Hibadiagnosztika elosztott rendszerekben

Bevezető

A mérés során megvalósítandó feladatok két csoportba oszthatók: determinisztikus és valószínűségi diagnosztikai algoritmusokra.

- A **determinisztikus algoritmusok** adott feltételek teljesülése esetén képesek teljes és konzisztens diagnosztikai képet előállítani. *Teljesnek* akkor nevezünk egy diagnosztikai eredményt, ha a rendszer minden egysége kapott valamilyen (hibátlan/hibás) minősítést, *konzisztensnek* pedig akkor, ha minden egység valós és az algoritmus által feltárt hibaállapota megegyezik. A teljes és konzisztens diagnosztika eléréséhez teljesítendő további feltétel(eke)re példa a mérési segédletben tárgyalt, a hibás egységek számára adott felső korlát, az ún. t-korlát.
- A **valószínűségi algoritmusok** nem garantálnak biztosan teljes és konzisztens megoldást, de nagy valószínűséggel helyes eredményt szolgáltatnak. A tévedés lehetőségért cserébe ezek a módszerek csak a teszteredményekből kinyert információt használják fel, azaz nem igényelnek a rendszerre nézve olyan további előírásokat, mint a t-korlát a determinisztikus módszereknél.

  A valószínűségi algoritmusok lehetséges diagnosztikai tévedései három csoportba sorolhatók:

  1. Az *ismeretlen egység* egy olyan processzort jelent, amelynek hibaállapotát az algoritmus nem tudta meghatározni. Ebben az esetben a diagnosztika nem teljes.
  2. *Jóindulatú tévedés* esetén egy hibátlan processzort hibásnak minősítünk. Így csökkentjük ugyan a rendelkezésre álló erőforrások számát, de a biztonság javára tévedünk.
  3. *Rosszindulatú tévedés* esetén egy hibás processzort hibátlannak minősítünk. Ekkor a rendszerből nem távolítjuk el a hibás komponenst, ami hibás adatokkal és hibaterjesztéssel az egész folyamat hibáját okozhatja. Az erőforrások számát nem csökkentjük.

A mérési feladat

A mérési feladat a következő pontban leírt determinisztikus diagnosztikai algoritmus megvalósítása egy C nyelven megírt diagnosztikai szubrutin formájában. A szubrutint a szimulációs környezetbe ágyazva le kell fordítani, majd az algoritmus helyességét próbafuttatásokkal ellenőrizni. (A futtatások eredményét statisztikailag értékelve a mérési jegyzőkönyvben is fel kell tüntetni!)

A mérés kiértékelése

A determinisztikus algoritmusok az adott topológiára érvényes t-korlát értékével megegyező számú hibás egységig helyesen (teljes és/vagy konzisztens módon) *kell* működniük. A t-korlát felett azonban már nincs erre garancia. Ezért a determinisztikus algoritmusok ellenőrzésekor:

1. Több pontban elvégzett mérések alapján (adott rendszerméret esetén növekvő hibaszám, vagy adott számú hiba mellett növekvő rendszerméret, stb.) igazolnia kell, hogy a t-korlátig az algoritmus nem téved!

2. Ha ez teljesült, adjon statisztikát (táblázatos és diagram alakban) a t-korlát feletti hibaszámokra a jóindulatú és rosszindulatú diagnosztikai tévedések (és ha van, az ismeretlen egységek) alakulásáról.

Szorgalmi feladat: ha tud, adjon ötleteket az adott algoritmus továbbfejlesztéséhez (ha van ideje, valósítsa is meg)! Cél, hogy a diagnosztikai hatékonyság növekedjen (ilyen lehetőség pl. a felfedezett ellentmondások alapján meghozható biztos következtetések felhasználása).

A megvalósítandó algoritmus leírása

A mérési feladat a Meyer és Masson-féle diagnosztika algoritmus megvalósítása, az eredetihez képest egy kis módosítással.

A Meyer és Masson algoritmus leírása a mérési segédletben részletesen olvasható. Röviden összefoglalva: az általuk adott eljárás egy $n \times n$ méretű táblázatot tölt ki soronként úgy, hogy a *hibátlannak feltételezett* processzorokon lépdel végig, miközben azok teszteredményeit írja be a táblázat megfelelő helyeire. Ezután a táblázatot oszloponként összegezve, a t-korláton alapuló döntéssel minősíti az egységeket.

A Meyer és Masson által eredetileg leírt módszer azonban csak a speciális $D_{1,t}$ tesztelési elrendezés esetén használható, hiszen a táblázat kitöltése során kihasználja ennek a speciális topológiának a sajátosságait. A feladat tehát az algoritmus megvalósítása az alábbi kiegészítésekkel:

1. Módosítsa az algoritmust úgy, hogy az *tetszőleges tesztelési elrendezésre* működjön! Ez a gyakorlatban annyit jelent, hogy a táblázat kitöltése során nem (feltétlenül) a szomszédos cellákat járjuk be, hanem az adott topológia összeköttetési viszonyainak megfelelően egymástól távol álló cellákhoz tartozó processzorok is lehetnek az adott egység szomszédai. Így egy olyan *bejárási stratégiát* kell implementálni, ami biztosítja, hogy minden cella a hibátlannak feltételezett processzorokon végiglépkedve kitöltésre kerüljön bármilyen topológiai viszonyok közt a Meyer és Masson algoritmusnak megfelelően.

2. További fejlesztési lehetőség a *visszafelé irányú következtetések* felhasználása. Az eredeti Meyer és Masson algoritmus csak előre következtet: ha egy adott processzor hibátlan, akkor a teszteredményei minősítik a tesztelt processzorokat. Egyes esetekben azonban vissza is következtethetünk: ha egy processzorról feltétezzük, hogy hibátlan, és egy tesztelője *hibás eredménnyel* tesztelte, akkor arra következtethetünk, hogy az *adott tesztelő hibás*. Egészítse ki az algoritmust a visszafelé következtetés mechanizmusával!

3. Harmadik lehetséges kiegészítés a *biztos következtetések* felderítése. Ha a sor kitöltése során két külön következtetési láncon egy adott egységre két különböző feltételezés adódik, akkor *ellentmondásra* jutottunk. Ez azt jelenti, hogy a kiinduló feltételezésünk (hogy az adott sorhoz tartozó processzor hibátlan) nem állja meg a helyét, azaz ez a processzor *biztosan hibás*. Építse be a biztos következtetések kezelését az algoritmusba!

A mérés során a továbbfejlesztett algoritmus ellenőrzését és kiértékelését a *kétdimenziós tórusz* topológia esetére végezze el részletesen!