A Classroom Scheduling Service for Smart Classes

Chao Wang, *Member, IEEE*, Xi Li, *Member, IEEE*, Aili Wang, and Xuehai Zhou, *Member, IEEE*

Abstract—During past decades, the classroom scheduling problem has posed significant challenges to educational programmers and teaching secretaries. In order to alleviate the burden of the programmers, this paper presents SmartClass, which allows the programmers to solve this problem using web services. By introducing service-oriented architecture (SOA), SmartClass is able to provide classroom scheduling services with back-stage design space exploration and greedy algorithms. Furthermore, the SmartClass architecture can be dynamically coupled to different scheduling algorithms (e.g. Greedy, DSE, etc.) to fit in specific demands. A typical case study demonstrates that SmartClass provides a new efficient paradigm to the traditional classroom scheduling problem, which could achieve high flexibility by software services reuse and ease the burden of educational programmers. Evaluation results on efficiency, overheads and scheduling performance demonstrate the SmartClass has lower scheduling overheads with higher efficiency.

Index Terms—Services-oriented, classroom scheduling, resources modeling

1 Introduction

The classroom scheduling problem has been regarded as one of the most important challenges for educational programmers and teaching secretaries. In most cases, this is still a manual process, especially in the developing countries with a large amount of students and lectures. During the past decades, the problem has been concentrated and widely conducted using computer-aid design (CAD) methodology. However, due to that the classroom scheduling has been long proved to be NP-Complete problem [1], thereby the focus of current researches are shifting towards more practical instead of optimal technical sound solutions.

Unfortunately, the growing complexity of the input factors is posing significant challenges to solve the classroom scheduling problem. Several significant constrained factors must be considered in the modeling: such as timing, teacher, lecture, classroom, and students. Meanwhile, there are also additional constraints such as the multimedia requirements and seats constraints. It takes quite a lot of time and effort to make a practical and high efficient classroom scheduling plan.

Learning from the current state-of-the-art disciplines, the classroom scheduling problem is encountering following serious challenges:

1) First, at a specific time, the classroom is solely mapped to a certain class. That is, each class could be mapped to only one room, while it is also preferable to allocate all the lectures from a same class to the exactly same classroom. This potential requirement has increased the complexity of the classroom scheduling scheme;
2) Second, if a conflict appears, e.g. no more available classrooms, or no more available time slots, the scheduling method needs to roll back to previous solutions, which has a strong effect on the partial finished scheduling plan, thereby may bring new conflicts;
3) Last but not least, the additional constraints, such as the seat number and multimedia requirements also need special consideration in the scheduling method.

To address the above problems, numerous researches have been devoted to the scheduling problem. Most state-of-the-art approaches normally utilize one or multiple following methodologies: integer linear programming (ILP), greedy algorithm, simulator approach, graph theory, or Lagrange etc. Due to the complex maze of the constraints, most computing-aided-design methodologies encounter the design-space-exploration problem; therefore how to design a practical and scalable scheme is still one of the serious challenges of the problem.

Considering the high-level overview, the classroom scheduling problem can be modeled as a traditional resource scheduling problem. Regarding all the classroom resources as function units, a group of lectures are treated as a series of tasks. Furthermore, in order to reduce the maze, the problem can be divided into two sub issues: classroom scheduling and time slot scheduling, respectively. The time slot scheduling is similar to the conventional timetabling issue, the main contribution is to calculate the timetable for the lectures. Compared to the time slot scheduling, the classroom scheduling is more flexible with a number of constraints posed to the designers. In particular, the classroom scheduling needs to consider more constrained variables, such as the capacity of the classroom, the enrolled student number, the multimedia facilities of the classroom, and the teacher's optional preference. These parameters

• *The authors are with the University of Science and Technology of China, Hefei 230027, Anhui, China.*
E-mail: saintwc@mail.ustc.edu.cn, {llxx, wangal, xhzhou}@ustc.edu.cn.

Manuscript received 27 Dec. 2014; revised 5 Apr. 2015; accepted 28 May 2015. Date of publication 12 June 2015; date of current version 7 Apr. 2017.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSC.2015.2444849

significantly increase the complexity of the scheduling problem, which is quite time consuming to be calculated in the client. Thus given the fixed time slot parameters, how to efficiently allocate the classrooms with the constraints is becoming one of the key issues to be figured out.

Another challenge is that if the complex computation is solved by the educational programmer, then the quality of scheduling depends on the experiences of the programmer, the efficiency of the algorithm, and also the speed of the computer. In order to alleviate the burden of the terminal users, we can demonstrate the effectiveness of services-oriented architecture (SOA) [2] in classroom scheduling research paradigm. Traditionally SOA provides effective measures with better flexibility and extensibility at lower cost through adopting re-usable modules in software engineering. From the exploration of the benefits of SOA concepts, we can conclude that there are at least two significant advantages through integrating SOA concepts into classroom scheduling. First, it can largely reduce the design complexity for classroom scheduling. This feature allows user to employ a light-weight client, while all the scheduling threads running on the server will map the different computing resources as services. Second, due to the advantages of SOA, the flexibility of the scheduling could be improved. It can be easily adopted to fit in different level of requirements.

To our best knowledge, few researches use SOA concepts to solve the classroom scheduling problem. In this paper, SmartClass introduces SOA concept to classroom scheduling problem, and constructs a Services-Oriented framework for classroom scheduling. The contributions of this paper are listed as below:

1) First, SmartClass provides a light-weight classroom scheduling architectural framework to alleviate the burden of educational programmers.
2) Second, SmartClass regards different resources as services, including classroom, student number, teacher, interval and multimedia factors.
3) Finally, in order to efficiently reduce the complexity, a greedy algorithm based approach is incorporated to the SmartClass framework.

The rest of the paper is organized as follows. Related work and motivation is summarized in Section 2. Section 3 discusses the architecture and main concepts of SmartClass, including architecture, processing flow, organization and hierarchical models. The greedy based classroom scheduling algorithm is also presented. Thereafter Section 4 describes the demonstration of SmartClass case studies. Then Section 5 presents the detailed evaluation results of efficiency and overheads. Finally, conclusion and further work are presented in Section 6.

2 Motivation and Related Work

2.1 Problem Definition

The classroom scheduling problem is defined to find a solution that efficiently assigns a number of lectures to classrooms taking into consideration constraints like classroom capacities and university regulations. The problem also attempts to optimize the performance criteria and distribute the courses fairly to classrooms depending on the ratio of classroom capacities to course enrollments. The problem is a classical scheduling problem and considered to be NP-complete [1]. In prior to describing the architecture of SmartClass, we need to figure out all the considerable constraints.

First, the scheduling method includes the necessary following constraints:

- A teacher cannot teach two classes at the same time;
- A student cannot follow two lectures at the same time;
- Some teachers must have at least one day off during the week (Optional);
- All the days of the week should be covered by the time table;
- Subject X must have exactly so-and-so hours each week;
- The seat number and multimedia demands should be met.

Second, optional preferences should also be considered at the same time:

- Each teachers schedule should be as compact as possible (i.e., the teacher should work all hours for the day in a row with no pauses if possible);
- Teachers who have days off should be able to express a preference on which day;
- Teachers who work part-time should be able to express a preference whether to work in the beginning or the end of the school day.

2.2 Related Work

Classroom based research has been widely conducted during the past decades. Of these state-of-the-art literatures, many formulations and algorithms have been proposed to solve the traditional classroom scheduling problem. For example, [3] presents a Flat Fading based method for Classroom Instruction. However, as the authors use MATLAB for simulation, therefore the computational complexity could drag down the performance dramatically. In an Upper-Division Engineering Course, [4] compares the effectiveness of an inverted classroom to a traditional classroom. Meanwhile, considerable number of algorithms are based on local search techniques, namely hill climbing, simulated annealing, and tabu search [5], [6]. Such algorithms cannot prove unsatisfiability or guarantee that a solution is optimal. In other words, if a solution is found, it cannot guarantee that this solution has the best possible optimization cost.

Recently, [7] proposes an integer linear programming approach to solve the scheduling problem. The ILP model is developed and solved using the three advanced ILP solvers based on generic algorithms and Boolean Satisfiability (SAT) techniques [8]. The SAT problem is an important problem in artificial intelligence and computer science and has received considerable attention from researchers. The authors have also successfully solved many complex engineering problems using SAT including routing [9], power optimization [10], and graph coloring [11], etc.

Simultaneously, the graph theory based approaches are still focused on the scheduling problem. For example, [12] and [13] both conduct a graph coloring scheme in scheduling. Of the two approaches, [12] propose a heuristic approach to promote uniform distribution of courses over the colors and also to balance course load for each time slot over the available class rooms. In contrast, [13] present an alternative graph coloring method for university timetabling that incorporates room assignment during the coloring process. Similarly, in [14] a meta-heuristic algorithm based on the principles of particle swarm optimization (PSO) is proposed for course scheduling problem.

Meanwhile, Choice Function (CF) [15] and Simple Random (SR) [16] techniques have been widely used to investigate the performance of different hyper-heuristics on the classroom scheduling problem. The authors in [17] investigated that the hyper-heuristic with CF techniques achieved the best performance comparing with other similar approaches.

To sum up, most state-of-the-art related works have serious drawbacks: (1) most of the scheduling approaches are performed with high-complex algorithms, which are not an effective or even practical solution for NP-Complete scheduling; (2) the approaches of the scheduling schemes are lack of modularity, which will cause excessive and inefficient workloads for different objective of scheduling methods. What's worse, when some factors are modified (e.g. lecture switching in the classroom scheduling problem), the algorithm must be reconsidered from the beginning, which brings significant redesign and simulation overheads.

2.3 The Novel Features of SmartClass

To tackle the above problems, this paper introduces SOA concepts into classroom scheduling problem [18], to construct a service-oriented scheduling framework SmartClass. In this paper we present a services-based classroom scheduling mechanisms to utilize SOA architecture's benefits. The mechanism of classroom scheduling is implemented with the mapping scheme between resources and services. Based on this mechanism, in the paper we propose a classroom scheduling prototype supporting following features:

1) First, SmartClass provides a light-weight classroom scheduling architectural framework optimized with multiple factors and targets in particular. This could facilitate the educational programmers by easing the burden of classroom scheduling.
2) Second, on the server side SmartClass regards resources as services, including classroom, student number, teacher, interval and multimedia. The software services could be reused for fast prototyping.
3) Third, the scheduling algorithm is invisible to terminal users. In order to efficiently reduce the complexity, greedy algorithm and design space exploration method are presented to maintain the scheduling plan, respectively. Also other scheduling methods can be easily adapted to SmartClass framework.

3 SmartClass Architecture and Concepts

In this section, we first introduce the SmartClass architecture model, followed by the SmartClass processing flow. Thereafter, we introduce the underlying hierarchical layers and the integrated services. Subsequently, we detail the proposed algorithm to solve the classroom scheduling problem.

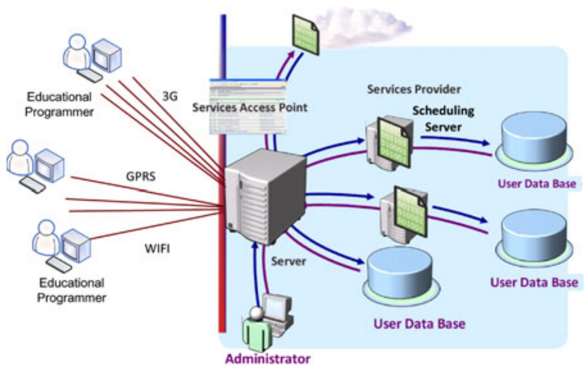


Fig. 1. Architecture framework for SmartClass.

3.1 Architecture Framework for SmartClass

Fig. 1 illustrates the high level architecture framework for SmartClass. The main components of the system are deployed as follows:

First, the classroom and user information are stored in the database server. After the scheduling process, a classroom list is provided as services to the educational programmers to user clients. In order to provide a flexible and scalable service, data are packaged in the form of web services deployed on the server.

From each user's point of view, the client is a customized web service that can be accessible via 3G, GPRS and WIFI internet. After the users send their requests to the web services, the main function running on the server is launched to calculate the scheduled classroom and return the results to the clients. The data communication is mainly based on internet-based web services, in which data are packaged in specific data formats (such as XML and JSON).

Moreover, SmartClass architecture provides a set of back-stage control web pages which allow administrator to work with database management. The classroom schedule process can be performed either by the scheduling algorithm, or operated by the administrator manually.

3.2 Processing Flow of SmartClass

SmartClass applies SOA concepts to solve the classroom scheduling problem, which could facilitate educators in classroom arrangement in each semester. Due to that the scheduling functions are deployed as services, educational programmers and teaching secretaries can be significantly eased from scheduling the classroom resource manually.

The general processing flow of the SmartClass framework is illustrated in Fig. 2. The system includes following components:

1) First, both the lecture and classroom information is modeled as the input of the scheduling scheme. The class information can be abstracted into a five-dimensional tuple T(I,C,S,Te,M), and the definition of the parameters are listed in Table 1. In particular, the interval parameter indicates the specific time slot of the arranged lessons. For example, if 10 time slots are divided for the entire school day, the interval

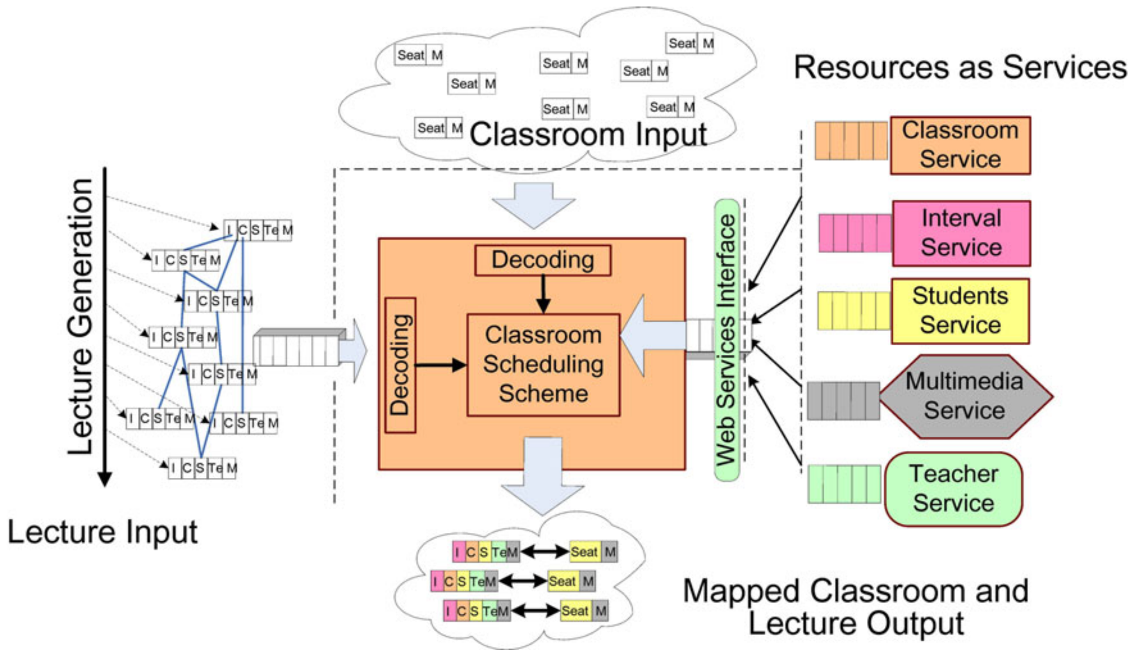


Fig. 2. General processing flow of SmartClass.

may set to [1], [2], [9], [10], referring to that the lecture will be held as the first two lessons and the last two lessons.

Similarly, the classroom information contains a two-dimensional tuple C (Seat, M), in which the Seat represents the available seat number, while the Boolean variable M represents that whether the lecture requires multimedia support.

2) The input data also contains services from different resources, such as classroom, interval, multimedia and teacher constraints.
3) SmartClass architecture provides a decoder that is oriented from both the lecture and classroom input, thereby it allows scheduling scheme to analyze the detailed information. Once the data is decoded, then the scheduling scheme can automatically map the lectures to different classrooms simultaneously

Similarly, the classroom information also contains a two-dimensional tuple C(Seat,M), in which the Seat represents the available seat number, and M indicates the multimedia support of the classroom.

3.3 SOA Architecture and Hierarchical Model

The service abstract model construction includes following issues: which resources are modeled, how they are managed, and what services are provided to the users. Fig. 3 illustrates the hierarchical models of the SOA architecture, which consist of three layers: services layer, servant layer and physical layer. The functionality of each layer is introduced as below:

TABLE 1
Input Parameters of the Scheduling

ID	Type	Description
I	Interval	[StartTime, EndTime]
C	Integer	Classroom
S	Integer	Student Number
Te	String	Teacher Name
M	Boolean	Multimedia

3.3.1 Services Layer

Services layer is composed of three modules: services provider, scheduler and transmitter.

First, services provider prepares service access points (SAPs) to clients. Each SAP takes charge of one specific kind of service. All the SAPs are provided based on data format packaging mechanism. When a request arrives, the services provider first decodes the target request to identify which service is requested and then the specified request will be sent to the service scheduler.

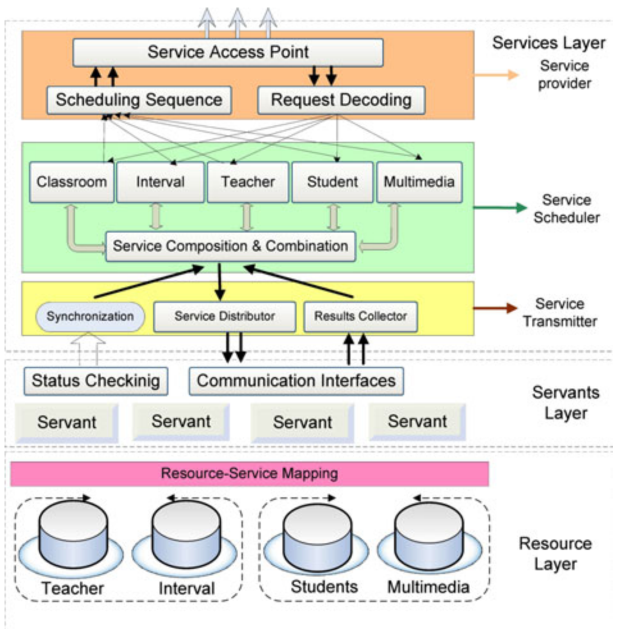


Fig. 3. Hierarchical layer of servers.

TABLE 2
Communication Interfaces of Five Services

Name	Request	Responses	Services Description
Get Classroom	ClassID	Classroom No.	*@Path("/classroom/") @GET* *@Produces("/application/xml,application/json")* *PublicList<Classroom>getClassroom(@QueryParam("ClassID") Long ClassID)*
Get Number	ClassID	Student Number	*@Path("/number/") @GET* *@Produces("/application/xml,application/json")* *Public Int getStudentNo (@QueryParam("ClassID") Long ClassID)*
Get Name	ClassID	Teacher Name	*@Path("/number/") @GET* *@Produces("/application/xml,application/json")* *Public String get TeacherName (@QueryParam("ClassID") Long ClassID)*
Get Interval	ClassID	Timing Interval	*@Path("/Interval/") @GET* *@Produces("/application/xml,application/json")* *Public List get Interval (@QueryParam("ClassID") Long ClassID)*
Get Multimedia	ClassID	Multimedia Configurations	*@Path("/multimedia/") @GET* *@Produces("/application/xml,application/json")* *Public Boolean get Multimedia (@QueryParam("ClassID") Long ClassID)*

Second, services scheduler is in charge of service scheduling and mapping procedure. Each request for the client may be decomposed into several service requests. Therefore if more than one servant is available, then each service request must be mapped and scheduled to a certain servant dynamically, according to the system load-balancing status. Hence, all the provided services are directly connected to a composition and combination module.

Third, the services transmitter dispatches the sub-tasks to different servants for execution. After a task is finished, the results are collected by the services transmitter. Furthermore, a synchronization interface is provided to the scheduler when multiple services are returned at the same time. In this situation, the scheduler should decide the priority of the returned services dynamically.

Corresponding to the features of scheduling module, SmartClass server provides five services, as is listed in Table 2:

1) Classroom service: This service is in charge of providing available classroom information. The servant receives *ClassID* as the input and returns available classroom owned by *ClassID* in XML or JSON formats.
2) Interval service: This service determines the output of the timing interval for each proposed lecture. The servant receives a *ClassID* as input, and return with a list composing of all the available intervals.
3) Teacher Name service: This service provides the teacher's name of the selected service. The servant receives a *ClassID*, and returns the teacher name in string patterns.
4) Student Number service: The student number service receives a *ClassID* as input parameter, while the exact number of students enrolled in this lesson is returned to the scheduling scheme for further process.
5) Multimedia service: Multimedia service indicates whether current lecture requires multimedia support or not. Input is noted as *ClassID*, while the output returns the Boolean values.

3.3.2 Servants Layer

As described above, services are mapped to different servants. In order to provide a complete match for each service request, at lease one servant should be integrated for each service. All the servants are efficiently managed to make sure that each issued service request is mapped to a specific servant. The data transmission between the servants layer and services layer is through communication interfaces and status checking interface.

In particular, the status checking interface is responsible for providing synchronization information of servants for services mapping and scheduling, such as loading-balancing and services bottleneck exploration. The services can be migrated for load balancing at runtime.

3.3.3 Physical Layer

The physical layer consists of database and object-relation (O-R) mapping mechanisms. In General, all the classroom and lecture information are stored in databases, which may locate in distribution physical machines. Dealing with the state-of-the-art relation based database models, the OR mapping methods are employed for object-oriented abstractions, such as Hibernate and Toplink.

3.4 Integrated Services

Table 2 illustrates several examples of communication interfaces when users send their requests for a specific service. All the requests are sent to the server with the directly access with the parameter of ClassID. After the servant finishes data base query with ClassID key, the corresponding results will be packaged into certain data format (JSON in Table 2) and sent back to the scheduling process.

In order to provide the above services, currently we are using REST web services framework for demonstrating the five services. Table 2 refers to the proposed five services description in RESTful web service manners. Each service is deployed at an individual path, by which the scheduled results are transferred with JSON format, after the input parameters are well received.

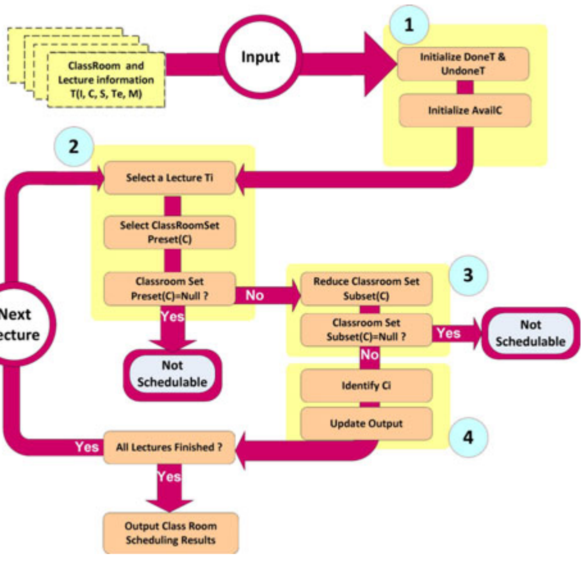

ClassRoom and Lecture information T(I, C, S, Te, M)
Input
1
Initialize DoneT & UndoneT
Initialize AvailC
2
Select a Lecture Ti
Select ClassRoomSet Preset(C)
Classroom Set Preset(C)=Null ?
No
Yes
Not Schedulable
3
Reduce Classroom Set Subset(C)
Classroom Set Subset(C)=Null ?
Yes
Not Schedulable
No
Identify Ci
Update Output
4
Next Lecture
Yes
All Lectures Finished ?
Yes
Output Class Room Scheduling Results

Fig. 4. Scheduling process.

3.5 Greedy Based Classroom Scheduling Algorithm

With respect to the scheduling process at server, we utilize a greedy algorithm based scheduling approach. The procedure of the scheduling algorithm is illustrated in Fig. 4. More specifically, the main scheduling procedure is composed of four steps:

3.5.1 *Initialization*

In the initialization phase all the classes are divided into two sets: all the classes those have been mapped to certain classrooms belong to DoneT, while the unmapped classes are included in UndoneT. Furthermore, the occupied classrooms are considered as AvailC.

In the initialization of the algorithm, since no classes have been mapped to any classrooms, both the DoneT and AvailC are empty, and UndoneT is equal to T.

3.5.2 *Exclude the Time Interval Conflict*

The classroom scheduling method contains multiple iterations. During each iteration, greedy algorithm is applied that a lesson with the longest interval is selected with highest priority. In the mapping procedure, if a classroom set Preset (C) is selected, then all the classrooms that have time interval conflicts with current class are ruled out.

3.5.3 *Detect the Seat and Multimedia Conflict*

After the time interval conflict is eliminated, the available classroom sets are selected as Preset (C). Then the next step is to take the seat and multimedia factors into consideration. All the seat and multimedia conflicts are removed from the Preset (C), and A Subset (C) is selected from Preset (C).

3.5.4 *Identify the Final Classroom*

In the final stage, the selected classroom is identified from the Subset (C). Two pre-references are taken into account: 1) First, if a classroom that is already used by another lesson is still available for the current class, then it should be used with high priority. Otherwise, a new classroom could be selected. 2) Second, if multiple classrooms are suitable for current class, then a classroom that has the least seats should be chosen.

We describe scheduling procedure in pseudo code, as illustrated in Algorithm 1:

Algorithm 1. Classroom Scheduling Algorithm

```
1   Input: Class Spaces T(I,C,S,Te,M)
2   Output: Selected classroom Ci.
3   DoneT = $\emptyset$, UndoneT = T, AvailC = $\emptyset$
4     //Initialization:
    Repeat
6
7   Select $t \in T$ and t->I is the max
8   $\forall$ Tj $\in$DoneT && Tj.I $\cap$ Ti.I $\neq \emptyset$, Tj.C $\notin$ Preset(C)
9     //Exclude the time conflict
10
11  Subset(C) $\subseteq$ Preset(C)
12  $\forall$ Ci $\in$Subset(C), Ci. Seat > Ti. S && Ci. M = Ti. M
13    //Exclude seat and multimedia conflict
14
15  IF AvailC$\cap$Subset(C) = $\emptyset$,
16    THEN SELECT Ci $\in$ Subset (C) WHERE Ci.
    Seat = MIN(Cj. Seat) ($\forall$ Cj$\in$Subset (C))
17  ELSE
18    SELECT Ci $\in$ AvailC$\cap$Subset(C) WHERE Ci.
    Seat = MIN(Cj. Seat) ($\forall$ Cj$\in$AvailC$\cap$Subset(C))
19    //Identify the classroom
20  end
```

The input of the algorithm is the class spaces set T and the output the classroom for each class, as described in the Line 1~2. The DoneT set includes the lessons that have been assigned with a classroom. In contrast, UndoneT indicates the classes that have not been arranged. AvailC refers to the classrooms that have been occupied by DoneT. At start-up, both DoneT and AvailC are initialized to empty set, while UndoneT is set to T.

Subsequently, the class that has the longest duration time in the T will be scheduled at first. For each selected class, first the time conflict with the classes that have been already scheduled should be excluded, as presented from Line 7~9. Then other significant constraints like the seat and multimedia conflict should be removed, as is described in Line 11~13. Finally Line 15~19 identifies the final classroom.

Let N be the scale of the classroom spaces and M is the scale of the lecture space, from the description we can get the complexity of the algorithm is O (M $\times$ N).

4 A Case Study

To demonstrate the effectiveness of the SmartClass architecture, we evaluate the SmartClass with a real classroom scheduling application.

4.1 Demonstration Input Arrays

A sample test case is illustrated with eight lectures and four classrooms. The five-dimensional lecture tuple T(I,C,S,Te,M) and two-dimensional classroom tuple C(Seat,M) are described in Table 3.

The basic input includes both lecture and classroom datasets. On one hand, for lecture information, a five-dimensional tuple is given as the features include

TABLE 3
A Demonstration Testcase

Input T(I,C,S,Te,M)	C(Seat,M)
T1([1-2,9-10],C, 100,“Cheung”,Y)	C1(150,Y)
T2([4-5,9-10],C, 50,“Lee”,N)	C2(50,N)
T3([3-4,11-15],C, 75,“Wong”,Y)	C3(100,N)
T4([6-8,12-15],C, 50,“Chung”,N)	C4(120,Y)
T5([1-3,18-20],C, 50,“Yang”,N)	
T6([4-5,12-13],C, 80,“Lee”,Y)	
T7([1-3,8-9],C, 100,“Zhao”,N)	
T8([16-17],C, 50,“Lew”,N)	

[StartTime, EndTime], Classroom, Student Number, Teacher Name and Multimedia. For instances, T1 ([1], [2], [9], [10], C, 100,”Cheung”,Y) indicates that the lecture T1 is taken at 1-2 and the 9-10 time slots. The lecture is given by Professor Cheung, and requires a classroom can accommodate 100 students with a multimedia support. By the time of the input process, it is clear that the classroom has not been selected yet, therefore the Classroom is denoted as “C” instead of a specific value. On the other hand, the classroom information includes the seat number and the Multimedia features.

4.2 Scheduling Process

The scheduling procedure should decide the priority for all the lectures. Regarding the input lectures from T1 to T8, the scheduling order of the lectures is T3, T4, T5, T7, T1, T2, T6 and T8, sorted by descending order of the timing intervals. Fig. 5 illustrates the detailed scheduling procedure and the experimental results. In particular, for each scheduling step, the Preset, Subset and the selected classroom is described as below:

- For T3, Preset = {C1,C2,C3,C4}, Subset = {C1,C3}, and final selected classroom is C3.
- For T4, Preset = {C1,C2,C4}, Subset = {C2,C4}, and final selected classroom is C2.
- For T5, Preset = {C1,C2, C4}, Subset = {C2,C4}, and final selected classroom is C2.
- For T7, Preset = {C1, C4}, Subset = {C3}, and final selected classroom is C3.
- For T1, Preset = {C1, C4}, Subset = {C1,C4}, and final selected classroom is C4.
- For T2, Preset = {C1,C2}, Subset = {C2}, and final selected classroom is C2.
- For T6, Preset = {C1, C4}, Subset = {C1,C4}, and final selected classroom is C4.
- For T8, Preset = {C1,C2,C3,C4}, Subset = {C2,C3}, and final selected classroom is C2.
- Finally, the final classroom set is a combination of all the available classroom IDs, which results in the AvailC is {C2,C3,C4}.

4.3 Timing Complexity Analysis

The above case study demonstrates the effectiveness of the SmartClass concepts and the timing complexity for the algorithm. Since the time complexity of our proposed greedy algorithm is only O(M×N), SmartClass approach is less timing consuming than most state-of-the-art evolutional algorithm literatures. In contrast, the ILP and greedy algorithm always provide an alternative low timing complexity solution. From [5], which converts the classroom scheduling problem into a 0-1 ILP problem, and its timing complexity is O(M × N + k), where the parameter k indicates the number of course restrictions. Generally k is a constant value which means the introduction of the greedy algorithm employed in this paper is similar to the ILP approaches. Furthermore our SmartClass framework is loose coupled to the proposed scheduling algorithms, thus it leverages the modularity of different resources and provides a promising metrics to the combination with ILP approaches as well as other scheduling algorithms.

5. Evaluation of Efficiency and Overheads

In order evaluate the efficiency and overhead metrics of the SmartClass architecture, we present a detailed quantitative results and data analysis.

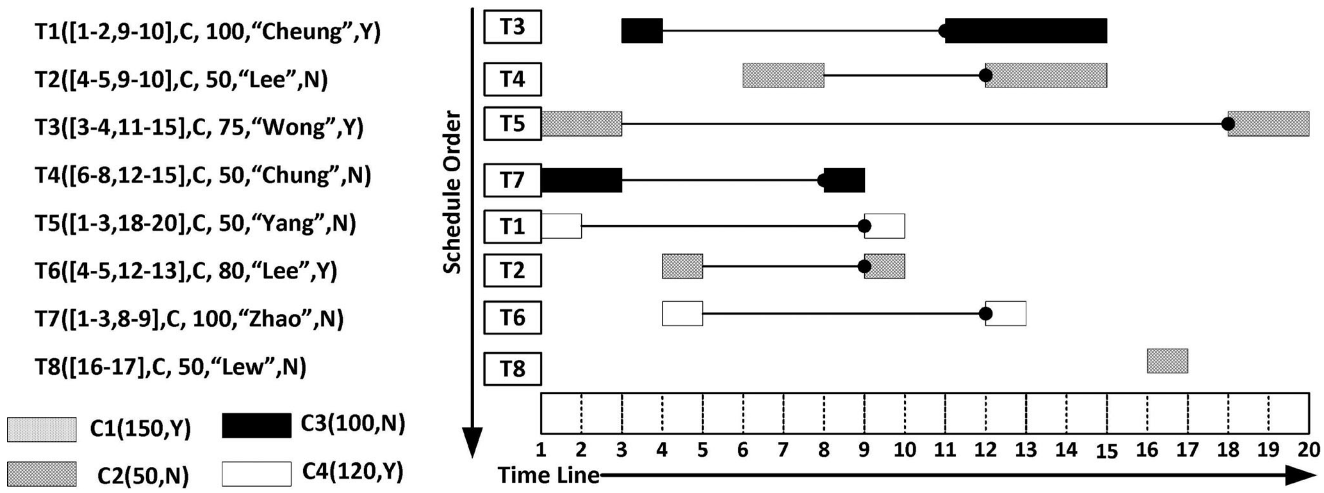


Fig. 5. The scheduling results for the test case. Different classrooms are filled with distinct colors. The right part illustrates the scheduling process of the eight subjects, in which *x*-axis represents the time line, while the *y*-axis is the scheduling order. For each subject, a specific classroom is assigned, and class C1 is unnecessary finally.

TABLE 4
Average Percent of Classroom Scheduling Problems That Find a Match in the Case Base

Number of cases in the case base	15-course simple case base	15-course complex case base	20-course simple case base
5	76.67%	78.3%	95%
10	90%	90%	95%
15	90%	98.33%	95%
20	90%	98.33%	95%

5.1 Efficiency

The efficiency is a major concern of the scheduling algorithm. We use a similar case as [19] to measure the efficiency of the greedy based scheduling.

A large number of systematically designed experiments have been performed on different number of cases (5, 10, 15, 20) with differing complexity (15-course simple, 15-course complex, and 20-course simple cases). The complex cases and simple cases are defined in the [19]. Table 4 is presented by the authors in [19], which shows average percentages of experimental results that have found a match in these cases.

5.2 Overheads Evaluation

The advantage of the SmartClass is to alleviate the burden of the terminal users. However, a side effect is the communication overheads between the central server and the terminal users. The overheads contain two fold:

1) With respect to the execution on the server, it would be favorable to evaluate the time required by the algorithms to compute an output. We have evaluated the execution time of classroom scheduling problems that find a match. Experimental results in Table 5 and Fig. 6 demonstrate the execution time.

   Experiments have shown that we the calculation can get a linear execution time, growing from 10.51 seconds to 17.99 seconds, when the number of courses increases from 5 to 20. Results on the execution time demonstrate that the SmartClass approach is able to achieve a satisfying scalability.
2) Regarding the response time of the terminal users, it will be also useful to measure the response time, which stands for the data communication time between the terminal users and the central server. We measured the data communication overheads. Table Table 6 and Fig. 7 demonstrate that the data transfer takes only less than 1.5 seconds while the execution takes more than 20 seconds. The data transfer time is quite insignificant comparing to the execution.

TABLE 5
Execution Time of Classroom Scheduling Problems (Seconds)

Number of lessons	15-course simple	15-course complex	20-course simple
5	10.51	11.00	15.01
10	12.92	13.47	19.96
15	14.55	16.01	21.40
20	17.99	19.07	28.99

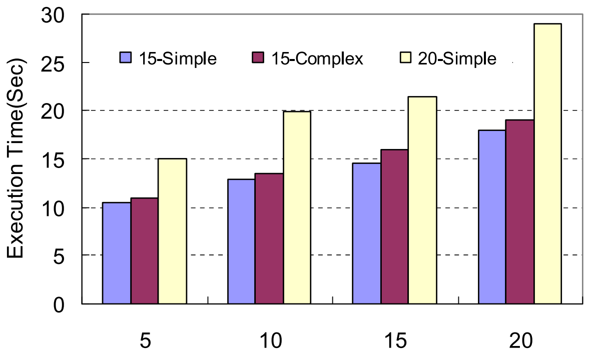


Fig. 6. Execution time.

Another perspective is the scalability analysis of the data transfer time. Experiments have shown that the data transfer time increases from 0.09 seconds to 0.71 seconds, when the number of courses increases from 5 to 20.

5.3 Comparison with State-of-the-arts

To comparing our scheduling algorithm with the state-of-the-art scheduling methodologies, we have measured the performance rankings of general greedy based scheduling, the choice function [15] scheduling and simple random [16] scheduling. Table 7 illustrates the calculated success ratios (the ratio of successful runs in which the optimal solution is found to the total number of runs), the average best fitness values and the average best duration values of the tests. The results show that greedy based algorithm, in the overall, performs better than SR and CF methods. Anyway, the focus of this paper is the SOA concepts, therefore the greedy selection algorithm can be replaced by other similar approaches.

5.4 Discussion of the Limitations

In this Section we discuss the limitations of the proposed approach from the experimental results.

1) Due to that the classroom scheduling is a NP-Complete problem; therefore in some cases the scheduler is not able to find an optimal result. As is revealed from Section 5.1, the successful rate for the greedy scheduling can achieve up to 98 percent, which is acceptable in most cases.

TABLE 6
Data Transfer Time of Classroom Scheduling Problems (Seconds)

Number of cases in the case base	15-course simple case base	15-course complex case base	20-course simple case base
5	0.09	0.11	0.19
10	0.18	0.21	0.36
15	0.36	0.38	0.72
20	0.71	0.74	1.43

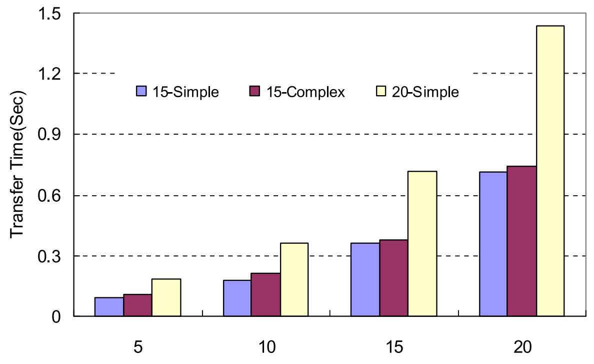


Fig. 7. Data Transfer time.

2) Regarding the performance of the proposed approach with state-of-the-arts, experimental results in Table 7. The execution time of the greedy based scheduling is much less than the SR and CF functions. Also, from the timing complexity analysis in Section 4.3, the greedy based approach has a very close complexity with the ILP based approaches. The experimental results are derived from state-of-the-art literatures such as [15] and [16], which demonstrates the SOA concept can be used in many different situations.
3) Scalability analysis: A significant advantage against state-of-the-arts is the scalability. As we use SOA architecture in the framework design, therefore the SmartClass can easily incorporate different scheduling algorithms. Meanwhile, the communication overhead is quite scalable with the increase of the number of courses.

TABLE 7
Performance Ranking of Different Methodologies

Dept	Classes	Lectures	Greedy	Choice function	simple random
Dept. 1	200	64	2.5	2.5	2.5
Dept. 2	200	64	2	2	2
Dept. 3	400	128	2	2	2
Dept. 4	400	128	1	2	3
Dept. 5	800	256	2	2	2
Dept. 6	800	256	1	2	3
Dept. 7	1,600	512	1	2	3
Dept. 8	1,600	512	1	2	3
Dept. 9	200	64	2.5	2.5	2.5
	Average		1.67	2.11	2.56

6 Conclusion and Future Work

This paper has proposed SmartClass, which introduces services-oriented concept into conventional classroom scheduling approach. Each type of resources is abstracted as a service and the services are provided to the terminal users. Greedy based scheduling is incorporated into the SmartClass framework to calculate the results. Experimental results on case studies illustrate that SmartClass can efficiently improve the flexibility and scalability. Due to that the SmartClass architecture operates the scheduling process at server side, therefore it could significantly alleviate the burden of the educational programmers.

The initial results are promising but the study is still undergoing since there are a lot of directions worth pursuing. Future work includes following subjects: First, we are to extend the server to distributing cloud systems to achieve high throughput and integration of services. Second, we also plan to integrate the timing interval design-space-exploration into the scheduling scheme to get optimized solution. Last but not least, the greedy strategy utilized in this paper could be replaced by other alternative scheduling schemes, such as the ILP and graph theoretical methodologies.

Acknowledgments

This work was supported by the National Science Foundation of China under grants (Nos. 61379040, 61272131, 61202053, 61222204, 61221062), Jiangsu Provincial Natural Science Foundation (No. SBK201240198), Fundamental Research Funds for the Central Universities No. WK0110000034, Open Project of State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences (No. CARCH201407), and the Strategic Priority Research Program of CAS (No. XDA06010403). The authors deeply appreciate many reviewers for their insightful comments and suggestions. Xi Li is the corresponding author.

References

[1] J. D. Ullman, "NP-complete scheduling problems," *J. Comput. Syst. Sci.*, vol. 10, no. 3, pp. 384–393, 1975.
[2] T. Erl, "*Service-Oriented Architecture: Concepts, Technology, and Design*. Englewood Cliffs, NJ, USA: Prentice-Hall PTR 2005.
[3] G. S. Prabhu and P. M. Shankar, "Simulation of flat fading using MATLAB for classroom instruction," *IEEE Trans. Edu.*, vol. 45, no. 1, pp. 19–25, Feb. 2002.
[4] G. S. Mason, T. R. Shuman, and K. E. Cook, "Comparing the effectiveness of an inverted classroom to a traditional classroom in an upper-division engineering course," *IEEE Trans. Edu.*, vol. 56, no. 4, pp. 430–435, Nov. 2013.
[5] E. Mooney, R. Dargen, and W. Parameter, "Large-scale classroom scheduling," *IIE Trans.*, vol. 28, no. 5, pp. 369–378, 1996.
[6] V. Tam and D. Ting, "Combining the min-conflicts and look-forward heuristics to effectively solve a set of hard university timetabling problems," in *Proc. 15th IEEE Int. Conf. Tools Artif. Intell.*, 2003, pp. 492–496.
[7] A. Wasfy and F. A. Aloul, "Solving the university class scheduling problem using advanced ILP techniques," in *Proc. IEEE GCC Conf.*, 2007, pp. 1–5.
[8] F. A. Aloul, A. Ramani, I. L. Markov, and K. A. Sakallah, "Generic ILP versus specialized 0–1 ILP: An update," in *Proc. Int. Conf. Comput. Aided Des.*, 2002, pp. 450–457.
[9] G.-J. Nam, F. Aloul, K. Sakallah, and R. Rutenbar, "A Comparative study of two Boolean formulations of FPGA detailed routing constraints," in *Proc. Int. Symp. Phys. Des.*, Sonoma, California, USA, 2001, pp. 222–227.
[10] F. A. Aloul, S. Hassoun, K. A. Sakallah, and D. Blaauw, "Robust SAT-based search algorithm for leakage power reduction," in *Proc. Workshop Power Timing Modeling, Optim. Simul.*, 2002, pp. 167–177.
[11] A. Ramani, I. L. Markov, K. A. Sakallah, and F. A. Aloul, "Breaking instance-independent symmetries in exact graph coloring," *J. Artif. Intell. Res.*, vol. 26, no. 1, pp. 289–322, 2006.
[12] A. Dandashi and M. Al-Mouhamed, "Graph coloring for class scheduling," in *Proc. ACS/IEEE Int. Conf. Comput. Syst. Appl.*, 2010, pp. 1–4.
[13] T. A. Redl, "University timetabling via graph coloring: An alternative approach," *Dept. Comput. Math. Sci.*, University of Houston, vol. 187, pp. 174–186, 2007.

[14] D.-F. Shiau, "A hybrid particle swarm optimization for a university course scheduling problem with flexible preferences," *Expert Syst. Appl.*, vol. 38, no. 1, pp. 235–248, 2011.

[15] M. Maashia, E. Ozcana, and G. Kendall, "A multi-objective hyper-heuristic based on choice function," *Expert Syst. Appl.*, vol. 41, no. 9, pp. 4475–4493, 2014.

[16] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," *J. Oper. Res. Soc.*, vol. 64, pp. 1695–1724, 2013.

[17] P. Cowling, G. Kendall, and E. Soubeiga, "A hyper-heuristic approach to scheduling a sales summit," in *Proc. 3rd Int. Conf. Practice Theory Autom. Timetabling*, 2002, pp. 176–190.

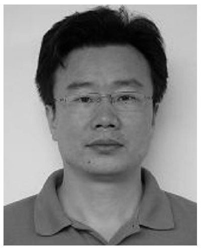
[18] A. Wang, C. Wang, Xi Li, and X. Zhou, "SmartClass: A services-oriented approach for university resource scheduling," in *Proc. 10th Int. Conf. Serv. Comput.*, 2013, pp. 745–746.

[19] E. K. Burke and S. Petrovic, "Recent research directions in automated timetabling," *Eur. J. Oper. Res.*, vol. 140, no. 2, pp. 266–280, 2002.

Chao Wang received the BS and PhD degree from the University of Science and Technology of China, in 2006 and 2011, respectively, both in computer science. He is an assistant professor with the Embedded System Lab in the Suzhou Institute of University of Science and Technology of China, Suzhou, China. His research interests focus on service-oriented multicore systems and reconfigurable computing. He has authored or co-authored more than 20 publications and patents. He serves as an editor board member of MICPRO, IET CDT, and IJBPIM, publicity co-chair of HiPEAC 2015 and ISPA 2014. He is also a TPC member of SCC, ICPADS, EUC, ISPA, FPL, ReConfig, and ARC conferences, a guest editor for *Transactions on Computational Biology and Bioinformatic and Indian Journal of Physiology and Pharmacology*, and a reviewer for DAC, FCCM, FPT, TRETS, TECS, JSA, Micropro, and TJS. He is a member of the IEEE, ACM, and senior member of CCF.

Xi Li is a professor and vice dean in the School of Software Engineering, University of Science and Technology of China. There he directs the research programs in Embedded System Lab, examining various aspects of embedded system with the focus on performance, availability, flexibility, and energy efficiency. He has lead several national key projects of CHINA, several national 863 projects and NSFC projects. He is a member of ACM and the IEEE and a senior member of CCF.

Aili Wang is a lecturer of the School of Software Engineering, University of Science and Technology of China. She serves as the guest editor of *Applied Soft Computing*, and *International Journal of Parallel Programming*. Meanwhile she is a reviewer for *International Journal of Electronics*. She has published more than 10 International journal and conference articles in the areas of software engineering, operating systems, and distributed computing systems.



Xuehai Zhou is a professor in the School of Computer Science, and the School of Software Engineering, University of Science and Technology of China. He serves as a general secretary of steering committee of computer College fundamental Lessons, and technical committee of Open Systems, CCF. His research interests include various aspects of multicore and distributing systems. He has lead several national 863 projects and NSFC projects. He has published more than 100 International journal and conference articles in the areas of software engineering, operating systems, and distributed computing systems. He is a member of ACM and the IEEE and a senior member of CCF.