Ergonómiailag helyes Web oldalak automatikus generálása

Innováció és kommunikáció speciálkollégium III. félév

Készítette: Egri Tamás és Polgár Péter Balázs

Budapest, 2008. április 14 .

1 Tartalomjegyzék

2 Összefoglaló

Dolgozatunkban a Web oldalak automatikus generálásának lehetőségét vizsgáltuk meg, amely oldalak a szoftver-ergonómia alapelveinek is megfelelnek. Ez a probléma jellegéből adódóan nem oldható meg konvencionális eszközökkel, ezért a megalkotott rendszert mesterséges intelligencia algoritmusokkal képzeltük el. Ezek közül is a genetikus algoritmusokat választottuk, ezek számtalan előnyös tulajdonságuk mellett alkalmasak egy XHTML és CSS alapú oldal generálására, ezáltal a gyakorlati felhasználásra is megfelelők.

A megoldó genetikus algoritmust úgy hoztuk létre, hogy elsősorban a felület tervezők, fejlesztők munkáját támogassa felület tervek létrehozásával, de alkalmas legyen a használhatósági szakemberek munkáját is támogatni heurisztikák befogadásával. Az explicit ergonómiai heurisztikák beilleszthetőségével így az elvek nem fejlesztőfüggően épülnek be, az algoritmus ezeket az irányelveket automatizáltan fel tudja használni.

Tekintve a feladat nagyságát a bemutatott megoldás nem generálja le egy oldal teljes felületét, csak a felületi elemek elhelyezkedését meghatározó dobozokat. Viszont a rendszer keretrendszer jelegű felépítése folytán van lehetőség a kiterjesztésre, amelyre több példát is megvizsgáltunk.

A feladat mélyebb elemzése során arra jutottunk, hogy nemcsak a generálás, hanem a felület tervek és konkrét felületek automatikus kiértékelése is megvalósítható. Ehhez a rendszert egy további komponenssel kell kiegészíteni, egy neurális hálót használó kiértékelő modullal. Ez a felhasználói interakciók elemzésével és tanulás során elsajátított szoftver-ergonómiai szabályok segítségével vagy csak támogatói szinten vagy saját jogon képes egy konkrét design használati értékének a meghatározására.

3 Bevezetés

Az elmúlt néhány évben robbanásszerűen terjedt el a Web, mint az Internet legszélesebb körben használt alkalmazása. Lassan, mint információ forrás is vezető szerepet tölt be, miközben mostanra már a korábban csak a helyi számítógépen meglévő alkalmazások is megtalálhatóak a Weben, csaknem azonos szolgáltatásokkal. Ezért hovatovább Web alapú operációs rendszerről beszélhetünk, miközben a számítógépünkön megtalálható helyi alkalmazások közül néha már csak a böngészőt használjuk.

Miközben egyre fontosabb lenne, hogy a Weben keresztül elérhető alkalmazások és különösen a tartalmak a felhasználók számára a lehető legkönnyebben érthető, elérhető, használható módon jelenjenek meg, gyakran korántsem ez a helyzet. A technikai jellegű problémák ma már többnyire megoldódtak, a szabványosított technológiák (például: XHTML, CSS, SVG) már gyakorlatilag bármit lehetővé tesznek, egy Web oldal létrehozásakor csak a fantázia szab határt a lehetőségeknek. Azonban még mindig nagyon kevés hangsúly kerül a befogadói oldalra. A felhasználói igények felmérése inkább csak szórványosan történik meg, bár kétségtelenül sokkal jobb a helyzet, mint 10 vagy akár 5 évvel ezelőtt, de számos szoftver-ergonómiai, használhatósági probléma még ma is fent áll (Nielsen, **Hiba! A hivatkozási forrás nem található.**).

A legnagyobb probléma, hogy a programfejlesztési eszközök továbbra is elsősorban a technikai kérdésekre koncentrálnak, és nem támogatják a fejlesztők munkáját ezen a téren. Ahol alkalmazzák a szoftver-ergonómiai elveket, ott a használhatósági vizsgálatok és az ezzel foglalkozó szakemberek munkája beépült a fejlesztési folyamatba, de ez még így sem elég. Bármennyire is alapos egy Web oldal használhatósági vizsgálata, azt továbbra is emberek végzik, a több kézen is átfutó információk értelmezése is módosuláson eshet át a résztvevők esetlegesen eltérő háttértapasztalatai, hozzáállása, mentális reprezentációja miatt. Ezen kívül továbbra is kérdéses, hogy a fejlesztők által végül javasolt felület tervek között megtalálható-e az, amit a felhasználók igényeit a lehető legjobban lefedi, hiszen a felületiek használhatósági vizsgálata továbbra is csak az elkészült tervekre vonatkozhat, nem képes újszerű javaslatokat létrehozni.

Dolgozatunkban a fenti problémák kiküszöbölésére egy olyan rendszer készítésének lehetőségét vizsgáltuk meg, amelyben a Web oldalak felülete automatikusan generálódik úgy, hogy ebben a szoftver ergonómiai tudás is hasznosul. Egy ilyen rendszer előnyei a következőkben foglalhatóak össze:

- Vizsgálatok lefolytatásának időbeli és költség csökkentése.
- Új felület terv változatok gyors előállítása.
- Változó igényekhez adaptívan elkészülő új felületek.
- Tesztelés, kiértékelés és felületek előállításában csökkenteni az emberi résztvevők saját nézetrendszeréből adódó torzulásokat.
- Újszerű megoldások vizsgálata és kiértékelése

Az általunk ajánlott módszer genetikus algoritmusokkal állít elő weboldal variációkat. Ezzel elsősorban a felület terv variációk gyors előállítását segíti elő, olyan megoldásokat is megtalálva, amelyek kezdetben kívül estek a fejlesztők látókörén. Másrészről a módszer ezek kiértékelését is támogatja, illetve hosszabb távon alapját képezheti egy automatikus kiértékelést megvalósító keretrendszernek.

A genetikus algoritmusokra számtalan előnyös tulajdonsága miatt esett a választásunk:

- Képesek sok megoldási javaslat közül (nagyon nagy és töredezett keresési tér esetében is) gyorsan megtalálni a megfelelőt.

- Alkalmasak heurisztikus tudás befogadására, így a szofter-ergonómiai irányelvek könnyen és hatékonyan építhetőek bele.

- Az automatikus kiértékelés távlati céljaihoz is jól illeszkednek, könnyen integrálhatóak más rendszerekhez.

Az automatikus generáláson túl megvizsgáltunk a kiértékelési támogatására egy további módszert, a neurális hálókat. Elképzelésünk szerint ezzel a kiegészítéssel a vázolt rendszer képes lenne a heurisztikus információk és a felhasználói viselkedés elemzésével a felhasználói igényeknek a lehető legjobban megfelelő Web oldalakat előállítani.

A dolgozat további részében először bemutatjuk a megoldáshoz szükséges általános ismereteket a szoftver-ergonómiából (4. fejezet) és a felhasznált mesterséges intelligencia módszerekből (5). Ezután áttekintjük a területen már létező munkákat (6. fejezet), majd vázoljuk az általunk javasolt megoldást (7. és 8. fejezetek) és a hozzá tartozó teszt implementációt (9. fejezet). Végül az utolsó fejezetben áttekintjük, hogy milyen további lehetőségek várhatóak ezen a területen (10 fejezet.).

4 Web oldalak ergonómiája

A szoftver-ergonómia, más néven használhatóság alatt az informatikai rendszerek felületi elemeinek (felhasználói felületének) használati értékét értjük.

A használhatóság alapgondolata, hogy a számítógépet felhasználók elsősorban nem számítógépet akarnak használni, sokkal szívesebb foglalkoznak bármi mással. Például szívesebben mennek fagyizni, néznek meg egy filmet, vagy éppen játszanak a gyerekükkel. Ha legvégső esetben mégis számítógép elé kell ülniük, akkor is a feladatukat szeretnék gyorsan és hatékonyan elvégezni.

Egy jól használható felület (szoftver-ergonómiailag helyesen felépített) a felhasználókat támogatja az elvégzendő feladatban, hogy a feladat elvégzéséhez nem kapcsolódó műveleteket, az azokból adódó kognitív terhelést lehetőség szerint minimalizálja. Az így keletkezett nyereség nem csak az egyes emberek szintjén értelmezhető, hanem konkrétan vizsgálható az így keletkezett üzleti haszon is a következő területeken:

- Munkavégzési idők csökkentése:
    - Gyakran használt műveleteknél az időbeni megtakarítás számottevő pénzbeli megtakarítást is jelent (órabérrel arányos).
    - Ügyfél kiszolgálásnál a válaszadási idők csökkentése növelik az ügyfél elégedettséget.
    - Gépies feladatok gyorsabb elvégzése után több idő marad az értékteremtő tevékenységekre.
- Emberi minőség növelése:
    - Jól használható szoftver csökkenti az emberekre nehezedő nyomást és stresszt, hozzájárulva az eredményes és hatékony munkavégzéshez, az alkalmazottak elégedettségéhez.
    - Kisebb kognitív terhelés miatt kevésbé fáradt, a feladatra jobban koncentrálni tudó, így hatékonyabb munkatársak lesznek.
- Bevezetés sikeressége:
    - A szoftver sikeres működése nagyban függ a felhasználók elégedettségétől, ezeket a használhatóság pszichológiai elveinek alkalmazásával lehet növelni.
    - A felhasználók bevonása a tervezési / megvalósítási folyamatba növeli az elkötelezettséget a rendszer irányába, ez pozitív visszacsatolást képez.
    - Eszközöket adhat a szoftver projektek pozitív kommunikációjához (funkcionális eredmények helyett feladat végzés támogatásának kiemelése).
- Költségcsökkentés:

    - A jól használható felület könnyebben tanulható, ezért kevesebb képzés kell a megtanulásához.
    - A képzések során elsajátított képességek tartósabbak, így az újraképzés költségei is kisebbek.
    - Kevesebb támogatás kell a felhasználóknak a használat során.
- Manuális hibázások számának csökkentése:
    - Kevesebb hibás adat keletkezik. Ez a felhasználó szempontjából is pozitív, mert a hibázásból adódó frusztrációt is csökkenti.
    - Kevesebb hibázás miatt további időmegtakarítás érhető el, nem kell újra elvégezni egy folyamatot.
- E-kereskedelem hatékonyságának növelése
    - Jobb konverziós arány. A látogató felhasználók nagyobb arányban lesznek vásárlók, ha a vásárlás folyamata kellően egyszerű.
    - Az adott szolgáltatásról vagy termékről szóló felhasználókat érdeklő információk gyorsabban eljutnak a célcsoporthoz.
    - A felhasználók a szolgáltatás minőségét a felület alapján ítélik meg, vagyis ez jelentős befolyásolja például hogy mennyire érzik megbízhatónak az adott szolgáltatót.

A fenti felsorolásból is kitűnik, hogy sok területen érhető el üzleti haszon egy jó felület hatására. A Web esetén ez egyébként nemcsak a for-profit szektorra igaz, hiszen a non-profit szervezetek honlapjainak is kell egymással versenyezni a támagtók gyűjtésében, az információk szolgáltatásában,, míg az állami szektor honlapjai az ügyfelek minél jobb kiszolgálására kellene egyre nagyobb hangsúlyt helyezni.

A pszichológia és a műszaki ergonómia eredményeire építve a használhatóság alap és alkalmazott kutatásban is hosszabb múltra tekint vissza. A szoftverek területén a gyakorlati alkalmazása viszont az elmúlt években látszik fellendülni. Ennek legfőbb oka a számítógépek elmúlt évtizedben tapasztalható robbanásszerű elterjedése, és az ezzel járó felhasználó szám növekedés, amelyek között ma már lényegesen nagyobb számban vannak olyanok, akiknek az elsődleges szakterületük nem függ össze az informatikával. A másik ok a technikai lehetőségek növekedése, a hardverek nagyobb felkészültsége mellett (például grafikus kijelzők elterjedése) a programfejlesztés egyéb szakaszai mára már megfelelő módszertanokkal rendelkeznek, ezért jut energia a felületek jobb megtervezésére is.

A fentiek ellenére számtalan hiba és buktató van ma is még a szoftverfelületek tervezésében:

- Eltérő mentális modellek: A felhasználók és a fejlesztők a feladatokról nagyon eltérő mentális modellel rendelkeznek. Ennek egy részproblémája, hogy gyakran a felhasználó sem tudja megmondani, mire van szüksége. A fejlesztők viszont a felületben a saját gondolatvilágukat és elképzeléseiket valósítják meg. Amennyiben a tervezésben részt vesz használhatósági szakember, javíthat a helyzeten, csakhogy ekkor az ő mentális modellje is befolyásolja a fejlesztést.

- ’Karácsonyfa’-hatás: A fejlesztők minél díszesebb, szebb, az adott technikai lehetőségeket maximálisan kihasználó felületet terveznek. A használható felületnek nem kell feltétlenül kiemelkedő esztétikai élményt nyújtania vagy minél több lehetőséget biztosítania az egyes műveletek elvégzésére, sokkal fontosabb az elvégzendő feladat megfelelő támogatása.

- Szabványok nagyvonalú értelmezése: A szoftverfejlesztő eszközök viszonylag nagy szabadsági fokot biztosítanak a felületek elkészítésében még a jól szabványosított környezetekben (például a Windows ablakkezelőjében, vagy a QT-ban). Ebből kifolyólag a fejlesztők gyakran használják nem szabványos módon a felületi elemeket, megzavarva ezzel a felhasználók munkáját.

- Nem egységes felületi célok: Különösen nagy szoftver rendszerek fejlsztése közben fordulhat elő, hogy az egyes alrendszereket fejlesztő csoportok egymástól eltérően értelmezik a felületi elemeket, amelynek a következményeképpen inkonzisztencia lép fel. Emiatt a felület ellentétes utasításokat ad a felhasználónak a különböző helyeken, eltérően jelöl azonos fogalmakat, megzavarva ezzel a felhasználót a munkafolyamat végzésében.

Ábra 1: Tipikus használhatósági hiba, ellentmondó utasítás és választási lehetőségek

A szoftver-ergonómialig helyes felületek tervezéshez az alábbi általános alapelveket lehet megfogalmazni, a sorrendiség nem tükrözi az elv fontosságát (Galitz, [9]):

- Beállítható: A felületi elemek könnyű beállíthatóságával támogatható a rendszer aktív megértése és ez az irányítás érzését is kiváltja a felhasználónál.

- Belemerülés: A felület lehetőség szerint támogassa a belemerülés (flow) érzését.

- Biztosítás: A felhasználót meg kell védeni a hibázások ellen.

- Egyszerű: A felület a lehető legegyszerűbb legyen.

- Elérhető: A rendszer minden része legyen elérhető a sorrendtől és az időtől függetlenül. El kell kerülni az olyan nézetek, üzemmódok használatát, amelyben egy vagy több funkció nem elérhető.

- Észlelhető: A felhasználók szenzoros képességeitől függetlenül legyenek felismerhetőek a felületi elemek.

- Esztétikus: A felületnek az általános grafikai tervező elveket kell követnie, vagyis biztosítson az egyes elemek között megfelelő kontrasztot, létesítsen értelmes

csoportosításokat, a színek és grafikai elemek egyszerűen és hatékonyan legyenek felhasználva.

- Hatékony: Lehetőség szerint minimalizálni kell a fizikai mozgást, azaz a kéz-, szem- és egyéb mozdulatokat.

- Hiba toleráns: A felhasználói hibák legyenek azonnal javíthatóak, ne okozzanak a rendszerben visszavonhatatlan változásokat.

- Hozzáférhetőség: A felületnek a lehető legtöbb ember számára kell hozzáférhetőnek lennie módosítás nélkül (például gyengén látó, eltérő nyelvet beszélők számára).

- Irányítás: Az interakciót a felhasználónak kell irányítania. Vagyis a rendszerben ne történjen olyan akció, amelyet nem a felhasználó váltott ki explicit módon.

- Ismerős: A felhasználó számára ismerős elveket használjon a felület, vagyis a meglévő metaforákat hatékonyan kell kiaknázni (erre a legjobb példa a desktop metafora).

- Kezelhetőség: A fizikai adottságoktól függetlenül legyen kezelhető a rendszer.

- Kompatibilis: A felhasználói szempontokat kövesse a felület, azaz kompatibilisnek kell lennie a személyével, a feladatával és folyamat végeredményét adó termékkel.

- Kompromisszumok: Amennyiben egymástól eltérő követelmények vannak, a felhasználói követelményeknek elsőbbségük van a technikai követelményekkel szemben.

- Konzisztens: A rendszer egészében a hasonló dolgok működjenek hasonlóan, az azonos dolgok legyenek elnevezve ugyanúgy.

- Közvetlen: A feladatok elvégzésére közvetlen módon legyen lehetőség, és a végrehajtott módosítások is közvetlenül jelenjenek meg.

- Láthatóság: A rendszer állapota és használata mindig legyen tiszta a felhasználó számára.

- Megjósolható: A felületen végrehajtott akciók következményei legyenek előre láthatóak, az egyes akciók következményei legyenek jól lokalizálhatóak.

- Nyilvánvaló: A felület legyen könnyen tanulható, a feladatok végrehajtása során legyen mindig nyilvánvaló a felhasználó számára, hogy hova kell figyelni, mit lát maga előtt, mik a lehetőségei a továbblépésre, hogyan hajthat végre egy feladatot.

- Rugalmas: A felületnek fel kell készülnie az egyes felhasználói csoportok eltérő igényeire (például informatikai felkészültség).

- Transzparens: A felhasználónak legyen lehetősége a feladatára koncentrálni a felület működésének mélyebb ismerete nélkül.

- Világos: Mind vizuálisan, mind nyelvileg, mind koncepcionálisan áttekinthető felületet kell készíteni.

- Visszafordíthatóság: A rendszerben legyen lehetőség az akciók visszafordítására, a felhasználó elvégzett munkája semmilyen körülmény között ne vesszen el.

- Visszajelzés: A felületen mindig jelenjen meg visszajelzés az elvégzett akció következményeiről.

A fenti elveknek megfelelést számtalan módszer támogatja. Ezek egy részről a felhasználói felület tervezésének alapelvei, másrészről a vizsgálati módszerek.

A felhasználói felületek tervezése és fejlesztése a szoftverek egyéb részeihez hasonló elemekhez hasonlóan épül fel, speciálisan a felhasználói igények megismerésével is. Nagy vonalakban az alábbi lépések különíthetőek el:

- Felhasználó megismerése

- Üzleti folyamat megismerése

- Navigáció kialakítása

- Képernyő felépítések meghatározása

- Interakciós eszközök kiválasztása

- Komponensek használata

- Szöveges üzenetek megfogalmazása

- Visszacsatolás és segítségnyújtás megalkotása

- Nemzetközi és fogyatékkal élőknek szóló verziók kialakítása

- Grafikai megvalósítás

- Ablakok, oldalak elrendezése

- Tesztelés

Az utolsó lépés, a tesztelés takarja a felületek tervezése során felhasználható vizsgálati módszereket. Ezen a módszerek segítenek a felhasználói igények megismerésében, számszerűsítésében, illetve több felület terv közül a felhasználók számára jobb kiválasztásában. A teljesség igénye nélkül néhány ilyen módszert felsorolunk itt.

- Heurisztikus elemzés: A felület szakértői elemzése a létező irányelvek alapján.

- Interjú: Felhasználói igények felmérése személyes megbeszélés során.

- Megfigyelés: Felhasználói interakció megfigyelése a rendszer konkrét használata során.

- Kérdőív: Statisztikai információk gyűjtése széles felhasználói körből.

- Kognitív bejárás: Egy feladat végrehajtásához szükséges út bejárása és hibák feltárása a felületalkotó elemek ismeretében.

- GOMS analízis: Célok (Goals), operátorok (Operators), módszerek (Methods), kiválasztási szabályok (Selection rules) betűszó alapján. A felhasználói interakció atomi elemekre bontásával talál javítási lehetőségeket.

- Fiziológiai módszerek: Például szemmozgás követés, a használat során a felhasználó élettani és más értékeinek mérése majd elemzése alapján lehet következtetéseket levonni a felület használat hatékonyságáról.

Végül fontos megjegyezni, hogy a használhatóság nem alkalmas egy adott szoftver funkcionális teljességének leírására vagy ellenőrzésére, csak a felület emberi tényezőivel foglalkozik.

5 Intelligens rendszerek

Dolgozatunkban egy olyan rendszert vázolunk fel, amely a **Hiba! A hivatkozási forrás nem található.**. fejezetben felírt célokat egy intelligens rendszerrel, elsősorban genetikus algoritmusok felhasználásával oldja meg. A megoldás kontexusba helyezéséhez áttekintjük, hogy pontosan mik azok az intelligens rendszerek és milyen módszereket használnak fel a létrehozásukhoz.

A választott probléma megoldását genetikus algoritmusokkal képzeljük el, illetve megmutatjuk, hogy további kiegészítés tehető egy neurális hálóval, ezért ezt a két módszert a későbbiekben nagyobb részletességgel is bemutatjuk.

5.1 Bevezetés

Az elmúlt ötven év technikai fejlődése nyomán jelentősen bővültek a lehetőségeink a hétköznapi feladatok elvégzésére. A számítógépek segítségével a numerikus műveleteket gyorsabban és pontosabban tudjuk végrehajtani, mintha saját erőből tennének ugyanezt. Azonban ez a sebességnyereség csak az egyszerű és mechanikusan végrehajtható feladatok esetében jelentkezik, azaz ahol a végrehajtott műveletek önmagukban egyszerűek, vagyis a számítógép ezekben az esetekben nem több mint egy nagyon gyors számológép.

A mesterséges intelligencia kutatásban elért eredmények mára viszont már azt is lehetővé teszik, hogy az egyszerű matematikai, feldolgozó műveleteken túlmutató feladatokat is számítógép segítségével oldjunk meg. A mesterséges intelligenciát is tartalmazó szoftverrendszereket összefoglaló néven intelligens rendszereknek nevezzük (Hopgood, [1]).

Azok a területek, ahol szükség lehet intelligens rendszerekre, valamilyen szempontból túlmutatnak az egyszerű számításokon, többnyire jelentős emberi beavatkozást igényelnek vagy nehezen formalizálható, matematikailag rosszul felírható feladatokból állnak. A mechanikusan ismétlődő feladatok automatizálásával (mind a szoftveresen megoldható problémák, például gazdasági számítások, mind a hardveresen megoldható problémák, például gépek összeszerelése) egyre nagyobb figyelmet kapnak ezek a területek.

Az intelligens rendszerekben alkalmazott módszereket általánosan három nagy területre oszthatjuk fel:

- Tudás alapú rendszerek
- Lágy számítási módszerek
- Hibrid rendszerek

5.2 Tudás alapú rendszerek

A hagyományos szoftverekhez képest a tudás alapú rendszerek eltérő megközelítéssel rendelkeznek. A klasszikus értelemben vett szoftverrendszerekben a program maga tartalmazza a tartomány függő tudást, vagyis a feladat megoldásához szükséges információkat közvetlenül kódolva tartalmazza. A tudás alapú rendszerekben elkülönülnek a vezérlő

utasítások, és a feladat megoldásához szükséges információk tároló egységei. Ez két, egymástól explicit módón elváló komponens a tudásbázis, és a következtető motor. Ez a különválasztás ahhoz is segítséget nyújt, hogy a tartományban keletkező új tudást a vezérlő elemek megváltoztatása nélkül be lehessen építeni a rendszerbe, amely jelentősen megkönnyíti azok felhasználását.

A tudásbázis egyszerűsítve szabályokat és tényeket tartalmaz. Mindkét elem lehet egyszerű (például egy mondat) vagy összetett (egy eljárás hosszú leírása). A tények olyan állításoknak felelnek meg, mint például „Az autóm kék.”, míg a szabályokba tetszőleges számú további elem is behelyettesíthető, például „Minden $SZÍN autóval rendelkező ember behajthat a $HELY-be”, ahol a $ jellel kezdődő szavak helyére lehet más tényeket behelyettesíteni.

Amennyiben a tartomány függő tudásfelhasználása további speciális ismereteket igényel, vagyis az információk önmagukban nem használhatóak fel, akkor speciális információkkal, metaheurisztikákkal írják le a tudás felhasználásának módját.

A következtető motor feladata a tudásbázis elemeinek felhasználása, azaz a tények és a szabályok segítségével kell a megadott kérdésre (feladatra) megoldást találni. Két nagy csoportot különítünk el a következtetés irány szerint: célirányos, vagy adatirányos. A célirányos következtető a megadott célra próbálja a tudásbázisban lévő információkat illeszteni, míg az adatirányos a meglévő adatokból építkezik. Mindkettőnek megvan az előnyes és a hátránya is, általában a probléma ismeretében kell döntést hozni a megfelelő forma használatáról.

A tudásalapú rendszerek speciális fajtái a szakértő rendszerek, amelyek egy speciális tartományban meglévő információkat használják fel (például orvosi rendszerek). A tudás alapú rendszereket önállóan többnyire ebben a formában használják fel (például MYCIN [2]).

A tudásalapú rendszerek sikeres felhasználási területei az orvosi tanácsadó rendszerek, ügynöki termék konfigurációs rendszerek, automatikus hitelminősítő rendszerek.

Egy jó leírása található a tudás alapú technológiákról és a szakértő rendszerekről (Sántáné, [10])-ban.

5.3 Lágy számítási módszerek

A tudás alapú rendszereket kifejezetten emberi közreműködésre alkalmas módon hozták létre, ezáltal az adott terület szakértői a fejükben lévő információkat könnyen átadhatják. A számítási intelligenciát felhasználó rendszerekben ugyanez a reprezentáció kódolva áll elő, ezáltal a számítógép sebességét jobban ki tudják használni.

Az elnevezés onnan ered, hogy ezek a módszerek egyrészről olyan területeken is eredményesen használhatóak, ahol nincsenek jól meghatározott információk, vagy a bemeneti adatok bizonytalanok. Másrészről a keletkezett eredmények a klasszikus módszerekkel ellentétben gyakran nem pontosak, csak egy bizonyos hibahatáron belül használhatóak, ami viszont egy valós alkalmazás esetén már többnyire elegendő szokott lenni. Viszont az előállított megoldások nagyon számításigényes problémák (például NP nehéz problémák) estén is viszonylag gyorsan megjelennek, ami a gyakorlati felhasználás szempontjából szintén nagyon fontos.

A lágy számítási módszerek közé a következő főbb módszerek tartoznak:

- Genetikus algoritmusok: A természetes kiválasztódás és genetikai koncepciójának alkalmazásával előálló optimalizációs módszerek.

- Neurális háló: Az agy működési elvén alapuló statisztikai modellező eszköz.

- Fuzzy rendszerek: Többértékű logikát alkalmazó eszközök, amelyek az emberi világ fogalmainak leírására fokozottan alkalmasok.

- Raj intelligencia: Központi irányítás nélküli, viselkedés alapú, önszerveződő rendszerek, elsősorban vezérélési problémák megoldására.

A dolgozatunkban a felhasznált genetikus algoritmusokat és neurális hálókat itt részletesen is ismertetjük.

5.3.1 Genetikus algoritmusok

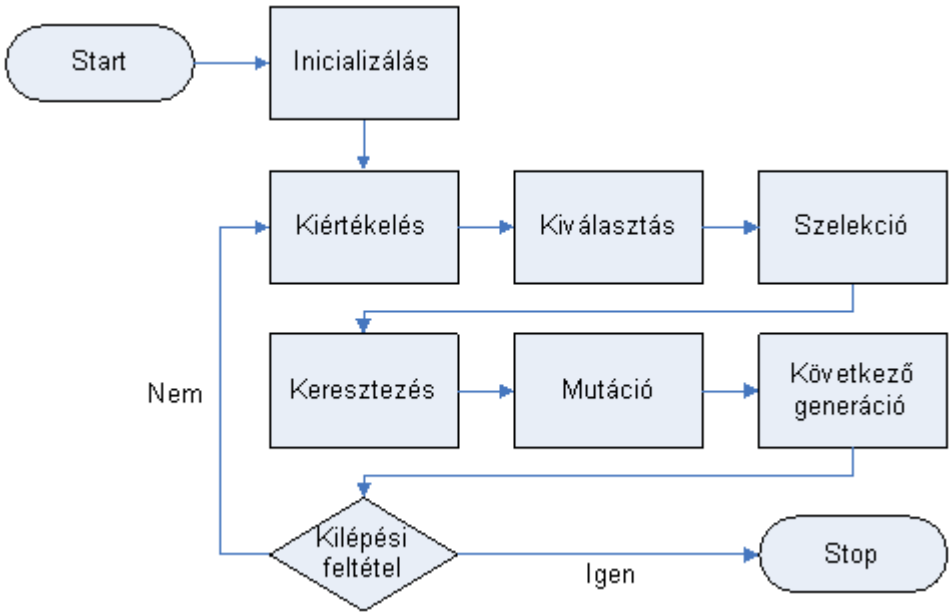
Az első genetikus algoritmust John Holland alkotta meg 1975-ben (Holland, [11]), azóta nevezzük összefoglaló néven így ezt a keresési és optimalizálási feladatok megoldására javasolt lágy számítási módszert.

A genetikus algoritmusok heurisztikus módszerekkel oldanak meg optimalizálási problémákat. Az alapötlet Darwin evolúciós elméletéből jön, amely elsősorban a természetes kiválasztódásra épít. Érthető az informatikával foglalkozók érdeklődése az evolúció iránt, hiszen a biológiában rendkívül bonyolult szervezetek jöttek létre, amelyek az életre jellemző kihívásokra könnyedén választ adnak, és túlélik a megpróbáltatásokat. A biológusok már felismerték az ide vezető utat, az evolúciót természetes kiválasztódással. A számítógépekkel foglalkozó tudósok másrészről gyakran találkoznak olyan problémákkal melyek optimális megoldására nincs kellően gyors (polinom időben lefutó) algoritmus, ezért szükség van olyan módszerekre, amelyek ésszerű időn belül, legalább megközelítő megoldásokat tudnak adni. Ebből kifolyólag jó megoldás lehet az evolúció sikeres mechanizmusát alkalmazni számítástechnikai problémákra.

5.3.1.1 A kanonikus genetikus algoritmus

Az alábbiakban bemutatjuk a genetikus algoritmusok részletes működését. Az egyes részek részletes leírása ezután következik. Az itt bemutatott alapelemek a Holland által először leírt, kanonikus genetikus algoritmus felépítését követik, a később leírt formák ennek módosításai.

Az inicializálás előállítja a megfelelő méretű populációt. A populációban lévő egyedek az eddigiek szerint egy probléma megoldásai, amelyek értékek sorozatából állnak. Ezután következők a fő ciklus rész, melynek során a populáció továbblép a következő generációba. Az egyedekhez tartozó célfüggvény értékek meghatározására az evolúciós műveletek előtt van szükség, hiszen azokra szükség van ott. Ezután választjuk ki azokat az egyedeket, melyek a keresztezésben szülőként részt fognak venni. A keresztezéshez a szülő egyedeket véletlenszerűen párosítjuk, akik vagy lemásolják magukat, vagy a keresztezés operátorok segítségével állítják a gyerekek génkombinációit. Az így előálló utódok mutálódhatnak, mielőtt eláll a reprodukció eredményeképpen kapott új populáció. Ezután vizsgáljuk a kilépési feltételt, ami lehet például bizonyos rátermettségű egyed létezése a populációban, konvergencia egy értékhez, vagy bizonyos generációszám elérése. A következőkben részletesen megvizsgáljuk az egyes lépéseket.



Ábra 2: Genetikus algoritmusok sematikus felépítse

5.3.1.2 Populáció mérete

A populáció mérete határozza meg azt a keresési teret, amelyet a genetikus algoritmus a keresés során bejárhat. A keresési tér minden pontja egy megoldás, ami egy egyednek felel meg. Ha nagyobb a keresési tér, akkor a keresés finomabb lehet, hiszen több a rendelkezésre álló és felhasználható genetikus anyag, génváltozat Ekkor a helyes megoldásról alkotott kép is részletesebb lesz. Másrészről a nagy populáció több számítást, így nagyobb erőforrásokat igényel. Így egyensúlyt kell teremteni a kívánt megoldás finomsága, és a felhasználni kívánt számítási kapacitás között.

5.3.1.3 Kódolás

Genetikus algoritmusok készítése közben a legelső döntés az, hogy milyen kódolást kell majd alkalmazni, azaz a valós problémát hogyan reprezentáljuk az egyedek génjeiben. Az egyedek reprezentálta megoldásokat kromoszómákkal írjuk le, azaz gének sorozatával, melyek egyenként az allélek véges halmazából vehetnek fel értékeket. A kódoláshoz használt allélek halmazát a megoldás ábécéjének hívjuk, mely egyúttal meghatározza a lehetséges allélek számát is. A kódolás lehet pusztán a problémát leíró változók egymás után konkatenálása, vagy ezek binárisan kódolt alakja.

A kódolás jó megválasztása nagyban befolyásolhatja az algoritmus hatékonyságát. Szintén összefügg az alkalmazott operátorok megvalósításával is, főként a keresztezés operátoréval. Fontos azt is felismerni, hogy az összetartozó információkat hordozó géneket jobb a kódolás során egymás közelében elhelyezni, így ezek a keresztezés során is nagyobb valószínűséggel maradnak együtt (erről bővebben lásd a keresztezést bemutató pontnál).

Az építő elem hipotézis (Goldberg [12]) megállapítása szerint a genetikus algoritmus akkor talál jó megoldásokat, ha építő elemeknek nevezett kisebb részeket kombinál. Egy ilyen elem csupán néhány, majdnem optimális génből áll, melyek együtt az egyed rátermettség értékének nagy részét adják. A keresés sikeréhez létfontosságú, hogy ezeket a részeket a gyerekek is örököljék, hiszen ettől lesznek fittek. Hogyha a keresztezés operátor a szülőben lévő elemet elosztja a gyerekek között, akkor ez a rátermettség érték elveszik. Ebből kifolyólag a kódolást

és a keresztezés operátorokat együtt kell megtervezni, hogy a jó építőelemek öröklődése biztosítva legyen.

5.3.1.4 Inicializálás

Az inicializálás biztosítja a kezdeti populáció létrehozását, melyet az algoritmus futása során evolválunk. A jó kezdeti populáció kellő változatosságot biztosít, hogy a genetikus algoritmusnak minél több feldolgozható nyersanyagot biztosítson. A populációt gyakran generálják a gének véletlenszerű feltöltésével a lehetséges allélek közül.

Egy jó hasonlat, ha a genetikus algoritmust úgy képzeljük el, hogy a keresési tér egy tájkép. A rátermettség térkép a keresési tér alapján készül, egy további dimenzió hozzá vételével, amely minden ponthoz egy magasságot rendel, vagyis a hozzá tartozó rátermettség értéket. Ha a kódolás két gént használ, akkor a rátermettség térképet úgy is elképzelhetjük, mint egy táj a három dimenziós térben, melyben hegyek , síkságok és völgyek találhatóak. Ha a megoldandó probléma maximalizálás, akkor a keresett lokális optimum pont az a pont, mely a környező pontoknál mind magasabb, azaz jelen esetben egy hegy csúcs, a globális optimum pedig a tájkép minden pontjánál magasabb, azaz a legmagasabb hegy (minimalizálási probléma esetén völgyeket keresük). A genetikus algoritmus ezt a pontot próbálja megtalálni.

Leggyakrabban nem rendelkezünk előzetes információkkal a legmagasabb csúcsról, ezért az a legjobb módszer, ha a kezdeti populáció egyenletesen beteríti a teljes térképet, hogy minden részről elegendő információval rendelkezzen. Ezek segítségével a genetikus algoritmus meghatározza, hogy melyik a legmagasabban elhelyezkedő régió, melyben várhatóan az optimum pont is megtalálható lesz. A keresztezés és a mutáció új pontokat hoz létre, melyek tovább irányítják a keresést.

Az inicializálás az építő elemeket is biztosíthat, melyek az algoritmus futása során nagyobb valószínűséggel állhatnak elő, ha kezdetben elég nagy a változatosság.

5.3.1.5 Rátermettség (fitnesz) függvény

A genetikus algoritmusok egy megoldás jóságának az eldöntésére egy rátermettség függvénynek nevezett költségfüggvényt használnak. Ezt a függvényt nem mindig a probléma határozza meg, olykor a rossz megoldásokat büntető szempontok is felmerülhetnek, amelyek vagy részben vagy teljesen alkotják a rátermettség függvényt. A legtöbb genetikus algoritmus futása során a számítások döntő részét a rátermettség függvény foglalja le, így ezt úgy kell megalkotni, hogy a számítás hatékony maradjon.

A rátermettség függvényt nem feltétlenül kell kiszámítható matematikai alakban, számítási módszerként megadni. Manapság több helyzetben is alkalmazták genetikus algoritmusokat humán kiértékeléssel (Interaktív Genetikus Algoritmusok, IGA), például képek vagy zenedarabok előállítására.

5.3.1.6 Kiválasztás

A kiválasztás segítségével biztosítja a genetikus algoritmus a jó megoldási sémák továbbvitelét a következő generációba, ugyan úgy, mint a természetben, ahol azok az egyedek szaporodnak jobb eséllyel, akik jobban felkészültek a korlátozott erőforrásokért folytatott küzdelemre. A kiválasztást leginkább az jellemzi, hogy mennyire hangsúlyozza a jó megoldásokat, ezt a biológiához hasonlóan szelekciós nyomásnak nevezzük. A szelekciós

nyomás mértéke lehet egészen minimális (véletlenszerűen választott egyedek), vagy akár maximális (csak a legjobb egyedek kerülnek kiválasztásra) is, e két véglet közt kell megtalálnunk a helyes utat. A kiválasztást számszerűen a szelekciós intenzitás adja meg, melyet a teljes populáció és az egyed rátermettség értéke ad ki.

A kanonikus algoritmus rátermettség arányos (más néven rulett kerék) kiválasztást alkalmaz, melyben minden egyednek a rátermettségével arányos esélye van a kiválasztásra. Ennek a módszernek a hátránya, hogy túl nagy szerepet kap benne az egyedek rátermettség értéke, így a futás vége felé, amikor a fitneszek egyre kisebb mértékben térnek el egymástól a szelekciós nyomás jelentős mértékben csökkeni fog.

Másik elterjedt módszer a versengő (tournament based) kiválasztás, mely a populáció véletlenszerűen kiválasztott egyedeit versenyezteti, ahol a legmagasabb rátermettséggel rendelkező egyed nyer és kerül kiválasztásra. Ez a módszer folyamatosan fent tartja a szelekciós nyomást, hiszen nem az abszolút rátermettség értékek közti különbségekre épít, hanem a rátermettség értékek helyezésére, így kerüli el az imént említett problémát. Ezért nevezzük az ehhez hasonlóan állandó szelekciós nyomást biztosító eljárásokat helyezéses módszereknek. A szelekciós nyomás erősségét a versengő egyedek számának növelésével vagy csökkentésével lehet állítani. Egy lehetséges probléma (amely a rátermettség arányos módszernél is előfordulhat), egy jó egyed génjei elveszhetnek, hogyha gyengébb gyermeki lesznek, melyek a következő generációban már nem kerülnek kiválasztásra. Ezt az elitizmus használatával előzetjük meg, amely során a régi populáció legjobb k egyedét átmásoljuk az új generációba.

5.3.1.7 Keresztezés

A kiválasztás a populációban már jelen lévő információkat használja fel (exploit), ezért a genetikus algoritmusnak szüksége van olyan mechanizmusra is, mely további információkat tár fel (exploration), melyet később ugyancsak fel lehet használni. A feltárás a keresztezés és a mutáció operátorok segítségével történik. A keresztezéstől (más néven rekombináció) azt várjuk, hogy a szülőkben lévő hasznos információ darabokat a gyerekekben egy nagyobb egésszé kombinálja. A biológiában a keresztezés a kromoszómák részeinek cseréjével megy végbe. Ez adta az ötletet az egypontos keresztezéshez, mely a kromoszómában véletlenszerűen kiválaszt egy pontot, majd a szülők génjeit megcseréli a pont előtti szakaszon. A kétpontos keresztezés ehhez hasonlóan működik, csak két pontot választ ki. Ezek általánosítása az uniform keresztezés, amely minden génnél véletlenszerűen dönti el, hogy megcseréli azokat vagy nem.

Azt, hogy melyik keresztezés a legjobb, mindig az adott probléma dönti el. Mint a kódolásnál már megmutattuk, gének alkothatnak építő elemek, melyek optimális értékeit együtt jó mozgatni. Inicializálás után ezek az építő elemek a populációban szétszórtan találhatóak, mely építőelemeket valahogy egy egyedbe kell összegyűjteni, hogy az optimális közeli eredmény előálljon.

5.3.1.8 Mutáció

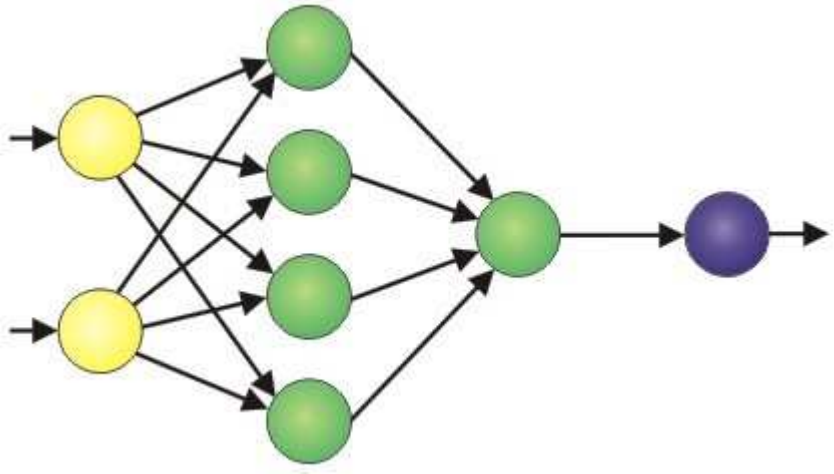
A keresési tér felfedezésének másik útja a mutáció. Minden gént egy véletlenszerű értékkel mutálunk, azaz az allélek közül véletlenszerűen választ egyet, amivel helyettesíti a gén aktuális értékét. A mutáció célja a változatosság biztosítása, és a korábban elveszett allélek visszahozása a populációba. A változatossággal a keresztezés új építő elemeket tud létrehozni, vagy szerencsés esetben a mutáció önmagában is létrehozhat építő elemeket.

Általában kompromisszumot kell kötni a szerencsés és a kevésbé szerencsés (amikor a mutáció egy építő elemet megszakít) eset között. A mutációs rátára egy jó érték az 1/k, ahol k a kromoszóma hossza.

5.3.2 Neurális hálók

Informatikai környezetben a neurális hálók alatt többnyire a mesterséges neurális hálókat értjük, szemben az agyban lévő természetes neurális hálókkal, ezért a dolgozatunkban is így használjuk ezt a fogalmat. Ezek sok kicsi számítási elemből állnak, amelyek hálóba rendeződnek, azaz valamilyen térbeli rendezettségük van. A számítási elemeket neuronoknak nevezzük, amelyek között valamilyen súlyozással rendelkező kapcsolatok vannak. Bár a mesterséges neurális hálók a természetben előforduló idegi rendszerekhez hasonlítanak, és sok elgondolás született az emberi és az állati idegrendszerek megfigyeléséből, mégis jelentős különbségek is vannak. A legfontosabb különbség, hogy a gyakorlatban használt neurális hálók valamilyen konkrét és jól meghatározott probléma megoldására kialakított, kis kiterjedésű rendszer, ezért az általa produkált viselkedés nem közelíti meg a természetben tapasztalt bonyolultságot.

A neurális hálók atomi eleme a neuron, amely a teljes hálózat számítási kapacitásának legkisebb alkotó része. Minden neuronnak több ki és bemente, ezek mindegyikéhez tartozik súly is, amely meghatározza az adott kapcsolat fontosságát. Az egyes neuronok egy előre meghatározott műveletet hajtanak végre a bemeneteken, amelynek az eredményét minden kimenet felé tovább adják, amelyet további neuronok foghatnak fel. Az egyes neuronok a saját műveletüket függetlenül hajtják végre, ebből kifolyólag a neurális hálók párhuzamos struktúrával rendelkeznek.



Ábra 3: Egy egyzerű neurális háló bemeneti (sárga), köztes (zöld) és kimeneti (kék) rétegekkel

A hálót alkotó neuronok mindegyike egy vagy több másik neuronhoz kapcsolódik mind a kimeneti, mind a bemeneti oldalon. Az egyes neuronok rétegekbe rendeződnek, amelyek a probléma egy-egy részét oldják meg, vagyis az egy rétegben található neuronok hasonló adatfeldolgozási feladatokkal és képességekkel rendelkeznek. Vannak speciális neuronok, amelyek bemeneti oldalon nem kapcsolódnak más neuronokhoz, hanem bemeneti információkhoz, ezek alkotják a bemeneti réteget. Ennek megfelelően épül fel a kimeneti

réteg is. A kettő között található többi réteget rejtett rétegeknek is szokták hívni, mert ezek a külvilág felöl nem láthatóak.

A neurális háló számítási eredményeit a meghatározott súlyok, és a háló topológiája határozza meg. A súlyokat nem kell előre megadnunk, egy tanulási eljárás során is be lehet őket állítani, erre háromféle eljárást használnak:

- Felügyelt tanulás: A tanulás során adottak ki és bementi párok. A kiértékelés során a kimeneteken az elvárt kimenettől való távolságot mérik.

- Felügyelet nélküli tanulás: Csak bemeneti adatsorokat adunk meg, és a háló az ezekben található összefüggések alapján épül fel.

- Megerősítéses tanulás: Általában előre meghatározott bementi adatsor sincs megadva, hanem a környezetből érkező értékekhez rendelődik költség.

A neurális hálókat széles körben alkalmazzák, például: függvény approximáció, osztályozási problémák, jel feldolgozás, folyamatirányítás, adatbányászat, mintafelismerés.

5.4 Hibrid rendszerek

A tudás alapú rendszerek és egy vagy több lágy számítási módszer egyidejű felhasználásával állnak elő a hibrid rendszerek. Ezek tervezése során az egyes módszerek gyengéit más módszerek erősségei egészítik ki. A kombinálást a következő területeken jelent előnyt:

- Multiterületes probléma: A valós életben előforduló problémák gyakran több területre épülnek, az egyes területek kezelésére más-más módszer alkalmas

- Teljesítményjavítás: Részfeladatok optimlaziálására gyakatan alkalmasabb egy másik módszer.

- Paraméter állítás: A lágy számítási módszerek sok szabadsági fokkal rendelkezenk, ezért a paraméterek beállítása tervezést és gyakran kisérletezést igényel, amelyet ki lehet váltani, ha egy másik módszert használunk fel a paraméterek meghatározására.

6 Előzmények

A használhatósági problémák megoldásának automatizálásában gyakran merül föl a genetikus algoritmusok használata (Hun és Yang [3]; Asslani és Lari [4]; Ivory és Hearts [5]). Ennek oka a módszer változatos feltételek mellett nyújtott magas teljesítménye szakterülettől függetlenül.

Genetikus algoritmusok felhasználást Web oldalak felépítésében veti fel (Syett [8]). Bár a javaslata egy speciális problémát old meg, az oldalakon található hirdetések számának és elhelyezésének optimális meghatározását. A javasolt módszer összességében használható koncepciónak tűnik, azonban nem mutat részletes megoldási javaslatot, különösen a visszacsatolás meggondolása hiányzik, amely az egyes generált oldalak kiértékelését hajtaná végre.

Lényegesen használhatóbb eredmények találhatók (Quiroz [6],[7]) munkáiban. Quiroz nem webes felületekkel, hanem vastagklienses felületekkel, konkrétabban XUL felületekkel dolgozott (XUL: XML User Interface Language, XML alapú felület leíró nyelv). Az általa leírt megoldásban genetikus algoritmus, pontosabb interaktív genetikus algoritmus segítségével hoz létre a felhasználói felületből változatokat. A vázolt megoldás leglényegesebb része a kiértékelés, amelyet humán közreműködéssel képzel el. Részletesen megvizsgálja az ezzel kapcsolatos legfőbb problémát, vagyis az ezzel járó emberi kifáradást a tervezői oldalon, amely összességében az algoritmus hatékonyságát rontja le. Ezzel kapcsolatban arra mutatott rá, hogy nem fontos a populáció minden tagját kiértékelni, azaz meghatározhatóak metrikák, amelyek szerinte a populációnak csak egy részét kell kiértékelni, a többi egyed rátermettség meg lehet határozni interpoláció segítségével. Ugyancsak nem fontos a minden generációt így kiértékelni, elég csak meghatározott generációszám után a kiértékelést végrehajtani.

A fentiek alapján megállapítható a következő megállapításokat tehetjük:

- Lehetséges olyan genetikus algoritmust alkotni, amellyel webes felületek generálhatóak, mégpedig a felhasználói igényeknek megfelelően.

- A tervezői oldalt is figyelembe kell venni, azaz a humán kiértékelést úgy kell megtervezni, hogy a gyakorlatban is működő képes legyen a módszer keresési hatékonyságának megőrzése mellett.

- A humán kiértékelés kevésbé hatékony, ezért törekedni kell egy olyan módszer létrehozására, amely automatikus kiértékelést is végre tud hajtani.

7 A tervezett rendszer

Egy olyan rendszert szeretnénk készíteni, amelyben genetikus algoritmusok illetve további mesterséges intelligencia algoritmusok segítségével weboldalak készítését támogatja a felhasználói igények figyelembe vételével.

7.1 Megoldási dimenziók

A megoldás elkészítéséhez tekintsük át, hogy a megoldást milyen dimenziókban hozhatjuk létre, azaz a Web oldalak tekintetében technikai szinten is hol értelmezhető a generálás fogalma, illetve ez hogyan illeszkedik be egy fejlesztési modellbe.

A weboldalak felépítésénél mára már elválasztásra került a tartalom és a forma, azaz általában az oldalon található szöveges információktól függetlenül módosíthatók az elrendezés és az egyéb felületi elemek kinézete. Ennek kereteit a CSS szabvány ([14]) teszi lehetővé. A szabványt áttekintve megállapíthatjuk, hogy elkülönülten lehet kezelni az oldalon lévő tartalmi elemek elhelyezését és az elemek kinézetét meghatározó paramétereket.

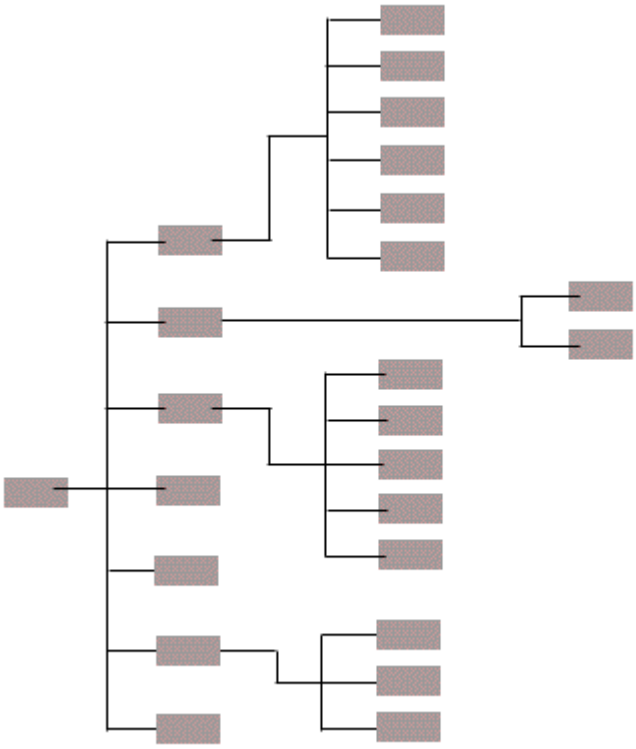
A tartalmi elemek elhelyezésére a szabvány több lehetőséget ad, de azt mondhatjuk, hogy mezőkbe, dobozokba szervezhetően lehet őket elhelyezni egy Web oldalon. Lehetőség van a dobozok többféle paraméterének meghatározására, bár minket elsősorban a pozíció megadása és a doboz mérete érdekel.

Az elemek kinézetét gyakorlatilag minden ponton meg lehet határozni (például szöveg esetén a betűtípust, a méretet, a színt, az igazítást, a kereteket, az aláhúzást, a felhasználói akciótól függő effektusokat), a paramétereket közvetlenül meg lehet adni.

```
h3 {
     font-size: 11px;
     color: #4A444D;
}
a:visited {
     color: #993333;
}
#container {
     width:700px;
     margin-left: auto;
     margin-right: auto;
}
#banner {
     height: 230px;
     padding: 5px;
     margin-bottom:1px; ;
     background-image: url(images/head.jpg);
     background-repeat: no-repeat;
}
```

Ábra 4: CSS kód többféle elemhez tartozó paraméterekkel

Van egy további lehetőség is a Web oldalak generálására, amely viszont már túlmutat egy oldal leírásán. Nemcsak az egyes oldalak felületét lehet automatikusan generálni, hanem akár az egész website felépítését.

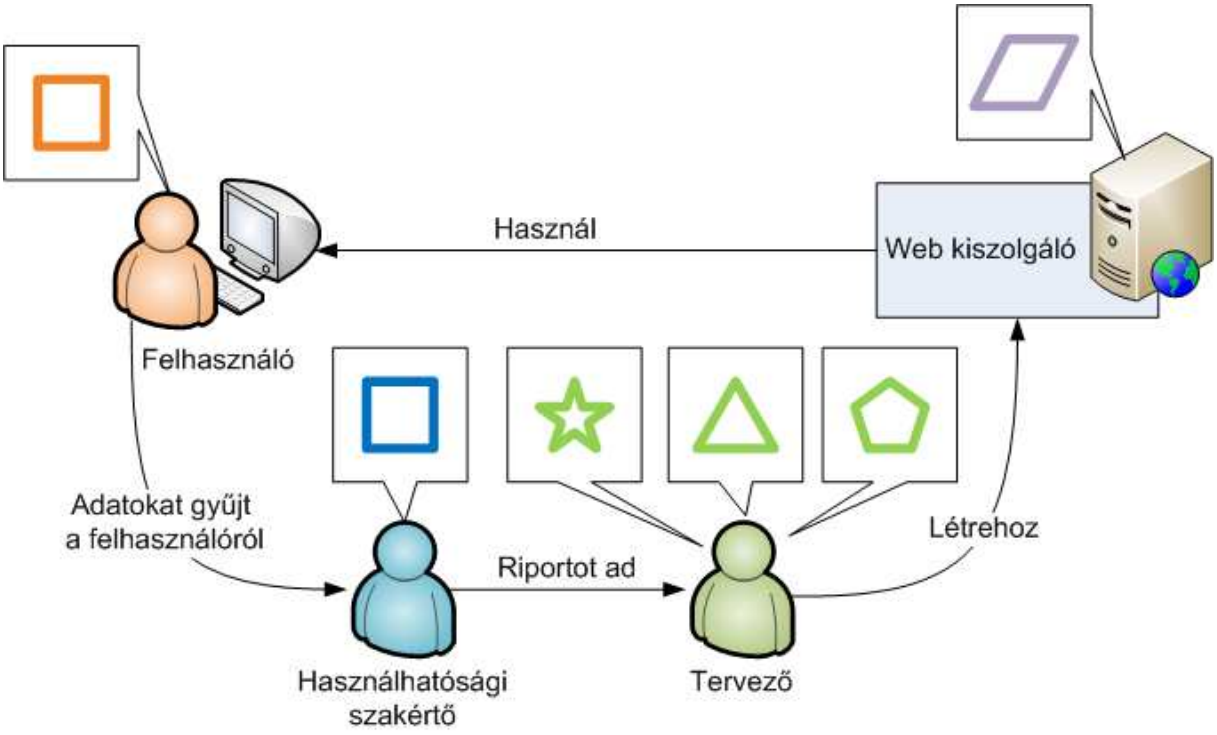


Ábra 5: Egy egyszerű Web oldalhoz tartozó oldaltérkép

A website-ok többsége az elsődleges navigáció szerint hierarchikusan fába rendezhető. Ezt a fát többféle képpen is elő lehet állítani, attól függően, hogy egy-egy elemet hova csoportosítunk. A struktúrát általában az egyes oldalakon szereplő navigáció testesíti meg, ezekben hiperlinkek határozzák meg, hogy az egyes oldalakról mely további oldalak érhetőek el. Ezek a linkek az oldalak generálásakor meghatározhatók és leírhatók.

Felmerülhet továbbá, hogy az oldalakon megjelenő szöveges információk is jelentősen befolyásolják az oldal használhatóságát. Sajnos manapság még nem áll rendelkezésre olyan szofisztikált módszer, amely képes lenne például a szöveges tartalmak dinamikus újrafogalmazására a visszajelzésektől függően, miközben a teljes információ tartalom megmarad. Egy alternatíva, ha a tartalmi elemeket a szerkesztők több verzióban is biztosítják, azonban ennek a lehetőségnek a vizsgálata túlmutat jelen dolgozat keretein.

Fontos megvizsgálnunk azt is, hogy az oldalak generálása hogyan kapcsolódik az oldal tervezés folyamatába. A Ábra 6: Web felület tervezés jelenlegi modellje mutatja be, hogy hogyan néz ki jelenleg egy weblap megtervezése, mint folyamat. Az egyes szereplők fejénél lévő buborékok mutatják be a meglévő mentális modelleket. Míg a használhatósági szakértő által felmért állapot közelít a felhasználó eredeti mentális modelljétől, a tervező sokféle ötletet rak bele a saját gondolataiból, így végül csak nagy vonalakban hasonlít a végeredmény arra, amit a felhasználó eredetileg szeretett volna.



Ábra 6: Web felület tervezés jelenlegi modellje

A genetikus algoritmusokkal generált oldalak elsősorban a tervezők és a fejlesztők munkáját hivatottak megkönnyíteni. Ennél eggyel magasabb szint, ha a közvetlenül a felhasználói interakciókról van visszacsatolás a genetikus algoritmus futásába, de továbbra is szükség van tervezők közreműködésére, végül legmagasabb szinten a kiértékelés teljesen automatizált, a tervezők csak felügyeleti tevékenységet látnak el. Ezen a szinten a használhatósági szakértők munkájára is csak felügyeleti szinten van szükség.

A következő táblázatban foglaltuk össze a lehetőségeket.

Szintek \ Dimenziók	**Elhelyezés**	**Kinézet**	**Felépítés**
Tervezés támogatása	1 fázis	2. fázis	3 fázis
Felhasználói visszacsatolás feldolgozása	2 fázis.	3 fázis	4 fázis
Automata kiértékelés	3 fázis	4. fázis	4. fázis

Ábra 7: Automatikus lap generálás lehetőségei

A feladat nagyságát tekintve a rendszer tervezését 4 szakaszra bontottuk. Jelen dolgozatban az 1. szakasszal foglalkozunk, a célunk elsősorban az, hogy a módszer helyességéről meggyőződjünk, illetve előkészítsük a további fázisokat. Ez azt is jelenti, hogy a kérdéskör

általános vizsgálata mellett elsősorban az elhelyezés automatikus kialakítására koncentráltunk.

7.2 A megoldás első formája

A fentiek alapján következzék az általunk javasolt megoldás. Mint az előző pontban megnéztük, itt egy, az oldal felépítését meghatározó dobozok elrendezését automatikus generáló módszert fogunk bemutatni, illetve elhelyezzük. Ezt a módszert a tervezési folyamatban szeretnénk elsősorban felhasználni.

Tehát olyan genetikus algoritmust szeretnénk létrehozni, amely képes egy Web oldalt felépíteni a rajta található dobozok segítségével, úgy, hogy a tervek generálásával a tervező vagy fejlesztő munkáját támogassa. Ehhez nézzük végig az 5.3.1 fejezetben tárgyaltak szerint mely elemekből építjük fel az algoritmusunkat.

7.2.1 Paraméterek

Az algoritmus alap paraméterei, vagyis a populáció és a generációszám meghatározására nem végeztünk külön számításokat, hanem tapasztalati tények alapján próbáltuk meg beállítani. A populációszámnál elvileg figyelembe kellene venni a dobozok számát, azonban tekintve, hogy ez az érték a gyakorlati felhasználást figyelembe véve 20-30-nál nem lesz több, így a populáció méret választható konstansra is, mi ezt az értéket 200-ra választottuk.

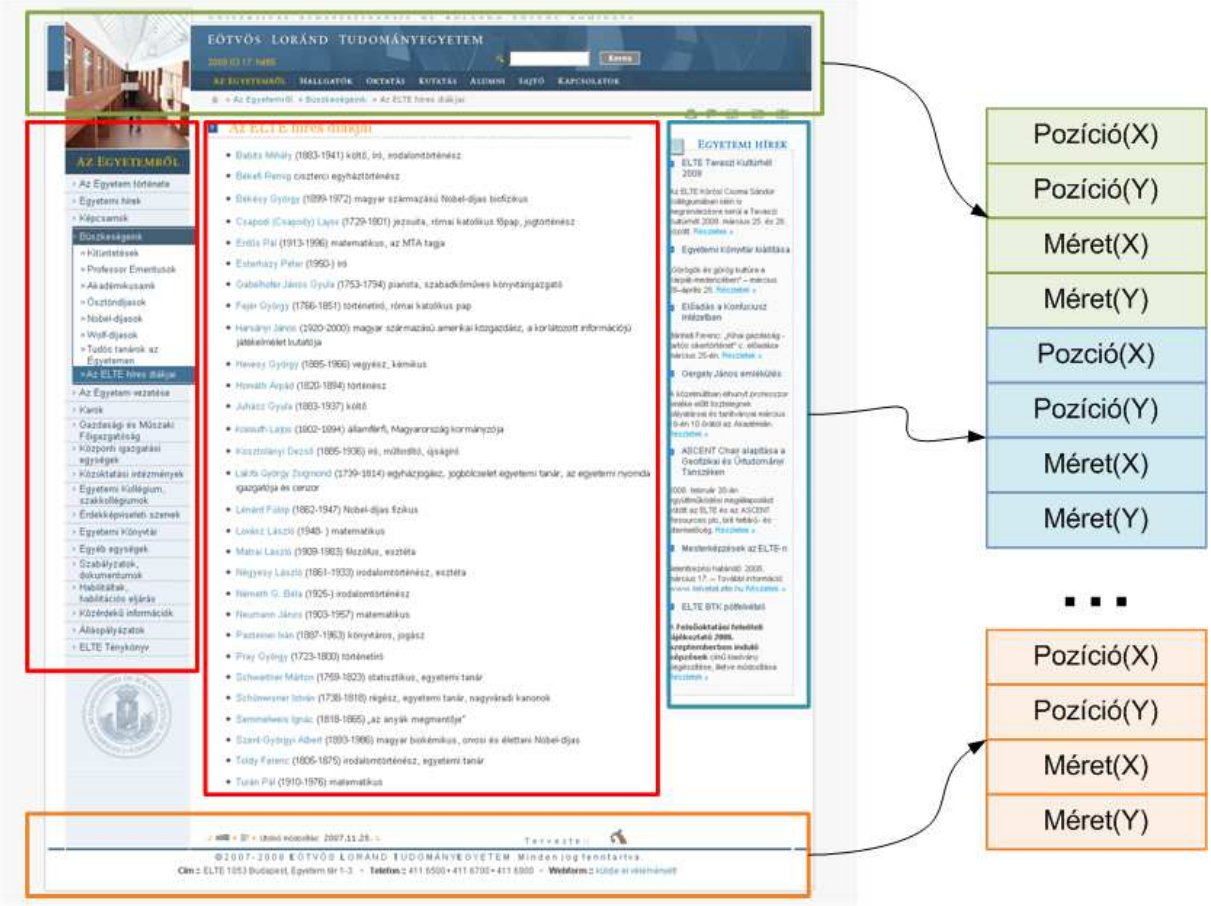
A generációszámot nem határoztuk meg konkrét értékben, hanem a humán kiértékeléstől függ a futási idő.

7.2.2 Kódolás

A CSS szabványnak köszönhetően minden dobozt le lehet írni négy paraméterrel, a doboz oldalon belüli elhelyezkedését meghatározó x és y koordinátákkal, illetve a dobozok méretét leíró szélességgel és magassággal (x és y irányhoz tartozóan). A dobozhoz tartozó kezdőpontot a képernyő bal felső sarkából kiindulva kell meghatározni. Mind a koordináták, mind a méretek egész számok.

Az algoritmusban ezekhez az értékekhez tartozó természetes kódolást fogja alkalmazni, erre egy példát láthatunk a következő ábrán. Vegyük észre, hogy ekkor a kromoszóma hossza a dobozok számának négyszerese.

Az egyes paramétere viszont érték tartományt is meg kell adni, hiszen nincs értelme úgy beállítani a pozíciót, hogy kilógjon a képernyőről (ehhez alapul lehet venni a ma már általánosan elterjedt 1024x768-as képernyő felbontást). A méreteknél is dobozonként meg lehet adni egy olyan tartományt, amely a doboz tartalmához igazodik.



Ábra 8: Web oldalon lévő dobozok és azok kódolása

7.2.3 Inicializálás

Az általános ökölszabály szerint az inicializáláskor érdemes véletlenszerű egyedekkel feltölteni a populációt, hogy a keresési tér egyenletesen le legyen fedve. Ezt jelen esetben annyival módosítható, hogy az egyes paraméterekhez tartozó értéktartományokat is vegye figyelembe. Ekkor az algoritmus a későbbiekben sem visz a tartományon kívülre, az egész értékekre alkalmazott genetikus műveletek nem állítanak elő új értékeket.

7.2.4 Rátermettség függvény

A módszerünk humán kiértékelést fog elsősorban alkalmazni, vagyis ez egy interaktív genetikus algoritmus (IGA) lesz. Ez jelen esetben azt fogja jelenteni, hogy a generált megoldásokat nem egy konkrét függvény értékeli ki, hanem az algoritmust futtató ember, aki minden generációban eldönti, hogy a generált laptervek közül melyek számítanak jónak.

Ez így azonban nagyon lassú és fárasztó módszer lenne, pedig pont a tervezők és a fejlesztők feladatát szeretnénk megkönnyíteni. Egyszerre több száz, egymáshoz esetleg nagyon hasonló terv kiértékelése több generáción keresztül nem segíti elő a módszer gyors alkalmazását. Ezért módosítjuk a következőképpen.

A humán kiértékelés nem minden egyedet, hanem csak 9 egyedet értékel minden lépésben. Ezeket úgy különítjük el, hogy a populációban lévő jellemző egyedeket próbálunk ide kiválasztani. Ezen belül is 6 helyre olyanokat választunk, amelyek az egyedekhez tartozó

egyes gén értékek tekintetében középső helyen vannak. Vagyis ha egy k hosszú kromoszómát veszünk, és a gének értékei koordinátákat határoznak meg a k dimenziós térben, akkor azokat az egyedeket szeretnénk ide választani, amelyek a körülírt gömb közepétől minimális távolságban helyezkednek el. A maradék 3 helyre pedig minél távolabbi egyedeket választunk ki. Ezzel a módszerrel biztosítható az, hogy ahogyan fejlődik a populáció egyre jobb egyedek kerülnek a kiértékelő elé, illetve a gömb széléről választott egyedekkel a központi értéktartománytól távol eső területek is felfedezésre kerülhetnek. A humán kiértékelő minden kiválasztási lépésben a felajánlott 8 egyed közül 0, 1 vagy 2 egyedet adhat meg, amelyet a legjobbnak tart. A kiválasztott egyedek rátermettség értéke a legnagyobb lesz, a pontos értéket a gömb sugarától függően lehet skálázni.

Ez így még nem rendel értéket a többi egyedhez, illetve azt is szeretnénk, hogy nem minden generációban legyen humán kiértékelés. Ezért a további egyedek értékét egy távolság függvény alapján fogjuk meghatározni. A távolság függvény legyen az adott egyed kiválasztott egyedektől történő átlagos távolsága. Egy egyed esetén ez egyértelmű, két egyed esetén a távolságok minimuma. Amennyiben nincs kiválasztott egyed akkor sem a gömb közepe, sem a gömb pereme nem tartalmaz alkalmas egyedeket, vagyis teljesen meg kell változtatni az egyedek eloszlását. Ezért a populáció felét újra generáljuk véletlenszerűen, így vélhetően az egyedek által kifeszített tér kellően átalakul. Az így kapott eredmény további finomítása, hogy azok az egyedek, amelyek egymást átfedő dobozokat tartalmaznak büntetőpontokat kapjanak.

Mivel így már nem kell minden generációban humán kiértékelés, ezért nem kell minden generációt kiértékelni, mi a ezt úgy választottuk meg, hogy minden 4. generációban legyen erre szükség. Ezekben a generációkban a korábban kiválasztott egyedek jelentik továbbra is a referencia pontot.

Ezzel a fitnessz kiválasztással a rendszer megtartja humán kiértékelés előnyét, miközben képes nagy számú megoldást egyszerre kiértékelni.

7.2.5 Kiválasztás

Rátermettség arányos kiválasztást választottunk. Ebben a rendszerben a kiválasztott egyedek automatikusan a legmagasabb fitneszt kapják meg. Elitizmust is alkalmazunk, ekkor a kiválasztott egyedek a következő humán kiértékelési körig minden generációban visszakerülnek a populációba változtatás nélkül.

7.2.6 Keresztezés

A megoldások értékeiről nem rendelkezünk előzetes információval, így a legjobb megoldásnak az uniform keresztezés tűnik. Ezzel a választással a gén értékek nem fognak kimenni a megadott értéktartományból.

7.2.7 Mutáció

A mutációs rátát a kromoszómahosszhoz állapítjuk meg. Vegyünk egy k hosszú kromoszómát, ekkor a mutációs ráta legyen 1/k. Ezt minden génre hajtsuk végre. Az új érték a génnek megfelelő értéktartományból legyen véletlenszerűen kiválasztva.

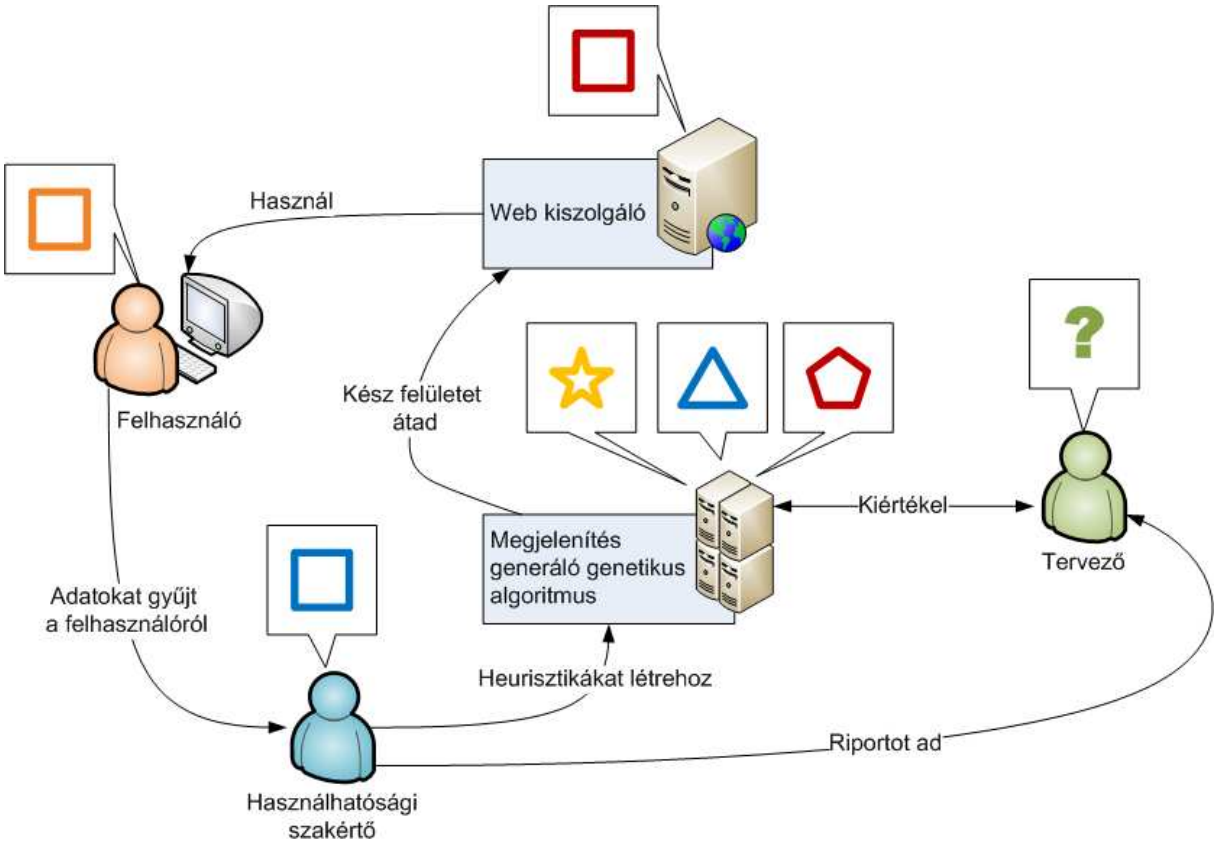
7.2.8 Heurisztikák

Azt szeretnénk, ha már az algoritmus ebben a formában is be tudna fogadni további heurisztikákat, a dobozok elhelyezkedésére vonatkozóan. Ez jelen esetben azt jelenti elsősorban, hogy az egyes dobozokra meghatározható értéktartományokat a felhasználói visszajelzések alapján lehet módosítani. Erre egy jó példa lehet, hogy a felhasználók az elsődleges navigációt a bal oldalon szokták keresni, ezért érdemes olyan értéktartomány megadni ide, amelyik ezt az elhelyezést támogatja.

7.2.9 A rendszer összeállítása

Az algoritmus felépítése után már csak az kérdéses, hogyan fog ez a teljes folyamatba illeszkedni.

Az elképzelésünket az alábbi ábra mutatja be.



Ábra 9: Web felület tervezés genetikus algoritmus segítségével

A genetikus algoritmust futtató modul a Web kiszolgáló elé épül be, a megjelenítendő oldalak vázát generálja le neki. Ehhez egyrészről a tervezővel, aki a humán kiértékelő, áll interakcióban. Végül további heurisztikák kerülhetnek beépülésre a használhatósági szakértő részéről.

7.3 Összegzés

Bemutattunk egy genetikus algoritmust, amely képes az oldalak felépítését meghatározó dobozok elhelyezkedését megadni, illetve használhatósági heurisztikákat is be tud fogadni. Ez

a módszer így már képes arra, hogy a tervezők munkáját támogatva olyan verziókat is megvizsgáljon, amely egyébként nem merült volna fel. A következő fejezetben megvizsgáljuk, hogy a tervezés további fázisaiban milyen további kiterjesztési lehetőségek képzelhetőek el erre az alapra építkezve.

8 Kiterjesztések

Az előző fejezetben ismertetett algoritmus már eléri a kitűzött célt, azaz képes szoftver ergonómiai információkat befogadni, miközben automatikusan állít elő Web oldalak felépítéseket. Ebben a fejezetben áttekintjük a kiegészítési lehetőségeket, illetve bemutatjuk, hogy milyen további eljárások építhetőek erre az alapra.

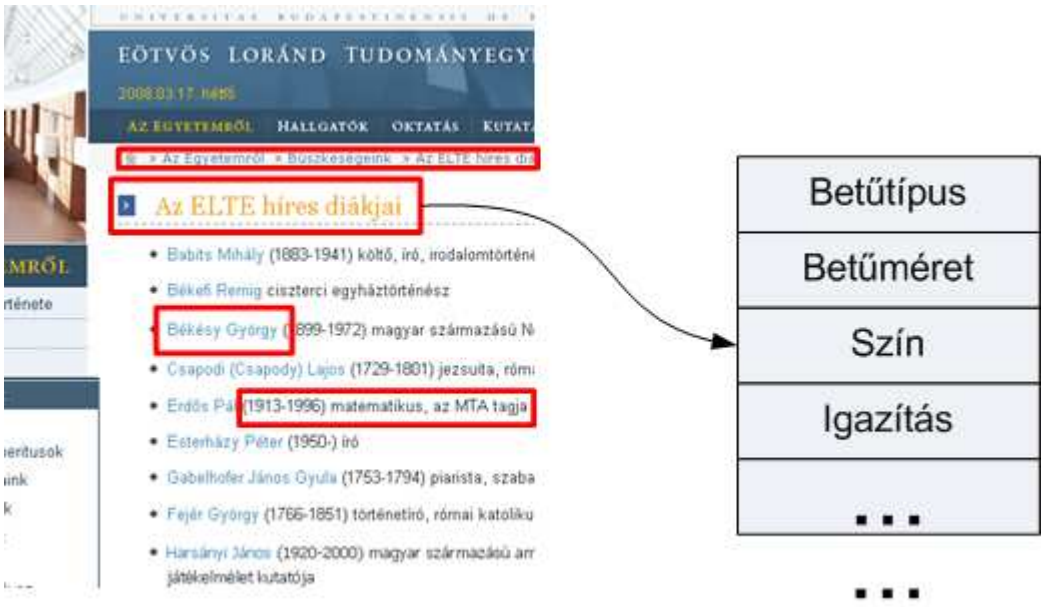
8.1 Elrendezési algoritmus javítása

Az elrendezés kialakító algoritmus jelenlegi formájában nem képes előzetes preferenciákat kezelni. Egy egyszerű és logikus kiegészítés lehet, hogyha az egyes dobozokhoz tartozó gén sorozatokat kiegészítjük egy további génnel, amely az adott dobozhoz tartozó priorítást tartalmazná.

Ezeket a priorítás értékeket a tartalmi elemeket tartalmazó dobozokhoz előre kellene meghatározni. Az elsődleges elv itt az lehet, hogy kellően alacsony priorítással a doboz el is tűnhetne a generálás közben (a kezdő koordináták egy extremális értékre kerülnek). A magas priorítású elemek mindig megjelennek, és lehetőleg az oldal felső vagy bal oldalára rendeződnek (a felhasználók elsőre ezeket nézik meg).

8.2 Oldal elemek generálása

Az automatikus generálásban a következő lépcső az lehet, hogy nem csak a tartalmi elemeket tartalmazó dobozokat, hanem a tartalmi elemek kinézetét is automatikusan generáljuk. Erre a CSS szabvány kifejezetten alkalmas, gyakorlatilag az elemek minden tulajdonsága beállítható, és ezek a paraméterértékek véges halmazok elemei.



Ábra 10: Weboldal reprezentációja stíluslapok alapján

A fenti ábrán látható egy egyszerű példa, hogy a szöveges elemek különböző kategóriáit önálló génszekvenciákkal reprezentálhatjuk, amelyek minden lényeges információt tartalmaznak az adott elemről.

Ebben az esetben már mindenképpen szükség van heurisztikus információ beépítésére, amelynek több oka is van:

- Nem értelmezhető vagy nem releváns paraméterek. A CSS-ben az elemek tetszőleges formázásra is lehetőség van, de előfordulhat hogy ezek egy nagyobb részére nincs is szükségünk (például keretek felhelyezése egy lista elemei köré). Ezek feleslegesen lassítják az algoritmus teljesítményét, tehát hatékonyabb, ha már nem is veszi őket figyelembe. Ezt előzetesen kell beállítani.

- Általános ergonómiai elveket is figyelembe kell venni (például fehér alapon fehér írás vagy zöld alapon piros szöveg). Elvileg ezek szerepelhetnek a fitnessz függvényben is, bár ez lassítja az algoritmus futását. Hatékonyabb, ha a reprezentációt úgy alkotjuk meg, hogy az összetartozó vagy egymást kizáró értékekre (értéktartományokra) további szabályokat határozunk meg. A genetikus algoritmus logikájából adódóan ezeket a szabályokat is beépíthetjük az algoritmusba, tovább finomítva azokat.

- Az elemek közti relációkat is meg lehet fogalmazni (például a címsor betűmérete legyen nagyobb, mint a rendes szövegé). Ezzel az előzetes vizsgálatok eredményét is be lehet építeni a fitnessz függvénybe büntetőpontok formájában. Így a egy újfajta megoldást adó eredmény sem záródik ki, a keresés fő irányát mégsem ez lesz.

Ezzel a módszerrel már gyakorlatilag egy egész oldal design-ja automatikusan generálható.

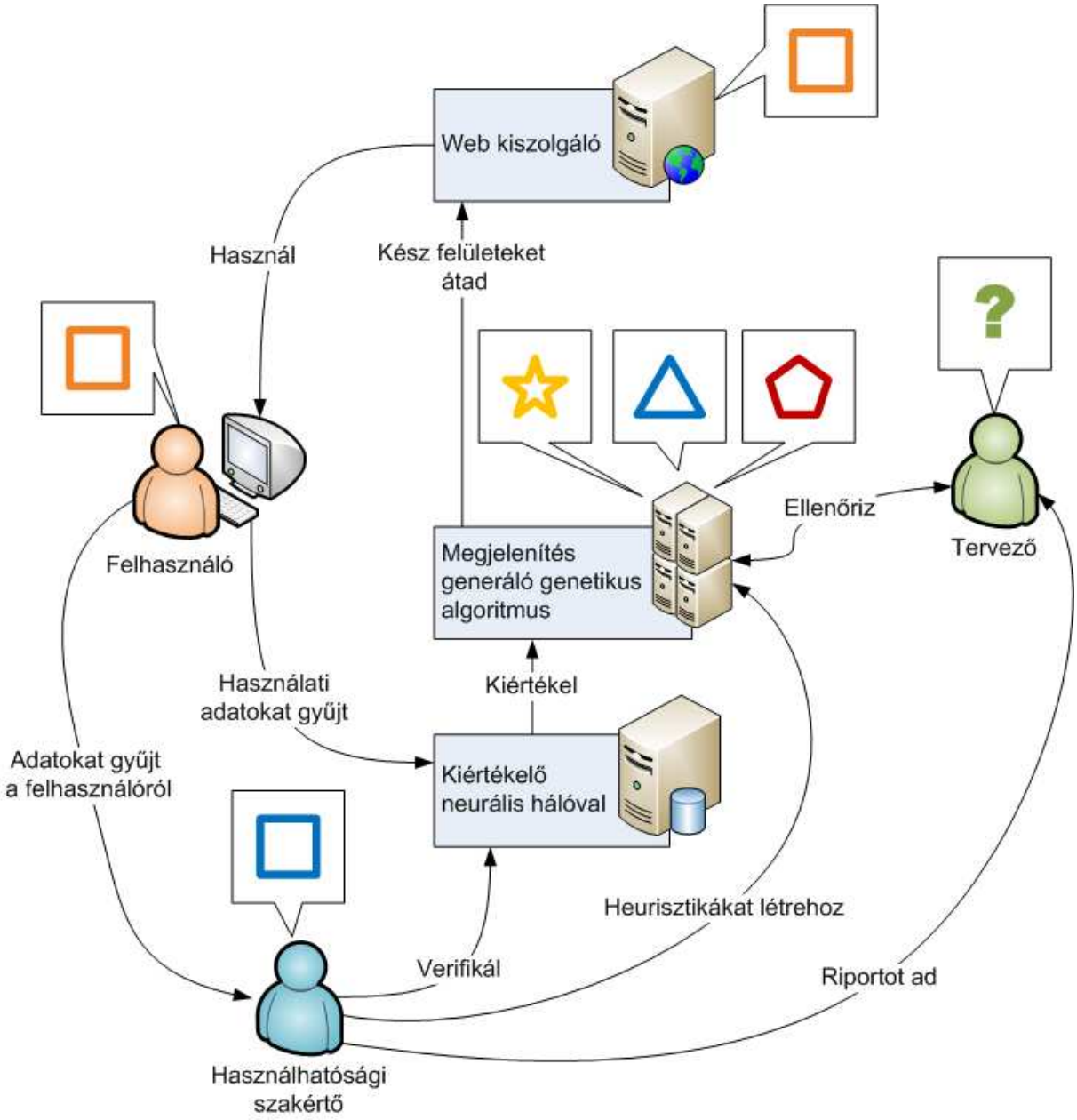
8.3 Automatikus kiértékelés

A végső cél továbbra is az, hogy a generálás automatikusan menjen végbe, és a humán beavatkozás csak a felügyeletre korlátozódjon.

Ehhez arra van szükség, hogy a felhasználói akciók eredménye közvetlenül egy automata kiértékelő rendszerbe fusson bele. A javaslatunk szerint ez a kiértékelő rendszer egy neurális háló lenne, amely egyrészről folyamatos tanulásként kaphatna további használhatósági bemenetet a klasszikus módszerekkel gyűjtött információkból Másrészről a felhasználói interakciókat közvetlenül is megkapná elemzésre, további mintákat felismerve benne. Véleményünk szerint azért lenne alkalmas erre egy neurális hálóra épülő konkrét rendszer, mert ezek az eszközök már bizonyítottak a mintafelismerés területén, és itt ez a lényeg, a felhasználó viselkedésében kimutatható minták felderítése a kiértékelő rendszer feladata.

A kiértékelés eredménye a generáló alrendszer bemente lenne, mint további heurisztikák egyrészről, másrészről a konkrét egyedek rátermettségét határozná meg. Ez utóbbihoz további kutatás szükséges, meg kell határozni azokat a metrikákat, amelyek egy konkrét felületi elemhez kapcsolódó sikeres felhasználói akciókat írják le.

Az alábbi ábrán látható az automatikus kiértékeléssel kombinált generálót tartalmazó rendszer vázlatos képe.



Ábra 11: Automatikus web felület tervezés

A konkrét felhasználás kettős lehetne:

- On-the-job felhasználás: A Web oldalt működés közben, folyamatosan elemezné, és bizonyos idő elteltével a felgyülemlett adatok alapján a genetikus algoritmus segítségével előállítaná a oldal új verzióját. Ez nem zavarná meg a felhasználókat a web oldal folyamatos változtatgatásával, de lehetőséget adna az inkrementális változtatásokra.

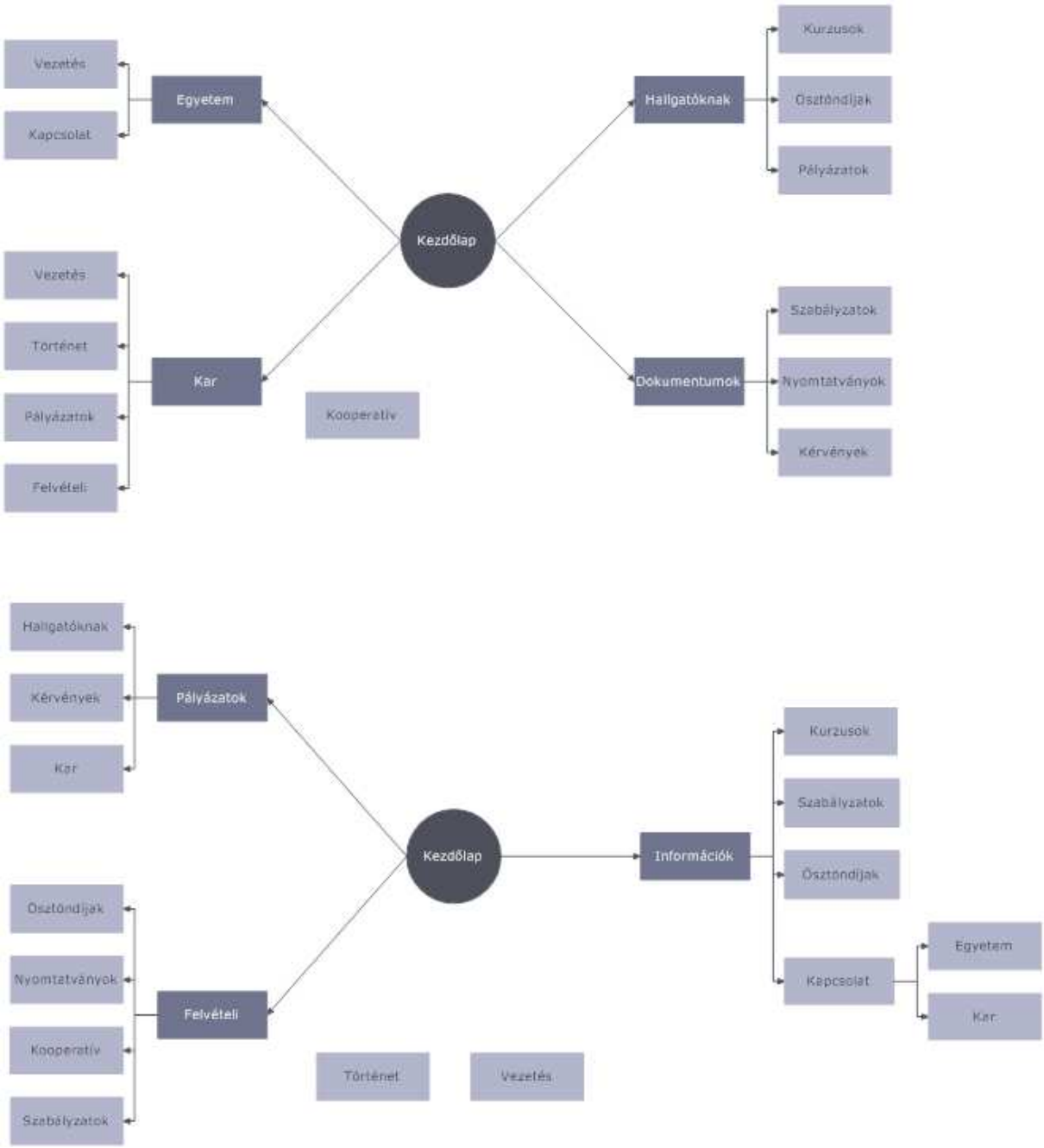
- Fejlesztési segítség: Egy oldal fejlesztése közben a rendszer kiegészítheti a más használhatósági módszerekkel kapott információkat, illetve támogatja az új verziók elkészítésével a fejlesztők munkáját.

A rendszer fizikai megvalósítása a fentiekkel együtt is további elemzést igényel, hiszen mind a megvalósító neurális háló paraméterei beállításához, mind az eredmények interpretációjához további vizsgálatok szükségesek.

8.4 Oldal struktúra generálása

Az ergonómiailag helyes oldalak még nem garantálják azt, hogy a felhasználók ténylegesen megtalálják az őket érintő információkat.

Nézzük meg az alábbi ábrán található két website-ot.



Ábra 12: Két hasonló információtartalmú, de eltérő felépítésű website

A két website egymástól eltérő struktúrában, de hasonló információkat tartalmaz. Viszont lehet, hogy az egyik vagy a másik felépítés jobban megfelel a felhasználóknak

Az ilyen és ehhez hasonló felépítések könnyen elállíthatóak genetikus algoritmusok segítségével is, egy ehhez hasonló módszer található a genetikus programozásban is (Koza, [16]). A genetikus programozással analóg módon az algoritmus fa formátumban állítja elő az egyedeket, amely pont megfelel a honlapok hierarchikus felépítésének. A módszer általánosításával lehetővé válik az eltérő navigációs struktúrájú honlapok előállítása (fa helyett gráfokat kell előállítani).

A genetikus programozásban meglévő eredményeket több szempontból ki lehet használni, például a heurisztikák meghatározására.

A kiértékelés más design elemekhez képest könnyebben meghatározható, a bejárt út hosszát és a felhasználókat érdeklő végcélt kelt összevetni.

Kérdéses viszont egy így generálódó website karbantarthatósága. Feltételezhető, hogy a karbantartás egy olyan felületen történik, ahol a tartalmakat nem hierarchia szerint, hanem valamilyen más eszközzel csoportosítjuk, így ez a költség is a nem dinamikusan generált felépítésekhez hasonló.

9 Teszt implementáció

A 'WebGen Projekt' keretredszerszerűen az ergonómiailag helyes weboldalak automatikus generálására ad egy teszt megoldást. Ezt a projekt során keletkezett eredmények vizsgálatára hoztuk létre. A projekt *.Net Visual Studio 2008* fejlesztőkörnyezetben C# programozási nyelven dinamikus ASP web-alapú eszköz segítségével készült.

9.1 A genetikus algoritmus implementálása

Az objektumelvű programozási technikát (OOP) kihasználva az algoritmus osztálya (GenAlg) tartalmazza azon populációt, amely egyedeinek értékei határozzák meg az oldalak Cascade Style Sheet (CSS) elemeinek értékeit. Az algoritmus a populáció egyedeiből választva épít fel weboldaltípusokat a tervező számára, melyek közül választással a populáció a következő generációba lép.

9.2 A szükséges megjelenítési eszközök

Web Form Page (System.Web.UI.Page) elemei:

- *Panel*
- *Label*
- *List<>*
- *Table*
- *Button*

Az ASP környezet lehetővé teszi ezen elemek XHTML szintű kód generálását, így a böngészők számára feldolgozhatóvá válnak. Minden ilyen elem ellátható egyedi azonosítóval, amelyre a generált stíluslapon hivatkozhatunk. A jelenlegi implementáció a stílusformázásokat az XHTML oldalba generálja, de látható, hogy az ilyen formázások egy .css kiterjesztésű CSS fájlba is gyűjthetők.

9.3 A CSS stíluslap

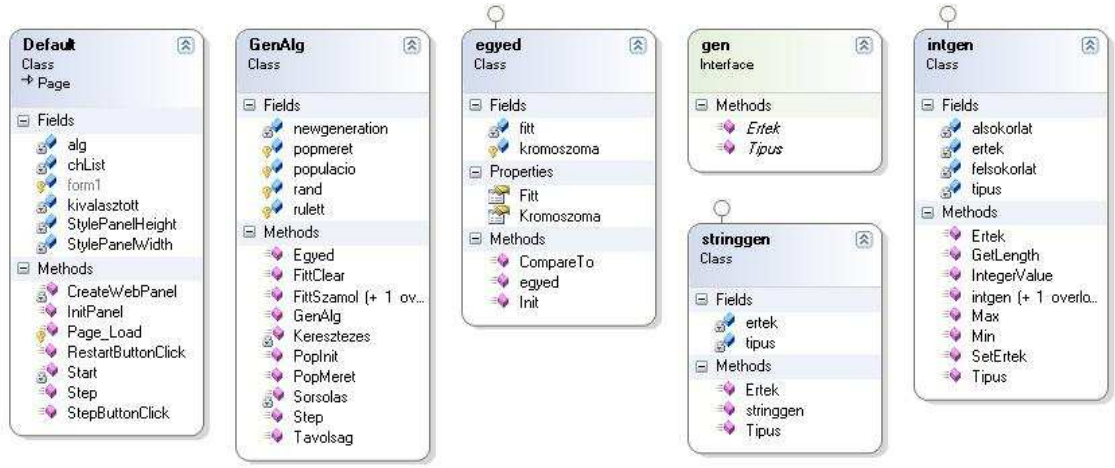
A stíluslapok bármely tulajdonságát a számunkra megfelelő értékre állíthatjuk. A formázás ASP-beli eszköze a *Style* osztály. Az alábbiakban látható az elrendezések teszteléséhez használt stíluslap.

```
.div
{
     height : 100px;
     width : 200px;
     background-color : #777777;
     padding : 2px;
     margin-top : 2px;
```

```
    margin-bottom : 2px;
    margin-left : 2px;
    margin-right : 2px;
}
```

9.4 A genetikus algoritmus osztályai

A *GenAlg* osztály tartamazza a populációt (List<egyed>), amely *egyed* osztályú elemekből áll. Az egyedek attributumai között szerepel a rájuk jellemző génlista (kromoszóma). A gének a feladat szemponjából tárolt értékek két fajtájával rendelkezik. Ezek közül az egyik az egész számokat(*intgen*), a másik a karaktersorozatokat tartalmazó gének(*stringgen*). A gének típusa a stíluslapban látható tulajdonságok, az értékei a gének konkrét értékei.



Ábra 13: A megvalósított osztályok

9.5 A futtatás

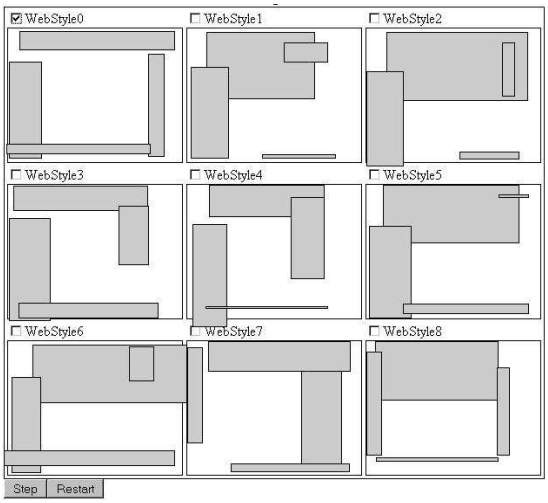
A genetikus algoritmus az előző fejezetekben felvázolt módon fut. A rátermettség függvénybe beépíthető:

- forráskódszinten az egyedek sáját értékeiből számított vagy az egyedek egyes génjeinek egymáshoz való viszonya,
- egyéb magasabb szintű heurisztikák.

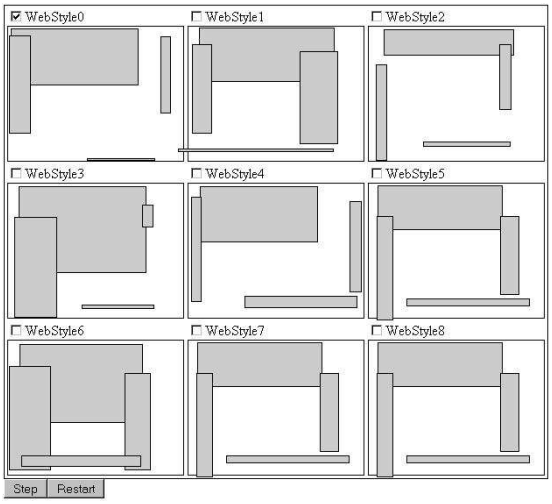
Ilyen például az, ha egy-egy egyed, jelen esetben például HTML panel oldalainak viszonyát is díjazhatjuk. Az egész populációra kiszámítva a fittértékeket és azok szerint a populációt sorrendbe téve további függvényvizsgálatot tehetünk, amely alapján a generációk közötti fittértékek különbségéből adódó számítási pontosságok javíthatóak. Az egyedek távolságára több távolságfüggvényt is definiálhatunk (lineáris, négyzetes).

A futtatás során ki kell választani a megfelelő WebStyle-t, amely stílust meghatározó egyedhez közel azonos tulajdonságokkal rendelkező egyedeket részesíti előnyben az algoritmus. A következő lépés során az első hat legjobb egyedet és három legrosszabb egyedet jelenítünk meg, amelyek közül a lejobban tetsző stílust választhatjuk. Az alábbi ábrákon megfigyelhető a kezdeti véletlenszerű tulajnonságok a 25-30 lépés (generáció)

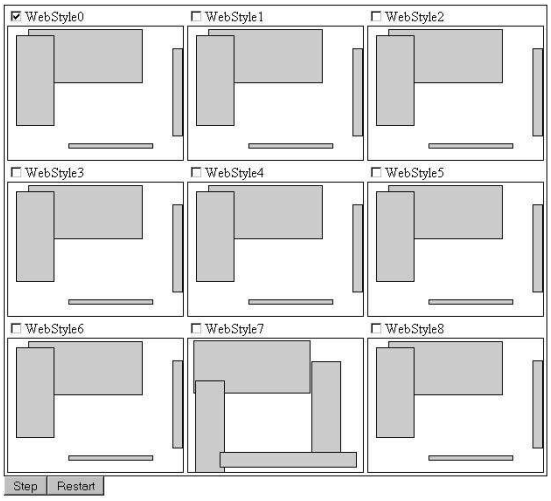
folyamán hogyan terjesztik el a 'jó' tulajdonságokat. A végső generációban már csak néhány mutálódott egyed található.



Ábra 14: Az első generáció



Ábra 15: Az algoritmus futás közben



Ábra 16: Eredmény, közel azonos egyedek

10 Jövőkép

Az elmúlt évek fejlődési tendenciáit figyelembe véve valószínűsíthető, hogy az egyes fejlesztési feladatok automatizálása már nem csak a mechanikus feladatokra fog kiterjedni, hanem olyan területekre is, amelyeket ma még teljesen emberek végeznek el. Ezért gondoljuk azt, hogy az itt bemutatott módszerekhez nagyon hasonló vagy éppen pontosan ilyen eszközök fogják támogatni a honlapok készítőit a közeljövőben.

Hosszú távon érdekes kérdés lehet, hogy van-e annak reális esélye, hogy ezek a módszerek ne csak támogatást nyújtsanak, hanem valóban a mesterséges intelligencia hozzon meg kreativitást igénylő döntéseket is. A szerzők meg vannak arról győződve, hogy ez a nem túl közeli, de nem is a nagyon távoli jövőben egy reális lehetőség,. Bár ezzel együtt továbbra is kérdéses, hogy ezt pontosan milyen algoritmusok vagy módszerek segítségével érik el, illetve milyen mértékig képzelhető el az emberi munkaerő kiváltása. A legvalószínűbbnek az emberi és számítógépi rendszerek előnyeit ötvöző ember-számítógép rendszerek tűnnek a legvalószínűbbnek.

A technológia állandó fejlődése már a közeljövőben is jelentős változásokat hozhat a felületek tervezésében. Ennek egyik jó példája a vektoros felületek megjelenése. Ezzel a teljesen automatikus generálás egyik nagy akadálya, a grafikai elemek fix méretezése is eltűnik, míg ugyanis ezek raszteres formátumban vannak jelenleg, már meg is jelentek azok a böngészők, amelyek támogatják az SVG vektoros képi formátum megjelenítését (például FireFox, [15]).

A Web oldalak tervezésében már ma se csak információ közlő oldalakat kell vizsgálni, hanem alkalmazásokat is .Úgy tetszik, hogy a közeljövőben a webes alkalmazások elterjedése csak további problémákat szül, hiszen az új felületek megjelenése újfajta felületi problémák megjelenését is előreveti. Erre felkészülve a rendszereket, így az általunk bemutatottat is előbb-utóbb fel kell készíteni az ilyen alkalmazások megtervezésének támogatására is.

Zárszóként annyit mondhatunk, hogy az ember a jövőben várhatólag még több időt fog eltölteni számítógép előtt, még több feladatának végrehajtásához van szüksége a felületek gyors és hatékony használatára. Ezért szükséges fokozott figyelmet fordítani az ezt megkönnyítő módszerek kialakítására, a támogató rendszerek fejlesztésére.

11 Irodalomjegyzék

[1] Hopgood, A. A.: Intelligent Systems for Engineers and Scientists 2nd Edition; CRC Press; 2001

[2] http://www.aaaipress.org/Classic/Buchanan/buchanan.html

[3] Hun, S. H.; Yang, H.: Screening important design variables for building a usability model: genetic algorithm-based partial least-squares apporach; International Journal of Industrial Ergonomics 33, 159-171; 2004

[4] Asslani, A.; Lari, A.: Using genetic algorithm for dynamic and multiple criteria web-site optimizations; European Journal of Operational Research 176, 1767-1777; 2007

[5] Ivory, M. Y.; Hearts, M. A.: The State of the Art in Automating Usability Evaluation of User Interfaces; ACM Computing Surveys, Vol 33., No 4., 470-516; 2001

[6] Quiroz, J. C.; Dascalu, S. M.; Louis, S. J.: Human guided evolution of xul user interfaces; CHI 2007; ACM Press; 2007

[7] Quiroz, J. C.; Louis, S. J. ; Dascalu, S. M.:Interactive evolution of xul user interfaces; GECCO 2007; ACM Press; 2007

[8] http://www.marketingprofs.com/5/syrett6.asp

[9] Galitz, W. O.: The Essential Gudie to User Interface Design, Third Edition; Wiley Publishing; 2007

[10] Sántáné-Tóth E.: Tudásalapú technológia, szakértő rendszerek – Javított és bővített kiadás; Dunaújvárosi Főiskola Kiadó Hivatala, Dunaújváros; 2000.

[11] Holland, J. H.: Adoptation in natural and artificial systems; University of Michigan Press, Ann Arbor, 1975

[12] Goldberg, D. E.: Genetic Algortihms in Search, Optimization and Machine Learning; Addison-Wesley; 1989

[13] http://www.useit.com/alertbox/9605.html

[14] http://www.w3.org/Style/CSS/

[15] http://www.mozilla.org/projects/svg/

[16] Koza, J. R.: Genetic Programming; MIT Press; 1992