A simple assessed exercise

Ciaran McCreesh & Patrick Prosser + 21

Simple assessed exercise

10 credit course

- 10 weeks
- 30 lectures
- Equivalent to 100 hours in total
    - 30 lectures
    - 20% coursework
    - Self study

Simple assessed exercise

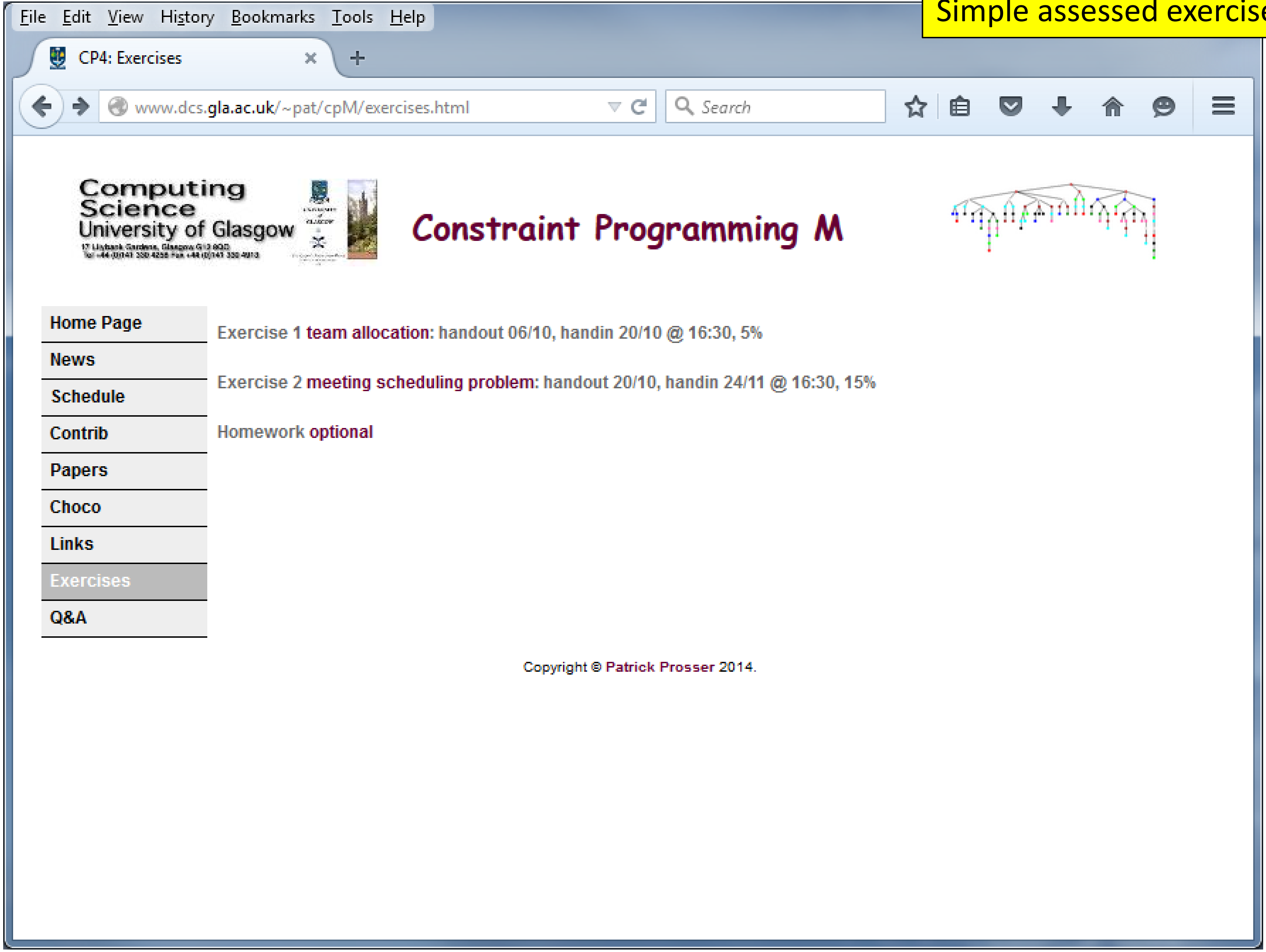
Simple exercise is 5% (about 1 day’s effort)
Handed out 2nd week of course
Get students using CP (get hands dirty)
Students have a rough idea about how CP works

Simple assessed exercise

Goals

- Must be easy to make progress
- Must be interesting
- Should be fun
    - students want to solve this problem
- Google-proof

Simple assessed exercise


File Edit View History Bookmarks Tools Help
CP4: Exercises
www.dcs.gla.ac.uk/~pat/cpM/exercises.html
Search
Computing Science University of Glasgow

Constraint Programming M

Home Page
News
Schedule
Contrib
Papers
Choco
Links
Exercises
Q&A

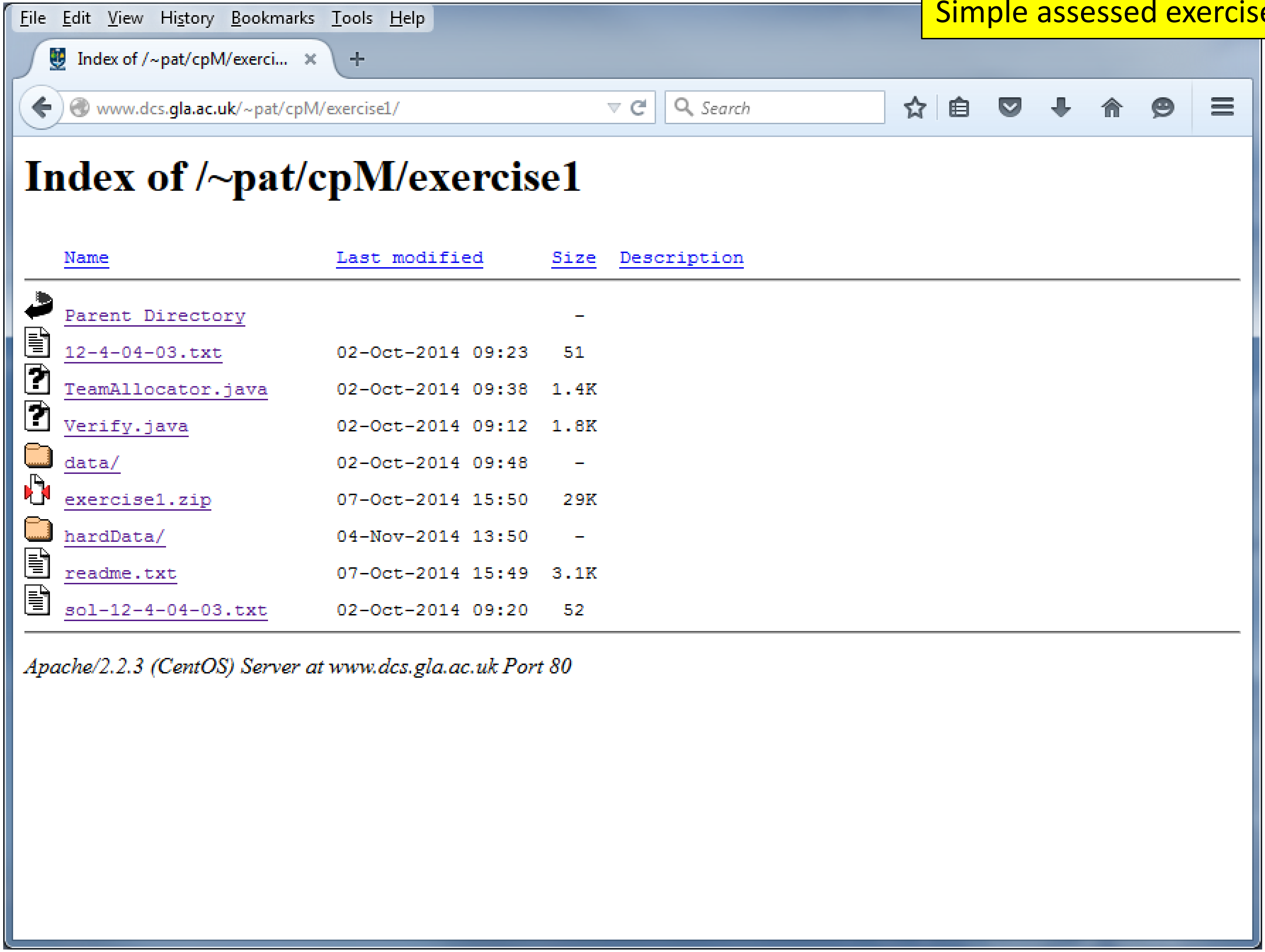
Exercise 1 team allocation: handout 06/10, handin 20/10 @ 16:30, 5%

Exercise 2 meeting scheduling problem: handout 20/10, handin 24/11 @ 16:30, 15%

Homework optional

Copyright © Patrick Prosser 2014.

Simple assessed exercise


File Edit View History Bookmarks Tools Help
Index of /~pat/cpM/exerci...
www.dcs.gla.ac.uk/~pat/cpM/exercise1/
Search

Index of /~pat/cpM/exercise1

```
     Name                   Last modified      Size  Description
--------------------------------------------------------------------
     Parent Directory                             -
     12-4-04-03.txt         02-Oct-2014 09:23    51
     TeamAllocator.java     02-Oct-2014 09:38   1.4K
     Verify.java            02-Oct-2014 09:12   1.8K
     data/                  02-Oct-2014 09:48     -
     exercise1.zip          07-Oct-2014 15:50    29K
     hardData/              04-Nov-2014 13:50     -
     readme.txt             07-Oct-2014 15:49   3.1K
     sol-12-4-04-03.txt     02-Oct-2014 09:20    52
--------------------------------------------------------------------
```

Apache/2.2.3 (CentOS) Server at www.dcs.gla.ac.uk Port 80

Team Allocator

Simple assessed exercise

You are given n players to be allocated to m teams (where n % m = 0).

There are constraints of the form together(i,j) and apart(i,j) where
- together(i,j) means that players i and j must be in the same team
- apart(i,j) that players i and j must be in different teams.

By default, players can be in any team with any other player.

Team Allocator

Simple assessed exercise

You are given n players to be allocated to m teams (where n % m = 0).

There are constraints of the form together(i,j) and apart(i,j) where
- together(i,j) means that players i and j must be in the same team
- apart(i,j) that players i and j must be in different teams.

By default, players can be in any team with any other player.

```
12 4
together 3 9
together 5 9
apart 2 8
apart 6 8
```

Team Allocator

Simple assessed exercise

You are given n players to be allocated to m teams (where n % m = 0).

There are constraints of the form together(i,j) and apart(i,j) where

- together(i,j) means that players i and j must be in the same team
- apart(i,j) that players i and j must be in different teams.

By default, players can be in any team with any other player.

```
12 4
together 3 9
together 5 9
apart 2 8
apart 6 8
```

$\leftarrow$ 12 players split into 4 teams (each of 3 players)

Team Allocator

Simple assessed exercise

You are given n players to be allocated to m teams (where n % m = 0).

There are constraints of the form together(i,j) and apart(i,j) where
- together(i,j) means that players i and j must be in the same team
- apart(i,j) that players i and j must be in different teams.

By default, players can be in any team with any other player.

```
12 4
together 3 9
together 5 9
apart 2 8
apart 6 8
```

Players 3 and 9 in same team
Players 5 and 9 in same team

Team Allocator

Simple assessed exercise

You are given n players to be allocated to m teams (where n % m = 0).

There are constraints of the form together(i,j) and apart(i,j) where
- together(i,j) means that players i and j must be in the same team
- apart(i,j) that players i and j must be in different teams.

By default, players can be in any team with any other player.

```
12 4
together 3 9
together 5 9
apart 2 8
apart 6 8
```

Players 2 and 8 in different teams
Players 6 and 8 in different teams

Team Allocator

Simple assessed exercise

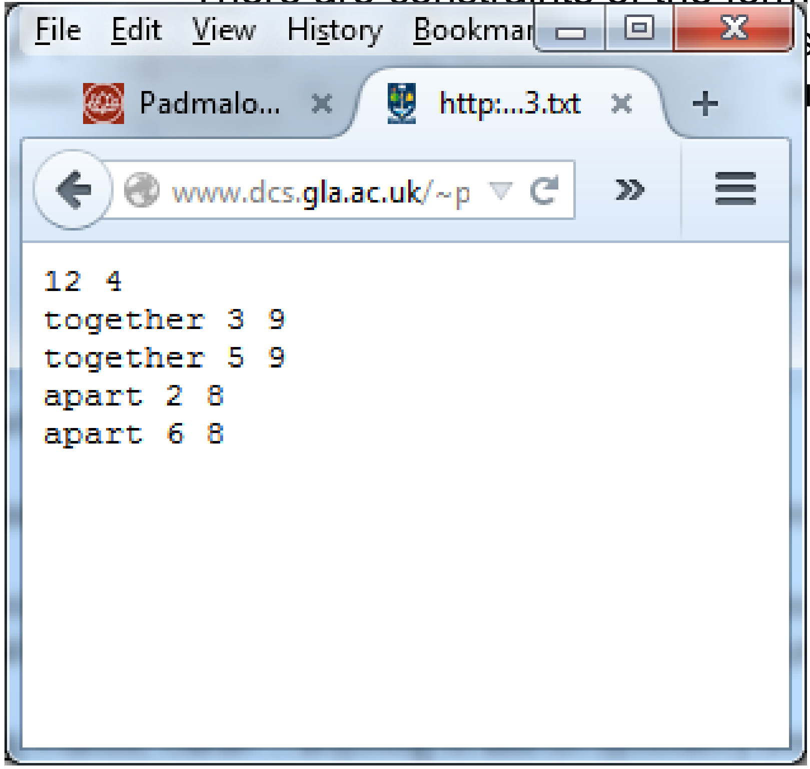
You are given n players to be allocated to m teams (where n % m = 0).

There are constraints of the form together(i,j) and apart(i,j) where

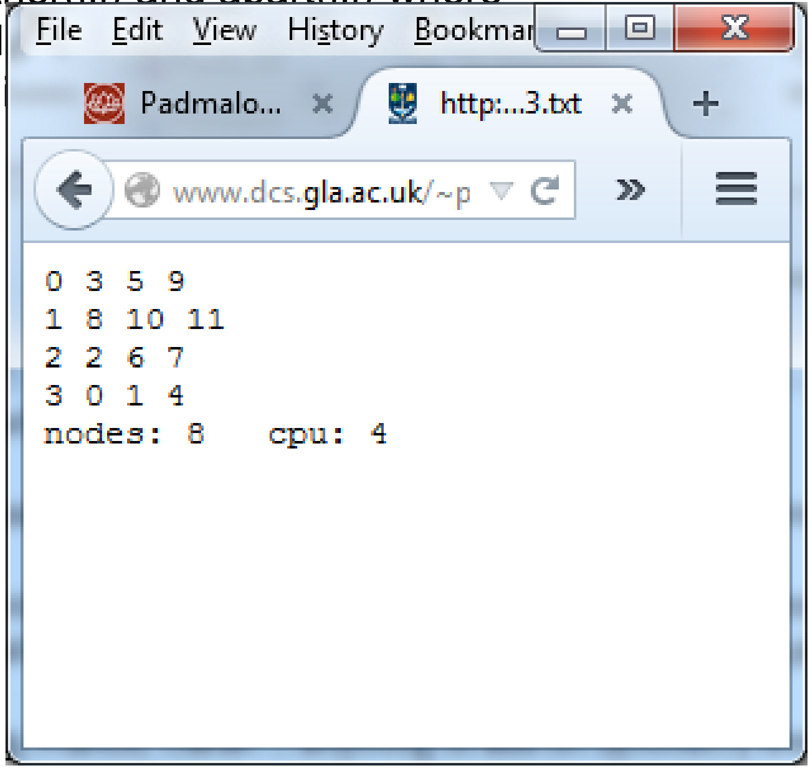
s i and

ıst be i

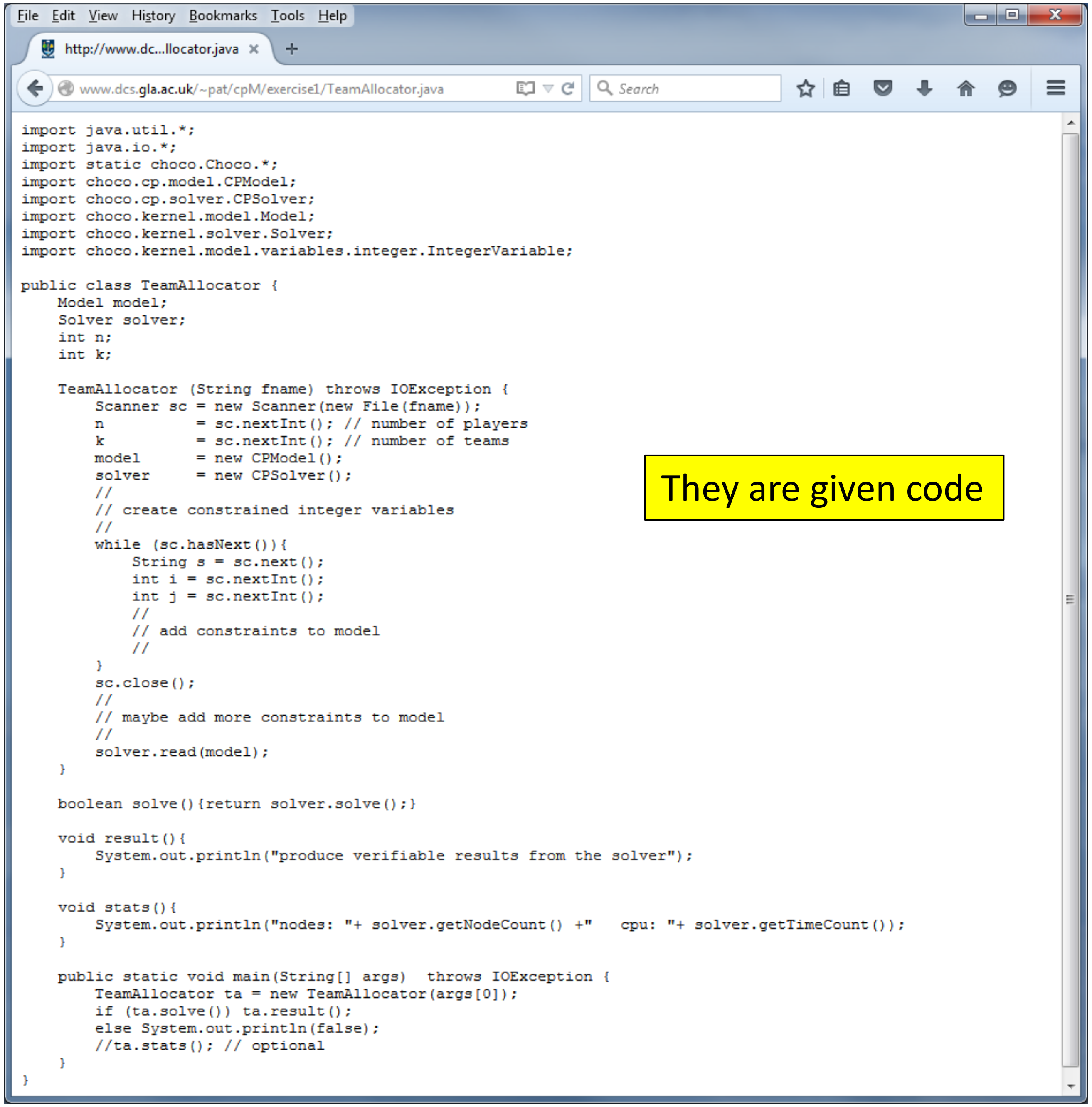
team



```
12 4
together 3 9
together 5 9
apart 2 8
apart 6 8
```



```
0 3 5 9
1 8 10 11
2 2 6 7
3 0 1 4
nodes: 8   cpu: 4
```

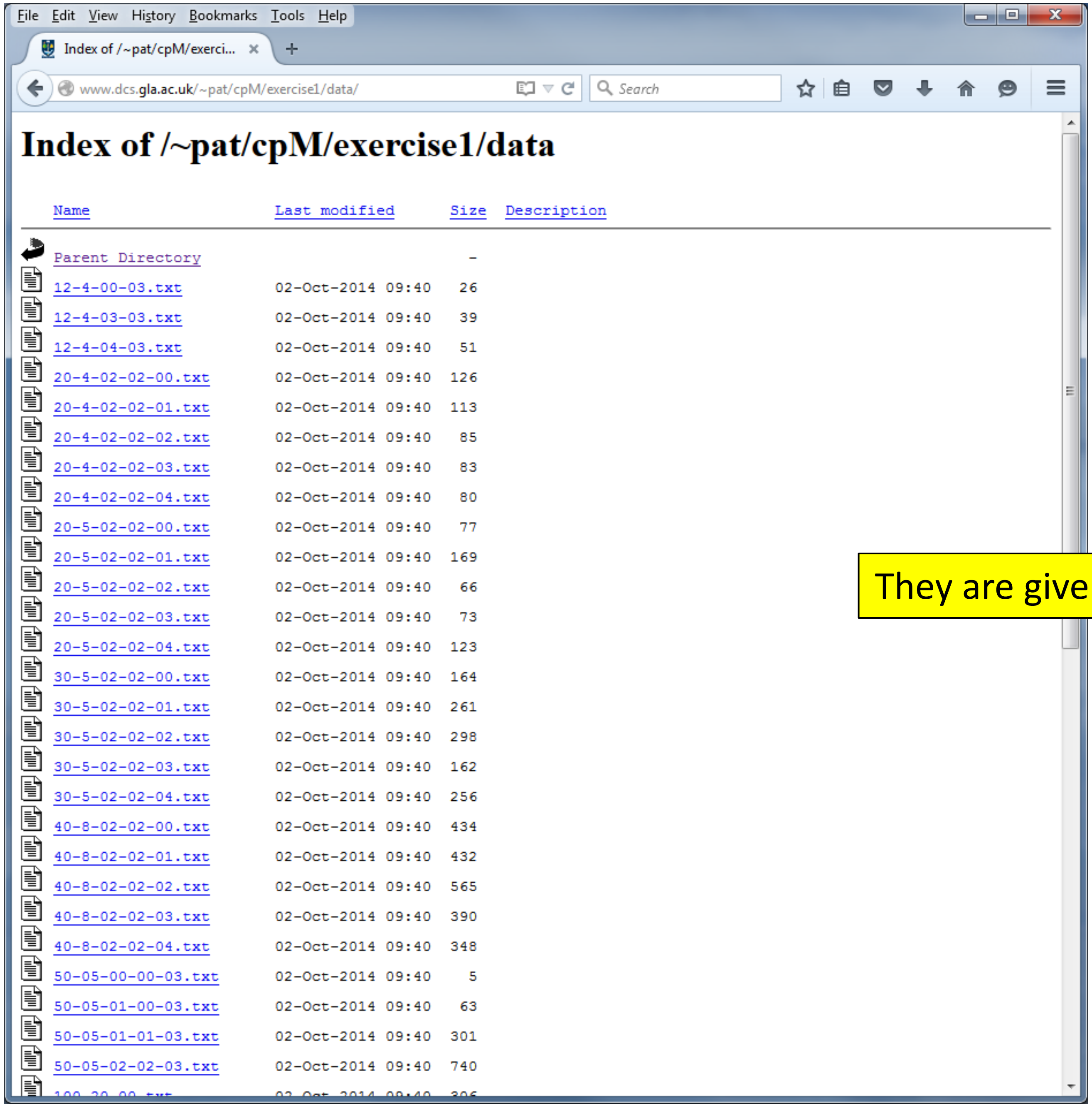

File Edit View History Bookmarks Tools Help
http://www.dc...llocator.java
www.dcs.gla.ac.uk/~pat/cpM/exercise1/TeamAllocator.java
Search

```
import java.util.*;
import java.io.*;
import static choco.Choco.*;
import choco.cp.model.CPModel;
import choco.cp.solver.CPSolver;
import choco.kernel.model.Model;
import choco.kernel.solver.Solver;
import choco.kernel.model.variables.integer.IntegerVariable;

public class TeamAllocator {
    Model model;
    Solver solver;
    int n;
    int k;

    TeamAllocator (String fname) throws IOException {
        Scanner sc = new Scanner(new File(fname));
        n          = sc.nextInt(); // number of players
        k          = sc.nextInt(); // number of teams
        model      = new CPModel();
        solver     = new CPSolver();
        //
        // create constrained integer variables
        //
        while (sc.hasNext()){
            String s = sc.next();
            int i = sc.nextInt();
            int j = sc.nextInt();
            //
            // add constraints to model
            //
        }
        sc.close();
        //
        // maybe add more constraints to model
        //
        solver.read(model);
    }

    boolean solve(){return solver.solve();}

    void result(){
        System.out.println("produce verifiable results from the solver");
    }

    void stats(){
        System.out.println("nodes: "+ solver.getNodeCount() +"   cpu: "+ solver.getTimeCount());
    }

    public static void main(String[] args)  throws IOException {
        TeamAllocator ta = new TeamAllocator(args[0]);
        if (ta.solve()) ta.result();
        else System.out.println(false);
        //ta.stats(); // optional
    }
}
```

They are given code

```
TeamAllocator (String fname) throws IOException {
    Scanner sc = new Scanner(new File(fname));
    n          = sc.nextInt(); // number of players
    k          = sc.nextInt(); // number of teams
    model      = new CPModel();
    solver     = new CPSolver();
    //
    // create constrained integer variables
    //
    while (sc.hasNext()){
        String s = sc.next();
        int i = sc.nextInt();
        int j = sc.nextInt();
        //
        // add constraints to model
        //
    }
    sc.close();
    //
    // maybe add more constraints to model
    //
    solver.read(model);
}
```

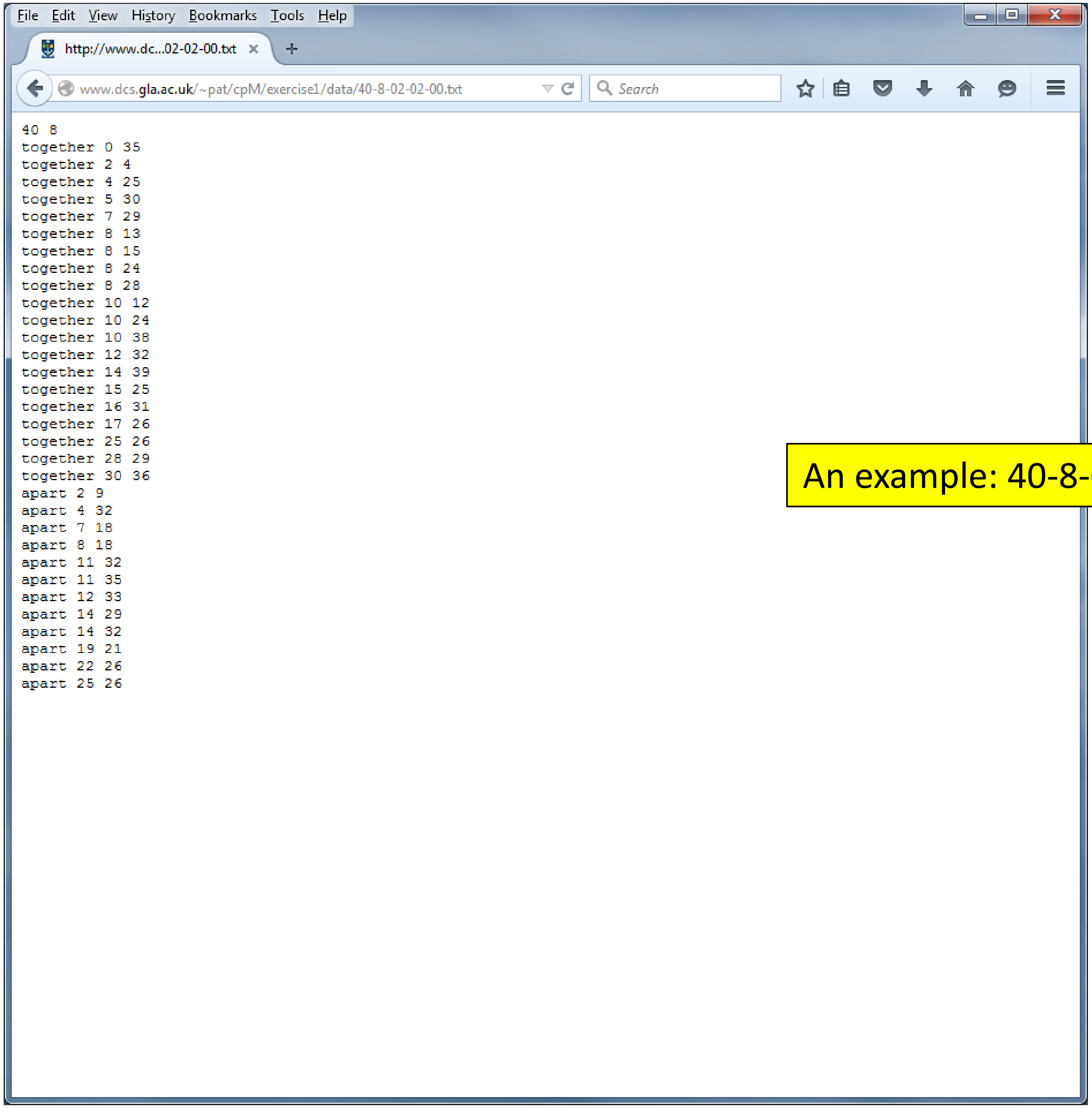
They have to add code


File Edit View History Bookmarks Tools Help
Index of /~pat/cpM/exerci...
www.dcs.gla.ac.uk/~pat/cpM/exercise1/data/
Search

Index of /~pat/cpM/exercise1/data

```
      Name                    Last modified       Size  Description
-------------------------------------------------------------------
      Parent Directory                               -
      12-4-00-03.txt          02-Oct-2014 09:40     26
      12-4-03-03.txt          02-Oct-2014 09:40     39
      12-4-04-03.txt          02-Oct-2014 09:40     51
      20-4-02-02-00.txt       02-Oct-2014 09:40    126
      20-4-02-02-01.txt       02-Oct-2014 09:40    113
      20-4-02-02-02.txt       02-Oct-2014 09:40     85
      20-4-02-02-03.txt       02-Oct-2014 09:40     83
      20-4-02-02-04.txt       02-Oct-2014 09:40     80
      20-5-02-02-00.txt       02-Oct-2014 09:40     77
      20-5-02-02-01.txt       02-Oct-2014 09:40    169
      20-5-02-02-02.txt       02-Oct-2014 09:40     66
      20-5-02-02-03.txt       02-Oct-2014 09:40     73
      20-5-02-02-04.txt       02-Oct-2014 09:40    123
      30-5-02-02-00.txt       02-Oct-2014 09:40    164
      30-5-02-02-01.txt       02-Oct-2014 09:40    261
      30-5-02-02-02.txt       02-Oct-2014 09:40    298
      30-5-02-02-03.txt       02-Oct-2014 09:40    162
      30-5-02-02-04.txt       02-Oct-2014 09:40    256
      40-8-02-02-00.txt       02-Oct-2014 09:40    434
      40-8-02-02-01.txt       02-Oct-2014 09:40    432
      40-8-02-02-02.txt       02-Oct-2014 09:40    565
      40-8-02-02-03.txt       02-Oct-2014 09:40    390
      40-8-02-02-04.txt       02-Oct-2014 09:40    348
      50-05-00-00-03.txt      02-Oct-2014 09:40      5
      50-05-01-00-03.txt      02-Oct-2014 09:40     63
      50-05-01-01-03.txt      02-Oct-2014 09:40    301
      50-05-02-02-03.txt      02-Oct-2014 09:40    740
      100-20-00.txt           02-Oct-2014 09:40    306
```

They are given problem instances


File Edit View History Bookmarks Tools Help
http://www.dc...02-02-00.txt
www.dcs.gla.ac.uk/~pat/cpM/exercise1/data/40-8-02-02-00.txt
Search

```
40 8
together 0 35
together 2 4
together 4 25
together 5 30
together 7 29
together 8 13
together 8 15
together 8 24
together 8 28
together 10 12
together 10 24
together 10 38
together 12 32
together 14 39
together 15 25
together 16 31
together 17 26
together 25 26
together 28 29
together 30 36
apart 2 9
apart 4 32
apart 7 18
apart 8 18
apart 11 32
apart 11 35
apart 12 33
apart 14 29
apart 14 32
apart 19 21
apart 22 26
apart 25 26
```

An example: 40-8-02-00.txt

Simple solution (and an instance)

```
12 4
together 9 10
apart 1 6
apart 7 9
```

- Create an array of constrained integer variables player[0] to player[11]
- Each has a domain {1..4}, the teams they can be in
- For apart(i,j) post constraint player[i] ≠ player[j]
- For together(i,j) post constraint player[i] = player[j]
- Use occurrence or cardinality constraint to ensure that each team occurs n/m times (i.e. number of players per team is satisfied)

Easy to get hands dirty

Simple solution (and an instance)

```
12 4
together 9 10
apart 1 6
apart 7 9
```

- Create an array of constrained integer variables [illegible]ayer[11]
- Each has a domain {1..4}, the teams they c[illegible]
- For apart(i,j) post constraint player[i[illegible]
- For together(i,j) post constrain[illegible]ayer[j]
- Use occurrence or cardin[illegible] to ensure that each team occurs n/m times (i[illegible] players per team is satisfied)

A variant of equitable graph colouring

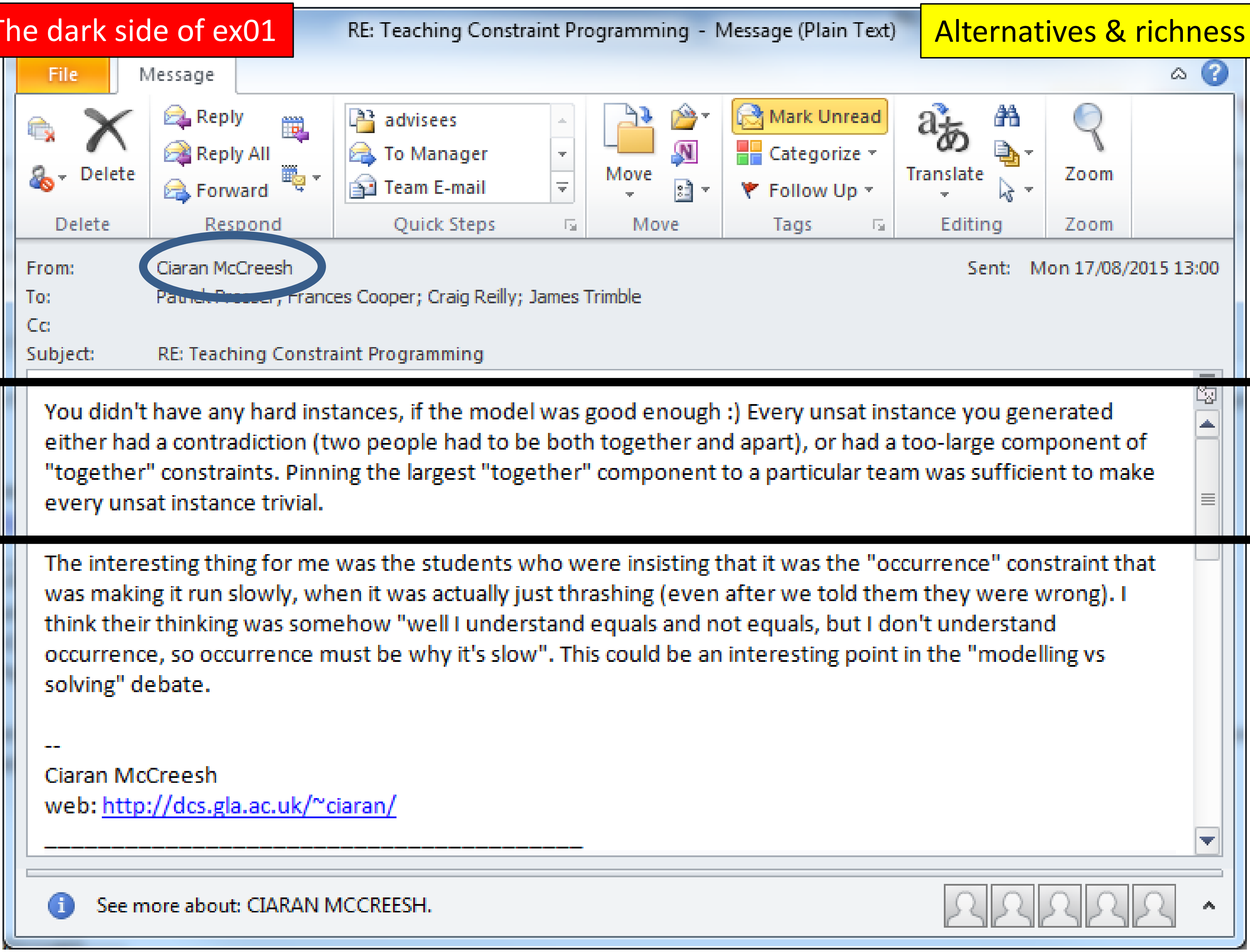
Easy to get hands dirty

Alternatives & richness

- Use a 0/1 model, 2D array, row for team, column for player
- Use set variables, a set for each team
- Pre-processing
- Symmetry breaking
- Variable ordering heuristics
- We have hard instances (>12 hours to solve)
- Devoted 1 lecture to discussing problem after deadline

The dark side of ex01

Alternatives & richness



From: Ciaran McCreesh

Sent: Mon 17/08/2015 13:00

To: Patrick Prosser; Frances Cooper; Craig Reilly; James Trimble

Cc:

Subject: RE: Teaching Constraint Programming

You didn't have any hard instances, if the model was good enough :) Every unsat instance you generated either had a contradiction (two people had to be both together and apart), or had a too-large component of "together" constraints. Pinning the largest "together" component to a particular team was sufficient to make every unsat instance trivial.

The interesting thing for me was the students who were insisting that it was the "occurrence" constraint that was making it run slowly, when it was actually just thrashing (even after we told them they were wrong). I think their thinking was somehow "well I understand equals and not equals, but I don't understand occurrence, so occurrence must be why it's slow". This could be an interesting point in the "modelling vs solving" debate.

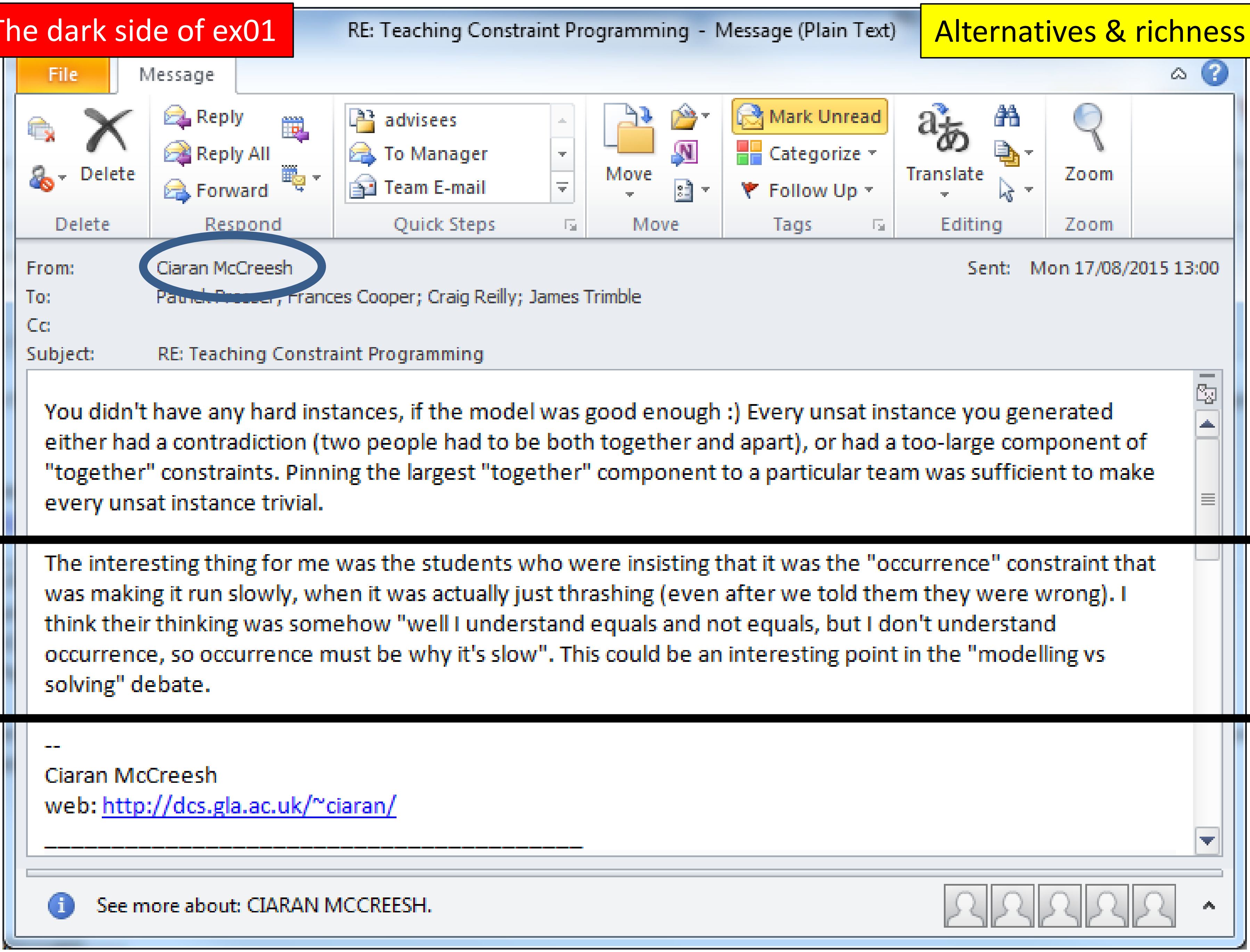
--
Ciaran McCreesh
web: http://dcs.gla.ac.uk/~ciaran/

____

See more about: CIARAN MCCREESH.

The dark side of ex01

Alternatives & richness



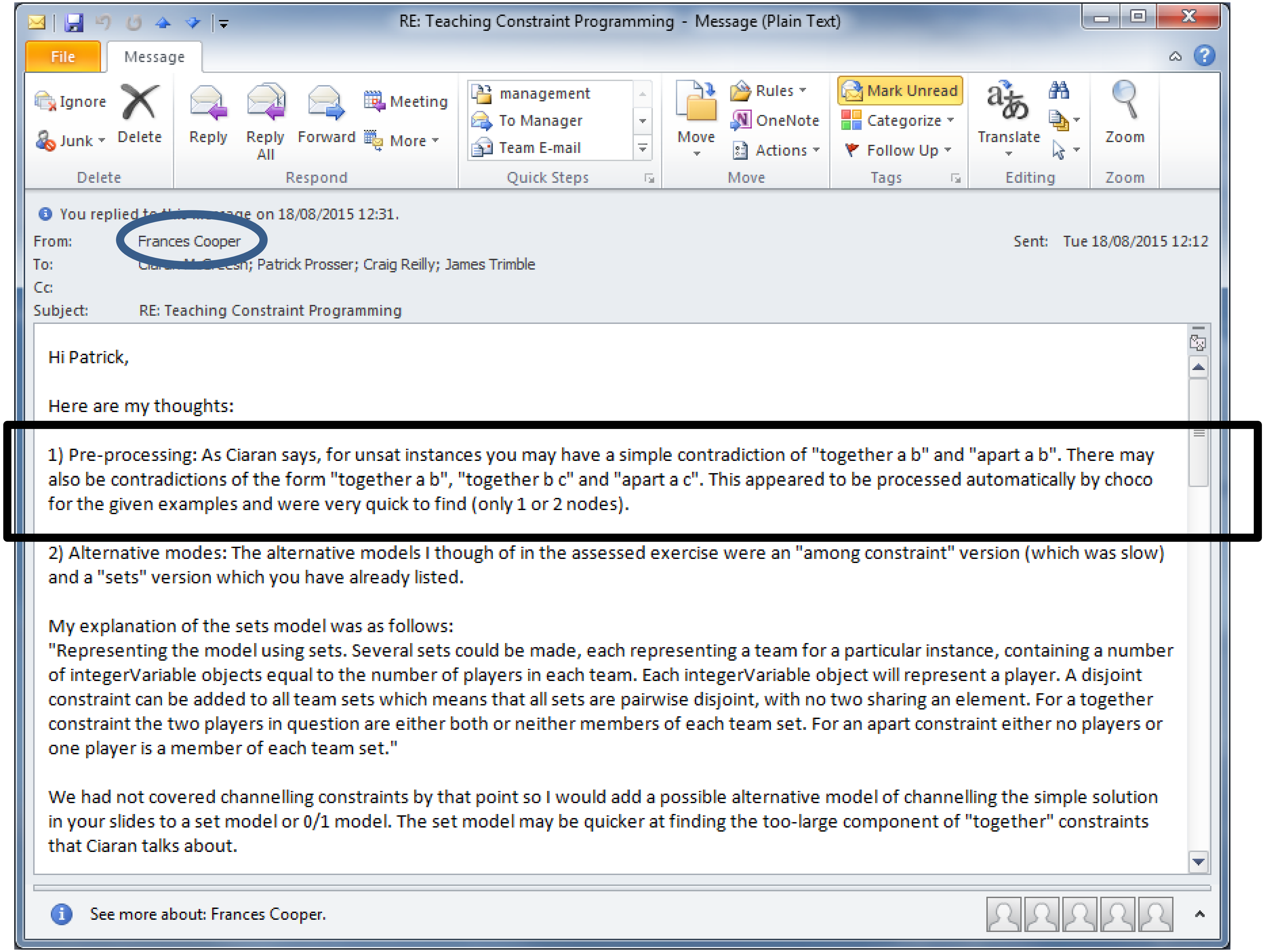
From: Ciaran McCreesh
Sent: Mon 17/08/2015 13:00
To: Patrick Prosser; Frances Cooper; Craig Reilly; James Trimble
Cc:
Subject: RE: Teaching Constraint Programming

You didn't have any hard instances, if the model was good enough :) Every unsat instance you generated either had a contradiction (two people had to be both together and apart), or had a too-large component of "together" constraints. Pinning the largest "together" component to a particular team was sufficient to make every unsat instance trivial.

The interesting thing for me was the students who were insisting that it was the "occurrence" constraint that was making it run slowly, when it was actually just thrashing (even after we told them they were wrong). I think their thinking was somehow "well I understand equals and not equals, but I don't understand occurrence, so occurrence must be why it's slow". This could be an interesting point in the "modelling vs solving" debate.

--
Ciaran McCreesh
web: http://dcs.gla.ac.uk/~ciaran/
____

See more about: CIARAN MCCREESH.



You replied to this message on 18/08/2015 12:31.

From: Frances Cooper

Sent: Tue 18/08/2015 12:12

To: Ciaran McCreesh; Patrick Prosser; Craig Reilly; James Trimble

Cc:

Subject: RE: Teaching Constraint Programming

Hi Patrick,

Here are my thoughts:

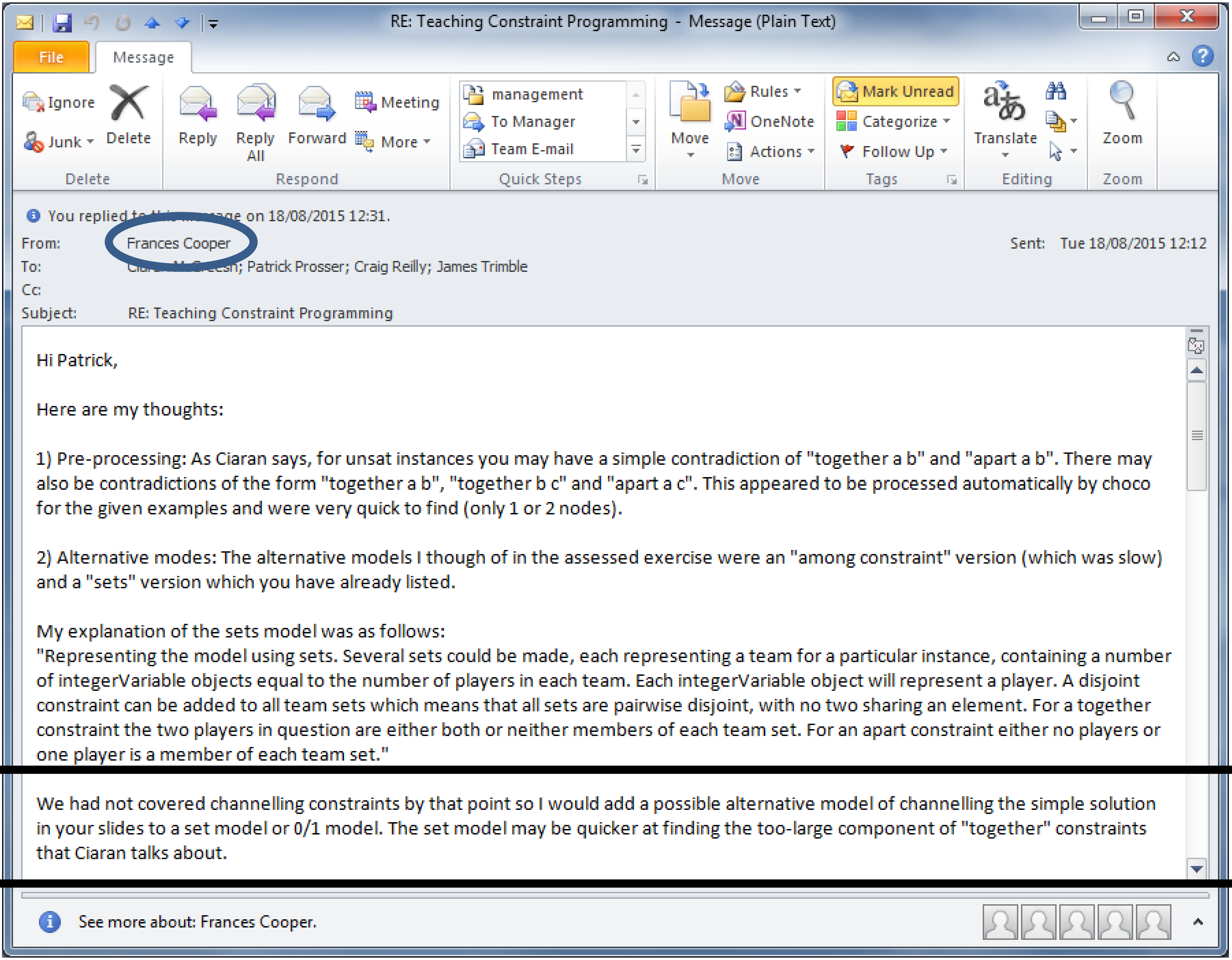
1) Pre-processing: As Ciaran says, for unsat instances you may have a simple contradiction of "together a b" and "apart a b". There may also be contradictions of the form "together a b", "together b c" and "apart a c". This appeared to be processed automatically by choco for the given examples and were very quick to find (only 1 or 2 nodes).

2) Alternative modes: The alternative models I though of in the assessed exercise were an "among constraint" version (which was slow) and a "sets" version which you have already listed.

My explanation of the sets model was as follows:
"Representing the model using sets. Several sets could be made, each representing a team for a particular instance, containing a number of integerVariable objects equal to the number of players in each team. Each integerVariable object will represent a player. A disjoint constraint can be added to all team sets which means that all sets are pairwise disjoint, with no two sharing an element. For a together constraint the two players in question are either both or neither members of each team set. For an apart constraint either no players or one player is a member of each team set."

We had not covered channelling constraints by that point so I would add a possible alternative model of channelling the simple solution in your slides to a set model or 0/1 model. The set model may be quicker at finding the too-large component of "together" constraints that Ciaran talks about.

See more about: Frances Cooper.



You replied to this message on 18/08/2015 12:31.

From: Frances Cooper

Sent: Tue 18/08/2015 12:12

To: Ciaran McCreesh; Patrick Prosser; Craig Reilly; James Trimble

Cc:

Subject: RE: Teaching Constraint Programming

Hi Patrick,

Here are my thoughts:

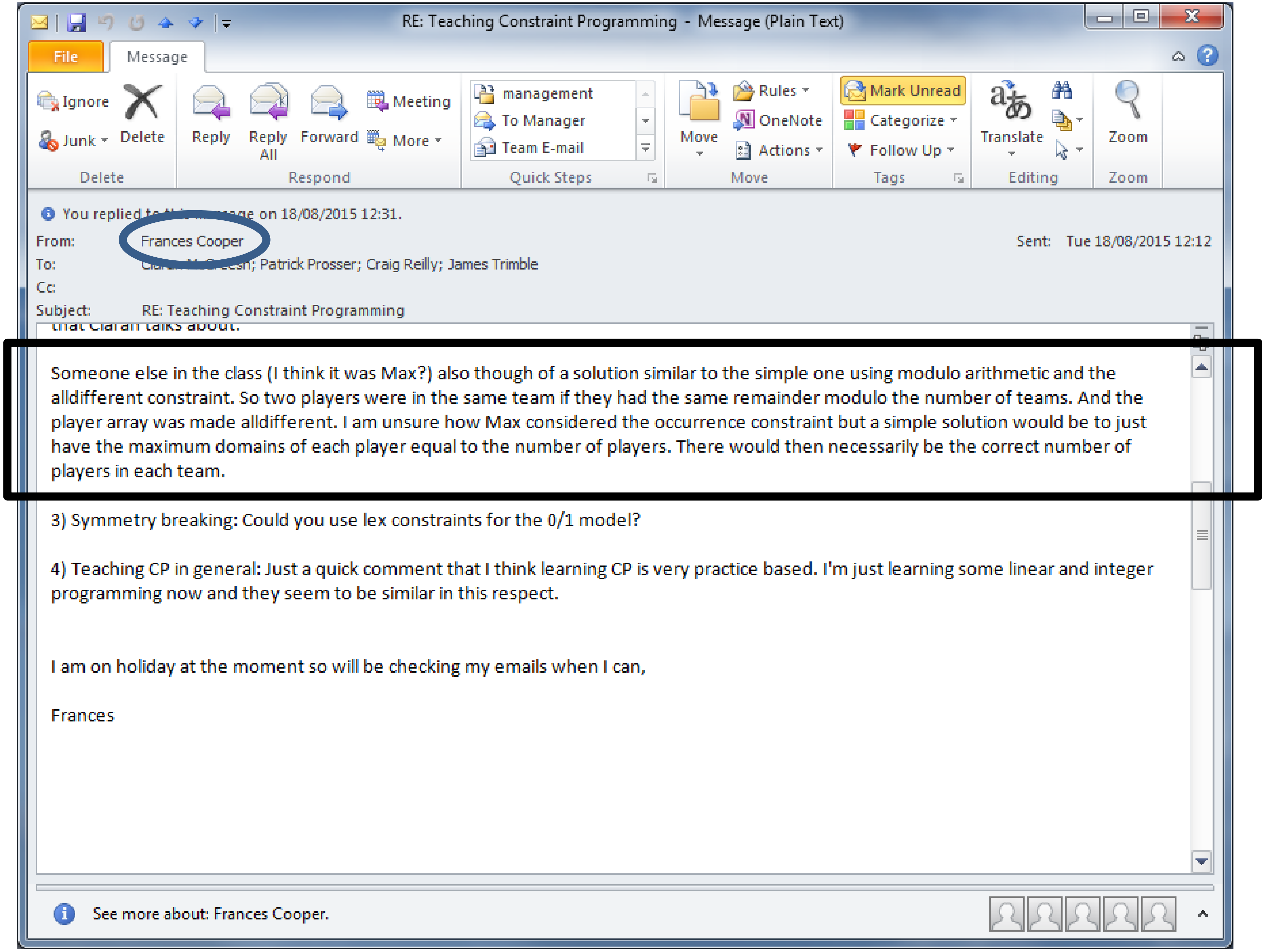
1) Pre-processing: As Ciaran says, for unsat instances you may have a simple contradiction of "together a b" and "apart a b". There may also be contradictions of the form "together a b", "together b c" and "apart a c". This appeared to be processed automatically by choco for the given examples and were very quick to find (only 1 or 2 nodes).

2) Alternative modes: The alternative models I though of in the assessed exercise were an "among constraint" version (which was slow) and a "sets" version which you have already listed.

My explanation of the sets model was as follows:
"Representing the model using sets. Several sets could be made, each representing a team for a particular instance, containing a number of integerVariable objects equal to the number of players in each team. Each integerVariable object will represent a player. A disjoint constraint can be added to all team sets which means that all sets are pairwise disjoint, with no two sharing an element. For a together constraint the two players in question are either both or neither members of each team set. For an apart constraint either no players or one player is a member of each team set."

We had not covered channelling constraints by that point so I would add a possible alternative model of channelling the simple solution in your slides to a set model or 0/1 model. The set model may be quicker at finding the too-large component of "together" constraints that Ciaran talks about.

See more about: Frances Cooper.



You replied to this message on 18/08/2015 12:31.

From: Frances Cooper

Sent: Tue 18/08/2015 12:12

To: Ciaran McCreesh; Patrick Prosser; Craig Reilly; James Trimble

Cc:

Subject: RE: Teaching Constraint Programming

that Ciaran talks about.

Someone else in the class (I think it was Max?) also though of a solution similar to the simple one using modulo arithmetic and the alldifferent constraint. So two players were in the same team if they had the same remainder modulo the number of teams. And the player array was made alldifferent. I am unsure how Max considered the occurrence constraint but a simple solution would be to just have the maximum domains of each player equal to the number of players. There would then necessarily be the correct number of players in each team.

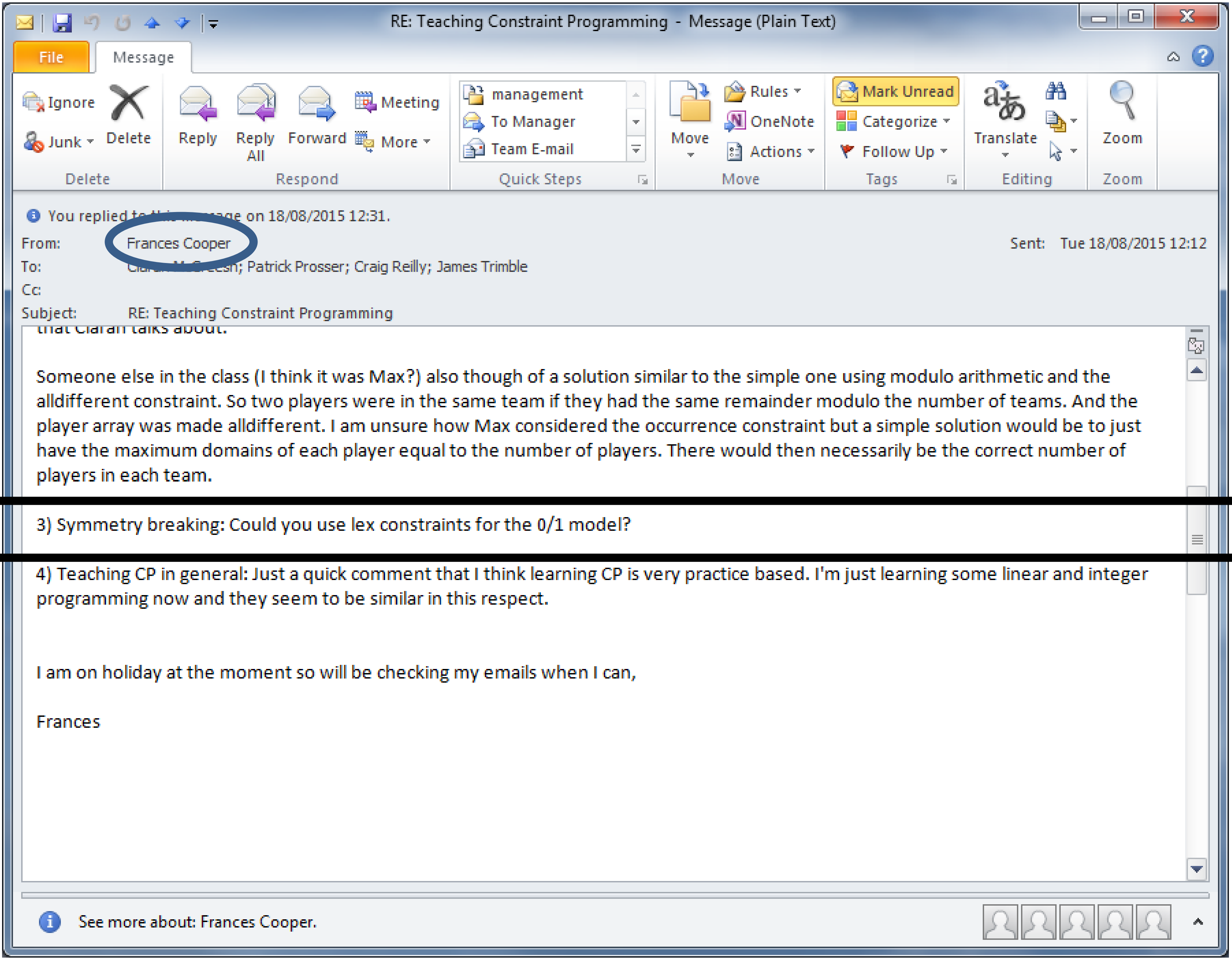
3) Symmetry breaking: Could you use lex constraints for the 0/1 model?

4) Teaching CP in general: Just a quick comment that I think learning CP is very practice based. I'm just learning some linear and integer programming now and they seem to be similar in this respect.

I am on holiday at the moment so will be checking my emails when I can,

Frances

See more about: Frances Cooper.



RE: Teaching Constraint Programming - Message (Plain Text)

You replied to this message on 18/08/2015 12:31.

From: Frances Cooper

Sent: Tue 18/08/2015 12:12

To: Ciaran McCreesh; Patrick Prosser; Craig Reilly; James Trimble

Cc:

Subject: RE: Teaching Constraint Programming

that Ciaran talks about.

Someone else in the class (I think it was Max?) also though of a solution similar to the simple one using modulo arithmetic and the alldifferent constraint. So two players were in the same team if they had the same remainder modulo the number of teams. And the player array was made alldifferent. I am unsure how Max considered the occurrence constraint but a simple solution would be to just have the maximum domains of each player equal to the number of players. There would then necessarily be the correct number of players in each team.

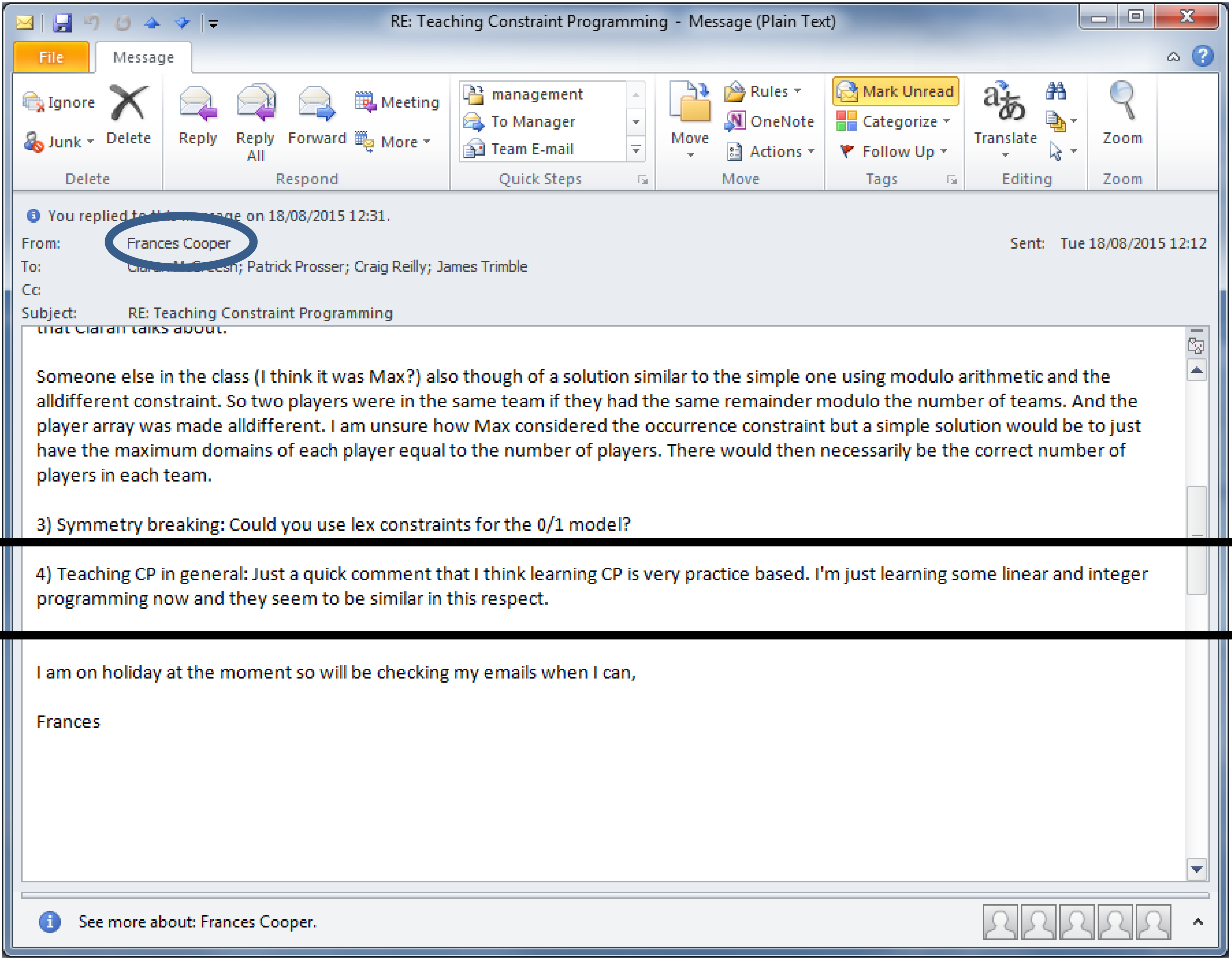
3) Symmetry breaking: Could you use lex constraints for the 0/1 model?

4) Teaching CP in general: Just a quick comment that I think learning CP is very practice based. I'm just learning some linear and integer programming now and they seem to be similar in this respect.

I am on holiday at the moment so will be checking my emails when I can,

Frances

See more about: Frances Cooper.



You replied to this message on 18/08/2015 12:31.

From: Frances Cooper

Sent: Tue 18/08/2015 12:12

To: Ciaran McCreesh; Patrick Prosser; Craig Reilly; James Trimble

Cc:

Subject: RE: Teaching Constraint Programming

that Ciaran talks about.

Someone else in the class (I think it was Max?) also though of a solution similar to the simple one using modulo arithmetic and the alldifferent constraint. So two players were in the same team if they had the same remainder modulo the number of teams. And the player array was made alldifferent. I am unsure how Max considered the occurrence constraint but a simple solution would be to just have the maximum domains of each player equal to the number of players. There would then necessarily be the correct number of players in each team.

3) Symmetry breaking: Could you use lex constraints for the 0/1 model?

4) Teaching CP in general: Just a quick comment that I think learning CP is very practice based. I'm just learning some linear and integer programming now and they seem to be similar in this respect.

I am on holiday at the moment so will be checking my emails when I can,

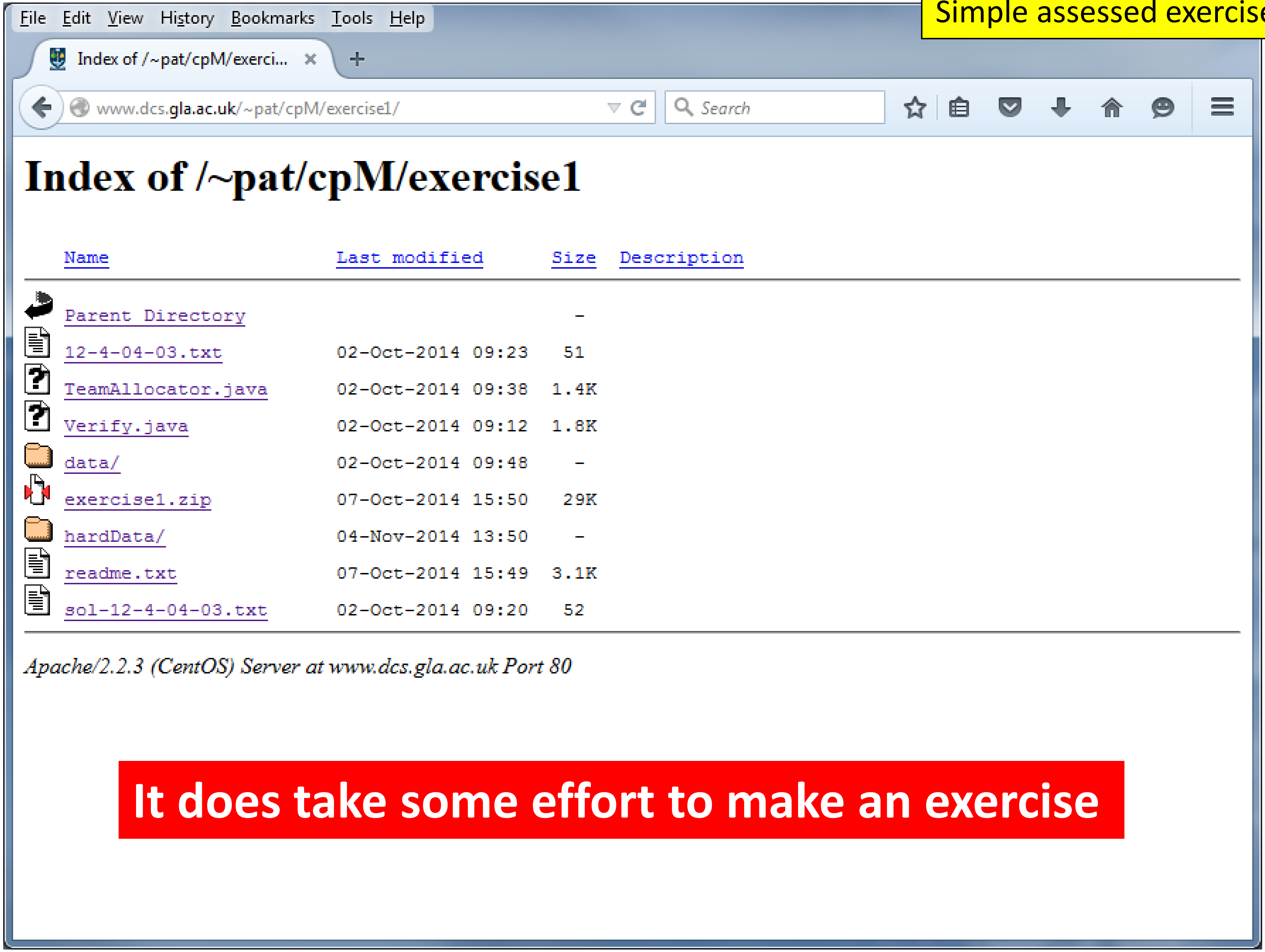
Frances

See more about: Frances Cooper.

Google-proof?

- 20 animals escape from the zoo
- We have 5 cages to put them in
- Each cage can take at most 4 animals
- The following animals cannot be in the same cage
  - The rabbit and the fox
  - The spider and the fly
  - The worm and the robin
  - …

Simple assessed exercise


File Edit View History Bookmarks Tools Help
Index of /~pat/cpM/exerci...
www.dcs.gla.ac.uk/~pat/cpM/exercise1/
Search

Index of /~pat/cpM/exercise1

```
      Name                         Last modified         Size  Description
--------------------------------------------------------------------------------
      Parent Directory                                      -
      12-4-04-03.txt              02-Oct-2014 09:23    51
      TeamAllocator.java          02-Oct-2014 09:38   1.4K
      Verify.java                 02-Oct-2014 09:12   1.8K
      data/                       02-Oct-2014 09:48     -
      exercise1.zip               07-Oct-2014 15:50    29K
      hardData/                   04-Nov-2014 13:50     -
      readme.txt                  07-Oct-2014 15:49   3.1K
      sol-12-4-04-03.txt          02-Oct-2014 09:20    52
--------------------------------------------------------------------------------
```

Apache/2.2.3 (CentOS) Server at www.dcs.gla.ac.uk Port 80

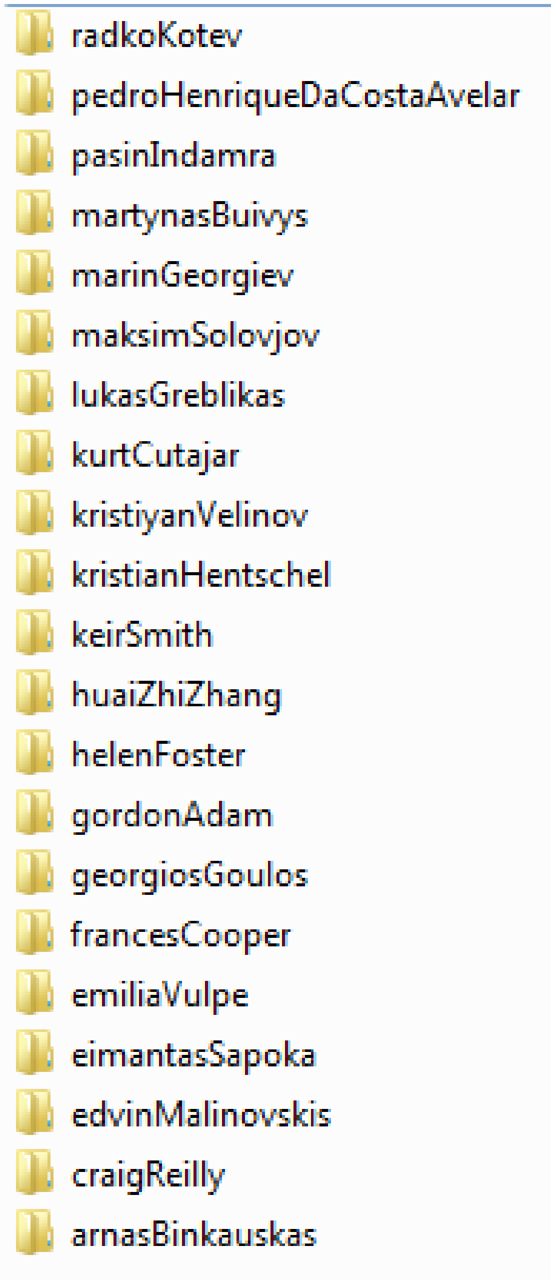
It does take some effort to make an exercise

conclusion

This went surprisingly well

- I think they liked the problem
- Generated a lot of discussion & interaction
- I think they got the idea of CP and the problems we can solve
    - Not just mashing up data

… with a little help from my friends



- radkoKotev
- pedroHenriqueDaCostaAvelar
- pasinIndamra
- martynasBuivys
- marinGeorgiev
- maksimSolovjov
- lukasGreblikas
- kurtCutajar
- kristiyanVelinov
- kristianHentschel
- keirSmith
- huaiZhiZhang
- helenFoster
- gordonAdam
- georgiosGoulos
- francesCooper
- emiliaVulpe
- eimantasSapoka
- edvinMalinovskis
- craigReilly
- arnasBinkauskas