UNIVERSITÉ LIBRE DE BRUXELLES

Applying Parallel Computation Algorithms in the Design of Serial Algorithms

INFO-F-420: Computational Geometry

Quinten Rosseel

Vrije Universiteit Brussel
Student No. 0536798

Presentation Outline

Presentation Outline

Project Goals

- Make parametric search more comprehensible with a bottom-up (examples) approach rather than a top-down (formal) approach.
- Get intuition for how and when to use parametric search.
- Provide visualisations of parameterization.
- Do something with graphs.

Presentation Outline

Parametric Search

Informal Explanation

Use *decision problem* algorithm to solve *optimization problem*.

- Decision problem: Check if a condition holds or not.
  e.g. For input value λ, is $\lambda < \lambda^*, \lambda = \lambda^*$ or $\lambda > \lambda^*$?
- Optimization problem: Find optimal solution for problem.
  e.g. Minimize $f(\lambda)$ when λ has a set of constraints.

Presentation Outline

Components

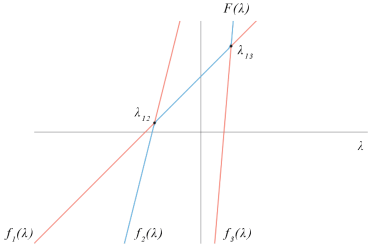
Definitions and Notation

- Let $f_i(\lambda) = a_i + b_i\lambda$ with $b_i > 0$.
- Let $\{f_1(\lambda), \ldots, f_n(\lambda)\}$ be a set of pairwise distinct functions.
- Let $F(\lambda)$ be the median of set $\{f_1(\lambda), \ldots, f_n(\lambda)\}$ for all $\lambda \in \mathbb{R}$.
- Let λ_{ij} denote the intersection two distinct functions f_i and f_j in the set $\{f_1(\lambda), \ldots, f_n(\lambda)\}$, such that $a_i + b_i\lambda_{ij} = a_j + b_j\lambda_{ij}$ with $i \neq j$.

Components

Notes on $F(\lambda)$

- Monotone increasing segments, because $b_i > 0$ in each $f_i(\lambda) = a_i + b_i\lambda$ of set $\{f_1(\lambda), \ldots, f_n(\lambda)\}$.
- $O(n^2)$ breakpoints, because the maximum of intersections for n straight lines is $\frac{n^2-n}{2}$ intersections
- Evaluable in linear-time when all $f_i(\lambda)$'s have been computed.



Presentation Outline

Solving $F(\lambda) = 0$

A Non-Parametric Algorithm Outline

```
1. let intersections ← All λ_ij ∈ {f_1, ..., f_n};
2. sort(intersections);
3. let interval ← binary_search(
     function pivot_comparison(λ_ij) {
       return λ_ij < 0;
     }
   );
4. linear_root(interval);
```

Algorithm Asymptote is Quadratic

- Find intersections: $O(n^2)$
- Sorting: $O(n \log(n))$
- Binary Search: $O(\log(n))$

Solving $F(\lambda) = 0$

Employing Parametric Search

- **Idea:** use decision procedure on critical points λ_{ij}, *i.e.* $\lambda_{ij} > 0, \lambda_{ij} = 0$ or $\lambda_{ij} < 0$ to find the smallest open interval $(\lambda_{min}, \lambda_{max})$ where $F(\lambda) = 0$.
- λ^* denotes the unkown solution to $F(\lambda^*) = 0$.

A Parametric Algorithm Outline

1. let $\lambda_{min} \leftarrow -\infty$ and $\lambda_{max} \leftarrow \infty$;
2. let intersections $\leftarrow$ All $\lambda_{ij} \in \{f_1, \ldots, f_n\}$;
3. for λ_{ij} in intersections:

    if($\lambda_{ij} \in (\lambda_{min}, \lambda_{max})$ AND $F(\lambda_{ij}) < 0$)) {
      $\lambda_{min} \leftarrow \lambda_{ij}$;
    } else if($\lambda_{ij} \in (\lambda_{min}, \lambda_{max})$) {
      $\lambda_{max} \leftarrow \lambda_{ij}$;
    }

4. linear_root((λ_{min}, λ_{max}));

Solving $F(\lambda) = 0$

A Parametric Algorithm Outline

```
1. let λ_min ← −∞ and λ_max ← ∞;
2. let intersections ← All λ_ij ∈ {f_1, ..., f_n};
3. for λ_ij in intersections:
      if(λ_ij ∈ (λ_min, λ_max) AND F(λ_ij) < 0)) {
        λ_min = λ_ij;
      } else if(λ_ij ∈ (λ_min, λ_max)) {
        λ_max = λ_ij;
      }
4. linear_root((λ_min, λ_max));
```

Algorithm Assymptote is Quadratic

- Finding intersections: $O(n^2)$
- Only $O(n)$ of $O(n^2)$ intersections require an $O(n)$ evaluation of F: $O(n^2)$

Solving $F(\lambda) = 0$

From Parametric to Parallelism

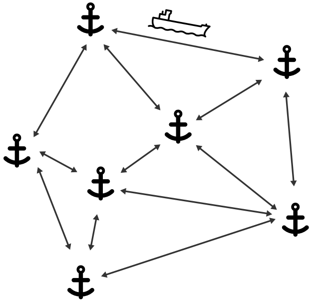
- λ_{min} and λ_{max} are shared variables of the concurrent processes.
- Each intersection λ_{ij} can be evaluated by F independently, on a different thread.
- Yield slightly better theoretical bounds: $O(nlog(n)^2)$ and $O(nlog(n)^2 log(log(n)))$.

Presentation Outline

The problem

Informal Explanation

- Coined by Dantzig et al. in the context of a ship routing problem posed by the American Office of Naval Research.
- Concerns a ship owner that wants to **maximize his mean daily profit** over time while making a round trip through multiple ports.
- Can be transformed to a **minimization problem** by looking at the minimal cost.
- The solution to MRC gives exactly the path that the skipper should take to maximize his profit.



The MRC problem

More Formally

- A directed graph $\mathcal{G} = (V, E)$, with V a set of vertices and E a set of edges.
- No self-loops in $\mathcal{G}$
- Each vertex i has an associated cost c_{ij} and travel time t_{ij} to reach a vertex j.

Representation

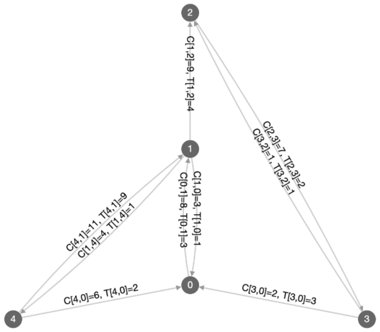
- Adjacency matrix $A \in \{0, 1\}^{|V| \times |V|} \rightarrow$ Directed graph $\mathcal{G}$
- A cost matrix $C \in \mathbb{R}^{|V| \times |V|} \rightarrow$ All travel costs c_{ij}
- A time matrix $T \in \mathbb{R}^{|V| \times |V|} \rightarrow$ All travel times t_{ij}

The MRC problem

Representation Example

- Interactive graph visualisations made using the Cytoscape Library.
- Should be formulated as an optimization problem (minimization).

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix} \quad C = \begin{bmatrix} 0 & 8 & 0 & 0 & 0 \\ 3 & 0 & 9 & 0 & 4 \\ 0 & 0 & 0 & 7 & 0 \\ 2 & 0 & 1 & 0 & 0 \\ 6 & 11 & 0 & 0 & 0 \end{bmatrix} \quad T = \begin{bmatrix} 0 & 3 & 0 & 0 & 0 \\ 1 & 0 & 4 & 0 & 1 \\ 0 & 0 & 0 & 2 & 0 \\ 3 & 0 & 1 & 0 & 0 \\ 2 & 9 & 0 & 0 & 0 \end{bmatrix}$$



Presentation Outline

Megiddo's Theorem

Problem A. Minimize $c_1x_1 + \cdots + c_nx_n$ subject to $x = (x_1, \ldots, x_n) \in D$

Problem B. Minimize $\dfrac{a_0 + a_1x_1 + \cdots + a_nx_n}{b_0 + b_1x_1 + \cdots + b_nx_n}$ subject to $x = (x_1, \ldots, x_n) \in D$

With D a set of conditions or constraints to which $x = (x_1, \ldots, x_n)$ must adhere in order to be a valid solution.

Theorem. If problem A is solvable within $O\Big(p(n)\Big)$ comparisons and $O\Big(q(n)\Big)$ additions, then B is solvable in time $O\Bigg(p(n)\Big(q(n) + p(n)\Big)\Bigg)$.

The General Problem

Idea of the theorem (1)

- Given a problem of type B :

$$\min\left(\frac{a_0}{b_0} + \frac{a_1}{b_1}x_1 + \cdots + \frac{a_n}{b_n}x_n\right) \text{ with } x = (x_1, \ldots, x_n) \in D$$

- Pick a fixed number $t \in \mathbb{R}$
- Solve problem A with parameters of problem B and t :

$$\min\left(c_1x_1 + \cdots + c_nx_n \mid c_i(t) = a_i - tb_i\right) \text{ with } x = (x_1, \ldots, x_n) \in D$$

The General Problem

Idea of the theorem (2)

- Suppose that v is the optimal value for problem A :

$$v = \min\Big((a_1 - tb_1)x_1 + \cdots + (a_n - tb_n)x_n\Big) \text{ with } x = (x_1, \ldots, x_n) \in D.$$

- If v can be written as $tb_0 - a_0$ then:
$tb_0 - a_0 = \min\Big((a_1 - tb_1)x_1 + \cdots + (a_n - tb_n)x_n\Big) \iff$
$t = \min\Big(\frac{a_0}{b_0} + (a_1 - tb_1)x_1 + \cdots + (a_n - tb_n)x_n\Big)$
- How does this compare to problem B?

$$\min\Big(\frac{a_0}{b_0} + \frac{a_1}{b_1}x_1 + \cdots + \frac{a_n}{b_n}x_n\Big) \text{ with } x = (x_1, \ldots, x_n) \in D$$

The General Problem

Idea of the theorem (3)

- Notice that $\forall i \in \{1 \dots n\}$: $\frac{a_i}{b_i}$ is the root of $c_i(t) = a_i - tb_i$ such that $c_i(\frac{a_i}{b_i}) = 0$.
- Megiddo's ratio-minimization trick: replace functions in problem with their roots.

$$tb_0 - a_0 = \min\left(c_1(t)x_1 + \dots + c_n(t)x_n\right) \iff$$
$$t = \frac{a_0}{b_0} + \min\left((a_1 - tb_1)x_1 + \dots + (a_n - tb_n)x_n\right) \iff$$
$$t = \min\left(\frac{a_0}{b_0} + \frac{a_1}{b_1}x_1 + \dots + \frac{a_n}{b_n}x_n\right)$$

- **We found a relation**: t is the optimal value for problem B when the optimal value v of problem A can be written as $tb_0 - a_0$.

The General Problem

Idea of the theorem (4)

- **Idea:** use the relation between problem A and problem B as a decision procedure for solving problem B, using algorithm A.
- If $v = tb_0 - a_0$: algorithm B can be solved by employing algorithm A.
- If $v < tb_0 - a_0$: test a smaller t value.
- If $v > tb_0 - a_0$: test a bigger t value.
- **Key question:** How many values of t have to be tested before $v = tb_0 - a_0$.
- Gradually find an interval such that $v \in [e, f]$
- Notice the similarity with the preliminary example.

Presentation Outline

Applying the General Problem to MRC

Analogies

- Problem A: Minimize $c_1x_1 + \cdots + c_nx_n$ subject to $x = (x_1, \ldots, x_n) \in D$
  $\rightarrow$ Shortest path between two nodes
- Problem B: Minimize $\frac{a_0}{b_0} + \frac{a_1}{b_1}x_1 + \cdots + \frac{a_n}{b_n}x_n$ subject to $x = (x_1, \ldots, x_n) \in D$
  $\rightarrow$ Minimum ratio cycle.
- Megiddo's theorem indicates that problem B can be solved using problem A.

Applying the General Problem to MRC

The Floyd Warshall Algorithm

- Below: get shortest distance from node i to node j.
- Uses dynamic programming to construct shortest path.
- Find shortest path between all nodes in $O(n^3)$.
- Can detect negative cycles.

```
1. let m ← Distance-encoded adjacency matrix
2. let n ← length(m);
3. for k ← 0 ... n:
     for i ← 0 ... n:
         for j ← 0 ... n:
             if(m[i][k] + m[k][j] < m[i][j]) {
               m[i][j] ← m[i][k] + m[k][j];
             }
```

Applying the General Problem to MRC

Notation

- $u_{ij}^{(m)}$: length of a shortest simple path from i to j, only using nodes from the set $\{1 \dots m-1\} \cup \{i, j\}$ and using a distance function $c_{ij}(t) = a_{ij} - tb_{ij}$
- This is a beefed-up A algorithm that will be run to demarcate the solution bound for problem B.
- Needs to be able to cope with negative cycles.

Applying the General Problem to MRC

Algorithm Outline (1)

1. let $n \leftarrow |V(\mathcal{G})|$
2. let $[e, f] \leftarrow [-\infty, \infty]$, let $i \leftarrow j \leftarrow m \leftarrow 0$ with $0 \leq i \leq j \leq n$
3. let t' $\leftarrow$ $solve\left(u_{ij}^{(m)}(t) = u_{im}^{(m)}(t) + u_{mj}^{(m)}(t)\right)_t$;
4. if (unique_solution(t')) {
        check_cycles(t');
   } else {
        update_parameters();
   }
5. $u_{ij}^{(m+1)}(t) \leftarrow \min\left(u_{ij}^{(m)}(t), u_{im}^{(m)}(t) + u_{mj}^{(m)}(t)\right)$;
6. update_parameters();
7. MRC $\leftarrow$ find k such that $u_{kk}^{n+1}(f) < 0$.

Applying the General Problem to MRC

Algorithm Outline (2)

8. `check_cycles(t')`:

   let $\mathcal{G} \leftarrow$ graph with distances $c_{kl}(t') = a_{kl} - t' b_{kl}$

   if$\Big($zero_cycle($\mathcal{G}$) AND !negative_cycle($\mathcal{G}$)$\Big)$ {

     let MRC $\leftarrow$ zero_cycle($\mathcal{G}$);

     return MRC;

   } else if $\Big($negative_cycle($\mathcal{G}$)$\Big)$ {

     $[e, f] \leftarrow [e, t']$;

   } else if $\Big($all_cycles_positive($\mathcal{G}$)$\Big)$ {

     $[e, f] \leftarrow [t', f]$;

   }

Applying the General Problem to MRC

Algorithm Outline (3)

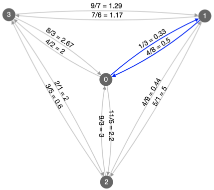
9. update_parameters(t'):

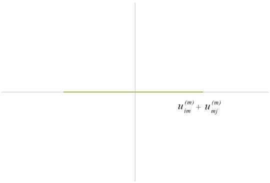
```
if (j < n) {
  j ← j + 1;
  go_to(1);
} else if (j = n AND i < n){
  i ← i + 1;
  go_to(1);
} else if (i = j = n AND m < n){
  i ← 1;
  j ← 1;
  m ← m + 1;
  go_to(1);
} else if(i = j = n AND m = n){
  go_to(5);
}
```

Applying the General Problem to MRC

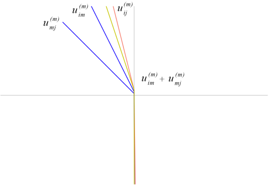
Decision Procedure Visualisation

- The result t' of $\mathit{solve}\left(u_{ij}^{(m)}(t) = u_{im}^{(m)}(t) + u_{mj}^{(m)}(t)\right)_t$ determines how the solution interval $[e, f]$ is updated.

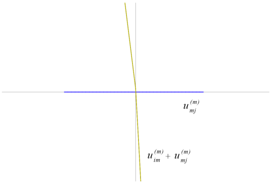




$i = j = m = 0$



$i = 0,\ j = 2,\ m = 3$



$i = 1,\ j = 0,\ m = 0$

Conclusions

- The basic principle of parametric search is not too difficult to grasp, but applying it in practice requires creativity and ingenuity.
- Visuals help in algorithm intuition and proof argumentation.
- Papers should reference / illustrate non-obvious steps in proofs better. *e.g.* Ratio-minimization trick.

References

- Megiddo, N. (1981, October). Applying parallel computation algorithms in the design of serial algorithms. In 22nd Annual Symposium on Foundations of Computer Science (sfcs 1981) (pp. 399-408). IEEE.
- Dantzig, G. B., Blattner, W. and Rao, M. R. (1967). Finding a Cycle in a Graph with Minimum Cost to Time Ratio with Application to a Ship Routing Problem. In Theory of Graphs, P. Rosenstiehl, ed. Dunod, Paris, and Gordon and Breach, New York. 77-84.
- Megiddo, N. (1978, May). Combinatorial optimization with rational objective functions. In Proceedings of the tenth annual ACM symposium on Theory of computing (pp. 1-12). ACM.