A tool-supported framework for work planning on construction sites based on constraint programming

Azahara Camacho[1], Pablo C. Cañizares[2], Sonia Estévez[2] and Manuel Núñez[2]
[1]Department of Aerospace & Defense
Carbures Engineering
Cádiz, Spain
[2]Dept. Sistemas Informáticos y Computación
Facultad de Informática
Universidad Complutense de Madrid, Spain
[1]azahara.camacho@carbures.com, [2]{pablocc,soesteve,manuelnu}@ucm.es

Abstract

This paper presents a framework taking advantage of the capabilities of current constraint solvers to plan the work on construction sites. It combines different constraints under a common framework to facilitate the definition of temporal relations between different tasks and provides a user-friendly web interface, which facilitates the planning of construction sites. Even though the framework uses complex mathematical models, the users do not need to know the underlying theoretical framework. Another important feature of the framework is, in contrast to usual *static* planning, that solutions can be dynamically adapted to take into account delays happening during the actual construction process. In order to show the applicability of the methodology, the papers shows how a real construction project can be planned by using the framework.

Keywords: Construction sites planning, Constraint programming, Formal specification and analysis

1. Introduction

In order to construct any structure, even the smallest ones, it is usual to have a *model* of the building that it is going to be developed.[1] Therefore, planning is an important step in construction and it is fundamental in order to ensure the successful management of construction sites. In fact, the awareness of the importance of *formalizing* the planning of different tasks in construction sites is increasing [2]. In this line, it is very important to plan how the different teams and machinery will be deployed in the construction site. It is necessary to take into account both the availability of these resources and the time constraints associated with the different tasks. There are tasks that should be sequentially performed (e.g. terrain movement should be completed before the construction of a concrete surface bed is started) while others can be performed in parallel and a correct planning will minimize the amount of idle resources. The temporal planning of a construction site is reflected in a *graph*. In this line, Gantt diagrams are specially popular. These diagrams show, in a very intuitive way, causal relations between different tasks and phases of the site. Unfortunately, these graphs are usually made by hand, based on the experience of the architect(s) planning the site. Even though experienced planners were in charge of this phase, it would be more accurate, and reduce the number of errors, to automatically compute solutions based on a formal design [3]. The framework strongly facilitates the generation of *static* plans but it is possible to go one step forward. In addition, the use of constraint programming [4] strongly simplifies the management of unexpected situations after the construction has started in the site. Specifically, it is not necessary to manually recompute the impact of delays on the tasks that depend on its conclusion.

The main goal of the framework is to provide good solutions, while minimizing human intervention, in the computation of optimal temporal assignment of resources to tasks. As a first step, it facilitates the work of the planner by providing tools where (s)he can easily describe the temporal constraints between activities. Afterwards, the framework will automatically compute solutions, taking into account different priorities (e.g. minimize the construction time). A description of the framework will be given in Section 3 of the paper; its basic scheme is:

1. There is an HTML5-based interface where the user can introduce the data of the project and visualize the

[1]While this is common sense in Architecture, it is not the case in other disciplines such as Computer Science where researchers still advocate the use of formal modeling before initiating the *construction* of a software system [1].

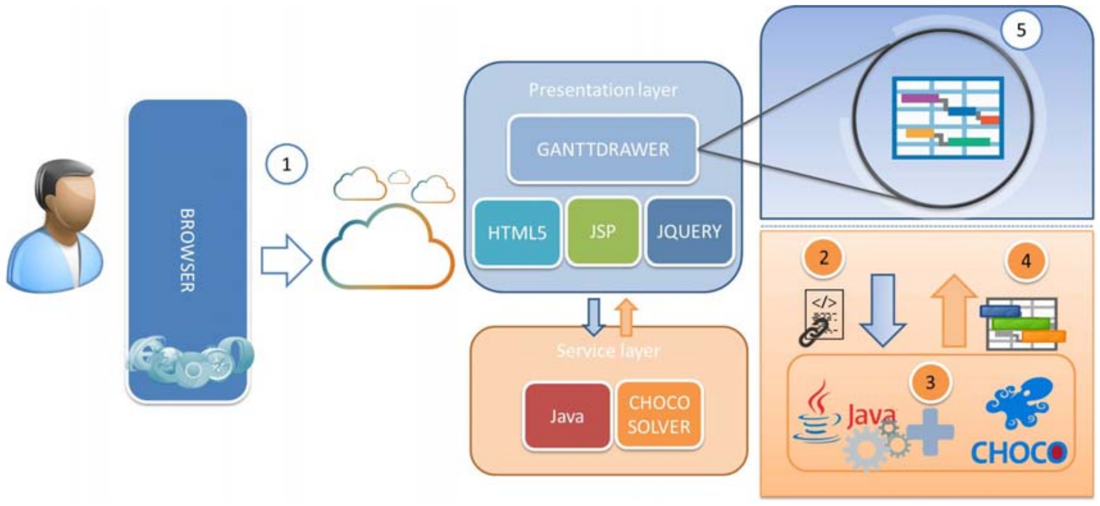


Figure 1: Basic scheme of the approach.

current state of the provided information. In particular, the user will provide preferences on the solution (e.g. minimize the project time).

2. The visualization layer will be connected to a Java engine. This engine will provide the data of the project to a constraint solver.
3. Constraints are grouped into different sets, according to their characteristics (e.g. precedence, optimization). The constraint solver is launched.
4. The constraint solver sends the result back to the resolution engine and the Gantt diagram is presented.
5. The user can access the temporal evolution of the project and can input new information (e.g. one resource is not available when it was planned to be, meteorological conditions produce a delay).

A graphical description of the previous process is shown in Figure 1. It is worth noting that only the first and last steps require human intervention. If modifications to the original plan arise (the last step of the scheme presented above), then the interaction of the user with the system is simple and no previous knowledge about the implementation details is necessary. It is worth to emphasize the importance of the last step. Usually, planners prepare a Gantt diagram. Since this is a *static* artifact, any unplanned changes cannot be easily reflected. In the best case, the planner can manually redo the original diagram by delaying all the tasks accordingly. However, this is not necessarily the best solution because it is very likely that resources can be assigned in an alternative way and/or a different ordering of tasks can be found. Several examples of these situations will be presented.

In order to show the relevance and usefulness of the framework, the planning of a real project is presented: the construction of a water treatment works. All the phases that conform the project are considered, from the removal of trees and vegetation in all areas to the installation of pipes. It is also necessary to take into account the needed tools, from trucks to the software used to design blueprints. During this process the benefits of the approach are shown, in particular, it is possible to optimize the use of available resources.

Concerning related work, the use of formalisms to facilitate the planning of construction sites is not new. In fact, many formalisms have been adapted from Computer Science (CS) and Artificial Intelligence (AI), most notably, multi-agent systems [5, 6, 7, 8]. In this line, it is worth mentioning the role of Building Information Modelling (BIM). During the last years, and mainly due to the increase of the computational power of even the cheapest systems, the BIM framework [9, 10, 11] is becoming very popular and there exist many successful applications [12]. Finally, CS and AI have also influenced planning activities by contributing with novel mechanisms and methodologies based on semantics and ontologies [13, 14, 15, 16]. The framework strongly depends on the use of constraints. Since this field does not belong to the main topics of the journal, Section 2 includes the basic rudiments concerning how constraints can be defined, how constraint programming works, and different types of constraints and constraint solvers.

The rest of the paper is structured as follows. Section 2 reviews the main concepts related to constraints. Since this is a very big field, and far from the interest of most readers of the journal, this part of the paper will concentrate on the use of constraints in this specific setting. Section 3 describes the main inputs of the chain of tools, the basic functioning of these tools, and the methodology. Section 4 explains the constraint solving resolution method. Section 5 presents a case study showing how the methodology and tools can be applied to plan a real construction project. Finally, Section 6 presents the conclusions and some lines for future work.

2. A brief introduction to constraints

Constraint programming (CP in short) has its beginnings in the 1970s [17, 18]. The big advancement of the field happens during the 1980s when classical unification of logic programming is replaced by constraint solving to create constraint logic programming [19]. Afterwards, other programming languages have been extended to include constraints. The use of constraints has clear advantages in terms of conciseness and neat formulation. In fact, many programming languages, such as Prolog, C++, Java and Python, have constraints libraries. This work uses the Java constraint library Choco [20].

Constraints are defined over some specific domains, being *finite domains* the most used ones. In this setting, constraints are defined over finite sets. A *constraint solver* is a decision procedure that checks whether a constraint, or a set of constraints, can be satisfied. A *finite domain solver* is a constraint solver where variables range over finite sets of values, usually finite subsets of the set of integer numbers. Finite domain solvers rely on a systematic exploration of the search space until either a solution is found, or it is shown that the problem does not have a solution. In order to reduce the search space, these solvers are combined with inconsistencies filtering techniques that narrow variable domains.

CP is a broad research area and has many applications in real life. For instance, some systems with applications in industry are the constraint logic programming system CHIP [21] and the IBM ILOG CPLEX Optimizer [22, 23].

Constraint satisfaction models are often benchmarked on hard problems. For example, CSPLib [24] is a library, which is independent of any particular constraint solver, containing different problems organized by subject area. In addition, there exists a world-wide competition of CP solvers: the MiniZinc challenge [25].

It is impossible to cover all the current CP applications. Therefore, three classic problems have been chosen to show the essence of this paradigm. The first problem is "The n queens". This problem belong to a family of problems, including sudoku, solitaires, and others games and puzzles, that can be solved by using an elegant constraint satisfaction mechanism. The second problem is "The car sequencing", a problem of particular interest to the car industry [26]. The last reviewed problem is the "Nurse Rostering problem". In this case, the goal is to find an optimal way to assign shifts to nurses.

2.1. The n queens problem

A solution of this problem consists in placing n queens on a $n \times n$ chessboard so that no pair of queens attack each other [27]. Figure 2, taken from the Choco documentation, shows the code to solve the problem. First, n integer domain variables are generated with values from 1 to n. Next, constraints are *posted*. Note that the constraint `alldifferent` imposes the disequality of all variables (different rows). Finally, all the solutions fulfilling all the constraints are obtained.

```java
// Declaring the domain of the variables
v = new IntVar[n];
for (int i = 0; i < v.length; i++) {
 v[i] = VF.enumerated("Q_"+i,1,n,solver);
}

//Declaring constraints on the declared var.
solver.post(ICF.alldifferent(v, "BC"));
for (int i = 0; i < n - 1; i++) {
 for (int j = i + 1; j < n; j++) {
  int k = j - i;
  solver.post(ICF.arithm(v[i],"!=",v[j],"+",-k));
  solver.post(ICF.arithm(v[i],"!=",v[j],"+",k));
 }
}

// Search for the solutions
solver.findAllSolutions();
```

Figure 2: Choco code for the n-queens problem.

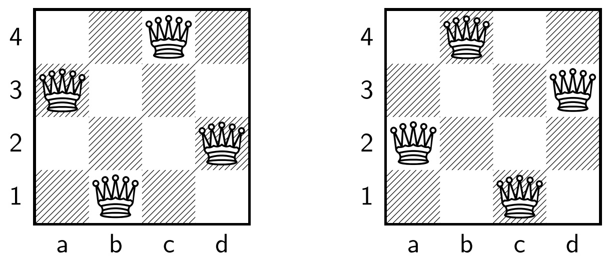


Figure 3: Solutions to the 4-queens problem.

The problem has 2 solutions for $n = 4$ (see Figure 3), 14,200 solutions for $n = 12$, and 227,514,171,973,736 solutions for $n = 24$. These numbers show how quickly the problem grows and the amount of constraints that are created. Currently, it is possible to scale this problem because computers are faster and algorithms and fundamentals of programming have notoriously improved.

2.2. The car sequencing problem

A number of cars are to be produced. These cars are not identical because different options, associated with different variants, can modify the basic model. The assembly line has different stations, which install the various options (air-conditioning, sun-roof, etc). These stations have been designed to handle at most a certain percentage of the cars passing along the assembly line. Furthermore, the cars requiring a certain option must not be bunched together; otherwise, the station will not be able to orderly produce the required cars. Consequently, the cars must be arranged in a sequence so that the capacity of each station is never exceeded. This problem has been shown to be NP-hard [28] and has been widely studied [29, 30].

		Cars					
Opt.	Cap.	1	2	3 4	5 6	7 8	9 10
1	1/2	✓				✓	✓
2	2/3			✓	✓		✓
3	1/3	✓				✓	
4	2/5	✓	✓		✓		
5	1/5			✓			
Configuration		1	2	3	4	5	6

Figure 4: An instance of the car sequencing problem.

Figure 4 shows an example with 10 cars, 5 options and 6 classes. For each option, the maximum number of cars with that option in a block are 1, 2, 1, 2, and 1, respectively. For each option, the block size to which the maximum number refers is 2, 3, 3, 5, and 5, respectively. So, the station for option 2 can process only 2 out of every sequence of 3 cars. Besides, car class 4 requires the installation of options 2 and 4, and 2 cars of this class are required. A valid sequence for this set of cars is 1, 2, 6, 3, 5, 4, 4, 5, 3, and 6.

2.3. The nurse rostering problem

This problem involves assigning shifts to nurses. The solution must take into account holidays, days off and constraints such as "a night shift cannot be followed by a morning shift the very next day." Conventionally, a nurse has three kinds of shifts: day shift, night shift and late night shift. It is necessary to find solutions satisfying as many wishes as possible while all shifts are filled. This problem is known to be NP-hard and there are software applications exclusively devoted to this problem. For example, *Gymnaste* is a nurse planing package jointly developed by *COSYTEC* and the University Joseph Fourier of Grenoble, France.

There are many studies about this problem [31, 32, 33, 34, 35] and there is even an *International Nurse Rostering Competion*.

3. Methodology and associated tools

This section gives a detailed presentation of the tool supporting the framework. The main goal of this tool is to provide an accessible and intuitive interface, which facilitates the construction site planning by using complex mathematical models. It is important to remind that the users of the tool do not need specific knowledge about the underlying theoretical framework. Next, the main components, such as the software architecture design and the diagram editor, are described. In order to facilitate the usage of the tool to potential users, a simple modeling example is included.

3.1. Software architecture of the system

The architecture of the framework is depicted in Figure 5. The framework, developed following the Spring Tool Suite MVC, uses HTML5 [36], JSP [37] and JQuery [38] as the main components of the presentation layer, and the Java engine [39] and Choco solver [20] to conform the service layer.

Initially, the user accesses the system pointing a web browser to `http://antares.sip.ucm.es:8180/automation_in_construction` ①. Then, the problem can be modelled using the diagram editor (see Section 3.2) and its graphical interface, which is considered as the main core of the presentation layer. This editor allows users to define constraints in an easy and intuitive way. In particular, precedence and temporal restrictions can be easily added with the drag & drop and gesture interaction methods included in the diagram builder ②. Once the whole construction site has been designed, the information is converted to a JSON (JavaScript Object Notation) format ③ (an example of the format is shown in Figure 6). Then, the JSON data is sent to the Spring Front Controller through a jQuery request ④, where it is processed by the Jackson parser in order to convert it into structures compatible with the Java engine ⑤. The next step consists in preprocessing these structures and combining them with the Choco constraint solver ⑥. Then, the Java engine sends the structures to the constraint solver ⑦. The information used in the constraint resolution process, composed of several parameters such as duration, cost and temporal constraints, is analysed and processed by the constraint solver. At this point, the constraint resolution has finished and the solution is converted into JSON using GSon (acronym of the Java Google JSON library that can be used to convert Java objects into their JSON representation) and passed to the Spring Front Controller ⑧. Finally, the solution is sent back to the diagram editor using jQuery and this component presents the results to the user using the web interface ⑨.

3.2. Diagram editor

The diagram editor is based on the Gantt editor twProject [40]. This component is illustrated in Figure 7 and it has three main parts: a *button panel* ①, an *editor panel* ② and a *visualization panel* ③.

The *button panel* allows users to model the different tasks of the construction site. The first group of buttons is oriented to tasks edition, allowing to undo/redo the last action ⓐ, insert a task above/below the selected one ⓑ, indent/unindent tasks to create subtasks ⓒ, move up/down a task ⓓ, make zoom in/out and refresh the diagram ⓔ, delete either a specific task or the whole diagram ⓕ, print the diagram ⓖ and identify the critical path of the diagram ⓗ. In addition, control buttons are included. The *export button* saves the current configuration into a file located in the local system ⓘ. The *process button* invokes the constraint solver engine to handle the current configuration ⓚ. The *holidays button* allows users to include non-working days into the diagram, in order to consider several contingencies ⓙ. *Total budget* sets the total money spent in the whole project ⓛ.

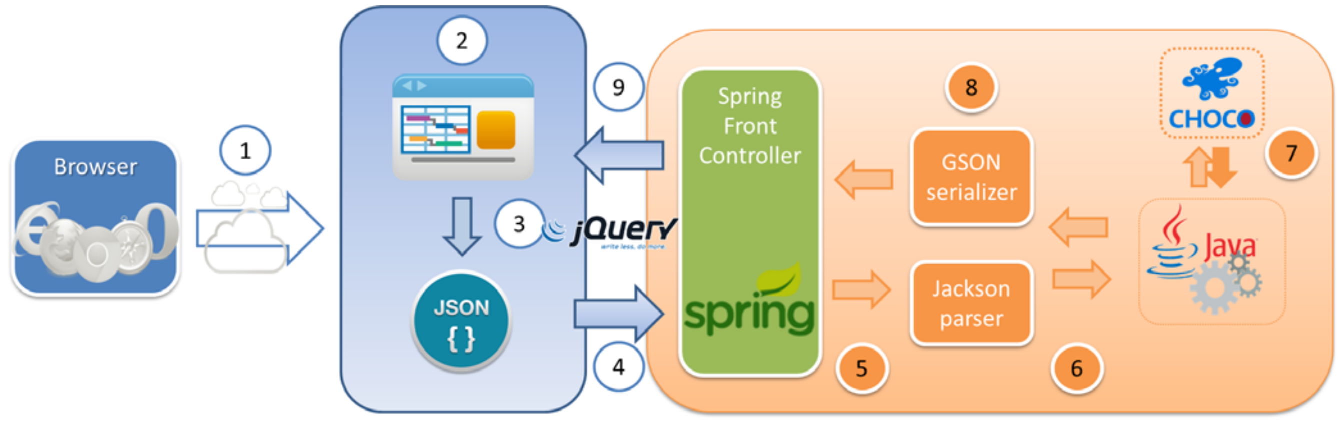
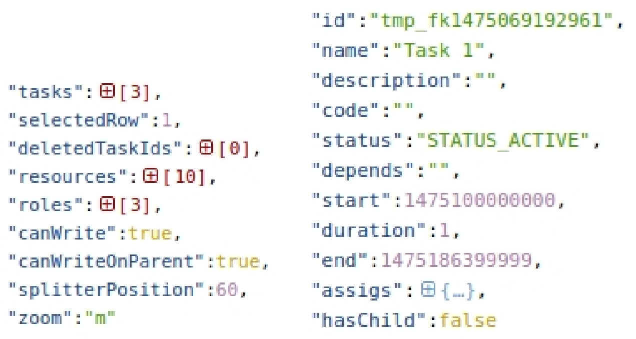


Figure 5: Graphical view of the methodology.



```
"tasks":⊞[3],                          "id":"tmp_fk1475069192961",
"selectedRow":1,                       "name":"Task 1",
"deletedTaskIds":⊞[0],                 "description":"",
"resources":⊞[10],                     "code":"",
"roles":⊞[3],                          "status":"STATUS_ACTIVE",
"canWrite":true,                       "depends":"",
"canWriteOnParent":true,               "start":1475100000000,
"splitterPosition":60,                 "duration":1,
"zoom":"m"                             "end":1475186399999,
                                       "assigs":⊞{…},
                                       "hasChild":false
```

Figure 6: Simplified version of the JSON format.

Figure 8 shows the editor to insert *non-working days*. The *date* field ① indicates the selected day to be considered as non-working; it can be chosen using a graphic calendar ③. The *description* field is used to include a specific description of the chosen day ②. In addition, these days can be managed, either creating new instances or deleting them, with the ⊕ and ⊗ symbols, respectively. Finally, the *save* button allows users to insert the designed non-working days into the system.

The *editor panel* (see Figure 7 ②) helps the user to model the construction site in a simple and intuitive way. The user can create tasks by introducing a reduced set of parameters: *id* is the unique identifier of the task, *code* is used to assign a tag, *status* indicates whether the task is active, completed, failed, suspended or undefined, *description* is a brief description of the task, *start* is the date when the task begins, *end* is the date when the task finishes, *duration* is the period of time, in days, taken to complete the task, *dependencies* represent the temporal constrains between tasks, and *cost* is the amount of money spent in this task.

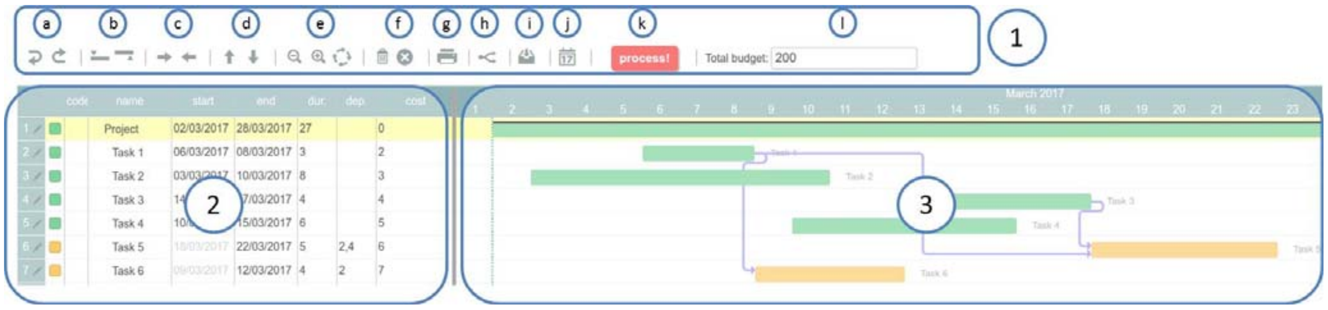
The *visualization panel* (see Figure 7 ③) contains a calendar where each task is located. The position of each task depends on its starting and duration time values and also shows the temporal dependencies between tasks. In order to facilitate the interpretation of the status of the project, each task has an associated colour to show its current state: green when the task is *active*, blue when it is *completed*, purple when the task has been *suspended*, yellow if the task has *failed* and, finally, white when the task has entered in an *undefined* state. In addition, the constrains can be generated in a graphical way, see Figure 9, by joining tasks with directed arrows.

3.3. A simple modelling example

The framework incorporates a strong mathematical model, which provides a set of mechanisms to optimize the resolution of the scheduling problem. The tool supporting the formal framework allows users to profit from the underlying powerful mathematical techniques, in an automatic way, without needing any knowledge of their intricacies.

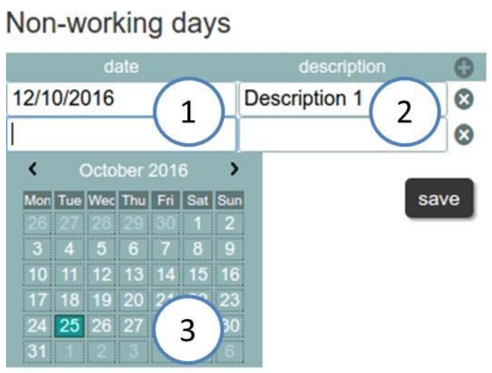
Next, a simple example shows potential users the modelling process, using the tool, of a scheduling problem. The notation provided in Figure 7 will be used in order to refer to the different elements of the diagram editor. First, in order to analyse its construction project, the user must access the graphical diagram trough the web interface `www.antares.sip.com:8180/automation_in_construction/gantt_template`. Initially, the editor is empty and the user must introduce new tasks in the diagram by pressing the *insert top/above* buttons ⓐ. Afterwards, the user fills the information fields related to the duration, dependencies and cost associated to the task under design (see Figure 7 ②). The result of these initial steps is shown in Figure 10. The dependencies between tasks can be designed using the diagram editor in a graphical way, as it is shown in Figure 9. Once the problem has been totally designed (see Figure 11), the optimization process can be invoked by pressing the *process* button ⓚ. Then, the scheduling is processed in the system core and the results are presented to the user as a diagram (see Figure 12).

At this point, several constraints can be applied through different optimization iterations. For example, non-working days can be introduced on-the-fly by using the



		code	name	start	end	dur.	dep	cost
1	🟩		Project	02/03/2017	28/03/2017	27		0
2	🟩		Task 1	06/03/2017	08/03/2017	3		2
3	🟩		Task 2	03/03/2017	10/03/2017	8		3
4	🟩		Task 3	14	7/03/2017	4		4
5	🟩		Task 4	10/	15/03/2017	6		5
6	🟧		Task 5	18/03/2017	22/03/2017	5	2,4	6
7	🟧		Task 6	09/03/2017	12/03/2017	4	2	7

Figure 7: Gantt editor.



date	description
12/10/2016	Description 1

Figure 8: Non-working days editor.

dedicated editor (see Figure 8). If something unforeseen happens, then a non-working day can be added by using the corresponding editor. For this, the user must press the ⊕ button, fill the *date* and *description* fields and press the *save* button. Once introduced, the system automatically recomputes the optimal scheduling of the project (see Figure 13).

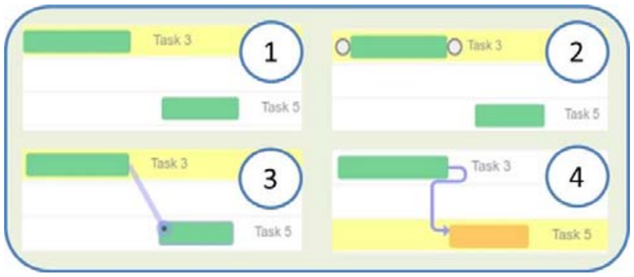


Figure 9: Graphical constraint edition.

4. The constraint solving process

This section reviews the main concepts and ideas underlying the mathematical model used to resolve the scheduling problem. The example given in Section 3.3, having six tasks identified with six variables $\texttt{task}_i$, with $1 \leq i \leq 6$, is used again.

First, the domain of the variables is declared. There are different kinds of constraint domains in constraint programming. Among the most used, it is possible to mention the *boolean domain*, where only true/false constraints apply, and the *finite domains*, where constraints are defined over finite sets. The scheduling problem considered in this paper has a *hybrid domain*: it uses both boolean and finite domain variables.

The task_i is defined by using two finite domain variables, representing the beginning ($\texttt{startVars}_i$) and the end ($\texttt{endVars}_i$) of the task. Each of them has an integer domain, ranking from 1 to n (that is, the interval $[1, n]$), where n is either the deadline date given by the user or, by default, the time spent on executing all tasks sequentially.

The first imposed constraint is that the end of each task is equal to its beginning plus its duration. These constraints enable a scenario as the one given in Figure 10.

$$(1) \quad \texttt{endVars}_i \texttt{ - startVars}_i = \texttt{durations}_i\texttt{-1}, \forall i \in [1, 6]$$

Constraint propagation is a well-known mechanism in constraint programming. It reduces the domains of variables and, correspondingly, the search space. By reducing the search space, a more efficient performance can be achieved. If the values associated with one variable are reduced to the empty set, then the constraint system is inconsistent. In this case, the propagation of the constraint (1) removes the extreme interval values that cannot be satisfied, while the next two constraints enforce the so-called *bound consistency*.

$$(2) \quad \texttt{[1,durations}_i\texttt{-1]} \notin \texttt{dom(endVars}_i\texttt{)}, \forall i \in [1, 6]$$
$$(3) \quad \texttt{[n-durations}_i\texttt{+1]} \notin \texttt{dom(startVars}_i\texttt{)}, \forall i \in [1, 6]$$

Once the domains of the variables are established, and pruned by propagation, constraints can be imposed. Specifically, there are three types of restrictions: precedence, no-overlapping, and optimization constraints.

The next precedence constraint reflects the order between tasks i and j, where the end of the task i is restricted to be less than the beginning of task j. By abuse of notation, the relational operator $<$ is used to establish the relation of order between finite variables.

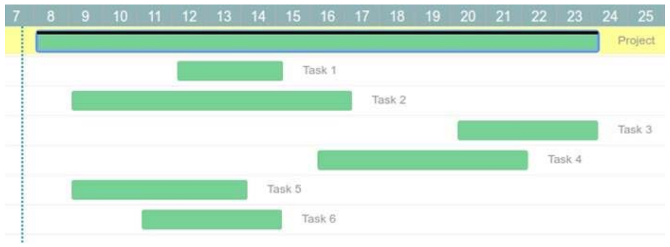


Figure 10: Diagram without dependencies.

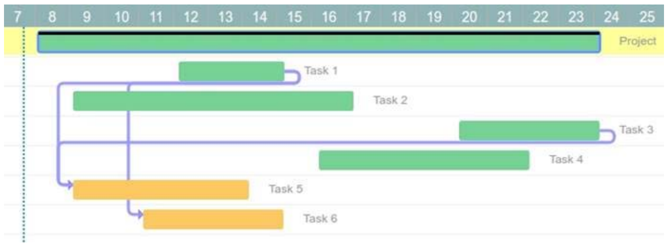


Figure 11: Diagram with dependencies included.

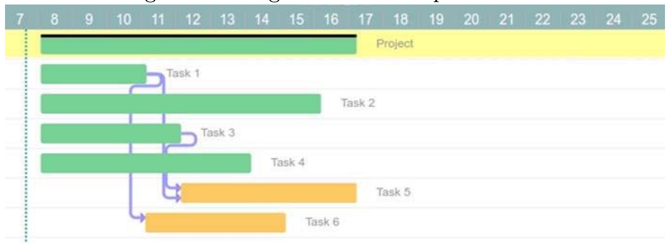


Figure 12: Diagram processed by the constraint solved.

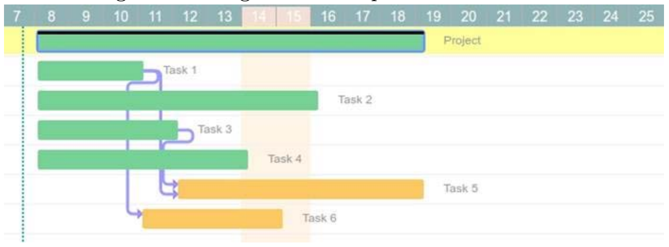


Figure 13: Diagram processed with non-working days.

$$(4) \quad \texttt{precedence(i,j)} = \texttt{endVars}_i < \texttt{startVars}_j$$

A possible scenario of a scheduling problem with precedence constraints is given in Figure 11. In this case, only a small number of values can be eliminated by propagation because the corresponding variables, $\texttt{endVars}_i$ and $\texttt{startVars}_j$, do not have yet concrete values. In contrast, many values of the variables associated by precedence constraints are removed after the variables are instantiated.

In addition to precedence constraints, it is common to require some resource optimization. The example requires that the tasks are planned in the shortest time, that is, the minimization of the execution time. The heuristic applied in these situations is known as *makespan*. The idea consists in creating a new *artificial* task that will be the last to execute and it is required to be executed as soon as possible. The makespan task is defined with the finite domain variables `startMakespan` and `endMakespan`. Initially, the makespan domain is the interval $[1, n+1]$, but it is quickly narrowed by constraints propagation. The duration of the makespan is zero. Finally, in order to ensure that makespam is the last task, precedence constraints are set for all tasks, see next constraint, as it is shown in Figure 14.

$$(6) \; \{ \; \texttt{endVars}_i < \texttt{startMakespan}_j \mid 1 \leq i \leq 6 \; \}$$

The solution of the problem is one of the possible assignments minimizing either the start or end times. This is so because the duration is zero (see Figure 12). Then, the respective minimization constraints are posted over the finite domain variable `startMakespan` as given by the last constraint of the system.

(7) `findOptimalSolution(minimize, makespan)`

At this point, all restrictions are already set and all the inconsistent values of the domains of the variables have been removed. Then, the next step is to explore the solutions space. In order to find the different candidate solutions, the *labeling constraint* assigns ground values to variables. The choice of the variables to be instantiated and the concrete values depend on the different selection strategies. For each instantiation, it is necessary to check that the system is satisfactory; if it is unsatisfiable then it is tested with another instantiation by means of a *backtracking* mechanism.

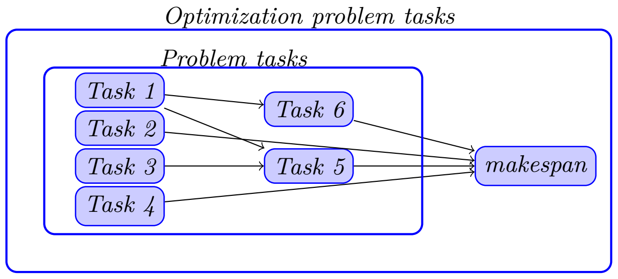


Figure 14: Makespan for minimization problems.

In addition to the above, the tool allows to establish non-working days due to, for example, meteorological conditions. When a non-working day is set, all tasks affected by this day are extended by one more day in their duration. For example, Figure 13 shows how `task3` is affected by the non-working day and its duration is increased by one more day. Consequently, the dates associated with `task5` must be recomputed.

5. Project: Water Treatment Works

This sections shows how all the capabilities of the framework can be used in order to obtain an efficient planning of a real project performed in the United Kingdom (UK) in 2015. The project implemented the construction of a water treatment works with two essential goals: treat waters by introducing Ultra Violet Disinfection (UVD) and reduce plumbosolvency in the network by installing a sodium dihydrogen orthophosphate dosing equipment. Some of the data have been anonymized with the aim of preserving the privacy of clients and contractors. Notwithstanding this fact, all the requirements and conditions of the project have remained the same. However, in order to reduce the length of the case study and concentrate on the most relevant features, only time durations, labour and the cost of their extra contracting have been considered in this section. The details related to the amounts and costs of materials have been discarded and, for simplicity, it has been supposed that these amounts are fixed.

5.1. Description of the site

The future water treatment works is located in the UK and it is designed to produce around 3Ml/day. Its location is close to natural springs where water is extracted for potable water supply. This source uses a piped gravity system into a small storage reservoir where raw water gravitates to the suction of three pumps. After the corresponding treatment, which will take place in these pumps, the treated water feeds a distribution supply zone at a head of approximately 100 metres. The flow/pressure control valve is operated to feed a lower distribution area at a head of approximately 40 metres.

The project will comprise all disinfection works, chemical dosing works, buried services, electrical works and road works. In addition to these tasks, the automation of the site will be fulfilled by introducing raw water monitoring, installation of CCTV (Closed Circuit Television) and installing SCADA (Supervisory, Control And Data Acquisition) with facilities to remotely view information.

5.2. Eco-friendly construction

One of the requirements to take into account is the eco-friendly performance of this construction. The contractor of the works will help the clients to reduce the impact in the environmental sustainability. These improvements are related to:

- Reduce the generation of construction waste.
- Reduce energy consumption.
- Reduce CO_2 emissions.
- Protect biodiversity of the area to be constructed.
- Reduce the proportion of materials.
- Reduce the nuisance and minimise inconvenience caused to local residents and businesses.

The achievement of these requirements implies numerous financial benefits, such as reductions in the capital cost of the project, reductions in operating costs and the completion of the project at an earlier date. The framework facilitates the modeling of these actions with the dynamical modifications of the constraints that conform the project. The only requirement is the correct definition of the conditions for building the works. This will allow users to make the most of the resources that are available for the development of the project.

Principal Activities	
Activity	**Cost**
A - Detailed design	£ 21750
B - Surveys and Investigation	
C - Preparation of H&S Plans in respect to the CDM regulations (2015)	
D - Preparation of quality plans, methods statements and programmes	
E - Risk contingency	
F - Production of Operation and Maintenance documentation	
G - Production of recorded Drawings	
H - Fee	
I - Contractor's accommodation and mobilisation	£ 27709
J - Supply, construction, installation, testing and commissioning activities	£ 398193
K - Training of DVW Operatives	£ 2550
L - Commissioning	£ 5450
TOTAL	**£ 455652**

Table 1: Main activities of the project.

5.3. Original activity schedule

The initial planning of the project is depicted in Table 1. This list includes the 12 activities composing the works and their estimated costs. Given the fact that the main goal is to find optimal time durations to plan construction projects, the most relevant part of the project is Activity J: *Supply, construction, installation, testing and commissioning activities*. This generic activity includes the most important construction activities and the commissioning activities performed by the personnel of the project. Activities A to H represent the initial steps that should be taken to start the project such as the design of the project, preparation of the plans, risks control and the planning of the fees/taxes that have to be paid at the end of the project. Activities I and J deal with the formation and contracting of labour. Finally, Activity L covers all the commissioning activities that are developed by external personnel of the project, with the aim of checking the correctness, security and safety of the final product obtained from this planning.

Table 1 reflects that Activity J is the most expensive task of the project. Therefore, potential time/labour improvements will lead to a reduction in the final budget. Table 2 presents the duration of the 15 sub-activities conforming Activity J. The original organization of the project,

Activity J		
Task	Description	Total Days
J1	**Roads, fencing gates**	**58**
J1.1	Removal of trees and vegetation in all areas	10
J1.2	Road works	40
J2	**Excavation and associated temporary works**	**13**
J2.1	Raw water main trench	5
J2.2	Excavations and levelling in kiosk and tanks area	8
J3	**Contractor's compound access**	**2**
J4	**Kiosk for UVD equipment**	**2**
J5	**Kiosk base slab**	**3**
J6	**UVD equipment**	**3**
J7	**Raw water monitoring instruments**	**60**
J7.1	Sodium dihydrogen Orthophosphate storage tank	45
J7.2	Blind tank and bunded area	15
J8	**Sodium dihydrogen Orthophosphate plant**	**3**
J9	**Installation of the new war water main**	**47**
J9.1	Installation of new polyethylene raw water main	32
J9.2	Ductile iron pipes & fitting inside kiosk	15
J10	**Electrical works**	**25**
J11	**Ducts**	**5**
J12	**Software and SCADA**	**5**
J13	**CCTV Camera**	**2**
J14	**Safety shower**	**2**
J15	**Chemical lines**	**5**

Table 2: Activity J: Tasks and durations.

concerning this activity, is reflected in the Gantt diagram given in Figure 15.

5.4. Management of activities scheduling

This section presents a usual scenario where the application of the tool produces an important reduction of effort and time needed to manage scheduling problems. Specifically, it considers a situation where there are unexpected problems due to inclement weather and the result is a delay in the project. These situations show the usefulness of the framework because adding a small number of data concerning the new non-working dates is enough to automatically recompute the new optimal scheduling of tasks.

Consider the concepts and notation given in Figure 15 and Table 2 and suppose that during the implementation of the activities J1.1, J2 and J2.1 of the project, the project suffered a storm, which lasted 2 days. As a consequence, all the activities were stopped during these 2 days, producing the corresponding delays. According to the Gantt diagram of the project, the activities J1.2, J2.2, J6, J7.1, J9.1, J11, J12, J14 and J15 depend on the delayed activities. Therefore, the planner needs to study and manage, very likely by hand, these activities and all their dependencies. However, using the capabilities of the framework, the person in charge only needs to follow the following steps:

- Represent the "Water treatment works" project in the web interface. Note that if the tool was used before to manage the current project, then there is no need to add again the activities and the relations between them.
- As previously mentioned, the activities that suffer the delays are J1.1, J2 and J2.1. Therefore, it is necessary to modify their duration with the option that the diagram editor offers: double click over them and modify their date of ending.
- With each activity, this last action triggers the opening of the "task editor" window, where its end date can be modified by adding two more days. After accepting the changes, it is possible to observe that the description fields of the activity show some changes.
- In order to apply these modifications to all the project and the activities that depend on them, it is necessary to push the "process!" button. In a few seconds, the original duration changes from 72 to 89 days due to the chain of dependencies between the activities.
- The new planning shows that the dependent activities have changed too and this is the reason why the number of days is greater. However, this solution is completely applicable to the project because it considers all the restrictions that were initially defined.

After all these steps, the new planning has been obtained but without the necessity of manually reorganizing all the activities. This example highlights the benefits that the tool provides to its potential users. It will help the professionals to elaborate their work in less time, in particular, if they have to recompute solutions after unexpected changes, by putting in practise the advantages of the processing power of current computers.

6. Conclusions and future work

Construction projects require to consider numerous constraints and requirements, which should be fulfilled during the whole project. However, there are unexpected situations modifying the original plan. These include holidays, strikes, weather conditions and lack of resources, including personnel. In order to avoid the extra work to recompute the working plan, a tool-supported planning framework is provided. The tool reorganizes the initial planning by applying the new restrictions. This saves time and effort to the professionals. If they have to face a scenario where the initial conditions change, then they obtain a new solution within seconds. This new schedule can be directly applied because the tool takes into account the constraints that have been previously indicated.

As future work, it would be useful to add some further improvements to the formal framework and include them in the tool. In order to facilitate the interaction of new users with the tool, it should be possible to include the option of uploading Gantt diagrams produced in other formats to the tool. With this feature, users do not need to build the project from scratch when they have to manage changes in the evolution of the planning and they did not use the tool to originally develop the project. Another short-term enhancement would be the inclusion of more constraint conditions such as managing the restrictions in terms of staff, materials and possible dependencies with other projects. These would provide a more realistic view of the planning of a construction project, showing the different priority relations between the involved elements.

Acknowledgements

Research partially supported by the Spanish MINECO/FEDER project DArDOS (TIN2015-65845-C3-1-R) and the Comunidad de Madrid project SICOMORo-CM (S2013/ICE-3006). The authors would like to thank the collaboration of the construction engineer Jaime Domínguez Medina for his productive comments on the scenario presented in this paper.

[1] L. Lamport, Who builds a house without drawing blueprints?, Communications of the ACM 58 (4) (2015) 38–41, ISSN 0001-0782, doi: 10.1145/2736348.

[2] L. Wang, F. Leite, Formalized knowledge representation for spatial conflict coordination of mechanical, electrical and plumbing (MEP) systems in new building projects, Automation in Construction 64 (2016) 20–26, ISSN 0926-5805, doi: 10.1016/j.autcon.2015.12.020.

[3] J. Wang, X. Wang, W. Shou, H.-Y. Chong, J. Guo, Building information modeling-based integration of MEP layout designs and constructability, Automation in Construction 61 (2016) 134–146, ISSN 0926-5805, doi: 10.1016/j.autcon.2015.10.003.

[4] F. Rossi, P. van Beek, T. Walsh (Eds.), Handbook of Constraint Programming, vol. 2 of *Foundations of Artificial Intelligence*, Elsevier, ISBN 978-0-444-52726-4, 2006.

[5] A. Sawhney, H. Bashford, K. Walsh, A. Mulky, Agent-based Modeling and Simulation in Construction, in: S. E. Chick, P. J. Sanchez, D. M. Ferrin, D. J. Morrice (Eds.), 35th Winter Simulation Conference: Driving Innovation, New Orleans, Louisiana, USA, December 7-10, 2003, vol. 2, IEEE Computer Society, ISBN 0-7803-8131-9, 1541–1547, doi: 10.1145/1030818.1031027, 2003.

[6] K. Kim, K. J. Kim, Multi-agent-based simulation system for construction operations with congested flows, Automation in Construction 19 (7) (2010) 867–874, ISSN 0926-5805, doi: 10.1016/j.autcon.2010.05.005.

[7] C. Molinero, M. Núñez, Planning of work schedules through the use of a hierarchical multi-agent system, Automation in Construction 20 (8) (2011) 1227–1241, ISSN 0926-5805, doi: https://doi.org/10.1016/j.autcon.2011.05.006.

[8] F. Taillandier, P. Taillandier, E. Tepeli, D. Breysse, R. Mehdizadeh, F. Khartabil, A multi-agent model to manage risks in construction project (SMACC), Automation in Construction 58 (2015) 1–18, ISSN 0926-5805, doi: 10.1016/j.autcon.2015.06.005.

[9] B. Succar, Building Information Modelling framework: a research and delivery foundation for industry stakeholders, Automation in Construction 18 (2009) 357–375, ISSN 0926-5805, doi: 10.1016/j.autcon.2008.10.003.

[10] C. Eastman, P. Teicholz, R. Sacks, K. Liston, BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors, John Wiley, 2nd edn., ISBN 9781118021699, 2011.

[11] V. Quirk, A Brief History of BIM, Arch Daily, http://www.archdaily.com/302490/a-brief-history-of-bim, last access: 17/10/2017, 2012.

[12] H. Son, S. Lee, C. Kim, What drives the adoption of building information modeling in design organizations? An empirical investigation of the antecedents affecting architects' behavioral intentions, Automation in Construction 49 (A) (2015) 92–99, ISSN 0926-5805, doi: 10.1016/j.autcon.2014.10.012.

[13] B. T. Zhong, L. Y. Ding, P. E. D. Love, H. B. Luo, An ontological approach for technical plan definition and verification in construction, Automation in Construction 55 (2015) 47–57, ISSN 0926-5805, doi: 10.1016/j.autcon.2015.02.002.

[14] W. Terkaj, A. Šojic, Ontology-based representation of IFC EXPRESS rules: An enhancement of the ifcOWL ontology, Automation in Construction 57 (2015) 188–201, ISSN 0926-5805, doi: 10.1016/j.autcon.2015.04.010.

[15] J. Oraskari, S. Törmä, RDF-based signature algorithms for computing differences of IFC models, Automation in Construction 57 (2015) 213–221, ISSN 0926-5805, doi: 10.1016/j.autcon.2015.05.008.

[16] M. Niknam, S. Karshenas, Integrating distributed sources of information for construction cost estimating using Semantic Web and Semantic Web Service technologies, Automation in Construction 57 (2015) 222–238, ISSN 0926-5805, doi: 10.1016/j.autcon.2015.04.003.

[17] U. Montanari, Networks of constraints: Fundamental properties and applications to picture processing, Information Sciences 7 (1974) 95–132, ISSN 0020-0255, doi: 10.1016/0020-0255(74)90008-5.

[18] A. K. Mackworth, Consistency in networks of relations, Artificial Intelligence 8 (1) (1977) 99–118, ISSN 0004-3702, doi: 10.1016/0004-3702(77)90007-8.

[19] J. Jaffar, M. J. Maher, Constraint Logic Programming: A Survey, Journal of Logic Programming 19/20 (1994) 503–581, ISSN 0743-1066, doi: 10.1016/0743-1066(94)90033-7.

[20] C. Prud'homme, J.-G. Fages, X. Lorca, Choco Documentation, TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S., URL `http://www.choco-solver.org`, last access: 17/10/2017, 2016.

[21] H. Simonis, Building Industrial Applications with Constraint Programming, in: H. Comon, C. Marché, R. Treinen (Eds.), Constraints in Computational Logics: Theory and Applications, International Summer School, CCL'99 Gif-sur-Yvette, France, September 5-8, 1999, Springer, New York, NY, USA, ISBN 3-540-41950-0, 271–309, doi: 10.1007/3-540-45406-3_6, 2001.

[22] PR Newswire, Lufthansa Adopts ILOG Optimization to Enhance Its Crew Assignment Process, http://www.prnewswire.com/news-releases/lufthansa-adopts-ilog-optimization-to-enhance-its-crew-assignment-process-56194152.html, last access: 17/10/2017, 2006.

[23] PR Newswire, ILOG CPLEX offers industry-first features for real-world planning and scheduling tasks, http://www.prnewswire.com/news-releases/ilog-cplex-offers-industry-first-features-for-real-world-planning-and-scheduling-tasks-55368292.html, last access: 17/10/2017, 2006.

[24] I. P. Gent, T. Walsh, CSPlib: A Benchmark Library for Constraints, in: J. Jaffar (Ed.), 5th Int. Conf. on Principles and Practice of Constraint Programming, CP'99, Alexandria, Virginia, USA, October 11-14, 1999, LNCS 1713, Springer, London, UK, ISBN 978-3-540-48085-3, 480–481, doi: 10.1007/978-3-540-48085-3_36, 1999.

[25] P. J. Stuckey, R. Becket, J. Fischer, Philosophy of the MiniZinc challenge, Constraints 15 (3) (2010) 307–316, ISSN 1383-7133, doi: 10.1007/s10601-010-9093-0.

[26] C. Solnon, V. D. Cung, A. Nguyen, C. Artigues, The car sequencing problem: Overview of state-of-the-art methods and industrial case-study of the ROADEF'2005 challenge problem, European Journal of Operational Research 191 (3) (2008) 912–927, ISSN 0377-2217, doi: 10.1016/j.ejor.2007.04.033.

[27] J. Bell, B. Stevens, A survey of known results and research areas for n-queens, Discrete Mathematics 309 (1) (2009) 1–31, ISSN 0012-365X, doi: 10.1016/j.disc.2007.12.043.

[28] T. Kis, On the complexity of the car sequencing problem, Operations Research Letters 32 (4) (2004) 331–335, ISSN 0167-6377, doi: 10.1016/j.orl.2003.09.003.

[29] M. Fliedner, N. Boysen, Solving the car sequencing problem via Branch & Bound, European Journal of Operatio-

nal Research 191 (3) (2008) 1023–1042, ISSN 0377-2217, doi: 10.1016/j.ejor.2007.04.045.

[30] M. Siala, E. Hebrard, M. J. Huguet, A study of constraint programming heuristics for the car-sequencing problem, Engineering Applications of Artificial Intelligence 38 (2015) 34–44, ISSN 0952-1976, doi: 10.1016/j.engappai.2014.10.009.

[31] B. Cheang, H. Li, A. Lim, B. Rodrigues, Nurse rostering problems - a bibliographic survey, European Journal of Operational Research 151 (3) (2003) 447–460, ISSN 0377-2217, doi: 10.1016/S0377-2217(03)00021-3.

[32] E. K. Burke, P. De Causmaecker, G. V. Berghe, H. Van Landeghem, The State of the Art of Nurse Rostering, Journal of Scheduling 7 (6) (2004) 441–499, ISSN 1094-6136, doi: 10.1023/B:JOSH.0000046076.75950.0b.

[33] M. Stølevik, T. E. Nordlander, A. Riise, H. Frøyseth, A Hybrid Approach for Solving Real-world Nurse Rostering Problems, in: J. Lee (Ed.), 17th Int. Conf. on Principles and Practice of Constraint Programming, CP'11, Perugia, Italy, September 12-16, 2011, LNCS 6876, Springer, ISBN 978-3-642-23785-0, 85–99, doi: 10.1007/978-3-642-23786-7_9, 2011.

[34] P. Smet, P. D. Causmaecker, B. Bilgin, G. V. Berghe, Nurse Rostering: A Complex Example of Personnel Scheduling with Perspectives, in: A. S. Uyar, E. Ozcan, N. Urquhart (Eds.), Automated Scheduling and Planning - From Theory to Practice, vol. 505 of *Studies in Computational Intelligence*, Springer, ISBN 978-3-642-39303-7, 129–153, doi: 10.1007/978-3-642-39304-4_6, 2013.

[35] E. K. Burke, T. Curtois, New approaches to nurse rostering benchmark instances, European Journal of Operational Research 237 (1) (2014) 71–81, ISSN 0377-2217, doi: 10.1016/j.ejor.2014.01.039.

[36] I. Hickson, HTML5 Specification, https://www.w3.org/TR/html5/, last access: 17/10/2017, 2015.

[37] M. Hall, More servlets and JavaServer pages, Prentice Hall PTR, ISBN 978-0130676146, 2001.

[38] B. Bibeault, Y. Kats, jQuery in Action, Dreamtech Press, 2nd edition, ISBN 978-1935182320, 2008.

[39] J. Gosling, B. Joy, G. L. S. Jr., G. Bracha, A. Buckley, The Java language specification, Addison-Wesley Professional, 8th edition, ISBN 978-0133900699, 2014.

[40] Twproject, Twproject Gantt, Available at `https://gantt.twproject.com/`, last access: 17/10/2017, 2015.

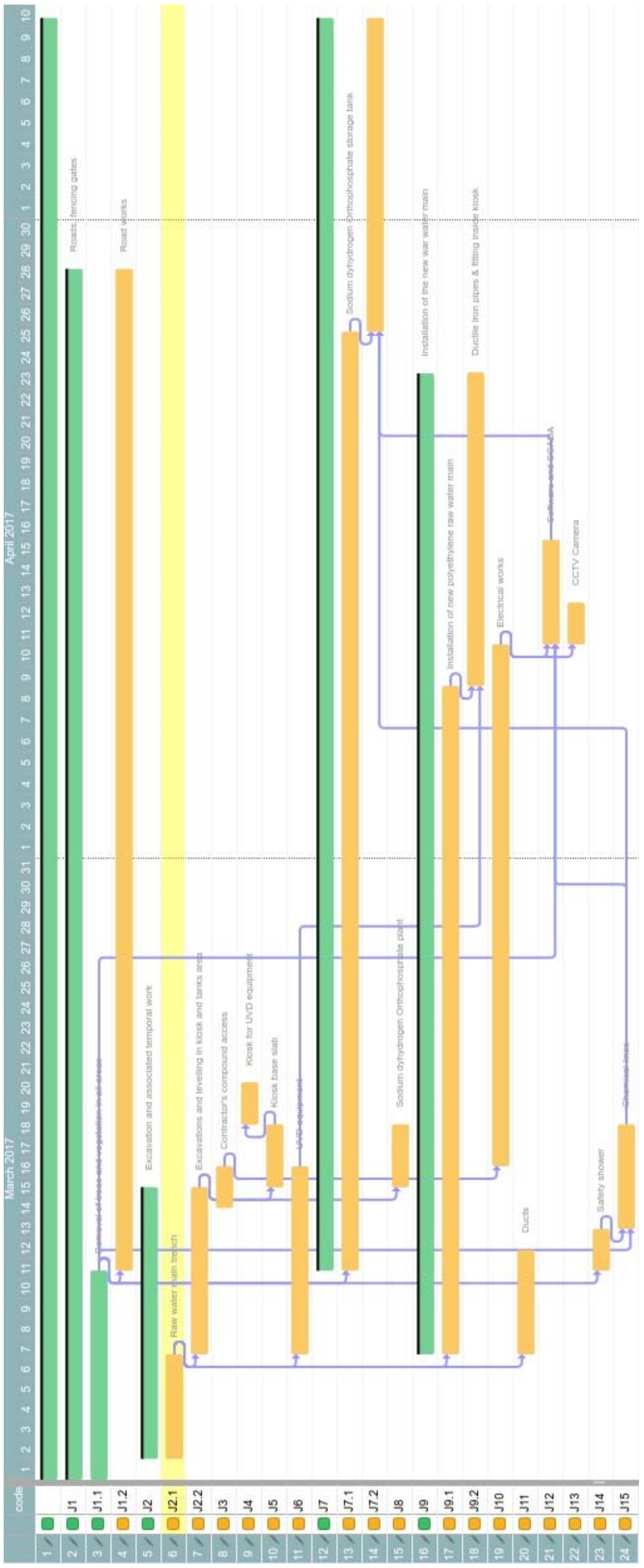


Figure 15: Original Gantt diagram of the project.