Using Constraint Programming to Assign Students to First-year Seminars

Research Report

by

Xiang Yao (yaox@dickinson.edu)

Advisor: Timothy Wahls (wahlst@dickinson.edu)

Submitted in partial fulfillment of Honors Requirements for the

Computer Science Major

Dickinson College, 2013

November 28, 13
Abstract

Many colleges or universities in the United States have First-Year Seminars. For the incoming students, First-Year Seminar is designed to introduce students to the liberal art education and encourages students to participate the classroom discussion. Therefore, it is important to achieve a balance between student genders and origins. Also, in Dickinson College, each student can have 6 options indicating the rankings for their seminar choices. The college should satisfy students by choosing seminars they prefer. In this research, we use constraint programming to develop the program to solve the course assignment problem and also achieve a better balance between student groups as well as giving students their preferred course choices.

Introduction

Constraint satisfaction problem is the type of problems we can use constraints to limit the outcomes and generate preferable results. Constraint Programming is used for solving constraint satisfaction problems and combinatorial optimizations. It has wide commercial applications for many optimization problems, such as scheduling, graph problems and resource allocation. Moreover, it is also a popular method for many education institutions for university timetabling. Different methods within the field of operation research, constraint programming and artificial intelligence have been developed to solve this real-world scheduling problem.

I will research using constraint programming to assign incoming students to first-year seminars at Dickinson College. I will investigate the most efficient way to set constraint in constraint programming. There are more than six hundred students for each incoming class and each student can choose six seminars with different rankings. Every student can be either International or American. Also, each student is Male or Female. For my research, I will start from the most basic scenario of having each student choosing six seminars, and each seminar limited to around sixteen students. Then, I will add multiple objective functions to achieve higher ranking of the students’ class choices, better gender balance and the balance between international students and domestic students. We will compare our results with the operation research approach by Professor Dick Forrester in terms of the quality of solutions found, the time needed to find solutions and flexibility in adapting to changing requirements.

Background

The constraints are used to restrict the domains of the variables. This is why the constraint programming is faster than brute force search. There are many ways to set up constraint programming models. One way is to use finite domain constraints. A finite domain variable has its values restricted to a finite set of integers. We can specify
constraints to the finite domain variables and use different searching strategies to search possible solutions. Finite domain constraint is widely used for assignment problem and is one of the most popular approaches used by many constraints programming research.

Another way is to use finite set constraints. Finite set variables have a lower bound and an upper bound. The lower bound is the elements that are definitely in the set, while the upper bound is the elements that are possible in the set. The cardinality of the set can limit the number of elements in the set. Also, other set constraints such as “disjoint”, “union” and “the elements of” can be useful to model the scenario. (Francisco 12) Using set constraint, we can easily model the scenario of students assigned to certain seminars. These sets of students assigned to different seminars have to appear only in one seminar, add up to the total number of students and have the seminar sizes limited to around sixteen. Building onto this prototype, we will take students’ rankings, genders and nationalities into consideration to find the balanced group and prioritize on students’ class choices.

The branch and bound algorithm can be used to find optimal solutions with both finite set constraint and finite domain constraint. It is the searching algorithm we use throughout this paper. The branch and bound algorithm is covered in Marriott and Stuckey (1998) and is actively used in the artificial intelligence community, operations research and constraint programming. Using the branch and bound algorithm, we expect to find a good solution quickly. The algorithm iteratively finds better and better solutions. First, it finds a solution $\Theta_{\text{best}}$. Then, it adds a constraint to exclude additional solutions, making $\Theta'$ no better than $\Theta$. Next, it solves the new constraints with the one added and updates the previous solutions to find a better solution. Finally, this process keeps going until the program cannot find any feasible solutions. Since this is a NP-complete problem, the upper bound of the algorithm is still exponential. However, depending on the pruning by adding additional constraints in the process, we may get much better performance than exponential. Since we have many variables in our models, we are expecting to use this algorithm to reduce the search tree in the searching process and come to a good solution quickly.

**Literature Review**

For constraint programming, finite domain constraint has been a popular method to model constraint satisfaction problems. Marte (2002) has thoroughly discussed constraint programming in his PhD Thesis and developed a basic finite domain constraint model to solve school timetabling problems. He used the German school timetable problem as an example and came up with a model to schedule the meetings, curriculums and gymnasiums for both students and teachers. Each student and teacher fulfills a certain
requirement for the classes and meetings. Also, each time slot and class has constraints on how many students and teachers can be assigned. Different from our problem, students cannot have options to choose classes. They are confined to a certain type. For example, a 9th grade student who has English as his or her second language can be one type. Then, a set of core classes, gymnasiums and meetings need to be assigned into different time blocks and also fit the students’ schedule.

The school-timetabling problem is diverse in its nature with different requirements in different districts and systems. Large parts of his work have been to explore different possible constraints in the school timetabling system and apply different constraints for different situations. For example, he applied global cardinality constraints to limit the frequencies with which each class is assigned to students. Global cardinality constraint is the built-in constraint in the constraint-programming library that applies to multiple finite domain variables. In one hand, he emphasizes coming up with the same constraint to deal with different problems in order to expand the flexibility of the program. In another hand, he fully utilized the built in global constraint of the problems to improve the running time.

Since finite domain will be used in this research, we will also try to adopt as many global constraints as possible to apply to the problem. Different from the school timetabling problem, which has diverse constraints with different requirements, our constraints will likely be much more uniform and simple. The special characteristic of the uniformity actually paves the way for a more focused constraint solver then the finite domain constraint. This partially explains why we have adopted finite set constraints as our constraint solver.

Delgado and Perez (2005) take another approach to using constraint programming to build an application. The problem has 1600 events and includes different constraints in from different departments in at Universidad Javerana. Each department has its own requirement for the time, class size and so on. Therefore, it is possible the constraints set by different department may conflict with each other. Finally, they solve the timetabling problem to find the first solution with 1600 events in 46 seconds.

Instead of focusing on modeling the whole scenario, this research focuses on the process of modifying the constraints to deal with inconsistencies, called an over-constraints situation. Over-constraints situations are common in the constraint programming model and will result in an infeasible solution. They applied an incremental approach to increase the size of the constraints for the model and keep checking for inconsistency. If an inconsistency is detected, the checking algorithm will return false and enable the user to use a graphical interface to modify the model. This semi-automated constraint solver makes it possible to find a feasible solution to the timetabling problem.
This paper has proved the concept of human interaction for the constraint programming
decision-searching process. If a certain constraint is too strict, a user can physically
loosen it in order to generate a better global solution. One down side of this is that this
approach will require a constraint programming application instead of a script to solve
the problem. Also, it will potentially take longer to manually decide which constraint
should be loosened and which should stay.

Different from the university timetabling problems, the course assignment problem
related to class choices has been approached solely using operation research methods.
Specifically, the course assignment problem involves assigning students to classes based
on their top choices (For our research, it is top 6 choice.). In 2006, Willoughby and Zappe
(2006) consider the assignment of students to first year seminars at the liberal art college
Bucknell, which is really similar to the problem size of Dickinson College. They are
minimizing the weighted average for the ranking of the students’ choices and applying a
linear programming approach to solve the problem efficiently.

They can solve this large set course assignment problem quickly, since they are using
linear programming for their model. Linear programming has a great performance
advantage compared to other Operation Research approaches. With unimodularity
assumption for the input data, it avoids setting integer constraints. Specifically, for the
unimodularity assumption in linear programming, if all the input are integer, all the output
will be integer even if we do not have integer constraint for the program. Fully using this
assumption, a linear model will be the best fit. They are able to solve the problem in 30
seconds.

At the end of their research, they relax the constraint on the number of students assigned
to each seminar. With that relaxation, they come up with a lower total weighted average
and a better ranking for the student options. There is a tradeoff between the number of
students assigned to each class and the optimization result. For our research, we can also
consider relaxing a certain constraint, for example seminar size, to optimize other
objectives.

In 2013, Forrester, Hutson, and To (2013) solve the same course assignment problem for
Dickinson College. Instead of only considering the ranking of student choices, one
concentration of this paper is to balance the students’ gender and nationality. To include
the balance between student gender and origin, the paper adopts a deviation-based
objective approach and utilizes quadratic integer programming to solve the problem.
They used square differences and calculated a Gender Penalty as the sum of the square
value of males minus females in each seminar. Also, they calculated a Citizen Penalty by
calculating the sum of the square value of international students minus American students
in each seminar. They applied standard mixed-integer convex quadratic solvers, Xpress,
to the problem and received a good quality solution within two minutes. By using a
deviation based objective approach, they avoid setting a fixed constraint for the number of certain students with a certain gender and nationality in a seminar, therefore avoiding the over-constraint problem as previously discussed.

After this research, they submitted another related paper to the *Journal of Operational Research Society*. They expanded the model to have multi-objective convex quadratic integer program to take the ranking into consideration. The deviation for different objectives is calculated and then added up with weight to get the total deviation.

They came up with a way to give priority for the ranking of classes with different weights. For example, in one way they can always weight the earlier choices more than the later choices: the first option is prioritized over the second option, and the second option is prioritized over the third option, so on and so forth. In another way, they can weight more on the top three choices than the other bottom three choices. Also, they can divide the choices to prioritize over top two, middle two and finally the last two choices. The difference is that the first one always prioritizes the students’ highest choices, and the second one will consider the top three choices the same. So the second assumption is that as long as we are giving students their top three choices, they are satisfied. Also, it is more likely to give more students their top three choices when we consider the top three the same priority group, since we are essentially relaxing the students’ option to first group (top three) and second group (bottom three). In terms of the running time, using the same constraint solver, a good quality solution can be found within two minutes.

Another example from Drass and Ovchinnikov (2006) to achieve the student balance is assigning students to balanced groups at the MBA classes in Toronto. The business school is trying to give better discussion and project groups for MBA students. They are trying to have each project and discussion group maintain certain characteristics of students. However, the difference between assigning MBA students to groups and assigning students to seminar is that business students do not choose the discussion or project groups in which they will participate. They are basically able to assign to all different groups. Therefore, it is easier to find a balance group in MBA students. Additionally, great care is taken in the admission process to accept enough students with certain characteristics.

Different from the deviation-based approach adopted by Forrester, Hutson, and To (2013), Drass and Ovchinnikov (2006) use a different method by setting a lower bound and an upper bound in a certain criterion for a certain group. For example, they want every group to have from 4 to 6 international students. They were always able to find a feasible solution in their three years’ dataset and can solve the problem using integer programming within a couple of minutes. However, this approach might not work for us, since some seminars are so unpopular for a certain characteristic of students. For instance, it can be really rare for international students to choose literature. For MBA students
problem, the paper proposes a solution to loosen the bounds of the constraints to a wider range if they can not find a feasible solution. We can potentially simplify our course assignment problem and improve running time if we set hard constraints on the number of certain characteristics in one group, since we do not need to optimize quadratic terms like the deviation approach by Forrester, Hutson, and To (2013).

One advantage of using the Operation Research technique is the running time of the program. Especially for linear programming, constraints with thousands of variables can be solved within seconds. In comparison, finite domain constraint is much slower. Finite domain constraint performs well when there are limited variables. When the number of variables is large, it can increase the size of the search tree and will potentially cause the program to perform in unacceptable speed. This has hindered the researchers in applying constraint programming to solve this type of course assignment problem.

One of the guidelines in Marriott and Stuckey (1998) for using constraint programming is to model the problems with fewer variables. One example that has been used is to compare two different modeling approaches to the same problem of assigning workers to products. Four workers are assigned to four products to increase the total profit of the factory. The sum of the worker/profit pair variables should be one. The first approach is to use a typical operation research approach to have binary values for each worker and product pair and limit all the workers and product pairs to a size of one. The other approach is to use finite domain constraint to set each worker as a finite domain variable with all products as its domain set and apply constraints to these finite domain variables to ensure that they have all have unique values between each other. This will automatically limit the size of workers’ choices to one different product.

The second model is more efficient for constraint programming since it has less variables and less constraints to solve. Applying the second model to our research model, we can have each student as a finite domain variable with 6 options in its domain. However, we would need at least 600 more variables to model the students, not to mention considering the students’ origin and gender. This might expose us to bad performance for the constraint programming compared to the operation research method. Apparently, the finite domain constraint as the model of the basic course assignment problem is not preferred.

Another way to solve combinatorial optimization problems is using finite set constraint. Finite set constraint as well as the Cardinal constraint solver has been explained in Francisco (2012). Finite set constraint is a variable limited to a set of values. It can have a lower bound and upper bound. The lower bound is the elements that are definitely in the set while the upper bound is the elements that are possibly in the set. Additionally, cardinality of the set is a major constraint for finite set variables since this can indicate
the size of a finite set and is particularly useful for our model, since we are trying to limit the size of the students in each seminar to 16.

Set constraint is a natural constraint to apply to set variables to solve set covering, set partitioning and set packing problems. The course assignment problem is a typical set packing problem and can be easily model and interpreted using finite set constraint. For example, golf tournament assignment problem has been explained in Müller, T. (2008). A golf tournament has 32 players and needs four players for each game. Each player cannot play with another player twice. We can set each tournament as a finite set variable and limit cardinality to a size of four. Then, we need to apply additional set constraints to fulfill that each player cannot play with another player twice. Finite set constraints can model the course assignment problems with much fewer variables, since we only need to apply constraints to each set. Specifically, for our example, we can have around 50 sets for different seminars and only apply constraints to those sets. The course assignment problem can potentially be solved much faster.

However, the existing constraint-programming solver still cannot solve the problem efficiently. This also explains why researchers have avoided applying finite set constraint and constraint programming methods as a whole to solve large size course assignment problems. Using the Conjunto finite set constraint solver, the basic course assignment problem cannot be solved after eight hours.

The Cardinal in finite set constraint library made public in ECLiPSe has been major breakthrough for finite set constraint solver. (ECLiPSe is the open source constraint programming application and library we use for our assignment problem.) It has speed up to two orders of magnitude comparing to Conjunto, the previous constraint solver for ECLiPSe. Also, it actively uses the information about the cardinality of the sets. It uses different techniques to apply cardinality for the pruning process and reduces the time needed to solve different set constraints for the problem. This is a simple example to applying cardinality. Suppose we have set A with lower bound of [2,3] and upper bound of [1,2,3,4,5]. If we have a cardinality finite domain variable to be [1:5], the cardinality constraint will automatically change the domain to [2:5] since our lower bound has two variables. As the paper argued, the cardinality has to be included into the bound of the set. One simple example is set equality. When two set are set to be equal, this will also pass to its cardinality and set them to be equal. Another more complicated example is set union. When the first set has a cardinality of C1, and the second set has a cardinality of C2, the union of these two sets will have cardinality not greater than C1+C2. The Cardinal is used in our research at the beginning. However, since it has trouble working with some other constraint libraries, we finally adopt ic_hybrid finite set solver, a solver similar to Cardinal.
Experiment & Preliminary Results

Summary

We applied a different model to the seminar problem to try to find an optimal solution with reasonable running time as well as a good quality of the solution. The model needed to be simplified enough to incorporate the students’ options. Also, we needed to pay great attention to the running time of the program since we are trying to find a feasible solution in a reasonable amount of time.

From the experiments in the following, we observe that the biggest problem of using constraint programming to assign students to different seminars is the running time. In most cases, we encounter the situation that the problem did not have any output or only obvious non-optimal output after running for a week. This appears meaningless for this application. It will take too much time, and we do not know what time the script will finish. Therefore, it is critical to find a model that can search for a solution quickly. As I mentioned in the literature review, there are different factors that can decide the running time of the model. First, we are choosing between finite set constraint and finite domain constraint. This comes down to how many variables we will include in our models. Secondly, for the finite set constraint, different constraint solvers apply different pruning rules for the finite set, therefore having different running times for the same problem. For example, Cardinal has much better performance than Conjunto since it applies different pruning rules. Thirdly, the order in which we consider the sets when doing searching has a great impact on performance. Therefore, we need to use a different searching strategy to optimize our output. In terms of the model, we also need to decide whether we are using fixed constraint to achieve the balance between different groups or a quadratic objective function to find more optimal balance within the group. This will also have a great impact on our model’s running time and the quality of the output.

After the experiment, we decided to use finite set constraint with ic_hybrid finite set solver. We have successfully achieved the balance between the international students and domestic student with reasonable running time. Currently, we are taking the ranking of students’ course options into consideration and putting it as part of the objective function together with the objective function for gender and nationality. We have successfully built up the model and continuous search for the optimal solution for ranking, gender and nationality.

Detailed Experiment

For finite set constraint, first we adopt the Conjunto constraint solver and use about 50 finite set variables for each seminar to model all the students who have chosen a certain
We put all students’ choices into around 50 different sets and used cardinality constraint to limit the size of each set to around 16. This requires that each seminar has around 16 students. Also, we need to use disjoint constraint for all seminars to make sure that no seminars are chosen by two students. Finally, we add up the students from each seminar and make sure the sum equals the total number of students in that incoming class. This constraint makes sure that all seminars are chosen by some students, and no students are left out. We did a prototype with a small data set of 10 students and 3 seminars and successfully put different students into different classes. However, when we apply the big data set for the incoming class in 2011 with 47 seminars and 667 students the constraint solver cannot find a single solution after running for 1 day. Since it is slow to find one solution for the basic model, we do not think it will be possible to build a more complicated model with international students and gender balance and search for the optimal solution among the solution sets. Therefore, we are considering an alternative approach.

Cardinal is an alternative option for Conjunto. It has much better running time as I have mentioned in the literature review, since it has better pruning for various finite set constraints. The basic model is exactly the same as the one used in Conjunto. The only difference is that we change the constraint solver. With this change, we are able to find one solution within 2 seconds.

Then, we need to decide how to search the optimal solution to balance between international students and domestic students. We tried various model and finally decide to split the original sets into different subsets for each student group: male, female, international and domestic. We added 4 additional finite set variables for each seminar, having close to 200 finite set variables for all the seminars. For the four finite set variables for each seminar, we set the upper bound of the set as the students’ choices from their featured groups. The unions of these two pairs of finite set variable (gender pair and origin pair) have to be equal to the set of students assigned to a certain seminar.

Using this model, we can easily apply the constraint on the size on each subset and calculate the gender and origin difference in each seminar. Using this model we can easily search for one solution quickly within 2 seconds. Next, we need to calculate the objective function using branch and bound search to find more optimal solutions within those solution sets. We are using the same objective function from Forrester, Hutson, and To (2013)’s research by calculating the gender and nationality penalty coefficient for each seminar. Specifically, as I mentioned before, we are calculating a Gender Penalty as the sum of the square value of males minus females in each seminar and calculating a Citizen Penalty by calculating the sum of the square value of international students minus American students in each seminar.
However, we encounter a problem when working with the finite domain library in EcLiPse with Cardinal, since it does not support optimization. Finally, we have to find another finite set constraint solver to replace Cardinal. We found ic_hybrid finite set constraint solver, which is similar to Cardinal, but it works with ic library seamlessly. Ic library is the finite domain library used to create and apply constraints on finite domain variables in EcLiPse. Using the ic_hybrid finite set constraint solver, we can quickly search for one feasible solution. After applying the objective function for branch and bound searching, it continuously looks for a better solution and the first couple of optimal solutions appear reasonably balanced after running for 2 minutes.

Finally, we expanded upon our previous model to take the ranking into consideration. For ranking, we added 6 additional finite set variables for each seminar, having totally close to 500 finite set variables for all the seminars. The 6 variables for each seminar represent all the students who have chosen a particular seminar as their first through sixth choices. Basically, we are splitting the one set for each seminar into six sets using the ranking of students’ choices. For the six finite set variables for each seminar, we set the upper bound of the set as the students’ choices. The unions of these six variables for each seminar have to be equal to that seminar set.

Using this model, we can add an additional “ranking penalty” into the original objective function easily. We simply add up all the students who have their certain choices from all seminars together. Basically, we are calculating the total number of students assigned to their first choice, second choice, third choice and so on. Finally, we assigned a different coefficient for every different choice of group. For example, the ranking penalty from set one to set six can be calculated as such, S1 + 4 * S2 + 9 * S3 + 16 * S4 + 25 * S5 + 36 * S6. (Si indicate the number of students in seminar i) Later, we can adjust these coefficients according to Forrester, Hutson, and To (2013)’s research. Different coefficients indicate different emphasis on the ranking group and will come to a different solution set. For instance, if the first three groups have the same penalty, it is more likely to find the solutions with more students having first, second and third options as their choices.

**Future Work**

Currently, we have successfully applied finite set variables to achieve the balance between student gender and origin and also optimize on the students’ class choices. From Forrester’s later research on finding the optimal ranking for students, we will use the objective function to decide how we can combine three objective functions together and decide on the coefficient for each penalty variable. We will use their calculated coefficient to set our optimal weight between different objective functions and try to compare different performance in terms of running time and quality of the result.
Specifically, we can set each variable with different weights, set the first three ranking groups into equal weight (The rest into much higher weight), and set the first two ranking groups into equal weight (The rest into much higher weight).

Additionally, we can also take the students’ nationalities into consideration. For example, we do not want too many students from the same country in the same first year seminar. It is likely to happen since around fifty percent of all international students come from China. By applying this rule, we can further optimize the balance between different student origins. One difficulty we might encounter is the running time. By achieving the balance between different nations, we will need additional variables to set the finite set domain and the searching process will be longer. We can start from adding a certain major country first, ex. China, and then decide how many different nationalities we are able to include.

Also, we decide to reconsider finite domain constraint approach. As I mentioned in the result section, from our previous experiments, the finite domain constraint do not perform well for the running time. We find out that this is not due to the model itself. We apply some fixes to the original model by moving part of the data processing code out of the optimization to reduce the backtracking. This improves running time since the program does not have to do extra work in the optimization. After this fix, we improved the running time of the program. Currently, we are only optimizing the ranking for the objective function. Using naïve search strategy, we can find two solutions within minutes. For the future, we will try different searching strategies to optimize the running time. First, we will try naïve-based searching strategy. Then, we will consider using first-fail searching strategy. For the naïve distribution strategy, search is done in the order that variables are passed to the search strategy. First-fail searching strategy is to search the variable with the smallest domain first. It is more likely that we will have a feasible solution faster. We are expecting improvement on the running time after making these new changes.
Reference


Forrester, R and Hutson, K Balancing Student and Faculty Preferences in the Assignment of First-Year Seminars, submitted to International Journal of Information Technology and Decision Making

