THE REPUBLIC OF TURKEY
BAHÇEŞEHİR UNIVERSITY

CONFERENCE SESSION SCHEDULING USING MODIFIED ANT COLONY ALGORITHM

Master Thesis

EFE AÇIKGÖZ

İSTANBUL, 2013

THE REPUBLIC OF TURKEY
BAHÇEŞEHİR UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

INFORMATION TECHNOLOGY

CONFERENCE SESSION SCHEDULING USING MODIFIED ANT COLONY ALGORITHM

Master Thesis

EFE AÇIKGÖZ

Supervisor: Prof.Dr. Adem Karahoca

İSTANBUL, 2013

THE REPUBLIC OF TURKEY
BAHÇEŞEHIR UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
INFORMATION TECHNOLOGY

Name of the thesis: Conference Session Scheduling using Modified Ant Colony Algorithm
Name/Last Name of the Student:Efe Açıkgöz
Date of the Defense of Thesis: 12 September 2013

The thesis has been approved by the Graduate School of ____________.

ASSOC.PROF.DR.TUNÇ BOZBURA
Graduate School Director

I certify that this thesis meets all the requirements as a thesis for the degree of Master of Arts.

Prof.Dr. Adem KARAHOCA
Program Coordinator

This is to certify that we have read this thesis and we find it fully adequate in scope, quality and content, as a thesis for the degree of Master of Arts.

Examining Comittee Members	Signature
Thesis Supervisor Prof.Dr. Adem KARAHOCA	----------------------------------
Member Asst.Prof.Dr. M.Alper TUNGA	---------------------------------
Member Asst.Prof.Dr. Yalçın ÇEKİÇ	----------------------------------

DEDICATION

I dedicate this thesis to my teacher, Adem Karahoca, who has been helping me since the graduate school. It is his tireless efforts that motivated me to set higher targets. I also dedicate this thesis to my cousin Sarp Erdag who always helped me aim learning new techniques and technologies.

ABSTRACT

Conference Session Scheduling using Modified Ant Colony Algorithm

Efe Açıkgöz

Information Technology

Thesis Supervisor: Prof.Dr. Adem Karahoca

July 2013, 40 pages

This study deals with the issue of automatizing the session schedules of conferences. One of the biggest concerns for many conferences is time management considering there are hundreds of presentations and doing the session scheduling manually is a time consuming and challenging operation.
In planning the conference program, you can create an almost infinite number of possible designs. This is an important process because even high-quality sessions can lose their value if the program is not properly planned.

To calculate the possible presentation schedules, we started using ant colony algorithm and successfully implemented the algorithm, which was successful for building session schedules but was lacking the performance and flexibility. To fight the performance issues the ant colony algorithm was modified, and to give it flexibility, some of the properties of genetic algorithm was imitated.

In the end, both increasing the performance and flexibility of the generic ant colony algorithm was making it a bit more specific and focused.

Keywords: Session Scheduling, Ant Colony Algorithm, Genetic Algorithm

ÖZET

GÜNCELLENMİŞ KARINCA KOLONİSİ ALGORİTMASI İLE KONFERANS OTURUMLARININ PLANLANMASI

Efe Açıkgöz
Bilgi Teknolojileri
Tez Danışmanı: Prof.Dr. Adem Karahoca

Temmuz 2013, 40 sayfa

Bu tez çalışmasında, konferans sunumlarını planlamaya yardımcı olabilecek güncellenmiş karınca kolonisi algoritması kullanan bir yazılım geliştirilmiştir. Konferanslarda en büyük problemlerden biri zaman yönetimidir ve yüzlerce sunumun programını el ile oluşturmak çok zaman alıcı ve zorlayıcı bir görevdir. Konferans programı planlarken neredeyse sonsuz sayıda olası tasarım bulunmaktadır. Program planlaması çok önemli bir konudur çünkü en yüksek kaliteli sunumlar bile kötü programlama dolayısı ile değerini yitirebilirler.

Olası sunum oturumlarını hesaplayabilmek için ilk olarak karınca başarılı bir şekilde kullanılmıştır. Karınca kolonisi algoritması, sunum programı oluşturma konusunda başarılı idi, fakat performans ve esneklik konularında başarısızdı. Performans problemlerinden kurtulmak için karınca kolonisi algoritması güncelledik ve esneklik katmak için genetik algoritmasının bazı özellikleri örnek alındı.

Sonuç olarak genel karınca kolonisi algoritmasına göre hem performansı hem de esnekliği daha yüksek spesifik ve odaklı bir sonuç elde edildi.

Anahtar Kelimeler: konferans oturum planlama, karınca sürüsü algoritması, genetik algoritması

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVATIONS

ACA	:	Ant Colony Algorithm
GE	:	Genetic Algorithm
DB	:	Database
UI	:	User Interface
UX	:	User Experience
ERD	:	Entity-relationship diagram
WLAN	:	Wireless local area network
CAPS	:	Controlled Access Phase Scheduling
NEMO	:	Network Mobility
CASA	:	Community-aware scheduling algorithm

1. INTRODUCTION

Scheduling sessions at a large conference is a persistent challenge. The problem is, given a large number of sessions, rooms, time slots and constraints, it is required to mine for the best schedule that contains all sessions within the given time slots while satisfying the set of constraints. This is really hard to achieve manually (2011 World Statistics Congress included approximately 1150 oral and 250 poster presentations, with as many as 20 sessions running in parallel at any one time)

To eliminate this problem, we decided to implement an algorithm to check possible session schedules while satisfying the constraints. After some research, we decided to use Ant Colony Algorithm. The reason we picked ACA is because it is a fairly efficient probabilistic technique for finding paths through graphs.

Ant colony algorithm is based on the behavior of ants seeking a path between their colony and a source of food. Ant colony optimization algorithms have been applied to many optimization problems ranging from quadratic assignment to routing vehicles and a lot of derived methods have been adapted to dynamic problems in real variables, multi-targets and parallel implementations. It has also been used to produce near-optimal solutions to the travelling salesman problem.

Although the ant colony algorithm was successful for creating session schedules, it was not very effective performance-wise. To fight this issue, we had to modify algorithm, making it more specific and focused at the problem at hand.

Even after the performance modifications, the algorithm was not flexible enough for specific issues like handling constraints, certain presenters might had, which was fixed by merging ACA with other algorithms like Genetic Algorithm.

2. LITERATURE REVIEW

2.1 SESSION SCHEDULING ALGORITHMS

A lot of possible algorithms have been evaluated for our conference session scheduling task, comparing their strengths, weaknesses, and effectiveness, to select the appropriate algorithm. So far, mostly Genetic Algorithm, and cluster analysis have been used for conference session scheduling. Houlding and Haslett used a hybrid cluster analysis to do the session scheduling in their paper "Scheduling Parallel Conference Sessions: An Application of a Hybrid Clustering Algorithm for Constrained Cardinality" (Houlding and Haslett, 2011). In the paper they stated that they needed to cluster similar topics since it would not be appropriate to hold similar sessions at the same time but in different venues, making the audience for those sessions split

"If sessions are run in parallel, then it would not be appropriate to hold similar sessions at the same time but in different venues, as the likely audience for those sessions would be split. Such problems are further expounded if a conference focuses on a general discipline that has many diverse elements, e.g., statistics, which has elements of theoretical statistical development, applied statistical analysis, collection of data, formulation of national statistics, and the use of statistics in industry"

They approach to the issue by using constraints on the cardinality of the clusters directly within the steps of a hybrid analysis, avoiding sensitivity to the random initialization of the clustering process. Although they have a simply approach to the issue, the solution might not be optimal.

Burns and Wellings also tried to achieve a fixed priority scheduling that is proposed to be applied to tasks that communicate only asynchronously where sessions are abstract constructs and can be used to represent presentations as well as conversations, atomic actions and such. Their paper "Synchronous sessions and fixed priority scheduling" (Burns, Wellings 1996) defines a means to incorporating synchronous behavior into the computational model of fixed priority scheduling. Ceiling priorities are assigned to sessions and appropriate scheduling analysis is derived.

The paper also describes how all tasks involved in a session must be released at the same time. Tasks may block within a session but will otherwise execute asynchronously. A ceiling priority protocol for sessions was introduced that allows blocking time to be calculated and hence effective response time analysis can be achieved. The more sessions a set of tasks involves themselves in, the more their completions time will be similar.

Thompson also wrote how effective scheduling can increase the ability of meeting participants to attend their preferred sessions. In his paper “Improving conferences through session scheduling” (Thompson 2002) Thompson reports the results of two actual conferences used to experiment for comparing approaches to conference scheduling. The results indicate that participants’ ability to attend their preferred sessions is no higher in manually scheduled conferences that take into account participants’ preferences that what one would obtain by simply generating random schedules without taking into account participants’ meeting-selection preferences.

There is also the blog post of Swanson in 2008 named “Conference Session Scheduling using a Genetic Algorithm” where Swanson proposes an algorithm with constrains like:

a) Only one session can be presented per room during any given time slot.
b) A speaker can only present one session during any given time slot (i.e. can not be in two places at the same time).
c) A speaker may only be available on specific days.
d) A session may require audio/video equipment that is only present in specific rooms.
e) Popular sessions should be scheduled in larger rooms.

Which he tries solving by grouping sessions into tracks, and scheduling them in parallel almost like mini conferences running alongside each other, even though it is rare for someone to sit through all sessions in a single track which he tries to fix by taking extra factors into account like

i. The total walking distance required for a participant to attend all of their favorite sessions. The algorithm prefers shorter routes.
ii. The degree to which room capacities are "balanced." This means that—on average—the algorithm prefers schedules that leave relatively equal space in each room. Otherwise, one room may be near 100% capacity while another is only at 25%.

In his version of the algorithm, each solution in the population represents a conference schedule. The fitness function takes all of the aforementioned factors into account, and penalizes solutions with undesirable attributes. At the end of each generation, an elite group of solutions is retained, and the remainders are subject to both crossover and mutation.

There is also another blog post by Tarhini in 2012 "Genetic Algorithm for Conference Session Scheduling", where he Tarhini proposes an algorithm with sessions, rooms, timeslots and set of preference sessions to implement a "survival of the fittest" algorithm which means that solutions are generated, and bad solutions are eliminated and good solutions are carried over to next step, while using a fitness function to validate a given solution with the set of constraints and list of preference sessions to return a value indicating how relevant the solution is, which is an interesting approach

2.2 OTHER ALGORITHMS

Even though there were quite a few algorithms that can be specifically used for conference session scheduling, we checked other algorithms with capability to be used in conference session scheduling. Naime and Taherinejad in 2008 wrote a paper "New robust and efficient ant colony algorithms: Using new interpretation of local updating process" proposing two new ant colony algorithms by updating the ant colony algorithm by adding local updating rules. They go from start to end point of a tour while the ants' freedom to make local changes on links is gradually restricted. This idea is implemented in two different forms, leaving two new algorithms, KCC-Ants and ELU-Ants. To evaluate the new algorithms, they run them along with the old one on the standard TSP library where in almost all the cases the proposed algorithms had better solutions.

Huang, Bessis, Norrington, Kuonen, Hirshbrunner also wrote a paper talking about job scheduling strategies and how new technologies like grid and cloud, metascheduling can be an important scheduling pattern since it is responsible for orchestrating resources managed by independent local schedulers("Exploring decentralized dynamic scheduling for grids and clouds using the community-aware scheduling algorithm"). To overcome issues such as bottleneck, single point failure and impractical unique administrative management are normally led by conventional centralized or hierarchical schemes, and how decentralized scheduling scheme can be a promising approach because of its capability with regards to scalability and flexibility. They introduce a decentralized dynamic scheduling approach entitled the "Community-aware scheduling algorithm (CASA)", a two phase scheduling solution comprised of a set of heuristic sub-algorithms to achieve optimized scheduling performance over the scope of overall grid or cloud. They also use a real grid workload trace dataset and show how centralized scheduling scheme with the use of CASA leads to a 30-61 percent better average job slowdown and a 68-86 percent shorter average job waiting time in a decentralized scheduling manner.

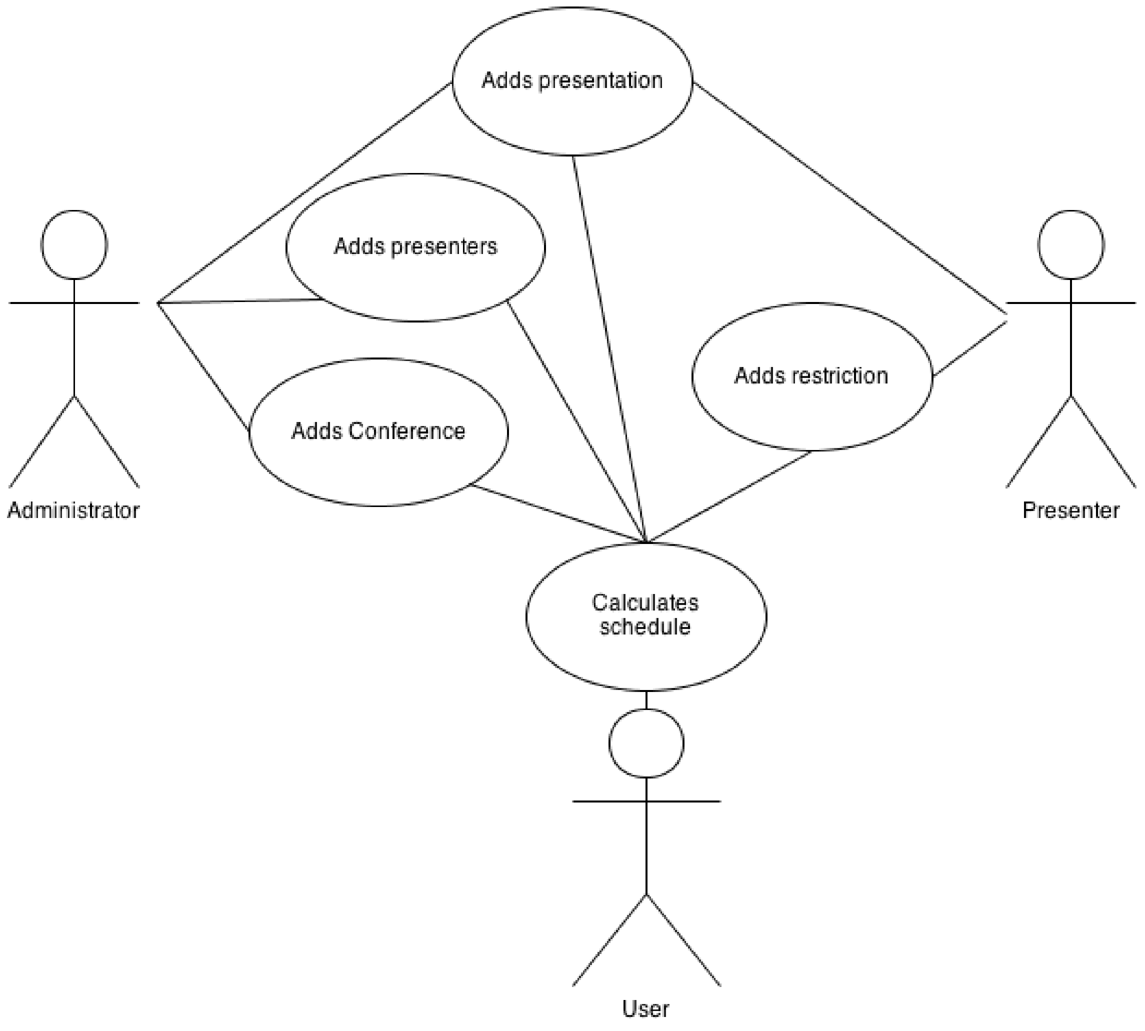
There is also another paper named "Analysis of temporal and throughput fair scheduling in multirate WLANs" (Alnuweiri, Fallah 2008) where the authors talks about how the Wireless Local Area Network (WLAN) provides controlled access features that can be used in conjunction with scheduling algorithms to provide guaranteed per-session services. They also talk about how the multirate operation of the WLAN complicates the design of scheduling and how Controlled Access Phase Scheduling (CAPS) can achieve guaranteed fair services in WLAN. They present a modified start time fair queuing based scheduler as their choice and analyze its performance under dynamic and static multirate operations. Wang and Tofozolli also aimed to find a solution to the problem about network mobility supporting a network moving as a whole and it causing the bandwidth on its wireless link varying with time and locations in their paper "Performance Comparison of scheduling algorithms in network mobility environment". The frequent bandwidth fluctuation makes the resource reservation and admission control lack of scalability. They propose a solution by using scheduling algorithms to optimize the resource distribution based on the varying available bandwidth. They compare the performance of well known queuing algorithms and their advantages and disadvantages in the NEMO environment. The tests show that the Adaptive Rotation priority Queue (which operates with a priority first fairness second policy) outperforms all existing scheduling algorithms in mobile networks whose capabilities are time-varying and location-dependent.

Even though there were quite a few articles and papers where genetic algorithm used for session scheduling, the paper "Genetic algorithms and neural networks: optimizing connections and connectivity (Whitley, Starkweather, Bogart 1990) is an important one since in this paper the authors propose that the genetic algorithms need not to search along the contours of the function being optimized instead using selective reproduction and recombination of binary strings and changing the sampling rate of hyperplanes in the search space so as to reflect the average fitness of strings that reside in any particular hyperplane. They give an overview of several different experiments applying genetic algorithms to neural networks and optimizing the weighted connections in feed forward neural networks using both binary and real-valued representations and using genetic algorithm to discover novel architectures in the form of connectivity patterns for neural networks that learn using error propagation. They also talk about the future applications in neural network optimization in which genetic algorithm can play a significant role.

3. DATA AND METHODOLOGY

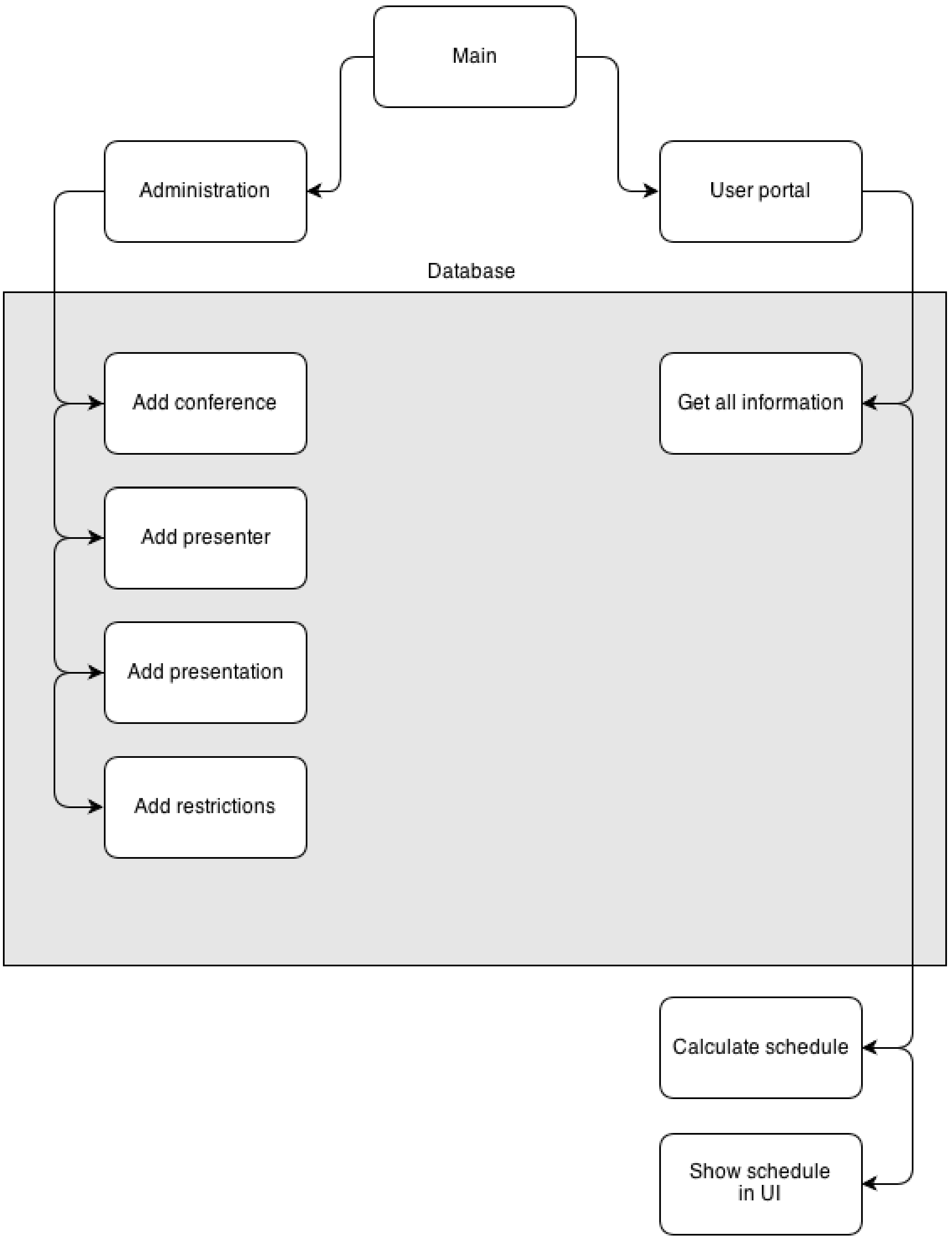
To calculate the scope of the project, we started to research similar researches and gathered what kinds of designs their projects had. From the information gathered, first use case diagram, data flow diagram and block diagram have been created. Then using that information we created our models and created test data. After research, we created our use case diagram to represent the user interaction with the system

Figure 3.1 – Use case diagram



As the use case diagram shows, we have an administrator who adds conferences, presenters and presentations. Then we have a presenter who adds restrictions and at last we have a user that uses all those information to calculate the schedules.

Figure 3.2 – Block diagram

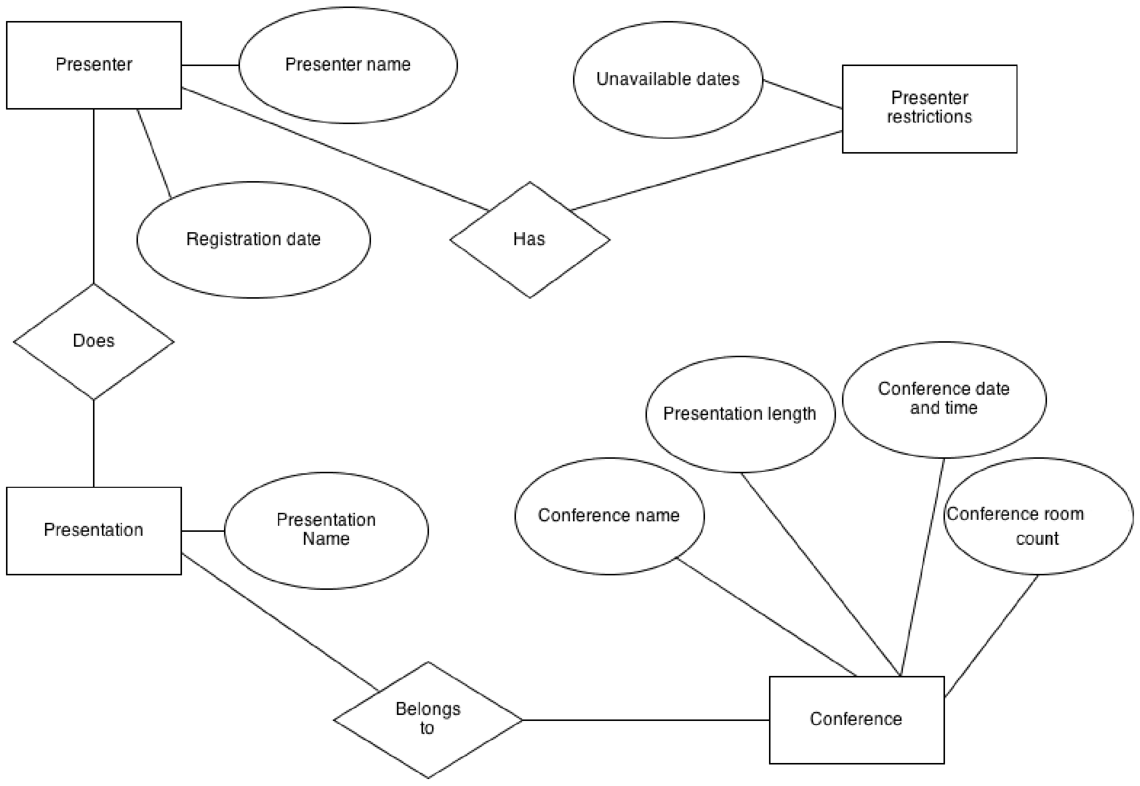


Block diagram above shows the application is designed and how components are related to each other

3.1 DATA

For data, random names for presenters, random presentations for those presenters, constraints and different conferences for the presentations have been generated; following the model-view-controller architecture.

Figure 3.1.1 – Entity-relationship diagram



ERD above shows how the models of the application have been designed. We have a presenter that has presenter name and registration date; presenter has restrictions which has dates showing when the presenter is unavailable. Presenters are also connected to presentation, which has presentation name. Presentation also belongs to conference. A conference has a conference name, a presentation length, conference date and time, and conference rooms count

Different numbers of presentations for different conferences to test the performance of the algorithm with different pool sizes have been generated.

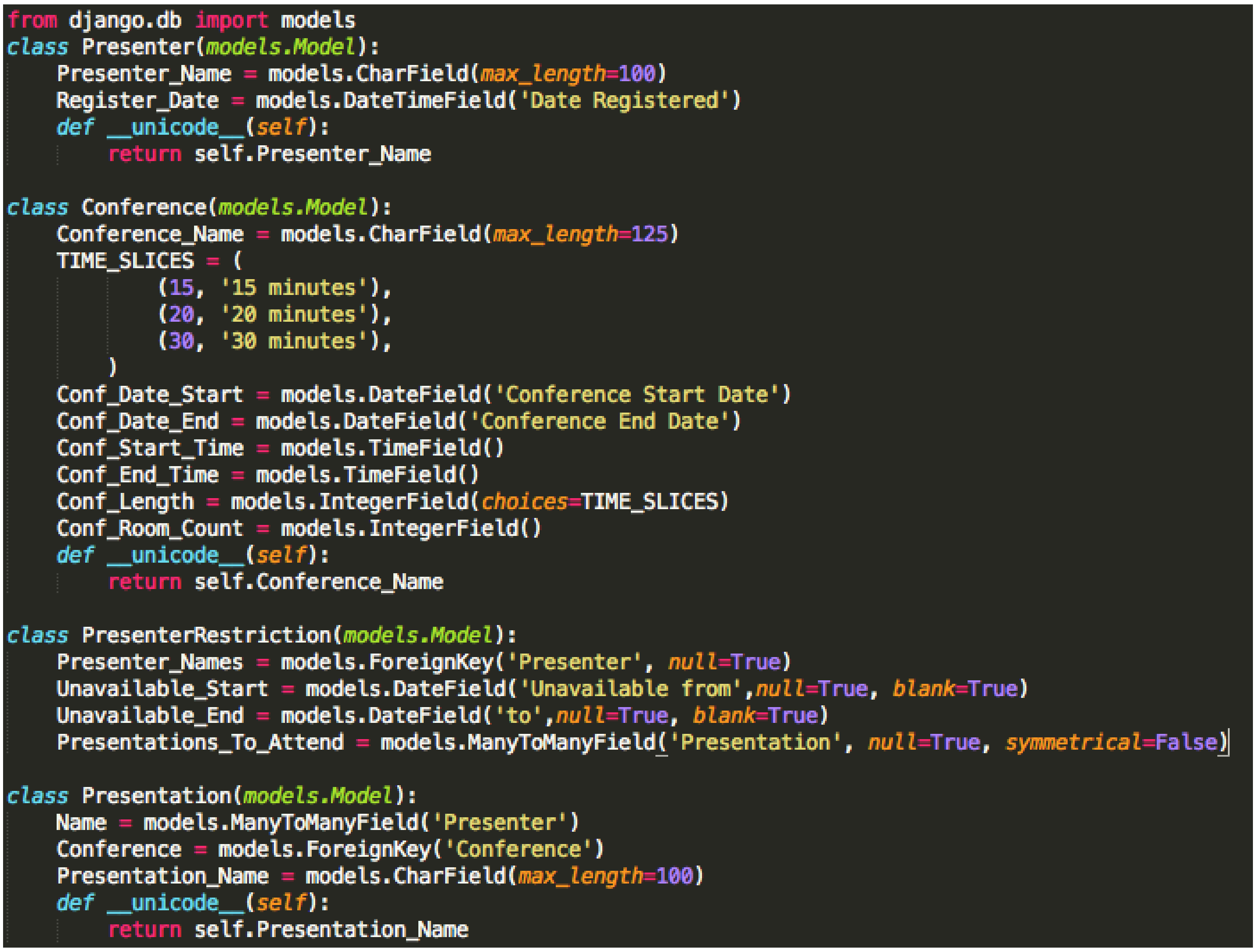
We started small with 42 Presentations in 3 days, 30 minute per presentation, 2 rooms simultaneously between 10:00-14:30

And then pool size has been increased to 90 Presentations in 3 days, 30 minute per presentation, 3 rooms simultaneously between 10:00-16:00 then 160 presentations and so on.

We tested every different pool size for a few hundred times to get an average time spent to calculate the results and how many iterations needed to calculate those results.

Database in Django is not directly accessible; you can either use administration page or view controllers to add/remove/get data from the database.

Figure 3.1.2 – Models of the project



```
from django.db import models
class Presenter(models.Model):
    Presenter_Name = models.CharField(max_length=100)
    Register_Date = models.DateTimeField('Date Registered')
    def __unicode__(self):
        return self.Presenter_Name

class Conference(models.Model):
    Conference_Name = models.CharField(max_length=125)
    TIME_SLICES = (
            (15, '15 minutes'),
            (20, '20 minutes'),
            (30, '30 minutes'),
        )
    Conf_Date_Start = models.DateField('Conference Start Date')
    Conf_Date_End = models.DateField('Conference End Date')
    Conf_Start_Time = models.TimeField()
    Conf_End_Time = models.TimeField()
    Conf_Length = models.IntegerField(choices=TIME_SLICES)
    Conf_Room_Count = models.IntegerField()
    def __unicode__(self):
        return self.Conference_Name

class PresenterRestriction(models.Model):
    Presenter_Names = models.ForeignKey('Presenter', null=True)
    Unavailable_Start = models.DateField('Unavailable from',null=True, blank=True)
    Unavailable_End = models.DateField('to',null=True, blank=True)
    Presentations_To_Attend = models.ManyToManyField('Presentation', null=True, symmetrical=False)

class Presentation(models.Model):
    Name = models.ManyToManyField('Presenter')
    Conference = models.ForeignKey('Conference')
    Presentation_Name = models.CharField(max_length=100)
    def __unicode__(self):
        return self.Presentation_Name
```

We also tested cases with more spots than presentations like a conference with 200 open spots and 50 presentations, which has better performance with initial algorithm but has no visible increase with latest changes to the algorithm.

3.2 METHODOLOGY

We started working on the thesis by researching potential algorithms for session scheduling. After enough information have been gathered on algorithms and methods like simplex method, Ant Colony Algorithm, Genetic Algorithm, we started researching previous works and papers on this subject. Even though there are very few works on this specific problem, they all used Genetic Algorithm and there were no researches about ACA with session scheduling. After picking algorithm too, we started researching other works done with ACA on other problems to see its downsides, upsides, and possible methods to implement the algorithm as well as its effectiveness and implementation difficulty.

After my researches were done, we picked Python for my programming language and Django framework to build a server / client type application.

We also used JQuery and Twitter Bootstrap to help create the user interface, Postgre SQL for DB.

3.2.1 ANT COLONY ALGORITHM

Ant colony optimization algorithm first proposed by Marco Dorigo in 1992 aiming to search for an optimal path in a graph, based on the behavior of ants seeking a path between their colony and a source of food.

In the natural world, ants initially wander randomly and upon finding food return to their colony leaving behind pheromone trails. If other ants find these pheromones, they will follow these pheromone paths instead of wandering randomly.

Over time, these pheromone trails start to evaporate reducing its attractive strength. The longer it takes to travel down a path, the longer pheromones evaporate, thus making long paths less attractive to ants than shorter paths.

In short, when one ant finds a good path from the colony to a food source, other ants are more likely to follow that path. The idea of the ant colony is to mimic this behavior.

At each stage, the ant chooses to move from one city to another according to some rules:

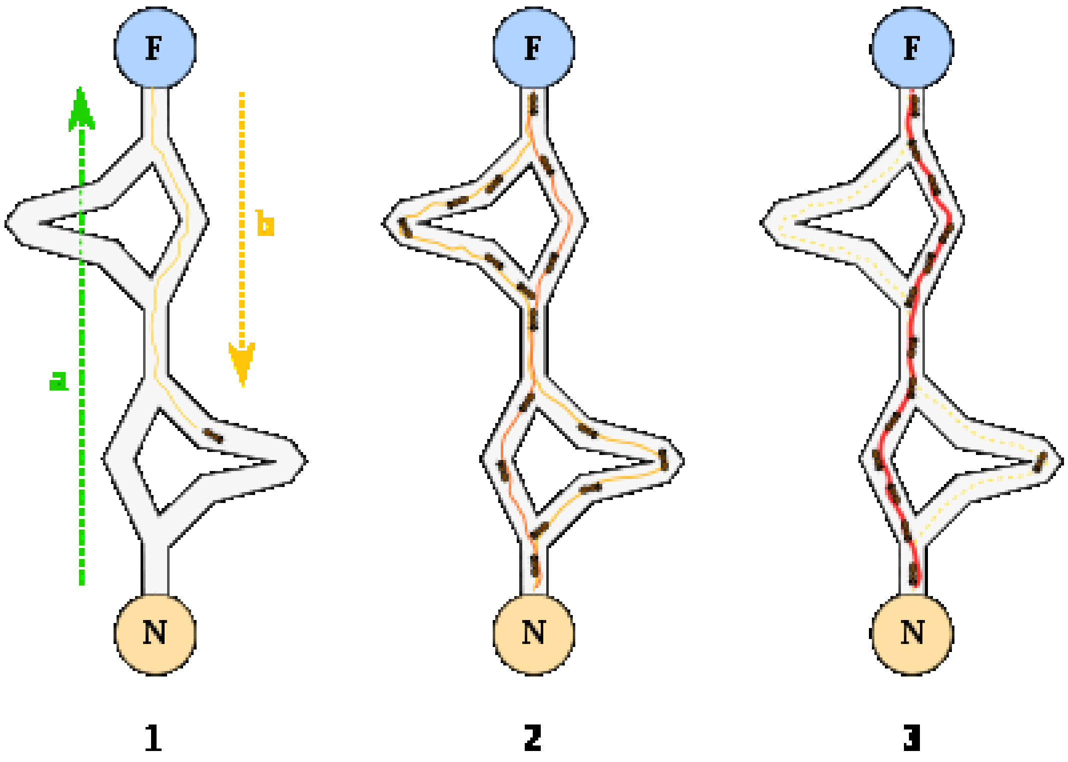
1. It must visit each city exactly once;
2. A distant city has less chance of being chosen (the visibility);
3. The more intense the pheromone trail laid out on an edge between two cities, the greater the probability that that edge will be chosen;
4. Having completed its journey, the ant deposits more pheromones on all edges it traversed, if the journey is short;
5. After each iteration, trails of pheromones evaporate.

We checked a few of the common extensions of ant colony optimization algorithms, like elitist ant system where the global best solution deposits pheromone on every iteration, max-min ant system where a maximum and minimum pheromone amounts are added and only global best or iteration best tour deposits pheromone, rank based ant system where all solutions are ranked according to their length, recursive ant colony optimization where there is a recursive form of ant system which runs nested ant systems to increase precision of output. We ended up using ant colony system.

In order to make ant colony algorithm work for session scheduling, we needed to represent the presentations on a graph and calculate a path without any blocks on graph. So if we think presentations as possible paths, and constraints as roadblocks, the algorithm works and it calculates possible session schedules.

Also Ant Colony Algorithm has one of the biggest advantages is that it has multiple threading (multiple ants searching for best path to food), we can search for possible paths relatively faster than other algorithms. Whereas most other algorithms depend on single threading, multiple threading is the main property of Ant Colony Algorithm

Figure 3.2.1 – Ant colony path finding example (F = Food, N = Nest) (online http://pagmo.sourceforge.net/pagmo/ant.png)



3.2.1.1 Example ant colony algorithm

```
while(not_termination)
  generateSolutions()
  edgeSelection()
  pheromoneUpdate()
end while
```

Edge selection

An ant is a simple computational agent in the ant colony optimization algorithm. It iteratively constructs a solution for the problem at hand. The intermediate solutions are referred to as solution states. At each iteration of the algorithm, each ant moves from a state x to state y, corresponding to a more complete intermediate solution. Thus, each ant k computes a set $A_k(x)$ of feasible expansions to its current state in each iteration, and moves to one of these in probability. For ant k, the probability $p^k_{x,y}$ of moving from state x to state y depends on the combination of two values, viz.,
the *attractiveness* $\eta_{x.y}$ of the move, as computed by some heuristic indicating the *a priori* desirability of that move and the *trail level* $T_{x,y}$ of the move, indicating how proficient it has been in the past to make that particular move.

The *trail level* represents a posteriori indication of the desirability of that move. Trails are updated usually when all ants have completed their solution, increasing or decreasing the level of trails corresponding to moves that were part of "good" or "bad" solutions, respectively.

In general, the kth ant moves from state x to state y with probability

$$P_{x,y} = (T^{\alpha}_{x,y})(\eta^{B}_{x,y}) / \mathrm{SUM}(T^{\alpha}_{x,y})(\eta^{B}_{x.y}) \qquad \textbf{(3.2.1.1)}$$

Where $T_{i,j}$ is the amount of pheromone deposited for transition from state i to j, $0 \leq \alpha$ is a parameter to control the influence of $T_{i,j}$, $\eta_{i,j}$ is the desirability of state transition i.j (*a priori* knowledge, typically $1/d_{x.y}$, where d is the distance) and $B \geq 1$ is a parameter to control the influence of $\eta_{i,j}$. $T_{i,j}$ and $\eta^{B}_{i,j}$ represent the attractiveness and trail level for the other possible state transitions.

Pheromone update

When all the ants have completed a solution, the trails are updated by

$$T_{x,y} = (1\text{-}p)T_{x,y} + \Delta\, T^{k}_{x,y} \qquad \textbf{(3.2.1.2)}$$

where $T_{x,y}$ is the amount of pheromone deposited for a state transition x,y, p is the *pheromone evaporation coefficient* and $\Delta\, T^{k}_{x,y}$ is the amount of pheromone deposited by kth ant, typically given for a travelling salesman problem (with moves corresponding to arcs of the graph) by

$\Delta T^{k}_{x,y} = 1/L_k$ if ant k travels on edge I,j

$\Delta T^{k}_{x,y} = 0$ otherwise **(3.2.1.3)**

where L_k is the cost of the kth ant's tour (typically length) .

3.2.2 PYTHON AND DJANGO

Python is a general-purpose high-level programming language. The main reason we picked Python for this thesis was to learn a new programming language. Python emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a multi-paradigm object-oriented programming and structured programmings are fully supported and there are a number of language features which support functional programming and aspect-oriented programming (metaprogramming). Many other paradigms are supported using extensions including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and cycle-detecting garbage collector for memory management which helped with memory problems with algorithm modifications where we decrease available spots array size. An important feature of Python is dynamic name resolution which binds method and variable names during program execution.

We did not want to make the project a console application so we also started learning Django web framework, which is a free and open source web application framework. It is written in Python and follows the model-view-controller architectural pattern. Primary goal of Django is to ease the creation of complex, database-driven websites. Django emphasizes reusability and pluggability of components, rapid development, and the principle of do not repeat yourself (A principle we try to use with every programming language). Python is used throughout even for settings files and data models.

We also used scripting library JQuery for client-side scripting, Twitter Bootstrap for dynamic user interface. JQuery is a multi-browser JavaScript library designed to simplify the client-side scripting of HTML. Syntax of JQuery is designed to make it easier to navigate a document, select DOM (Document object model) elements and create animations, handle events and develop Ajax applications. Twitter Bootstrap is a free collection of tools for creating websites and web applications. It contains HTML and CSS based design templates for typography, forms, buttons, charts, navigation and other interface components as well as optional JavaScript extensions. Bootstrap is modular and consists essentially of a series of LESS Stylesheets that implement the various components of the toolkit. A stylesheet called bootstrap.less includes the components stylesheets. Developers can adapt the Bootstrap file itself, selecting the components they wish to use in their project.

Adjustments are possible to a limited extent through a central configuration stylesheet. More profound changes are possible by the LESS declarations.

The use of LESS stylesheet language allows the use of variables, functions and operators, nested selectors, as well as so-called mixins.

We are using Twitter Bootstrap to fit rooms to the interface using spans

```
<div class="span{{ span }}">
```

3.2.3 GENETIC ALGORITHM

A genetic algorithm is a search heuristic that mimics the process of natural evolution. This heuristic is routinely used to generate useful solutions to optimizations and search problems. In a genetic algorithm a population of candidate solutions are evolved toward better solutions.

The evolution usually starts from a population of randomly generated individuals and an iterative process, with the population in each iteration called a generation and in each generation; the fitness of every individual in the population is evaluated. The fitness means the value of the objective function in the optimization problem being solved or simply; given a solution with the set of constraints and list of preference sessions fitness returns a value indicating how relevant the solution is, The more fit individuals are selected from current population and each individual has a genome that is modified to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. The algorithm usually terminates when either a maximum number of generations have been produced or a satisfactory fitness level has been reached for the population.

Initially many individual solutions are (usually) randomly generated to form an initial population. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Traditionally, the population is generated randomly, allowing the entire range of possible solutions (the *search space*). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found. The fitness function is defined over the genetic representation and measures the *quality* of the represented solution. The fitness function is always problem dependent.

During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process where fitter solutions (calculated from fitness function) are typically more likely to be selected to breed. Certain selection methods rate the fitness of each solution and preferentially select the best solutions.

3.2.4 MODIFICATIONS TO THE ALGORITHM

We initially implemented the ant colony algorithm and was able to create session schedules with no problem, but even without presentation constraints, the performance was lacking (More than 200 iteration average to calculate a 42 presentation conference with 0 constraints). Initial algorithm follows ant colony algorithm with no modifications on it.

It should also be kept in mind that a presentation might have more than 1 presenter. So when filling presentation spots we have to check if the presenters of the presentation has any other presentation at another room.

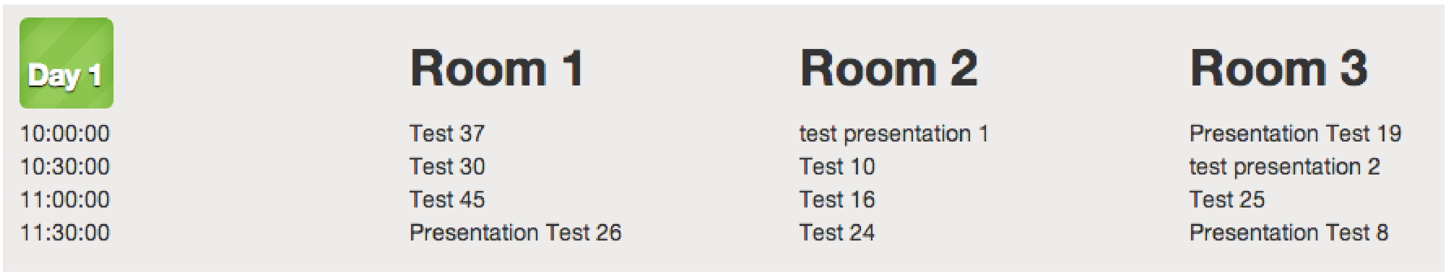
There is also the case with presenters having more than one presentation in a row. If we assume a presentation takes 20 minutes a presenter has a presentation in room A at 14:00, he cannot have another presentation at room B at 14:20, but he can have another one in the room he is in which is room A.

To make sure this worked, we first implemented grouping on presenters where all presentations of a presenters were in a row with 1 empty spot between them to make sure the case above does not occur, but this caused quite a lot of problems, like day ending before filling all presentations of the presenter, or simply the presenter having more presentation than the number of available half the available spots in a day (1 for presentation 1 for travelling). When that approach did not work, we went for an “if check” to see if the presenter had another presentation on an earlier session and rolled for another spot if he does, which also had downsides like only available spots left were those unavailable to presenter.

To fight the issue with multiple presenters, we tried to modify the algorithm so that all presentations of the presenters were grouped in a sequence, one after another, but decided to scrap the idea since it might not be convenient to presenters.
The pseudocode we used was:

```
1-  leastResistance = x
2-  while currentResistance<leastResistance
3-    foreach presentation in presentationList
4-      while spot not suitable
5-        checkPheromones()
6-        pickSpotFromConference()
7-        checkIfSpotSuitable()
8-        updatePheromones()
9-      end while
10-   end foreach
11-   calculateCurrentResistance()
12- end while
```

Figure 3.2.4.1 – Example filled spots



Day 1	Room 1	Room 2	Room 3
10:00:00	Test 37	test presentation 1	Presentation Test 19
10:30:00	Test 30	Test 10	test presentation 2
11:00:00	Test 45	Test 16	Test 25
11:30:00	Presentation Test 26	Test 24	Presentation Test 8

checkPheromones function checks if there are pheromones on possible spots to help decide picking a spot
pickSpotFromConference picks a spot for the presentation
checkIfSpotSuitable checks if the picked spot empty, or if there are any constraints about the spot picked
updatePheromones updates the pheromone or decays it depending on the function above
calculateCurrentResistance() calculates the current resistance of the schedule

Resistance value tells us how hard it was to fill the current session, or simply, how hard it was to fill all spots. The less resistance it has, the easier and faster it fills the session.

Most effective results we got with this initial algorithm were when we used conferences with more spots than presentations (ex: a conference with 200 empty spots and 100 presentations).

Although this algorithm works like this, it is still pretty general and we can increase performance by making it a bit more specifically for session scheduling. First of all, we know the possible paths, so do we have to randomly pick a spot and check if said spot is available? Since we have all empty spots, we do not, so we changed the algorithm and got rid of the iteration.

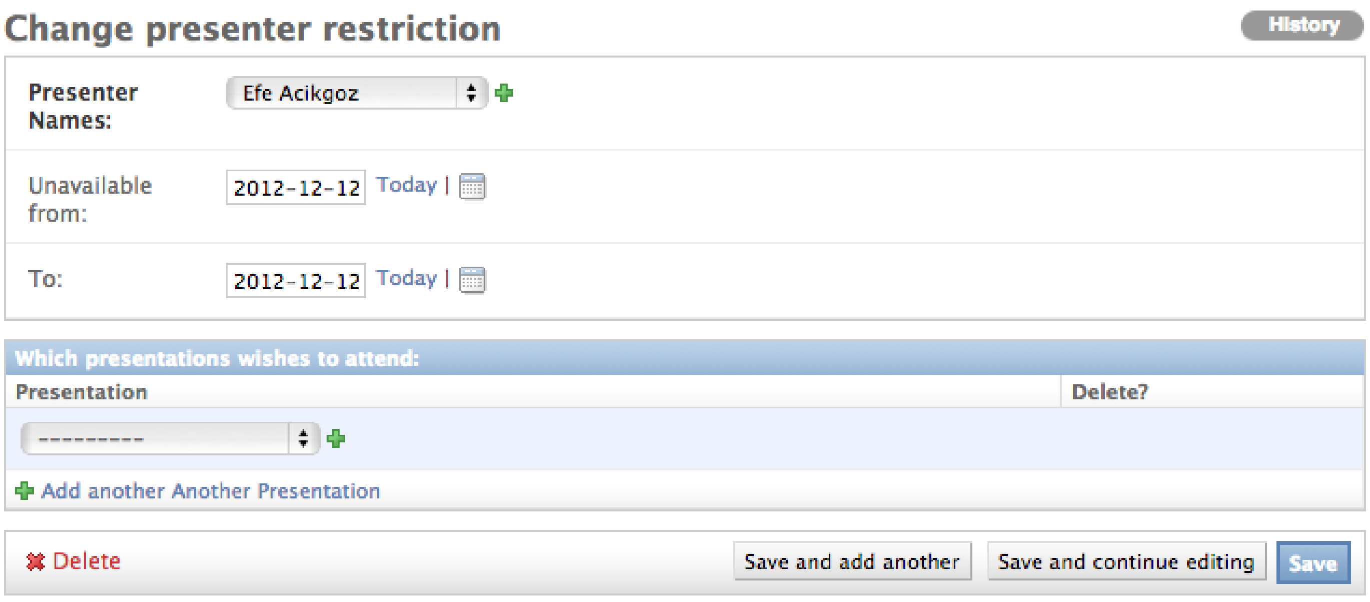
New pseudocode of the algorithm:

```
1-  leastResistance = x
2-  while currentResistance<leastResistance
3-    availableSpotsList = allSpotsForConference()
4-    foreach presentation in presentationList
5-        checkPheromones()
6-        spot = getSpotFromAvailableSpots()
7-        availableSpotsList.remove( spot )
8-        updatePheromones()
9-    end foreach
10-   calculateCurrentResistance()
11- end while
```

allSpotsForConference() returns all available spots in a conference as an array

getSpotFromAvailableSpots() picks a spot for the presentation.

Figure 3.2.4.2 – Presenter restrictions page


Change presenter restriction

Presenter Names:	Efe Acikgoz
Unavailable from:	2012-12-12
To:	2012-12-12

Which presentations wishes to attend:

Presentation	Delete?

The available spots array is a sequencing integer array created automatically for every spot in the conference (presentations in a day * number of rooms * number of days), and calculates the picked spot using the sequence

```
hit_time = hit_spot%(morning_size+afternoon_size)
if hit_time>morning_size:
   afternoon = "True"
   hit_time = hit_time-morning_size
else:
   afternoon = "False"
hit_day = hit_spot/((morning_size+afternoon_size)*day_count)
hit_room = (hit_spot/(morning_size+afternoon_size))%room_count
```

This would be fairly easy if there were no constrains available for presenters/presentations like a presenter might wish to pick specific days of the conference he / she is available for presentation. So at this point we either needed to add loop for picking spots where presenter unavailable, and loop upon picking an unavailable spot, or change the algorithm further. we went with changing the algorithm and decided to get presenters available spots and pick a spot from the intersecting part of presenter's available spots and available spots of the conferences.

After picking a spot, we remove it from available spots list and update pheromones. So did this at all increase performance? A 90-presentation conference used to take 450 iterations and 0.12 seconds in average and after the algorithm change; iterations went down to 0 (no more loop for picking full or unavailable spots) and time it took to calculate went down to 0.01 seconds in average, which is a good thing but are there any downsides to this change? Yes there is one: Increased ram usage. If there is a 1000 presentation conference, we need to track all the spots, and we used short integers for the integer array, so it is 2byte * 1000 = 2KB extra ram space, but considering todays computers at least has 4GB of ram this number is not important

At this point the algorithm calculates sessions pretty fast, but there is a case where a presenter with most restrictions can be last person to be placed and he might not have a spot that is not in his restrictions. This is not a big issue with small presentation pools and presenters with low restrictions, but will be an issue with big conferences with a lot of restrictions. Normally it will get fixed sooner or later thanks to pheromones setting a path with less resistance but what if we merged the algorithm with some other algorithms? “Survival of the fittest” behavior of the genetic algorithm might be more fitting. Survival of the fittest behavior of the genetic algorithm forces it to get paths with least resistance, and examining this behavior reveals that to have the least amount of resistance we need to get rid of highest resistance presenters, or presenters with most restrictions by getting them spots first.

Another advantage of calculating all available spots was that the presentation spots’ pool stays the same with initial algorithm (if there were 250 spots, there would be 250 spots even at the end). But with available spots change, the presentation spot pool decreases on every step, which means less time required to pick a spot for the presentation. On the algorithm this means the further you travel towards the food source, the less possible paths to choose from left.

After that change, the pseudocode of the algorithm evolves into:

```
1-  leastResistance = x
2-  while currentResistance<leastResistance
3-    availableSpotsList = allSpotsForConference()
4-    sortedPresentationList = presentationList.sortByHighestRestriction()
5-    foreach presentation in sortedPresentationList
6-        checkPheromones()
7-        spot = getSpotFromAvailableSpots()
8-        availableSpotsList.remove( spot )
9-        updatePheromones()
10-   end foreach
11-   calculateCurrentResistance()
12- end while
```

Where sortByHighestRestriction() function sorts the presentation list ordering by the restriction size of presenters

So to summarize, so far to calculate the sessions we first create an array with an integer representing every empty spot in the conference to track the empty spots. Then we sort presenters ordering by their restriction size (presenter with highest restriction first) so we will have less restriction, check pheromones and pick a spot for the presentation that Is available and not in the restriction list of the presenter, remove the picked spot from available spots array and update the pheromones.

Now that we sorted the presenter list so that we will have the least amount of resistance, checking for schedule resistance becomes kind of redundant. The whole reason we checked resistances was to pick the least amount of resistance for the schedule but our latest change already manages to do just that, so we can now remove it

```
1- availableSpotsList = allSpotsForConference()
2-   sortedPresentationList = presentationList.sortByHighestRestriction()
3-   foreach presentation in sortedPresentationList
4-       checkPheromones()
5-       spot = getSpotFromAvailableSpots()
6-       availableSpotsList.remove( spot )
7-       updatePheromones()
8-   end foreach
```

3.2.4.1 POSSIBLE MODIFICATIONS TO THE ALGORITHM

There were a few ideas we had but did not implement into the application but wrote the algorithm modifications of.

First one was adding room sizes.

Adding room sizes to the algorithm where preferred presentations gets the bigger room or even possibly get a second presentation if it has a bigger audience than biggest room available. To do this addition, we would need update conference table to add room size to the database. Then we would have to implement a visitor system to the application where users can register and select the presentations they wish to see. When we have enough registrations to the presentations (or when registration date for presentations ends).

When we have the visitor data we would also have to modify the algorithm so that it would respect the visitor requests to see presentations. The algorithm we came up with was:

```
1-  availableSpotsList = allSpotsForConference()
2-  sortedPresentationList = presentationList.sortList()
3-  sortedPresentationListWithRepeat = sortedPresentationList.checkRepeats()
4-  foreach presentation in sortedPresentationListWithRepeat
5-      checkPheromones()
6-      spot = getSpotFromAvailableSpots()
7-      availableSpotsList.remove( spot )
8-      updatePheromones()
9-  end foreach
```

sortList() function first function sorts the presentation list ordering by the restriction size of presenters, then it sort the array ordering by the audience size
checkRepeats() function checks if any presentation has more audience than room size. If so, it splits the room so all audience can attend it. After split, the second part is put at the end array.
getSpotFromAvailableSpots() function also gets a modification where it checks if the presenter has any more presentation at the picked time

The upside of this implementation would be the ability to have different room sizes, but the downsize is that it requires user input before execution. There might also be cases where after split the only available spot being the same time as the presenter's presentation, or in the restriction time zone of the presenter, which will increase execution time.

Another possible addition to the algorithm was addition of travel distance between rooms. Which meant that we had to keep in mind the travelling distance between the rooms. To do this we had to alter database and add room distances in conference table. This would be a fairly useful to keep in mind about, but since we did not implement audience preferences, this addition became redundant.

The way we designed this implementation was; if users had preferred to see 2 different presentations, we can not put those 2 presentations until time required to be spent moving between them has passed. For example if users wanted to see presentation A and B, if presentation A was in room 1 and B in room 2 if we assume it takes 15 minutes to walk between those rooms, presentation A and B should be at least 15 minutes apart (if presentations take 20 minutes and presentation A is at 14:00 presentation B can start latest at 13:25, or earliest at 14:35)

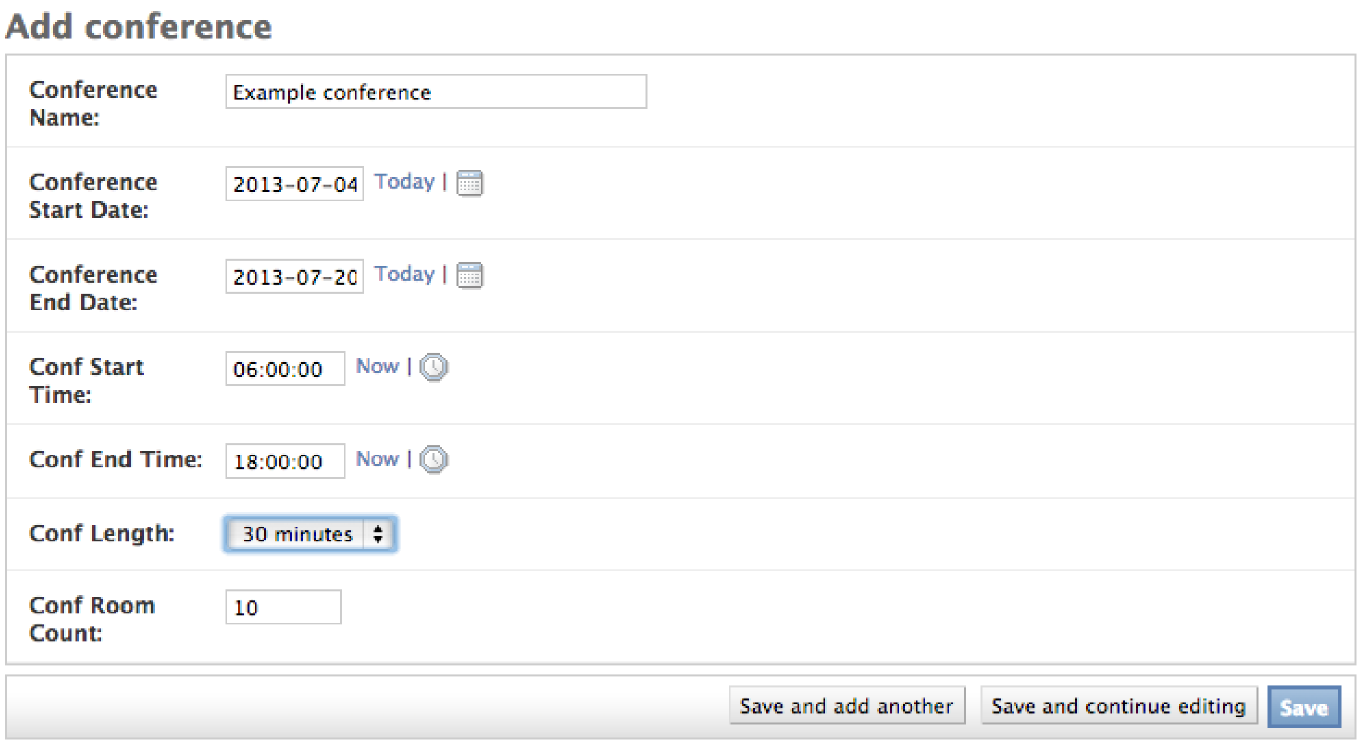
3.2.5 IMPLEMENTATION

Since we had a decent algorithm at this point, we decided to implement them all to see their performances and if my improvements actually had any effect.

First of all, we implemented the initial algorithm built the models, views and controllers. we automatized the UI so different length, size conferences with different presentation length can be viewed without any problems.

Conferences can start at any given time and can be either 15, 20 or 30 minutes long.

Figure 3.2.5.1 – Add conference screen



Add conference

Conference Name:	Example conference
Conference Start Date:	2013-07-04 Today |
Conference End Date:	2013-07-20 Today |
Conf Start Time:	06:00:00 Now |
Conf End Time:	18:00:00 Now |
Conf Length:	30 minutes
Conf Room Count:	10

There are 4 models in the application, which are Conference, Presenter, Presentation, and Presentation restriction

Conference model has conference name, 3 different time slices for presentation length (15, 20, 30 minutes), conference start and end dates as well as conference start and end times for the days (Ex: 6 am to 8 pm), and number of rooms.
Presenter model has presenter name and registration date.
Presentation model has presenter name, which is a foreign key to presenter, and another foreign key is to the conference model. Presentation model has presentation name field.
Last model is presenter restrictions, which has presenter name, which is a foreign key to presenter model, unavailable from and to dates.

When a session scheduling is requested, we get the conference restrictions, presentations and presenter data from the database and calculate the number of slices in a day

```
def get_time_slice (period, start, end):
  start_hours = str( start ).split(':')
      end_hours = str( end ).split(':')
      slices = (( int( end_hours[0]) * 60 + int (end_hours[1])) -
                 (int (start_hours[0])*60+ int (start_hours[1]))) / period
```

As well as number of slices in the morning and afternoon, there is an hour of noon break and slices through the day split between morning and afternoon, and calculate the days.

We save various data like number of slices in the morning, afternoon, start and end time and date, room count, presentations in a dictionary to create the UI. We then calculate session schedules

```
def calculate_without_recursion(conference, presentations, restrictions, morning, afternoon, days_count, count):
    room_count = conference.Conf_Room_Count
    available_spots = fill_available_spots_array(len(morning),len(afternoon),room_count,days_count)
    schedule = []
    del schedule[:]
    for presentation in presentations:
        plan = {}
        plan['room'], plan['day'], plan['time'], plan['afternoon'], available_spots = get_spot_no_recursion (available_spots,
room_count,days_count,len(morning),len(afternoon))
        plan['name'] = presentation.Presentation_Name
        schedule, success = add  to  list(schedule, plan)
```

We also calculate the time it takes to run the program and add it to the dictionary, and fill the templates.

There is also an admin page that helps adding new conferences, presenters, presentations, restrictions, users and much more. Admin page is the main way to add information to the database, you cannot directly connect to db and add objects, you can only create an object on view controllers and save or get data from db to these objects

There are more than 20 files that helps us with user experience and calculations, but the files above has the main code that does calculations and user interface like urls.py which tells server which functions are connected to which urls, or admin.py pages that create the admin interface

To make sure we did not affect test performances, we uploaded my project to an external server. We used Heroku to host my server, which offers a cloud platform as a service and supports Python / Django framework.

4. RESULTS

When we first started working on this thesis we aimed to create an algorithm that can check possible session schedules while satisfying the presentation restrictions, and we managed to create an algorithm that can create session schedules, and increased initial performance.

After implementing the application for initial algorithm, we did around 1000 performance runs to make sure we got a big enough test pool so that we was sure the performance was consistent.

We grouped tests like “Initial algorithm – 42 presentation”, “Initial algorithm – 90 presentation”, “Algorithm change 1 – 42 presentation” and so on for every combination of algorithm and 4 different presentation size (42, 90, 160, 250)

All iterations here are from hitting filled spots, not including the loop for all presentations.

Test results were fairly consistent for **initial algorithm**

42 Presentations: 3 days, 30 min presentation, 2 rooms, 10:00-14:30 | 25 runs

312 retry, 0.04 second
137 retry, 0.02 second
226 retry, 0.03 second
183 retry, 0.04 second
180 retry, 0.07 second
198 retry, 0.03 second
214 retry, 0.07 second
274 retry, 0.09 second
249 retry, 0.03 second
162 retry, 0.04 second

140 retry, 0.04 second

165 retry, 0.02 second

173 retry, 0.01 second

138 retry, 0.01 second

190 retry, 0.04 second

188 retry, 0.01 second

179 retry, 0.03 second

244 retry, 0.06 second

389 retry, 0.06 second

277 retry, 0.07 second

250 retry, 0.04 second

209 retry, 0.05 second

289 retry, 0.04 second

286 retry, 0.04 second

156 retry, 0.03 second

90 Presentations: 3 days, 30 min presentation, 3 rooms, 10:00-16:00 | 25 runs

440 tries, 0.11 second

406 tries, 0.09 second

403 tries, 0.09 second

360 tries, 0.08 second

324 tries, 0.08 second

627 tries, 0.16 second

431 tries, 0.11 second

308 tries, 0.08 second

272 tries, 0.04 second

684 tries, 0.12 second

757 tries, 0.08 second

437 tries, 0.06 second

340 tries, 0.07 second

530 tries, 0.13 second

503 tries, 0.10 second

421 tries, 0.12 second

333 tries, 0.09 second

357 tries, 0.06 second

397 tries, 0.10 second

407 tries, 0.11 second

586 tries, 0.12 second

564 tries, 0.15 second

518 tries, 0.06 second

531 tries, 0.13 second

526 tries, 0.13 second

160 Presentations: 4days, 30 min presentation, 4 rooms, 10:00-16:00 | 25 runs

1021 retry, 0.37 second

992 retry, 0.35 second

812 retry, 0.31 second

752 retry, 0.31 second

842 retry, 0.41 second

676 retry, 0.29 second

876 retry, 0.31 second

867 retry, 0.33 second

811 retry, 0.34 second

801 retry, 0.31 second

842 retry, 0.33 second

865 retry, 0.32 second

799 retry, 0.30 second

1010 retry, 0.34 second

942 retry, 0.33 second

963 retry, 0.34 second

675 retry, 0.24 second

878 retry, 0.33 second

879 retry, 0.43 second

769 retry, 0.29 second

980 retry, 0.35 second

848 retry, 0.34 second

901 retry, 0.36 second

743 retry, 0.24 second

798 retry, 0.30 second

250 Presentations: 5 days, 30 min presentation, 5 rooms, 10:00-16:00 | 25 runs

2112 tries, 0.80 second

1990 tries, 0.78 second

1879 tries, 0.65 second

1921 tries, 0.68 second

1821 tries, 0.60 second

2045 tries, 0.71 second

1890 tries, 0.69 second

2012 tries, 0.72 second

2051 tries, 0.73 second

2394 tries, 0.76 second

2109 tries, 0.71 second

2494 tries, 0.74 second

2038 tries, 0.69 second

1890 tries, 0.72 second

1955 tries, 0.68 second

2343 tries, 0.71 second

2012 tries, 0.74 second

2117 tries, 0.72 second

1782 tries, 0.70 second

1679 tries, 0.65 second

1927 tries, 0.71 second

2046 tries, 0.72 second
2594 tries, 0.74 second
2942 tries, 0.73 second
1821 tries, 0.69 second

The test results above are randomly selected from the test pool of every test group and shows that the initial algorithm low performance on larger number of presentations.

Average numbers of initial algorithm are:

0,04 seconds average for 42 presentation conference
0,11 seconds average for 90 presentation conference
0,34 seconds average for 160 presentation conference
0,77 seconds average for 250 presentation conference

216 iterations for 42 presentation conference
443 iterations for 90 presentation conference
844 iterations for 160 presentation conference
1941 iterations for 250 presentation conference

Table 4.1 – Conference size - Time

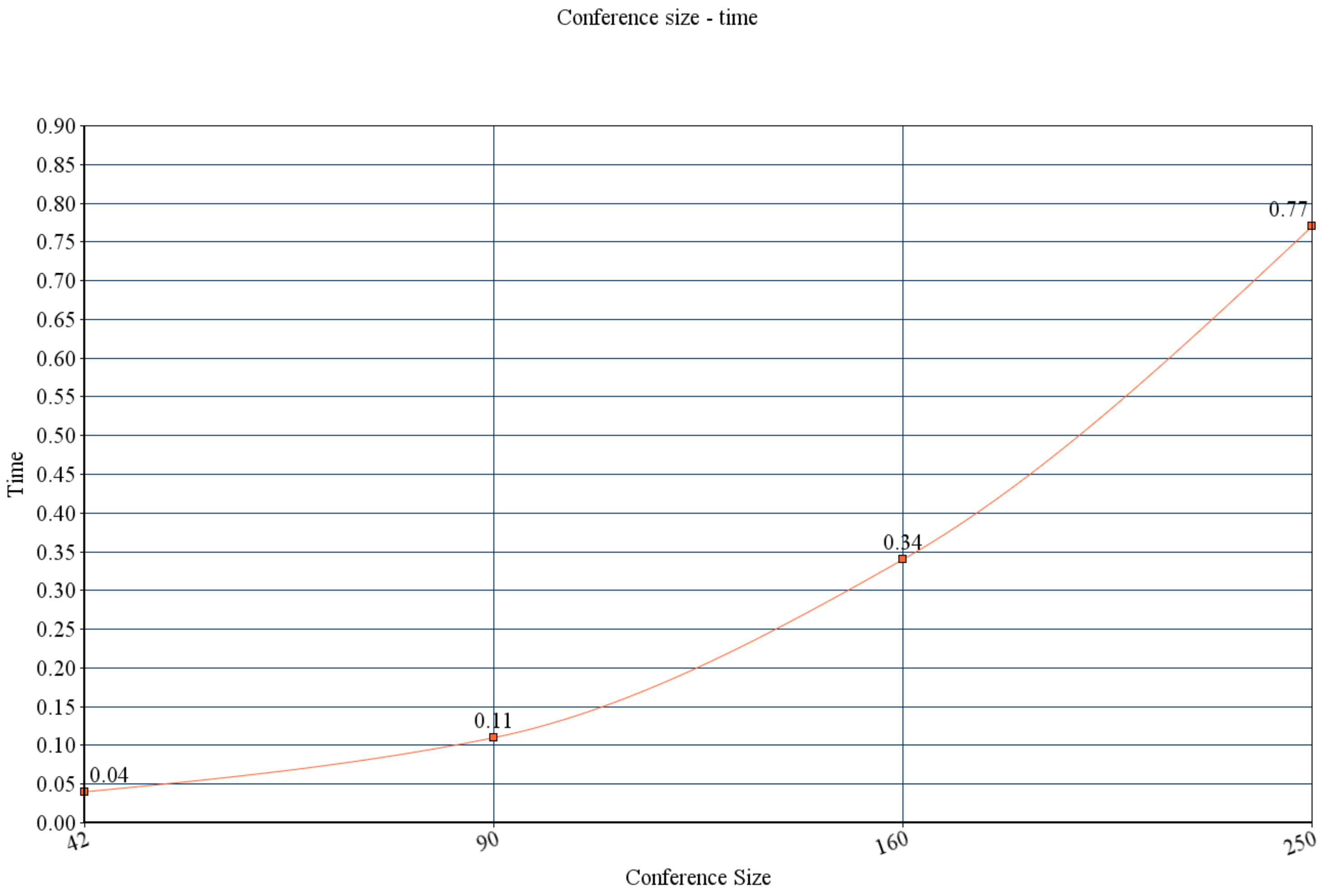
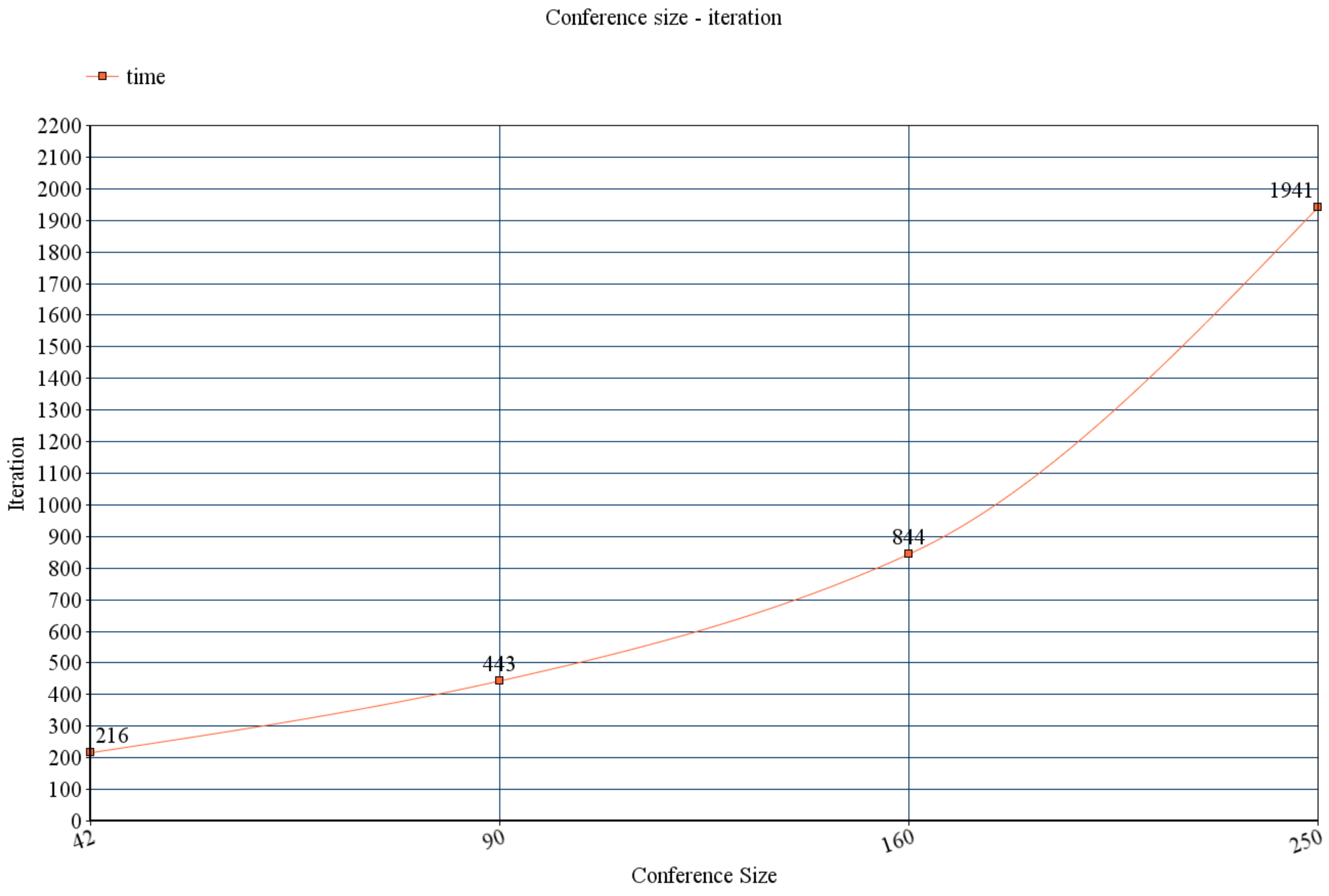


Table 4.2 – Conference size - Iteration



To compare the performance of **latest algorithm**, the test results are:

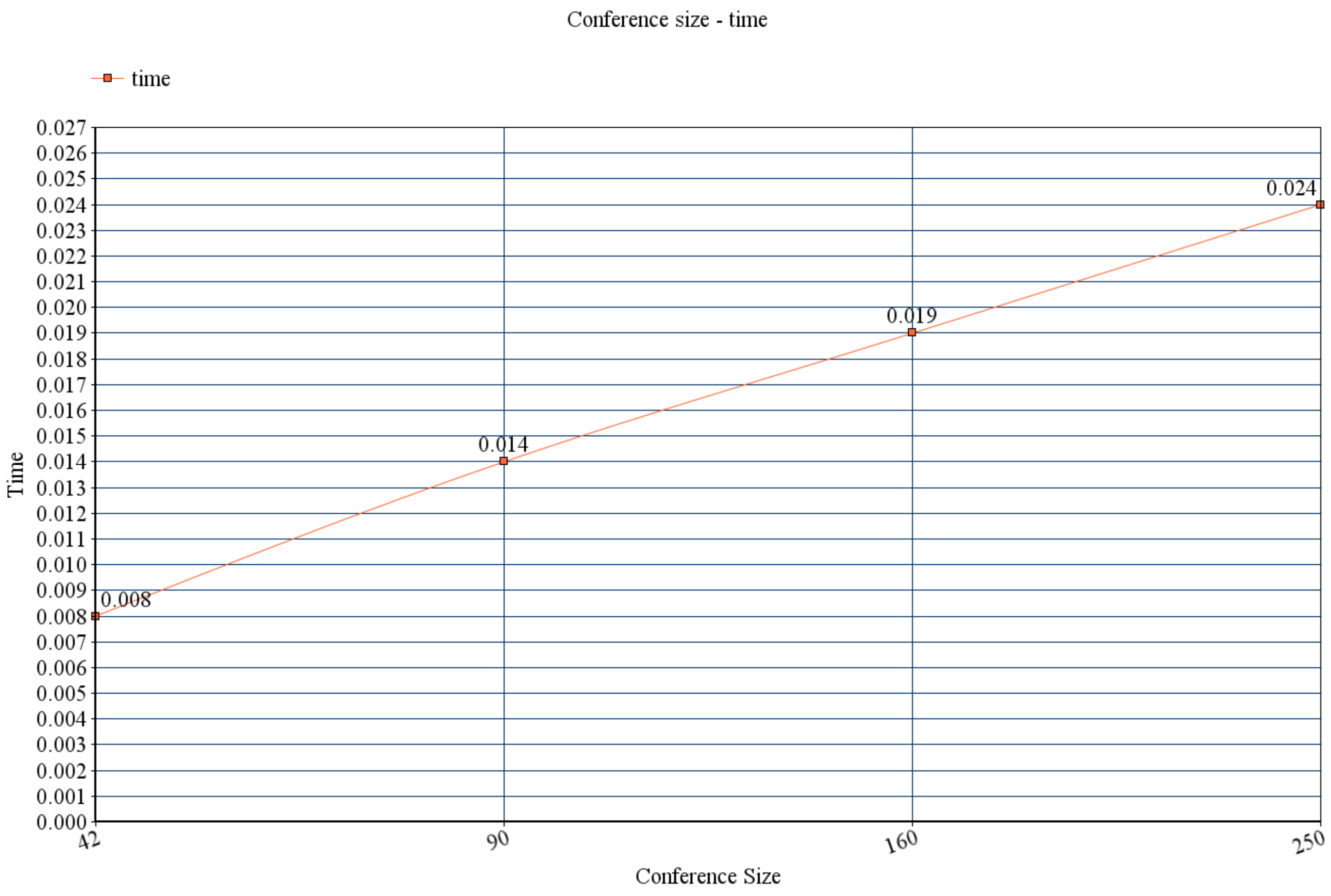
0,008 seconds average for 42 presentation conference

0,014 seconds average for 90 presentation conference

0,019 seconds average for 160 presentation conference

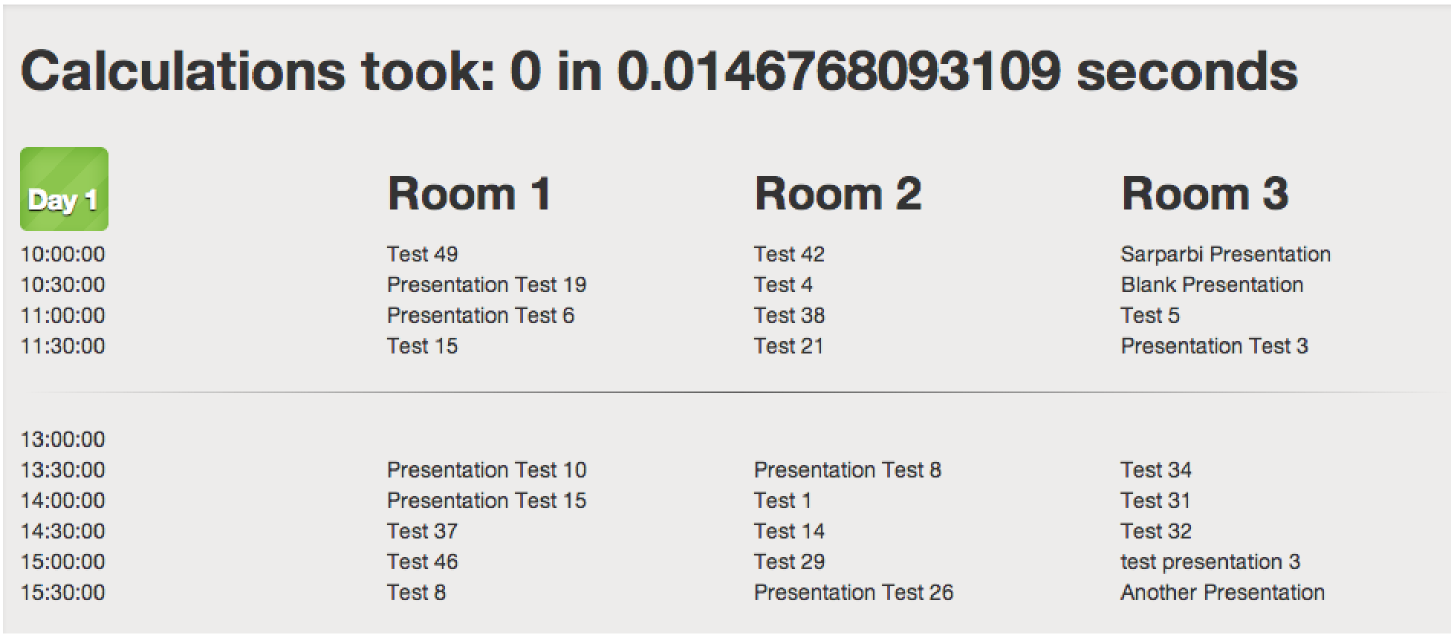
0,024 seconds average for 250 presentation conference

Table 4.3 – Latest algorithm performance chart



There are no iterations (from filled spots, iterations can still happen for some cases) for latest algorithm since presentations cannot hit unavailable spots.

Figure 4.1 - Example performance of latest algorithm with 90 presentations



Calculations took: 0 in 0.0146768093109 seconds

Day 1	Room 1	Room 2	Room 3
10:00:00	Test 49	Test 42	Sarparbi Presentation
10:30:00	Presentation Test 19	Test 4	Blank Presentation
11:00:00	Presentation Test 6	Test 38	Test 5
11:30:00	Test 15	Test 21	Presentation Test 3
13:00:00			
13:30:00	Presentation Test 10	Presentation Test 8	Test 34
14:00:00	Presentation Test 15	Test 1	Test 31
14:30:00	Test 37	Test 14	Test 32
15:00:00	Test 46	Test 29	test presentation 3
15:30:00	Test 8	Presentation Test 26	Another Presentation

Performance difference between algorithms:

From 0,04 seconds to 0,008 seconds average for 42 presentation conference

From 0,11 seconds to 0,014 seconds average for 90 presentation conference

From 0,34 seconds to 0,019 seconds average for 160 presentation conference

From 0,77 seconds to 0,024 seconds average for 250 presentation conference

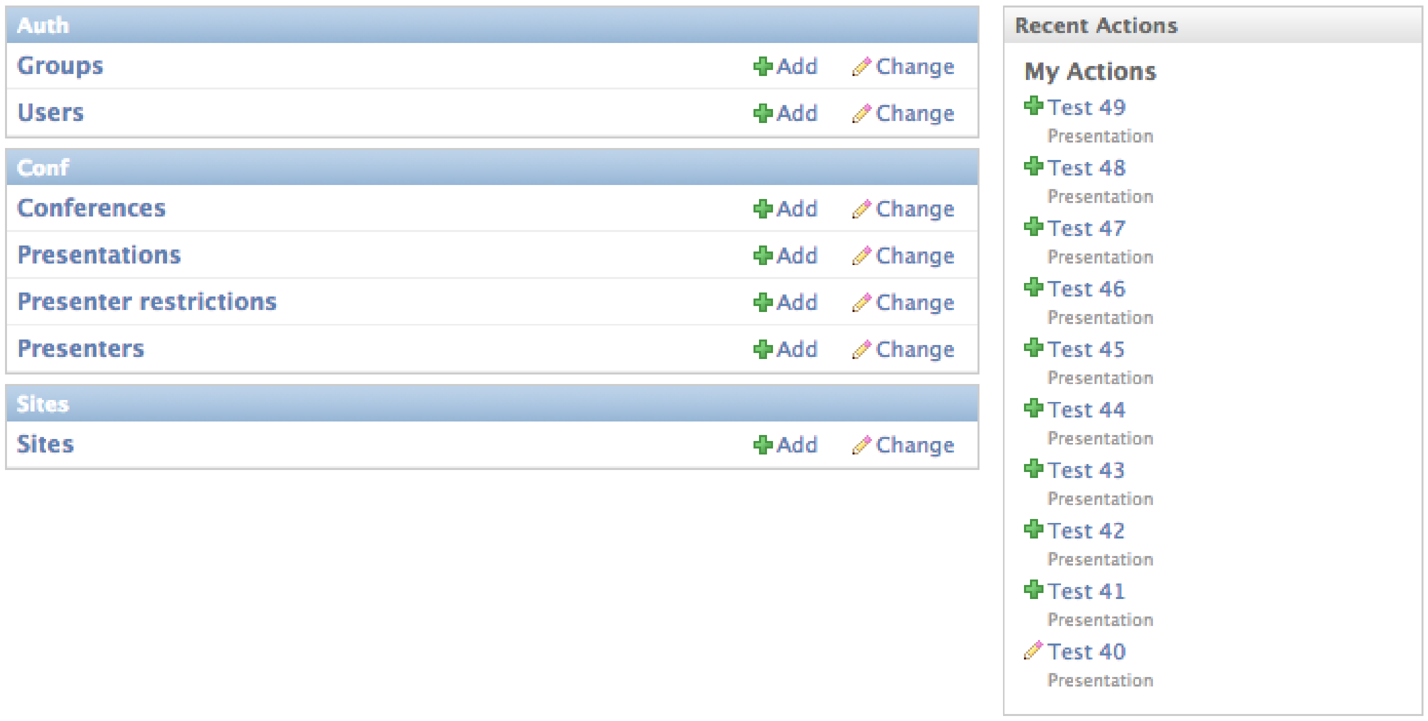
The reason latest algorithm does not scale as much as initial algorithm is that with latest algorithm, the further you travel towards the food source (or the further you fill the schedule) the less possible roads left to pick from, which increases the performance.

5. DISCUSSION

There was a paper and a few blog posts about doing session scheduling using different algorithms like Genetic Algorithm, but there were no published works about doing session scheduling using Ant Colony Algorithm. Using different algorithms to do a job is important, since you can compare performance, advantages and disadvantages of different algorithms for that job when you have multiple algorithms. You can compare efficiency, complexity, performance, memory use, time needed to finish operations. For example efficiency refers to the algorithmic efficiency as the size of the input grows large, how does the iterations scale with it?

When we first started this thesis, we aimed to create an application that does session, and not only we managed to do it, we modified and merged it with other algorithms like Genetic algorithm to increase performance and flexibility (presenter restrictions and such)

Figure 5.1 – Administration page



Auth		
Groups	Add	Change
Users	Add	Change
Conf		
Conferences	Add	Change
Presentations	Add	Change
Presenter restrictions	Add	Change
Presenters	Add	Change
Sites		
Sites	Add	Change

Recent Actions

My Actions

Test 49
Presentation

Test 48
Presentation

Test 47
Presentation

Test 46
Presentation

Test 45
Presentation

Test 44
Presentation

Test 43
Presentation

Test 42
Presentation

Test 41
Presentation

Test 40
Presentation

6. CONCLUSION

In conclusion, we did manage to create an application that does session scheduling for conferences using Ant Colony Algorithm. We also managed to modify the algorithm to see if we can increase the performance of the algorithm for this specific job.

We used python Django framework to implement the application, and implemented all modifications to the algorithm to test performance, effectiveness, and memory usage changes.

Tests indicate that ant colony algorithm can be an effective way to do session scheduling for conferences, but is it better than other algorithms like genetic algorithm?

For example, Genetic Algorithm can have sessions, rooms, timeslots and set of preference sessions to implement a "survival of the fittest" algorithm which means that solutions are generated, and bad solutions are eliminated and good solutions are carried over to next step, while using a fitness function to validate a given solution with the set of constraints and list of preference sessions to return a value indicating how relevant the solution is.

We too had an algorithm with different sizes of room, sessions, timeslots and set of presentation restrictions to search for eligible and effective solutions. Given the advantages of the Genetic Algorithm's "Survival of the fittest" behavior instead of one algorithm, it will be most effective to create a merge of Ant Colony Algorithm and Genetic Algorithm to do session scheduling. With Genetic Algorithm's survival of the fittest behavior and Ant Colony Algorithm's multi threading and pheromone technique, we can eliminate bad solutions faster and calculate the better solutions faster and more effective.

To summarize, it is possible to do session scheduling using ACO, and there is room for algorithm to be improved for doing specific jobs. We also created a client / server application that implements the algorithm we modified using python Django framework.

To anyone else who wishes to do research on this subject, should also research Tabu search as well as Simulated annealing which is a generic probabilistic metaheuristic for global optimization problem of locating a good approximation to global optimum of a given function in a large search space. It is often used when the search space is discrete. Tabu search is a local search method used for mathematical optimization where it does a local search to take a potential solution to a problem and check its immediate neighbors(solutions that are similar with few minor differences) in hope of finding an improved solution. Tabu search enhances the performance by using memory structures that describe the visited solutions or user-provided sets of rules. If a potential solution has been previously visited within a certain short-term period or if has violated a rule it is marked as tabu so that the algorithm does not consider that possibility repeatedly.

REFERENCES

Periodicals

Rafael S. Parpinelli, Heitor S. Lopes, Alex A. Freitas2 "*Data Mining with an Ant Colony Optimization Algorithm*" CEFET-PR, CPGEI, Av. Sete de Setembro, 3165, Curitiba - PR, 80230-901, Brazil

B. Houlding & J. Haslett (10 Jan 2012) "*Scheduling parallel conference sessions: an application of a novel hybrid clustering algorithm for ensuring constrained cardinality*" Journal of Applied StatisticsVolume 40, Issue 5, 2013

Sha Fan (Jan 2010) "*Session Scheduling Algorithm of Grid Computing*" Knowledge Discovery and Data Mining, 2010. WKDD '10. Third International Conference on

A.J Wellings(Nov 1997) "*Synchronous sessions and fixed priority scheduling*" Journal of Systems Architecture Volume 44, Issue 2, November 1997, Pages 107-118

Thompson, G. M. (2002) "*Improving conferences through session scheduling.*" Cornell Hotel and Restaurant Administration Quarterly 2002 Vol. 43 No. 3 pp. 71-76

Yaser Pourmohammadi Fallah, Hussein Alnuweiri (2008) "*Analysis of temporal and throughput fair scheduling in multirate WLANs*" Computer Networks Volume 52, Issue 16 Pages 3169-3183

Yaning Wang, Linghang Fan, Dan He, Rahim Tafozolli (2008) "*Performance comparison of scheduling algorithms in network mobility environment*" Computer Communications Volume 31, Issue 9 Pages 1727-1738

Ye Huang, Nik Bessis, Peter Norrington, Pierre Kuonen, Beat Hirsbrunner (Jan 2013) "*Exploring decentralized dynamic scheduling for grids and clouds using the community-aware scheduling algorithm*" Clouds and Service-Oriented Architectures Volume 29, Issue 1 Pages 402-415

Hossein Miar Naimi, Nima Taherinejad (Jan 2009) "*New robust and efficient ant colony algorithm: Using new interpretation of local updating process*" Experts Systems with Applications Volume 36, Issue 1, Pages 481-488

D Whitley, T Sharkweather, C Bogart (Aug 1990) "*Genetic algorithms and neural networks: optimizing connections and connectivity*" Parallel Computing Volume 14, Issue 3, Pages 347-361

Other publications

Michael Swanson (May 2008) “PDC 2008 Conference Scheduling Using a Genetic Algorithm” [online] http://blogs.msdn.com/b/mswanson/archive/2008/05/03/pdc-2008-conference-scheduling-using-a-genetic-algorithm.aspx

Ali Tarhini (Feb 2012) “Genetic Algorithm for Conference Schedule Mining” [online] http://alitarhini.wordpress.com/2012/02/27/genetic-algorithm-for-conference-schedule-mining/

Wikipedia “Ant colony optimization algorithms” [online] http://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms

APPENDICES

Appendices A1 - Code:

Admin page (admin.py)

```
from conf.models import Presenter, Conference, PresenterRestriction, Presentation
from django.contrib import admin
class PresenterInline(admin.TabularInline):
    model = Presentation.Name.through
    verbose_name = 'Another presentor'
    verbose_name_plural = 'Presenters:'
    extra = 1

class PresentationInline(admin.TabularInline):
    model = PresenterRestriction.Presentations_To_Attend.through
    verbose_name = 'Another presentation'
    verbose_name_plural = 'Which presentations wishes to attend:'
    extra = 1

class presenterAdminPage(admin.ModelAdmin):
            list_display = ('Presenter_Name', 'Register_Date')
            fieldsets = [
                        ('Name',                {'fields': ['Presenter_Name']}),
                        ('Date Added',          {'fields': ['Register_Date']}),
            ]

class conferenceAdmin(admin.ModelAdmin):
    list_display = ('Conference_Name', 'Conf_Date_Start', 'Conf_Date_End')

class presentationAdmin(admin.ModelAdmin):
    list_display = ('Presentation_Name', 'Conference')
    inlines = (PresenterInline,)
    exclude = ('Name',)

class PresenterRestrictionAdmin(admin.ModelAdmin):
    list_display = ('Presenter_Names', 'Unavailable_Start', 'Unavailable_End')
    inlines = (PresentationInline,)
    exclude = ('Presentations_To_Attend', )

admin.site.register(Presenter,presenterAdminPage)
admin.site.register(Conference, conferenceAdmin)
admin.site.register(Presentation, presentationAdmin)
admin.site.register(PresenterRestriction, PresenterRestrictionAdmin)
```

Calculate ant colony (calculate_ant_colony.py)

```
import random
import math

def calculate(conference, presentations, restrictions, morning, afternoon, days, count):
    room_count = conference.Conf_Room_Count
    schedule = []
    del schedule[:]
    for presentation in presentations:
        success = False
        while not success:
            plan = {}
            plan['room'], plan['day'], plan['time'], plan['afternoon'] =
get_random_position(room_count, days, len(morning)+len(afternoon), len(morning))
            plan['name'] = presentation.Presentation_Name
            schedule, success = add_to_list(schedule, plan)
            count = count+1
    return schedule, count

def get_random_position(rooms, days, slices, morning):
    room = random.randrange(rooms)
    day = random.randrange(days)
    time = random.randrange(slices)
    afternoon = "False"
    if time >= morning:
        print ("time: " + str(time) + " morning: " + str(morning))
        time = time - morning
        afternoon = "True"
    return room, day, time, afternoon


def add_to_list(to_return, item_to_add):
    flag = True
    success = True
    for element in to_return:
        if element['room'] == item_to_add['room'] and element['day'] ==
item_to_add['day'] and element['time'] == item_to_add['time'] and element['afternoon']
== item_to_add['afternoon']:
            flag = False
    if flag:
        to_return.append(item_to_add)
    else:
        success = False
    return to_return, success

def calculate_without_recursion(conference, presentations, restrictions, morning,
afternoon, days_count, count):
    room_count = conference.Conf_Room_Count
    available_spots =
fill_available_spots_array(len(morning),len(afternoon),room_count,days_count)
    schedule = []
    del schedule[:]
    for presentation in presentations:
        plan = {}
        plan['room'], plan['day'], plan['time'], plan['afternoon'], available_spots =
get_spot_no_recursion(available_spots,room_count,days_count,len(morning),len(aftern
oon))
        plan['name'] = presentation.Presentation_Name
        schedule, success = add_to_list(schedule, plan)
    return schedule, 0

def fill_available_spots_array(morning_size, afternoon_size, room_count, days_count):
    count = 0
    to_return = []
    for x in xrange(0,(morning_size + afternoon_size)*room_count*days_count):
        to_return.append(count)
        count= count+1
    return to_return

def
get_spot_no_recursion(available_spots,room_count,day_count,morning_size,afternoon_
size):
    if len(available_spots)>0:
        hit_spot = available_spots[random.randrange(len(available_spots))]
        available_spots.remove(hit_spot)
        hit_time = hit_spot%(morning_size+afternoon_size)
        if hit_time>morning_size:
            afternoon = "True"
            hit_time = hit_time-morning_size
        else:
            afternoon = "False"
        hit_day = hit_spot/((morning_size+afternoon_size)*day_count)
        hit_room = (hit_spot/(morning_size+afternoon_size))%room_count
        print(hit_room, hit_day, hit_time, afternoon)
        return hit_room, hit_day, hit_time, afternoon, available_spots


def get_random_position(rooms, days, slices, morning):
    room = random.randrange(rooms)
    day = random.randrange(days)
    time = random.randrange(slices)
    afternoon = "False"
    if time >= morning:
        print ("time: " + str(time) + " morning: " + str(morning))
        time = time - morning
        afternoon = "True"
    return room, day, time, afternoon
    #print("room = " + str(room)+ " day = "+str(day) + " time = " + str(time))


def add_to_list(to_return, item_to_add):
    flag = True
    success = True
    for element in to_return:
        if element['room'] == item_to_add['room'] and element['day'] ==
item_to_add['day'] and element['time'] == item_to_add['time'] and element['afternoon']
== item_to_add['afternoon']:
            flag = False
    if flag:
        to_return.append(item_to_add)
    else:
        success = False
    return to_return, success
```

Day slices calculation (functions.py)

```
import datetime as dt

def get_time_slice(period, start, end):
    start_hours = str(start).split(':')
    end_hours = str(end).split(':')
    slices = ((int(end_hours[0]) * 60 + int(end_hours[1])) -
              (int(start_hours[0])*60+int(start_hours[1]))) / period
    return slices


def fill_time_values(period, start, end):
    slice_size = get_time_slice(period, start, end)
    to_return = []
    for until in range(slice_size):
        time_slice = dt.timedelta(minutes = (period*until))
        to_return.append(str((dt.datetime.combine(dt.date(1,1,1),start) +
time_slice).time()))
    return to_return

def get_days(start, end):
    days  = end-start
    to_return = days.days
    return to_return
```

Database models (models.py)

```
from django.db import models
```

```
class Presenter(models.Model):
    Presenter_Name = models.CharField(max_length=100)
    Register_Date = models.DateTimeField('Date Registered')
    def __unicode__(self):
        return self.Presenter_Name

class Conference(models.Model):
    Conference_Name = models.CharField(max_length=125)
    TIME_SLICES = (
            (15, '15 minutes'),
            (20, '20 minutes'),
            (30, '30 minutes'),
        )
    Conf_Date_Start = models.DateField('Conference Start Date')
    Conf_Date_End = models.DateField('Conference End Date')
    Conf_Start_Time = models.TimeField()
    Conf_End_Time = models.TimeField()
    Conf_Length = models.IntegerField(choices=TIME_SLICES)
    Conf_Room_Count = models.IntegerField()
    def __unicode__(self):
        return self.Conference_Name

class PresenterRestriction(models.Model):
    Presenter_Names = models.ForeignKey('Presenter', null=True)
    Unavailable_Start = models.DateField('Unavailable from',null=True, blank=True)
    Unavailable_End = models.DateField('to',null=True, blank=True)
    Presentations_To_Attend = models.ManyToManyField('Presentation', null=True,
symmetrical=False)

class Presentation(models.Model):
    Name = models.ManyToManyField('Presenter')
    Conference = models.ForeignKey('Conference')
    Presentation_Name = models.CharField(max_length=100)
    def __unicode__(self):
        return self.Presentation_Name
```

URL resolver (urls.py)

```
from django.conf.urls import patterns, url

urlpatterns = patterns('conf.views',
    url(r'^$', "index"),
    url(r'^(?P<conf_id>\d+)/$', 'viewConference'),
    url(r'^(?P<conf_id>\d+)/calculate/$', 'calculate'),
    )
```

View Controller (views.py)

```
from django.http import HttpResponseRedirect, HttpResponse
from django.shortcuts import render_to_response
from django.template import RequestContext
import datetime, time
from time import clock as c, time as t
import functions, calculate_ant_colony
from conf.models import Presenter, Conference, Presentation, PresenterRestriction

def index(request):
    template_data = {}
    template_data['conferences'] = Conference.objects.order_by('-Conf_Date_Start')
    return render_to_response('conferences.html', template_data,
context_instance=RequestContext(request))

def viewConference(request, conf_id):
    template_data = {}
    template_data['Presentations'] = Presentation.objects.filter(Conference = conf_id)
    template_data['conf_id'] = conf_id
    return render_to_response('_details.html', template_data,
context_instance=RequestContext(request))

def calculate(request, conf_id):
    before = t()
    template_data = {}
    #variables
    conference = Conference.objects.get(id=conf_id)
    restrictions = PresenterRestriction.objects.all()
    FMT = '%H:%M:%S'
    noon = datetime.time(12,0,0)
    afternoon = datetime.time(13,0,0)
    cstart = conference.Conf_Start_Time
    cend = conference.Conf_End_Time
    clen = conference.Conf_Length
    cstart_date = conference.Conf_Date_Start
    cend_date = conference.Conf_Date_End
    croom_count = conference.Conf_Room_Count
    slices_morning = functions.fill_time_values(clen,cstart , noon)
    slices_afternoon = functions.fill_time_values(clen, afternoon, cend)
    presentations = Presentation.objects.filter(Conference = conf_id)
    days = functions.get_days(cstart_date, cend_date)
    #context data
    template_data['morning_slices'] = slices_morning
    template_data['afternoon_slices'] = slices_afternoon
    template_data['days'] = range(days)
    template_data['span'] = int(10 / croom_count)
    template_data['span_rest'] = 12-(template_data['span']*croom_count)
    template_data['rooms'] = range(croom_count)
    template_data['schedule'], template_data['count'] =
calculate_ant_colony.calculate_without_recursion(conference, presentations,
restrictions, slices_morning, slices_afternoon, days,1 )
    template_data['morn'] = len(slices_morning)
    template_data['Presentations'] = presentations
    after = t()
    delta = after - before
    print("page took " + str(delta) + " seconds to load")
    template_data['timing'] = str(delta) + " seconds"
    return render_to_response('calculate.html', template_data,
context_instance=RequestContext(request))
```

Conference details template (_details.html)

```
{% extends "base.html" %}
{% block title %}Conference Details{% endblock %}
{% block content %}
    <div class="row">
        <div class="span12">
            <h1> Presentations for conference </h1>
        </div>
        <div class="span12" style="padding-bottom:300px;">
        {% if Presentations %}
        {% include "presentation_details.html" %}
        <br /><a href="calculate/">Run conference planner</a>
        </div>
    {% else %}
        There are no presentations in this conference yet!<br />
        <a href="../"> Other Conferences </a>
    {% endif %}
        </div>
    <div>

{% endblock %}
```

Template base (base.html)

```
<!DOCTYPE html>
<html>
    <head profile="http://www.w3.org/2005/10/profile">
                    <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <meta name="description" content="Thesis project for Efe Acikgoz" />
        <meta name="author" content="Efe Acikgoz" />
        <title>{% block title %}{% endblock %} - Conference Planner</title>
        <link href="/static/css/base.css" rel="stylesheet" />
        <link href="/static/css/bootstrap.min.css" rel="stylesheet" />
        <link
href='http://fonts.googleapis.com/css?family=Yanone+Kaffeesatz:400,300,700'
rel='stylesheet' type='text/css' />
    </head>
    <body>
        <div class="container cont">
            {% spaceless %}{% block content %} {% endblock %}{% endspaceless %}
        </div>
    </body>
</html>
```

Schedule screen (calculate.html)

```
{% extends "base.html" %}
{% block title %}Conference Plan{% endblock %}
{% block content %}
            {% comment %}
            <div class="row">
                <div class="span12">
                    <h1> Calculation </h1><br />
                    {% for day in days %}
                    <div class="row">
                        <div class="span{{span_rest}}">
                            <span
style="background:url(/static/img/date.png);height:63px;width:58px;float:left;margin-
bottom:2px; margin-right:5px;">
                                <h2 style="margin-left:15px;">
                                    <font color="white" style=" text-shadow: 0px 1px 1px #000;">
                                        Day {{ forloop.counter }}
                                    </font>
                                </h2>
                            </span>
                        </div>
                        {% for room in rooms %}
                        <div class="span{{ span }}">
                            <h2> Room {{ forloop.counter }} </h2>
                        </div>
                        {% endfor %}f
                        {% for slice in morning_slices %}
                            <div class="span{{span_rest}}">
                                {{ slice }}
                            </div>
                            {% include "rooms.html" %}
                        {% endfor %}
                        <div class="span12">
                        <hr class="style-two">
                        </div>
                        {% for slice in afternoon_slices %}
                            <div class="span{{span_rest}}">
                                {{ slice }}
                            </div>
                            {% include "rooms.html" %}
                        {% endfor %}
                    </div><br /><br />
                    {% endfor %}
                </div>
            </div>
            {% endcomment %}

            <div class="row">
                <div class="span12">
                    <h1> Calculations took: {{count}} in {{timing}}</h1><br />
                    {% for day in days %}
                    <div class="row">
                        <div class="span{{span_rest}}">
                            <span class="date_img">
                                <h2>
                                    <font size="4" color="white" style="margin-top:5px;padding-
        left:5px;text-shadow: 0px 1px 1px #000;">
                                        Day {{ forloop.counter }}
                                    </font>
                                </h2>
                            </span>
                        </div>
                        {% for room in rooms %}
                        <div class="span{{ span }}">
                            <h2> Room {{ forloop.counter }} </h2>
                        </div>
                        {% endfor %}
                        {% spaceless %}{% for slice in morning_slices %}
                            <div class="span{{span_rest}}">
                                {{ slice }}
                            </div>
                            {% include "calculated_rooms_morning.html" %}
                        {% endfor %}{% endspaceless %}
                        <div class="span12">
                        <hr class="style-two">
                        </div>
                        {% spaceless %}{% for slice in afternoon_slices %}
                            <div class="span{{span_rest}}">
                                {{ slice }}
                            </div>
                            {% include "calculated_rooms_afternoon.html" %}
                        {% endfor %}{% endspaceless %}
                    </div><br /><br />
```

```
                {% endfor %}
            </div>
        </div>
        {% include "presentation_details.html" %}
{% endblock %}
```

Calculated room (calculated_rooms)

```
                    {% spaceless %}
                    {% for room in rooms %}
                    <div class="span{{ span }}">
                    {% for plan in schedule %}{% if day = plan.day and room =
plan.room %}{% if plan.afternoon == "True" %}{% if plan.time =
forloop.parentloop.parentloop.counter0 %}
                                {{ plan.name }}
                    {% endif %}{% endif %}{% endif %}{% endfor %}
                    &nbsp;</div>
                    {% endfor %}{% endspaceless %}
```

Conference list (conferences.html)

```
{% extends "base.html" %}
{% block title %}Conferences{% endblock %}
{% block content %}
   <div class="row">
      <div class="span12" style="padding-bottom:250px;">
         <h1> Conferences </h1>
         {% for conference in conferences %}
            <h3><a href="{{ conference.id }}/"> {{ conference.Conference_Name }}
</a><br /></h3>
         {% endfor %}
      </div>
   </div>
{% endblock %}
```

presentation details page (presentation_details)

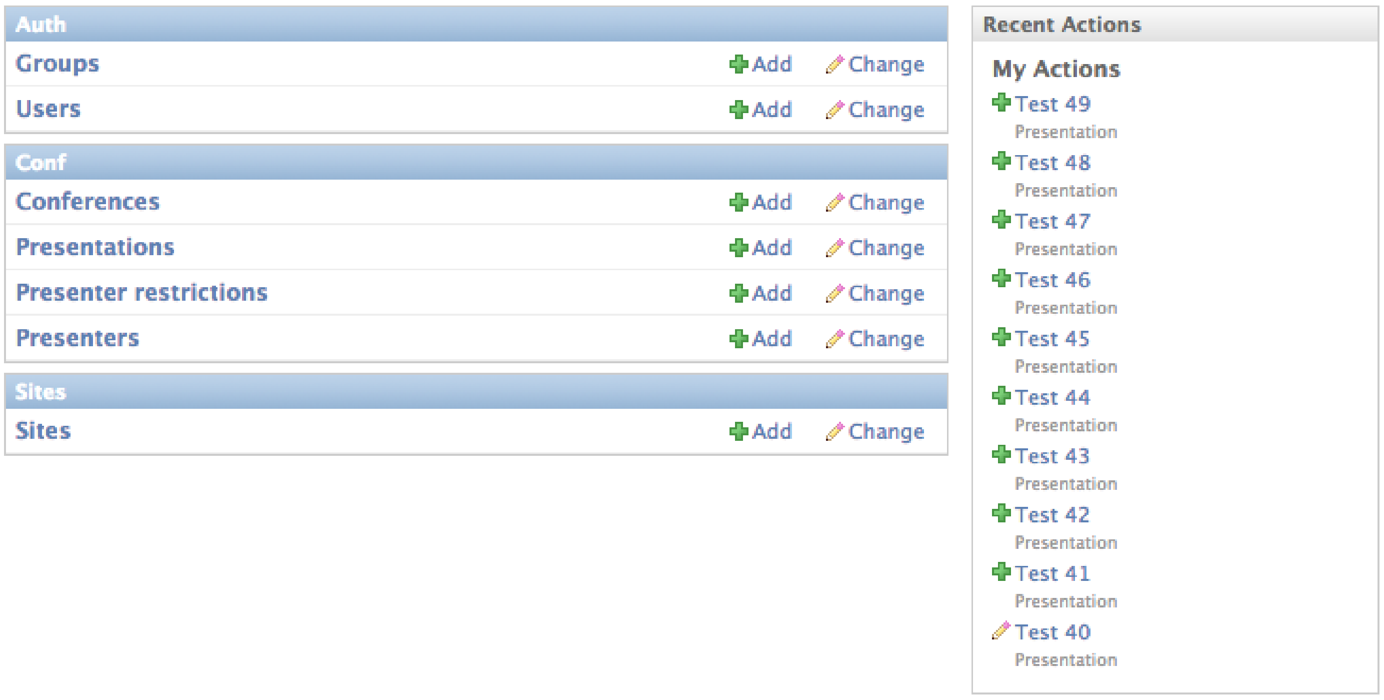
```
      <a class="btnShow">Show Details</a>
      <a class="btnHide">Hide Details</a>

      <div id="details">
         <div class="row">
            {% for Presentation in Presentations %}
               <div class="presentation">
                  <h4>{{ Presentation.Presentation_Name }}</h4>
                     {% for Presenter in Presentation.Name.all %}
                        {{ Presenter }}<br />
                     {% endfor %}
               </div>
            {% endfor %}
        </div>
    </div>
    <script src="/static/js/jquery-1.5.1.min.js"></script>
    <script src="/static/js/jquery.masonry.min.js"></script>
    <script>
        $(document).ready(function(){
            $('#details').masonry({
                                itemSelector: '.presentation',
                                        columnWidth : 230,
                                });
            $("#details").hide();
            $(".btnHide").hide();
            $(".btnShow").click(function() {
                $("#details").fadeIn(1000);
                $(".btnShow").hide();
                $(".btnHide").show();
            });
            $(".btnHide").click(function() {
                $("#details").fadeOut(1000);
                $(".btnHide").hide();
                $(".btnShow").show();
            });
        });
    </script>
```

Settings (settings.py)

```
# Django settings for thesis project.
import os.path

PROJECT_ROOT = os.path.abspath(os.path.dirname(__file__))
DEBUG = True
TEMPLATE_DEBUG = DEBUG

ADMINS = (
    # ('Your Name', 'your_email@example.com'),
)

MANAGERS = ADMINS

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3', # Add 'postgresql_psycopg2', 'mysql',
'sqlite3' or 'oracle'.
        'NAME': PROJECT_ROOT+'/db',                    # Or path to database file if using
sqlite3.
        'USER': '',                    # Not used with sqlite3.
        'PASSWORD': '',                  # Not used with sqlite3.
        'HOST': '',                  # Set to empty string for localhost. Not used with sqlite3.
        'PORT': '',                  # Set to empty string for default. Not used with sqlite3.
    }
}

TIME_ZONE = 'Europe/Istanbul'

LANGUAGE_CODE = 'en-us'
SITE_ID = 1
USE_I18N = True
USE_L10N = True
USE_TZ = True
MEDIA_ROOT = ''
MEDIA_URL = ''
STATIC_ROOT = ''
STATIC_URL = '/static/'

STATICFILES_DIRS = (
    os.path.join(PROJECT_ROOT, 'static'),
)

STATICFILES_FINDERS = (
    'django.contrib.staticfiles.finders.FileSystemFinder',
    'django.contrib.staticfiles.finders.AppDirectoriesFinder',
)

SECRET_KEY = '&amp;kt14m%dpz=f4qhx5wafsrx=h0uvxst(l45n3a43i9$$zo9$3d'

TEMPLATE_LOADERS = (
    'django.template.loaders.filesystem.Loader',
    'django.template.loaders.app_directories.Loader',
)

MIDDLEWARE_CLASSES = (
    'django.middleware.common.CommonMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',

)

ROOT_URLCONF = 'urls'

WSGI_APPLICATION = 'wsgi.application'

TEMPLATE_DIRS = (
    os.path.join(PROJECT_ROOT, "templates").replace('\\','/'),
)

INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.admin',
    'conf',
)


LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'filters': {
        'require_debug_false': {
            '()': 'django.utils.log.RequireDebugFalse'
        }
    },
    'handlers': {
        'mail_admins': {
            'level': 'ERROR',
            'filters': ['require_debug_false'],
            'class': 'django.utils.log.AdminEmailHandler'
        }
    },
    'loggers': {
        'django.request': {
            'handlers': ['mail_admins'],
            'level': 'ERROR',
            'propagate': True,
        },
    }
}
```

Appendices A.2 - Screenshots:

Admin pages:

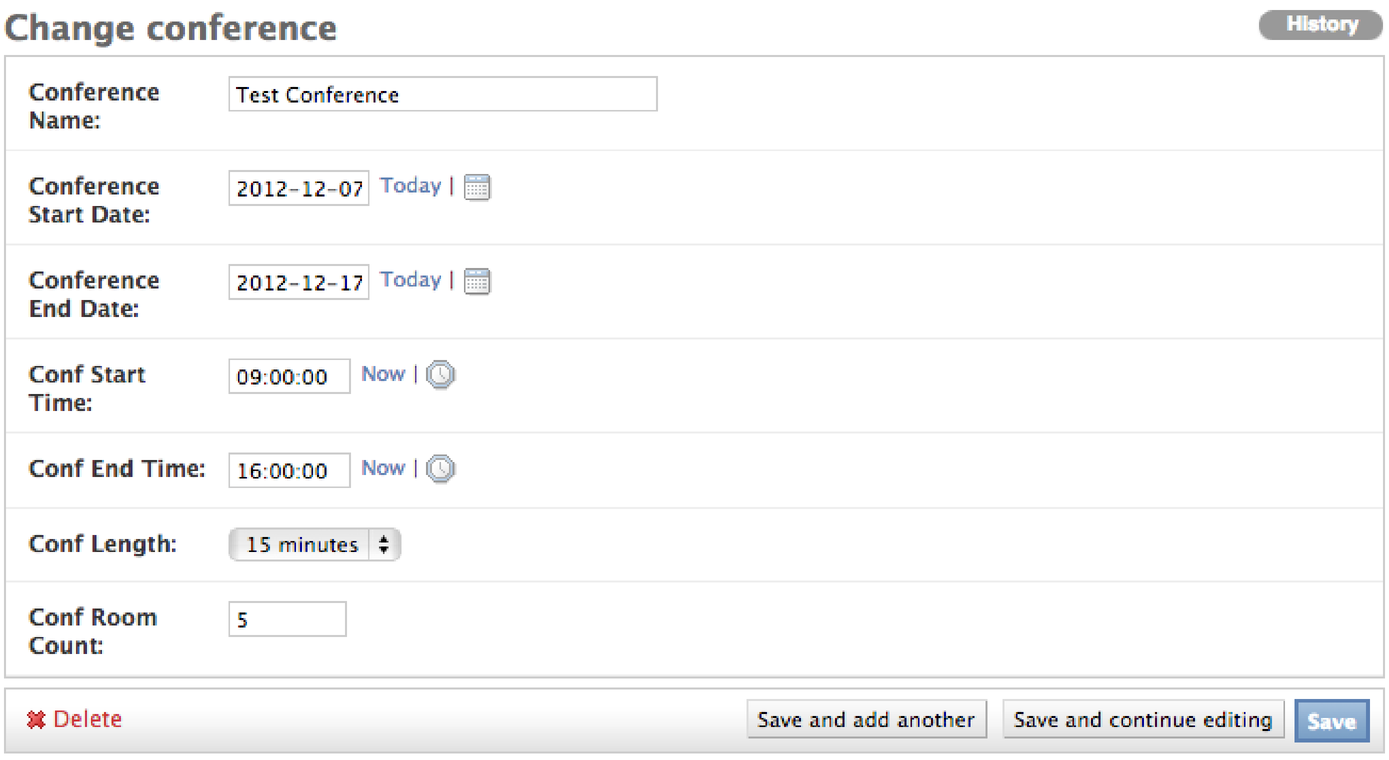


Auth		
Groups	Add	Change
Users	Add	Change
Conf		
Conferences	Add	Change
Presentations	Add	Change
Presenter restrictions	Add	Change
Presenters	Add	Change
Sites		
Sites	Add	Change

Recent Actions

My Actions
- Test 49 — Presentation
- Test 48 — Presentation
- Test 47 — Presentation
- Test 46 — Presentation
- Test 45 — Presentation
- Test 44 — Presentation
- Test 43 — Presentation
- Test 42 — Presentation
- Test 41 — Presentation
- Test 40 — Presentation

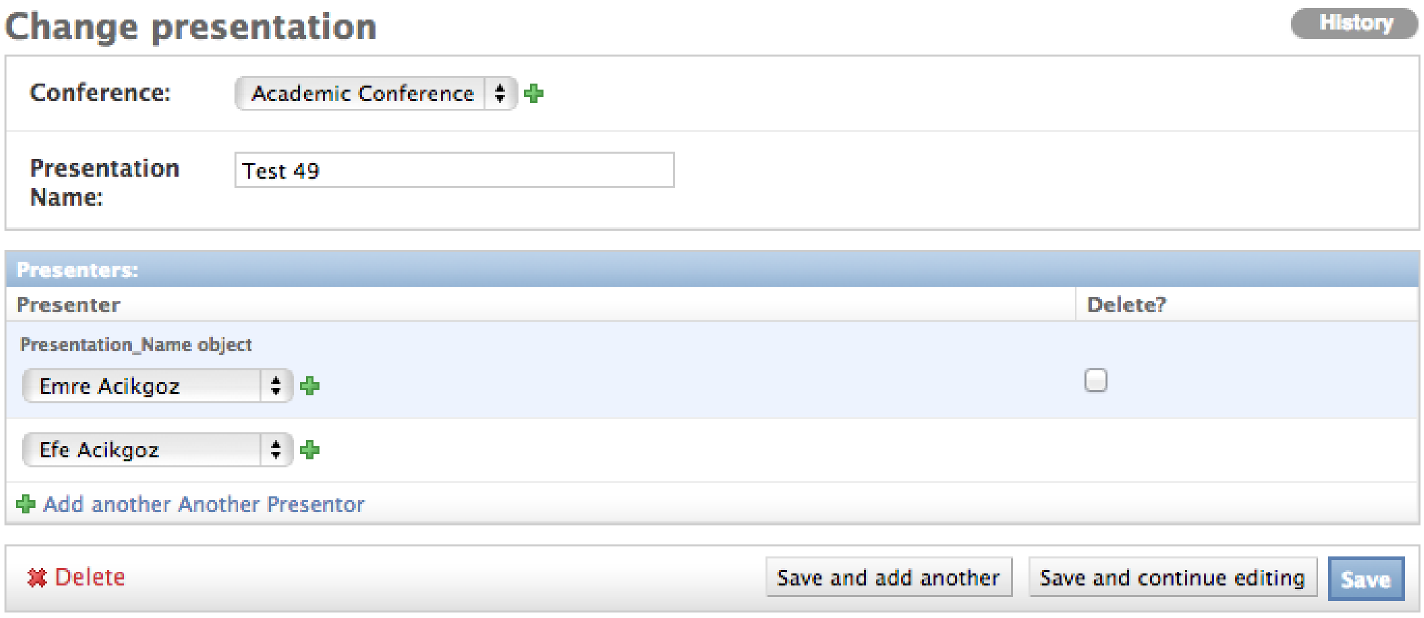
Main administration page



Change conference

Conference Name:	Test Conference
Conference Start Date:	2012-12-07 Today |
Conference End Date:	2012-12-17 Today |
Conf Start Time:	09:00:00 Now |
Conf End Time:	16:00:00 Now |
Conf Length:	15 minutes
Conf Room Count:	5

Conference admin page



Change presentation

History

Conference:	Academic Conference
Presentation Name:	Test 49

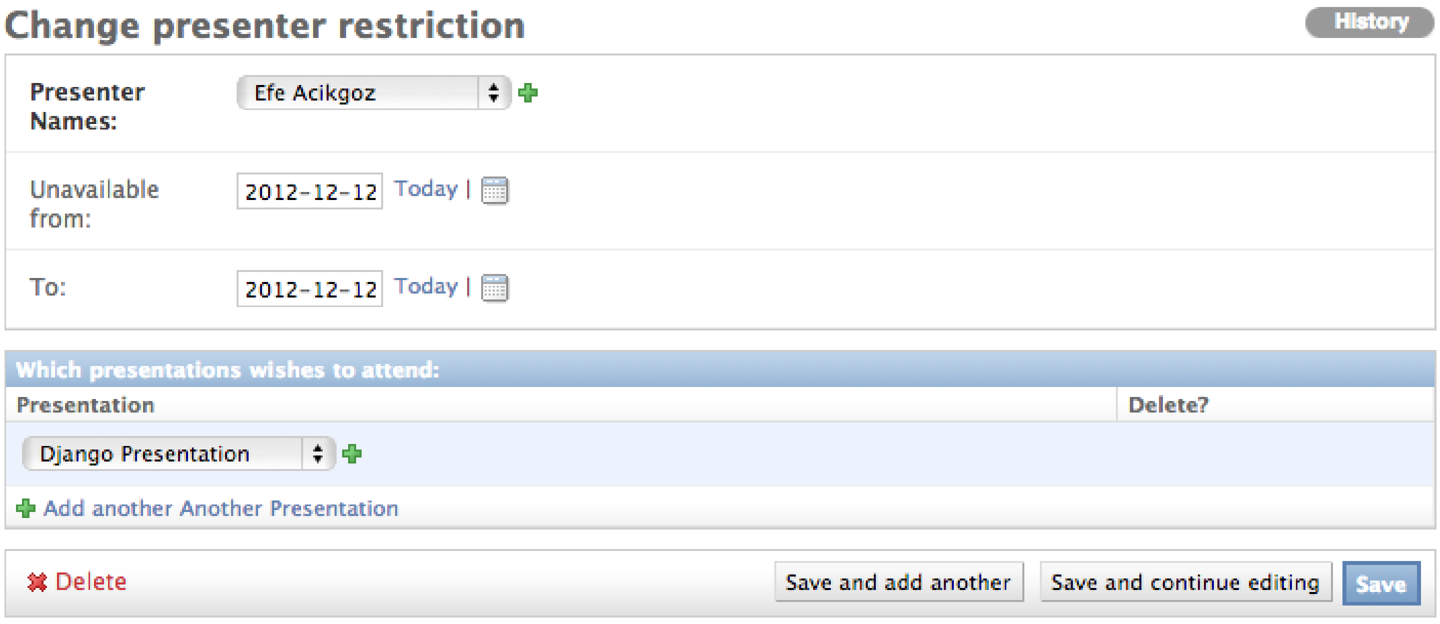
Presenters:

Presenter	Delete?
Presentation_Name object Emre Acikgoz	☐
Efe Acikgoz	

Add another Another Presentor

Delete | Save and add another | Save and continue editing | Save

Presentation admin page



Change presenter restriction

History

Presenter Names:	Efe Acikgoz
Unavailable from:	2012-12-12 Today |
To:	2012-12-12 Today |

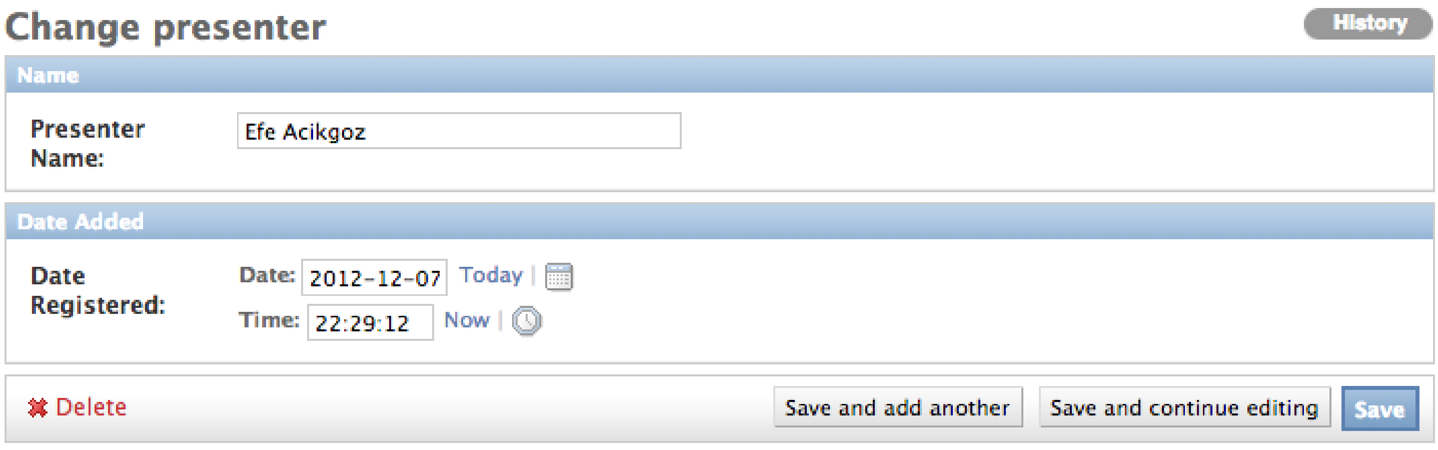
Which presentations wishes to attend:

Presentation	Delete?
Django Presentation	

Add another Another Presentation

Delete | Save and add another | Save and continue editing | Save

Presentation restrictions admin page


Change presenter

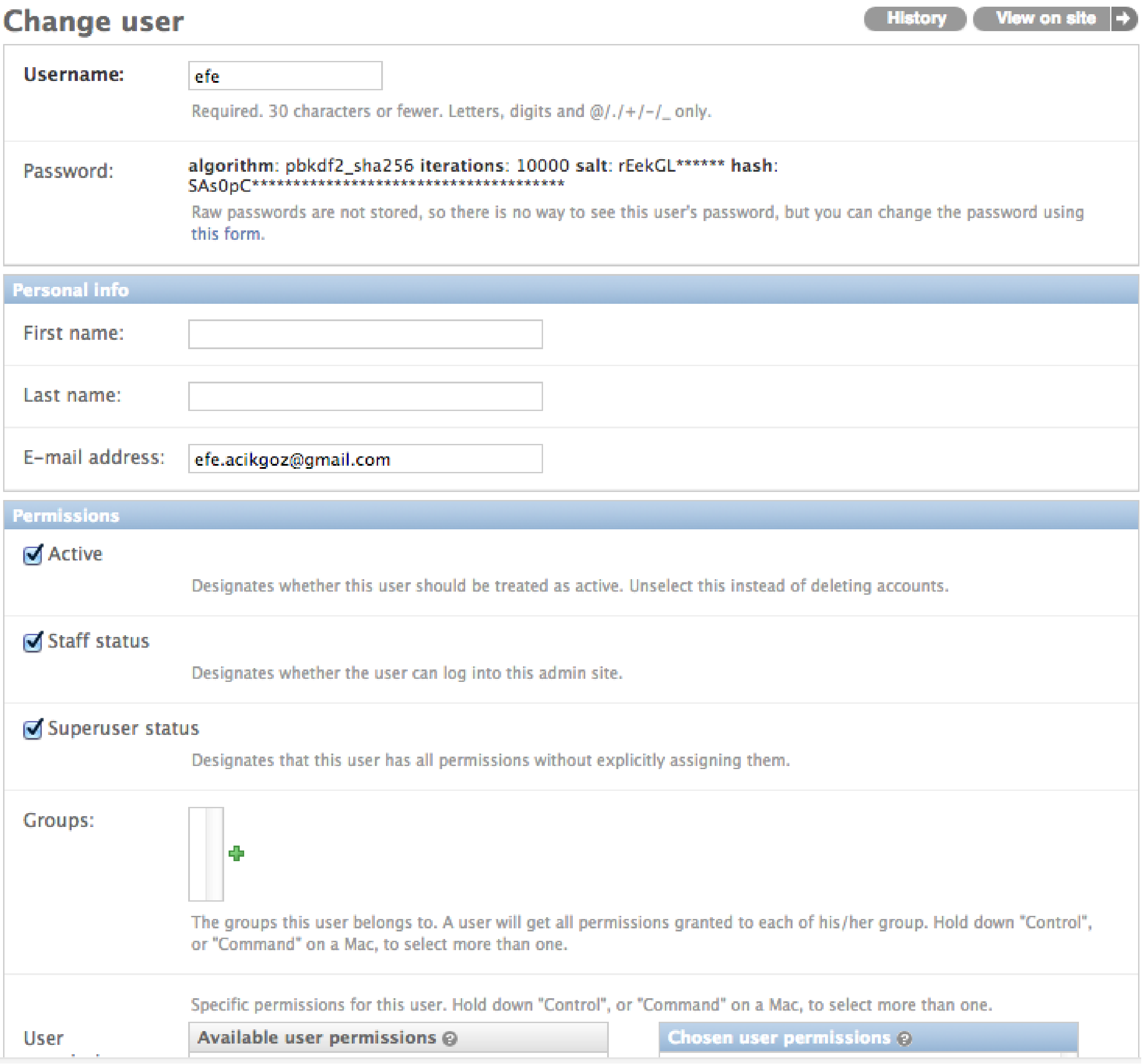
Name

Presenter Name: Efe Acikgoz

Date Added

Date Registered: Date: 2012-12-07 Today | Time: 22:29:12 Now |

Presenter admin page


Change user

Username: efe

Required. 30 characters or fewer. Letters, digits and @/./+/-/_ only.

Password: **algorithm**: pbkdf2_sha256 **iterations**: 10000 **salt**: rEekGL****** **hash**: SAs0pC**

Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.

Personal info

First name:

Last name:

E-mail address: efe.acikgoz@gmail.com

Permissions

☑ Active

Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

☑ Staff status

Designates whether the user can log into this admin site.

☑ Superuser status

Designates that this user has all permissions without explicitly assigning them.

Groups:

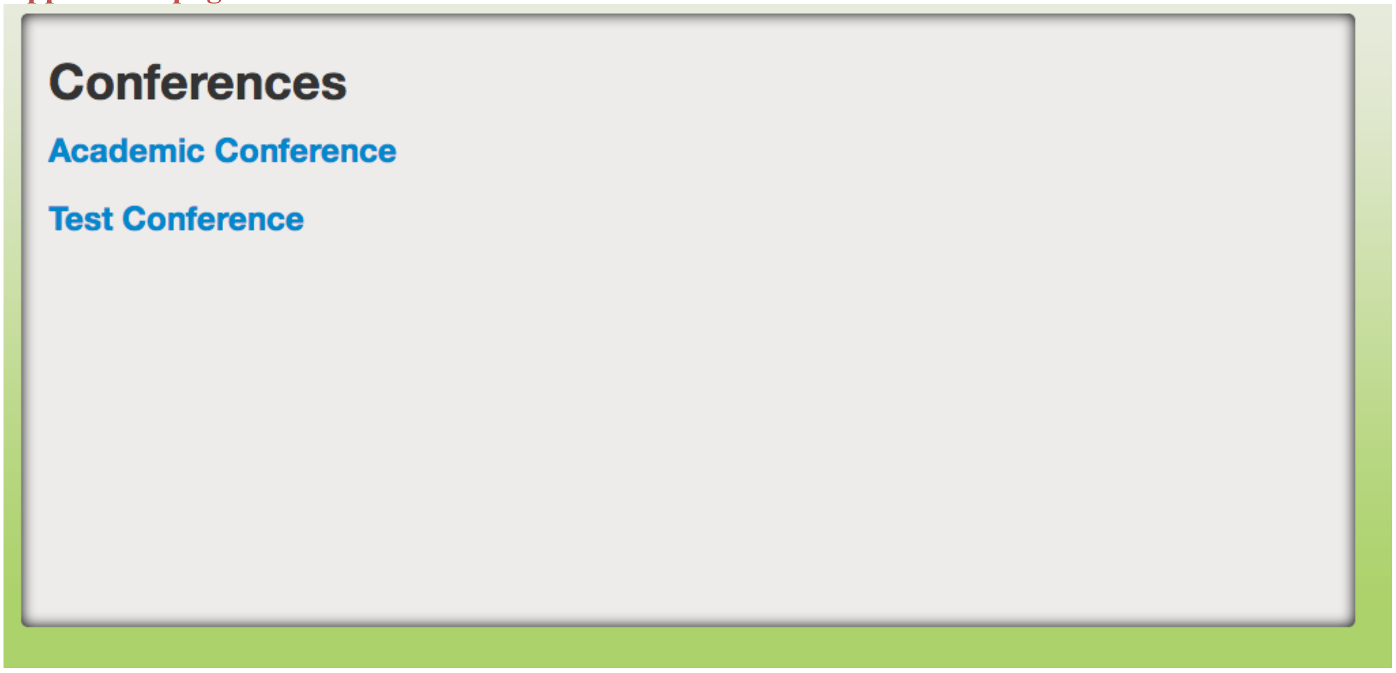
The groups this user belongs to. A user will get all permissions granted to each of his/her group. Hold down "Control", or "Command" on a Mac, to select more than one.

Specific permissions for this user. Hold down "Control", or "Command" on a Mac, to select more than one.

User

Admin user details page

Application pages:

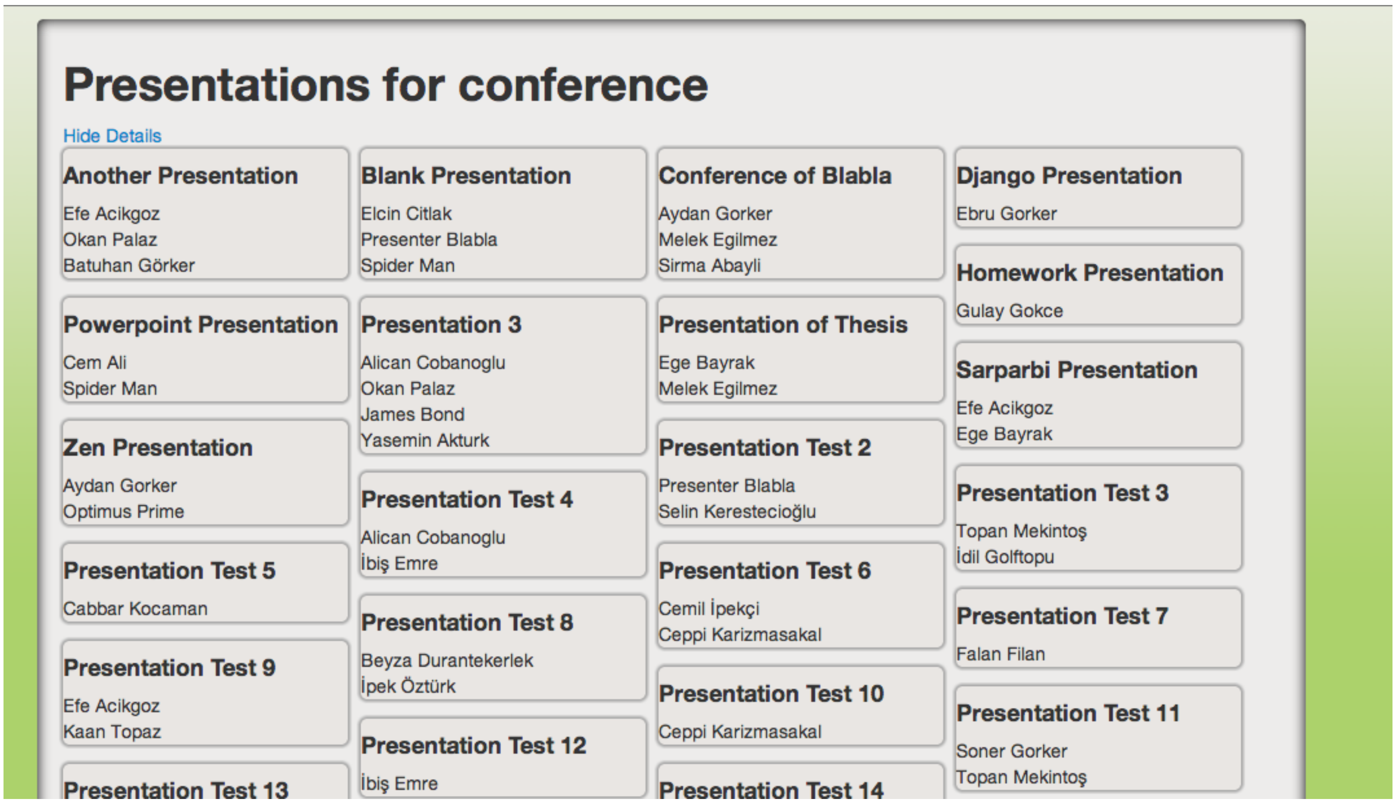


Conferences

Academic Conference

Test Conference

Conferences list page



Presentations for conference

Hide Details

Another Presentation
Efe Acikgoz
Okan Palaz
Batuhan Görker

Blank Presentation
Elcin Citlak
Presenter Blabla
Spider Man

Conference of Blabla
Aydan Gorker
Melek Egilmez
Sirma Abayli

Django Presentation
Ebru Gorker

Homework Presentation
Gulay Gokce

Powerpoint Presentation
Cem Ali
Spider Man

Presentation 3
Alican Cobanoglu
Okan Palaz
James Bond
Yasemin Akturk

Presentation of Thesis
Ege Bayrak
Melek Egilmez

Sarparbi Presentation
Efe Acikgoz
Ege Bayrak

Zen Presentation
Aydan Gorker
Optimus Prime

Presentation Test 2
Presenter Blabla
Selin Kerestecioğlu

Presentation Test 4
Alican Cobanoglu
İbiş Emre

Presentation Test 3
Topan Mekintoş
İdil Golftopu

Presentation Test 5
Cabbar Kocaman

Presentation Test 6
Cemil İpekçi
Ceppi Karizmasakal

Presentation Test 8
Beyza Durantekerlek
İpek Öztürk

Presentation Test 7
Falan Filan

Presentation Test 9
Efe Acikgoz
Kaan Topaz

Presentation Test 10
Ceppi Karizmasakal

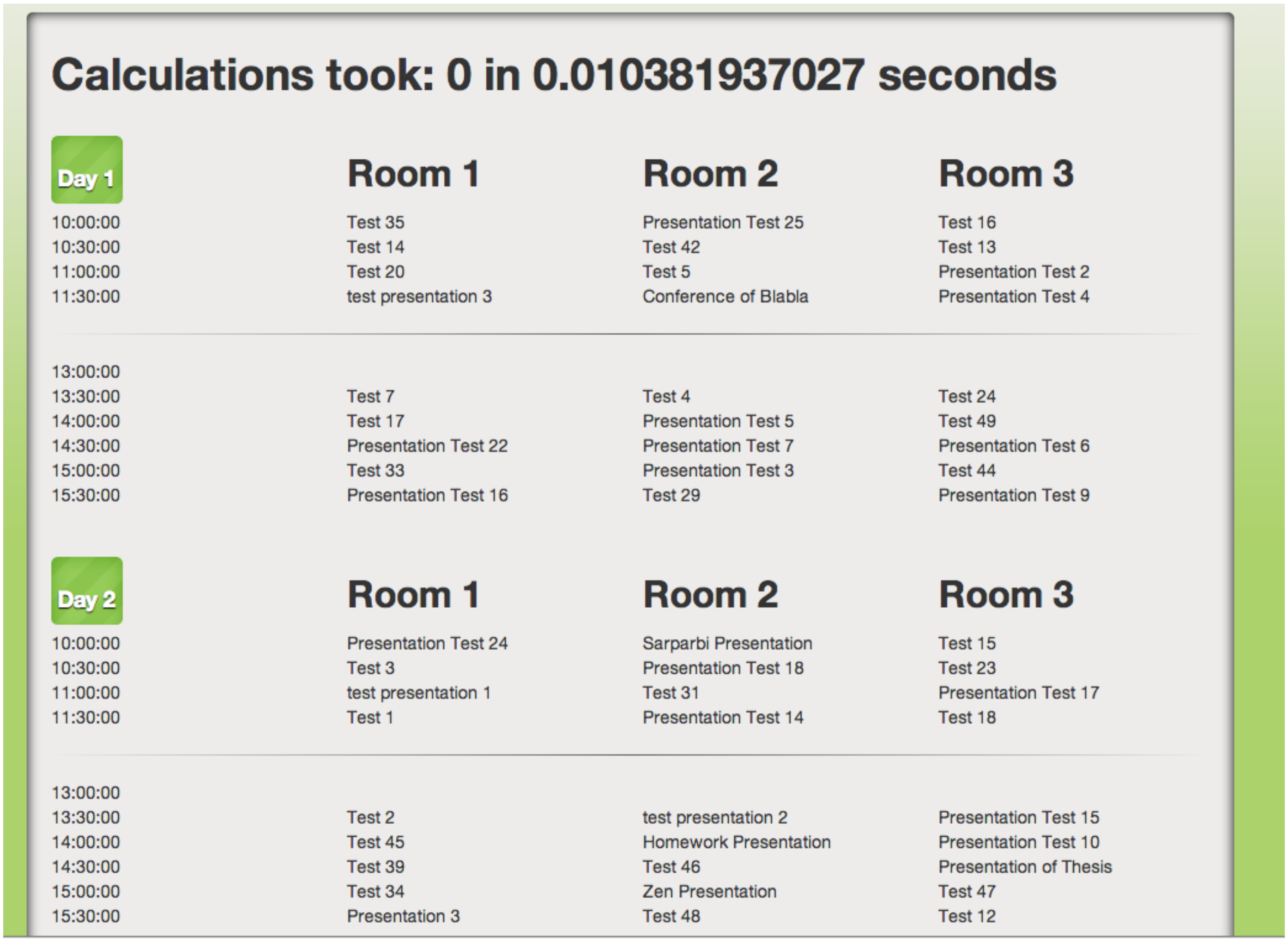
Presentation Test 11
Soner Gorker
Topan Mekintoş

Presentation Test 12
İbiş Emre

Presentation Test 13

Presentation Test 14

Conference details



Calculations took: 0 in 0.010381937027 seconds

Day 1	Room 1	Room 2	Room 3
10:00:00	Test 35	Presentation Test 25	Test 16
10:30:00	Test 14	Test 42	Test 13
11:00:00	Test 20	Test 5	Presentation Test 2
11:30:00	test presentation 3	Conference of Blabla	Presentation Test 4
13:00:00			
13:30:00	Test 7	Test 4	Test 24
14:00:00	Test 17	Presentation Test 5	Test 49
14:30:00	Presentation Test 22	Presentation Test 7	Presentation Test 6
15:00:00	Test 33	Presentation Test 3	Test 44
15:30:00	Presentation Test 16	Test 29	Presentation Test 9

Day 2	Room 1	Room 2	Room 3
10:00:00	Presentation Test 24	Sarparbi Presentation	Test 15
10:30:00	Test 3	Presentation Test 18	Test 23
11:00:00	test presentation 1	Test 31	Presentation Test 17
11:30:00	Test 1	Presentation Test 14	Test 18
13:00:00			
13:30:00	Test 2	test presentation 2	Presentation Test 15
14:00:00	Test 45	Homework Presentation	Presentation Test 10
14:30:00	Test 39	Test 46	Presentation of Thesis
15:00:00	Test 34	Zen Presentation	Test 47
15:30:00	Presentation 3	Test 48	Test 12

Conference schedule

ÖZGEÇMİŞ

Adı Soyadı :Efe Açıkgöz

Sürekli Adresi : Salnameci Sok. H. Suleyman Temur Apt. No:45 D:9 Şişli/İstanbul

Doğum Yeri ve Yılı : Balıkesir 07/03/1988

Yabancı Dili : İngilizce

İlk Öğretim : Atatürk İlköğretim Okulu 2002

Orta Öğretim : Rahmi Kula Anadolu Lisesi 2006

Lisans : Bahçeşehir Üniversitesi 2010

Yüksek Lisans : Bahçeşehir Üniversitesi

Enstitü Adı : Fen Bilimleri Enstitüsü

Program Adı : Bilgi Teknolojileri

Çalışma Hayatı :

Apperto 2012 – Devam

Mobinex 2010 – 2012