Project Work
Combinatorial Decision Making and Optimization
2021/2022

Topic: Modelling & solving a combinatorial optimization problem
Authors: Zeynep Kiziltan and Roberto Amadini
Date: May 13, 2022

This document describes the project work of the Combinatorial Decision Making and Optimization course for the academic year 2021/2022, which is about modelling and solving a combinatorial optimization problem. The students are free to choose one of the proposed problems, as well as to propose another problem taken from the literature provided that they get our approval before start working on it. The project work involves approaching the chosen problem using (i) Constraint Programming (CP), (ii) propositional SATisfiability (SAT) and/or its extension to Satisfiability Modulo Theories (SMT), and (iii) Linear Programming (LP). The students can form a group of 3 members, and it suffices for point (ii) to develop a SAT or an SMT solution. If they do both, they obtain bonus points. They can also form a group of 4 members, but in that case they are required to develop both SAT and SMT solutions.

ATTENTION While the students can start working using CP and SAT as of the end of the first module of the course, they should wait for the SMT and LP until the end of the second module.

1 Problem 1: VLSI Design

1.1 Problem description

VLSI (Very Large Scale Integration) refers to the trend of integrating circuits into silicon chips. A typical example is the smartphone. The modern trend of shrinking transistor sizes, allowing engineers to fit more and more transistors into the same area of silicon, has pushed the integration of more and more functions of cellphone circuitry into a single silicon die (i.e. plate). This enabled the modern cellphone to mature into a powerful tool that shrank from the size of a large brick-sized unit to a device small enough to comfortably carry in a pocket or purse, with a video camera, touchscreen, and other advanced features.

As the combinatorial decision and optimization expert, the student is assigned to design the VLSI of the circuits defining their electrical device:

given a fixed-width plate and a list of rectangular circuits, decide how to place them on the plate so that the length of the final device is minimized (improving its portability). Consider two variants of the problem. In the first, each circuit must be placed in a fixed orientation with respect to the others. This means that, an $n \times m$ circuit cannot be positioned as an $m \times n$ circuit in the silicon plate. In the second case, the rotation is allowed, which means that an $n \times m$ circuit can be positioned either as it is or as $m \times n$.

1.2 Format of the Instances

This section describes the format in which the VLSI instances are written, as well as the expected format of the corresponding solutions.

Instance Format An instance of VLSI is a text file consisting of lines of integer values. The first line gives w, which is the width of the silicon plate. The following line gives n, which is the number of necessary circuits to place inside the plate. Then n lines follow, each with x_i and y_i, representing the horizontal and vertical dimensions of the i-th circuit. For example, a file with the following lines:

```
9
5
3 3
2 4
2 8
3 9
4 12
```

describes an instance in which the silicon plate has the width 9, and we need to place 5 circuits, with the dimensions 3×3, 2×4, 2×8, 3×9, and 4×12. Figure 1 shows the graphical representation of the instance.

Solution Format Where to place a circuit i can be described by the position of i in the silicon plate. The solution should indicate the length of the plate l, as well as the position of each i by its $\hat{x}_i$ and $\hat{y}_i$, which are the coordinates of the left-bottom corner i. This could be done by for instance adding l next to w, and adding $\hat{x}_i$ and $\hat{y}_i$ next to x_i and y_i in the instance file. To exemplify, the solution of the instance depicted in Figure 1 could look like:

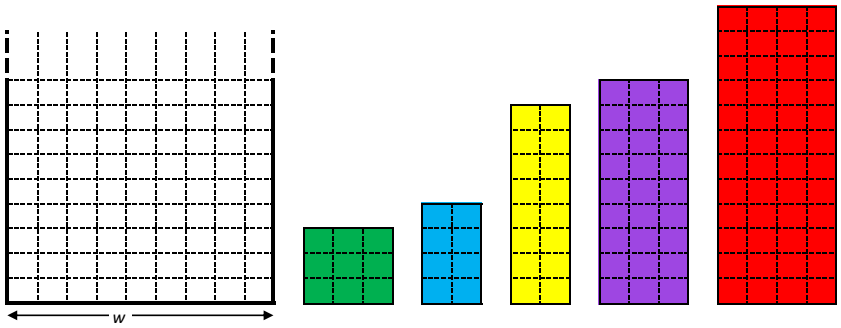

Figure 1: Graphical representation of a VLSI design instance.

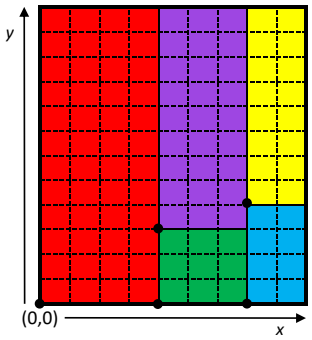

Figure 2: Graphical representation of a VLSI design solution.

```
9 12
5
3 3 4 0
2 4 7 0
2 8 7 4
3 9 4 3
4 12 0 0
```

which says for instance that the left-bottom corner of the 3×3 circuit is at $(4, 0)$. The solution can be represented graphically as in Figure 2.

For correctness check, the students are advised to visualize the solutions as in Figure 2. This can be done manually by hand, or automatically by a program which takes as input the solution of the problem.

2 Problem 2: Multiple Couriers Planning

2.1 Problem description

It has become very common nowadays, especially after the outbreak of the COVID-19 pandemic, to send or receive items online (food, goods, clothes, documents, ...). Hence, it is more and more important for companies to efficiently schedule the dispatching of items through different couriers, in order to minimize the transportation costs.

As the combinatorial decision and optimization expert, the student is assigned to solve the Multiple Couriers Planning (MCP) problem which is defined as follows. We have m couriers that must distribute $n \geq m$ items at different customer locations. Each courier i has a maximum *load size* l_i. Each item j has a *distribution point* d_j and a *size* s_j (which can represent for instance a weight or a volume). The goal of MCP is to decide for each courier the items to be distributed and plan a *tour* (i.e. a sequence of location points to visit) to perform the necessary distribution tasks. Each courier tour must start and end at a given origin point o. Moreover, the maximum load l_i of the courier i should be respected when items are assigned to it. The objective is to minimize the total tour distance.

2.2 Format of the Instances

Instance Format An instance of MCP is a text file of the form:
m
n
$l_1\ l_2\ \ldots\ l_m$
$s_1\ s_2\ \ldots\ s_n$
$dx_1\ dx_2\ \ldots\ dx_n\ ox$
$dy_1\ dy_2\ \ldots\ dy_n\ oy$

where m, n, l_i, s_j have the meaning explained in Sect. 2.1, (dx_i, dy_i) are the coordinates of distribution points, and (ox, oy) are the coordinates of the origin point. A square *distance matrix* D of order $n + 1$ can be built from these coordinates where $D_{i,j}$ is the distance between point i and point j. The student can choose their favourite distance for D (e.g., Manhattan or rounded Euclidean distance) and place the rows/columns corresponding to the d_j's and o points as they prefer. For example, from the following input file:

```
3
7
15 10 7
3 2 6 8 5 4 4
1 2 2 4 5 5 6 3
3 1 5 0 2 5 4 3
```

we can build a symmetric distance matrix D using Manhattan distance such that the points on rows and columns follow the order $d_1, \ldots, d_n, o$:

$$\begin{array}{cccccccc}
0 & 3 & 3 & 6 & 5 & 6 & 6 & 2 \\
3 & 0 & 4 & 3 & 4 & 7 & 7 & 3 \\
3 & 4 & 0 & 7 & 6 & 3 & 5 & 3 \\
6 & 3 & 7 & 0 & 3 & 6 & 6 & 4 \\
5 & 4 & 6 & 3 & 0 & 3 & 3 & 3 \\
6 & 7 & 3 & 6 & 3 & 0 & 2 & 4 \\
6 & 7 & 5 & 6 & 3 & 2 & 0 & 4 \\
2 & 3 & 3 & 4 & 3 & 4 & 4 & 0
\end{array}$$

In this way, for instance, $D_{2,n+1} = D_{2,8} = 3$ is the distance between d_2 to o, while $D_{4,7} = 6$ is the distance between d_4 to d_7.

Note that the problem can be formulated in different ways. One way is to consider an equivalent problem where we find a single tour with n distribution points and $2m$ "dummy" origin points having null distance to each other.

Solution Format There is no standard way of representing the solution, but surely the tour of each courier (showing the assigned items) and the total distance must be properly visualised. For example, a solution of the previous example could look like:

```
Courier 1: o -> d2 -> d4 -> d5 -> o
Courier 2: o -> d3 -> d6 -> o
Courier 3: o -> d1 -> d7 -> o
Total distance: 34
```

For correctness check, the students are advised to visualise the solutions as in Figure 3. This can be done manually by hand, or automatically by a program which takes as input the solution of the problem.

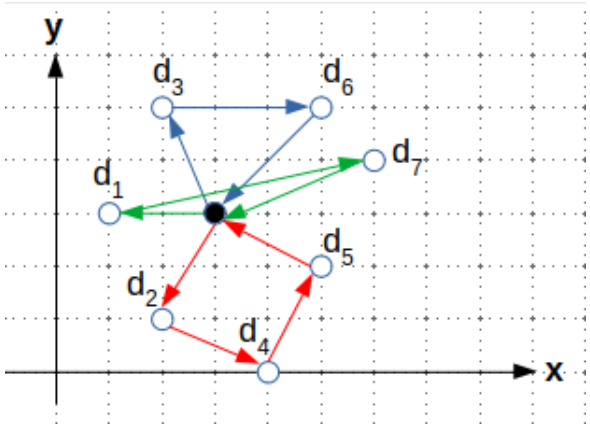

Figure 3: Graphical representation of an MCP solution. The black node is the origin point, the white nodes are the distribution points and the arrows denote the tour of each courier.

3 Project Work

The purpose of this project is to model and solve the chosen problem using (i) Constraint Programming (CP), (ii) propositional SATisfiability (SAT) and/or its extension to Satisfiability Modulo Theories (SMT), and (iii) Linear Programming (LP). The students will decide themselves the problem constraints around the problem description, and build the models/encodings accordingly. A suite of problem instances are provided in the specified format for experimental purposes. No assumptions should be made based on these instances. The approach should work for any random instance.

ATTENTION While the students can start working on the CP and the SAT solutions as of the end of the first module of the course, they should wait for the SMT and LP until the end of the second module when the details on how to apply them become clear.

Use of CP and SAT The students should use the MiniZinc and Z3 solvers. Though a trivial model/encoding is a good starting point, the students should come up with the best model/encoding to tackle the relatively more difficult instances of the problem in the most efficient way. It is advised to proceed as follows:

1. Start with the problem parameters, decision variables, the main problem constraints and the objective function.
2. Constrain your objective function with tight lower and upper bounds.
3. Consider different SAT encodings so as to reduce the encoding size.
4. Investigate if there are any implied constraints that you can benefit from.
5. Use global constraints to impose the main problem constraints and the implied constraints in your CP model.
6. Observe if any symmetry can be exploited to improve the CP model and the SAT encoding by adding symmetry breaking constraints. Note that when you enforce multiple symmetry breaking constraints, they should not conflict with each other (i.e., there must be at least one solution satisfying all the symmetry breaking constraints).
7. Investigate the best way to search for solutions in CP which does not conflict with the symmetry breaking constraints. The symmetry breaking constraints should not conflict with the SAT search either.

4 Project Report

The students should describe clearly in a report how they tackled each of the points above as well as any remarks or comments they consider appropriate. While a MiniZinc notation could be acceptable to describe a CP model and search annotations, it is required in general to use propositional logic, first-order logic and appropriate mathematical notations. **Do not copy and paste code!**

The report should also describe the experimental results obtained by the provided instances. The students should show experimental results of their best combination of model and search in CP, and of their best encoding in the others. As a reference, solving processes that exceed a time limit of 5 minutes (300 secs) should be aborted. Depending on the solving approach, on the machine, etc., some of the instances may be too difficult to solve within the time limit. For this reason, it is not strictly necessary to succeed in solving all instances to pass the project. However, the students are encouraged to solve as many instances as possible.

5 Submission

The project will be submitted via Virtuale. When working as a group, it suffices that only one group member submits the project, provided that all the members' names and e-mail addresses are indicated in the comments of the submission and in the report. For submission, create a separate folder for each approach (named appropriately such as CP, SAT, etc) and add the following files in each folder:

- a project report in PDF.
- a directory `out` with the output files of those instances that could be solved successfully. Name an output files in a way to make a correspondence with the related input file. For instance, the output file corresponding to an instance file $ins-i$.txt could be named as $out-i$.txt.
- a directory `src` with all the source code used to carry out the project, together with a `README` file with basic instructions for execution (so that results can be reproduced).

Upload a `tgz` or `zip` compressed archive after naming it with the surnames of the group members.

6 Evaluation, Examination and Grading

Project evaluation will be based on the students' ability to model and solve a combinatorial optimization problem using the approaches and the tools discussed in the course. While it suffices to produce either a SAT or an SMT encoding for groups of max 3 students, bonus points will be given to those who produce both. Groups of 4 students are required to produce both.

As part of the project evaluation, group members will be invited to Teams for an oral discussion of the project (submitted at an earlier date) with one of the course teachers. No additional presentation is required. Project submission and discussion dates will be announced in due course. Expect to have a session in July'22, September'22, December'22 and February'23. Those who have not yet discussed by February'23 will be able to submit later with the risk of having to wait until the summer of 2023 for discussion.

ATTENTION if you have any specific deadline by which you need to register the final mark (for instance due to a scholarship, graduation etc), please contact us well in advance! We cannot

guarantee to meet the student's personal deadlines which have not been communicated earlier.

A successful project evaluation will result in a mark between 18 and 30L. An insufficient project will not obtain a mark. In that case, the group members will be asked to rework on the project and resubmit.

7 Academic Integrity

Intellectual development requires honesty, responsibility, and doing your own work. Plagiarism is the use of someone else's work, words, or ideas as if they were your own. Any idea taken from any person or resource should be cited appropriately. Plagiarised work will not be evaluated.