Genetikus algoritmusok párhuzamosítási lehetőségei

az implementáció bemutatása egy alkalmazási példán keresztül

Sári Zoltán
Neumann János Informatikai Kar
Óbudai Egyetem
Budapest, Magyarország
Email:sari.zoltan.tamas@gmail.com

***Kivonat*—Az NP-teljes problémák kezelésére alkalmas eszközt kínáló genetikus algoritmusok jellemzője a párhuzamos működés. Az eljárás futási idejének csökkentése érdekében a párhuzamossítási lehetőségek kihasználása kulcsfontosságú a több-, sokprocesszoros környezetben. A dolgozat célja, hogy hozzájáruljon a futási teljesítmény optimalizálásához alkalmas modell kiválasztásához. A dolgozat bemutatja a potenciális párhuzamosítási lehetőségeket, majd értékeli az egyes párhuzamosítási modellek implementálhatóságának előnyeit és hátrányait. Végül egy alkalmazási példán keresztül illusztrálja a párhuzamosítás implementációját.**

I. Bevezetés

A Holland[1] által bevezetett genetikus algoritmusok az NP teljes problémák megoldásában nyújtanak segítséget. [1] Az algoritmust a biológiai fejlődés folyamatai ihlették, az evolúció analógiájára épül és sokrétűen használható a mesterséges intelligencia területein (pl.: gépi tanulás). Genetikus algoritmusok használata különösen releváns ha [2]:

- probléma túl bonyolult, keresési tere kezelhetelenül nagy a kimerítő kereséshez
- probléma szerkezete nem ismert
- nem áll rendelkezésre egzakt, gyors algoritmus a probléma megoldására
- nincs igény az optimumra, elegendő egy elfogadhatóan jó megoldás

A genetikus algoritmus egy problémafüggetlen, globális optimalizáló eljárás, szemben a gradiens alapú módszerekkel, amelyek hajlamosak egy lokális optimumon beragadni.

A módszer fontos jellemzője, hogy a keresési tér független pontjait egyszerre vizsgálja, képes párhuzamosan működni. A párhuzamosság vértezi fel azokkal az előnyökkel, amelyek révén képes hatalmas méretű keresési terekkel dolgozni és nem ragad be egy lokális optimumba, végső soron alkalmas nem lineáris problémák megoldására.

A párhuzamos működés többprocesszoros környezetben hatékonyan támogatható, így a futási idő csökkentése érdekében a párhuzamossítási lehetőségek kihasználása kulcsfontosságú.

II. Működési elv

Az algoritmus a feladat összes lehetséges megoldását tartalmazó keresési tér minél rátermettebb elemeit hivatott felfedezni. A genetikus algoritmus a keresési tér egy részhalmazával, azaz egyszerre több potenciális megoldással (populáció), párhuzamosan dolgozik. Az algoritmus futása során a populáció újabb és újabb, időben együtt létező egyedekből álló generációja jön létre. Az időben egymást követő állapotokat nem csupán egy állapot módosításával, hanem több, a megoldás közelítése szempontjából rátermettnek ítélt állapot (szülő) összekombinálásával állítja elő a biológiai öröklődés mintájára. A működés alapelve, hogy a kombinálásnak köszönhetően minden generáció az előzőnél rátermettebb elemeket tartalmaz, így a keresés előrehaladásával egyre jobb megoldások állnak rendelkezésre.

A fentieknek megfelelően egy genetikus algoritmus iteratív eljárás. Minden iteráció során kiszámításra kerül az egyedek jóságfoka, amelyek alapján a rátermettség megítélhető és a legalkalmasabb megoldások a jellemzők örökítése érdekében kiválaszthatók. Az iteráció általában addig tart, amíg a probléma megoldása szempontjából kielégítő megoldás születik vagy a ciklusok száma elér egy előre meghatározott lépésszámot.

A genetikus algoritmus iterációjának sémája:

1) mesterséges kromoszóma szerkezetének kidolgozása
2) kezdeti populáció létrehozása
3) populációt alkotó egyedek rátermettségének értékelése
4) genetikus operátorok alkalmazásával új populáció létrehozása
5) a 3. és 4. lépés ismétlése a megállási feltételig

Egy egyszerű genetikus algoritmus formális modelljét Vose és Liepens[3] dolgozta ki.

III. Reprezentáció

A genetikus algoritmusok végrehajtása az iteráció megvalósításával, problémafüggetlen módon zajlik. A problémaspecifikus tudás érvényesítésére a reprezentáció ad lehetőséget. A genetikus algoritmusok alkalmazásának egyik sarokpontja a probléma szempontjából lényeges tulajdonságok kiválasztása és kódolása. A kiválasztott tulajdonságoknak a potenciális megoldásokra jellemző értékeit a kromoszómának nevezett (1) adatstruktúra kódolja és tárolja. Egy $c \in C$ kromoszóma (*chromosome*) az *n*-dimenziós tér attribútum (*gen*) halmazaiból felvett értékek rendezett sorozata.

$$C = A_1 \times A_2 \times \ldots \times A_n \tag{1}$$

[1] pl.: utazóügynök probléma, ütemezési feladatok, gráf színezés

Az eljáras végrehajtása egy kiindulási populáció létrehozásával kezdődik, amely általában véletlenszerűen megválasztott kromoszómákból áll.

Az algoritmus egy olyan mérési módszer alkalmazását feltételezi, mellyel az aktuális generáció elemeinek jóságát meg lehet határozni. A rátermettséget a jóságfüggvény jellemzi, amely a legalkalmasabb kromoszómák kiválasztásának alapját képezi.

$$fit : C \to \mathbb{R} \tag{2}$$

A (2) jóságfüggvény (*fitness function*) a genetikus algoritmussal megoldandó probléma célfüggvényéből indul ki, amelynek maximalizálására törekszik az eljárás. A függvény a lehetséges kromoszómák halmazán értelmezett, általában nem negatív értékkészletű leképezés. A feladat $f(c)$ célfüggvényének lineáris jósági függvényre való leképezésére a (3) általános alak használható.

$$fit(c) = af(c) + b \qquad a, b \in \mathbb{R} \tag{3}$$

A (4) függvények az egyedek rátermettségében jelentkező különbségek elnyomására, míg a (5) formulák a különbségek kiemelésére alkalmasak.

$$fit(c) = f(c)^{\alpha}, \quad fit(c) = ln(f(c)) \quad \alpha < 1 \tag{4}$$

$$fit(c) = f(c)^{\alpha}, \quad fit(c) = e^{f(c)} \quad \alpha > 1 \tag{5}$$

Gyakran a jósági függvény az egyedek célértektől való távolságából, azaz a hibából fejezhető ki. Ekkor a függvény alakja (6)-nak felel meg.

$$fit(c) = \frac{1}{1 + err_f(c)} \tag{6}$$

A struktúrában tárolt információk megőrzése, örökítése érdekében az algoritmus a kromoszómákon különböző műveleteket hajt végre.
Az alapoperátorok:

- szelekció (*selection*): új generáció létrehozása érdekében a legrátermettebb elemek kiválasztása, amelyeken a rekombinációs és mutációs operátorok alkalmazandók
- keresztezés (*crossover*) és rekombináció (*recombination*): több kromoszómából (szülőből) állítanak elő újabb kromoszómát, a szülők kódjainak valamilyen kombinációja segítségével
- mutáció (*mutation*): a kiválasztott egyed véletlenszerűen meghatározott attribútumának megváltoztatása (egy kromoszómából állít elő egy újabbat)

Az operátorok meghatározása során alapvető elvárás, hogy alkalmazásukkal a leszármazott generációk rátermettsége legalább ne csökkenjen.

A kiválasztás tipikusan az adott generációra vetített relatív jósággal arányos vagy egy meghatározott küszöb értéket meghaladó, csak a legjobbakat kiválasztó mechanizmus. Az egyik leggyakrabban alkalmazatott szelekciós operátor a rulett-kerék kiválasztás.

$$p_i^{select} = \frac{fit(c_i)}{F}, \quad \text{ahol} \quad F = \sum_{k=1}^{N} fit(c_k) \tag{7}$$

A (7) adott egyed kiválasztásának valószínűsége, (8) pedig a kiválasztás karakterisztikus függvénye.

$$f(c_i) = \begin{cases} 1 & \text{ha } p_i^{select} > r \\ 0 & \text{ha } p_i^{select} \leq r \end{cases} \tag{8}$$

ahol: $r \in [0, 1]$ véletlenszám.

A keresztezés operátorok speciális esetei az átrendező operátorok: részlegesen megfeleltetett keresztezés (*partially matched crossover, PCX*), átrendezéses (*order crossover, OX*) és a ciklikus keresztezés (*cycle crossover, CX*). A keresztezés operátorokat részletesen bemutatja Álmos et.al [4].

Az algoritmus tervezésekor a műveleteken túl meghatározandó további paraméterek[5]:

- kezdeti populáció mérete
- kereszteződésre kijelölt szülők száma
- kereszteződés során generált új egyedek száma
- kereszteződési pontok előfordulásának valószínűségi eloszlása
- mutáció alkalmazásának valószínűsége

Rugalmasságot jelent, ha a paraméterek futás időben változtathatók.

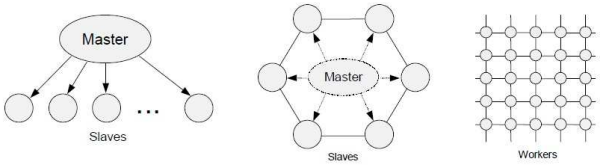
IV. Párhuzamosítási lehetőségek

A legtöbb keresési algoritmus egyszerre csak egy lehetséges megoldást tud vizsgálni, és csak egy irányba iterál. Ezeknél egy lokális optimum esetében nincs alternatív választási lehetőség, el kell vetni az addigi eredményeket és újra kezdeni az eljárást. A genetikus algoritmusok fontos jellemzője, hogy a keresési tér független pontjai egyszerre vizsgálja, képes párhuzamosan működni.

A genetikus algoritmusok generációszámának tipikus értéke 50 és 500 közötti[4], ezért a futási idő jó részét a rátermettségi értékek kiszámítása teszi ki. A jóságfüggvény megválasztásánál lényeges szempont, hogy minél kisebb számítási bonyolultságú legyen, azonban ennek a feladat komplexitásától függően korlátai vannak. Magas populáció és komplex jóságfüggvény esetén jelentősen nő a futási idő, így nehéz gyors eredményt elérni. A genetikus algoritmusok működésében természetszerűleg meglévő párhuzamosság kihasználása kulcsfontosságú[2]. A párhuzamosítás célja a futási idő csökkentése és/vagy a pontosság növelése. A párhuzamosítási lehetőség arra épül, hogy az adatok függetlenségéből eredően a populáció feloszthatható részhalmazokra, amelyek egymással párhuzamosan dolgozhatók fel.

[2]A jóságfüggvény mellett további lépés lehet a párhuzamosításban a csekélyebb számításigényű genetikus operátorok párhuzamosítása. Előfordulhat azonban, hogy globális információt igényel (pl.: rulettkerék módszerhez az összesített fitnesz érték). Az esetek nagy részében az operátorok párhuzamosítás nélkül is kellően gyorsak, a megnövekedett kommunikáció miatt elképzelhető, hogy az algoritmus összességében lassabb lesz.

A genetikus algoritmusok szekvenciális és párhuzamos végrehajtása közötti hatékonyságot számos kutatás elemzi[3][6][7][8][9].



1. ábra. A párhuzamos genetikus algoritmusok alapmodelljei[11]: (a) globális, (b) durva szemcsés, (c) finom szemcsés

A szakirodalom[4][12][13] a GA-k esetében a párhuzamosság kihasználásának, a populáció granularitása szerint 3 alapmodelljét különbözteti meg (1.ábra). Mindegyik modell az adatpárhuzamosságra épít és a kromoszómák részhalmazait rendeli valamilyen módon a végrehajtó egységekhez:

- globális populációk: egyetlen populáció
- szigetmodell (durva szemcsés): a teljes populáció felbontása egymással páronként diszjunkt részhalmazokra
- celluláris (finom felbontású): szélsőséges esetben minden részhalmaz egy kromoszómát tartalmaz

IV-A. Globális populáció

Globális populációt használva az egyedek kiértékelése, és/vagy a genetikai operátorok explicit módon párhuzamosítottak, amely által a kiválasztás jelentős mértékben gyorsítható.A párhuzamosság kihasználásának legközvetlenebb módja a kiválasztás során a versengő szelekció használata. Ahogyan a versengő szelekció lényege, hogy míg a hagyományos szekvenciális algoritmusnál a kiválasztást a rátermettségi érték alapján felállított globális sorrend dönti el, addig a párhuzamosított globális modellnél a kromoszómák páronként mérkőznek meg génjeik örökítéséért (átmeneti populációba kerülésért) és a mérkőzések kimenetele határozza meg a kiválasztást.

Minden végrehajtó egység két független mérkőzést rendez a populációból véletlenszerűen kiválasztott 2-2 egyed között, majd megtartja a mérkőzések győzteseit. A végrehajtó egységeknél tartózkodó egyedek adják az átmeneti populációt, ezután a rekombináció és a kiértékelés párhuzamosan történhet meg. A globális modell három alapvető megvalósítási módját emeli ki az irodalom[4].

*IV-A1. Szinkron mester-szolga.*A modell megvalósítása egy elosztott memóriával rendelkező architektúra, amelyben egy mesterprocesszor *k* darab szolga feldolgozó egységet irányít: vezérli az új egyedek létrehozását és a genetikus operátorok működését.

A populáció a mesterhez tartozó memóriában található. A mester osztja szét az egyedeket a szolgaprocesszorokhoz, amelyek egyszerű függvénykiértékelést végeznek, a populáció egyedeinek jósági mértékét számítják ki. Az értékeket a mester összegyűjti, majd alkalmazza a genetikus operátorokat az új generáció előállítása érdekében.

IV-A2. Félszinkron mester-szolga. A szinkron mester-szolga megvalósítás egyik gyengesége a robosztusság hiánya. Ha a mester meghibásodik, a rendszer működésképtelenné válik. További hátrány, hogy az időveszteség csökkentése szempontjából szűk keresztmetszetet jelent a mesterprocesszor és a leglassabb szolga. A működés során teljesítménycsökkentő a szolgák abból fakadó tétlensége, hogy mindaddig várakoznak, amíg a mester dolgozik. Másrészt jelentős időveszteséget eredményeznek a jósági függvény kiértékelési idejében mutatkozó különbségek. A félszinkron modell a szűk keresztmetszetek csökkentését gyengébb szinkronizációval éri el. A következő egyed kiválasztását és populációba illesztését, azonnal elvégzi, amikor egy szolga befejezi tevékenységét.

IV-A3. Elosztott, aszinkron konkurens. Az elosztott, aszinkron architektúra fő komponenseit egy osztott hozzáférésű központi memória és *k* egyenértékű processzor alkotja. A feldolgozó egységek egymástól függetlenül végzik a genetikus műveleteket és a jósági függvény kiértékelést, egy közös osztott memórián keresztül.

Az aktuális generáció egyedei az osztott közös memóriában találhatók. Az ütközések elkerülése érdekében biztosítani szükséges, hogy minden processzor csak a számára kijelölt egyedhez férhessen hozzá és jósági mértékét függetlenül számíthassa ki. A modell a generációk között szinkronizációt igényel. Nehezebb megvalósítani mint a mester-szolga modelleket, de a mester kiesésével járó hátrányok kizárásával a rendszer megbízhatósága nagyobb.

IV-B. Sziget modell

A szigetmodell alapján több populáció fejlődik egyszerre párhuzamosan, egymástól elszigetelve. Minden alpopuláción egy önálló genetikus algoritmus fut. Az alapvető operátorok alkalmazása csak az alpopuláción belül történik, de szükség van egy új, migrációs operátor bevezetésére. Egy előre meghatározott szabályrendszer szerint minden egyes populációból a legrátermettebb egyedek vándorolnak a szigetek között. A migráció paramétereként meg kell határozni a migrálandó egyedek számát (migrációs ráta) és a migráció gyakoriságát (intervallum). A migráció révén lehetőség nyílik arra, hogy az eltérő kezdeti állapotokból induló és különböző irányokba tartó populációk egymás információit felhasználhassák. A migrációs modell egy tipikus esete, hogy minden éppen sorra kerülő sziget átadja a legjobb egyedét az összes többi populációnak. Alkalmazható olyan migráció is, amelynek során az alpopulációk egy gyűrű mentén felfűzve adnak át információt egymásnak.

A migráció gyakoriságának paraméterezése során tekintettel kell lenni az alábbiakra:

- túl gyakori egyed vándorlás esetén, a nagyfokú globális keveredés miatt a szigetek lokális különbségei elvesznek
- túl ritka migráció mellett nem lesz elég a szigetek közötti információcsere, ezért az egyes alpopulációk egymástól elkülönülő optimumhoz fognak konvergálni

A migráció akkor a legsikeresebb, ha az alpopulációk már konvergáltak. A konvergencia előtt még a szigetekben sem terjednek el a sikeres gének, utána pedig időveszteséget eredményez a csekély marginális konvergencia. A migráció gyakori megvalósításaként, rendszeres időközönként (bizonyos számú generációnként) minden feldolgozó egység kihirdeti legjobb egyedét a többieknek. Ez a megvalósítás egy lazán csatolt architektúrát eredményez. A *k* független processzoron 1-1 hagyományos genetikus algoritmus fut független memóriában, független operátorokkal és jósági mérték kiértékeléssel. A végrehajtó egységek között csak a kommunikációt kell biztosítani üzenetek segítségével A független processzorok autonómiája miatt a rendszer megbízhatósága magas és az időveszteségek nagy része kiküszöbölhető. A modellt bővebben tárgyalja Whitley et al.[10].

IV-C. Celluláris modell

A celluláris megközelítés során az alpopuláció rendkívül kicsi, gyakran 1 egyedből álló populációt jelent. A populációk kétdimenziós rácsba szerveződnek és minden részpopuláció csak lokálisan, a szomszédjaival kommunikálhat. A kommunikáció véletlenszerűen vagy rátermettség érték alapon történik, amely során az elem választ egyet a szomszédjai közül és azzal kereszteződve létrehozza az új egyedet és lecseréli magát (pl.: mindegyik végrehajtó egység veheti a legjobb fitneszű egyedet a lokális környezetében).

Mivel a szelekció és a keresztezés a szomszédságra korlátozódik, több generáció után kisebb lokális csoportosulások alakulhatnak ki, lokális evolúciós irányokkal. Mivel az üzenetek környezetét csak a szomszédok alkotják, a kommunikációs kötöttségek mérsékeltebbek, mint a másik két modellben. Az architektúra előnyei a mátrixos szervezésű sokprocesszoros számítógép környezetben használhatók ki (pl. GPGPU).

V. A PÁRHUZAMOSÍTÁSI MODELLEK ÉRTÉKELÉSE

Mivel a genetikus algoritmusok esetén a populáció egyedei egyszerre, egymástól függetlenül léteznek, a párhuzamosság elvben egyenesen arányos a populáció méretével. A párhuzamosság kihasználásának korlátját jelentik azonban, hogy a populáció egyes részhalmazai között szinkronizáció szükséges. Hatékony a párhuzamosítás, ha az egyedeket úgy képezi le feldolgozóegységekre, hogy maximalizálja a párhuzamosságot és csökkenti a felesleges kommunikációt közöttük. Az egyes modellek értékelése öt kiemelt szempont szerint foglalható össze (I. táblázat).

A globális modellben kezelni kell a rátermettség értékek begyűjtését és összesítését, valamint a holtidők minimalizálása érdekében csökkenteni kell a végrehajtó egységek várakozását. A globális modell hátránya abból adódik hogy a szolgaprocesszoroknak meg kell várniuk, amíg a mester összegyűjti a fitnesz értékeket és létrehozza a következő generációt. Előnye, hogy könnyen implementálható és nem igényel speciális hardvert. A modell akkor ajánlott, ha a kiértékelés és a genetikai operátorok nagy számításigényűek.

modell/ szempont	**globális**	**sziget**	**celluláris**
párhuzamosítás mélysége	egyedek kiértékelése	teljes genetikus algoritmus	teljes genetikus algoritmus
szemcsézettség	egy globális populáció	több önálló populáció	egyedek mátrix elrendezésben
szinkronizáció	értékek összegyűjtése, új generáció létrehozása	migráció, az alpopulációk legjobb egyedeinek kihirdetése	kommunikáció csak a lokális környezetben
architektúra	hagyományos mester-szolga	lazán csatolt, autonóm processzorok (MIMD)	mátrix szervezesű, sokprocesszor (GPGPU)
szűk keresztmetszet	időveszteség, várakozás a mesterre, leglassabb szolgára	migrációs ráta és intervallum helyes megválasztása	elegendően sok processzor hiánya

I. táblázat. Az egyes párhuzamosítási modellek főbb jellemzői

A több populációra épülő modellekben többek között a kommunikáció és a migráció jelent kezelendő problémákat:

- figyelmet igényel a populációk méretezésénél, hogy a feldolgozóegységek mind kihasználtak legyenek
- az egymástól független szigetek közötti kommunikáció bonyolítja a modellt
- bizonyos evolúciós lépések nehezen reprodukálhatók a populációk feldolgozásának és a migrációk aszinkron volta miatt

Előnyük, hogy az alpopulációk hamarabb konvergálnak, mint a globális populáció. Ha a feladat jól felbontható, akkor a különböző alpopulációk részmegoldásai a migráció segítségével állnak össze egy jobb globális megoldássá, mint a szekvenciális algoritmus eredményeként. A szigetmodell jól implementálható és skálázható többprocesszoros (MIMD) környezetben. A celluláris modell pedig sok végrehajtó egység esetén bizonyul hatékonynak. (pl.: 50x50 rács esetében 2500 feldolgozó egység).

VI. A BERUHÁZÁS-PORTFOLIÓ INTERTEMPORÁLIS OPTIMALIZÁLÁSÁNAK FELADATA

A vállalatok hosszú távú, stratégiai pénzügyi döntéseinek egyik alapvető problémáját az optimális beruházási portfólió összeállítása jelenti. A tervezés célja, hogy adott beruházási lehetőségek, tőkekorlát és tolerált kockázat mellett egy adott időszakra vonatkozóan meghatározza a beruházás-portfólió egy olyan összetételét, amelynek jövedelmezősége meghalad egy elvárt hozamszintet. A feladat visszavezethető a klasszikus hátizsák problémára, így NP-teljes. A hátizsák problémához képest a komplexitást növelik az erőforrás korlát feltételek, a beruházási lehetőségek függőségei és az intertemporális allokáció. A feladat megoldására hatékony eszközt kínál a genetikus algoritmus. A probléma az alábbiak szerint strukturálható.

Bemenet:
$X \neq \emptyset$ – a beruházási lehetőségek véges halmaza
ahol $\forall x \in X$ beruházás esetén adott

$i(x) \in \mathbb{Z}^+$	tőkeigény
$npv(x) \in \mathbb{Z}$	várható hozam (nettó jelenérték), $npv(x) \sim N(m, \sigma^2)$ eloszlása normális eloszlást követ
$\sigma(x) \in \mathbb{R}^+$	kockázat, hozam szórása [3]
$t_{max}(x) \in \mathbb{N}^+$	megengedett legkésőbbi megvalósítás periódusa
$D(x) \subset X$	azok a beruházások, amelyektől függ a megvalósítás
$x' \in D(x) < x$	részben rendezés X-en: x'-t hamarabb kell elvégezni, mint x-et

továbbá minden beruházás 1 periódus alatt megvalósítható további inputok:

$n \in \mathbb{N}^+$	periódusok száma
$\mathbf{I}_{max} \in \mathbb{Z}^{+^n}$	erőforrás korlát vektor, időszakonként maximálisan befektethető összeg
$NPV_{min} \in \mathbb{Z}^+$	elvárt hozam összege a teljes időszak alatt
$\alpha \in [0, 1]$	konfidenciaszint
$VaR_{\alpha,max} \in \mathbb{Z}^+$	tolerált kockáztatott érték
$r \in \mathbb{R}^+$	diszkontkamatláb, a korábbi megvalósítás priorizálása érdekében (NPV diszkontálása)

Feladat:

$P \subseteq X$	beruházási lehetőségek részhalmaza, ahol $\forall x \in P$ beruházás esetén adott $s(x) \in \mathbb{N}$ a beruházás megvalósításának periódusa
$NPV(P) \in \mathbb{R}^+$	a portfólió összesített hozama $NPV(P) = \sum_x npv(x) * (1+r)^{-s(x)}$, $x \in P$
$\mathbf{I}(P) \in \mathbb{Z}^{+^n}$	befektetés vektor, a portfólió befektetési igénye periódusonként $I_i(P) = \sum_x i(x)$, $x \in P$, $s(x) = i$
$VaR_\alpha(P) \in \mathbb{R}^+$	a portfólió összesített kockázatott értéke $Var_\alpha(P) = \sum_x VaR_\alpha(x) * (1+r)^{-s(x)}$, $x \in P$

A feladat egy olyan P portfólió megkeresése, amely esetében teljesülnek az alábbi megszorítások.

Feltételek:

1	erőforráskorlát	$\mathbf{I}(P) \leq \mathbf{I}_{max}$
2	sorrendi korlát	$s(x') < s(x)$ ha $x' < x$, $x', x \in P$
3	időbeli korlát	$s(x) \leq t_{max}(x)$
4	kockázat korlát	$VaR_\alpha(P) \leq VaR_{\alpha,max}$
5	cél	$NPV(P) \geq NPV_{min}$

Az el nem költött $\mathbf{I}_{max} - \mathbf{I}(P)$ összegek nem csoportosíthatóak át az időszakok között. Felfogható úgy, hogy a megtakarítás alternatív eszközökbe kerül befektetésre (pl. államkötvények) vagy a vállalkozás tulajdonosai részére kerül kifizetésre osztalékként.

A feladat megoldásával válasz kapható arra, hogy lehet-e legalább egy olyan portfóliót összeállítani, amelynek hozama elér egy elvárt szintet, emellett tőkelekötési igénye nem haladja meg a vállalkozás rendelkezésére álló kereteket, a függő beruházások csak a függőséget okozó beruházást követően valósulnak meg és a portfólió kockáztatott értéke nem ér el egy meghatározott szintet.

VII. Implementáció

VII-A. Kromoszóma

Az optimalizálási feladat reprezentációjának egyik sarokpontja az egyes beruházási lehetőségek és a beruházási portfólió összeállításának, ütemezésének kódolása. A II. táblázat a beruházási lehetőségek adatstruktúráját foglalja össze a bemutatott modellnek megfelelően.

attribútum	**leírás**
name : string	azonosítás
capex : int	a beruházás tőkeigénye
npv : int	a beruházásból származó jövedelem (nettó jelenérték)
deviation : int	a jövedelem szórása, a kockázat jellemzése
deadLine : int	a beruházás elvégzése esetén az utolsó elfogadható periódus
schedule : int	a optimalizálás során a beruházáshoz rendelt megvalósítási periódus
isDetermined : bool	igaz esetén, megvalósítási kényszert jelent

II. táblázat. A beruházási lehetőségek adatstruktúrája

Az optimalizálási feladat reprezentációja során rögzített hosszúságú, egészértékű kódolás került alkalmazásra a kromoszóma tekintetében (2. ábra).

2. ábra. A feladat kromoszóma reprezentációja

A kromoszóma minden egyes génje egyértelműen meghatározza, hogy egy beruházási lehetőség része-e a beruházási portfóliónak vagy sem (gén értéke 0-át vesz fel). Ha egy adott beruházás a portfólió részét képezi, akkor a kromoszóma adott génje hozzárendeli a megvalósítás ütemezését, azt a periódust, amelyikben a megvalósítás javasolt. A populáció inicializálásakor a kromoszómák génjei véletlenszerűen kerülnek feltöltésre, amelynek alsó határa 0, maximális érték pedig a feladat paramétereként megadott periódusok száma.

VII-B. Kiértékelés

A kiértékelés konkrét megvalósításának alappillérét a probléma megszorításai (erőforrás, időbeli, kockázat) és a megszorítások mellett a jövedelem maximalizálását célzó szempontok adják. A fitnesz értékek kiszámításához a feladatspecifikus bemeneteket (beruházások halmaza, megszorítások) és a portfólió jellemzői (portfólió teljes forrásigénye, teljes jövedelme és kockázata) szükségesek. A fitnesz értéket kalkuláló függvény az alábbi szempontokat veszi figyelembe:

– tőkekorlát (capex): a tőkekorlátot megsértő periódusok száma (távolságfüggvény a korlátot meg nem sértő állapottól)

– nettó jelenérték maximalizálása: az értékelés alapja a portfólió által elért és a paraméterként megadott elvárt jövedelem hányadosa
– kockázatminimalizálás: a paraméterként megadott, konfidencia szinttől függő tolerált kockázat (VaR) és a portfóliót jellemző kockázat minél nagyobb különbségének priorizálása
– megvalósítási határidő betartása: távolságérték a portfólió elemei által megsértett határidők függvényében
– megvalósítási determináció: a megvalósításra kijelölt elemeket nem tartalmazó portfóliók büntetése

A fitnesz függvény lineáris, a fenti tényezők paraméterezett súlyokkal vehető figyelembe. A fitnesz értékek kiszámítását követően az értékek normalizálásra lineáris skálázás alapján kerül sor. A skálázás után kapott értékek beillesztéses eljárással kerülnek rendezésre.

VII-C. Genetikus operátorok

A kiválasztás a lineáris sorrendezésen alapul: a genetikus algoritmus paramétereként megadott számú, legmagasabb jósági értékkel rendelkező egyed kerül szülőként kiválasztásra. Az új populációba átlapolt, a kiválasztott szülők automatikusan bekerülnek, a populáció további egyedei a keresztezés után kerülnek feltöltésre. A keresztezés alapértelmezetten az egyszerű egypontos keresztezés, de a keresztezési pontok számára vonatkozó paraméter megváltoztatásával többpontossá tehető. A feladat természetéből adódóan a permutáció megőrzése nem szükséges. A mutáció a valószínűség génenként történő figyelembe vételével valósul meg (szintén paraméterezhető).

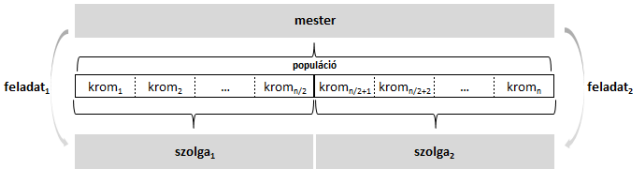
VIII. Párhuzamosítás

VIII-A. Választott modell

Az algoritmus párhuzamosítási lehetőségeinek három alapvető módszere közül a párhuzamosítás választott modellje a globális szinkron mester-szolga. A választott modell mellett szól, hogy

– az optimalizálási eljárás szekvenciális futási idejének jelentős részét a viszonylag összetett fitnesz függvény és a kiértékelés teszi ki,
– a fejlesztési és futtatási környezetet egy kétmagos, hagyományos desktop gép adta, így a tipikusan sokmagos hardverkörnyezetet igénylő, több populációs modellek előnyeinek kihasználása korlátozott, és
– a modell könnyen implementálható (szoftvertechnológiai szempontból)

A szekvenciális megvalósításhoz képest változást jelent, hogy a fitnesz értékek kiszámítására irányuló kérés a mester feladatot ellátó objektumhoz érkezik, amely gondoskodik a populáció particionálásáról és a független partíciók egyedeinek kiértékelésével járó feladatokat kiosztja a szolga objektumok részére. A feldolgozást minden szolga külön szálban végzi. A feladatok kiosztását követően a mester megvárja a szolgák feldolgozási feladatának befejezését, majd begyűjti és összesíti a rátermettségi értékeket. Ezután a következő generáció létrehozása érdekében a kiértékelt populációt továbbítja a szekvenciális feldolgozás számára. A szolgák ugyanazon populáció ugyanazon objektumát, de annak eltérő partícióit (kromoszómáit) dolgozzák fel, így az implementáció adatfüggőségekkel nem jár. A feldolgozandó memóriaterületet a feladat objektum határolja be.(3. ábra).



3. ábra. A párhuzamosítás implementációja

VIII-B. Teljesítményjavulás

A fejlesztői és tesztelési környezet főbb paraméterei:

– Microsoft .NET Framework v4.5
– Microsoft Visual Studio Express 2013 for Windows Desktop
– Intel Core2Duo E4500 2,20 GHz
– 2 GB RAM
– Windows 7 Pro 64bit

A futtatási teljesítmény mérésére 6 szcenárió mentén került:

– 2 tesztállomány: egy 20 és egy 50 beruházási lehetőségből (kromoszómák hossza) álló bemenet
– 3 forgatókönyv a genetikus paraméterekre:
  – iteráció: 1000, populáció mérete: 200
  – iteráció: 2000, populáció mérete: 400
  – iteráció: 4000, populáció mérete: 800
– 3 forgatókönyv a párhuzamosításra: 1, 2 és 4 szolga objektumot és szálat használva

A futási teljesítmény mérésének eredményeit a III. táblázat foglalja össze.

kromoszóma hossza / iterációk száma / populáció	**átlagos futási idő (s)**			
	1 szál (a)	**2 szál (b)**	**4 szál (c)**	**(c)/(a)**
20 / 1000 / 200	6,7	6,7	5,9	88%
20 / 2000 / 400	32,1	29,2	25,8	81%
20 / 4000 / 800	141,9	147,3	138,8	98%
50 / 1000 / 200	19,8	14,4	13,1	66%
50 / 2000 / 400	81,6	54,3	56,0	69%
50 / 4000 / 800	367,7	267,6	271,1	74%

III. táblázat. A mért futási teljesítmény

A mérési adatok alapján a tesztkörnyezetben a párhuzamosítás előnye a kromoszómák hosszával arányosan mutatkozik meg. Nem meglepő a jelenség, hiszen a szinkronizációval járó relatív holtidők csökkennek, ha egy-egy szál hosszabb ideig dolgozhat egy kromoszóma adott (hosszabb) partícióján.

IX. Konklúzió

A genetikus algoritmusokban rejlő párhuzamosítási lehetőségek kihasználására nem javasolható általános modell. A hatékony megoldás kiválasztása során tekintettel kell lenni a feladat jellege mellett az eljárás kommunikációs és szinkronizációs igényre, valamint a hardverkörnyezet architektúrájára.

A globális modell speciális hardver hiányában, a kiértékelés és a genetikai operátorok nagy számításigénye esetén javasolható. A szigetmodell többprocesszoros, MIMD környezetben implementálható eredményesen. A celluláris modell pedig nagyon sok, százas nagyságrendű végrehajtó egység rendelkezésre állása esetén előnyös.

Hivatkozások

[1] J. H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.

[2] I. Futó, Mesterséges intelligencia, Aula Kiadó, 1999.

[3] M. D. Vose, G.E. Liepens, Punctuated Equilibria in Genetic Search, Complex Systems, vol. 5, pp. 31–44, 1991.

[4] A. Álmos, S. Győri, G. Horváth, K. A. Varkonyiné, Genetikus algoritmusok, Typotex Kiadó, 2012.

[5] E. Sántáné-Tóth,M. Bíró, G. András, A. Kő, L. Lovrics, Döntéstámogató rendszerek, Panem Könyvkiadó, 2008.

[6] A. Chipperfield, P. Fleming, Parallel Genetic Algorithms, Parallel and Distributed Computing Handbook, pp. 1118-1143, MacGraw-Hill, 1996.

[7] D. E. Goldberg, Sizing Populations for Serial and Parallel Genetic Algorithms, Proceedings of the 3rd ICGA, pp. 70-79, Morgan Kaufmann, 1989.

[8] H. Mühlenbein, Evolution in time and space - the parallel genetic algorithm, Foundations of Genetic Algorithms pp. 316–337, Morgan-Kaufmann, 1991.

[9] S. Szénási, Adatpárhuzamos sejtmagkeresĂŠsi eljárás fejlesztése és paramétereinek optimalizálása, OE doktori disszertáció, 2013.

[10] D. Whitley, S. Rana, R. B. Heckendorn, The Island Model Genetic Algorithm: On Separability, Population Size and Convergence, Journal of Computing and Information Technology, vol. 7, pp. 33–47, 1998.

[11] E. Alba, B. Dorronsoro, Cellular Genetic Algorithms, Operations Research/Computer Science Interfaces Series, Springer Verlag London Ltd., 2008.

[12] E. Alba, M. J. Troya, A Survey of Parallel Distributed Genetic Algorithms, Complexity, vol. 4, pp. 31–52, 1999.

[13] E. Cantú-Paz, A Survey of Parallel Genetic Algorithms, Calculateurs paralleles, reseaux et systems repartis, vol. 12, pp. 141–171, 1998.