Parallel Computing
Parallels the ShortL-BiLS algorithm for the stable marriage problem

Quoc Dai Tran, Tuan Canh Bui

Ho Chi Minh City University of Technology, VNU-HCM

1770019,1770318@hcmut.edu.vn

November 15, 2019

Content

Introduction - the stable marriage problem (SMP)

General statement - posed by Gale and Shapley, 1962

Given an equal number of man and woman, the problem is to find an one-one correspondence matching such that:

- There are no two people of opposite sex who would both matched each other than their current partners.

Such a matching is said to be stable.

Introduction - ShortL-BiLS algorithm

M_{left} := GALESHAPLEY(I,Men); ▷ *men* propose *women*;
M_{right} := GALESHAPLEY(I,$Women$); ▷ *women* propose *men*;
forward := *true*; *backward* := *true*;
while *(true)* **do**
    if *(forward)* **then**
        neighborSet := $\emptyset$;
        for *(each man m in the men set)* **do**
            stableMatching := BREAKMARRIAGE(M_{left}, m);
            neighborSet := *neighborSet* $\cup$ *stableMatching*;
        if *(small random probability p)* **then**
            M_{next} := a random matching in *neighborSet*;
        else
            $M_{next} := \arg\min_{M \in neighborSet}(f(M))$;
        if $(f(M_{next}) > f(M_{left}))$ **then**
            forward := *false*;
            if $f(M_{best}) > f(M_{left})$ **then**
                $M_{best} := M_{left}$;
        $M_{left} := M_{next}$;

Introduction - ShortL-BiLS algorithm (cont)

```
while (continued) do
    if (backward) then
        neighborSet := ∅;
        for (each woman w in the women set) do
            stableMatching := BREAKMARRIAGE(M_right, w);
            neighborSet := neighborSet ∪ stableMatching;
        if (small random probability p) then
            M_next := a random matching in neighborSet;
        else
            M_next := arg min_{M ∈ neighborSet}(f(M));
        if (f(M_next) > f(M_right)) then
            backward := false;
            if f(M_best) > f(M_right) then
                M_best := M_right;
        M_right := M_next;
    if ((not forward) and (not backward)) then
        if (sm(M_left) ≤ sm(M_right)) then
            forward := true;
            backward := true;
        else
            break;
```

Content

Approach

- Use **cProfile** to find bottlenecks in python code

Name	Call Count	Time (ms)
shortl_bils.py	1	12 100.0%
shortl_bils	1	11 91.7%
main	1	11 91.7%
deepcopy	3671	7 58.3%
_deepcopy_list	415	6 50.0%
break_marriage_woman	40	2 16.7%
_load_unlocked	3	1 8.3%
_find_and_load_unlocked	3	1 8.3%
_find_and_load	3	1 8.3%
exec_module	3	1 8.3%
get_code	3	1 8.3%

Approach (cont)

- Bottlenecks in **ShortL-BiLS** algorithm

```
while (true) do
    if (forward) then
        neighborSet := ∅;
        for (each man m in the men set) do
            stableMatching := BreakMarriage(M_left, m);
            neighborSet := neighborSet ∪ stableMatching;
        if (small random probability p) then
            M_next := a random matching in neighborSet;
        else
            M_next := arg min_{M ∈ neighborSet}(f(M));
        if (f(M_next) > f(M_left)) then
            forward := false;
            if f(M_best) > f(M_left) then
                M_best := M_left;
        M_left := M_next;
```

Approach (cont)

```
while (continued) do
    if (backward) then
        neighborSet := ∅;
        for (each woman w in the women set) do
            stableMatching := BreakMarriage(M_right, w);
            neighborSet := neighborSet ∪ stableMatching;
        if (small random probability p) then
            M_next := a random matching in neighborSet;
        else
            M_next := arg min_{M ∈ neighborSet}(f(M));
        if (f(M_next) > f(M_right)) then
            backward := false;
            if f(M_best) > f(M_right) then
                M_best := M_right;
        M_right := M_next;
```

Approach (cont)

- Master-Slave architecture
- **mpi-master-slave** library (by **Luca Scarabello**)
  - *https://github.com/luca-s/mpi-master-slave*

Content

Experiments Results

Experimental result when using sequence and parallel method to find the solution

Approach	Execution time
Sequential	0.008949041366577148s
Parallel - 2 processors	3.21801495552063s
Parallel - 3 processors	1.318950176239013 7s
Parallel - 4 processors	0.903313159942627s

Experiments

The parallel method is too slow compared to the sequential method ???

The End