Redundáns kifejezések

A példák korlátozásainak modellezése során felmerült bennünk annak lehetősége, hogy redundáns kifejezések fordulhatnak elő a feladatokban, azaz vannak olyan megkötések vagy megkötés-halmazok, amelyek elhagyása esetén továbbra is egyetlen, az eredeti feladatkiírásnak megfelelő megoldást találunk. Kutatásunk utolsó szakaszában célunk ezek megkeresése, továbbá annak meghatározása volt, hogy elhagyásuk miképpen befolyásolja, vagy befolyásolja-e egyáltalán a különböző megoldók futási illetve megoldási idejét.

Mivel megoldóink gyorsasága már a korábbiakban bebizonyosodott, ezért jó eséllyel gondolhattunk arra, hogy amennyiben egy vagy több korlátozást eltávolítunk abban az esetben rövid idő alatt meg tudjuk határozni, hogy továbbra is egyértelmű-e az eredmény. A feladatok jellegéből adódóan egyértelműnek azt az esetet tekintettük, amikor a program lefutása után csak egy megoldást találtunk, amely az eredeti – minden kikötést tartalmazó – feladatkiírásnak is megfelel. Módszerünk a következő volt: minden kikötést egy elágazásban helyeztünk el az alábbi ábrán látható módon.

constraint if kivesszuk[4]=1 then true else

forall(p in PEOPLE)(necklace[p]=sapphire <-> age[p]=fiftyfive)

endif;

A „kivesszuk” egy, a korlátozások számával egyező elemszámú bináris értékeket tartalmazó tömb. Amennyiben a tömb adott indexű eleme 1 értéket tartalmaz, akkor a korlátozás a konkrét kifejezés értéke helyett egyszerűen igaz értéket vesz fel, így a kifejezést figyelmen kívül hagyjuk. Amennyiben viszont az adott sorszámú elem 0 értékű, akkor a különben ágban elhelyezett, a feladatleírásnak megfelelően modellezett kifejezést kezeljük korlátozásként.

Ezen elvet követve mindegyik korlátozást egy a fentihez hasonló vezérlési szerkezetbe ágyaztunk. A „kivesszuk” tömböt pedig a könnyebb szerkeszthetőség érdekében nem a modellen belül definiáltuk, hanem értékét külön fájl(ok)ból olvastuk be. Mivel egyes példák nagy mennyiségű kikötést tartalmaznak és ebből következően rengeteg féle kombinációban tudnánk ezeket elhagyni, így igyekeztük ezeket a teszteket valamilyen szinten automatizálni. Az adatfájlokat (amelyekben a „kivesszuk” tömböt definiáltuk) egy C# programmal generáltuk oly módon, hogy első körben mindegyikben a tömbnek pontosan egy eleme lesz 1 értékű, így értelemszerűen a korlátozások darabszámával egyező adatfájl keletkezett. Ezeket parancssorból a modellel együtt parancssorból futtattuk és feljegyeztük azokat a sorszámokat, amelyek esetén egyértelmű eredmény született. Második körben az imént említett C# program segítségével legeneráltuk ezen összegyűjtött sorszámok összes létező, 2 tagból álló kombinációját, majd újabb parancssoros futtatás következett. Megint feljegyeztük az eredményeket és ezt ismételtük mindaddig, amíg a sorszámok értékkészlete le nem szűkült teljesen. A kapott eredmények meglepőek voltak.

Még egy olyan kicsi és gyenge nehézségű, kevés kikötésből álló példánál is, mint amilyen a „Movies Night”, relatíve jelentős mennyiségű kikötés elhagyása esetén is teljesült, hogy az egyetlen kapott megoldás megegyezett a kiindulási feladat megoldásával. Ez esetben az összes 13-ból önmagában 3 korlátozás is redundánsnak bizonyult. Ezekből pedig 3 olyan különböző 2 tagból álló kombináció volt alkotható, amelyek egy időben is elhagyhatóak. Ezen számok a nagyobb példák esetében még tovább nőttek. Az általunk feldolgozott példák esetében a redundáns kikötések eloszlásait az alábbi táblázat tartalmazza.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *Movies Night* | *Fundraising Dinner* | *Gardens* | *Saját óriás példa* |
| Önmagában elhagyható korlátozások száma | 3 | 7 | 5 |  |
| 2 tagú kombinációk | 3 | 15 | 5 |  |
| 3 tagú kombinációk | 0 | 13 | 2 |  |
| 4 tagú kombinációk | 0 | 4 | 0 |  |

*Példánként az elhagyható korlátozások és azok lehetséges kombinációinak száma*

Az adatokból kitűnik, hogy azon korlátozások közül, amelyek önmagukban elhagyhatóak, nem választhatunk ki tetszőleges kombinációban 2-t vagy többet. Azaz már meg kellett válogatnunk, hogy pontosan melyek együttes elhagyása ad egyértelmű eredményt. Tehát például ha két kifejezés külön-külön elhagyható, az korántsem jeleni azt, hogy ezek egyszerre is elhagyhatók. Vegyünk egy konkrét példát: a „Fundraising dinner” esetében az elhagyható korlátozásokból(7) alkotható összes 2 tagú kombináció száma, de ezek közül csak 15 ilyen pár felelt meg a kitételeknek. Jellemzően minél több kifejezést szeretnénk egy időben feleslegessé minősíteni, annál inkább szűkül a választási spektrumunk, annál kevesebb kombináció jöhet számításba. Egy idő után elértük azt a küszöbértéket, amelynél több kifejezés nem hagyható el egyszerre. Ez jellemzően az összes kikötés számának 15-23%-a volt.

A fentiek továbbá azt is jelentik, hogy a feladat megalkotói jelentős redundanciával dolgoztak a kikötések meghatározásánál. Ám ezt megállapítani a klasszikus módszerrel, miszerint egyszerűen kevesebb korlátozással kiírt példát akarunk megoldani papíron, sokszor egy átlagos képességű embernek nem lehetséges vagy túlságosan hosszú ideig tart. Esetünkben viszont egzakt matematikai-logikai modellekkel leírt korlátozásokat a számítógéppel oldattuk meg, amely értelemszerűen az emberénél jóval nagyobb számítási kapacitással rendelkezik. Így a nem feltétlenül szükséges kifejezések felderítése viszonylag rövid idő alatt, egyszerű eszközökkel kivitelezhető volt. Ez felhasználható a feladványok készítésekor a minél nehezebb feladatok eléréséhez, melyekben minden információt fel kell használni a megoldás során.

Futásidők tekintetében a módosítatlan, teljes feladathoz képest a kihagyásoknál jelentős változás sem a parancssoros sem a grafikus felületről történő futtatás során nem történt, bár egyes esetekben kis mértékben a futásidők megnőttek. A megoldók hatékonyságáról ismét megbizonyosodhattunk, mivel kevesebb korlát, így a megoldáskeresés során kevesebb támpont esetén is komoly gyorsasággal adtak eredményt.