Illetve a vizualizáció is csak 2 tulajdonságra működik.

# Legkisebb körbe írható poligon szimulált lehűléssel:

A user interfacen csak a kiinduló hőmérsékletet és a lehűlési rátát lehet megadni. A Ploton belüli kódnál pedig lehet hozzá adni még konstans pontokat illetve a poligonhoz csúcsokat.( Ez a MainWindow.xaml.cs -ben a 268 sortól lefelé lehet megtenni)  
Gyakorlatilag ezt is a pszeudó alapján implementáltam. Jól működik, ha megfelelően nagy a kiindulási hőmérséklet, és ha viszonylag nagy a lehűlés (Sajnos itt nem a valósához közeli értékek működnek jól.).  
A rossz megoldás elfogadásán kellett változtatnom, mivel túl későn kezdett el jó helyre konvergálni. Ezért gyakorlatilag a rossz eredmény elfogadásának esélyét csökkentettem 0.01-szeresére. De természetesen még így is elég sokáig fogad el „rossz” megoldásokat tehát van lehetősége kimászni a lokális optimumból.  
Illetve próbálkoztam azzal is, hogy a hőmérséklettel arányos nagyságúakat lépkedhessen, de mivel alapvetően randomizálva vannak a lépések (irány és méret szempontjából), valószínűleg ezért nem segített, hanem rontott a helyzeten és ezért nem konvergált eléggé még alacsony hőmérsékletnél sem (a kis lépések miatt). (A lehűlés miatt kicsit későn kezdődik a konvergálás.)  
Ezért maradtam a szimpla -0.5 és 0.5 közötti lépések randomizálásánál.