UNIVERSITEIT GENT

Parallelle Algoritmen
Het Handelsreizigersprobleem

Project Algoritmen en Datastructuren III

Academiejaar 2013-2014

1 Inleiding

Het handelsreizigersprobleem is een goed gekend probleem in de informatica dat als volgt kan beschreven worden:

> Gegeven een lijst van steden en de afstanden tussen elk van deze steden; wat is de kortst mogelijke route die men kan volgen om **alle** steden **exact 1** keer te bezoeken en **terug bij het startpunt** uit te komen?

Dit probleem heeft niet alleen toepassingen in de praktijk (bv. planning en logistiek) maar wordt bijvoorbeeld ook gebruikt om optimalisatietechnieken en heuristieken te testen.

2 Opgave

2.1 Algoritme

Voor dit project is het de bedoeling om een programma te schrijven dat het handelsreizigersprobleem oplost met behulp van de gedistribueerde versie van de *branch and bound* techniek. Deze techniek wordt beschreven in de cursus in hoofdstuk 6.1. Voor de communicatie tussen de verschillende processen zullen we gebruik maken van MPI. Een inleiding tot MPI kan je vinden in hoofdstuk 6.2 van de cursus.

Naast de *branch and bound* aanpak dienen ook 2 heuristieken naar keuze geïmplementeerd te worden die elk een benadering van de oplossing zoeken. Deze heuristieken moeten parallel met het *branch and bound* algoritme uitgevoerd worden. De waarden die gevonden worden door de heuristieken kunnen bijgevolg gebruikt worden in het *bounding* gedeelte van het *branch and bound* algoritme. De bedoeling is om zo een efficiënter algoritme te bekomen.

Je zal zelf moeten beslissen hoeveel parallelle processen je inzet voor heuristieken en hoeveel voor *branch and bound*.

2.2 Verslag en testen

In het verslag verwachten we dat je de belangrijkste gemaakte implementatiekeuzes verduidelijkt en uitlegt. Bespreek ook de heuristieken die je gebruikt hebt en waarom je ze gekozen hebt. Test de invloed van het aantal processen waarmee je je programma start en de invloed van de gekozen splitsdiepte op de uitvoeringstijd. Test ook of het gebruik van heuristieken nuttig is in combinatie met *branch and bound* en hoe dicht hun oplossing bij de optimale oplossing ligt. Bespreek welke testen je gedaan hebt, welke testbestanden je gebruikt hebt en, indien je zelf testbestanden geconstrueerd hebt, waarom die belangrijk zijn.

3 Specificaties

3.1 Programmeertaal

In de opleidingscommissie informatica (OCI) werd beslist dat, om meer ervaring in het programmeren in C te verwerven, het project horende bij het opleidingsonderdeel Algoritmen en Datastructuren III in C geïmplementeerd dient te worden. Het is met andere woorden de bedoeling je implementatie in C uit te voeren. Je implementatie dient te voldoen aan de ANSI-standaard. Je mag hiervoor gebruikmaken van de laatste features in C99, voorzover die ondersteund worden door `gcc` op `helios`.

Voor het project kan je de standaardbibliotheken gebruiken; externe libraries zijn echter niet toegelaten. Het spreekt voor zich dat je normale, procedurale C-code schrijft en geen platformspecifieke APIs (zoals bv. de Win32 API) of features uit C++ gebruikt. Op Windows bestaat van een aantal functies zoals `qsort` een "safe" versie (in dit geval `qsort_s`), maar om je programma te kunnen compileren op een unix-systeem kan je die versie dus niet gebruiken. Er wordt natuurlijk een uitzondering gemaakt voor de MPI libraries, deze zijn reeds geïnstalleerd op `helios`.

Wat je ontwikkelingsplatform ook mag zijn, test zeker in het begin altijd eens of je op Helios wel kan compileren, om bij het indienen onaangename verrassingen te vermijden!

3.2 Helios

`Helios.UGent.be` is de interactieve linux omgeving van de Universiteit Gent die de nieuwste stabiele Debian distributie draait. Om toegang te krijgen tot helios moet een account aangevraagd worden. Dit kan, moest dit nog niet gebeurd zijn, via het formulier op `https://gaia.ugent.be/helios/account.cgi`. Merk op dat je voor deze pagina (en voor toegang tot `helios` zelf) op het UGent netwerk moet zitten. Indien je van

thuis toegang wil krijgen moet je eerst een VPN-verbinding opzetten. Meer informatie hierover kan je vinden op `http://helpdesk.ugent.be/vpn/`.

3.3 Input/Output en implementatiedetails

3.3.1 MPI

We raden aan om Open MPI te gebruiken. Deze libraries zijn op Helios geïnstalleerd en compileren kan daar met volgend commando:

```
$ mpicc -std=c99 main.c -o tsp
```

Het programma uitvoeren met 5 processen kan vervolgens met bijvoorbeeld:

```
$ mpirun -n 5 tsp "invoer.txt"
```

3.3.2 Invoer

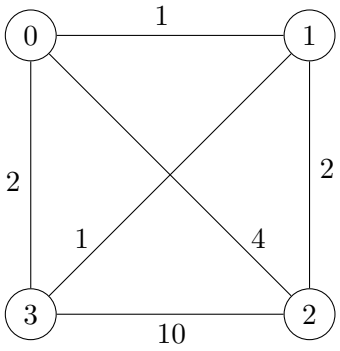
Als invoer gebruiken we gewone tekstbestanden. Op de eerste regel van een invoerbestand staat het aantal steden n waaruit het probleem bestaat. Hierna volgen n regels waarop telkens n integers staan, gescheiden door een spatie. Deze getallen stellen een $n \times n$ afstandsmatrix voor en bevat dus de afstanden tussen alle n punten. Op regel $i + 1$ kolom j staat dus de afstand tussen stad i en stad j. Je mag veronderstellen dat de afstand tussen stad A en B steeds gelijk is aan de afstand tussen stad B en A.

Het pad naar het invoerbestand moet als eerste en **enige** parameter aan je programma meegegeven worden.

3.3.3 Uitvoer

Je programma dient de gevonden oplossing uit te schrijven naar standaard uitvoer. Op de eerste regel schrijf je de totale afstand van de gevonden oplossing uit. De tweede regel bestaat uit een reeks van integers gescheiden door een spatie. Deze getallen stellen de volgorde van (de volgnummers van) de steden van de gevonden oplossing voor. Deze reeks begint en eindigt steeds met stad 0. Als er meerdere oplossingen mogelijk zijn, mag er slechts 1 oplossing uitgeschreven worden.

3.3.4 Voorbeeld



In bovenstaande figuur vind je een voorbeeld met 4 steden voorgesteld als graaf. De getallen in de toppen zijn de volgnummers van de steden en de afstanden tussen deze steden zijn telkens weergegeven bij de corresponderende bogen. Onderstaande sessie zoekt een oplossing voor dit voorbeeld:

```
$ cat invoer.txt
4
0 1 4 2
1 0 2 1
4 2 0 10
2 1 10 0

$ mpirun -n 5 tsp "invoer.txt"
9
0 2 1 3 0
```

4 Indienen

4.1 Directorystructuur

Je dient één zipfile in via `http://indianio.ugent.be` met de volgende inhoud:

- `src/` bevat alle broncode
- `tests/` alle testcode.
- `extra/verslag.pdf` bevat de elektronische versie van je verslag. In deze map kan je ook eventueel extra bijlagen plaatsen.

4.2 Compileren

De code zal door ons gecompileerd worden op `helios` met behulp van de opdracht `mpicc -std=c99 -lm`. Test zeker dat je code compileert en werkt op helios voor het indienen.

De ingediende versie dient een bestand met de naam `sources` te bevatten waar je de dependencies voor het compileren kan aangeven. Dit bestand bevat slechts 1 regel waarop na `tsp:` alle nodige `*.c`-betanden voor het compileren opgelijst worden. Een voorbeeld hiervan is:

```
tsp: main.c hulpfuncties.c
```

4.3 Belangrijke data

Tegen *zondag 3 november* dien je een tussentijdse versie in te dienen via indianio. Voor deze versie verwachten we een werkend *Branch and Bound* algoritme. Heuristieken en verslag zijn dus voor deze versie nog niet nodig.

De uiteindelijke deadline is *donderdag 28 november* om 17u. Naast een zip-bestand op indianio verwachten we ook een papieren versie van je verslag. Dit verslag kan ingediend worden in lokaal 110.016 op S9 (eerste verdieping, halfweg de gang, bureau Bart Mesuere) of tijdens de oefeningenles.

4.4 Algemene richtlijnen

- Schrijf efficiënte code maar ga niet overoptimaliseren: **geef de voorkeur aan elegante, goed leesbare code**. Kies zinvolle namen voor methoden en variabelen en voorzie voldoende commentaar.
- Het project wordt gequoteerd op **4** van de 20 te behalen punten voor dit vak, en deze punten worden ongewijzigd overgenomen naar de tweede examenperiode.
- Het is strikt noodzakelijk twee keer in te dienen: het niet indienen van de eerste tussentijdse versie betekent sowieso het verlies van alle punten.
- Projecten die ons niet bereiken voor de deadline worden niet meer verbeterd: dit betekent het verlies van alle te behalen punten voor het project.
- Het eerste deel is niet finaal. Je mag gerust na de eerste indiendatum nog veranderingen aanbrengen. We verbeteren in de eerste fase enkel op aanwezigheid van alle gevraagde features en het correct werken ervan.

- Dit is een individueel project en dient dus door jou persoonlijk gemaakt te worden. Het is uiteraard toegestaan om andere studenten te helpen of om ideeën uit te wisselen, maar **het is ten strengste verboden code uit te wisselen**, op welke manier dan ook. Het overnemen van code beschouwen we als fraude (van **beide** betrokken partijen) en zal in overeenstemming met het examenreglement behandeld worden. Op het internet zullen ongetwijfeld ook (delen van) implementaties te vinden zijn. Het overnemen of aanpassen van dergelijke code is echter **niet toegelaten** en wordt gezien als fraude.

- Essentiële vragen worden **niet** meer beantwoord tijdens de laatste week voor de deadline.