METHOD AND SYSTEM FOR EXAMINATION TIMETABLE SCHEDULING USING TEST PAPER CONFLICTIVE ANALYSIS AND SWARM INTELLIGENCE

Field of the Invention

This invention relates to a system for scheduling events with an overlapping set of participants in time slots. More particularly, this invention relates to a system that uses conflict analysis to reduce the number of comparisons needed to confirm events that are scheduled in a manner such that a common participant is not included in more than one event scheduled in the same time slot. Still more particularly, this invention relates to a system for scheduling examinations with an overlapping set of students in time slots using conflict analysis to reduce the number of comparisons needed to confirm that examination are scheduled in a manner such that students do not take more than one examination in a given time slot.

Prior Art

Event scheduling is a problem in many fields. Education is one particular field where event scheduling may be problematic. In particular, many educational institutions including colleges, universities, and training centers must schedule examinations for all of the courses offered by the institution within a certain examination period. Thus, N number of examinations must be given in M time slots where M is less than N and often substantially less the N. A primary constraint of examination scheduling is that no student is scheduled to sit for two or more examinations in the same time slot. Secondary constraints such as examinations having a common set of students should be spaced well apart from one another to allow study time between examinations and that examinations with a large number of students should be allocated an earlier time slot in the examination period.

Currently, those skilled in the art have provided many types of timetable solution searching algorithms to solve the scheduling problems. Some of these algorithms make use of artificial intelligence techniques such as, tabu-search, genetic algorithms, evolutionary computing, fuzzy expert systems, graph coloring and variable neighbor search (VNS). It is a problem that systems incorporating these algorithms require a great number of comparisons to check possible solutions. The great number of comparisons is needed to compare the student participants registered in each examination in the same and/or adjacent time slots to determine whether the primary and secondary constraints are met. Thus, a data set with a large number of student participants may require a great

amount of time to provide a solution. For example, Nelishia Pillay and Wolfgang Banzhaf reported that when using a genetic programming approach to solve the examination scheduling problem, the run time for examination schedule for a university having a typical number of students and offering a typical number of examinations was approximately three and a half hours and for larger data sets, the time required to return a result could be as long as four hours.

Furthermore, the time in these cases is often greatly increased by allowing manual modification of the schedule. The results from the clash checking again require a great number of comparisons of students registered in a pair of classes. Thus, those skilled in the art are constantly striving to provide an algorithm that can resolve event scheduling requiring fewer comparisons during clash checking to reduce the number of comparisons needed and improving run time of the program.

Summary of the Invention

The above and other problems are solved and an advance in the art is made by a method and system for performing event scheduling in accordance with this invention. A first advantage of a system and method in accordance with this invention is that a primary constraint is checked by using an indirect comparison of intersection of participants of events. Thus, the number of comparisons needed to verify a schedule meets a primary constraint is greatly reduced which in turn greatly reduces the run time for providing an acceptable schedule. A second advantage of this invention is that secondary constraints may also be checked using the indirect comparison. Thus, the number of comparisons needed to check a secondary constraint is also reduced which in turn also reduces the run time needed to provide the schedule. A third advantage of a method and system in accordance with the present invention is that the manual modifications to a schedule can be checked using the indirect comparisons further reducing the number of comparisons needed and improves the run time for providing a schedule.

In accordance with embodiments of this invention, a number of events are scheduled in a number of time slots where the number of time slots is less than the number of events. Each event has a set of participants and each participant participates in more than one event. A primary constraint of a schedule is that no two events having at least one common participant may be scheduled in the same time slot. In accordance with the embodiments, a schedule is produced by the following process. First, participant data for each of the participants is received. The participant data for each particular

participant includes a list of each of the events that include that particular participant. A list of participants for each event is then generated from the participant data received. All possible pairing of two events are then generated and the intersection of participants participating in both events in each pair is determined. Intersection coefficients for each pair is then determined where an intersection coefficient is the number of participants in the intersection of a pair of events. The method and/or system may then determine whether two events may be placed in a same time slot in a schedule by reading the intersection coefficient of the pair including the two events and comparing to a threshold. In accordance with particular embodiment of this invention, the method and/or system may be used to provide a schedule of examinations where the participants are students, the events are the examinations for courses of study and the threshold is zero in that no student can participate in two or more examinations during one time slot.

In accordance with some embodiments of this invention, the intersection coefficients are stored as a matrix representing the pairs in a memory. When a determination must be made, a coefficient relating to a particular pair of events is read from the matrix.

In accordance with some embodiments of this invention, lists of mutually exclusive events may be generated for each of the events. A list of mutually exclusive events for a particular event includes those other events that form a pair with a particular event that has an intersection coefficient greater than zero. In accordance with some of these embodiments, a determination that two events must be assigned different time slots may be made responsive to a first event being included in the list of mutually exclusive events of the other event.

In accordance with some embodiments of this invention, an artificial intelligence algorithm generates a schedule of events. In some particular embodiments, a swarm artificial intelligence is used. In accordance with some of these embodiments, the artificial intelligence algorithm uses the lists of mutually exclusive events to determine which events must be scheduled in different time slots in the schedule. In accordance with some of these embodiments, the validity scheduling of a pair of events in the same time slot in the schedule is determined using the intersection coefficients of the pairs of events. In still further of these embodiments, the schedule is optimized with respect to a second condition using information about the intersection of participants in the pairs of

events. The information may be selected from the intersections of the pairs, the intersection coefficients of the pairs and the list of mutually exclusive events.

In accordance with some of these embodiments, the artificial intelligence algorithm may provide multiple schedules that meet the primary constraint. A user may then select one of the schedules. Some of these embodiments allow the user to modify the selected schedule. The modified schedule is then checked via the intersection coefficients of each pair of events scheduled in the same time slot.

Brief Description of the Drawings

The above and other features and advantages of a system and method in accordance with this invention are described in the following detailed description and are shown in the following drawings:

Figure 1 illustrating an organizational diagram of courses offered by an educational institution;

Figure 2 illustrating an organization diagram of an examination for courses offered by an educational institution;

Figure 3 illustrating an organizational diagram of a course of study;

Figure 4 illustrating an organizational diagram of a course of study;

Figure 5 illustrating a network including processing systems that may provide a method and/or system in accordance with embodiments of this invention;

Figure 6 illustrating a block diagram of components of a processing system providing a method and/or system in accordance with embodiments of this invention;

Figure 7 illustrating a flow diagram of a process for scheduling events in a time slots in accordance with embodiments of this invention;

Figure 8 illustrating a flow diagram of a process for determining a schedule is acceptable in accordance with some embodiments of this invention;

Figure 9 illustrating a flow diagram of a process for determining a schedule is acceptable in accordance with other embodiments of this invention;

Figure 10 illustrating a flow diagram of software modules and data passed between modules in accordance with one embodiment of this invention;

Figure 11 illustrating a Venn diagram of the intersection of students taking two examinations in accordance with the embodiment of this invention shown in Figure 10;

Figure 12 illustrating a diagram of a matrix of intersection coefficients in accordance with the embodiment of this invention shown in Figure 10;

Figure 13 illustrating an organizational diagram of a list of mutually exclusive events for examinations in accordance with the embodiment of this invention shown in Figure 10;

Figure 14 illustrating a flow chart of a process performed by a test paper conflictive analysis module in accordance with the embodiment of this invention shown in Figure 10;

Figure 15 illustrating objects for performing the scheduling in accordance with the embodiment of this invention shown in Figure 10;

Figure 16 illustrating a sample schedule provided by the modules in accordance with the embodiment of this invention shown in Figure 10;

Figure 17 illustrating objects for providing artificial intelligence for providing a schedule of examinations in accordance with the embodiment shown in Figure 10;

Figure 18 illustrating a flow diagram for using artificial intelligence algorithms to provide a schedule of examinations in accordance with the embodiment shown in Figure 10; and

Figure 19 illustrating a flow diagram of an artificial intelligence algorithm in accordance with the embodiment of the invention shown in Figure 10.

Detailed Description

This invention relates to a system for scheduling events with an overlapping set of participants in time slots. More particularly, this invention relates to a system that uses conflict analysis to reduce the number of comparisons needed to confirm events are scheduled in a manner such that a common participant is not included in more than one event scheduled in the same time slot. Still more particularly, this invention relates to a system for scheduling examinations with an overlapping set of students in time slots using conflict analysis to reduce the number of comparisons needed to confirm that examinations are scheduled in a manner such that students do not take more than one examination in a given time slot.

In accordance with this invention, a method and/or system for scheduling events in a plurality of time slots is provided. One particular problem that the provided method and/or system may be used to solve is an examination scheduling time table. Figures 1-4 set forth the problem of scheduling examinations in a limited time period. Figure 1 illustrates an organizational chart of a course of study 100. Course of study 100 may include any number of modules 111-115. Each module 111-115 may be divided into

various activities 121-124. Activities may include, but are not limited to, lectures 121, seminars 122, tutorials 123, and laboratory sessions 124.

Figure 2 illustrates an organizational chart for an examination in accordance with some courses of studies. Examination 200, sometimes interchangeably described as paper 200, includes a predetermined number of modules 210-211 of the course of studies. The groups 221-225 are the groups of students 231-235 in each of the activities for a module that must sit for examination 100. Thus, with reference to figure 3, each examination 301-305 includes a group of students 310-314, termed $G_{k,1}$-$G_{k,Nk}$ and $G_{j,1}$-$G_{j,NJ}$, comprised of individual students 321-340 that must sit for the examination. The set of students represented as $S_{K,1}$-$S_{k,Nk}$ and $S_{j,1}$-$S_{j,NJ}$ respectively for the groups belonging to modules k 210 and j 211. Figure 4 illustrates an alternative representation showing a first group (SG_{Pi}) of students 411-415 ($S_{Pi,1}$-$S_{Pi,PiNs}$) that must sit for first examination 401, $paper_i$, and a second group (SG_{Pi}) of students 421-425 ($S_{Pj,1}$-$S_{Pj,PjNs}$) that must sit for a second examination 402, $paper_j$. From the above Figures, one skilled in the art can see that to perform conflict analysis for two examinations, a great number of comparisons are needed. The number of comparisons increases exponentially as the number of examinations is increased. Thus, those skilled in the art are constantly striving to find a way to reduce the number of comparisons needed in order to reduce the run time needed by a processing system to provide a schedule.

In accordance with embodiments of this invention, a method and/or system for providing a schedule of events is provided by a processing system. Figure 5 illustrates network 500 which includes one or more processing systems that alone or in combination may perform the method or provide a system in accordance with this invention. Router 520 connects server 505 and work stations 525, 526 to form a local network. Server 505 and work stations 525, 526 may be connected either through a wired or wireless connection without departing from this invention. Router 520 also connects to Internet 510 to allow systems on the local networks to communicate with other systems over the Internet. Server 505 is a conventional server that includes a processing system that performs various functions for interacting with work stations 525, 526. Work stations 525, 526 are conventional computers systems, such as laptop or desktop personal computer systems, that perform applications stored in memory and that may communicate with other processing systems via router 520.

Server 515 connects to work stations 531, 532 and Internet 510. Server 515 is a conventional server that provides many functions to work stations 531,532 over a local network including access to Internet 510. Work stations 531,532 are conventional computers systems, such as laptop or desktop personal computer systems, that perform applications stored in memory and that may communicate with other processing systems via server 515.

Figure 6 illustrates an exemplary processing system including the components needed to execute the applications from instructions stored in memory to perform a process for scheduling events in accordance with an embodiment of this invention. One skilled in the art will recognize that the exact configuration of each processing system may be different and the exact configuration for executing processes in accordance with embodiments of this invention will vary from processing system to processing system and this figure is given by way of example only.

Processing system 600 includes Central Processing Unit (CPU) 605. CPU 605 is a processor, microprocessor, or any combination of processors and microprocessors that execute instructions to perform the processes in accordance with the present invention. CPU 605 connects to memory bus 610 and Input/Output (I/O) bus 615. Memory bus 610 connects CPU 605 to memories 620 and 625 to transmit data and instructions between the memories and CPU 605. I/O bus 615 connects CPU 605 to peripheral devices to transmit data between CPU 605 and the peripheral devices. One skilled in the art will recognize that I/O bus 615 and memory bus 610 may be combined into one bus or subdivided into many other busses and the exact configuration is left to those skilled in the art.

A non-volatile memory 620, such as a Read Only Memory (ROM), is connected to memory bus 610. Non-volatile memory 620 stores instructions and data needed to operate various sub-systems of processing system 600 and to boot the system at start-up. One skilled in the art will recognize that any number of types of memory may be used to perform this function.

A volatile memory 625, such as Random Access Memory (RAM), is also connected to memory bus 610. Volatile memory 625 stores the instructions and data needed by CPU 605 to perform software instructions for processes such as the processes for providing a system in accordance with this invention. One skilled in the art

will recognize that any number of types of memory may be used to provide volatile memory and the exact type used is left as a design choice to those skilled in the art.

I/O device 630, keyboard 635, display 640, memory 645, network device 650 and any number of other peripheral devices connect to I/O bus 615 to exchange data with CPU 605 for use in applications being executed by CPU 605. I/O device 630 is any device that transmits and/or receives data from CPU 605. Keyboard 635 is a specific type of I/O that receives user input and transmits the input to CPU 605. Display 640 receives display data from CPU 605 and display images on a screen for a user to see. Memory 645 is a device that transmits and receives data to and from CPU 605 for storing data to a media. Network device 650 connects CPU 605 to a network for transmission of data to and from other processing systems.

This invention relates to a process and/or system that reduces the number of comparisons needed to verify a schedule of events in time slots. The invention reduces the number of comparisons to determine whether a primary constraint is met by using indirect comparisons. Thus, the intersections of sets of participants in pairs of events are determined and then compared to a threshold when the events in the pair are being compared. Thus, the participants in each event in all of the pairs are only compared once at the time the system is initialized. Subsequent comparisons of events will then require a lookup and then comparison to a threshold.

Figure 7 illustrates a flow diagram of a process for verifying a schedule of events into time slots where the participants of events scheduled in the same time slot cannot overlap. Process 700 begins in step 705 by receiving participant data. In the example of examination scheduling, the participant data is the registration information indicating each examination that each student must take. However, more generally, the participant data must show each event that includes the participant information for each of the participants. In step 710, the participant data is used to generate lists of participants for each event. In the examination example, a list of students taking the examinations is generated for each particular examination. The list of participants for each event is then stored in memory.

In step 715, pairs of events are formed for each event with each of the other events. A list of the intersection of participants of the events in the pair is then generated for each pair and stored in memory. In step 720, an intersection coefficient is determined for each of the pairs of events. The intersection coefficient is the number of participants

in the list of the intersection of participants for each pair. In the examination example, the intersection coefficient is the number of students taking both examinations in the pair. A matrix of the intersection coefficients is then generated and stored in step 725. The matrix is a two dimensional representation of the intersection coefficient data in which one axis represents a first event in a pair and the second axis represents a second event in a pair with the intersection coefficient of the pair at the intersection of the events.

In accordance with some embodiments, the intersection coefficients are then used to generate a list of mutually exclusive events for each of the events in step 730. Mutually exclusive events of a first event are those events that are in a pair with the first event that has an intersection coefficient greater than zero. In other words, for an event to be included in the list of mutually exclusive events for another event there must be at least one participant in common between the two events. Thus, the events cannot be scheduled in the same timeslot. The lists of mutually exclusive events are then stored in memory for use in the comparisons.

After step 730 has been performed, the system is initialized and the indirect comparison can be made to verify a schedule. Thus, an artificial intelligence algorithm is then used to generate a proposed schedule in step 735. The schedule is then checked to determine whether the schedule meets a primary constraint using the data generated in the above steps in step 740. Embodiments for processes for performing the check are described below with respect to Figures 8 and 9. If the constraints are not met, the process repeats from step 735 providing another schedule. If the constraints are met by the schedule, the process provides the schedule as a possible solution in step 745. In step 750, the process determines whether the process has provided an adequate number of solutions. If so, process 700 ends. Otherwise process 700 repeats from step 735.

Figure 8 illustrates process 800 that is a method for checking a schedule to determine if a primary constraint of no events having overlapping participants in a same time is met. Process 800 begins in step 805 by selecting a time slot. In step 810, the pairs of events in the time slot are determined. In step 815, one of the pairs in the same time slot is selected. The intersection coefficient of the pair is read from memory in step 820. In accordance with some embodiments, the intersection coefficient is read by indexing the events in the intersection coefficient matrix.

The intersection coefficient is then compared to a threshold in step 825. In the case of examinations, the threshold is zero. If the intersection coefficient is greater than the threshold, the two events in the pair have at least one common participant and cannot be placed in the same time slot. Thus, the schedule is rejected in step 827. If the intersection coefficient is not greater then the threshold, the two events may be in the same time slot and process 800 determines if the time slot includes another pair of events to test in step 830. If so, process 800 repeats from step 815 until all of the pairs of events for the time slot are tested. Otherwise, process 800 determines if the schedule includes another time slot in step 835. If so, process 800 repeats from step 805. Otherwise, if all of the pairs in all the time slots have been tested, the schedule is accepted in step 840 and process 800 ends. In process 800, only one read and one comparison must be performed to test each pair of events in time slot. Thus, the schedule can be checked in an order of N comparisons, where N is the number of events, greatly reducing the run time for checking the schedule.

Figure 9 illustrates process 900 that is an alternative a method for checking a schedule to determine if a primary constraint of no events having overlapping participants in a same time is met using list of mutually exclusive events. Process 900 begins in step 905 by selecting a time slot. In step 910, the pairs of events in the time slot are determined. In step 915, one of the pairs in the time slot is selected. The list of mutually exclusive events for one of the events in the selected pair is read from memory in step 920.

In step 925, the list of mutually exclusive events is read to determine whether the list includes the second event. If the second event is included in the list of mutually exclusive events for the first event, the events cannot be scheduled in the same time slot. Thus, the schedule is rejected in step 927. If the second event is not in the list of mutually exclusive event, the two events may be in the same time slot and process 900 determines if the time slot includes another pair of events to test in step 930. If so, process 900 repeats from step 915 until all of the pairs of events for the time slot are tested. Otherwise, process 900 determines if the schedule includes another time slot in step 935. If so, process 900 repeats from step 905. Otherwise, if all of the pairs in all of the time slots have been tested, the schedule is accepted in step 940 and process 900 ends. In process 900, the conflict check for each pair takes at most N comparisons where N is the number of events. Thus, the number of comparisons needed to check a schedule is in the order of N^2. Thus, the number of comparisons needed to check a

scheduled is constrained and is greatly improved as compared to checking the students in each of the events.

The following is a description of one embodiment of a system for generating an examination schedule in accordance with this invention. In accordance with this embodiment, the system and method are provided by software modules performed by a processing system. The instructions for performing the modules may be implemented using hardware, software or firmware. Figure 10 illustrates an organizational chart of the software modules in accordance with this invention. System 1000 starts in step 1052. Data loader 1054 receives course management information. The course management information includes student registration data and test paper setting data. The student registration data includes a list of examinations that each student must take. The test paper setting data includes all of the information for the course of study. Examples of the information in the test setting data includes, but is not limited to, the exam paper title, exam duration, a list of the test modules (such as modules 210, 211 Shown in Figure 2), other constraint information.

The student registration data and test paper setting data 1056 are provided to test paper conflictive analyzer 1058. Test paper conflictive analyzer 1058 uses the student registration data to generate the lists of students in each examination, the intersection coefficient for each pair of examinations, the lists of mutually exclusives examinations for each examination. A matrix of intersection coefficients and a matrix of mutually exclusive lists 1060 are generated by test paper conflictive analyzer 1058 and passed to Particle Swarm Optimization (PSO) based examination timetable scheduler 1062. PSO based exam timetable scheduler 1062 uses an artificial intelligence algorithm to provide an examination timetable. In this embodiment, PSO based examination timetable scheduler 1062 uses a swarm intelligence to generate the schedule and/or schedules. However, other artificial intelligence algorithms may be used without departing from this invention. The schedule and/or schedules 1064 produced by PSO based exam timetable scheduler 1062 are then provided to timetable uploader 1066 that uploads the schedule to the proper course and or administrator database and system 1000 completes operation in step 1068.

Figure 11 illustrates the use of an intersection coefficient to measure the conflictive level of two papers. As can be seen, $Paper_i$ 1142 and $Paper_j$ 1144 are two different examinations and $C_{Pi,Pj}$ 1182 represents the intersection coefficient of the

examinations which is the students that must sit for both examinations. The Venn diagram below show the students sitting for $Paper_i$ 1142 in circle 1184 and the students sitting for $Paper_j$ 1144 in circle 1188. The overlap 1186 of circles 1184 and 1188 represents the students that must sit for both examinations or $C_{Pi,Pj}$ 1182. Thus, by calculating the intersection coefficients of each pair of examinations ahead of time, the number of comparisons needed to check the scheduling of examinations in timeslots is greatly reduced. The primary constraint of examination scheduling is that no student can sit for two examinations in the same time slot. By calculating the intersection coefficients, the primary constraint can be quickly checked using conflict coefficients of pairs of exams or a derivation thereof.

Figure 12 illustrates a representation of matrix of intersection coefficients data structure 1200 for storing the intersection coefficients of each pairs of examinations. The matrix, φ, is an N x N matrix where N is the number of examinations and can be expressed as follows:

$$\Phi = [C_{i,j}]_{n \times n}$$

Where the element $C_{i,j}$ is an intersection coefficient of $Paper_i$ and $Paper_j$. Note that $C_{i,j} = 0$ if i=j as a paper can never be conflictive or mutually exclusive to itself. Furthermore matrix Φ is symmetrical, that is, $C_{i,j} = C_{j,i}$ as the conflict coefficient is the same for two examinations regardless of order in the pair.

As shown in figure 12, y-axis 1201 and x-axis 1202 of matrix 1200 list the examinations 1 through N represented as P_1 through P_n. Each row 1211-1216 represents the intersection coefficients for one exam denoted on y axis 1201. For example, row 1211 denotes all of the examination coefficients for exam P_1 where each coefficient is represented as $C_{x,y}$ where x and y denote the corresponding exams that make the pair. Thus, when comparing a pair of examinations, only one look-up in the matrix is needed to determine if the primary constraint is met.

Figure 13 illustrates a representation of a matrix data structure for lists of mutual exclusive examinations for each examination. In accordance with this embodiment, a list of mutually exclusive examinations is generated for each examination P_1 through P_n. from matrix 1200. To form the list for each paper, $paper_i$, each intersection coefficient, $C_{i,j}$, is associated with the compared j to form value pair $\lambda_{i,j}$ as follows:

$$\lambda_{i,j} = \langle P_j, C_{i,j} \rangle$$

To create the value pairs, matrix 1200 is read for every examination in the y-axis 1201. For each examination in the x-axis 1202 that has an intersection coefficient greater than 0, a value pair is created and added to the list of mutually exclusive examinations for the examination in 1201. For example, examination P_1 has intersection coefficient with P_2 that is 2. In other words, 2 students must take both examinations. Thus, value pair, $\lambda_{1,2}$ (P_2,2) is created and added to the list of mutually exclusive examinations of P_1. P_3 has intersection coefficient with P_1 that is 0. Thus, no value pair is made for P_3.

As can be seen, the list of mutually exclusive examinations for each examination is less than N where N is the number of examinations as at the very least a paper cannot be mutually exclusive of itself. List of mutually exclusive examinations 1321-1325 are generated for each examination 1301-1305 including each examination that is mutually exclusive to the particular examination and may be expressed as follows:

$$MEL_i = \{ \lambda_{1,j}, \lambda_{i,j+1}, \ldots \lambda_{i,nPj}\}$$

Where n is the number of examinations that are mutually exclusive to P_i. The lists of mutually exclusive examinations are used in the timetable for clash checking. Thus, if $T(P_i)$ denotes the time slot of exam P_i, for any exam P_j, which is denoted in value pair on the list of mutually exclusive exam of exam P_i, the following constraint cannot be violated:

$$T(P_i) \neq T(P_j)$$

If there are P_iN_p students, for example 500 students that must sit for examination P_i (as shown in Figure 11) and 400 students sitting for examination P_j (as Shown in Figure 11) conventional clash checking requires up to 500 * 400 or 200000 comparisons to determine a clash. However, only number of comparison equal to the member of value pairs the list of mutually exclusive list of P_i. (a determination if P_j is in a value pair in the list of P_i) is needed. As the typical size of the lists is typically limited to 10 or less, the number of comparisons is greatly reduced from the 200000 described for the prior clash checking.

Figure 14, illustrates a flow diagram of process 1400 performed by paper conflictive analyzer 1058 in accordance with the described embodiment of this invention. Process 1400 begins in step 1404 by receiving the student registration data and test paper setting data. In step 1406, a list of students sitting for the examination is prepared for each paper. The lists may be represented as follows:

$S_{Pi} = \{S_{Pi,1}, S_{Pi,2}, \ldots, S_{Pi,PiNs}\}$ where P_i is the examination and PiNs is the number of students taking the exam. After the list of students is generated for each exam, a nested

loop is used to determine the intersection set of students of all of the examinations from the lists of students for each examination in step 1408.

After the intersection sets are determined for each pair of examinations, the intersection coefficient for each pair of examinations is determined and the intersection matrix 1200 (Figure 12) is generated in step 1410. In step 1412, a value pair, $\lambda_{Pi,Pj} = <P_j, C_{Pi,Pj}>$, is determined for each pair in matrix 1200. The lists of mutually exclusive examinations are then generated in step 1414. In step 1416, the matrix 1200 and lists of mutually exclusive exams are stored in memory for use in schedule checking. Process 1400 then ends.

Figure 15 illustrates a definition of test paper object that is instantiated for each test paper (examination) to perform the time scheduling. The test paper class includes a Paper object attributes 1502, Time Scheduling and Resource allocation attributes 1504, Test paper Conflictive Analysis Parameters 1506, Paper Object Methods 1508, and Methods for Drag Drop features 1510. Paper object attributes 1502 includes a paper identification, title of paper, duration, and number of participants indicating the known attributes of the examination. Time Scheduling and Resource allocation attributes 1504 include booked_time, scheduled_time, venue_allocated, and invigilator_allocated that may be assigned using the scheduler. Test Paper Conflictive Analysis Parameters 1506 includes the number of mutually exclusive examinations and the list of mutually exclusive examinations for the examination instantiated by the object. Paper Object Methods 1508 includes a constructor function, draw function, and set_time slot functions. These are the functions needed to set the function in time slot and represent the function on a display schedule. Methods for Drag Drop features 1510 includes objects needed to use a mouse or other input device to manually change the scheduled time slot of an examination.

Figure 16 illustrates a display of a schedule of examinations 1600 provide by system 1000 in accordance with this embodiment. Display 1600 includes a y-axis 1602 displaying identifications of all of the examinations. Bottom x-axis 1608 includes a list of time slots available while top x-axis divides the timeslots into days 1611-1614. Indicia 1621-1626 are filled into the graph indicating an examination on the y-axis is scheduled in the corresponding time slot on the x-axis. The user may then use inputs to manually change the designated time slots. The resulting scheduled then is checked for constraint violations.

Figures 17-19 describe the objects and algorithm used to schedule using a swarm method of artificial intelligence. Figure 17 illustrates a particle object definition 1702 and Swarm intelligence definition 1704. Particle object 1702 includes the following attributes: particle_id, position p^{pk}, fitness $\sum F_{pk}$, and velocity v_{pk}. Particle object also has the following functions: particle best fitness F_{PBest} and particle best position $p^{Pbest()}$. Swarm 0bject 1704 includes the following attributes: swarm size N_{Psize}, particles(). Swarm object also includes the following functions: global best fitness F_{GBest} and global best position $p^{Gbest()}$. These objects are instantiated to perform the algorithms described in Figures 18 and 19.

Figure 18 illustrates process 1800 to output a schedule. Process 1800 begins in step 1804 by receiving problem definition data. The definition data includes test paper information data, time space information and constraints. In step 1806, the lists of mutually exclusive examinations for each examination are received. In step 1808, the swarm intelligence searching parameters are read and set. The searching parameters include swarm size, N_{Psize} maximum velocity V_{max}, w_i, c_1 and c_2. Typically, the numerical values of these parameters are set as follows. The swarm size or number of swarm particles, $N_{Psize,}$ is set from 50 to 200 depending on the size and complexity of the scheduling problem. The maximum velocity, V_{max}, is set to 1.0 to 1.2. Inertia weight, w_i, is from 0.2 to 0.5. The constants c_1 and c_2 can set to 1.2. The searching parameter N_{max}, the max number of searching cycles can set to C where C normally is set to 20 to 50 depending on the problem size. However, those skilled in the art will recognize that exact parameters will depend upon the number of exams being scheduled and the total number of students participating in the exams.

In step 1810 the search is initialized and performed by the swarm intelligence. The initialization of the swarm intelligence includes randomly locating each searching particle, N_{psize}, at a position in the search space and computing the initial best known solution among the N_{psize} searching particles in the search space. The best known solution is then derived by performing search cycles until the criterion is met and a schedule or list of possible schedules are presented. Details about the performance of the search cycles is described below with reference to Figure 19. Either a selected schedule or the provided schedule may then by manually changed in step 1812 and a schedule is output in step 1816. Process 1800 then ends.

Figure 19 illustrates process performed using swarm intelligence. Process 1900 begins in step 1904 by initializing the swarm intelligence. To initialize the swarm intelligence, the swarm size, N_{Psize}, is first defined. Preferably, N_{Psize} is defined as 50 to 200 particles depending on the size and complexity of the scheduling problem. Then, each particle, $p^k(i)$, is initialized in the search space with a velocity, v_k and the fitness $\sum F_{pk}$ of the particle is determined. Each $p^k(i)$ is the position of a particle, p^k, in the searching space. Thus, $p^k(i)$ represents a proposed scheduled time slot and each $p^k()$ has n elements such that $p^k() = \{ p^k_1, p^k_2, p^k_3, \ldots, p^k_n\}$ where the index i is from 0=1 to n, where n is the total number exam papers being scheduled. The value of $p^k(i)$ is a time slot selected from the available time slots, t_{Nt}, such as time slots 1608 shown in Figure 16. The best position of a particle is denoted as $p^{Pbest}()$ and is initialized to an initial position, $p^k()$, of the particle. Note that for every particle (k), the system initializes velocity of the particle, v_j, to zero and computes fitness, Σf_{pk}, of each particle according to the fitness of the particle with regards to the validation of the constraint. In a swarm having N_{Psize} particle objects, the system computes and finds the best fitness, F_{Gbest}, and correspondent best particle position in the swarm, denoted as $p^{Gbest}()$, in step 1904. In step 1904, the searching cycle counter N_C is set to zero. Note that at very beginning of each searching cycle, the particle index k is also set to zero in step 1908.

In Step 1910, the two random components, r_1 and r_2, are generated for each solution component represented by a particle, p^k, where r_1, r_2 values are in range from 0 to 1. The velocity, v_j, (j=1to n, n is number of papers) is updated for particle (k) as shown in the following equation where k =1 to N_{Psize}:

$$v_j = wv_j + c_1 r_1 (p_j^{Pbest} - p_j^k) + c_2 r_2 (p_j^{Gbest} - p_j^k) \quad (1)$$

Where, v_j is particle velocity for component j in particle (k)'s position vector; w is inertia weight; p^k_j is current position of paper index j in the solution space for particle (k). The j is a solution component, i.e., test paper index 1602 as shown in Figure 16; p_j^{Pbest} is the best location of the particle in solution space in current search cycle, p_j^{Gbest} is best location of a particle in the swarm; and c_1 and c_2 are constants.

The position of particle (k) at j^{th} solution component is moved to a new position, p^k_j, shown as follows:

$$p_j^k = p_j^k + v_j \qquad (2)$$

In the above equation, the velocity, v_j, and the new position, p^k_j, must be controlled by the searching program so that the new position is within the range of a pre-defined time space, ψ, where $t \sim \psi\ (1 - t_{Nt})$ 1608 as shown in Figure 16.

Each of the particles will move to a new position after p^k_j is computed using Equation (2), given above, and the fitness, $\sum F_{pk}$, of each particle is then calculated for the new position of the particular particle. The fitness, $\sum F_{pk}$, of a solution of a particle is computed according to the evaluation of the scheduling constraints imposed and criterion specified.

In step 1912, the calculated current fitness, $\sum F_{pk}$, of the particle is compared to the particle best fitness F_{PBest} and if the current fitness is better than the best fitness, then F_{PBest} is set to the current particle position. K is then incremented in step 1914 and compared to the swarm size, N_{Psize}. If $k \geq N_{Psize}$, process 1900 proceeds to step 1918. Otherwise, process 1900 repeats from step 1910.

In step 1918, the best position of the particle, F_{PBest} is compared to the global best position, F_{PBest}, to determine if F_{PBest} is a better solution. If F_{PBest} is a better solution, F_{GBest} is set to F_{PBest} and the best known position p^{pbest} of the swarm is updated. The search count is then incremented in step 1920 and compared to a maximum count in step 1922. If the search count is greater than or equal to the maximum count, the process 1900 ends. Otherwise, process 1900 repeats from step 1908.

The above describe embodiments of event scheduling system. In a particular an embodiment for examination time scheduling is described. It is envisioned that others can and will design alternatives that infringe on this invention as set forth in the claims below.

What is claimed is:

1. A method performed by a processing system for scheduling a plurality of events in a plurality of time slots wherein each of said plurality of events includes a plurality of participants, each of said plurality of participants participates in more than one of said plurality of events, and a requirement that two of said plurality of events scheduled in a same one of said plurality of time slots must less than a predetermined threshold of common participants, said method comprising:

receiving participant data for each of said plurality of participants wherein said participant data for each particular participant includes a list of each of said plurality of events that include said particular participant;

generating a plurality of lists of participants from said participant data wherein each of said plurality of lists of participants includes each of said plurality of participants participating in one of said plurality events;

generating a plurality of pairs of two of said plurality of events wherein said plurality of pairings includes all of the possible pairings of two of said plurality of events;

determining plurality of intersections of participants for said plurality of pairs wherein each of said plurality of intersections is a group of participants in each of said plurality of events of said two events in one of said plurality of pairs;

determining a plurality of intersection coefficients wherein each of said plurality of intersection of coefficients is a number of said plurality of participants in said group for one of said plurality of pairs;

determining whether two of said plurality of events may be scheduled in a same one of said plurality of time slots by determining whether said one of said plurality of intersection coefficients of said one of said plurality of pairs for said two of said plurality of events is equal to or less than a predetermined threshold.

2. The method of claim 1 wherein said each of said plurality of participants is a student.

3. The method of claim 2 wherein each of said plurality of events is an examination for a course of study.

4. The method of claim 3 wherein said predetermined threshold is zero.

5. The method of claim 1 further comprising:

storing said plurality of intersection coefficients in a matrix representing said plurality of pairs in a memory.

6. The method of claim 5 further comprising:

reading said one of said plurality of intersection coefficients from said matrix to perform said step of determining.

7. The method of claim 1 further comprising:

generating a plurality of lists of mutually exclusive events wherein each of said plurality of lists of mutually exclusive events is related to one of said plurality of events and includes each of said plurality of events that has an intersection coefficient with said one of said plurality of events that is greater than zero.

8. The method of claim 7 further comprising:

determining two of said plurality of events must be assigned to different ones of said plurality of time slots from said plurality of lists of mutually exclusive events.

9. The method of claim 7 further comprising:

generating a schedule of said plurality of events in said plurality of time slots using an artificial intelligence program.

10. The method of claim 9 wherein said artificial intelligence algorithm uses said plurality of lists of mutually exclusive events to determine each of said plurality of events that may be scheduled in one of said plurality of time slots.

11. The method of claim 9 further comprising:

checking a validity of each pair of said plurality of events scheduled in a same one of said plurality of time slots in said schedule using step of determining whether said two of said plurality of events may be placed in a same one of said plurality of time slots using one of said plurality of intersection coefficients.

12. The method of claim 8 further comprising:

optimizing said schedule with respect to a second condition using data related to said plurality of intersections of said participants for each of said pairs of said plurality of events.

13. The method of claim 8 further comprising:

generating a plurality of schedules of said events in said plurality of time slots including said schedule using said artificial intelligence algorithm.

14. The method of claim 12 wherein said artificial intelligence algorithm uses said plurality of lists of mutually exclusive events to determine each of said plurality of events that may be scheduled in one of said plurality of time slots.

15. The method of claim 12 further comprising:

checking a validity of each pair of said plurality of events scheduled in a same one of said plurality of time slots in each of said plurality of schedules using step of determining whether said two of said plurality of events may be placed in a same one of said plurality of time slots using one of said plurality of intersection coefficients.

16. The method of claim 8 further comprising:

returning a set of said plurality of schedules that meet said requirement.

17. The method of claim 16 further comprising:

receiving a selection of one of said plurality of schedules from said set of schedules;

receiving an input of a change to said select one of said plurality of schedules;

generating a modified schedule with said change incorporated into said selected one of said plurality of schedules; and

checking each pair of events that are scheduled in each one of said plurality of time slots in said modified schedule using said plurality of said intersection coefficients to determine whether said modified schedule meets said requirement.

18. The method of claim 16 further comprising:

optimizing said schedule with respect to a second condition using data related to said plurality of intersections of said participants for each of said pairs of said plurality of events.

19. A product for scheduling a plurality of events in a plurality of time slots wherein each of said plurality of events includes a plurality of participants, each of said plurality of participants participates in more than one of said plurality of events, and a requirement that two of said plurality of events scheduled in a same one of said plurality of time slots

must less than a predetermined threshold of common participants, said product comprising:

instructions for directing a processing unit to:

receive participant data for each of said plurality of participants wherein said participant data for each particular participant includes a list of each of said plurality of events that include said particular participant,

generate a plurality of lists of participants wherein each of said plurality of lists of participants includes each of said plurality of participants participating in one of said plurality events,

generate a plurality of pairs of two of said plurality of events wherein said plurality of pairings includes all of the possible pairs of two of said plurality of events,

determine plurality of intersections of participants for said plurality of pairs wherein each of said plurality of intersections is a group of participants in each of said plurality of events of said two events in one of said plurality of pairs,

determine a plurality of intersection coefficients wherein each of said plurality of intersection of coefficients is a number of said plurality of participants in said group for one of said plurality of pairs, and

determine whether two of said plurality of events may be scheduled in a same one of said plurality of time slots by determining whether said one of said plurality of intersection coefficients of said one of said plurality of pairs for said two of said plurality of events is equal to or less than a predetermined threshold; and

a media readable by said processing unit that stores said instructions.

20. The product of claim 19 wherein said each of said plurality of participants is a student.

21. The product of claim 20 wherein each of said plurality of events is an examination for a course of study.

22. The product of claim 21 wherein said predetermined threshold is zero.

23. The product of claim 19 further comprising:

instructions for directing said processing unit to store said plurality of intersection coefficients in a matrix representing said plurality of pairs in a memory.

24. The product of claim 23 further comprising:

instructions for directing said processing unit to read said one of said plurality of intersection coefficients from said matrix to perform said step of determining.

25. The product of claim 19 further comprising:

instructions for directing said processing unit to generate a plurality of lists of mutually exclusive events wherein each of said plurality of lists of mutually exclusive events is related to one of said plurality of events and includes each of said plurality of events that has an intersection coefficient with said one of said plurality of events that is equal to zero.

26. The product of claim 25 further comprising:

instructions for directing said processing unit to determine two of said plurality of events may be assigned to a same one of said plurality of time slots from said plurality of lists of mutually exclusive events.

27. The product of claim 25 further comprising:

instructions for directing said processing unit to generate a schedule of said plurality of events in said plurality of time slots using an artificial intelligence program.

28. The product of claim 27 wherein said artificial intelligence algorithm uses said plurality of lists of mutually exclusive events to determine each of said plurality of events that may be scheduled in one of said plurality of time slots.

29. The product of claim 27 further comprising:

instructions for directing said processing unit to check a validity of each pair of said plurality of events scheduled in a same one of said plurality of time slots in said schedule using step of determining whether said two of said plurality of events may be placed in a same one of said plurality of time slots using one of said plurality of intersection coefficients.

30. The product of claim 26 further comprising:

instructions for directing said processing unit to optimize said schedule with respect to a second condition using data related to said plurality of intersections of said participants for each of said pairs of said plurality of events.

31. The product of claim 26 further comprising:

instructions for directing said processing unit to generate a plurality of schedules of said events in said plurality of time slots including said schedule using said artificial intelligence algorithm.

32. The product of claim 31 wherein said artificial intelligence algorithm uses said plurality of lists of mutually exclusive events to determine each of said plurality of events that may be scheduled in one of said plurality of time slots.

33. The product of claim 31 further comprising:

instructions for directing said processing unit to check a validity of each pair of said plurality of events scheduled in a same one of said plurality of time slots in each of said plurality of schedules using step of determining whether said two of said plurality of events may be placed in a same one of said plurality of time slots using one of said plurality of intersection coefficients.

34. The product of claim 31 further comprising:

instructions for directing said processing unit to return a set of said plurality of schedules that meet said requirement.

35. The product of claim 31 further comprising:

instructions for directing said processing unit to:

receive a selection of one of said plurality of schedules from said set of schedules,

receive an input of a change to said select one of said plurality of schedules,

generate a modified schedule with said change incorporated into said selected one of said plurality of schedules, and

check each pair of events that are scheduled in each one of said plurality of time slots in said modified schedule using said plurality of said intersection coefficients to determine whether said modified schedule meets said requirement.

36. The product of claim 35 further comprising:

instructions for directing said processing unit to optimize said schedule with respect to a second condition using data related to said plurality of intersections of said participants for each of said pairs of said plurality of events.

37. A system provided by a processing device for determining a time schedule for a plurality of examinations in a plurality of time slots wherein said time schedule has a primary constraint that two of said plurality of examinations having common students are scheduled in a different ones of said plurality of time slots comprising:

a data loader module that loads student information data for a plurality of students that are participating in said plurality of examinations and course management information related to said plurality of examinations;

a test paper conflictive analyzer module that receives said student information, determines a plurality of sets of said plurality of students wherein each of said plurality of sets is a set of students required to sit for one of said plurality of examinations, and determines intersection information for each pair of said plurality of examinations; and

an examination time table scheduler module that uses an artificial intelligence algorithm to determine said time schedule using said intersection information to test for conflicts with said primary constraint.

38. The system of claim 37 further comprising:

a timetable loader module for providing said scheduler to an examination solution database.

39. The system of claim 37 wherein said test paper conflictive analyzer module generates an intersection coefficient for each pair of said plurality of examinations.

40. The system of claim 39 wherein said test paper conflictive analyzer module generates an intersection coefficient matrix including said intersection coefficient for each pair of said plurality of examinations.

41. The system of claim 40 wherein said test paper conflictive analyzer module generates a plurality of lists of mutually exclusive examinations wherein each one of said plurality of list of mutually exclusive examinations is associated with one of said plurality of examinations.

42. The system of claim 41 wherein examination time table scheduler module uses said plurality of list of mutually exclusive examinations to test for conflicts.

43. The system of claim 38 wherein examination time table scheduler module uses a swarm intelligence as said artificial intelligence algorithm.

44. The system of claim 38 wherein examination time table scheduler module allows a user to modify said schedule and test for conflicts in said modified schedule using said intersection information.

45. The system of claim 38 wherein said examination time table scheduler module optimizes said schedule for a secondary constraint.