Általános információk

A diplomaterv szerkezete:

1. Diplomaterv feladatkiírás
2. Címoldal
3. Tartalomjegyzék
4. A diplomatervező nyilatkozata az önálló munkáról és az elektronikus adatok kezeléséről
5. Tartalmi összefoglaló magyarul és angolul
6. Bevezetés: a feladat értelmezése, a tervezés célja, a feladat indokoltsága, a diplomaterv felépítésének rövid összefoglalása
7. A feladatkiírás pontosítása és részletes elemzése
8. Előzmények (irodalomkutatás, hasonló alkotások), az ezekből levonható következtetések
9. A tervezés részletes leírása, a döntési lehetőségek értékelése és a választott megoldások indoklása
10. A megtervezett műszaki alkotás értékelése, kritikai elemzése, továbbfejlesztési lehetőségek
11. Esetleges köszönetnyilvánítások
12. Részletesés pontos irodalomjegyzék
13. Függelék(ek)

Felhasználható a következő oldaltól kezdődő Diplomaterv sablon dokumentum tartalma. Ügyeljen a tanszék, a hallgató, a konzulens nevét és a beadás évét jelölő szövegdobozokra, mert azokra külön ki kell adni a frissítést. A mezők tartalma a sablonban a dokumentum adatlapja alapján automatikusan kerül kitöltésre (Fájl/Információ/Tulajdonságok/Speciális tulajdonságok).

A diplomaterv szabványos méretű A4-es lapokra kerüljön. Az oldalak tükörmargóval készüljenek (mindenhol 2.5cm, baloldalon 1cm-es kötéssel). Az alapértelmezett betűkészlet a 12 pontos Times New Roman, másfeles sorközzel.

Minden oldalon - az első négy szerkezeti elem kivételével - szerepelnie kell az oldalszámnak.

A fejezeteket decimális beosztással kell ellátni. Az ábrákat a megfelelő helyre be kell illeszteni, fejezetenként decimális számmal és kifejező címmel kell ellátni. A fejezeteket decimális aláosztással számozzuk, maximálisan 3 aláosztás mélységben (pl. 2.3.4.1.). Az ábrákat, táblázatokat és képleteket célszerű fejezetenként külön számozni (pl. 2.4. ábra, 4.2 táblázat vagy képletnél (3.2)). A fejezetcímeket igazítsuk balra, a normál szövegnél viszont használjunk sorkiegyenlítést. Az ábrákat, táblázatokat és a hozzájuk tartozó címet igazítsuk középre. A cím a jelölt rész alatt helyezkedjen el.

A képeket lehetőleg rajzoló programmal készítsék el, az egyenleteket egyenlet-szerkesztő segítségével írják le.

Az irodalomjegyzék szövegközi hivatkozása történhet a Harvard-rendszerben (a szerző és az évszám megadásával) vagy sorszámozva. A teljes lista névsor szerinti sorrendben a szöveg végén szerepeljen (sorszámozott irodalmi hivatkozások esetén hivatkozási sorrendben). A szakirodalmi források címeit azonban mindig az eredeti nyelven kell megadni, esetleg zárójelben a fordítással. A listában szereplő valamennyi publikációra hivatkozni kell a szövegben. Minden publikáció a szerzők után a következő adatok szerepelnek: folyóirat cikkeknél a pontos cím, a folyóirat címe, évfolyam, szám, oldalszám tól-ig. A folyóirat címeket csak akkor rövidítsük, ha azok nagyon közismertek vagy nagyon hosszúak. Internet hivatkozások megadásakor fontos, hogy az elérési út előtt megadjuk az oldal tulajdonosát és tartalmát (mivel a link egy idő után akár elérhetetlenné is válhat), valamint az elérés időpontját.

Fontos:

* a szakdolgozat készítő/diplomatervező nyilatkozata (a jelen sablonban szereplő szövegtartalommal) kötelező előírás Karunkon, ennek hiányában a szakdolgozat/diplomaterv nem bírálható és nem védhető!
* mind a dolgozat, mind a melléklet maximálisan 15 MB méretű lehet!

Jó munkát, sikeres szakdolgozat készítést ill. diplomatervezést kívánunk!

FELADATKIÍRÁS

A feladatkiírást a **tanszék saját előírása szerint** vagy a tanszéki adminisztrációban lehet átvenni, és a tanszéki pecséttel ellátott, a tanszékvezető által aláírt lapot kell belefűzni a leadott munkába, vagy a tanszékvezető által elektronikusan jóváhagyott feladatkiírást kell a Diplomaterv Portálról letölteni és a leadott munkába belefűzni (ezen oldal HELYETT, ez az oldal csak útmutatás). Az elektronikusan feltöltött dolgozatban már nem kell megismételni a feladatkiírást.



Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Jost Márk Benedek

Útvonaltervezési algoritmusok optimalizálása gpu-n

Szakdolgozat

Konzulens

Dr. Szegletes Luca

BUDAPEST, 2023

Tartalomjegyzék

[Összefoglaló 6](#_Toc147509803)

[Abstract 7](#_Toc147509804)

[1 Bevezetés 8](#_Toc147509805)

[1.1 Motiváció 8](#_Toc147509806)

[2 Példák 10](#_Toc147509807)

[2.1 Formázási tudnivalók 10](#_Toc147509808)

[2.1.1 Címsorok 10](#_Toc147509809)

[2.1.2 Képek 10](#_Toc147509810)

[2.1.3 Kódrészletek 10](#_Toc147509811)

[2.1.4 Irodalomjegyzék 11](#_Toc147509812)

[3 Utolsó simítások 12](#_Toc147509813)

[Irodalomjegyzék 13](#_Toc147509814)

[Függelék 14](#_Toc147509815)

Hallgatói nyilatkozat

Alulírott **Jost Márk Benedek**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2023. 10. 11.

...…………………………………………….

Jost Márk Benedek

Összefoglaló

Szakdolgozatom során az volt a célom, hogy különböző, nagy jelentőséggel bíró algoritmusok futását gyorsítsam videokártyán párhuzamosítás segítségével. Vizsgálódásom során exponenciális futásidejű problémákra (TSP, VRP, CVRP) kerestem közelítő heurisztikus megoldást GPU-n. A megoldást az Nvidia CUDA keretrendszer segítségével C/C++ nyelven valósítottam meg. Az implementációkat ismert tesztadatokon teszteltem annak érdekében, hogy meggyőződjek, hogy valóban helyesen dolgoztam.

A különböző útvonaltervezési algoritmusok gyakran nem oldhatóak meg polinomidőben.

Abstract

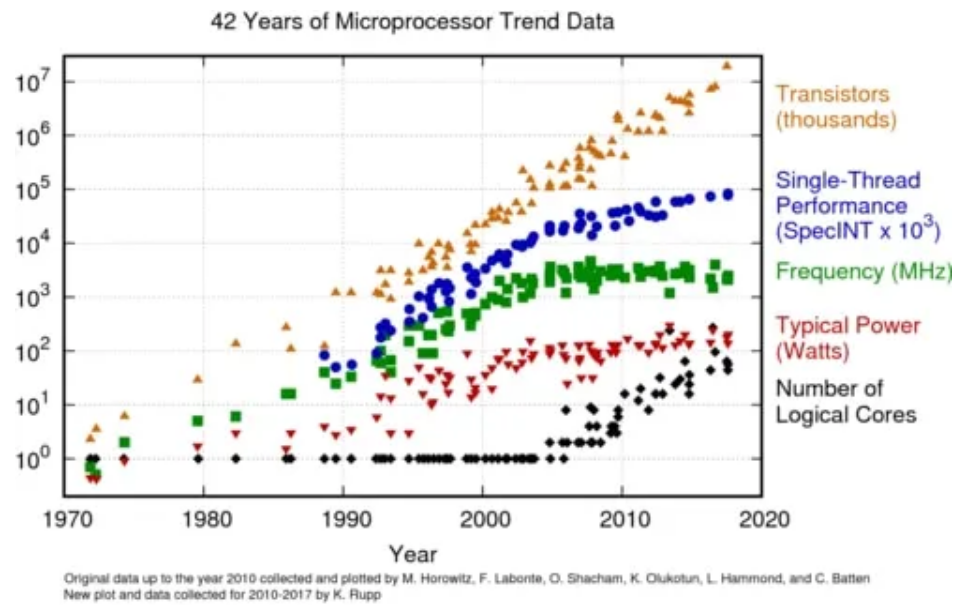
Ide jön a ½-1 oldalas angol nyelvű összefoglaló, amelynek szövege a Diplomaterv Portálra külön is feltöltésre kerül.

# Bevezetés

## Motiváció

Az 1980-as években megjelentek az első személyi számítógépek (PC-k), melyek központi feldolgozóegységei (CPU) kezdetekben néhány MHz-es belső órajellel működtek. Akkor az volt a számítástechnikai fejlesztőmérnökök fő eszköze a számítási gyorsaság növelésére, hogy az órajelfrekvenciát növelték. Ez értelemszerűen frekventáltabb utasításvégrehajtást biztosított, és evidens volt, hogy a nagyobb frekvencia nagyobb számítási erővel jár. Számos kiváló mérnöki megoldás született, ezek közül talán az egyik legjelentősebb találmány a fáziszárt hurok (Phase-Locked Loop - PLL). A PLL egy olyan Szabályozható hurok, amely (a részleteket mellőzve, nem tárgya dolgozatomnak) egy bemeneti referenciafrekvenciát tud megsokszorozni. Nélküle gyakorlatilag képtelenség lett volna felhasználói szinten 50-60 MHz fölé menni a személyi számítógépek belső órajelénél. Nagyjából 30 évvel később elérték a hardverfejlesztők, hogy a legtöbb asztali processzor órajele 1GHz és 4GHz között legyen képes működni, ez az eredeti PC-k frekvenciájának több, mint az ezerszerese. Napjainkban változás látható a fejlesztési trendekben, ugyanis az órajelnövelést a processzorok disszipációja erősen felülről korlátozza. Egyelőre nem tűnik könnyen lehetségesnek 5GHz fölé menni úgy, hogy közben az eszköz helyes működése garantálható legyen. A különböző hűtési technológiák (léghűtés, vízhűtés) bizonyos fokig tudnak javítani a sebességen, viszont nagyságrendeket ugorni velük sem lehetséges.

A számítógépgyártók éppen ezért új, alternatív megoldásokat kerestek a számítási teljesítmény növelésére. Legjobb ötletnek a számítási feladatok párhuzamosítása bizonyult. Napjainkban a kutatásoknak két nagy témája van. Egyik a kvantumszámítószépek témája, amit dolgozatomban nem részletezek. A másik a párhuzamosítás több szálon. Már a CPU-k fejlesztésénél is megfigyelhető, hogy inkább a minél több processzormag telepítése az iparági trend.



1.1. ábra Látható, hogy kb. 2010-re befejeződött a CPU-k órajelfrekvencia-növekedése, helyette egyre nőni kezdett a processzormagok száma. [1]

A párhuzamos munkavégzésre feltalálták a grafikus segédprocesszort, a GPU-t. Amíg a CPU feladata az, hogy műveletek egy adott szekvenciáját, és annak minden utasítását a lehető leggyorsabban hajtsa végre, addig a GPU célja minél több szál (akár több tízezer) párhuzamos futtatása. A videókártyák előnye akkor válik láthatóvá, ha ugyanazt az utasítást több, nagy adattömbön kell végrehajtani. Ez az úgynevezett SIMD megközelítés (Single Instruction Multiple Data) [3]. Az [HIÁNYZIK]. ábra szemlélteti, hogy a GPU-n arányában több tranzisztor van adatfeldolgozásra rendelvelve, cserébe a gyorsítótárazás és a folyamatvezérlés kisebb hangsúlyt kapott.

# Elméleti háttér

## Útvonaltervezési problémák

### Utazóügynök probléma (TSP)

### Jármű útvonaltervezési probléma (VRP)

### Korlátozott kapacitású járművek útvonaltervezése (CVRP)

### Korlátozott kapacitású, adott időablakra teljesítő járművek útvonaltervezési problémája (CVRPTW)

## Hangyakolónia algoritmus

### Alternatív megoldások

Akármelyik problémát tekintjük, legyen az TSP, VRP, CVRP vagy CVRPTW, megoldható egzakt, vagy közelítőleg is.

### „Brute force” algoritmus

A „Brute force” algoritmus lényege, hogy minden lehetséges bejárást megvizsgálunk, és kiválasztjuk a legrövidebb, a konkrét probléma feltétel(rendszer)ének eleget tevő esetet. **Ha n db csúcs**ból álló teljes gráfot nézünk (mint ahogy ez a valóságban igen gyakori), **akkor n! különböző lehetséges bejárás**t kell összehasonlítani. Kis n esetén még csak-csak elfogadható ez a módszer, viszont ha már csak pl. n=48 db csúcsunk van (mert esetleg szeretnénk bejárni TSP szerint az Amerikai Egyesült Államok 48 összefüggő államának fővárosait, később lesz rá példa), a vizsgálandó esetek száma felugrik 48! ≈ 1.24 \* 1061 -re. Tegyük fel, hogy csúcskategóriás, 5GHz-en pörgő szuperszámítógépünk képes átlagosan 1 órajelciklusonként (2\*10-10s időközönként, nagyjából lehetetlenül gyorsan) kiszámolni egy út hosszát, még így is kb. 2.5 \* 1051 s ≈ 8 \* 1043 évig vizsgálhatnánk az eseteket. Egy SMD beültetőgép egy szerelőlemezre akár alkatrészek százait pakolhatja fel, ilyen módszerrel lehetetlen lenne megmondani, hogy milyen sorrendben haladjon. Látható tehát, hogy ha véges időn belül meg akarjuk oldani a problémát, ravaszabbnak kell lennünk.

### Held-Karp algoritmus

A Held-Karp algoritmus célja egyesével felépíteni a bejárást. Az algoritmust szeretném röviden ismertetni.

Jelöljük a csúcsokat -nel. Legyen S halmaz a csúcsok valamely részhalmaza. Legyen g(S,e) -ből az S összes elemén keresztül az e csúcsba végződő legrövidebb út hossza.

kiválaszt egy kezdővárost (1-es számú város), amit betesz egy S halmazba. Ezt követően egyesével adja hozzá a csúcsokat a halmazhoz.

# Példák

A következő fejezet pár példán keresztül bemutatja a diplomatervekben és szakdolgozatokban szokásosan előkerülő formázások megvalósítását.

## Formázási tudnivalók

A dokumentum folyószövegéhez használjuk a **Normál** (angol Word esetén Normal) stílust.

### Címsorok

A fejezetcímek esetén a **Címsor 1-4** (Heading 1-4) stílusokat használjuk.

### Képek

A képhez használjuk a **Kép** stílust.

Képaláírást a képen jobb gombbal kattintva a Felirat beszúrása… opcióval adhatunk hozzá, így az automatikusan **Képaláírás** (Caption) stílusú lesz.



1.1. ábra: Példa képaláírásra

### Kódrészletek

Kódrészletek beillesztése esetén használjuk a **Kód** stílust.

using System;

namespace MyApp

{

class Program

{

static void Main( string[] args )

{

Console.WriteLine( "Szia Világ!" );

}

}

}

### Irodalomjegyzék

Az Irodalomjegyzékben szereplő hivatkozásokat **Irodalomjegyzék sor** stílussal formázzuk, a címüket pedig **Irodalomjegyzék forrás** stílussal emeljük ki.

A szövegbe a hivatkozásokat a Kereszthivatkozás beszúrása (Insert cross-reference) funkcióval helyezzük el (példa egy így beszúrt hivatkozásra: [1]), így azok automatikusan frissülnek a hivatkozások átrendezésekor.

# Utolsó simítások

Miután elkészültünk a dokumentációval, ne felejtsük el a következő lépéseket:

* Kereszthivatkozások frissítése: miután kijelöltük a teljes szöveget (Ctrl+A), nyomjuk meg az F9 billentyűt, és a Word frissíti az összes kereszthivatkozást. Ilyenkor ellenőrizzük, hogy nem jelent-e meg valahol a "Hiba! A könyvjelző nem létezik." szöveg.
* Dokumentum tulajdonságok megadása: a dokumentumhoz tartozó meta adatok kitöltése (szerző, cím, kulcsszavak stb.). Erre való a Dokumentum tulajdonságai panel, mely a Fájl / Információ / Tulajdonságok / Dokumentumpanel megjelenítése úton érhető el.
* Kinézet ellenőrzése PDF-ben: a legjobb teszt a végén, ha PDF-et készítünk a dokumentumból, és azt leellenőrizzük.

Irodalomjegyzék

1. 55th Anniversary of Moore’s Law.   
   https://www.infoq.com/news/2020/04/Moores- law- 55/  
   Accessed: 2020-04-27.
2. Tóth Márk Andor. “Algoritmusok Optimalizálása GPU-n”. diplomaterv. Budapesti Műszaki Egyetem, 2022.
3. Király Zoltán. “Kvantummechanikai számítások optimalizálása”. diplomaterv. Budapesti Műszaki Egyetem, 2022.
4. Held, M., & Karp, R. M. (1962). "A Dynamic Programming Approach to Sequencing Problems." Journal of the Society for Industrial and Applied Mathematics, 10(1), 196-210.

Függelék