Az épületek tartószerkezeti analízisének és optimalizálásának egyfajta szakaszos megközelítése

Turda Dan, Cucu Liviu, Dr. Gobesz Zsongor,
Dr. Chiorean Cosmin, Homorodean Daniel
Kolozsvári Műszaki Egyetem, Románia

1. Bevezető

A számítástechnika fejlődése természetszerűen maga után vonja más tudományterületek eszköztárának a minőségi és mennyiségi bővülését, ezáltal lehetőséget teremtve új elméleti modellek alkotására, illetve alkalmazására is. Az e cikkben vázlatos leírásra kerülő kutatás még jóformán gyerekcipőben jár. Mind a bemutatásra kerülő eszköztár, mind maga a nézőpont, valamint az eddig meghatározott jellemzők valószínűleg többszörös átalakuláson illetve fejlődésen fognak átesni az elkövetkező hónapokban. Az eszköztár saját szerkesztésű komponensei több programozási nyelven íródtak (Java, C, C++, perl és XML), többségük jelenleg is kísérleti fejlesztés alatt ál, ezért még nem üzemképesek .

1.1. Genetikai algoritmusok és genetikai programozás

A genetikai algoritmusok (GA) legfőbb jellemzője a biológiai fejlődés utánzása, vagyis: a legmegfelelőbb túlélése. Ennek megfelelően, egy bizonyos népességi alakzatra – több tervezési szempont figyelembevételével – véletlenszerű operátorokat alkalmazunk (determinisztikus operátorok helyett) a körülhatárolt népesség minden egyes tagja esetében. Ezek az operátorok, mint például a mutáció, az átfedés vagy a kiválasztás, biztosítják a szükséges egyedek népességen belüli számát, egyúttal biztosítva a legmegfelelőbb egyedek túlélését a következő generáció kialakulásához.

Lépésenként fogalmazva egy ilyen algoritmus a következőképpen íródna:

01. Tartószerkezetek véletlenszerű népességének a létrehozása;
02. Minden egyed kiértékelése, a feladat megszorításai, valamint a kizáró szabályok alapján;
03. Minden egyed megfelelésének a kiértékelése és a legmegfelelőbb egyedek kiválasztása;
04. Újrateremtés a mutáció és az átfedés genetikai operátorok alkalmazásával;
05. Visszatérés a második lépéshez, ha a kimeneti feltétel nem teljesül.

A genetikai megközelítés magába foglalja a genetikai programozást is. Míg a genetikai algoritmusok egy kromoszómát sorként (vagy egyéb lineáris alakzatként) ábrázolnak, a genetikai programozás az egyedek adatait ágazatként próbálja kezelni. Ezen ágazatokat pedig következetesen fel lehet használni az egyes egyedek genetikai adtainak a módosítására, egy jobb hatékonyság elérése érdekében. A genetikai programozás alkalmazása figyelemre méltó a tudáshalmazok (tudás-bázisok) keretén belül, amennyiben egy genotípusra alkalmazott eljárás általában egy jobb egyedet eredményez.

1.2. Algoritmusok gradiens-alapú eljárásokhoz

Egy ilyen algoritmus a következőképpen festene:

01. A javasolt tartószerkezet elemzése (belső erők, elmozdulások, feszültségek stb.);
02. Minden egyes tervezési változó tartalmi átértékelése egy csekély mennyiséggel;
03. Az érzékenység (befolyásoltság) elemzése;
04. A tartószerkezet megváltoztatása a kapott érzékenység alapján;
05. Visszatérés az első lépéshez, amennyiben a kimeneti feltétel nem teljesül.

A gradiens-alapú eljárást szekvenciális kvadratikus programozási (*SQP*) eljárásként határozzuk meg [2]. Az efféle eljárások a helyi görbületi információkat használják fel (melyek az eredeti függvények linearizálásából származnak), az optimizálási folyamat során kapott pontokban a tervezési változók figyelembevételével.

Helyi optimizáláshoz ezeket az algoritmusokat folytonos tervezési változókkal vesszük figyelembe. Egy esetleges példaként megemlíthetjük a gerendákat egy sík keretszerkezetben, mely esetben a gerendákon keresztmetszeti optimizálást hajthatunk végre. A tartószerkezet alkati optimizálása során a gerendákat kizárhatjuk a véletlenszerű generálásból, időt nyerve ezáltal.

Ezek az algoritmusok nem szükségeltetnek ágazatos ábrázolást.

1.3 Különálló változók

Egy ágazat-és-kötelék algoritmus van leírva a [4][1]-ben. Az eljárás tágasan alkalmazható döntéshozó feladatokban, ahol az eredmények csak átrendezés által viszonyulnak egymáshoz vagy többszörös objektív függvényeknek vannak alávetve.

Röviden, egy ilyenfajta algoritmus: inicializálást, particionálást és kiértékelést, kizárást illetve teljesülést foglal magába.

Ezek az algoritmusok nagyon alkalmasak ágazatos ábrázoláshoz.

1.1 1.4 Egy pár szó a tudás-alapú rendszerekről

Egy tudás-alapú rendszer (*KBS*) nagyon összetett objektum. Bár nem áll szándékunkban meghatározni e cikk keretében, hogy mit is jelent egy tudás-alapú rendszer, meg kell azért említenünk, hogy a mi szempontunkból melyek azok a leglényegesebb jellemzők, amelyek egy ilyen rendszer kéne kínáljon.

Egy tudás-alapú rendszer lehetőséget kéne nyújtson:

(i.) a tudás tárolására, többszörös összefüggések (elágazások) és feladatkörök szerint;
(ii.) keresésre;
(iii.) frissítésre.

Nem térünk ki arra, hogy miként születnek, tárolódnak vagy módosulnak az ágazatok. Viszont, hadd szemléltessünk egyszerű példaként egy többé-kevésbé XML-ben írt tudást ábrázoló ágazati részletet (lásd az 1.1-es ábrát).

```
01 <frame>
02 ...
03    <action>
04    ...
05       <increase>
06          <beam:analysis="elastic", cause="failure",\
07          value="yes, what="height", check="Iy_Iz_ratio"/>
08       </increase>
09    ...
10    </action>
11 ...
12 </frame>
13 ...
14 <Iy_Iz_ratio>
15    <seismic_strong maxvalue="3"/>
16    <seismic_medium maxvalue="5"/>
17    <seismic_weak   maxvalue="10" check="stability_local"/>
18    ...
19 </Iy_Iz_ratio>
20 ...
21 <stability_local>
22    <rect_sect action="null"/>
23    ...
24 </stability_local>
```

1.1

Egy egyszerű XML példa.

Persze, egy ilyen ágazatos ábrázolás kialakítása nagyon sok munkát igényel. Mérnöki körökben viszont számszerint aránylag kevés szó, kifejezést illetve adatábrázolási módot használunk. Amennyiben ezeket az elemeket sikerül jelentések és problémakörök szerint rendszerezni, akkor tudás- illetve szabály-forrást alkothatnak egy szakértői rendszerhez. Miután sikerül a tudás-bázist létrehozni, ez kell majd a különböző tervfuttatások során az összegyűlt tudást az általunk megszabott utasítások alapján beiktassa. Amint ezt az adathalmazt be iktatják és szervezik, értékes információforrást fog képezni mind az emberi szakértők, mind a gépi programok számára.

[1] Az eredeti kutatás szerzői Herve[5] és Roy[6]. Az ágazat-és-kötelék algoritmusok területe nagyon szerteágazó, mivel nagyon sok kutatást és fejlesztést foglal magába. Mivel tetszetősnek találjuk a szerkezetét és feltételezzük, hogy jó eredmények éerhetőek el vele, tesztelni szeretnénk a fenti művekben említett algoritmust.

2. Ember-gép kölcsönhatás

Bár az ember-gép kölcsönhatás olyan pszichológiai kérdéseket is érint, melyek messze állnak kutatási céljainktól, néhány megjegyzést mégis fontosnak tartunk ejteni róla .

Egy terv szervezett lebonyolítása során tudás- és képességszerzés történik. Nevezhetjük ezt egyszerűen tapasztalatnak. A mérnöki munka esetében a tapasztalat fontos szerepet játszik az elemzési folyamatban és nagyon fontosnak bizonyul az optimalizáló folyamatokban.

Nincs szándékunkban az emberi szakértelmet úgynevezett gépi szakértelemmel helyettesíteni. Bár korábban már említést tettünk a szakértői rendszer fogalmáról, a következő paradigmához kívánjuk kötni magunkat:

szakértői rendszer = (emberi kreativitás + emberi képességek + emberi tudás) + gép – tárolt tudás (1)

Ez a korlátozás a saját értékelésünkből adódik a jelenlegi kutatás illetve ipari implementálás szintjeire alapozva [1]. Véleményünk szerint a tudás-alapú rendszerek nem nyertek eddig sem tágas sem elégséges leszűkített alkalmazási területet. Mérnöki körökben pedig eddigi megfigyeléseink szerint aránylag kevés efajta technikát karoltak fel.

Mindezek mellett létezik egy félig-meddig filozófiai értékelés is. Azok a próbálkozások, melyek összességükben az emberi tapasztalatot a gépivel próbálják helyettesíteni, a *soft computing* (lágy programozás) fogalmához vezetnek. Ez a fogalom abban különbözik a szokványos (*hard*) programozástól, hogy ez utóbbitól eltérően az előbbi eltűri a pontatlanságot, a bizonytalanságot és a részleges igazat. Végül is ez természetes, hiszen a lágy programozás által követett modell az emberi agy működését próbálja utánozni [3].

Bár egyetértünk azzal, hogy a lágy programozás, vagy még inkább a *fuzzy* rendszerek, fontos szerepet játszanak a tartószerkezeti elemzés és optimizálás területén, mégsem tudják helyettesíteni az előbbi ((1)-esel jelölt) kifejezésben az emberi alkotóképességet és találékonyságot. Ahhoz, hogy a *fuzzy* ábrázolást valamennyire érzékeltessük, elég, ha az 1.1-es ábrával szemléltetett példában a 15., 16. és 17. sort megváltoztatjuk. Az eredmény a következő (2.1-es) ábrán követhető.

```
...
15 <seismic_strong maxvalue="3" tolerance="10%"/>
16 <seismic_strong maxvalue="5" tolerance="20%"/>
17 <seismic_strong maxvalue="10" tolerance="30%"\
      check="stability_local,stability_general"/>
...
```

2.1

Tűréshatár bevezetése, fuzzy-bb változat.

Ez a megközelítés lehetővé teszi, hogy amennyiben népességként kezeljük az érintett halmazt, több olyan egyedet kapjunk, amelyik megfelel a kiválasztási feltételeknek. Ennek a változatosságnak a lecsökkentése pedig az ember feladata lenne.

3. Használt eszköztár

3.1. Egy pár szó a nyitott rendszerekről

A számítástechnika különböző területein a nyitott rendszerek (*Open Systems*) fogalmát nagy általánosságban olyan programcsomagokra alkalmazzák, amelyek magukba foglalják, minden felhasználó számára hozzáférhetően, a forráskódot is.

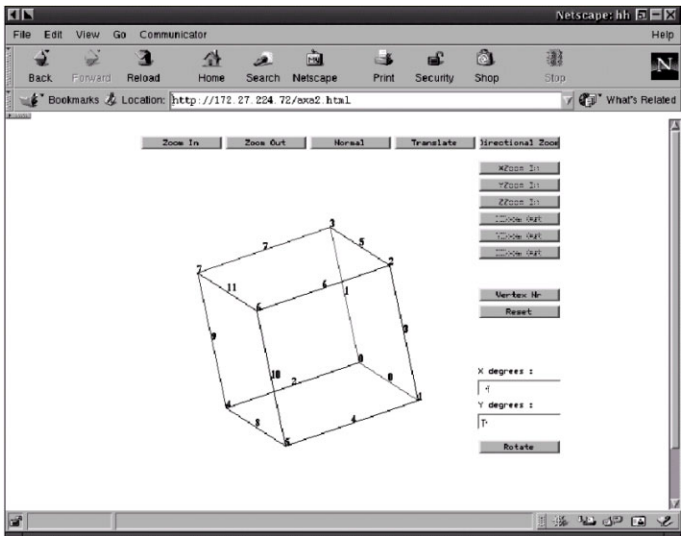
Ez a forráskód tulajdonképpen egy adag tudás. Saját tapasztalatunk szerint legalább olyan nehéz egy jó könyvet írni, mint egy jó programcsomagot. Ugyanakkor senki sem állíthatja teljességgel azt, hogy amit alkotott az mindenekfelett eredeti és egyedi (bár kikerüljük a tudományos kutatás berkeiben kerengő misztikus vitákat, ki kell hangsúlyoznunk, hogy az abszolút igazság valahol felettünk van – példának okáért ez a cikk is tartalmazhat hibákat!).

A továbbiakban is szabadon fogunk elméleteket, kijelentéseket és más tudományos munkákat használni. Gauss, Legendre, Bernoulli és még sokan mások sem igényelnek anyagi juttatásokat azért, hogy eredményeiket és eljárásaikat mi is használhassuk. A fontos az, hogy elismerjük bárkinek a munkáját, nem pedig az, hogy önző alapon elrejtsük a tudást.

Nem tartozik ezen írás céljához egy kiterjedő elemzés alkotása a kulturális (tudományos) kereskedelmi nézetekről, de optimisták kéne legyünk. Ez az optimizmusunk a nyitott rendszerek világára és ezen belül a nyílt forrás mozgalomra (*Open Source Movement*) alapozódik. Ez utóbbi talán érdemel egy pár sort ebben a cikkben.

Minden munkánk nyílt forrású programcsomagokra alapozódik. Mindenki számára nyilvánvaló, hogy egy olyan nyílt forrású operációs rendszer mint a Linux® megelőzi a kereskedelmieket, legalábbis ami a hálózati működést illeti. Egy igen hatékony relacionális adatbázis (*RDBMS*), a PostgreSQL kínál számunkra számos hasznos jellemzőt. A földgolyón talán leghasználtabb hálózati szerver (Apache) segítségével „böngészhetünk szerkezeteket". Nem hagyhatjuk említés nélkül a Java® platformot sem, valamint az XML-hez fűződő eszközök rengetegét.

Reméljük, az előbbi sorokból is kitűnik, nyitott rendszerek segítségével stabil és megbízható alkalmazási modellekhez vezetett az eredetileg elméleti kutatók által elindított munka. Sajnos nincs tudomásunk e téren eredményes közreműködésekről más mérnöki szakterületeken.



3.1
Egyszerű modell szerkezeti megjelenítése közismert hálózati böngészővel.

Talán az XML, valamint a szakterület-specifikus sémák[2] elterjedésével a közeljövőben lehetőség nyílik legalább a strukturált tudás-bázisok kialakulására és megjelenésére.

3.2. Szerverek és kliensek

Végezetül, visszatérve az általunk javasolt rendszerhez, tulajdonképpen egy kliens-szerver megoldást alkalmazunk. A hardver felől közelítve, a rendszer működéséhez gyakorlati szinten elégséges egy számítógép. A szoftver szempontjából viszont egy olyan architektúrát igyekszünk fejleszteni, ami a következő szervereket csoportosítaná:

- hálózati (Apache),
- adatbázis (PostgreSQL),
- számítási (Linux).

A számítási szerverrel való összeköttetés egy hálózati böngésző segítségével valósul meg, elméletileg korlátlan számú[3] kliens számára. A tartószerkezetek adatai adatbázisok keretén belüli táblázatokban tárolódnak (minden egyes projekthez egy külön adatbázis van rendelve). A tervbeli adatokat szabadon fel lehet használni, a felhasználónak pedig nem kell az adatmentéssel törődnie.

Irodalom

[1.] GOBESZ F.Zs.: Contribuţii la realizarea sistemelor expert în domeniul construcţiilor. Doktori disszertáció, Kolozsvári Műszaki Egyetem, Románia, 2000.

[2.] PAPADRAKAKIS M., LAGAROS N.D.: Advances in structural optimization. Szerkezetelemzési és földrengéskutatási intézet, Nemzeti Tehnikai Egyetem, Athén, Görögország, 1997.

[3.] VOSS M.S.: Complex Adaptive Systems + Soft Computing = Emergent Design Systems. Harmadik nemzetközi IASTED konferencia, Mesterséges intelligencia és lágy programozás, Banff, Alberta, Kanada, 2000.

[4.] CELLA A., SOOSAR K.: Discrete Variables in Strutural Optimization. Optimum Structural Design c. könyvből, John Wiley & Sons kiadó, editorok: Gallagher R.H., Zinkiewicz O.C., 1973.

[5.] HERVE P.: Les Procedures Arborescentes d'Optimisation. Revue d'Ing. et de Rech. Oper., #14, 69-80. old., 1968.

[6.] ROY B.: Procedure d'Exploration par Separation et Evaluation. Revue d'Ing. et de Rech. Oper., #3, V-1, 61-90. old., 1966.

[2] Tájékozódáshoz nagyon jó kiindulópontként szolgálhat a http://www.aecxml.org/ címen összegyűjtött és rendszerezett kutatások stádiuma. A problémakör interoperabilitási kérdésekkel is összefügg, de ezt a jelen írásban inkább mellőzzük.

[3] Gyakorlatilag a felvázolt rendszer tesztelését öt kliens egyidejű csatlakozásával végezzük.