Timetable scheduling using graph coloring

Cauvery N K[1]
[1]Associate Prof, Department of CSE, RVCE,
Bangalore - 560059, Karnataka, India.

Abstract

The problem of constructing an automated system for timetabling is a particularly well known one. Timetabling is a common example of a scheduling problem and can manifest itself in several different forms; the particular form of timetable required is specific to the environment or organization in which it is needed. Many programs exist for this task but they perform well only in particular isolated environment. With the help of graph coloring, it is proposed to develop a general system that can cope with the ever changing requirements of large educational institutions. Timetabling problem is a NP-hard problem. many combinatorial problems (optimization problems) are NP-hard. It is generally believed that NP-hard problems cannot be solved to optimality within times which are polynomial bounded functions of input size. Therefore there is much interest in heuristic algorithms which can find near optimal solutions within reasonable running time. One of the most studied NP-hard problems is the "graph coloring problem". Graph coloring has numerous applications in scheduling and other practical problem; "timetabling" is one of them. One of the heuristic approaches to solve graph coloring is "Ant algorithm" [1].

Keywords: *Graph coloring, Ant colony optimization, Pheremone trails*.

1. Introduction

Timetable Problem represents an important class of optimization problem in Operations Research. It is considered as one of the most difficult problems faced by universities and colleges today. The problem can be defined as allocation of given resources (teachers, classrooms) to objects (courses) being placed in space time satisfying all university constraints and optimizing utilization of existing facilities such that a set of desirable objectives are satisfied [1]. Basically, university timetable problem exists in two forms viz., course and exam timetable formats. Here focus is only on course timetable problem. The university course timetable requires several slots and with different categories such as lectures, tutorials and practical sessions, which fits within a week and repeats for whole semester. Each classroom has different capacities which make assignment of courses to classrooms complicated. Furthermore, it is not only enough to schedule course in classroom with higher capacity than the number of enrolled students, since this can still lead to inefficient utilization of classrooms which can cause difficulties for teachers and students. The automation of timetable problem is thus an important task as it saves lot of man-hours to institutions and provides optimal solutions that can boost productivity, quality of education and services. However, large-scale timetables such as university timetables may need many hours of work spent by qualified person or team in order to produce high quality timetables with optimal constraint satisfaction.

2. Graph coloring

In graph theory, graph coloring is a special case of graph labeling; it is an assignment of labels traditionally called "colors" to elements of a graph, subject to certain constraints. In its simplest form, it is a way of coloring the vertices of a graph such that no two adjacent vertices share the same color; this is called a vertex coloring [2]. Similarly, an edge coloring assigns a color to each edge so that no two adjacent edges share the same color, and a face coloring of a planar graph assigns a color to each face or region so that no two faces that share a boundary have the same color. Vertex coloring is the starting point of the subject, and other coloring problems can be transformed into a vertex version. For example, an edge coloring of a graph is just a vertex coloring of its line graph, and a face coloring of a planar graph is just a vertex coloring of its planar dual. However, non-vertex coloring problems are often stated and studied as is. That is partly for perspective, and partly because some problems are best studied in non-vertex form, as for instance is edge coloring.

The convention of using colors originates from coloring the countries of a map, where each face is literally colored. This was generalized to coloring the faces of a graph embedded in the plane. By planar duality it became coloring the vertices, and in this form it generalizes to all graphs. In mathematical and computer representations it is typical to use the first few positive or nonnegative integers as the "colors". In

general one can use any finite set as the "color set". The nature of the coloring problem depends on the number of colors but not on what they are.
Graph coloring enjoys many practical applications as well as theoretical challenges. Different limitations can also be set on the graph, or on the way a color is assigned, or even on the color itself. It has even reached popularity with the general public in the form of the popular number puzzle Sudoku. Graph coloring is still a very active field of research.

2.1 Definition and Terminology of Graph Coloring

The Graph Coloring Problem (GCP) is one of the most studied NP-hard problems in graph's theory, completeness theory and operational research [2]. Its importance is justified by its diverse and interesting applications such as timetabling and resource assignment.

Definition: A graph k-coloring which can be stated as follows: given an undirected graph G with a set of vertices V, and a set of edges E, a k-coloring of G consists of affecting to each vertex of V a color such that any two adjacent vertices have different colors. Formally, a k-colorings of G=(V,E) can be stated as a function C from V to a set of colors K such that |K|=k and $C(u) \neq C(v)$ whenever E contain an edge (u,v) for any two vertices u and v of V (assignment approach). The graph k-coloring can also be stated as a partition of V into k stables called color sets $S_1 \ldots S_k$ where every vertex in S_i has the same color i. The minimal number of colors k for which a k-coloring exists is called the chromatic number of G and is denoted by χ (G). An optimal coloring is one that uses exactly χ(G) colors. The GCP is the optimization problem that consists of finding an optimal coloring for a given graph G. Since the GCP is NP complete, it is necessary to use heuristics methods to solve it.

Vertex coloring: When used without any qualification, a coloring of a graph is almost always a proper vertex coloring, namely a labeling of the graph's vertices with colors such that no two vertices sharing the same edge have the same color. Since a vertex with a loop could never be properly colored, it is understood that graphs in this context are loopless. The terminology of using colors for vertex labels goes back to map coloring. Labels like red and blue are only used when the number of is small, and normally it is understood that the labels are drawn from the integers {1, 2, 3,…}.

Coloring using at most k colors is called a (proper) k-coloring. The smallest number of colors needed to color a graph G is called its chromatic number, χ(G). A graph that can be assigned a (proper) k-coloring is k-colorable, and it is k-chromatic if its chromatic number is exactly k. A subset of vertices assigned to the same color is called a color class, every such class forms an independent set. Thus, a k-coloring is the same as a partition of the vertex set into k independent sets, and the terms k-partite and k-colorable have the same meaning.

3. Ant Colony Optimization

Ant Colony Optimization (ACO) is a metaheuristic approach for solving hard combinatorial optimization problems [1]. It's an evolutionary method inspired from the foraging behavior of real ants that enables them to find shortest paths between a food source and their nest. This method differs from other evolutionary methods such as, Genetic Algorithms (GA) and Scatter Search (SS) by the fact that at each stage of the solution construction, it takes into account information resulting from the preceding iterations and the desirability of each element that can be added to the current solution. In an ACO algorithm, a complete graph vertices are the solution components associated to the problem. It is called "construction graph". Moreover, in an ACO algorithm, simple agents called artificial ants communicate indirectly to find good solutions for the optimization problem. Informally, the behavior of ants in ACO algorithms can be summarized as follows: The ants of a colony concurrently and independently move through adjacent states of the problem on the construction graph, applying a stochastic local decision. While moving, ants incrementally build solutions to the optimization problem. Typically, good quality solutions emerge as the result of the collective interaction of the ants, which is obtained via indirect communication (pheromone trails). During the construction of the solution, ants evaluate the partial solution and deposit pheromone trails on components or connections it used (online update). This information will guide the future ants search. To move on the construction graph, ants will make decision based on pheromone trails (Γ) and an information specific to the problem (η). In many cases η, is the cost, or an estimate of the cost, these values are used by the ant's heuristic rule to make probabilistic decisions on how to move on the construction graph. The probabilities involved in this case are commonly called transition probabilities [3]. The goal in combinatory optimization is not to reproduce the biological model but to be inspired usefully. Thus, besides ant's activities, an ACO algorithm includes two additional procedures: pheromone trail evaporation and daemon actions. Pheromone evaporation is the process by means of

which the pheromone trail intensity on the components decreases over time. Formally, pheromone evaporation is a useful form of forgetting, favoring the exploration of new search space areas. Daemon actions can be used to implement centralized actions that cannot be performed by an ant. For example, the daemon can observe the path (solution) found by each ant of the colony and deposit extra pheromone (additional pheromone) on the components used by the ant that built the best solution. Pheromone updates performed by the daemon are called off-line pheromone updates.

Improvement Methods: Ant Colony System (ACS) for the GCP has two strategies of construction and improvement [4][5]. In construction strategy, the self-adaptation phase uses a problem specific constructive method to create new solutions. These two constructive methods take into account the different updates of pheromone trails and thus the heuristic information relative to the problem. Indeed, two decisions have to be taken at each step of a constructive method. The first is about the choice of the next vertex to color and the second is about the color to assign to the chosen vertex. In an ordinary constructive method, these decisions are made in a myopic way by completing the current partial solution at best. In addition, ACO algorithms perform best when hybrid with local search algorithms, (which is a particular form of daemon action). In the improvement strategy, at each iteration of the algorithm, the best solution found is improved using Tabu Search and these locally optimized solutions are used in pheromone updates.

Constructive methods are largely used in combinatory optimization. Their efficiency is justified by their short execution time and their facility of implementation. Constructive methods are also used to find an upper bound for the chromatic number. A constructive method adapted to the GCP run over the vertices set sequentially, and assigns to each vertex the smallest possible color (colors: 1, 2..., k) until obtaining a complete solution. In construction strategy of ACS, the self-adaptation phase uses a constructive method specific to the considered problem. This method will take into account the different updates of pheromone and the heuristic information of the problem. Two of the famous methods are, Recursive Largest First (RLF) and Degree of Saturation (DSATUR), which are considered among the best resolution methods for the GCP.

Update of Pheromone trails: Values of pheromone are associated to pairs of nonadjacent vertices having the same colour [5][6]. Formally, the value $\Gamma^k(v_i, v_j)$ corresponds to the trace left by a given ant "k" having assigned the same colour to the vertices v_i and v_j ($1 \leq i \neq j \leq n$). Therefore, at the end of a cycle of the algorithm, $\Gamma(v_i, v_j)$ is the value of pheromone associated to the couple (v_i, v_j) for all colourings (ants) which coloured v_i and v_j with the same colour. Let $\Delta\Gamma(v_i, v_j)$ denotes values of pheromone added to $\Gamma(v_i, v_j)$ by all the ants during a cycle. The values $\Gamma(v_i, v_j)$ are stored in a square matrix denoted "si" of order n and initialized like this:

$$\Gamma(vi, vk) = \begin{cases} 1 \text{ si } (vi, vk) \notin E \\ 0 \text{ si } (vi, vk) \in E \end{cases} \quad .. (1)$$

Where E is the Edge set that contains all the edges of the graph.

Stage by stage update: for each ant k, with pheromone decay ρ, the updated matrix is:

$\forall (v_i, v_j) \in S_k : \Gamma(vi, vj) = (1-\rho) * \Gamma(vi, vj) + \rho * \Gamma_0$ eqn. (2)

Evaporation is carried out according to this rule:

$\Gamma(vi, vj) = (1-\rho) * \Gamma(vi, vj)$ eqn.(3)

Where: S_k is the solution built by the ant "k"..

Heuristic Information: For the first strategy, heuristic information is defined according to the used constructive method. If an ant is implemented as RLF, heuristic information $\eta(v_i, v_j)$ relative to the choice of the vertex vj starting from the current vertex vi is defined in three possible ways:

$\eta(v_i, v_j) = deg_B(v_j)$. $\eta(v_i, v_j) = |A| - deg_A(vj)$. $\eta(v_i, v_j) = deg_{A \cup B}(v_j)$. B eqn.(4)

However, if an ant behaves like DSATUR, heuristic information is defined simply as the degree of saturation of the vertex in question.

$\eta(vi, vj) = DSAT(v_j)$ eqn. (5)

Transition Rule: ACS improves AS algorithm by giving more importance to information collected by previous ants with respect to exploration of the search space [7]. This is ACS improves AS algorithm by giving more importance to information collected by previous ants with respect to exploration of the search space. This is acheaved using two mechanisms. First, a strong elitist strategy is used to update pheromone trails eqn(2), second, ants choose the next vertex "v" to move to (colour) applying a pseudo- random proportional rule: With probability q0 they move to the vertex j for which the product between pheromone trail and heuristic information is maximum, that is, v= arg $\max\{(\Gamma_{ij})^{\alpha}(t) . (\eta_{ij})^{\beta}(t)\}$.

Candidate List: The candidate list is an intelligent strategy used mainly when the size of the instance (order of the graph) is very large. This technique consists in not considering the totality of the

neighborhood but rather a subset of this (for example, containing the best solutions), which can lead search towards promising areas. In addition, this allows accelerating the solutions construction. Inspiration of other techniques as in Tabu Search, where candidate's lists are used, could be useful for the development of this strategy efficiently in ACS.

Construction Strategy: In this strategy, each ant is initially put on a vertex of the construction graph randomly, or according to a well-defined criterion (the vertex having the maximum degree) [8][9]. Pheromone values on all connections are initialized to Γo. Each ant repeatedly construct a feasible solution while inserting into each stage a component, couple (vertex, color), in the current partial solution until obtaining a complete solution according to the constructive method implementing the ant (RLF or DSATUR). This construction is done by repeating the following stages:

(i) The next vertex to be coloured is chosen by observing the rule of transition (*). It is selected among the vertices of the candidate list if this one is considered.
(ii) The vertex chosen in the stage (i) is put in the ant Tabu list, this list is used to save the path (solution) built by the ant. Tabu list is also used to make sure that a vertex already coloured will not be coloured a second time, and consequently to guarantee the feasibility of the built solution.
(iii) This stage consists of decreasing the pheromone values associated to pairs of vertices having the same colour (the vertex that has been just added to the stable in construction); this to avoid fast convergence to the same solution. This update replaces the phase of evaporation in the algorithm. As long as the ant did not build a complete solution yet, it repeats the phases of construction. After all the ants established their solutions, the best one is saved and compared with that of the preceding iteration for a possible improvement of the objective function. After this, the daemon adds extra pheromone to only the best solution found in the preceding stage, and precisely to all connections that constitute it. If the same solution appears in several iterations of the algorithm (stagnation), then, the daemon decides to make evaporation. If the stopping criterion (a maximum number of iterations defined preliminary, or a maximum execution time) is not filled, the algorithm is started again. Give below the pseudo code of ACS algorithm, construction strategy, for the GCP.

Begin ACS1
Initialization ; (Pheromone and parameters)
While (stop criterion not satisfied) do
Position ants on starting vertices
Repeat
For (each ant) do
Choose a vertex to color by applying the transition rule (*);
Tabu list Update;
Update Online stage by stage of pheromone trails
End for
Until (each ant construct a solution)
Select the best solution ;
Pheromone Offline update of the best solution
Evaporation if necessary
End While
End ACS1

Improvement Strategy

The strategy of improvement can be summarized as follows: Pheromone values are initialized to a value Γ_0 on all connections. Each ant is assigned a starting vertex chosen randomly or by a well-defined criterion. Staring from an initial coloring (obtained either in a random way or by a constructive method), each ant tries to improve it by recoloring (change the color) some vertices in conflict. Improvement is carried out by choosing the next vertex to recolor as the vertex having a maximum number of violations among vertices of the candidate list. If such a vertex does not exist (current solution is feasible), a vertex is selected arbitrarily. If Iter_Max (a certain fixed number of iteration) is not reached yet, the process of improvement is repeated again. After this, Tabu Search makes an improvement to the best solution found in the preceding stage (online delayed update). The daemon adds extra pheromone (Offline update) to the best solution and precisely all connections that compose it according to the equation (4). If for a situation of stagnation, the same solution appears during several iterations of the algorithm, the daemon decides to make evaporation using the equation (5). If the stop criterion (a maximum number of iterations defined preliminary, or a maximum execution time) is not satisfied, the algorithm is started again. The pseudo code of ACS improvement strategy for the GCP [3].

Begin ACS2
Initialization ; (Pheromone et parameters)
While (stop criterion not satisfied) Do
Position ants on starting vertices;
For (each ant)
Generate an initial colouring for the graph G.
Repeat
Choose the next vertex to recolour among

vertices of the condidate list and according to the transition rule (*).
Change its colour so that conflicts are minimised.
Until (Iter_Max reached)
Online delayed Update of pheromone trails;
End For
Select the best solution;
Improvement of the best solution (TS);
Evaporation if necessary;
End While
End ACS2

The various constraints that need to be satisfied are:

- No resources (teacher or rooms) should be assigned to different events at the same time
- Events of the same semester must not be assigned at the same time frame.
- There are maximum number of times periods per day, which must not exceed.
- There should not be any type of conflicting allocation of resources between departments.
- There should not have any hard assumption about the resources and events.
- Must be easy for handling temporary modification.

Functional Requirements:

- The Administrator should be allowed to enter the details about courses, lecturers, exam, and room along with various criteria for allocating them.
- The system should have some means of alerting the Administrator if sufficient amount of input is not provided.
- The hard constraints like how much hours should be held for a course, must meet at any circumstances.
- The requirement of each event may vary and that must be taken into consideration. Such as there, needs a single lecturer for theory classes but for lab it may vary depending upon the requirement.
- The inter-department conflict of resources should be taken into consideration.
- System should be flexible enough to do temporary modification in case resources are not available. For example, in case some faculty has not come it must provide interfaces to allocate those classes to available faculty.

The software is developed as a client server application. Administrator handles the server. Administrator inputs departmental information, lecturer information, examination information and other information into the server, server stores these information in the database, using this information the conflict graph generator module generates a conflict graph, the generated conflict graph is then colored by graph coloring module, this module uses ant algorithm to color the graph. Interval Assignment module assigns corresponding time slots for abstract colors, the end of this process is a colored graph. When an End Viewer, student or lecturer requests for timetable report, server generates the report required by End Viewer by passing the colored graph to the Report Generation module, server then transfers report to the End Viewer. Server authenticates End Viewers, this process is depicted in system architecture of Fig 1.

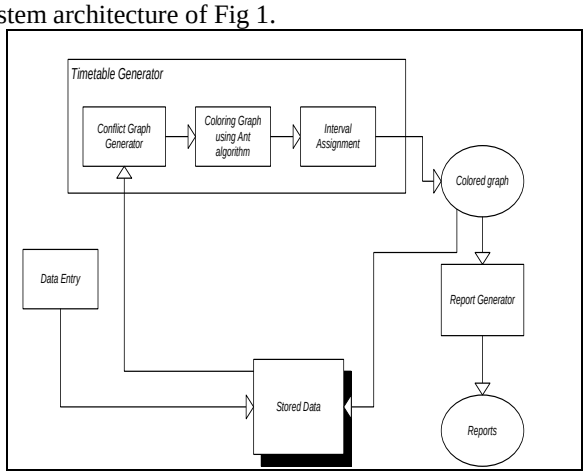

Fig 1 : System Architecture

4. Conclusion

"Timetable Scheduling" is scheduling and course planning software for any institution running under university. It provides an efficient scheduling of courses and events where complex combinations of resources must be assigned efficiently to timetables. The application is fast, flexible, user friendly and has extremely large capacity. It has a very rich set of efficient optimization and interactive approach to satisfy the user according to their varying needs.

This software works smoothly in face of scheduling difficulties with which institutions are often confronted when creating their timetables. The technical side of "Timetable Scheduling" has been implemented with the help of "Graph coloring using Ant Algorithm" in simple and self-contained manner as much possible. The application is extremely versatile and robust.

"Timetable Scheduling" software is an excellent choice for those institutions that find their current application outdated, restricted or inflexible to the changing demands of their increased scheduling requirements. The system makes use of very efficient data structure which can increase randomly it does not have the size limit of events.

5. Limitations

Following are the limitations of the project:

- More concern is given to efficient scheduling than the time required generating, but a trade-off between these two is always maintained.
- It does not handle more than one client at a time, but it can be easily extended by application of multi-thread.
- It works as a centralized system and does not support decentralized architecture.
- Temporary modification is limited to server, but it is always possible to enhance it for client module.

6. Further Enhancement

The application developed can be further enhanced to include following features:

- It can be extended to support the scheduling of university exams and inter-college test.
- The scheduling technique can be easily enhanced to support soft constraints, like preferred number of classes a lecturer wants to take.
- In case of decentralized administrator, this application can be further enhanced.
- It can be enhanced to support more than one client at a time
- Client could be enhanced to do temporary modification.

References

[1] *SangHyuck Ahn, SeungGwan Lee, TaeChoong Chung*, "**Modified Ant Colony System for Coloring Graphs**", ICICS-FCM 2003 15-18 December 2003. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1292787

[2] *Spyros Kazarlis, Vassilios Petridis and Pavlina Fragkou*, "**Solving University Timetabling Problems Using Advanced Genetic Algorithms**", International Journal, Vol. 4.2009, p no. 260-279.

[3] *Ehsan Salari and Kourosh Eshghi,"***An ACO Algorithm for the Graph Coloring Problem***",* International Journal Contemp. Math. Sciences, Vol. 3, 2008, no. 6, 293 – 304.

[4] *Malika Bessedik, Rafik Laib, Aissa Boulmerka et Habiba Drias*, "**Ant Colony System for Graph Coloring Problem**". J. Op. Res. Society, pp. 290-300 (2004)

[5] *D. Costa and Hertz A.* "**Ant can colour graphs**", J. Op. Res. Society, pp. 295-305 (1997)

[6] *E.K.Burke, D.G.Elliman, R.Weare,* "**A University Timetabling System based on Graph Colouring and Constraint Manipulation**".

[8] *Dorigo M, Maniezzo V, Colorni A*, "**The ant system:Optimization by a colony of cooperation agents**",IEEE Transaction of Systems, Man and Cybernetics-Part B, vol 26, no.2 pp. 29-41(1996).

[9] *Malika Bessedik, Rafik Laib*,"**Ant Colony System for Graph Coloring Problem**", International Conference on Computational Intelligence for Modeling, 2005.

First Author Biographies should be limited to one paragraph consisting of the following: sequentially ordered list of degrees, including years achieved; sequentially ordered places of employ concluding with current employment; association with any official journals or conferences; major professional and/or academic achievements, i.e., best paper awards, research grants, etc.; any publication information (number of papers and titles of books published); current research interests; association with any professional associations. Do not specify email address here.