
Aspen Scheduler: a Web-based Automated Master Schedule Builder for Secondary Schools

Baiyun Tao, Rick Dwyer

Follett Software Company/X2 Development Corporation, Hingham, MA, USA

btao@FollettSoftware.com

rdwyer@FollettSoftware.com

1 Introduction

Building a master schedule involves assigning courses, teachers and rooms to time slots, maximizing usage of resources and fulfilling student course requests. It is one of most complex tasks performed by administrators at a school each year. This is particularly true for secondary schools in the United States where the number and variety of constraints (such as multi-day rotation, combined courses, room capacities, teacher teaming, and student grouping, etc.) increases the complexity of the scheduling process.

Designed to accommodate both large and small schools with the most complex schedule needs, the system described here, Aspen master schedule builder, implemented by X2 Development Corporation, successfully deals with producing a satisfactory and efficient solution from the large solution space in a reasonable amount of time, while at the same time handling the usability, flexibility and completeness of the scheduling process (see Fig 1 for real life scheduling examples).

2 System Overviews

Aspen master schedule builder is part of a Student Information System that is completely web-based (see Appendix A for terms we use in this paper). It is a Java 2 Enterprise Edition (J2EE) application that runs on Apache Tomcat web server, uses Apache Struts framework, OJB and supports three database backend: MySQL, SQL Server and Oracle.

Example School	Scheduling Profile
School - 1	<p>Schedule terms: Full year academic courses, semester elective courses Number of days per cycle: 5 Number of periods per day: 7</p> <p>Total number of courses: 172 Total number of staff: 53 Total number of rooms: 50 Total number of Students: 700 Number of requests per student: 7-10 Total number of student requests: 5,600 Total number of user specified rules: 29 Total number of sections in master schedule: 370</p> <p>Time for building master schedule: 2 minutes Time for loading students: < 1 minute</p>
School - 2	<p>Schedule terms: All courses are semester length Number of days per cycle: 2 Number of periods per day: 8</p> <p>Total number of courses: 314 Total number of staff: 102 Total number of rooms: 90 Total number of Students: 1,200 Number of requests per student: 8 Total number of student requests: 9,600 Total number of user specified rules: 109 Total number of sections in master schedule: 800</p> <p>Time for building master schedule: 10 minutes Time for loading students: 5 minutes</p>
School - 3	<p>Schedule terms: All courses are semester length Number of days per cycle: 10 Number of periods per day: 15</p> <p>Total number of courses: 752 Total number of staff: 216 Total number of rooms: 253 Total number of Students: 4,200 Number of requests per student: 16 Total number of student requests: 67,200 Total number of user specified rules: 525 Total number of sections in master schedule: 3,500</p> <p>Time for building master schedule: 90 minutes Time for loading students: 45 minutes</p> <p>*A special constraint: 4 lunch waves splitting over 3 periods</p>

Fig.1. Real life scheduling examples

There are four main objectives used by Aspen master schedule builder to build a schedule:

1. Assign each section with an available teacher, an appropriate time slot and a suitable room.
2. Satisfy all hard constraints.
3. Maximize the satisfaction of student requests.
4. Balance student distribution cross different sections of the same course.

The system breaks the overall problem of building a master schedule into sub-problems, find solutions for each of the sub-problem and combine them to reach an overall solution. An example of a sub-problem is to schedule a section group, a group of sections that are related to each other because of blocking rules (see Fig.4.2) or tightly shared resources such as teacher and room.

The system first orders all sections by their difficulty to schedule. A section's difficulty to schedule is determined by analyzing the related course's attributes, teacher availability, room availability and other constraints of the section. Sections are scheduled in rank order with the most difficult ones being scheduled first. Section group is formed when a section is chosen to be scheduled based on previous analysis. A section group contains one or more sections.

To schedule each section group, an iterative metaheuristic is used that combines hill-climbing, simple random and greedy algorithms. Three key functions are defined: search, validation and evaluation. The search function first finds potential solutions to validate and evaluate. It starts with an initial solution and iteratively works to find better ones. A good solution must be a valid solution – one that meets all the hard constraints. A fitness function is used to check a solution against all hard constraints to objectively determine its validity. Once a solution is determined to be valid, its quality can be measured and scored. The quality of a solution is determined by three primary measures: the number of student requests satisfied, the enrollment balance cross sections of the same course, and the flexibility of the solution, i.e. the ability for students to change sections. An objective function evaluates the solution and assigns a value to the solution.

Solutions then can be compared and the solution with smallest value is considered as the best solution and will therefore be chosen for the section group.

While choosing a solution for the current section group, if the solution affects the ranking of sections still to be scheduled, unscheduled sections are either re-ranked or required resources will be reserved for the future.

When a solution cannot be found for the current section group without violating a hard constraint, the system attempts to repair the schedule by moving resources already scheduled to make the resources needed by the current section group available. If such repair attempt is not possible, the system stops with appropriate feedbacks to the user.

3 Key Features

3.1 Time Structure

Schools have their own unique ways to structure time based on the needs to schedule. Successfully building a master schedule depends on an efficient and flexible way to represent these unique time structures. The structure of a schedule is defined by three core parameters: the number of terms per year (TPY), the number of days per cycle (DPC) and the number of periods per day (PPD).

In addition, scheduling patterns are introduced to represent the valid ways to schedule a course of a particular shape. It helps users visualize their schedule (See Fig. 2). Patterns with similar shapes are grouped together as pattern sets, then assigned to courses to define the list of possible time slots a course can be scheduled into.

3.2 Input and Constraints

There are five fundamental input components to build and load a schedule: Course, Student, Staff, Room and Request with each imposing a set of constraints on the schedule. See Fig.3. for the most common constraints associated with each type of input.

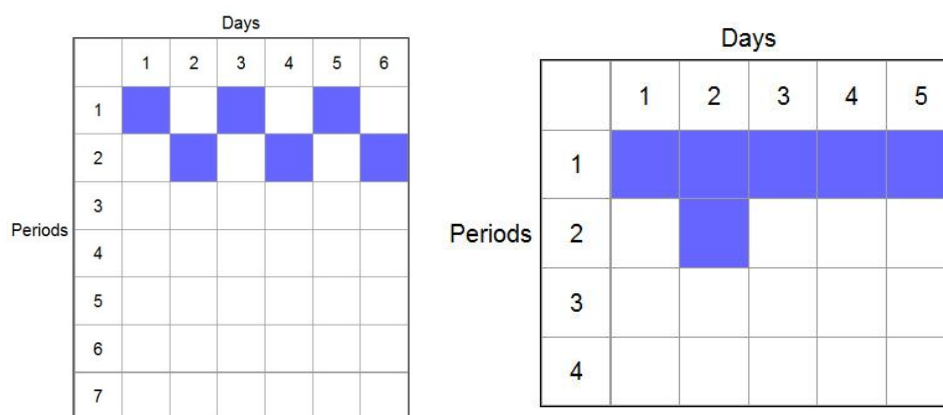


Fig.2. Different types of schedule patterns

Input	Constraint
Course	<ul style="list-style-type: none"> • Number of sections • Valid schedule patterns • Balance type cross section and subject • Load priority used for conflict resolution for individual student
Student	<ul style="list-style-type: none"> • Student grouping code: house, team and / or platoon • Unavailable time • Load priority • Enrollment weight
Staff	<ul style="list-style-type: none"> • Teacher assignments • Preferred room and location • Maximum number of rooms and locations allowed • Maximum number of consecutive teaching periods allowed
Room	<ul style="list-style-type: none"> • Max capacity • Department and location usage
Request	<ul style="list-style-type: none"> • Preferred section and / or teacher • Partial content • Inclusion • Section type

Fig.3. Common constraints for each type of input

3.3 Rules

Rules are introduced to allow additional and more complex constraints on the schedule. Rules can be defined as either hard or soft. Hard rules must be satisfied in order for the schedule to be valid. Soft rules are satisfied whenever possible - it

is at the discretion of the system. The system will decide which soft constraint to drop in the event of a conflict between soft constraints. The system can also drop a soft constraint if the impact on the overall schedule is too costly such as a valid solution cannot be achieved.

The system currently provides 27 different types of rules for the scheduling process including both build rules and load rules (See Fig.4.1 and Fig. 4.2).

Although the system separates the build and load process. All load rules are included and evaluated during the build process to ensure potential solution don't violate the load rules.

3.4 Sandbox Approach

The system takes a sandbox approach to allow users to create and save multiple scheduling scenarios within each school. Scenarios can be copied at any point to serve as a back-up. Each scenario can include different set of students, courses, teachers, and/or rules. Scenarios have their own master schedule that is built and loaded independently of all other scenarios. Users can use different scenarios to experiment with different constraints and even different types of schedules (See Fig.5.).


Scenarios					
		0 of 7 selected 		Custom Selection	
Details	<input type="checkbox"/>	Name	Term	DPC	PPD
Preferences	<input type="checkbox"/>	1. Basic Scenario	1/2,1/1	1	7
Terms	<input type="checkbox"/>	2. Demo Scenario Build	1/2,1/1,1/4	5	7
Days	<input checked="" type="checkbox"/>	3. Demo Scenario Master Matrix	1/2,1/1,1/4	5	7
Periods	<input type="checkbox"/>	4. Room Management Scenario	1/1,1/2	1	7
Rotations	<input type="checkbox"/>	5. Staff Management Scenario	1/1,1/2	1	7
Bell Schedules	<input type="checkbox"/>	6. Student Grouping Scenario	1/1,1/2	1	7
	<input type="checkbox"/>	7. Validate/Build/Load Scenario	1/1,1/2	1	7

Fig.5. Different scenarios

Rule Name	Rule Description
Bell - Course Restrictions	Restrict courses for a particular bell schedule
Bell - Room Restrictions	Restrict rooms for a particular bell schedule
Course - Alternates	Schedule alternate courses for students upon conflicts
Course - Blocking	Schedule a set of courses together following particular relationships
Course - Group Exceptions	Specify the courses that can be excluded from any grouping
Course - Pattern Sets Restriction	Restrict usage of patterns for a course
Course - Pull-out	Allow student to be pulled out from one course to go to another
Course - Sequence Concurrent	Schedule two courses in the same term for a student
Course - Sequencing	Schedule a courses in a student's schedule for completion before the start of another course
Course - Term Link	Specify the list of courses that a student must be scheduled in specified terms
Course - Terms Restriction	Restrict usage of terms for a course
Room - Course Restriction	Reserve rooms for a course
Room - Exceptions	Exclude rooms from being used by a course
Room - Proximity	Define room proximity
Room - Reservations	Reserve rooms for courses
Room - Unavailable	Room is unavailable for scheduling during these time slots
Room- Teacher Restriction	Reserve rooms for a teacher
Student - Avoid Teacher	Student should not be scheduled with specified teacher
Student - Student Relationship	Student should be/not be scheduled with specified student
Teacher - Avoid Student	Teacher should not be scheduled with specified student
Teacher - Common Planning	Schedule teachers on a team into a common planning time
Teacher - Concurrent	Allow a teacher to teach multiple courses concurrently
Teacher - Dovetail	Use the minimum periods when schedule a teacher for partial course
Teacher - Max-in-a-row Reset	Reset teacher's max-in-a-row count after a specified period
Teacher - Part-time	Schedule part-time teacher with a minimum span of periods
Teacher - Reserve Time Block	Reserve a time block in a teachers schedule from a list of possible periods
Teacher - Unavailable	Teacher is unavailable for scheduling during these time slots

Fig.4.1 Different types of rules

Rule Definitions :: Course - Blocking :: Schedule 121, 123 in a Simultaneous blocking

Save Cancel

Rules
▸ Details

General Patterns Teams

Rule priority: Hard

Type: Simultaneous

of section blocks:

Use classes:

Class ID: US History I & Skills Class max: 30

Use same teacher:

Courses

ID	Number	Description
<input type="checkbox"/> 1	121	US History I
<input type="checkbox"/> 2	123	US History I Skills

Save Cancel

Fig.4.2 Detail of a course blocking rule

4 Conclusions

The system demonstrated here provides a complete solution to the scheduling needs of secondary schools. It contains a flexible and friendly user interface with simple patterns and easy-to-understand rules that captures the uniqueness of all type of schedules. The fully automated solver accommodates complex scheduling needs and produces high quality schedules in a reasonable period of time. It also provides powerful tools to provide feedback and assist manual adjustments. In addition, multiple scheduling scenarios can be built separately for a school and then combined into an unified schedule.

Appendix A. Glossary

Section: Section is an instance of a course. A section needs to be scheduled with an available teacher, an appropriate time slot and a suitable room. A course has one or more sections.

Time slot: A time slot specifies when a section is scheduled. It contains two components: the schedule term and the day-period combinations within the duration of the schedule term. For example, a section can be scheduled on day 1 period 1 and day 2 period 2 for the first semester of the year.

Build: Build is the process which assigns teachers, time slots and rooms to each section in the master schedule. The build also ensures all hard constraints are satisfied.

Load: Load is the process which schedules students into sections based on their requests.

Bell schedule: A bell schedule specifies the starting time and duration of each period during a day. A school can have more than one mutually exclusive bell schedules used to schedule students in different grades or different campuses.

Partial course: A course that is not scheduled every day in the cycle and every term in the year. For example, a semester course or a full year course that meets only 3 days out of the 5 day cycle is considered as partial course.

Partial content: Students are allowed to take only a portion of a particular course. For example, a student who fails the second semester of a full year course can take just the second semester of the same course in subsequent year.

Inclusion: Inclusion students are special education students who are scheduled into the same section with regular education students.

Section type: It is an attribute on both a section and a student request that allows only some students to be scheduled into a particular section.