# Student – Class Assignment Optimization

K.Ciebiera, M.Mucha

# USOS

- The *University Study-Oriented System* is an integrated student management information system for handling student affairs at Polish universities.
- Its development and deployment is coordinated and supported financially by the consortium of Polish higher education institutions.

# Main problems to solve

Not enough “high quality” resources:

- lecturers
- exam and course slots
- interesting courses

***We (USOS authors) cannot solve these problems.***

# In the press

# Registration methods

1. Manual registration by dean's office.
2. Registration outside of the system.
3. Token based registration.
4. Preferences based registration.

# Two-phase registration model

In **phase 1** students are registered by dean's office to courses.

**Phase 2** consists of following steps:

1. Students define their preferences.
2. System is taken to read-only mode.
3. Engine performs student-class assignments.
4. Assignment results are shown to students.

# Constraints

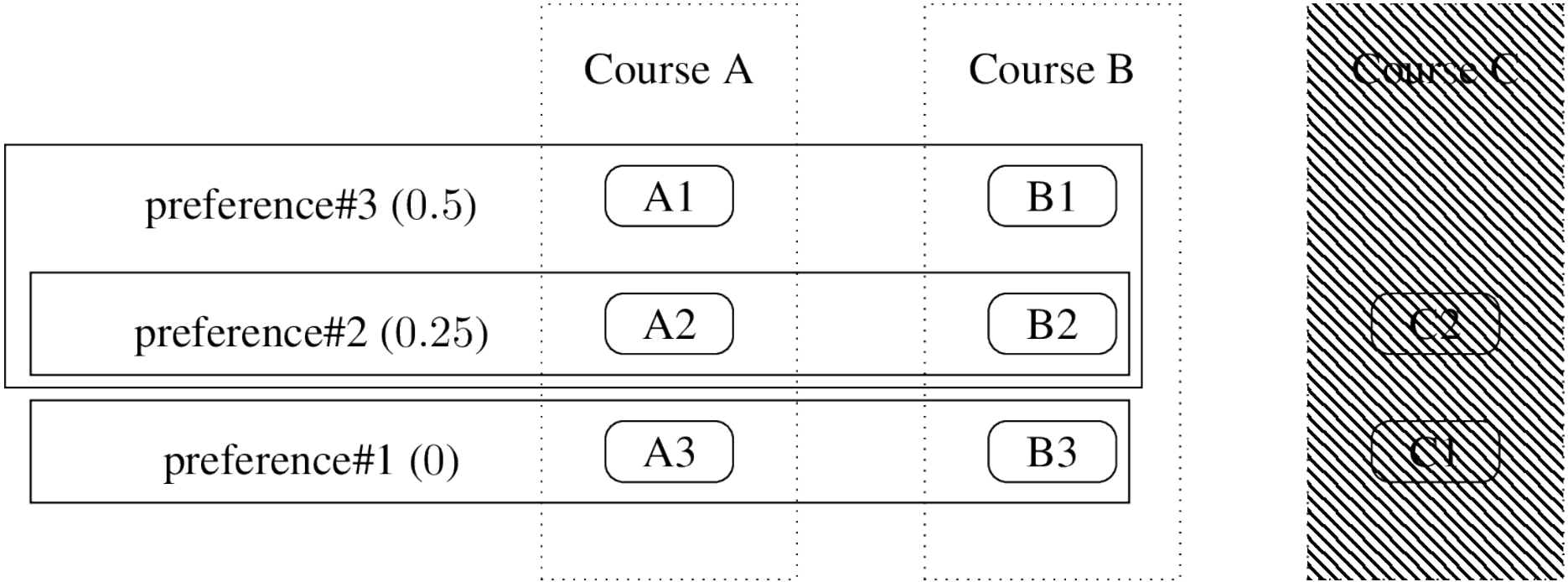
1. Each class has its own **schedule** and a **limit** on number of students.
2. Classes are in **conflict** one with another if their schedules overlap.
3. Some pairs of classes are **excluded**.

# Students preferences

1. **Number of conflicts**, student may mark some courses as not important for him.
2. Students may **prioritize classes** of all his important courses. They define sequence of subsets of all classes.

*Web based-self service system.*

# Preferences - example



# Unsuccessful solution (greedy)

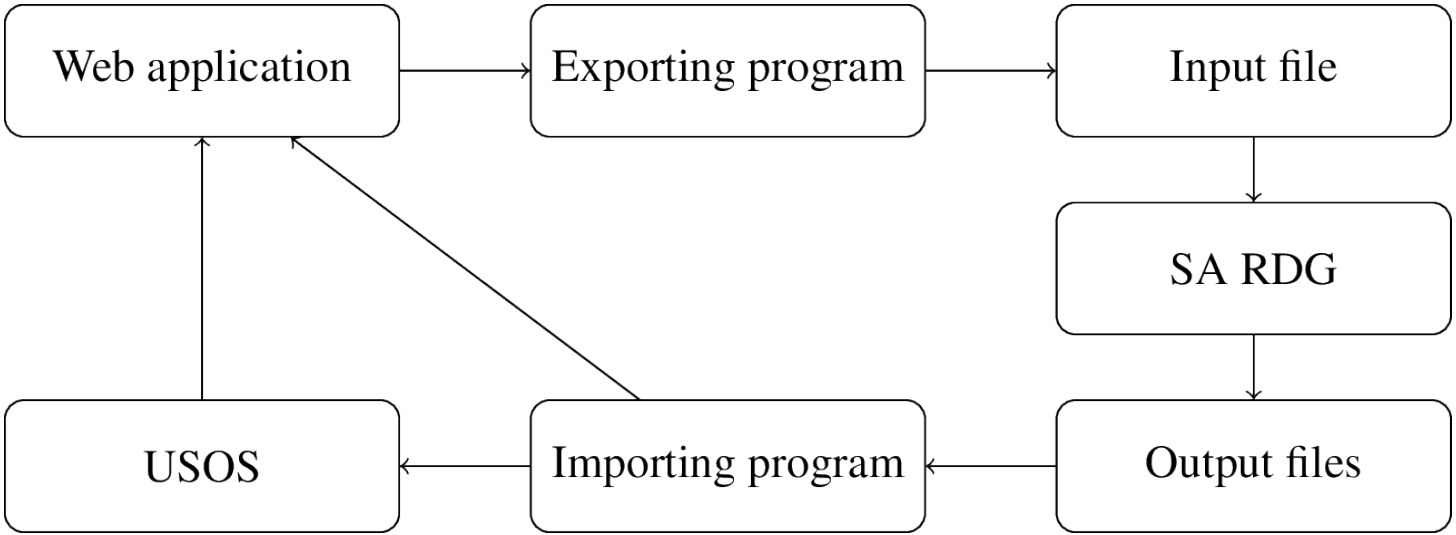
```
for all students:
    for all ordered preferences of current student:
        if matching preference does not exceed limits:
            assign student using preference
            take next student
    register student to least crowded classes without
            breaking exclusions
```

# Unsuccessful - monolithic app

1. Java based application
2. Automatic memory management
3. Direct connection to database
4. User friendly interface

*Bugs, bugs, bugs. Hard to debug (20 instances)*

# Successful - application architecture




SA RDG (engine) - command line app in C++

# Successful solution (SA)

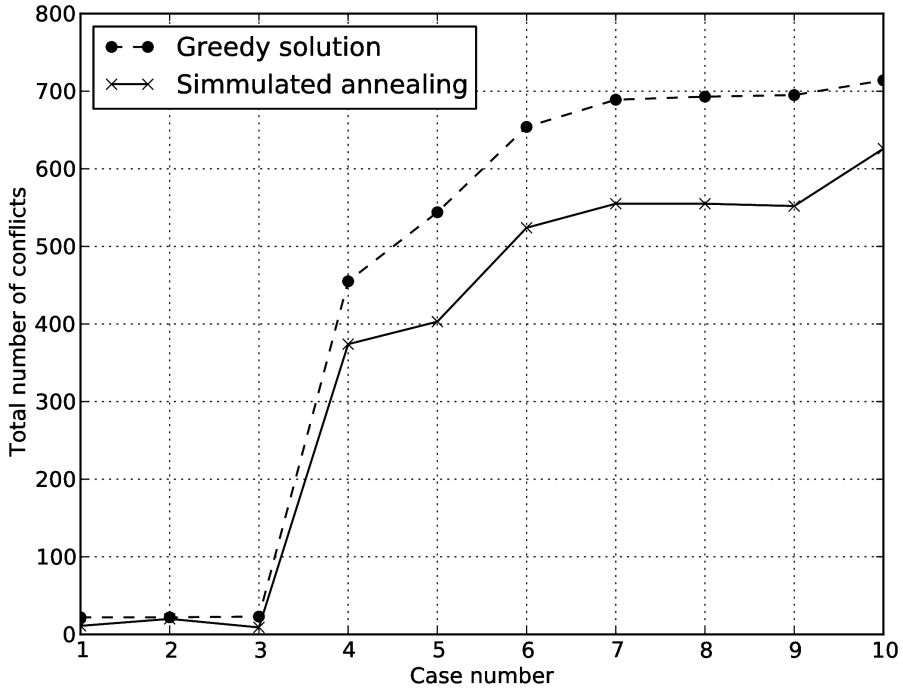
```
assign students randomly to classes
while we still have time:
    move random student from one class to another
    if move improves global happiness (*):
        commit move
    else:
        rollback move

(*) at the beginning we accept some not improving moves
```

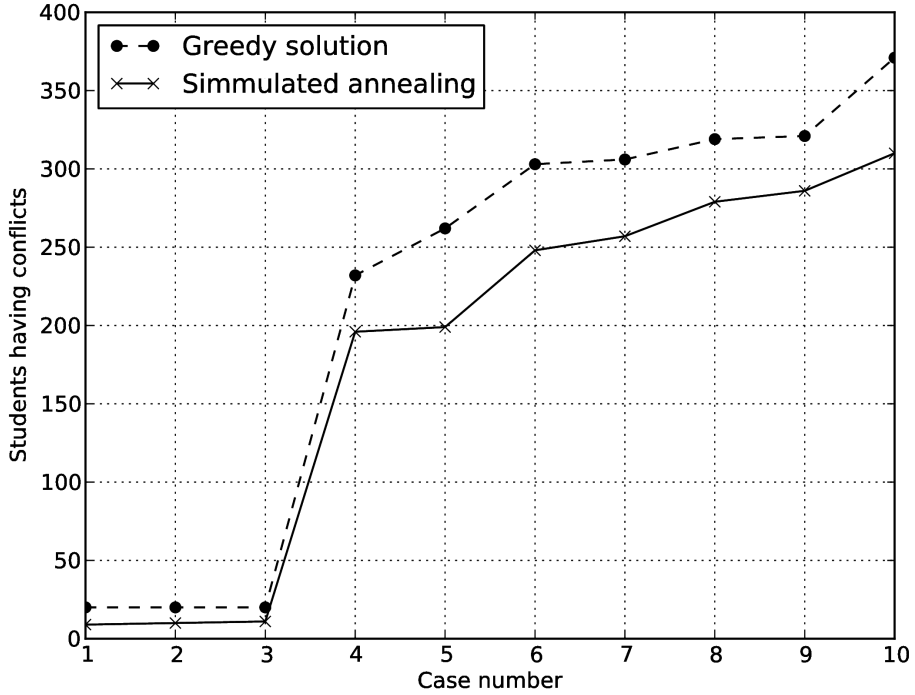
# Penalty

- number of broken class exclusions (w: 4)
- number of broken class limits penalties (w: 2)
- number of all conflicts (w:1)
- sum of met students priorities (w:0.2)

# Results - no. of conflicts



# Results - students having conflicts



# USOS - other optimizations

1. Class-course assignment
   a. global penalty
   b. penalty based on students ranking
2. Class exchange
   a. global penalty
   b. penalty based on students ranking
3. 5 minute rounds - exams registration

- Thank you!