$\mathcal{SINPA}$: SupportINg the automation of construction PlAnning

Pablo C. Cañizares[1], Sonia Estévez-Martín[2] and Manuel Núñez[3*]

[1]*Modelling & software engineering research group, Computer Science Department, Autonomous University of Madrid, Spain*

e-mail: pablo.cerro@uam.es

[2]*Design and Testing of Reliable Systems research group, Universidad Complutense de Madrid, 28040, Madrid, Spain*

e-mail: soesteve@ucm.es

[3]*Institute of Knowledge Technology, Universidad Complutense de Madrid, 28040, Madrid, Spain*

e-mail: manuelnu@ucm.es

Abstract

We present $\mathcal{SINPA}$: an integrated framework to support construction site planners. The most basic functionality of our tool provides a user-friendly framework to represent causality relations (precedence and parallelism) between the different tasks conforming a project. $\mathcal{SINPA}$ strongly relies on a constraint solver and makes an intensive use of constraints. $\mathcal{SINPA}$ automatically studies optimisations of the original planning, recomputes the solutions and provides recommendations in an intuitive way so that the planners can modify their original plan. It is important to emphasize that the users of $\mathcal{SINPA}$ do not need to work with or understand Constraint Programming.

Keywords: Constraint programming, Construction sites planning, Tools to support projects development

1. Introduction

The development of complex systems and structures should always be preceded by the creation of a *model* of the desired result (Magnani and Bertolotti, 2017). Although this is not always the case in Computer Science (Lamport, 2015), models are particularly important in Construction, where blueprints and plans to establish causality relations between different phases of even the simplest projects are available before the first brick is laid. Planning is a necessary step in Construction because it facilitates the successful organisation and management of construction sites. In fact, it is becoming more common to provide a semi-formal description of the different tasks involved in construction sites. In this line, Building information modelling (BIM) (Eastman et al., 2011) is a good example of modelling framework to support knowledge management in Construction (Wang and Meng, 2019). Planning must take into account how the different specialists, contractors and dedicated machinery will be deployed in the construction site. The final goal is to reduce both cost and time by analyzing the availability of these resources (including human resources) and the time constraints associated with the different tasks.

In order to do a proper planning of a construction site, and having in mind that the minimisation of idle resources will have a positive impact in cost and time, it is important to distinguish between tasks that must be sequentially implemented (for example, terrain movement must be finished before the construction of a concrete surface bed starts) and tasks that could be implemented in parallel. Due to its simplicity and the availability of tools, *Gantt diagrams* are a very good option to specify the temporal planning and different phases of a construction site. Even though experienced architects are usually in charge of producing these plans, it would be very useful to automatically produce an optimal plan, reflected in a notation similar to these diagrams, from some initial data. In particular, it is very important that the tools supporting the creation of Gantt diagrams are user-friendly and do not require that the user has knowledge about the internal design and behaviour of the tool.

In previous work (Camacho et al., 2018) we developed a system that generated Gantt-like diagrams, which could be use for planning of construction sites. The user has to provide some basic, but relevant, data concerning causality relation between tasks of the project. Our system would automatically compute a diagram showing the assignment of resources (including human resources) to the different tasks. The system also returns precise starting and end dates in such a way that the use of resources was optimised (minimising idle time). Our system includes, by default, several categories of guilds and machinery that are usually involved in construction sites, but new types of *resources* can be easily added.

In order to assess the usefulness of our system, we asked a construction engineer to provide us with a real construction project, where sensitive data were anonymised, developed in the United Kingdom. Specifically, the project implemented the construction of a water treatment works with two essential goals: treat waters by introducing Ultra Violet Disinfection and reduce plumbosolvency in the network by installing a sodium

*Corresponding author.

dihydrogen orthophosphate dosing equipment. Our tool was able to produce a solution similar in time and cost to the one used in the real project.

We were satisfied with our system but we wanted to *test* it in a real environment. Therefore, we contacted a construction engineer and asked him to evaluate the usefulness of our system with some of his projects. He gave us a set of recommendations and, after implementing them, we obtained a complete new system: $\mathcal{SINPA}$.[1] Next, we enumerate the main recommendations given by the expert and show how they are implemented to represent clear improvements with respect to our previous system (Camacho et al., 2018).

The first group of recommendations concerned different functionalities that should be enhanced to improve usability and facilitate certain routine steps.

- We have enriched the colour pallet including a new set of tones to provide each task with a specific colour. With this feature, the main idea is to group the diverse guilds that compose the construction – such as joiners, plumbers and bricklayers – with a different colour. In addition, all the tasks which are carried out by a specific guild will have the same colour. This feature makes easier the identification of the tasks performed by the same guild, facilitating a more fluid management.

- We have included a mechanism to automatically synchronise the holidays with the system calendar. With this update, public, bank and observance holidays for different countries, states and regions are taken into account in the construction site planning, following the ISO 3166-2.

A second group of recommendations pointed out additions that would (notoriously) improve the features provided by the system.

- New information and properties have been included in the system to solve more realistic and complex problems. Among others, we can mention the number of workers associated to each task, the possibility of including extra-workers to reduce the duration of the task, the total cost of workers per day and the maximum workers that can be associated to a task.

- $\mathcal{SINPA}$ can now manage the incorporation of external resources in the resolution process. That is, it allows users to add all the equipment used in the construction site, such as bulldozers, excavators, road rollers and dump trucks, to improve the overall time and cost of the construction project using resources constraints. Internally, this new information has been included in the constraint solving process.

- $\mathcal{SINPA}$ currently provides a very useful feature: if some of the conditions are changed (e.g. we have an unexpected delay in the middle of the project) then new solutions are automatically computed and displayed to the user through the GUI.

- $\mathcal{SINPA}$ includes a new report module to analyse the results obtained with the optimisations (see below) that have been included in the system. This report contains useful information so that the user can easily analyse the results and obtain significant improvements with respect to the original plan.

A final recommendation of our expert suggested to extend our system with *recommendation features*. Specifically, the user might provide an initial design of a construction site and the system should analyse whether the resources were used in an optimal way. We have included three optimisation strategies: *most suitable budget optimisation*, *minimum time optimisation* and *minimum cost optimisation*. $\mathcal{SINPA}$ provides the optimal solution for each different strategy, together with a textual explanation focused on effectiveness aspects (Tintarev and Masthoff, 2015). That is, our system now allows users to understand the reasoning behind a recommendation and helps them to make better decisions. The explanation is presented on a report that is based on the *knowledge and utility style* (Aggarwal, 2016) and facilitates the task of fully interacting with the system, selecting and adapting one of the proposed solutions to its requirements. Next we briefly sketch the type of recommendations that users receive. Suppose that a user specifies for a certain task what needs to be done, the amount of time needed to complete it when using certain resources (in particular, human resources) and the materials and resources that are necessary. If the faster completion of this task depends on the amount of resources[2] and $\mathcal{SINPA}$ detects that there are idling resources of this type, then it will suggest to change the initial estimation. If no idling resources are found, then $\mathcal{SINPA}$ will suggest to obtain more resources of this type and will provide the gaining in terms of completion time and the cost of adding these resources. Obviously, the users will make the final decision but they will have useful information to decide in the trade-off between increasing the cost and reducing the time.

A fundamental characteristic of $\mathcal{SINPA}$, although completely transparent to the final users, is that the computation of different construction plans strongly relies on finding a solution to a set of constraints. Constraint Programming is based on logic programming and graph theory and it is suited to find solutions to combinatorial optimisation problems (Apt, 2003; Rossi et al., 2006). A constraint program uses decision variables, constraints, and objectives that can be minimised or maximised. Next, a constraint solver is used to assign values to all variables that satisfy all constraints. Most current programming languages, for example, C++, Java, and Python, contain solver libraries. Taking as initial step the theoretical framework presented in previous work (Correas et al., 2018), we will use the

[1] Our tool can be downloaded from our web site `http://antares.sip.ucm.es/tools/expertSystems/SINPA.7z`

[2] For example, if we are painting a small room then we probably do not gain much by having more than one painter. However, if we have a very big room, then we will have an almost linear speed-up if we increase the number of painters.

Choco library for Java (Prud'homme et al., 2017) to organise the set of tasks conforming a construction plan.

Concerning related work, Building Information Modelling (BIM) (Eastman et al., 2011; Quirk, 2012; Wang and Meng, 2019) has been very popular during the last 10 years. In particular, BIM easily integrates IoT (Dave et al., 2018; Tang et al., 2019). Different approaches using multi-agent systems (Sawhney et al., 2003; Molinero and Núñez, 2011; Taillandier et al., 2015) and ontologies (Zhong et al., 2015; Niknam and Karshenas, 2015) have been used to plan the work in construction sites. Although these approaches rely on some kind of formalism, our work takes a different point of view because we advocate that there is room for a more formal approach like the one underlying the development and behaviour of $\mathcal{SINPA}$ and its predecessor (Camacho et al., 2018). Expert and recommender systems have been used in Construction before (Lin et al., 2012; Faghihi et al., 2014; Martínez-Rojas et al., 2016) but, to the best of our knowledge, $\mathcal{SINPA}$ is the first such system devoted to planning and scheduling. In addition, our tool uses a novel formal framework based on Constraint Programming, which improves the performance and quality of the results. $\mathcal{SINPA}$ includes temporal and resource constraints. With respect to the first group, we have included time optimisation and precedence constraints similar to the Gantt softwares *Gantt for Ext JS*[3] and *MindView*[4]. With respect to resource constraints, we ensure the non-overlapping of tasks that use the same resource, similar to MS Project[5]. An obvious advantage of our tool is that it is freely available, while including the most useful features of the previously mentioned tools. We also plan to upgrade the tool to include new work on constraint solving. The inclusion of these constraints in $\mathcal{SINPA}$ follows the techniques of *constraint satisfaction in scheduling* from the Artificial Intelligence point of view (Barták et al., 2010). $\mathcal{SINPA}$ uses inference rules in a similar way to previous work (Correas et al., 2018). In addition, we also use its operational semantics to define the *recommendation features* of $\mathcal{SINPA}$.

The rest of the paper is organised as follows. In Section 2 we introduce basic concepts on constraints and Constraint Programming. In Section 3 we introduce the formal framework supported by our tool. In Section 4 we give a simple case study to show how the previous theoretical framework works. In Section 5 we present the main features of $\mathcal{SINPA}$. In Section 6, we apply $\mathcal{SINPA}$ to analyse a case study based on a real construction site plan. Finally, in Section 7, we present our conclusions and some lines for future work.

2. A brief introduction to Constraint Programming

In Constraint Programming (Rossi et al., 2006), relationships between variables are expressed in the form of constraints that involve variables. The goal is to find values for the variables that satisfy the constraints. *Finite domains* are the most successful constraint programming domains where variables range over a finite set of values. A *Constraints Satisfaction Problem* (Rossi et al., 2006) (CSP) consists of a finite set of variables, a domain of values for each variable and a set of constraints that delimit the combination of values that the variables can take.

Definition 1. *A* Constraint Satisfaction Problem *is defined as a triple* (X, D, C)*, where*

- $X = (x_1, \ldots, x_n)$ *is a tuple of n variables. Abusing the notation, we will write* $x \in X$ *to denote that* x *is one of the variables of the tuple* X *and* $X' \subseteq X$ *to denote that the variables belonging to* X' *appear in the tuple* X*.*
- $D = D_1 \times D_2 \cdots \times D_n$*, where for each* $1 \leq i \leq n$ *we have that* $x_i \in D_i$*, that is,* D_i *is the domain of* x_i*.*
- $C = (C_1, \ldots, C_m)$ *is a tuple of constraints. Each constraint* C_j *is a pair* (R_j, X_j)*, where* $X_j \subseteq X$ *is the subset of variables involved in the constraint and* R_j *is a subset of the Cartesian product of the domains of these variables.*

An assignment *of the variables is a function from a subset of variables to a particular set of values in the corresponding subset of domains. An assignment satisfies a constraint* (R, X') *if the values assigned to the variables belonging to* X' *satisfy the relation* R*. An assignment is a* solution *of a CSP if it does not violate any of the constraints and includes all variables. If a CSP has at least one solution, then the CSP is* satisfiable*; otherwise, the CSP is* inconsistent*.*

If a given problem is modeled with constraints, a specific constraint solver is used to solve it, that is, to find a value for each variable in order to satisfy all the constraints of the problem. A *constraint solver* is a decision procedure that checks whether a constraint, or set of constraints, can be satisfied. The variables in a *finite domain solver* range over a finite set of values, usually integer. This kind of solvers performs a systematic exploration of the search space until either a solution is found or the whole tree is explored without success. In order to reduce the search space, *constraint propagation* techniques are applied to reduce the domains of variables (Choi et al., 2006). The application of constraint propagation usually implies that either the first solution is found more quickly or the unsatisfiability of the problem is proven before. Constraint propagation is combined with *backtracking*, a general algorithm that maintains a partial assignment of the variables for finding all solutions. For instance, consider a CSP with the finite domain variables x, y, z and t, each with a finite initial domain $\{1, 2, 3\}$. The notation $x :: [\underline{x} \,..\, \overline{x}]$ denotes that x lies within integer lower and upper bounds. In our example, we have $x, y, z, t :: [1 \,..\, 3]$. Next, the constraints $x + y < z + t$, $x < z$, and $y < t$ are added to the problem. Constraint propagation restricts the values of variables in the following way $x, y :: [1 \,..\, 2]$ and $z, t :: [2 \,..\, 3]$. Subsequently, different solutions can be found, via backtracking, where all the variables are assigned a value. One solution is $x = 1$, $y = 2$, $z = 2$, $t = 3$.

Common programming languages, including C++, Java and Python, have been extended to include constraints. In the development of $\mathcal{SINPA}$ we have used the Java library Choco Solver (Prud'homme et al., 2017).

[3] https://www.bryntum.com/products/gantt-for-extjs/
[4] https://www.matchware.com/gantt-chart-software
[5] https://project.microsoft.com

3. A formal representation of $\mathcal{SINPA}$

In this section we introduce the theoretical framework supported by our tool. In order to properly define the constraint system underlying $\mathcal{SINPA}$, we will present its formal representation. This will be given as a *tasks states transformation system*. This technique is based on an abstract technique applied to transformation systems (Estévez-Martín et al., 2009) and represents the changes of states of the tasks. First of all, it is necessary to define the state of a task as a set of characteristics encapsulated in a tuple.

Definition 2. *A* task *is a tuple* $\tau \in FD \times FD \times \mathbb{R} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{R} \times \mathbb{N}$ *where the first two components are finite domain variables, marking the start and end of the task, and are denoted by* $\tau.\alpha$ *and* $\tau.\omega$*. The next component is a variable whose domain is the real numbers and represents the cost per day (denoted by* $\tau.$`Cost`*). The next three values are natural numbers and represent, respectively, the duration, number of workers, and the minimum number of days required to complete the task (denoted by* $\tau.$`Duration`*,* $\tau.$`Workers` *and* $\tau.$`MinDays`*). The last two parameters represent whether there are fixed costs and the number of workers needed to reduce the task in one day (denoted by* $\tau.$`FixCost` *and* $\tau.$`WxD`*).*

Usually, we will omit the name of the task τ when referring to its components if it can be inferred from the environment (e.g. we will write α instead of $\tau.\alpha$). In the previous definition, the finite domain variables, α and ω, have initial finite domains of integer numbers ranging from 1 to $\mathcal{N}$ (that is, $\alpha, \omega :: [1 \,..\, \mathcal{N}]$), where $\mathcal{N}$ is the time needed to finish all the tasks if they were sequentially executed. Therefore, $\mathcal{N}$ is an upper bound on the time needed to complete the project. `Cost` indicates the amount of money a worker devoted to the task earns per day while `Duration` is the estimated time to complete the task when we have a number of workers equal to `Workers`. Some tasks have a fixed cost (`FixCost`) independent of the duration of the task. After planning tasks, our tool computes what parameters need to be changed to improve either the total time or cost of the project. Therefore, we need to know the number of workers necessary to reduce one task in one day (this is given by `WxD`). This parameter also indicates the number of workers left over if the task duration is increased in one day. Finally, `MinDays` denotes the minimum time needed to complete the task and it is used as a lower bound in the optimisation process.

The operational semantics of this transformation system is defined by means of *tuple transformation rules* in a similar way to our previous work (Correas et al., 2018). In this paper we only consider unary and binary rules (rules affecting, respectively, one and two tasks). A *tasks states transformation system* is specified by means of a set of tuple transformation rules that apply to a tuple of tasks.

Definition 3. *Given a task* τ*, a* unary tuple transformation rule *is an inference rule RL*

$$\frac{Conditions}{\tau \Rightarrow_{RL} \tau'}$$

where the values of τ' *are computed from* τ*. The upper part of the rule is a conditional part. If the conditions hold, then a tuple transformation step can be applied and the state of the task is modified accordingly.*

Similarly, given two tasks τ_1 *and* τ_2*, a* binary tuple transformation rule *is an inference rule RL*

$$\frac{Conditions}{\tau_1, \tau_2 \Rightarrow_{RL} \tau'_1, \tau'_2}$$

where the values of τ'_1 *and* τ'_2 *are computed from* τ_1 *and* τ_2*.*

Let $(\tau_1, \ldots, \tau_n)$ *be a tuple of n tasks. We write* $(\tau_1, \ldots, \tau_n) \Rightarrow_{RL} (\tau'_1, \ldots, \tau'_n)$ *if one of the following two conditions hold:*

- *RL is a unary rule applied to a task* τ_i*, which evolves to* τ'_i*, and for all* $1 \leq j \leq n$*, with* $i \neq j$*, we have* $\tau_j = \tau'_j$*.*
- *RL is a binary rule applied to two tasks* τ_{i1} *and* τ_{i2}*, which respectively evolve to* τ'_{i1} *and* τ'_{i2}*, and for all* $1 \leq j \leq n$*, with* $j \notin \{i1, i2\}$*, we have* $\tau_j = \tau'_j$*.*

We consider a special application of an inference rule RL. If we have a derivation $(\tau_1, \ldots, \tau_n) \Rightarrow_{RL} \blacksquare$*, then we have a* failing transformation step*, that is, we have reached a state where a failure is returned to the system and this forces the exploration of alternative computations via backtracking.*

We can extend the notion of tuple transformation to the successive application of several rules in the natural way. We write

$$(\tau_1, \ldots, \tau_n) \Rightarrow^{+}_{RL} (\tau'_1, \ldots, \tau'_n)$$

if there exists $i > 0$ *such that for all* $0 \leq j \leq i - 1$ *we have* $(\eta_1^j, \ldots, \eta_n^j) \Rightarrow_{RL} (\eta_1^{j+1}, \ldots, \eta_n^{j+1})$*, with* $(\tau_1, \ldots, \tau_n) = (\eta_1^0, \ldots, \eta_n^0)$ *and* $(\tau'_1, \ldots, \tau'_n) = (\eta_1^i, \ldots, \eta_n^i)$*. We write*

$$(\tau_1, \ldots, \tau_n) \Rightarrow^{+} (\tau'_1, \ldots, \tau'_n)$$

if there exists $i > 0$ *and rules* $RL_1, \ldots, RL_i$ *such that for all* $0 \leq j \leq i - 1$ *we have* $(\eta_1^j, \ldots, \eta_n^j) \Rightarrow_{RL_{j+1}} (\eta_1^{j+1}, \ldots, \eta_n^{j+1})$*, with* $(\tau_1, \ldots, \tau_n) = (\eta_1^0, \ldots, \eta_n^0)$ *and* $(\tau'_1, \ldots, \tau'_n) = (\eta_1^i, \ldots, \eta_n^i)$*.*

We say that a tuple of tasks is *irreducible* if a different state cannot be reached. Note that the concepts of irreducible and failure denote completely different situations. Irreducible means that a final state has been reached and it is not possible to progress, that is, it denotes a successful termination conducting to a solution. Failure means that the current configuration is not feasible and, therefore, a solution cannot be found by following the path that took us to this configuration.

Definition 4. *Let* $(\tau_1, \ldots, \tau_n)$ *be a tuple of tasks. We say that this tuple is irreducible, denoted by* $\mathtt{IRR}(\tau_1, \ldots, \tau_n)$*, if there do not exist a tuple transformation rule RL and a tuple of tasks* $(\tau'_1, \ldots, \tau'_n)$ *such that* $(\tau_1, \ldots, \tau_n) \Rightarrow_{RL} (\tau'_1, \ldots, \tau'_n)$*.*

In our system, we have two groups of tuple transformation rules. We denote by $\mathcal{RL}_C$ the set of tuple transformation rules related to constraints and defined in Figure 1 (that is, $\mathcal{RL}_C = \{RL1, RL2, RL3\}$) and by $\mathcal{RL}_O$ the set of tuple transformation

rules related to optimisation and defined in Figure 2 (that is, $\mathcal{RL}_O = \{RL4, RL5, RL6\}$). In these rules, we define the bounds of the finite domain variables by using underline and overline typography, that is, we use the notation $\alpha :: [\underline{\alpha} \,..\, \overline{\alpha}]$ and $\omega :: [\underline{\omega} \,..\, \overline{\omega}]$.

Definition 5. *Let us consider a tuple of tasks $(\tau_1, \ldots, \tau_n)$ constituting the problem that we want to solve. The constraints system underlying the behaviour of $\mathcal{SINPA}$ is given by the following algorithm.*

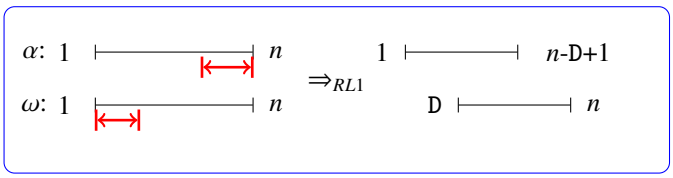
1. *Add rules related to constraints (see Figure 1).*
   - *For each task, add the* `temporal constraints` *to the solver (see Section 3.1 - RL1).*
   - *For each pair of tasks with a precedence relation (that is, one task must be completed before the beginning of the other), add the* `precedence constraints` *to the solver (see Section 3.2 - RL2).*
   - *For each pair of tasks sharing a resource, add a* `resource sharing constraint` *to the solver (see Section 3.3 - RL3).*
2. *Add rule(s) related to the chosen optimisation (see Figure 2). Choose only one.*
   - *Most suitable* `budget optimisation`*. For each task, add the RL4 rule to the solver.*
   - *Minimum* `time optimisation`*. For each task, add the two RL5 rules to the solver.*
   - *Minimum* `cost optimisation`*. For each task, add the RL6 rule to the solver.*
3. *Apply the tuple transformation rules previously added to the solver until we reach a tuple of tasks $(\tau'_1, \ldots, \tau'_n)$ such that $(\tau_1, \ldots, \tau_n) \Rightarrow^+ (\tau'_1, \ldots, \tau'_n)$ and* `IRR`$(\tau'_1, \ldots, \tau'_n)$.

Next we briefly describe how a set of tasks, representing the phases associated with a construction plan, is evolving after applying these rules. Internally, $\mathcal{SINPA}$ will follow this process. Let us emphasize that users of our tool do not need to know any of these *formalisation* details: a user-friendly interface will ask for the basic information defining the tasks and $\mathcal{SINPA}$ will automatically produce and solve the associated constraints system.

3.1. Temporal constraints

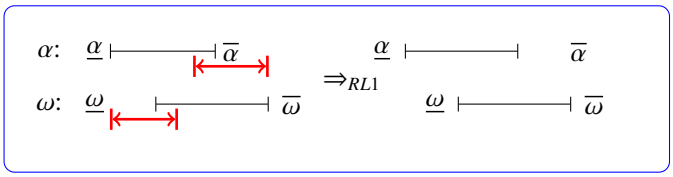
First of all, temporal constraints are posted in the solver, that is, constraints that relate the start and end of a task to its duration ($\omega - \alpha =$ `Duration` $- 1$). The behaviour of these constraints on tasks, that is, the changes that the lower and upper bound of the finite domain variables α and ω have, is specified by the $RL1$ rule (see Figure 1). We apply $RL1$ once to all the tasks $\tau_1, \ldots, \tau_n$. Intuitively, $RL1$ is applied to each task to eliminate non-feasible days due to the duration of each task. For example, if a task lasts three days, then it can neither start in the last two days of the project nor finish during the first two ones.

The application of the rule $RL1$ to a task τ narrows the lower and upper bounds of α and ω if the conditional part of the rule holds. Initially the α and ω domains ($\alpha :: [1 \,..\, \mathcal{N}]$ and $\omega :: [1 \,..\, \mathcal{N}]$) are defined by finite ranges of integer values included between 1 and $\mathcal{N}$. Remind that $\mathcal{N}$ is an upper bound on the time needed to complete the whole construction plan. The first application of this rule to this task narrows the initial domains to $\alpha :: [1 \,..\, \mathcal{N} -$ `Duration` $+ 1]$ and $\omega :: [1 +$ `Duration` $- 1 \,..\, \mathcal{N}]$, respectively. Further applications of this rule will narrow even more these domains. Below, we graphically present the result of applying the $RL1$ rule to α and ω for a given task. The `Duration` corresponding to this task (denoted simply by `D`) is drawn with a double-headed arrow.



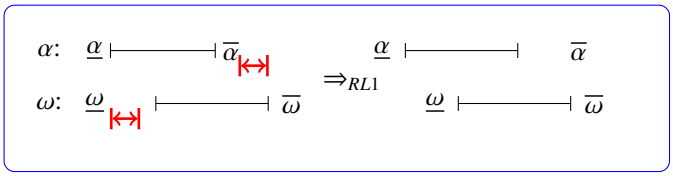
Application of $RL1$ to a task τ.

Note that the new upper bound of α is given by the expression $\min(\overline{\alpha}, \overline{\omega} -$ `Duration` $+ 1)$, that is, the minimum between the previous upper bound and a new upper bound computed by taking into account the `Duration`. This distinction is necessary because it will produce that our tool advises users to increase the number of days associated with the task and, consequently, temporary constraints are reevaluated on all tasks.



Possible new application of $RL1$ if `Duration` increases.

However, if our tool suggests to decrease the number of days, then it is not necessary to re-evaluate the temporal constraints because the bounds of the finite domain variables do not change, as shown below.



Unnecessary application of $RL1$ if `Duration` decreases.

The second tuple transformation associated with the rule $RL1$ indicates that the task cannot be completed in the established duration. If this situation happens, then the computation stops and the fault is returned by marking that this is a failing transformation step (indicated by the ■ symbol).

3.2. Precedence constraints

After dealing with temporal constraints, our tool processes the precedence constraints defined by the user. A precedence

In the bottom part of the rules we only show how the α and ω components of the tasks are affected because the rest of components are not affected by the application of these rules.

*RL*1: `Temporal constraints`

$$\frac{\overline{\omega} - \underline{\alpha} \geq \texttt{Duration}}{\alpha :: [\underline{\alpha} \,..\, \overline{\alpha}], \omega :: [\underline{\omega} \,..\, \overline{\omega}] \quad \Rightarrow_{RL1} \quad \alpha :: [\underline{\alpha} \,..\, \min(\overline{\alpha}, \overline{\omega} - \texttt{Duration} + 1)], \omega :: [\max(\underline{\omega}, \underline{\alpha} + \texttt{Duration} - 1) \,..\, \overline{\omega}]}$$

$$\frac{\overline{\omega} - \underline{\alpha} < \texttt{Duration}}{\alpha :: [\underline{\alpha} \,..\, \overline{\alpha}], \omega :: [\underline{\omega} \,..\, \overline{\omega}] \quad \Rightarrow_{RL1} \quad \blacksquare}$$

*RL*2: `Precedence` $(\tau_i \ll \tau_j)$

$$\frac{\overline{\alpha_j} > \underline{\omega_i}}{\alpha_i :: [\underline{\alpha_i} \,..\, \overline{\alpha_i}], \omega_i :: [\underline{\omega_i} \,..\, \overline{\omega_i}], \alpha_j :: [\underline{\alpha_j} \,..\, \overline{\alpha_j}], \omega_j :: [\underline{\omega_j} \,..\, \overline{\omega_j}] \quad \Rightarrow_{RL2} \quad \omega_i :: [\underline{\omega_i} \,..\, \min\{\overline{\omega_i}, \overline{\alpha_j} - 1\}], \alpha_j :: [\max\{\underline{\omega_i} + 1, \underline{\alpha_j}\} \,..\, \overline{\alpha_j}]}$$

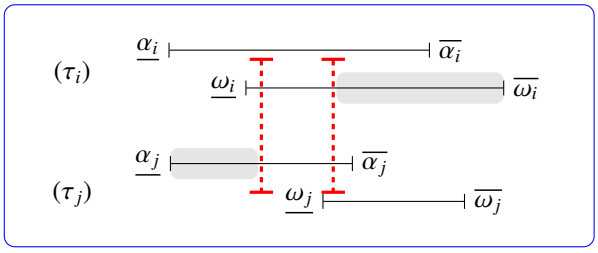
$$\frac{\overline{\alpha_j} \leq \underline{\omega_i}}{\alpha_i :: [\underline{\alpha_i} \,..\, \overline{\alpha_i}], \omega_i :: [\underline{\omega_i} \,..\, \overline{\omega_i}], \alpha_j :: [\underline{\alpha_j} \,..\, \overline{\alpha_j}], \omega_j :: [\underline{\omega_j} \,..\, \overline{\omega_j}] \quad \Rightarrow_{RL2} \quad \blacksquare}$$

*RL*3: `Resource sharing`

$$\frac{((\overline{\omega_j} > \underline{\alpha_i}) \wedge (\texttt{Duration}_i + \texttt{Duration}_j < \overline{\omega_j} - \underline{\alpha_i})) \;\wedge\; ((\overline{\omega_i} < \underline{\alpha_j}) \wedge (\texttt{Duration}_i + \texttt{Duration}_j < \overline{\omega_i} - \underline{\alpha_j}))}{\alpha_i :: [\underline{\alpha_i} \,..\, \overline{\alpha_i}], \omega_i :: [\underline{\omega_i} \,..\, \overline{\omega_i}], \alpha_j :: [\underline{\alpha_j} \,..\, \overline{\alpha_j}], \omega_j :: [\underline{\omega_j} \,..\, \overline{\omega_j}] \quad \Rightarrow_{RL3} \quad \blacksquare}$$

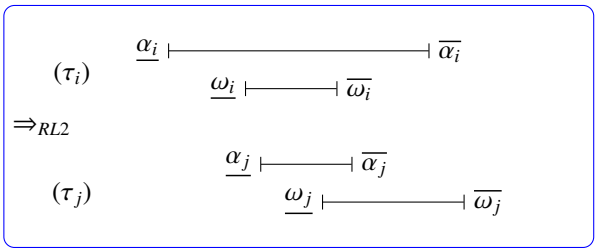
Figure 1: $\mathcal{RL}_C$: Tuple transformation rules related to constraints.

constraint reflects the order between two tasks: $\tau_i \ll \tau_j$ denotes that task τ_i must end before task τ_j begins. Therefore, we have to post the constraint $\omega_i < \alpha_j$ in the solver. The behaviour of this constraint with respect to the finite domain variables is specified in rule *RL*2. Next, we give a graphical representation to show how the domains are narrowed if the rule *RL*2 can be applied. We consider two tasks that have reached states where the rule associated with the condition $\tau_i \ll \tau_j$ can be applied.



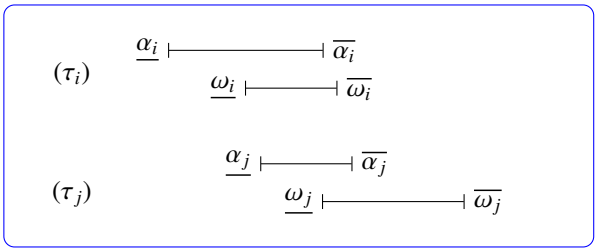
Application of *RL*2. Part I.

We have that $\tau_i \ll \tau_j$ imposes that τ_i is restricted to be completed before τ_j begins. Therefore, all values greater than $\overline{\alpha_j} - 1$ are removed from the domain of ω_i. Symmetrically, all values less than $\underline{\omega_i} + 1$ are removed from the domain of α_j. Then, we obtain the following configuration:



Application of *RL*2. Part II.

Finally, for consistency of the values associated with τ_i, we need to request $\overline{\alpha_i} < \overline{\omega_i}$.

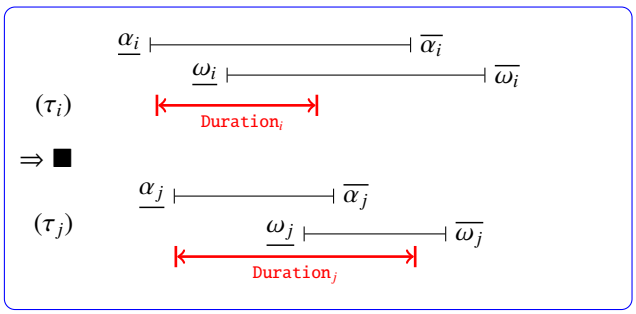


Propagation of the application of *RL*2.

Similar to *RL*1, the second tuple transformation rule of *RL*2 indicates that there is not enough time to impose the precedence constraint and this computation stops with a failure.

3.3. Resource sharing constraints

The last group of constraints deals with *non-overlapping restrictions*, that is, if two tasks τ_i and τ_j need the same resource, for example a crane machine, and there is only one unit of it, then they cannot be executed at the same time. Therefore, resource sharing is defined as $\tau_i \ll \tau_j \vee \tau_j \ll \tau_i$ and this is reflected by posting the $(\omega_i < \alpha_j \vee \alpha_j < \omega_i)$ constraint into the solver. Since this constraint is given by a a disjunction, the domains of the variables are not narrowed. In contrast, the constraint remains in the solver and when the variables of the tasks are instantiated, this constraint imposes that the execution of these tasks have no common time values. During the computation process, it will be checked whether it is not feasible that two tasks can share a resource due to time limitations. Rule *RL*3 checks that the sum of the duration of the tasks that share the same resource is greater than the available time, that is, whether $\texttt{Duration}_i + \texttt{Duration}_j < \overline{\omega_j} - \underline{\alpha_i}$ and $\texttt{Duration}_i + \texttt{Duration}_j < \overline{\omega_i} - \underline{\alpha_j}$. If this is the case, then planning is not possible and we reach the symbol ■. A graphical representation of this situation is shown below.



Application of *RL*3

3.4. Optimisation

Once all the constraints are established, and the values of the bounds of the tasks are narrowed, the user can demand from the system an *optimisation advice* with respect to different parameters.

- *Most suitable budget optimisation* (MSBO) with the goal of reducing the number of days as long as the total cost is lower than a maximum budget.
- *Minimum cost optimisation* (MCO) with the goal of reducing the cost by increasing/reducing the number of days.
- *Minimum time optimisation* (MTO) with the goal of reducing the number of days even if the cost is higher.

The tuple transformation rules given in Figure 2 are specific rules for the previous optimisations. *RL*4 checks whether the cost associated with the reduction of a task τ_i by one day (that is, $\texttt{Duration}_i - 1$), assuming that this reduction is allowed (that is, $\texttt{Duration}_i > \texttt{MinDays}_i$) and consequently increasing the number of workers (that is, $\texttt{Workers}_i + \texttt{WxD}_i$) plus the current cost of the other tasks does not exceed a certain budget given by the planner. If this is satisfied, then the parameters `Duration` and `Workers` are modified accordingly. Note that this rule modifies the `Duration` value. Therefore, *RL*1 is re-evaluated and, due to the side effects of the modification of the `Duration` and the variables α and ω, *RL*2 and *RL*3 are also evaluated again. That is, each time that the system recalculates the parameters, the constraint solver computes the satisfiability of the constraints and *constraint propagation* (Apt, 1999) reduces the domains of variables and, correspondingly, the search space. In our case, the propagation removes the extreme interval values that cannot be satisfied, forcing the so-called *bound consistency* (Choi et al., 2006).

The second optimisation focuses on the cost. In order to reduce the cost, *RL*5 checks whether reducing/increasing a day (that is, $\texttt{Duration} - 1/\texttt{Duration} + 1$) and consequently increasing/reducing the number of workers (that is, $\texttt{Workers} + \texttt{WxD}/\texttt{Workers} - \texttt{WxD}$), produces that the cost is lower. As in the previous rule, this optimisation produces changes that cause the reevaluation of rules related to constraints.

The third optimisation, *RL*6, has as goal to reduce the time of the project. For this, the duration of the tasks is reduced to the limit of `MinDays`.

4. Theoretical framework: a case study

We illustrate the behaviour of our tuple transformation system by using a (relatively) simple case study. Let us consider a project with six tasks $(\tau_1, \ldots, \tau_6)$. The initial planning of the project is a data estimation made by the project manager and must be introduced in the application. In this case study, the data of the next table represents the initial planning of our example. These values correspond to the numeric fields of the tuples (_, _, `Cost`, `Duration`, `Workers`, `MinDays`, `FixCost`, `WxD`) (see Definition 2). Note that the first two components are the finite domain variables marking the start and the end of the task. For example, data planning on task τ_1 assigns a duration of 8 days with 4 workers, the cost is £ 100 per worker and day plus £ 11 of fixed costs, and this task cannot be done in less than 2 days.

Task	Worker Cost	Dur.	Total Work.	Min days	Fix cost	Incr. WxD
τ_1	100	8	4	2	11	2
τ_2	150	3	3	1	12	1
τ_3	125	9	3	3	13	2
τ_4	90	6	1	3	14	1
τ_5	70	5	4	2	15	1
τ_6	80	4	2	2	16	0

First, the domains of the initial and final variables of all the tasks are established as a range between 1 and the upper bound estimation of the total duration of the project. This number is given by the sum of all the duration tasks. The following tuple of tuples represents the initial stage of the tuple transformation

In the bottom part of the rules we only show how the Duration and Workers components of the tasks are affected because the rest of components are not changed by the application of these rules.

*RL*4: Most suitable budget

$$\frac{(\texttt{Duration}_i > \texttt{MinDays}_i) \ \wedge \ (\sum_{j=1;j\neq i}^{n}(\texttt{Duration}_j * \texttt{Cost}_j * \texttt{Workers}_j) + (\texttt{Duration}_\texttt{i} - 1) * \texttt{Cost}_\texttt{i} * (\texttt{Workers}_\texttt{i} + \texttt{WxD}_\texttt{i})) < \texttt{Max.Budget})}{\texttt{Duration}_i, \texttt{Workers}_i \Rightarrow_{RL4} \texttt{Duration}_i - 1, \texttt{Workers}_i + \texttt{WxD}_i}$$

*RL*5: Minimum cost

$$\frac{(\texttt{Duration} > \texttt{MinDays}) \wedge (\texttt{Cost} * \texttt{Duration} * \texttt{Workers} < \texttt{Cost} * (\texttt{Duration} - 1) * (\texttt{Workers} + \texttt{WxD}))}{\texttt{Duration}, \texttt{Workers} \Rightarrow_{RL5} \texttt{Duration} - 1, \texttt{Workers} + \texttt{WxD}}$$

$$\frac{(\texttt{Workers} > \texttt{WxD}) \wedge (\texttt{Cost} * \texttt{Duration} * \texttt{Workers} < \texttt{Cost} * (\texttt{Duration} + 1) * (\texttt{Workers} - \texttt{WxD}))}{\texttt{Duration}, \texttt{Workers} \Rightarrow_{RL5} \texttt{Duration} + 1, \texttt{Workers} - \texttt{WxD}}$$

*RL*6: Minimum time

$$\frac{\texttt{Duration} > \texttt{MinDays}}{\texttt{Duration}, \texttt{Workers} \Rightarrow_{RL6} \texttt{Duration} - 1, \texttt{Workers} + \texttt{WxD}}$$

Figure 2: $\mathcal{RL}_O$: Tuple transformation rules related to optimisation.

system.

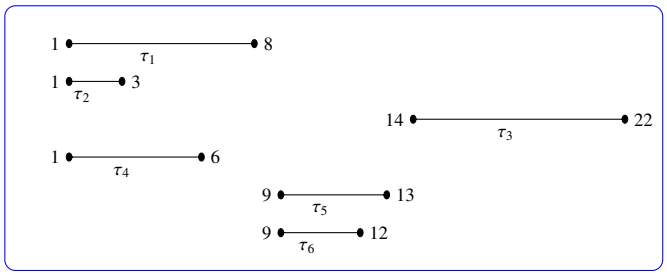
$$\left(\begin{array}{l}(\alpha_1 :: [1 \,..\, 35], \omega_1 :: [1 \,..\, 35], 100, 8, 4, 2, 11, 2),\\ (\alpha_2 :: [1 \,..\, 35], \omega_2 :: [1 \,..\, 35], 150, 3, 3, 1, 12, 1),\\ (\alpha_3 :: [1 \,..\, 35], \omega_3 :: [1 \,..\, 35], 125, 9, 3, 3, 13, 2),\\ (\alpha_4 :: [1 \,..\, 35], \omega_4 :: [1 \,..\, 35], 90, 6, 1, 3, 14, 1),\\ (\alpha_5 :: [1 \,..\, 35], \omega_5 :: [1 \,..\, 35], 70, 5, 4, 2, 15, 1),\\ (\alpha_6 :: [1 \,..\, 35], \omega_6 :: [1 \,..\, 35], 80, 4, 2, 2, 16, 0)\end{array}\right)$$

The first rule to be applied is *RL*1. The goal of this application is to set up the duration of each task. *RL*1 reduces the domains of variables and the search space, removing extreme values that cannot be satisfied. Specifically, the domain of α_1 and ω_1 are reduced to $[1 \,..\, \min(35, 35 - 8 + 1)]$ and $[\max(1, 1 + 8 - 1) \,..\, 35]$, respectively. We have similar reductions for the domains of all the tasks. Therefore, after applying $\Rightarrow_{RL1}$ we obtain

$$\left(\begin{array}{l}(\alpha_1 :: [1 \,..\, 28], \omega_1 :: [8 \,..\, 35], 100, 8, 4, 2, 11, 2),\\ (\alpha_2 :: [1 \,..\, 33], \omega_2 :: [3 \,..\, 35], 150, 3, 3, 1, 12, 1),\\ (\alpha_3 :: [1 \,..\, 27], \omega_3 :: [9 \,..\, 35], 125, 9, 3, 3, 13, 2),\\ (\alpha_4 :: [1 \,..\, 30], \omega_4 :: [6 \,..\, 35], 90, 6, 1, 3, 14, 1),\\ (\alpha_5 :: [1 \,..\, 31], \omega_5 :: [5 \,..\, 35], 70, 5, 4, 2, 15, 1),\\ (\alpha_6 :: [1 \,..\, 32], \omega_6 :: [4 \,..\, 35], 80, 4, 2, 2, 16, 0)\end{array}\right)$$

At this point, the project manager must indicate in the tool the precedence between tasks and they are included in the planning process. Suppose that the precedence between tasks is $\tau_1 \ll \tau_5$, $\tau_1 \ll \tau_6$, $\tau_4 \ll \tau_3$ and $\tau_5 \ll \tau_3$.

The application of the tuple transformation rule *RL*2 to the tasks τ_1 and τ_5 reduces the domain of the finite domain variables ω_1 and α_5 from [8 .. 35] and [1 .. 31] to [8 .. min(35, 31 − 1)] and [max(8 + 1, 1) .. 31], that is [8 .. 30] and [9 .. 31]. The consecutive application of the rest of the precedence restrictions narrows the corresponding domains. At this point, the planning of the tasks after applying $\Rightarrow_{RL2}$ can be graphically represented as follows.



Now, it is possible to add resource sharing constraints and/or to apply an advice optimisation. First, we will show how the values associated with the tasks evolve if we consider different optimisations.

If we want to compute MSBO, that is, if we would like to reduce the number of days as long as the new total cost is lower than a prefixed maximum budget, then we have to apply *RL*4. This optimisation can be applied if for all tasks of the tuple $(\tau_1, \ldots, \tau_6)$ the condition $\texttt{Duration}_i > \texttt{MinDays}_i$ is satisfied and the following condition holds:

$$\begin{array}{c}\sum_{j=1;j\neq i}^{n}(\texttt{Duration}_j * \texttt{Cost}_j * \texttt{Workers}_j) +\\ (\texttt{Duration}_\texttt{i} - 1) * \texttt{Cost}_\texttt{i} * (\texttt{Workers}_\texttt{i} + \texttt{WxD}_\texttt{i})\\ \wedge\\ \texttt{MaximumBudget}\end{array}$$

In our case study, *RL*4 can be applied to all the tasks. In conclusion, after applying this rule several times, that is, $\Rightarrow^{+}_{RL4}$, the `Duration` and `Workers` corresponding to the tasks vary as shown in the following table.

Steps in the most suitable budget optimisation		
Task	Duration	Workers
τ_1	$8 \to 7 \to 6 \to 5 \to 4 \to 3$	$4 \to 6 \to 8 \to 10 \to 12 \to 14$
τ_2	$3 \to 2 \to 1$	$3 \to 4 \to 5$
τ_3	$9 \to 8$	$3 \to 5$
τ_4	$6 \to 5 \to 4 \to 3$	$1 \to 2 \to 3 \to 4$
τ_5	$5 \to 4 \to 3 \to 2$	$4 \to 5 \to 6 \to 7$
τ_6	$4 \to 3 \to 2$	2
Maximum Budget: 13.500		
Budget: 10.586 → 13.261 → 13.291 → 13.211 → 13.011 → 12.411		
Completion time: 22 →20 →18 →16 → 15 → 14		

In order to compute the values after applying MCO, with the goal of reducing the cost by increasing/reducing the number of days, we need to meet the same condition as in rule *RL*4 (`Duration` > `MinDays`) and the next condition:

$$\texttt{Cost} * \texttt{Duration} * \texttt{Workers} \\ \wedge \\ \texttt{Cost} * (\texttt{Duration} + 1) * (\texttt{Workers} - \texttt{WxD})$$

The corresponding variations of the cost and days after $\Rightarrow^{+}_{RL5}$ are shown in the following table.

Steps in the minimum cost optimisations		
Task	Duration	Workers
τ_1	$8 \to 9$	$4 \to 2$
τ_2	$3 \to 2 \to 1$	$3 \to 4 \to 5$
τ_3	$9 \to 10$	$3 \to 1$
τ_4	6	1
τ_5	$5 \to 4 \to 3 \to 2$	$4 \to 5 \to 6 \to 7$
τ_6	$4 \to 3 \to 2$	2
Budget: 10.586 → 6.751 → 6.001 → 5.721		
Completion time: 22 → 23 → 22 → 21		

The last optimisation, MTO, reduces the number of days even if the cost is higher. After applying $\Rightarrow^{+}_{RL6}$, that is, the rule *RL*6 is successively applied several times, we obtain the following values for the tasks.

Steps in the minimum time optimisation		
Task	Duration	Workers
τ_1	$8 \to 7 \to 6 \to 5 \to 4 \to 3 \to 2$	$4 \to 6 \to 8 \to 10 \to 12 \to 14 \to 16$
τ_2	$3 \to 2 \to 1$	$3 \to 4 \to 5$
τ_3	$9 \to 8 \to 7 \to 6 \to 5 \to 4 \to 3$	$3 \to 5 \to 7 \to 9 \to 11 \to 13 \to 15$
τ_4	$6 \to 5 \to 4 \to 3$	$1 \to 2 \to 3 \to 4$
τ_5	$5 \to 4 \to 3 \to 2$	$4 \to 5 \to 6 \to 7$
τ_6	$4 \to 3 \to 2$	2
Budget: 10.586 → 13.421 → 14.416 → 14.961 → 14.886 → 13.911 → 12.036		
Completion time: 22 → 19 → 16 → 13 → 11 → 9 → 7		

Note that in this optimisation, a reduction on the number of days assigned to complete a task does not always imply that the cost of the task increases. This is because the cost of a task is calculated as the product of `Cost`, `Duration` and `Workers`, plus `FixCost`. Therefore, the new total cost of the task will be computed by considering `Duration` − 1 and `Workers` + `WxD` according to *RL*6, that is, we have `WorkerCost` ∗ (`Duration` − 1) ∗ (`Workers` + `WxD`) + `FixCost`.

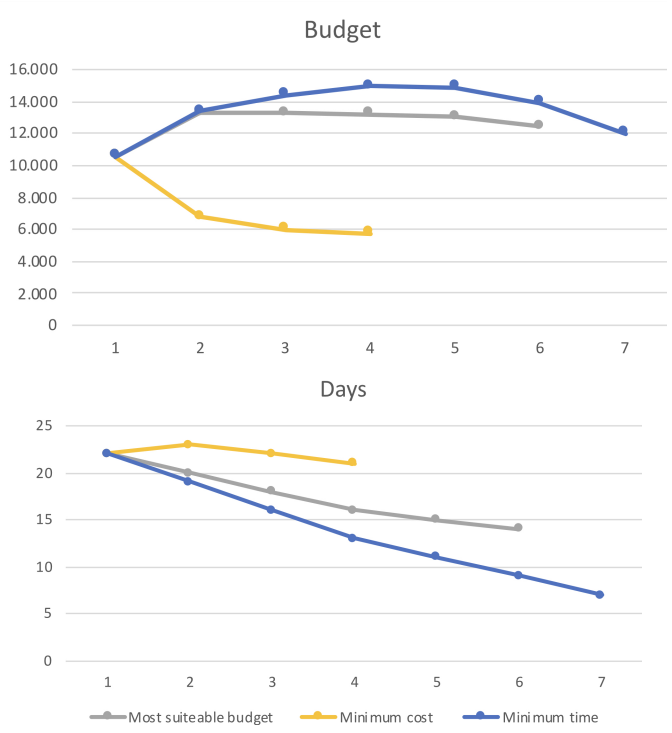
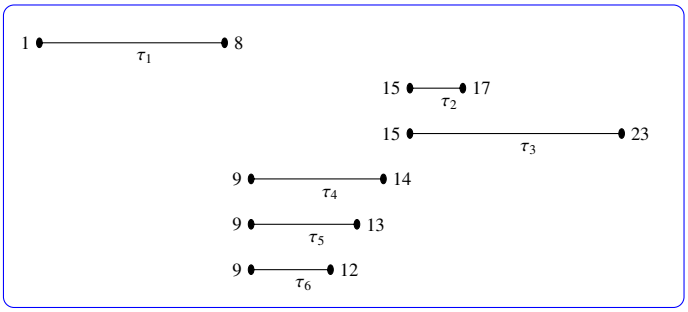


Figure 3: Different optimisations for budget and/or completion time.

Figure 3 compares the three optimisations that are offered to the user so that, if desired, one of them can be applied.

Now, we will show how the values of the variables associated with each task would vary if we consider a restriction: the tasks 1, 2 and 4 share one resource. The immediate effect of this restriction is that these tasks cannot overlap in time. Associated with *RL*3, we have that the arrangement of tasks and the total number of days to complete the project will change as follows.



If we want to apply optimisations after including this restriction, then the previous tables remain the same except for values of the *completion time* that now become:

- Most suitable budget: 23→23→23→23.

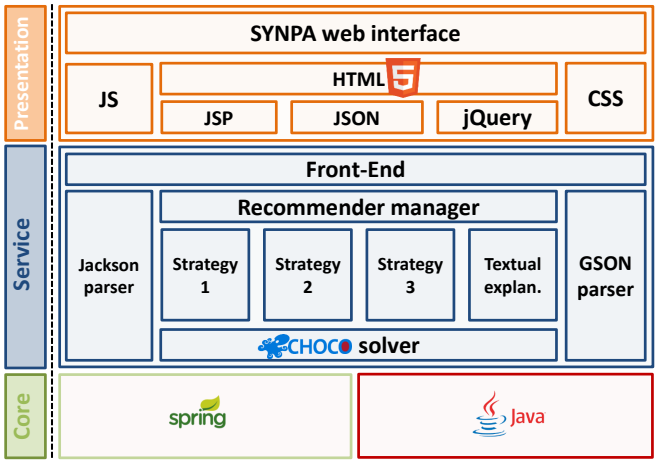


Figure 4: Architecture of $\mathcal{SINPA}$.

- Minimum cost: 23→25→25→25.
- Minimum time: 23→20→17→14→12→10→8.

5. Introducing $\mathcal{SINPA}$: main features

This section provides a description of the tool that supports the theoretical framework presented in Section 3. The main goal of $\mathcal{SINPA}$ is to aid users during the modelling phase of construction sites, trying to optimise its planning. It provides a friendly interface, which allows users to abstract the underlying mathematical theory. For this, our tool includes a layer-based architecture, whose main technologies and components are presented in Section 5.1. The graphical interface used to plan the construction site is described in Section 5.2. Finally, a modelling example for facilitating the usage of the tool is presented in Section 5.3.

5.1. Architecture of $\mathcal{SINPA}$

In order to provide our framework with a high level of flexibility and modularity, $\mathcal{SINPA}$ is based on a layered architecture. This architecture is depicted in Figure 4 and has three layers: *presentation*, *service* and *core*.

The *presentation layer* provides users with a friendly web interface, which allows them to model, analyse and optimise a construction site plan in a simple and easy way. The web interface is built over HTML5 (Hickson, 2015), which promotes important aspects for improving the users experience concerning, among others, usability and security. Moreover, in order to increase the functionality of the framework, the web interface has been integrated with the Spring Tool Suite MVC (Warin, 2015), alongside with several well-known technologies such as CSS (Powell, 2010), JS (Flanagan, 2006) and JSP (Hall, 2001).

The *service layer* is in charge of optimising the construction site planning. The *front-end* orchestrates the communications between the web interface and the system core. For this, the *GSON* (Friesen, 2019a) and *Jackson* (Friesen, 2019b) parsers are used to flatten and unflatten all the data, structures and information needed to properly perform the optimisation process. Then, the recommender manager invokes the constraint solver. We use the choco-solver (Prud'homme et al., 2017) with all the constraints provided by the user and using one of the available strategies to optimise the result. Finally, a report with the results of the optimisation process is provided, including a textual explanation of the recommendation.

The last layer, the *core layer*, supports the services provided by the framework. This layer includes the core of Spring and the Java engine.

5.2. Diagram editor

In order to facilitate the interaction with the constraints engine of the system, we have included a Graphic User Interface (GUI), which is based on the jQuery Gannt editor of twProject[6]. This editor has been extended with several improvements that provide users with an easy-to-use Gantt editor. Our editor allows users of $\mathcal{SINPA}$ to model and optimise a construction site plan without having specific knowledge of the theoretical framework. Our GUI is shown in Figure 5. It consists of three main parts: an *action panel* (label ①), an *editor panel* (label ②) and a *graphic panel* (label ③).

The *action palette* can be found at the top of the GUI and consists of 23 actions and three-panel indicators to model and design the construction site plan (see Table 1 for a complete list). The actions are grouped in four categories, having each of them a specific objective in the design process. The first category is oriented to facilitate the *tasks edition*. The second category focuses on adapting the aspect and management of the graphic panel to the users requirements. The third category provides four mechanisms to configure the constraint solver engine of the tool. Next, we briefly describe these mechanisms. The *holidays editor* allows the planner to specify days when the construction site is not open. The *resources editor* allows the planner to include new resources that are needed for the successful completion of the construction site. The *scheduling strategies editor* can be used to show the results of the constraint solving process according to the available strategies. Finally, the *general configuration editor* provides the planner with a panel to configure the constraint solver engine of the system.

The last category of the action panel consists of a set of elements that show the results of the constraint solving process in a concise and intuitive way. We will briefly describe the report panel because it is the most interesting one. This panel is shown in Figure 6 and has three main parts. The first one illustrates the evolution of the duration and cost of the construction site plan during the constraint solving process (label ①). Each iteration updates the total duration and cost needed to finish the construction. The y-axis of the graph represents the iterations while the z-axis represents two different aspects depending on the value of x. Specifically, z denotes total duration when x is equal to 0 and the total cost when x is equal to 1. The second part shows the *description* of the selected strategy, the *total* cost, the *duration*, the *start* and *end* dates of the planning site (label ②). Then, the report provides a textual explanation that simplifies

[6] https://gantt.twproject.com/

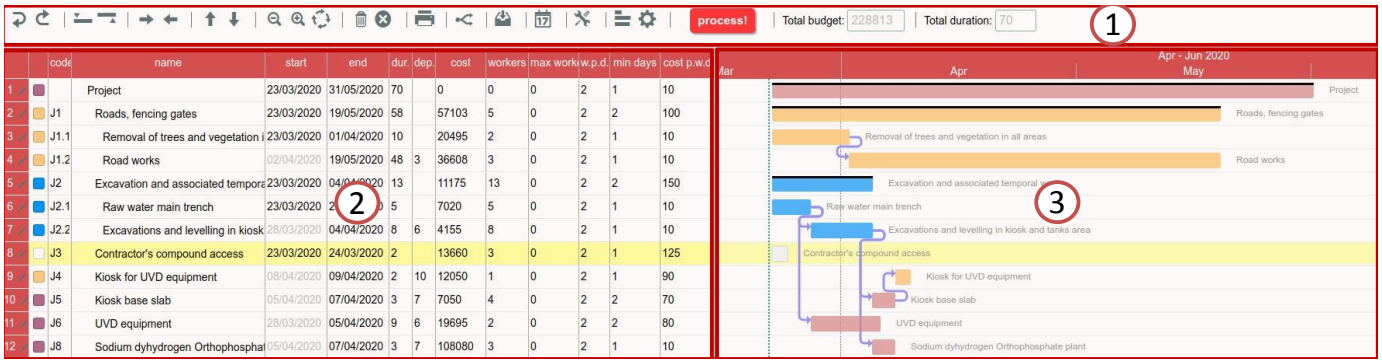


Figure 5: Gantt editor.

the understanding of the recommendation (label ③). This information is represented by following the knowledge/utility-based style (Tintarev and Masthoff, 2015) and shows a relation between the provided solution and the competing options. For this, we have analysed the construction planning domain to extract the specific knowledge and identify the most relevant aspects to effectively provide the explanation. Next, we review some of them, such as the total cost and duration of the planning, duration, cost and number of workers of each task. The recommendation gives the actual differences, concerning the previously mentioned variables, between the values originally provided by the user and the optimised version proposed by $\mathcal{SINPA}$. Listing 1 shows an excerpt of a recommendation explanation, where lines 1 and 2 show the difference of the duration and cost between the proposed solution and the reference planning provided by user. In line 3 we can see the number of tasks that have been adjusted while the corresponding differences are shown in lines 5-20. In order to facilitate the reading and comprehension of the information, these differences are grouped by task. For example, the information corresponding to task number 1 is presented in lines 7-10: line 8 shows the gain in the number of days, line 9 shows the difference in the number of workers (in this case, it should be increased to obtain the previous gain) and line 10 shows the modification concerning the suggested starting date between the reference and the planning proposed by our tool.

```
1  This configuration is 35 days faster
2  This configuration is 20605 $ cheaper
3  Number of tasks with changes: 7
4
5   The task#0 has a difference (1)
6    - This task is 35 days shorter
7   The task#1 has several differences (3)
8    - This task is 5 days shorter
9    - This task has 10 workers more
10   - This task starts before (15 days)
11
12   [...]
13
14  The task#4 has several differences (3)
15   - This task is 3 days shorter
16   - This task has 3 workers more
17   - This task starts before (22 days)
18  The task#6 has several differences (2)
19   - This task is 2 days shorter
20   - This task starts before (23 days)
```

Listing 1: Excerpt of recommendation explanation.

The *editor panel* (see Figure 5 ②) strongly supports users during the modelling phase of the construction site. Users can compose tasks by providing a reduced set of parameters.

The *visualisation panel* (See Figure 5 ③) provides a calendar, in a format resembling a Gantt diagram, where each task is temporally located. The position of each task depends on its starting and duration time values. This diagram also shows temporal dependencies between tasks. In order to facilitate the interpretation of the tasks, the tool provides a colour pallet to group each guild with a different colour. Hence, all the tasks performed by the same guild will have the same colour. In addition, the constraints can be generated in a graphical way by joining tasks with directed arrows.

5.3. Modelling example

As we said in the beginning of this section, our tool relies on a strong mathematical model that, in particular, can optimise the planning of construction sites. The main strength of $\mathcal{SINPA}$ is its ability to provide users with a friendly editor that allows them to optimise their construction planning without having specific knowledge of the underlying theory. In order to illustrate the usability of the tool, we present a modelling example based on the theoretical planning project provided in Section 4.

Initially, the user must access the editor template, which will be empty. Then, the tasks will be modelled and introduced using the editor panel (See Figure 5 ②). These tasks can be inserted by using actions 3 and 4 of the editor (see Table 1 and Figure 5 ③). In this example, we define six tasks. Next, the maximum duration of the planning and the available budget to carry out the construction site must be introduced to the system. This can be done by using action 20 of the actions panel (see Table 1) to invoke the *general config panel* and provide the corresponding values to the fields *Max-budget* and *Max-durations*. Afterwards, it is necessary to create the resources that must be used in the construction site. For this, it is required

Category	Id	Name	Notation	Description
Task Edition	1	Undo		Allows to reverse the effects if the last action.
	2	Redo		Allows to cancel the last redo.
	3	Insert above		Inserts a task above the selected one.
	4	Insert below		Inserts a task below the selected one.
	5	Indent task		Indents the selected task to create a subtask.
	6	Unindent task		Unindents the selected task.
	7	Move up		Moves up the selected task.
	8	Move down		Moves down the selected task.
View	9	Zoom in		Makes zoom in the diagram.
	10	Zoom out		Makes zoom out the diagram.
	11	Refresh		Refreshes the diagram.
	12	Delete		Deletes the selected task.
	13	Clear		Deletes all the tasks of the diagram.
	14	Print		Prints the diagram.
	15	Critical path		Identifies the critical path of the diagram.
	16	Save		Saves the content of the diagram to disk.
Config	17	Holidays		Shows the holidays editor, which allows to include non-working days in the construction site planning.
	18	Resources		Shows the resources editor that allows to include new resources, which are necessaries for the proper accomplishment of the construction site.
	19	Strategies		Shows the strategies editor, which allows to select an specific strategy to solve the constraints satisfaction in the current planning site configuration.
	20	General config		Shows the general configuration editor, which allows to configure the constraints solver engine of the system.
Report	21	Budget	Total budget	Shows the total money spent in the construction site.
	22	Duration	Total duration	Shows the duration of the planning.
	23	Process	process!	Starts the constraint solving process to handle the current configuration.

Table 1: Description of the actions panel.

to use action 18 of the actions panel (see Table 1) to invoke the *resources panel*, where the *name* of the resource and its *price* per day will be introduced. In this example, a unique resource is introduced, which will be shared by three different tasks. Once the resources have been created, they must be associated to the required tasks. In this example, the previously created resource will be shared by tasks 1, 2 and 4. For this, the task editor will be invoked to associate the resource to each task by selecting it from the available list and setting the *quantity* to 1. The result of these initial steps is shown in Figure 7 (a).

The temporal constraints between tasks can be designed in a graphical way by using the diagram editor. Once the problem has been completely defined, we have the situation given in Figure 7 (b). Then, in order to start the optimisation process, the user must use action 23 of the actions panel (see Table 1) to invoke the *process*.

The next step is to process scheduling in the system core. The results are presented in the *strategy panel*, where a brief overview of the results obtained by applying the three proposed strategies can be found. In order to obtain all the information related to the recommendation, the user must invoke the *report*. This information can be visualised by marking the option *view* in the *strategy panel*. Then, the results are presented to the user as a diagram (see Figure 7 (c)).

At this point, several constraints could be applied through different optimisation iterations. For example, we could say, by using the dedicated editor, that April 11 and 12 are non-working days. Once introduced, the system automatically recomputes the optimal scheduling of the project (see Figure 7 (d)).

6. Case study

In order to show the applicability and suitability of $\mathcal{SINPA}$, in this section we present a case study where we model, analyse and optimise a real construction site planning. We take as initial step our previous work (Camacho et al., 2018) and analyse

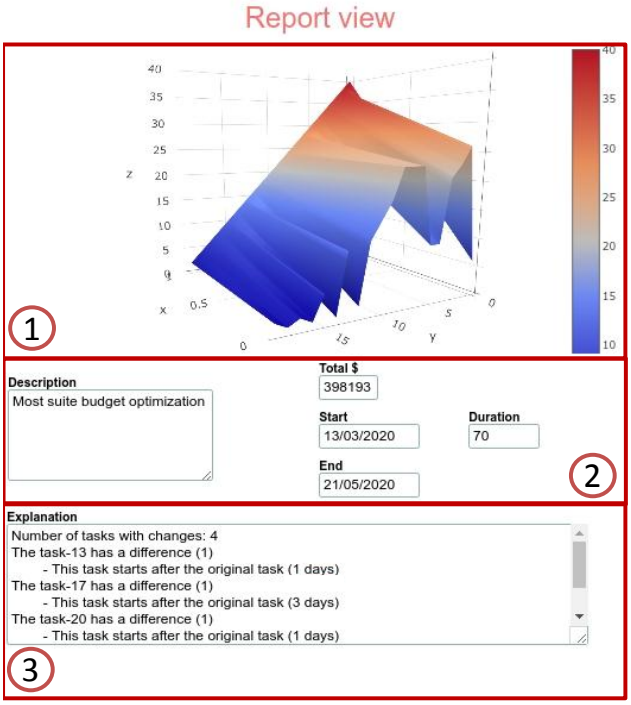


Figure 6: Report panel.

Main activities	
Activity	**Cost**
A - Detailed design	£ 21.750
B - Surveys and Investigation	
C - Preparation of H&S Plans in respect to the CDM regulations	
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	£ 27.709
J - Supply, construction, installation, testing and commissioning activities	£ 398.193
K - Training of DVW Operatives	£ 2.550
L - Commissioning	£ 5.450
TOTAL	**£ 455.652**

Table 2: Main activities of the project.

the scheduling of the construction site by using the new optimisations included in our framework. The first step has been to recover, from the original project, some required information that was discarded in our previous work because it was not used in our previous simpler framework. For example, we now need the minimum days needed to complete a task, the cost of each task and the cost of worker per day, among others. The complete description of the construction site under study can be found in Section 6.1, whereas the potential optimisations are presented and discussed in Section 6.2.

6.1. Description of the construction site

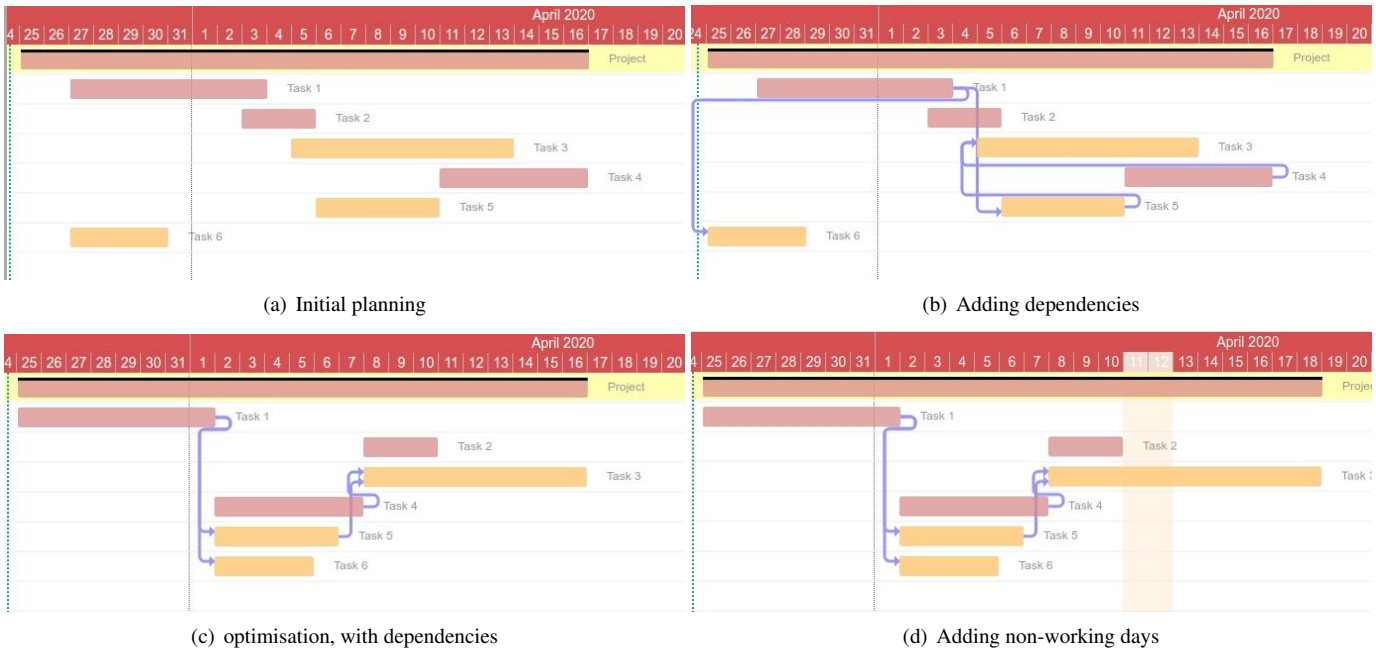
The construction site analysed in this study is a water treatment plant carried out in the UK. Specifically, this purification plant treats waters by using ultra violet disinfection and reducing plumbosolvency in the network by installing a sodium dihydrogen orthophosphate dosing equipment. Some information related to the project has been anonymised to preserve the privacy of customers and contractors. Despite this fact, all the requirements and conditions of the project have remained the same. However, with the aim of reducing the length of the case study and focus on its most relevant characteristics, only time durations, fixed costs, labour and the cost of their contracting have been considered in this case study. The values related to the amounts and costs of materials have been discarded and, for the sake of clarity, considered to be fixed.

The initial scheduling of the construction site, which consists of 12 tasks composing the works and their associated costs, is shown in Table 2. Taking into account that our main goal is to find optimal time durations to optimise the construction site planning, we have considered to focus our efforts on Activity J: *supply, construction, installation, testing and commissioning activities*. This task aggregates the most important construction and commissioning activities carried out by the project staff. Activities A to H represent those tasks that should be performed in the initial stages of the project. Among them, we can find the design of the project, preparation of the plans, risks control and the preparation of the set of fees/taxes that must be paid. Activities I and K focus on the training and contracting of the staff. Activity L involves the commissioning tasks, whose main goal is to check the correctness, security and safety of the final product obtained from this planning.

Table 2 shows that Activity J is the most expensive component of the project. Therefore, potential time/labour improvements will imply a reduction in the final budget. In order to apply our methodology to this activity, we present its 15 sub-activities and their characteristics (costs, time durations and labour). These activities are described in Table 3 and they reflect the data of the real construction. The third, fourth and fifth columns include values corresponding to the original plan. The sixth column indicates the cost of adding one day of one worker to that sub-activity. The last column indicates the minimum number of days needed to complete the sub-activity. In the real implementation of the project, all these time values and conditions were accomplished by the employers, according to the instructions of the contractor. The original organisation of the project is presented in the Gantt diagram given in Figure 11.

6.2. Optimising the construction site by using SINPA

This section presents the results obtained by applying the three proposed optimisations to the planning of our water treatment construction site. First, the original organisation of the construction, including the information related to the scheduling presented in the previous section, has been modelled with



(a) Initial planning

(b) Adding dependencies

(c) optimisation, with dependencies

(d) Adding non-working days

Figure 7: Steps of the construction site optimisation

$\mathcal{SINPA}$. Then, once the construction site has been modelled, it has been analysed and optimised by the constraint solving engine running under $\mathcal{SINPA}$. The results of this process are presented in Figure 8, which shows the evolution of the cost and the duration of the construction site during the optimisation process. In order to properly visualize this evolution, the optimisation process has been presented as a sequence of iterations, which are represented as intermediate solutions until reaching the most suitable scheduling. At the top of the figure, Figures 8(a), 8(b) and 8(c) show the evolution of the optimisation process by using the report panel of $\mathcal{SINPA}$. In addition, at the bottom of the figure, we have included Figures 8(d), 8(e) and 8(f) showing the same evolution but using a simplified view. In the graphs, the x-axis corresponds to the iteration number while the y-axis includes two different scales: the cost is shown at the left of the graph and the duration is shown at the right.

Figure 8(d) shows the result related to the *most suitable budget optimisation* (MSBO). Let us remind that this optimisation depends on the maximum budget of the client. In this case, we considered that an increment of 50% with respect to the original cost of the project might be assumed. The minimum duration reached with this optimisation is 50 days, while the cost reaches a total budget of £ 598.328. Therefore, $\mathcal{SINPA}$ will suggest a solution where the total cost increases by £ 211.161 while we are able to reduce the construction time in 20 days. The new scheduling resulting from applying this optimisation is automatically computed as presented in Figure 12. As can be seen in the diagram, the most optimised tasks are 7, 9 and 10, whose duration have been decreased in 40, 31 and 16 days, respectively. In order to obtain this acceleration, we need to increase the number of workers devoted to these tasks. Specifically, we need to add 160, 114 and 54 workers, respectively. The remaining tasks that have decreased their duration are 1, 2, 3, 4, 6, 8, 11, 14 and 15, which have shortened their duration by 8, 7, 1, 1, 8, 2, 3, 1 and 3 days, respectively. Again, we need to increase the total number of workers: 48, 28, 6, 6, 40, 10, 12, 6, and 12, respectively.

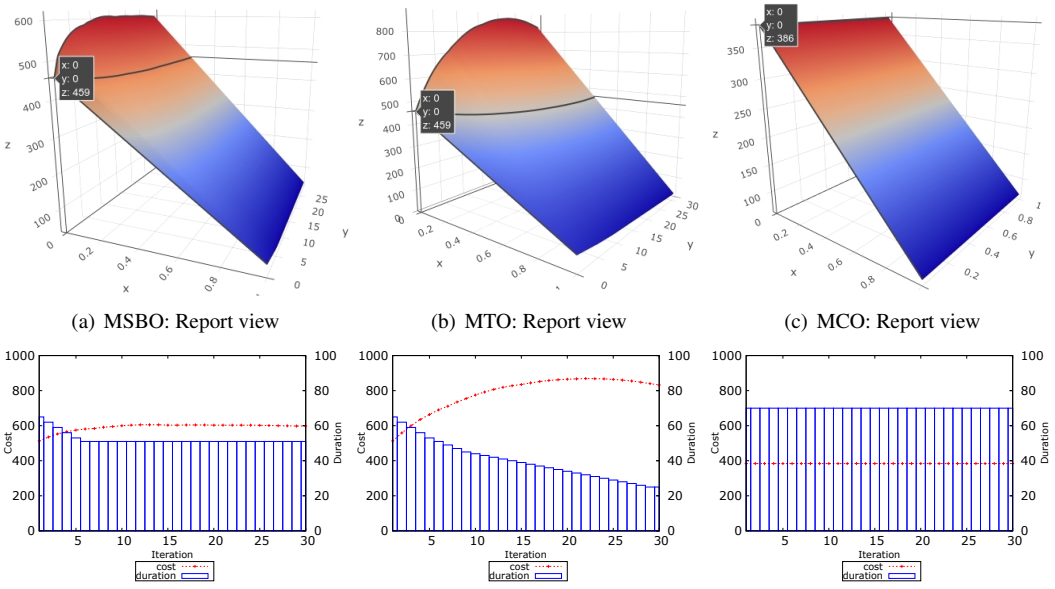
In order to show the adaptability of this optimisation scheme, he have used three different available budgets: an increase of 20%, 40% and 100% of the original cost of the project. The results are shown in Figure 9, where the graphs present a comparison of the final cost (see Figure 9(a)) and duration (see Figure 9(b)). It can be seen that when we increase the available budget, not surprisingly, the total days needed to complete the project is reduced.

Figure 8(e) shows the result of applying the *minimum time optimisation* (MTO) on the planning. In this case, the minimum duration achieved is 24 days, which entails a cost of £ 823.478. This reflects an increment of £ 425.285 and a reduction of 46 days with respect to the original planning. Similarly to the MSBO optimisation, several tasks have been optimised, causing a decrease in the total duration of the project. The main difference between applying MSBO and MTO to the original planning relies on task 1, which have reduced its total duration in 38 days, by increasing 228 workers.

The result concerning the application of the *minimum cost optimisation* (MCO) strategy are presented in Figure 8(f). The duration of the planning remains the same while the total cost has been reduced to £ 384.906. This is an important result because it shows that this specific optimisation is able to reduce the total budget by £ 13.287 while keeping the original duration of the project. The complete planning resulting after applying this optimisation can be seen in Figure 13. In contrast to previous optimisations, MCO only affects the tasks 3, 4 and 14, which have reduced the total duration in one day, by increasing 2 workers each.

Activity J						
Id	**Description**	**Total Days (plan)**	**Total workers (plan)**	**Total Cost (plan)**	**W.P.D. Cost**	**Min. Days**
J1	Roads, fencing gates	**58**	-	**£ 57.103**	-	-
J1.1	Removal of trees and vegetation in all areas	10	5	£ 20.495	£ 95,00	4
J1.2	Road works	48	5	£ 36.608	£ 95,00	16
J2	Excavation and associated temporal work	**13**	-	**£ 11.175**	-	-
J2.1	Raw water main trench	5	6	£ 7.020	£ 95,00	3
J2.2	Excavations and levelling in kiosk and tanks area	8	6	£ 4.155	£ 95,00	3
J3	Contractor's compound access	**2**	3	**£ 13.660**	£ 227,16	1
J4	Kiosk for UVD equipment	**2**	3	**£ 12.050**	£ 113,50	1
J5	Kiosk base slab	**3**	1	**£ 7.050**	£ 100,00	3
J6	UVD equipment	**3**	3	**£ 19.695**	£ 164,55	1
J7	Raw water monitoring instruments	**60**	-	**£ 25.993**	-	-
J7.1	Sodium dihydrogen Orthophosphate storage tank	45	3	£ 19.653	£ 25,00	15
J7.2	Blind tank and bunded area	15	3	£ 6.340	£ 25,00	5
J8	Sodium dihydrogen Orthophosphate plant	**3**	3	**£ 108.080**	£ 632,11	1
J9	Installation of the new raw water main	**47**	-	**£ 18.280**	-	-
J9.1	Installation of new polyethylene raw water main	32	3	£ 10.890	£ 23,00	11
J9.2	Ductile iron pipes & fitting inside kiosk	15	3	£ 7.390	£ 75,00	5
J10	Electrical works	**25**	3	**£ 62.915**	£ 120,50	9
J11	Ducts	**5**	3	**£ 9.845**	£ 130,33	2
J12	Software and SCADA	**5**	1	**£ 21.359**	£ 100,33	5
J13	CCTV Camera	**2**	3	**£ 3.445**	£ 25,88	1
J14	Safety shower	**2**	3	**£ 3.608**	£ 25,22	1
J15	Chemical lines	**5**	3	**£ 23.935**	£ 237,08	2
TOTAL		**235**	**63**	**£ 398.193**	-	-

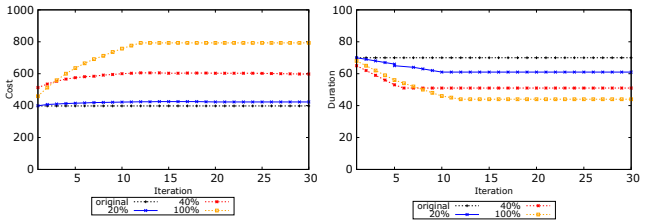
Table 3: Activity J. Tasks, days and costs.



(a) MSBO: Report view (b) MTO: Report view (c) MCO: Report view

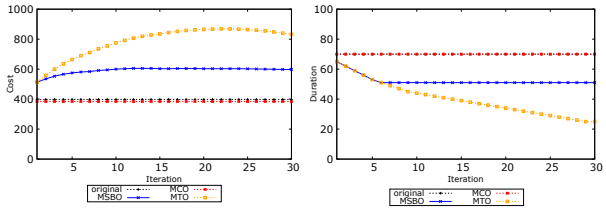
(d) MSBO: Simplified view (e) MTO: Simplified view (f) MCO: Simplified view

Figure 8: Evolution of the cost and duration of the construction site by using the proposed optimisations.



(a) Evolution of the cost (b) Evolution of the duration

Figure 9: Comparison of the results obtained with MSBO using different available budgets.



(a) Evolution of the cost (b) Evolution of the duration

Figure 10: Comparison between the results obtained by the proposed optimisations.

Figure 10 shows a comparison between the results obtained after applying the three proposed optimisations. Figure 10(a) shows the comparison in terms of cost while Figure 10(b) illustrates the comparison with regard to the total duration. MSBO is the most balanced option because it allows a reduction of the total duration of the project in a percentage of 29% while increasing the total budget in 49%. MTO is a suitable option to know how to finish the project on a fast track: the total duration of the project can be reduced by 72%. However, it is the most expensive optimisation: the total budget increases around 100%. MCO is the optimisation that has a lesser impact on the original scheduling, but it is really useful to (slightly) reduce expenses. In our case study, this optimisation would reduce the total cost of the project by 5% without increasing its total duration.

7. Conclusions and future work

We have presented $\mathcal{SINPA}$: a tool that facilitates the planning of construction sites by providing useful recommendations to their users. Although our tool strongly depends on Constraint Programming, users can skip the underlying mathematical theory because they only need to provide the information needed to plan the construction site and $\mathcal{SINPA}$ will transform this information into constraints that will be added to the constraint solver. In order to evaluate the usefulness of our framework, we have analysed a real construction site plan and show how the original plan can be optimised.

We currently contemplate three lines for future work. First, as suggested by a reviewer, we would like to obtain the opinion of other experts so that we have a better evaluation of our tool. Second, we would like to explore the inclusion of *mutation* techniques in our framework. Briefly, we would like to develop a framework such that given a *mutant*, that is, an incorrect version of a construction site plan, it is able to detect either its incorrectness (that is, it does not conform to the original plan) or its inefficiency (that is, either it needs more time or it is more expensive than the original one). In order to efficiently generate and process a huge amount of mutants, we will rely on current mutation testing techniques (Cañizares et al., 2018; Gómez-Abajo et al., 2018, 2021; Gutiérrez-Madroñal et al., 2019). A third line of research, and using our previous work (Cañizares et al., 2019; Cañizares et al., 2020; Hierons and Núñez, 2017; Hierons et al., 2008, 2018; Merayo et al., 2018b,a; Núñez et al., 2021) as starting point, considers whether we can adapt our constraints system to represent the causality relations established in the construction of cloud and distributed systems. Actually, the parallelism and exclusion that we are able to model and analyse with $\mathcal{SINPA}$ is similar in this type of systems.

Acknowledgements

We would like to thank Sergio Zorrilla, construction engineer, for his help during the development of the $\mathcal{SINPA}$ tool. His useful comments and hints have contributed to greatly improve $\mathcal{SINPA}$. We would also like to thank the reviewers of the paper for the careful reading and useful comments and suggestions that have improved the quality of the paper. This work has been supported by the Spanish MINECO/FEDER projects FAME (RTI2018-093608-B-C31) and the Comunidad de Madrid project FORTE-CM (S2018/TCS-4314) co-funded by EIE Funds of the European Union.

References

Aggarwal, C.C., 2016. Recommender Systems. The textbook. Springer.

Apt, K.R., 1999. The essence of constraint propagation. Theoretical computer science 221, 179–210.

Apt, K.R., 2003. Principles of constraint programming. Cambridge university press.

Barták, R., Salido, M.A., Rossi, F., 2010. New trends in constraint satisfaction, planning, and scheduling: a survey. The Knowledge Engineering Review 25, 249–279.

Camacho, A., Cañizares, P.C., Estévez-Martín, S., Núñez, M., 2018. A tool-supported framework for work planning on construction sites based on constraint programming. Automation in Construction 86, 190–198.

Cañizares, P.C., Núñez, A., de Lara, J., Llana, L., 2020. MT-EA4Cloud: A methodology for testing and optimising energy-aware cloud systems. Journal of Systems and Software 163, 110522:1–25.

Cañizares, P.C., Núñez, A., Lara, J.d., 2019. An expert system for checking the correctness of memory systems using simulation and metamorphic testing. Expert Systems with Applications 132, 44–62.

Cañizares, P.C., Núñez, A., Merayo, M.G., 2018. Mutomvo: Mutation testing framework for simulated cloud and HPC environments. Journal of Systems and Software 143, 187–207.

Choi, C.W., Harvey, W., Lee, J.H.M., Stuckey, P.J., 2006. Finite domain bounds consistency revisited, in: 19th Australian Joint Conf. on Artificial Intelligence, AI'06, LNCS 4304, Springer. pp. 49–58.

Correas, J., Estévez-Martín, S., Sáenz-Pérez, F., 2018. Enhancing set constraint solvers with bound consistency. Expert Systems with Applications 92, 485–494.

Dave, B., Buda, A., Nurminen, A., Främling, K., 2018. A framework for integrating BIM and IoT through open standards. Automation in Construction 95, 35–45.

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

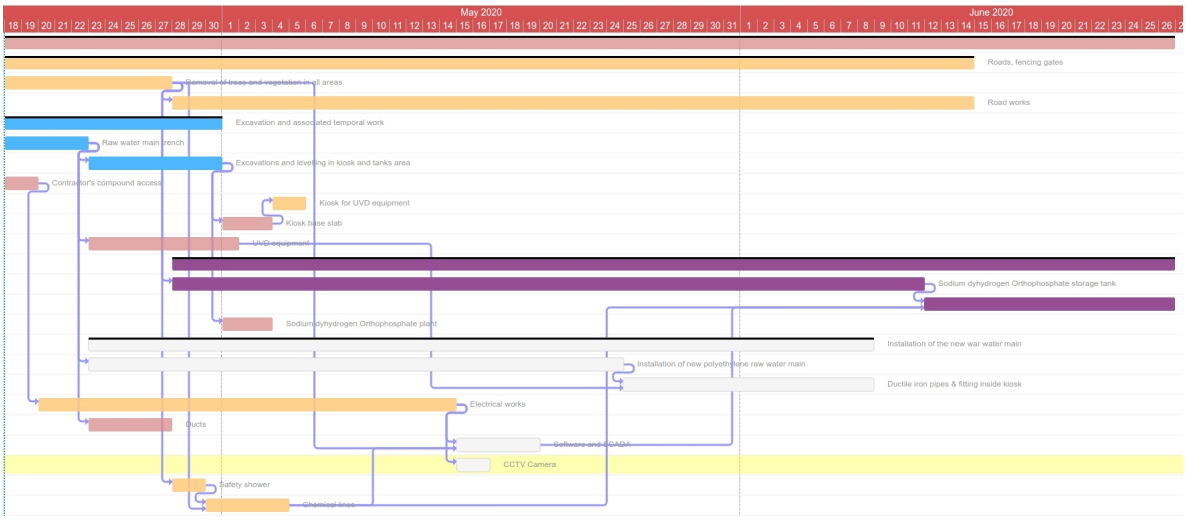


Figure 11: Original Gantt diagram of the project.

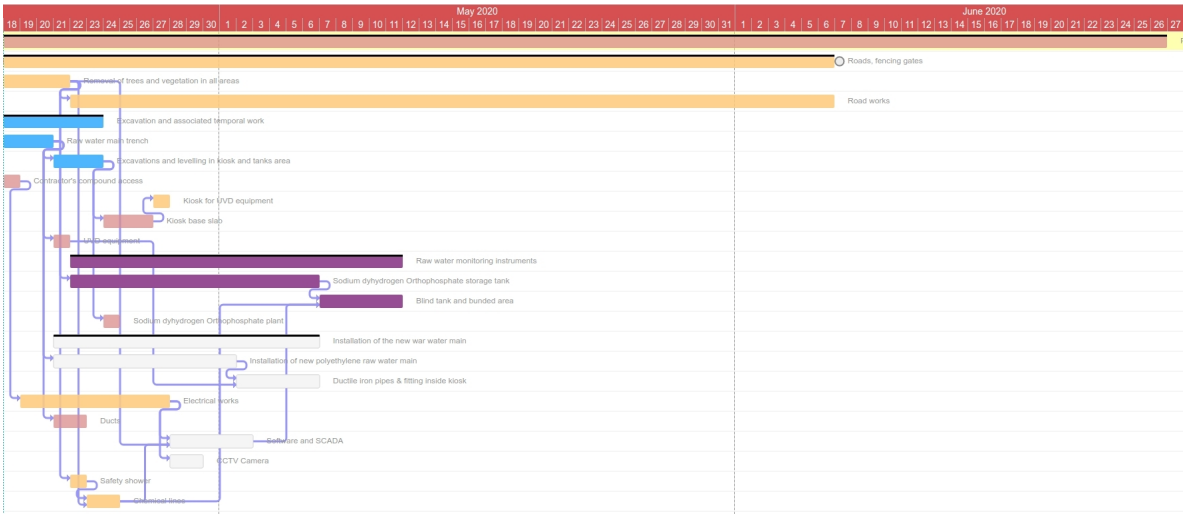


Figure 12: Gantt diagram of the project after applying MSBO.

Estévez-Martín, S., Fernández, A.J., Hortalá-González, M.T., Rodríguez-Artalejo, M., Sáenz-Pérez, F., Vado-Vírseda, R.D., 2009. On the Cooperation of the Constraint Domains $\mathcal{H}$, $\mathcal{R}$ and $\mathcal{FD}$ in *CFLP*. Theory and Practice of Logic Programming 9, 415–527.

Faghihi, V., Reinschmidt, K.F., Kang, J.H., 2014. Construction scheduling using Genetic Algorithm based on Building Information Model. Expert Systems with Applications 41, 7565–7578.

Flanagan, D., 2006. JavaScript: The Definitive Guide. O'Reilly Media, Inc.

Friesen, J., 2019a. Parsing and creating JSON objects with Gson, in: Java XML and JSON. Document Processing for Java SE. 2nd ed.. Springer, pp. 243–298.

Friesen, J., 2019b. Processing JSON with Jackson, in: Java XML and JSON. Document Processing for Java SE. 2nd ed.. Springer, pp. 323–403.

Gómez-Abajo, P., Guerra, E., de Lara, J., Merayo, M.G., 2018. A tool for domain-independent model mutation. Science of Computer Programming 163, 85–92.

Gómez-Abajo, P., Guerra, E., de Lara, J., Merayo, M.G., 2021. Wodel-Test: a model-based framework for language-independent mutation testing. Software and Systems Modeling 20, 767–793.

Gutiérrez-Madroñal, L., Medina-Bulo, I., Domínguez-Jiménez, J.J., 2019. Evaluation of EPL mutation operators with the MuEPL mutation system. Expert Systems with Applications 116, 78–95.

Hall, M., 2001. More servlets and JavaServer pages. Prentice Hall PTR.

Hickson, I., 2015. HTML5 specification. http://www.w3.org/TR/2012/WD-html5-20121025/.

Hierons, R.M., Merayo, M.G., Núñez, M., 2008. Controllable test cases for the distributed test architecture, in: 6th Int. Symposium on Automated Technology for Verification and Analysis, ATVA'08, LNCS 5311, Springer. pp. 201–215.

Hierons, R.M., Merayo, M.G., Núñez, M., 2018. Bounded reordering in the distributed test architecture. IEEE Transactions on Reliability 67, 522–537.

Hierons, R.M., Núñez, M., 2017. Implementation relations and probabilistic schedulers in the distributed test architecture. Journal of Systems and Software 132, 319–335.

Lamport, L., 2015. Who builds a house without drawing blueprints? Communications of the ACM 58, 38–41.

Lin, M.C., Tserng, H.P., Ho, S.P., Young, D.L., 2012. A novel dynamic progress forecasting approach for construction projects. Expert Systems with Applications 39, 2247–2255.

Magnani, L., Bertolotti, T. (Eds.), 2017. Handbook of Model-Based Science. Springer.

Martínez-Rojas, M., Marín, N., Vila Miranda, M.A., 2016. An intelligent sys-

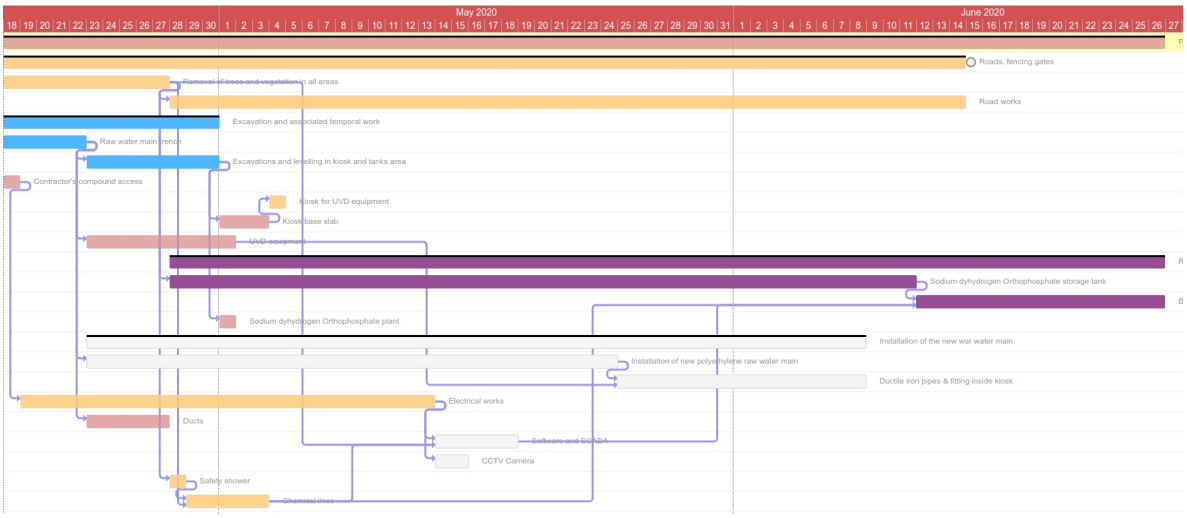


Figure 13: Gantt diagram of the project after applying MCO.

tem for the acquisition and management of information from bill of quantities in building projects. Expert Systems with Applications 63, 284–294.

Merayo, M.G., Hierons, R.M., Núñez, M., 2018a. Passive testing with asynchronous communications and timestamps. Distributed Computing 31, 327–342.

Merayo, M.G., Hierons, R.M., Núñez, M., 2018b. A tool supported methodology to passively test asynchronous systems with multiple users. Information & Software Technology 104, 162–178.

Molinero, C., Núñez, M., 2011. Planning of work schedules through the use of a hierarchical multi-agent system. Automation in Construction 20, 1227–1241.

Niknam, M., Karshenas, S., 2015. Integrating distributed sources of information for construction cost estimating using semantic web and semantic web service technologies. Automation in Construction 57, 222–238.

Núñez, A., Cañizares, P.C., Núñez, M., Hierons, R.M., 2021. TEA-Cloud: A formal framework for testing cloud computing systems. IEEE Transactions on Reliability 70, 261 – 284.

Powell, T.A., 2010. HTML & CSS: the complete reference. 5th ed., McGraw-Hill.

Prud'homme, C., Fages, J.G., Lorca, X., 2017. Choco Documentation. TASC - LS2N CNRS UMR 6241, COSLING S.A.S. URL: `\url{https://choco-solver.org/}`.

Quirk, V., 2012. A brief history of BIM. Arch Daily, http://www.archdaily.com/302490/a-brief-history-of-bim.

Rossi, F., van Beek, P., Walsh, T. (Eds.), 2006. Handbook of Constraint Programming. volume 2 of *Foundations of Artificial Intelligence*. Elsevier.

Sawhney, A., Bashford, H., Walsh, K., Mulky, A., 2003. Agent-based modeling and simulation in construction, in: 35th Winter Simulation Conference, IEEE Computer Society. pp. 1541–1547.

Taillandier, F., Taillandier, P., Tepeli, E., Breysse, D., Mehdizadeh, R., Khartabil, F., 2015. A multi-agent model to manage risks in construction project (SMACC). Automation in Construction 58, 1–18.

Tang, S., Shelden, D.R., Eastman, C.M., Pishdad-Bozorgi, P., Gao, X., 2019. A review of building information modeling (BIM) and the internet of things (IoT) devices integration: Present status and future trends. Automation in Construction 101, 127–139.

Tintarev, N., Masthoff, J., 2015. Explaining recommendations: Design and evaluation, in: Ricci, F., Rokach, L., Shapira, B. (Eds.), Recommender Systems Handbook. 2nd ed.. Springer. chapter 10, pp. 353–382.

Wang, H., Meng, X., 2019. Transformation from IT-based knowledge management into BIM-supported knowledge management: A literature review. Expert Systems with Applications 121, 170–187.

Warin, G., 2015. Mastering Spring MVC 4. Packt Publishing.

Zhong, B.T., Ding, L.Y., Love, P.E.D., Luo, H.B., 2015. An ontological approach for technical plan definition and verification in construction. Automation in Construction 55, 47–57.