Exploring Hydrogen Supply/Demand Networks: Modeller and Domain Expert Views

Matthias Klapperstueck
Department of Human-Centred Computing, Faculty of IT, Monash University, Clayton, VIC, Australia

Frits de Nijs
Department of Data Science and AI, Faculty of IT, Monash University, Clayton, VIC, Australia
ARC Industrial Training and Transformation Centre OPTIMA, Carlton, VIC, Australia

Ilankaikone Senthooran
Department of Data Science and AI, Faculty of IT, Monash University, Clayton, VIC, Australia
ARC Industrial Training and Transformation Centre OPTIMA, Carlton, VIC, Australia

Jack Lee-Kopij
Woodside Energy Ltd., Perth, Australia

Maria Garcia de la Banda
Department of Data Science and AI, Faculty of IT, Monash University, Clayton, VIC, Australia
ARC Industrial Training and Transformation Centre OPTIMA, Carlton, VIC, Australia

Michael Wybrow
Department of Human-Centred Computing, Faculty of IT, Monash University, Clayton, VIC, Australia

Abstract

Energy companies are considering producing renewable fuels such as hydrogen/ammonia. Setting up a production network means deciding where to build production plants, and how to operate them at minimum electricity and transport costs. These decisions are complicated by many factors including the difficulty in obtaining accurate current data (e.g., electricity price and transport costs) for potential supply locations, the accuracy of data predictions (e.g., for demand and costs), and the need for some decisions to be made due to external (not modelled) factors. Thus, decision-makers need access to a user-centric decision system that helps them visualise, explore, interact and compare the many possible solutions of many different scenarios. This paper describes the system we have built to support our energy partner in making such decisions, and shows the advantages of having a graphical user-focused interactive tool, and of using a high-level constraint modelling language (MiniZinc) to implement the underlying model.

2012 ACM Subject Classification Theory of computation → Constraint and logic programming; Theory of computation → Integer programming; Human-centered computing → Information visualization

Keywords and phrases Facility Location, Hydrogen Supply Chain, Human-Centric Optimisation

Digital Object Identifier 10.4230/LIPIcs.CP.2023.21

Funding Woodside Energy Ltd.

1 Introduction

Moving away from fossil fuels is needed to achieve (net) zero carbon emissions [12]. Part of this move focuses on the production of hydrogen and ammonia as storage carriers for their use in mobile applications and seasonal storage. It is estimated the world production of hydrogen needs to more than double by 2030 [9] and, given around 50% of the current production [5] relies on natural gas, the demand for "green" hydrogen is even higher. This

requires energy companies to expand their production capacity as efficiently as possible. To achieve this, energy companies need to solve a complex combinatorial optimisation problem with energy sourcing, operation, transport and demand constraints that can be summarised as making two interdependent decisions – *where to build* the hydrogen production plants and *how to operate* the production and supply process – that minimise costs.

This paper describes the system we have built to support our energy partner in solving this problem, which we refer to as the facility location and operation problem. While already deployed at our partner's servers, the system is in constant evolution to incorporate the new capabilities our partner requests, while they continue using it. Importantly, the system does not just consist of a problem model that is instantiated with input data and solved using a solver, as this did not satisfy our partner's needs. Instead, we took advantage of the capabilities of constraint programming modelling languages (MiniZinc [15]) and our knowledge of interactive user interfaces to build a system that supports users in obtaining, exploring and comparing solutions in different ways, tailored to our partner's needs.

In particular, the system provides users with (a) a *diverse* set of solutions that can be compared visually and numerically, (b) *conflict resolution* methods that, upon infeasibility, identify conflicted constraints and guide users on which to modify, (c) different kinds of *data and model approximations* that allow users to explore near-optimal solutions quickly, (d) the ability to *interactively add/remove/modify constraints* via the interface to explore different scenarios, (e) the ability to impose simple *robustness* constraints that ensure the solution is resilient to unexpected facility outages, and (f) two different models of the same problem that are used to cross-validate the correctness of the solutions. The amount of work required to implement the system was significantly reduced thanks to the use of MiniZinc: its compiler directly supports (a–b), while its high-level nature significantly simplifies (c–f).

Our industry partner has assessed our code against alternative tools and selected ours based both on quantitative and qualitative measures (software design & architecture, user interface, and technology stack), as well as its additional features (diversity and robustness). This provides some measure of its quality and usefulness.

2 Facility location and operation problem overview

Intuitively, the facility location and operation problem solved by our Hydrogen Network Optimisation System (or `HyNetOS`) aims to select the location and configuration of the hydrogen production plants needed to supply the demand of the given demand locations, in such a way as to minimise the total cost of building and operating the resulting production/supply

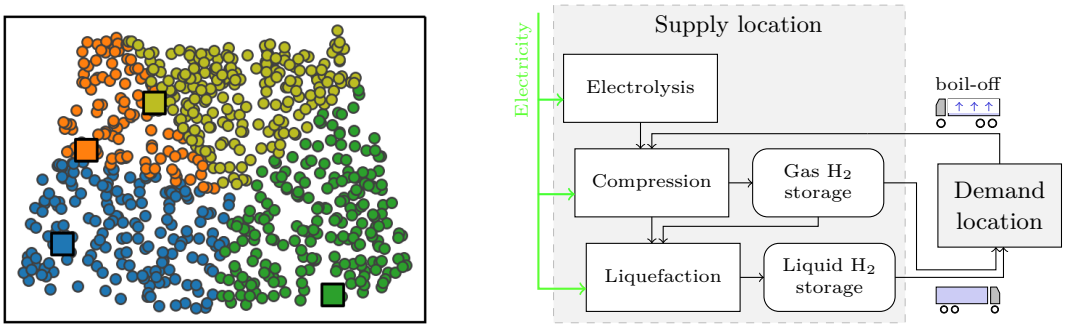

Figure 1 Left: solution showing 4 plant locations (squares) and the demand locations each of them supplies (circles in the plant's colour). Right: abstraction of a single plant.

network over a number of years. The left-hand-side of Figure 1 shows a visualisation of a particular solution, where 4 plant locations (the squares) out of all 9 plant locations given as input are selected by `HyNetOS` to supply all demand locations (the circles), also given as input. The decision of who supplies who is visualised via colour, with demand locations supplied by the same plant sharing the colour of that plant.

As shown in the right-hand-side of Figure 1, each production plant is itself built using one or more units of three production components – electrolysers, compressors and liquefiers – and two storage systems – for liquid and for compressed gas. The units in which each of these five types of hardware can be delivered, referred to as stock keeping units (SKUs), have different characteristics such as capacity, electricity consumption, and maximum change per hour in production rates, which are given as input. The hydrogen is transported between supply and demand locations using a truck-based transportation network, where the transport costs between each supply-demand location are given via an input table. Note that during the plant sizing, we take into account the impact boil-off has on the final quantity delivered after transport. Boil-off is the decompression due to the boiling of residual hydrogen liquid as a result of increased temperature inside a (nearly) empty truck's storage tank. Further, we also take into account the location specific price of the available electricity sources necessary to operate each facility, including the generation of a facility operation schedule to minimise electricity costs under variable electricity price and availability.

Importantly, the problem is defined across two timescales: *periods* and *hours*. A period is the number of consecutive years during which the demand for hydrogen is assumed to stay constant. Plants are built during some period and must only increase in size in later periods since, currently, demand is assumed never to decrease. In contrast, the electricity prices for some markets are given hourly for an entire year, yielding up to 8760 possible different prices per market. Because of this, we can potentially perform market arbitrage by adjusting production rates to fit the market price for electricity. However, this means the optimisation needs to make power consumption decisions on an hourly resolution.

The problem's objective is to minimise the total cost of the network, which can be broken down into the following interdependent cost elements: *building* (CAPEX) and *operating* (OPEX) the hydrogen production plants; *transporting* the product from supply to demand locations; and *powering* the production plants using the available electricity sources. The latter includes a monthly cost (*demand charge*) some electricity providers add to try to flatten their consumer's load profile, and is proportional to the plant's highest monthly load.

From an optimisation perspective, this problem is challenging due to the large search space created by the high number of demand locations to be supplied (several hundreds), the different electricity sources often available, and the hourly electricity prices the system must consider across an entire year (up to 8760 per source). This is further complicated by decisions on two different timescales: plant construction decisions are taken for each period, while operating decisions are taken hourly. In addition, the dramatic scale difference between CAPEX (millions) and electricity costs (cents per kWh), makes the model numerically unstable. Together, these factors mean that solving the hydrogen facility model to optimality can quickly become out of reach even for commercial mixed-integer solvers such as Gurobi [7]. Furthermore, due to the costs involved, plant construction decisions necessitate the examination of a range of scenarios, requiring the problem to be solved many times.

3 Modeller's view

As `HyNetOS` is developed for industry and there is no ground-truth, we put special attention on reducing modelling errors. To do this, we separately implemented and integrated two models of the problem whose functional equivalence is continuously checked by ensuring solutions to any model instance can be given to the other without drop in objective or infeasibility. This practice increases the redundancy of the programming task, thereby reducing the residual probability of errors [21]. This section presents the input data used by both models (referred to as `Ori` for the original model, and `Alt` for the alternative one, both always compiled into a MILP problem by MiniZinc), the constraints implemented by one (`Alt`) due to space constraints, and the modelling changes that most improved solving time.

3.1 Input data

Both models require the following input data, which MiniZinc refers to as *parameters*:

- $\mathcal{D}$ and $\mathcal{S}$: set of client demand locations and set of candidate supply locations, respectively.
- $\mathcal{K}$: set of products; currently $\mathcal{K} = \{\text{LIQ}, \text{GAS}\}$, i.e., liquid and gas hydrogen, respectively.
- $\mathcal{H}$: set of hardware; currently $\mathcal{H} = \{\text{ELEC}, \text{COMP}, \text{LIQF}, \text{GAS\_STR}, \text{LIQ\_STR}\}$ corresponding to electrolyser, compressor, liquifier, gas and liquid storage types, respectively.
- $\mathcal{P}$: set of constant demand periods and, for each period $p \in \mathcal{P}$, how many years $p_y \in \mathbb{N}$ it covers within the given plan horizon (typically 20 years).
- $\mathcal{SO}_i$: set of electricity sources (market labels) available at supply location $s_i \in \mathcal{S}$. Elements of set $\mathcal{SO}_i$ correspond to year-long time series of electricity prices, recorded at an hourly resolution. Markets are one of three types: (a) UTILITY, a conventional metered connection with mostly fixed prices, (b) PPA, a fixed-price power purchase agreement with a renewable energy provider (e.g., solar or wind farm), and (c) WHOLESALE, a market with generally the lowest price but exposed to volatile price fluctuations. We distinguish them for their unique features; demand charges typically only apply to UTILITY, while PPA is a zero carbon emissions source, necessary to achieve carbon targets in the future.
- $\mathcal{T}_i \subseteq \{t_1, t_2, \ldots, t_\tau\}$: set of time steps considered for a single representative year at supply location $s_i \in \mathcal{S}$. Each time step $t \in \mathcal{T}_i$ has a duration in hours $h_{i,t} \in \{1, \ldots, 24\}$, during which the price of electricity is constant. The value of τ ranges between 365 and 8760, corresponding to the cases where every time step represents 1 day and 1 hour, respectively. Their sum $\sum_{t \in \mathcal{T}_i} h_{i,t}$ must always be 8760.
- $D^k_{p,j} \in \mathbb{R}_{\geq 0}$: daily demand of product $k \in \mathcal{K}$ from location $d_j \in \mathcal{D}$ throughout period $p \in \mathcal{P}$, in tonnes per day.
- $T^k_{p,i,j} \in \mathbb{R}_{>0}$: transport cost in \$ per kg of product k sent from supply location $s_i \in \mathcal{S}$ to demand location $d_j \in \mathcal{D}$ during period $p \in \mathcal{P}$.
- $c^{\text{so}}_{p,i,t} \in \mathbb{R}_{>0}$: electricity cost of source $\text{so} \in \mathcal{SO}_i$ at supply location $s_i \in \mathcal{S}$ during time step $t \in p_y$ of any year of period $p \in \mathcal{P}$, in millions of \$ per MW.
- $\hat{c}^{\text{so}}_{p,i,t} \in \mathbb{R}_{\geq 0}$: demand charge of source $\text{so} \in \mathcal{SO}_i$ at supply location $s_i \in \mathcal{S}$ during time step $t \in p_y$ of any year of period $p \in \mathcal{P}$ in millions of \$ per MW. Common for UTILITY.
- $\text{LB}^{\text{so}}_i \in \mathbb{R}_{\geq 0}$: minimum annual energy usage in MWh required to be allowed to consume energy from source $\text{so} \in \mathcal{SO}_i$ at location $s_i \in \mathcal{S}$. In practice, only relevant for PPA.

For each hardware type $h \in \mathcal{H}$, we also need the set of concrete stock keeping units (SKUs) in which h can be delivered, and the following hardware-specific properties:

- N^h: number of available SKUs for h, from which we build index set $\mathcal{C} = 1..N^c$.
- $C^c \in \mathbb{R}_{>0}$: capacity of SKU $c \in \mathcal{C}$ in tonnes per day.
- K^c_p: ownership cost in millions of \$ for one SKU $c \in \mathcal{C}^h$ starting from period $p \in \mathcal{P}$.

- $\nu^h \in [0..1]$: minimum production (turndown) rate proportional to installed capacity for h.
- $e^h \in \mathbb{R}_{\geq 0}$: electricity usage in MWh per tonne of production needed to run h.
- $\rho^h \in [0..1]$: overhead to production and storage for $h \in \{\text{COMP}, \text{LIQF}, \text{GAS\_STR}, \text{LIQ\_STR}\}$ required to compensate for boil-off.
- $\mu \in [0..1]$: maximum ramping rate of the liquefaction units.

3.2 Decision variables, constraints and objective function

Figure 2 shows the objective and constraints in model `Alt`, where blue is used for decision variables and black for input data, split into the following six groups.

Network constraints. As hydrogen demand is assumed to stay constant throughout any period $p \in \mathcal{P}$, we assume the supply network will also stay constant throughout p. Thus, we only have to decide once per period p which location $s_i \in \mathcal{S}$ supplies (part of) the daily demand of location $d_j \in \mathcal{D}$ for product $k \in \mathcal{K}$. The network constraints encode these assumptions. Variable $x^k_{p,i,j}$ represents the percentage of the demand d_j of product k supplied by location s_i in period p. Constraint (1) ensures that every product of every demand location is serviced in its entirety by the supply locations. In order to satisfy the demand induced by the network, production plants must be built at one or more of the supply locations. Binary variable $b^k_{p,i}$ represents whether a plant for product k is built at location s_i at (or before) period p, i.e., whether there is a plant in s_i producing k in period p. Constraint (2) ensures $b^k_{p,i} = 1$ whenever any $x^k_{p,i} > 0$, thereby capturing the need for a plant. Currently, hybrid facilities (producing *both* gas and liquid for offtake) are disallowed via constraint (3); a requirement of our industry partner.

The objective, shown in expression (26), includes as its first component the sum of the total cost of transporting the product across the links selected with non-zero weight x, where transporting the entire demand (of $D^k_{p,j}$ tonnes per day) of product k from $s_i \to d_j$ costs $T^k_{p,i,j}$ in period p.

Offtake constraints. During a period $p \in \mathcal{P}$, the daily demand $D^k_{p,j}$ of product $k \in \mathcal{K}$ at every location $d_j \in \mathcal{D}$ is constant. Thus, we denote as $z^k_{p,i}$ the *offtake rate* (in tonnes per day) of any supply location $s_i \in \mathcal{S}$ for period p and product k, i.e., the rate at which the plant at location s_i must produce k over the long term, using storage to buffer production rate changes in the short term. Constraints (4) and (5) define $z^k_{p,i}$ for $k = \text{LIQ}$ and $k = \text{GAS}$, respectively. Note that the offtake of gas $z^{\text{GAS}}_{p,i}$ includes the material needed to produce liquid *from* that gas through liquefaction, and not just the transport requirements of gas.

Plant capacity constraints. Producing $z^k_{p,i}$ of product $k \in \mathcal{K}$ requires a production plant at location $s_i \in \mathcal{S}$ during period $p \in \mathcal{P}$ that is of suitable capacity. This can be modelled using the following five constraints. First, let variable $u^c_{p,i}$ represent how many of SKU $c \in \mathcal{C}^h$ of hardware type $h \in \mathcal{H}$ are *newly* installed at the start of period p (and thus available for use in p) in supply location s_i. The maximum (or peak) production capacity of the plant at location s_i in period p is defined by constraint (6) as the sum of the SKUs capacities installed *up to* p, and denoted by variable $y^h_{p,i}$. By tying the capacity to the cumulative number of newly installed units, we ensure plants can only grow in size.

Second, the plant must be big enough for its peak capacity to meet the offtake rate $z^k_{p,i}$. This is ensured by constraint (7), which adds the overhead factor ρ^h to the normal offtake to take transportation boil-off into account. The boil-off factor is applied differently to different

Network

$$\sum_{s_i \in \mathcal{S}} x^{k}_{p,i,j} = 1 \qquad \forall p, d_j, k \text{ where } D^{k}_{p,j} > 0 \qquad (1)$$

$$x^{k}_{p,i,j} \leq b^{k}_{p,i} \qquad \forall p, s_i, d_j, k \qquad (2)$$

$$\sum_{k \in \mathcal{K}} b^{k}_{p,i} \leq 1 \qquad \forall p, s_i \qquad (3)$$

Offtake

$$\sum_{d_j \in \mathcal{D}} D^{\text{LIQ}}_{p,j} x^{\text{LIQ}}_{p,i,j} = z^{\text{LIQ}}_{p,i} \qquad \forall p, s_i \qquad (4)$$

$$z^{\text{LIQ}}_{p,i} + \sum_{d_j \in \mathcal{D}} D^{\text{GAS}}_{p,j} x^{\text{GAS}}_{p,i,j} = z^{\text{GAS}}_{p,i} \qquad \forall p, s_i \qquad (5)$$

Plant capacity

$$\sum_{\substack{1 \leq p' \leq p, \\ c \in \mathcal{C}^h}} C^c u^{c}_{p',i} = y^{h}_{p,i} \qquad \forall p, s_i, h \qquad (6)$$

$$(1+\rho^h) z^{h \to k}_{p,i} \leq y^{h}_{p,i} \qquad \forall p, s_i, h \in \{\text{ELEC}, \text{COMP}, \text{LIQF}\} \qquad (7)$$

$$z^{h \to k}_{p,i} \geq \nu^h y^{h}_{p,i} \qquad \forall p, s_i, h \in \{\text{ELEC}, \text{COMP}, \text{LIQF}\} \qquad (8)$$

$$\rho^{h \to k} z^{h \to k}_{p,i} + m^{h \to k} b^{h \to k}_{p,i} \leq y^{h}_{p,i} \qquad \forall p, s_i, h \in \{\text{GAS\_STR}, \text{LIQ\_STR}\} \qquad (9)$$

$$m^{\text{LIQ}} z^{\text{LIQ}}_{p,i} \leq y^{\text{LIQ\_STR}}_{p,i} \qquad \forall p, s_i \qquad (10)$$

Production

$$\nu^h y^{h \to k}_{p,i} \leq \hat{p}^{h \to k}_{p,i} \qquad \forall p, s_i, h \in \{\text{ELEC}, \text{COMP}, \text{LIQF}\} \qquad (11)$$

$$\hat{p}^{h \to k}_{p,i} + p^{h \to k}_{p,i,t} + \rho^h z^{h \to k}_{p,i} \leq y^{h}_{p,i} \qquad \forall p, s_i, t, h \in \{\text{ELEC}, \text{COMP}, \text{LIQF}\} \qquad (12)$$

$$p^{\text{LIQ}}_{p,i,t+1} - p^{\text{LIQ}}_{p,i,t} \leq \mu h_{i,t} y^{\text{LIQF}}_{p,i} \qquad \forall p, s_i, t \qquad (13)$$

$$p^{\text{LIQ}}_{p,i,t} - p^{\text{LIQ}}_{p,i,t+1} \leq \mu h_{i,t} y^{\text{LIQF}}_{p,i} \qquad \forall p, s_i, t \qquad (14)$$

$$p^{\text{LIQ}}_{p,i,0} - p^{\text{LIQ}}_{p,i,|\mathcal{T}_i|} \leq \mu h_{i,t} y^{\text{LIQF}}_{p,i} \qquad \forall p, s_i, t \qquad (15)$$

$$p^{\text{LIQ}}_{p,i,|\mathcal{T}_i|} - p^{\text{LIQ}}_{p,i,0} \leq \mu h_{i,t} y^{\text{LIQF}}_{p,i} \qquad \forall p, s_i, t \qquad (16)$$

Storage

$$s^{h \to k}_{p,i,t} \leq y^{h}_{p,i} \qquad \forall p, s_i, t, h \in \{\text{GAS\_STR}, \text{LIQ\_STR}\} \qquad (17)$$

$$s^{\text{GAS}}_{p,i,t+1} = s^{\text{GAS}}_{p,i,t} + \frac{h_{i,t}}{24}\left(\hat{p}^{\text{GAS}}_{p,i} + p^{\text{GAS}}_{p,i,t} - \hat{p}^{\text{LIQ}}_{p,i} - \frac{p^{\text{LIQ}}_{p,i,t} + p^{\text{LIQ}}_{p,i,t+1}}{2} - z^{\text{GAS}}_{p,i} + z^{\text{LIQ}}_{p,i}\right) \qquad \forall p, i, t \qquad (18)$$

$$s^{\text{LIQ}}_{p,i,t+1} = s^{\text{LIQ}}_{p,i,t} + \frac{h_{i,t}}{24}\left(\hat{p}^{\text{LIQ}}_{p,i} + \frac{p^{\text{LIQ}}_{p,i,t} + p^{\text{LIQ}}_{p,i,t+1}}{2} - z^{\text{LIQ}}_{p,i}\right) \qquad \forall p, i, t \qquad (19)$$

$$s^{k}_{p,i,1} = s^{k}_{p,i,|\mathcal{T}_i|} = 0 \qquad \forall p, s_i, k \qquad (20)$$

Electricity usage

$$24 e_{p,i,t} = e^{\text{ELEC}}(\hat{p}^{\text{GAS}}_{p,i} + p^{\text{GAS}}_{p,i,t}) + e^{\text{COMP}}(\hat{p}^{\text{GAS}}_{p,i} + p^{\text{GAS}}_{p,i,t} + \rho^{\text{GAS}} z^{\text{GAS}}_{p,i}) + e^{\text{LIQF}}\left(\hat{p}^{\text{LIQ}}_{p,i} + \frac{p^{\text{LIQ}}_{p,i,t} + p^{\text{LIQ}}_{p,i,t+1}}{2} + \rho^{\text{LIQ}} z^{\text{LIQ}}_{p,i}\right) \qquad \forall p, i, t \qquad (21)$$

$$\sum_{\text{so} \in \mathcal{SO}} e^{\text{so}}_{p,i,t} = e_{p,i,t} \qquad \forall p, s_i, t \qquad (22)$$

$$\hat{c}^{\text{so}}_{p,i,t} e^{\text{so}}_{p,i,t} \leq \hat{e}^{\text{so}}_{p,i,m[t]} \qquad \forall p, s_i, t, \text{so} \qquad (23)$$

$$\left(\sum_{t \in \mathcal{T}_i} (h_{i,t} e^{\text{so}}_{p,i,t}) > 0\right) \implies \left(\sum_{t \in \mathcal{T}_i} (h_{i,t} e^{\text{so}}_{p,i,t}) \geq \text{LB}^{\text{so}}_i\right) \qquad \forall p, s_i, \text{so} \qquad (24)$$

$$u^{c}_{p,i} \in \mathbb{N}_0;\ b^{k}_{p,i} \in \{0,1\};\ 0 \leq x^{k}_{p,i,j} \leq 1;\ y, z, p, s, e \in \mathbb{R}_{\geq 0} \qquad (25)$$

Obj.

$$\min \sum_{p,i,j,k} \left(T^{k}_{p,i,j} x^{k}_{p,i,j}\right) + \sum_{p,i,c} \left(K^{c}_{p} u^{c}_{p,i}\right) + \sum_{p,i,t,\text{so}} \left(c^{\text{so}}_{p,i,t} e^{\text{so}}_{p,i,t}\right) + \sum_{p,i,m,\text{so}} \left(\hat{e}^{\text{so}}_{p,i,m}\right) \qquad (26)$$

Figure 2 Objective and constraints for the `Alt` model of the `HyNetOS` decision system.

types of hardware $h \in \mathcal{H}$ because boil-off comes in uncompressed gas form. Thus, it has to pass through the compressor and the liquefactor but not the electrolyser. We use $h \to k$ to indicate a mapping from hardware to product. For example, since the electrolyser $h = \text{ELEC}$ produces $k = \text{GAS}$, the electrolyser must be scaled to meet gas demand $z^{h\to k}_{p,i}$. Third, the plant should not be so big that running at minimum capacity yields more product on average than is demanded. Constraint (8) captures this requirement, where the minimum production rate depends on the minimum turndown capabilities ν^h of the hardware. Finally, we impose two kinds of minimum storage size constraints. The first constraint ensures two things: 1) that every plant has at least some gas storage for buffering boil-off during transport, which is captured as a fraction $\rho^{h\to k}$ of daily production rate $z^k_{p,i}$, and 2) that every plant meets a global minimum storage amount $m^{h\to k}$ if it is built ($b^{h\to k}_{p,i}$) (constraint (9)).

The second constraint ensures enough liquid storage is built to sustain liquid hydrogen offtake for a minimum number of days m^{LIQ} (constraint (10)).

The construction of any SKU $c \in \mathcal{C}^h$ of hardware $h \in \mathcal{H}$ incurs CAPEX and OPEX costs. These are aggregated into the period-specific component cost K^c_p (in millions of \$ per unit) and accumulated via the number of SKUs newly installed in p (given by $u^c_{p,i}$), yielding the objective term $K^c_p u^c_{p,i}$. The sum of these objective terms forms the second component of the objective, shown in expression (26).

Production constraints. For the plant operation, we model the average operating costs through a representative year per period. This means we create a production schedule for one year in each (constant demand) period, which then repeats for however many years the period is long. The production rate of hardware $h \in \mathcal{H}$ for product $k \in \mathcal{K}$ at supply location $s_i \in \mathcal{S}$ during period $p \in \mathcal{P}$ is allowed to change during each of the time steps $t \in \mathcal{T}_i$ defined for that location. The production rate is measured in tonnes per day and modelled via two terms: $\hat{p}^{h\to k}_{p,i} + p^{h\to k}_{p,i,t}$. Here, $\hat{p}^{h\to k}$ is the constant *baseline* production amount, and $p^{h\to k}_{p,i,t}$ the component that is variable in time step t. Constraint (11) ensures the baseline exceeds the minimum turndown production rate. Constraint (12) ensures the flexible component never exceeds the installed capacity, while also keeping sufficient headroom for the boil-off fraction ρ that has to be re-compressed and re-liquefied. Constraints (13) and (14) deal with $h = \text{LIQF}$ having a slow ramping speed. They constrain the ramp up and down rate, respectively, between two consecutive production rates (where the consecutive steps wrap around the year, via constraints (15) and (16)), such that the change in rate does not exceed the ramping capability μ of liquefaction (0.1 of total capacity, i.e., 10%) by the number of hours $h_{i,t}$ for which the ramping is maintained.

Storage constraints. Variable production is buffered via storage: overproduction causes the storage level to increase, while underproduction causes the constant offtake to drain the storage. At every time step $t \in \mathcal{T}_i$ of supply location $s_i \in \mathcal{S}$ in period $p \in \mathcal{P}$, we track the storage level of product $k \in \mathcal{K}$ in tonnes via variable $s^k_{p,i,t}$. Constraint (17) ensures we never store more than the plants' installed storage capacity. Changes in storage levels are captured by constraints (18) for $k = \text{GAS}$, and (19) for $k = \text{LIQ}$. In both, the storage in $t+1$ is an offset from the storage in t, plus hourly production (baseline plus flexible), minus offtake (either by the liquefaction unit, or by the demand locations constant offtake factor). In addition, we anchor the storage to a baseline with a wrap-around constraint in (20). Adding a baseline of 0 at each year's end removes some of the symmetries in storage schedule solutions and ensures the storage level is implementable across period changes.

Electricity cost constraints. The last set of constraints connects the daily hydrogen production to the electricity cost required to produce it. To this end, at every time step $t \in \mathcal{T}_i$ for the plant of supply location $s_i \in \mathcal{S}$ in period $p \in \mathcal{P}$, we represent the power consumption in MW of the plant via variable $e_{p,i,t}$. This variable is defined by constraint (21) as the sum of the power consumed by the plant's hardware $h \in \{\text{ELEC}, \text{COMP}, \text{LIQF}\}$ to produce the required hydrogen in t of p, i.e., as the electricity usage e^h multiplied by the production rate at that time which, as before, is formed by a baseline $\hat{p}^k_{p,i,t}$ component, plus a flexible and possibly a boiloff one. Since our production rates are daily, we downscale the power consumption by factor 24 to the per-hour rate. Constraint (22) distributes the power consumption among sources via variables $e^{so}_{p,i,t}$, which represent the power consumed from each electricity source $so \in \mathcal{SO}$ available at that location. These variables are used to derive the cost of this power, which is accumulated in the third sum of the objective. Constraint (23) defines variable $\hat{e}^{so}_{p,i,m[t]}$ representing the monthly demand charge for the month $m[t] \in \{1, \ldots, 12\}$ where time step t falls. It is the maximum demand charge (cost $c^{so}_{p,i,t}$ of each MW consumed $e^{so}_{p,i,t}$) that can be obtained due to the power consumed at each t of that month. This accounts for the last sum of the objective. Finally, constraint (24) ensures electricity source $so \in \mathcal{SO}$ is selected only if its minimum annual energy usage LB^{so}_i is met.

3.3 Effective modelling and solving in practice

One of the primary benefits of using a *Constraint Modelling Language* such as MiniZinc is that it takes care of the details of mapping models to solvers efficiently (e.g., linearisation). This is important because writing good (i.e. efficiently solvable) models can otherwise be a process of significant trial-and-error, as much an art form as a science. While there are important strategies for good modelling [22], the "last mile" of good practice involves rules of thumb, such as minimizing the number of integer variables in favour of floats, or avoiding equalities if possible. MiniZinc takes care of these during compilation, allowing the modeller to focus on semantic changes. Nevertheless, because of its high level nature, it is equally well-suited to rapid prototyping of model changes. This is why for our industry partner's quantitative evaluation of the system, we were able to use MiniZinc to try to find model improvements that reduce the total runtime. To do this we took a reference implementation of the model and evaluated the runtime of several model changes. The progression of these changes is shown as a box plot in Figure 3 and discussed below.

1. Split power equality. The power equality (21) involves variables $e^{\text{so}}_{p,i,t}$ that are directly minimized in the objective. As such, it is tempting to remove the equality constraint by replacing it with $\geq$, such that the model will always demand at least as much power as needed to produce the hydrogen. However, this change allows overconsumption of power to meet the PPA minimum requirement (24). Therefore, we split the power equality constraint into a greater than and a *conditionally applied* less than part, which is only applied if the source has a minimum usage requirement through a conditional constraint:

`forall (i) if (MIN_USE[I]) then` $\sum_{\text{so} \in \mathcal{SO}} e^{\text{so}}_{p,i,t} = e_{p,i,t}$ `else` $\sum_{\text{so} \in \mathcal{SO}} e^{\text{so}}_{p,i,t} \geq e_{p,i,t}$ `endif;`

2. Tighter constraints. Plant production boolean $b^k_{p,i}$ is tied to many floating point supply indicator variables. Instead of defining them separately, we can define the state activity indicator with a single constraint, which will be active if any demand is supplied, i.e.:

$$\sum_{d_j \in \mathcal{D}} x^k_{p,i,j} \leq |\mathcal{D}|\, b^k_{p,i}, \quad \forall p, s_i, k$$

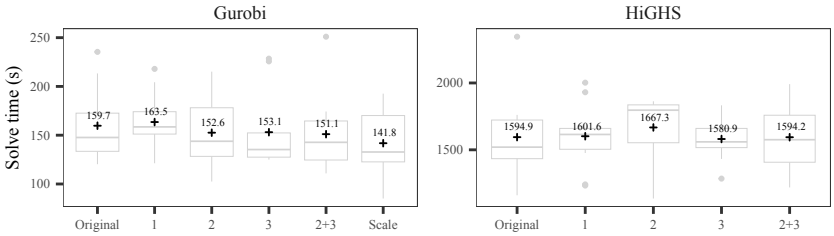

Figure 3 Response of model solve time as a result of model changes; mean time annotated.

3. Direct objective formulation. In our reference implementation, the objective terms are captured by intermediate variables for each of the sums that make up the objective. We can rewrite the objective directly in terms of the base variables, allowing the compiler to group terms together, and furthermore helping the MIP solver to prove optimality faster.

We evaluated the time-to-optimality with each model change over 10 runs with different seeds for the solver, to average out effects of solver-internal randomized branching choices and heuristics, and capture statistically meaningful averages on the total runtime. We used a Linux machine with AMD Ryzen 9 3950X 16-Core Processor (3.5-4.7 GHz) and the input data file A shown in Table 1, which has 9 supply locations, 100 demand locations, 3 SKUs per hardware type, and an average period length of 642 tiers (min 365, max 1338). Figure 3 tracks the distributions of total runtime as the changes are implemented, for the GUROBI 9.5.1 and HIGHS 1.5.0 [8] solvers. We also evaluated the combination of 2 and 3 which individually seemed to produce improvements. Nevertheless, the combination of compilation and pre-solving means that most changes have no significant effect. The biggest improvement comes from adjusting how GUROBI scales the objective terms, by changing the SCALEFLAG parameter. We believe this is due to the nature of the hydrogen facility location problem, where many candidate integer assignments are inherently feasible: we can always redistribute the supply network and adjust production rates. The optimisation is thus primarily guided by the (large) objective. We hypothesise that a core-guided search [6], which always assumes constraints can be satisfied and iteratively tightens the model upon detecting an infeasibility, would be a better strategy to solve this kind of model. However, due to the prevalence of floating point data, no core-guided solver available to us can be applied to this problem.

4 From modellers to users

To integrate our models into a useful decision system we extended them with functionality that its users (engineers and business experts) could access via the system's user interface (see Section 5). This includes minor changes to allow many decisions to be set to true/false by the user to explore different scenarios, as well as the more significant ones discussed here.

4.1 Run-time versus accuracy – approximations and warm-starts

The large number of variables involved in the hourly operational decisions, can make solving our models to optimality too time consuming for the user. Further, approximate solutions may be sufficient whenever users are interactively exploring what-if scenarios. Our system gives users several ways to control the time versus optimality trade-off, by means of the following two approximation options and warm-start strategy.

Constraint approximation – annual constant production. This approximation (referred to as `App`) allows each plant to run at a constant daily production rate sufficient to exactly meet the daily demand that the plant supplies. This means no intra-day or seasonal storage is required and plants can simply build the minimum storage capacity. As a result, in this version of the model all the production and storage constraints (11–20) are removed, and the electricity usage constraints are reformulated to operate on the total annual energy demand under a constant production. As such, the (hourly) energy cost coefficients $c^{\text{so}}_{p,i,t}$ are rescaled to the total annual cost for a given offtake rate in tonnes per day:

$$c^{\text{so}}_{p,i} = (e^{\text{E}} + e^{\text{C}} + e^{\text{L}}) \sum_{t \in \mathcal{T}_i} \left(c^{\text{so}}_{p,i,t} \cdot \frac{h_{i,t}}{24} \right)$$

Data approximation – price-tier grouping. In practice, many hours in a year often have identical prices for electricity. Since the only driver for operational decisions is a reduction in electricity price, we can group consecutive time steps with equal price. We use the utility market to inform the grouping operation, as utility prices often stay constant for hours, e.g. a daytime tariff and a cheaper night-time tariff from 11pm to 6am. For the other markets in the same compressed time step, we take the median price as the constant price. This approximation (referred to as `Tie`) reduces the cardinality τ of the set $\mathcal{T}_i$, thereby significantly reducing the total number of floating point variables in the instance (see Table 1).

Warm-starting strategy. Both `App` and `Tie` approximations are *valid*, i.e., if their solutions are fed to the non-approximated model (referred to as `Exa` and `Hou`, respectively) they also yield a solution: we can always set the production schedule at a finer time granularity to be equal to the constant-production (rate) assumptions at the coarser level. In addition, they can be combined to further speed up the search at the cost of accuracy. This observation informs our three-stage warm-starting strategy:

1. Solve using an approximate version of the model (i.e., `App-Hou` or `App-Tie`) to optimality;
2. Assign the integer decisions of that solution to the equivalent variables using an exact version of the model (i.e., `Exa-Hou` or `Exa-Tie`, resp) and resolve;
3. Warm-start that last used model with the solution from step 2 and resolve to optimality.

The final step arrives at the same objective as a traditional execution (referred to as *cold-start*) using the same model; however, it will find a good quality initial solution much faster.

Table 1 Size, solving time and objective value of the instances obtained with four data files.

	Input data size					Instance size for GUROBI solver					
File	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{D}\rvert$	N	M	t/day	Model	Float	Int	Cons.	Time(s)	Ratio
A	9	100	9	18	[40]	`Exa-Hou`	597.7 k	144	835.6 k	5114.85	1.000
						`Exa-Tie`	96.6 k	144	65.3 k	34.96	1.020
						`App-Hou`	2.0 k	144	3.4 k	0.22	1.055
						`App-Tie`	2.0 k	144	3.4 k	0.19	1.068
B	**7**	100	9	**14**	**[278]**	`Exa-Hou`	457.1 k	112	642.2 k	20982.06	1.000
						`Exa-Tie`	74.7 k	112	56.5 k	38.06	1.008
						`App-Hou`	1.6 k	112	2.7 k	0.19	1.017
						`App-Tie`	1.6 k	112	2.7 k	0.17	1.016
C	9	100	9	18	**[278]**	`Exa-Hou`	597.7 k	144	835.6 k	22605.64	1.000
						`Exa-Tie`	96.6 k	144	65.3 k	84.38	1.009
						`App-Hou`	2.0 k	144	3.4 k	0.31	1.017
						`App-Tie`	2.0 k	144	3.4 k	0.31	1.018
D	9	**709**	9	18	**[283, 293]**	`Exa-Hou`	1217.4 k	360	1706.7 k	—	—
						`Exa-Tie`	215.2 k	360	166.0 k	1309.47	> 1.016
						`App-Hou`	26.1 k	360	42.2 k	7.81	> 1.027
						`App-Tie`	26.1 k	360	42.2 k	5.37	> 1.027

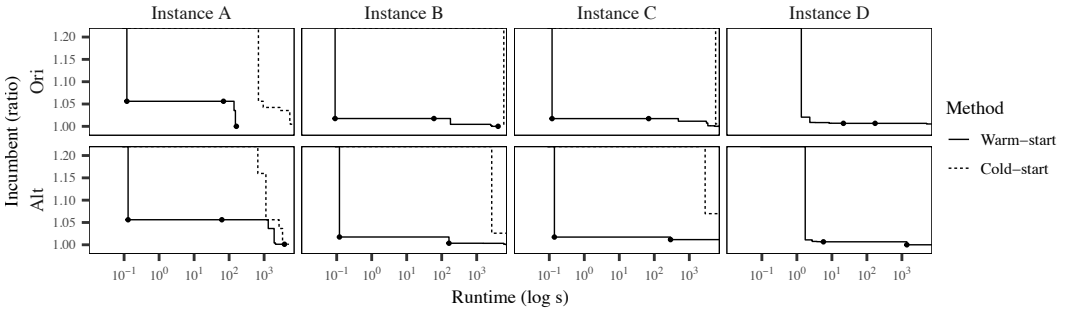

Figure 4 Quality of incumbent solution (ratio over optimum) as a function of log runtime obtained by the warm- and cold-start strategies on each instance. Stage solutions are annotated with points.

Experimental snapshot. Table 1 shows a snapshot of the trade-offs obtained by the above methods instantiated with four data files (A-D). For each file, it shows the number of supply locations $|\mathcal{S}|$; demand locations $|\mathcal{D}|$; SKUs over all hardware types, $N = \sum_h N^h$; markets $M = \sum_i |\mathcal{SO}_i|$, and total demand per period in tonnes/day. It then shows the number (in thousands) of floating-point variables, integer variables, and constraints in the instance resulting from compiling each data file with either an Exa-Hou, Exa-Tie, App-Hou or App-Tie model for the GUROBI solver. The last two columns show the time (where – indicates a timeout) to find an optimal solution and prove optimality when executing each instance in cold-start mode using MINIZINC 2.7.6 and GUROBI 10.0.0.2, and the ratio between the best objective value found using the most accurate Exa-Hou model and the others.

Figure 4 compares the improvement of feasible solutions over time obtained with a cold-start run of the `Exa-Hou` model instantiated with the data files of Table 1, and with our warm-start strategy, using MINIZINC 2.7.6 and GUROBI 10.0.0.2. Typically, by the time the cold-start finds its first feasible solution (15+ minutes), the warm-start is in its third stage and up to 5% gap to optimality.

4.2 Diversity of solutions

As is common in optimisation, every instance of our problem has many optimal (and even more close-to-optimal) solutions. It is thus useful for decision-makers to obtain a number of close-to-optimal alternative solutions that are meaningfully different in terms of the components being optimised (e.g., transport vs electricity cost) or some major decisions (e.g., plants locations or capacities). Ingmar et al. [11] proposed strategies to find N diverse solutions of a model instance where the model includes the required user-defined diversity measures. One of these strategies iterates at most $N-1$ times from an optimal solution looking for a new solution that is as close to optimality as desired by the user, and is maximally diverse w.r.t. all previously found solutions.

This strategy is now implemented[1] in MINIZINC and used in our system by simply importing from our models a set of diversity measures our users can choose from. The measure most used currently is the Manhattan distance between the vectors of the Boolean variables $b^k_{p,i}$ indicating there is a plant at supply location $i \in \mathcal{S}$ in period $p \in \mathcal{P}$. Figures 5 and 7(D) show the results of one of the implemented diversity metrics, i.e. maximum diverse set of supply locations. To highlight the differences between diverse solutions our

[1] It will be released as part of the MiniZinc Python package [3] before the end of the year.

system allows users to compare solutions in terms of their supply/demand network, the configuration of the plants they selected, their cost components, etc. Such comparisons have already been useful in identifying, for example, supply locations that appear in all optimal and close-to-optimal solutions due to their favourable cost coefficients (e.g. green location in Figure 5) and are thus prime candidates for construction.

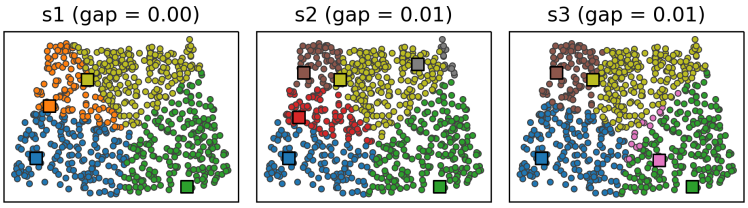

Figure 5 Three maximally diverse solutions within a maximum optimality gap of 0.02, using as distance measure the Manhattan distance of the vector of supply locations built by each solution.

4.3 Robustness against plant shutdown

In practice, optimal solutions may lack robustness against uncertain events. There are many sources of uncertainty in our problem, from changes to input data (e.g., costs, demand or hardware technology) to loss of production due to plant shutdowns. Our system is not yet required to deal with the former, as our users are still determining how to build reliable future scenarios and/or probability distributions. It can however be used to find solutions that, in the event of any plant failure, guarantee the supply of a user-defined minimum percentage of the total demand of the network (denoted as new parameter GS $\in [0..1]$, set to 0 as default). This allows users to compare optimal but non-robust solutions against robust but non-optimal ones. We achieved this by adding constraint (27) to the models to ensure that for every period $p \in \mathcal{P}$, supply location $s_i \in \mathcal{S}$, and product $k \in \mathcal{K}$, the user-defined percentage GS of the total demand $\sum_{d_j \in \mathcal{D}} D^k_{p,j}$ for product k in period p is covered by the sum of the peak production capacity of the plants in other supply locations $s_{i'} \in \mathcal{S} \setminus \{s_i\}$.

$$\text{GS} \sum_{d_j \in \mathcal{D}} D^k_{p,j} \leq \sum_{s_{i'} \in \mathcal{S} \setminus \{s_i\}} y^h_{p,i'} \qquad \forall p, s_i, k, h \in \{\text{ELEC}, \text{COMP}, \text{LIQF}\} \tag{27}$$

5 User's view

Our system (shown in Figure 6) connects two main components – optimisation and visualisation – via an application program interface (API). The optimisation calls MiniZinc with the model, input data and configuration to find and return its solutions. The visualisation calls the optimisation via the API and visualises the returned solutions on an interactive web-based dashboard application, referred to as the User Interface (UI). It was implemented using *Plotly Dash* [18], as requested by our industry partner.

Users experience our system mostly via its powerful UI, part of which appears in Figure 7. The UI allows domain-experts and non-expert users (e.g., decision makers) to load different input data, interactively turn some model constraints on/off, obtain solutions, compare them and explore them in detail. This not only helps them make decisions but also gain confidence and trust in the system's output. In addition, domain-experts and external tools can access the system's functionality independently of the UI via the API and the command line interface we implemented in Python. This allows them to automate the solving of instances, i.e. explore many scenarios with different sets of parameters.

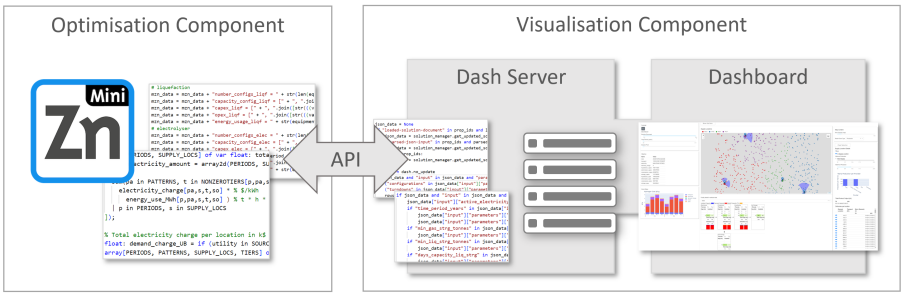

Figure 6 The two components integrated by our system and connected by an API.

5.1 Configuring an execution run

As described above, our system supports a wide range of execution modes (e.g., warm-starts, diversity, and robustness), which users can control via the UI. Fig.7(A) shows one of the panes provided for this. In it, users can select a pre-configured scenario (list of supply and demand locations, their demand, and electricity markets) or upload a new one; modify the scenario by turning on/off the allowed markets (Utility, PPA, Wholesale); select a model profile (e.g., `Ori-Exa-Tie`) from a list of available ones; determine whether to run in diversity, robustness or traditional mode; and modify the default configuration values for each of those. In addition, users can select one of the available solvers; set a timeout and number of threads; and set various solver specific parameters. Once users finish configuring the run, solving can be triggered by pressing a button. If solving time is expected to be long, users can leave the dashboard and load the solution later using the session manager, which keeps track of each execution and reports whether a solution was found, no solution was found due to infeasibility, or the solver is still running.

5.2 Visualising and interacting with solutions

The UI displays solutions via the two tabs shown in Figures 7(B) and (C). Figure 7(B) focuses on overview information and the supply/demand network, for which it combines different sub-windows (or *cards*). Figure 7(C) focuses on the operational parts of one plant.

Overview and Network Tab. This tab (Figure 7(B)) comprises cards (frames) that contain controller and visualisation elements. With the controller element a specific solution from a set of multiple solutions (for example, solutions from diversity, robustness, or different set of model parameters) and time period can be loaded and visualised. Below the controller a stacked barchart summarises the $/kg cost (i.e. each cost component) for all active supply locations for all periods. The map in the centre prominently shows the supply/demand network, with markers representing supply locations and coloured circles for each demand location, with the colour matching the supply location they are linked to. For each supply location a radial chart with 4 segments (north, west, south east) shows the amount of hydrogen delivery for each direction, and an area overlay highlights the covered area using semi-transparent rectangles drawn between each linked supply/demand locations, indicating compactness, spread or amount of overlap of supply areas. Supply locations can be selected and its details viewed in a separate card. To the right of the map it shows details about the delivery to each demand location; below it shows the plant configuration represented as a flow diagram with details about installed hardware capacity and number of SKUs, including a stacked bar-chart for each type of hardware showing utilisation to installed capacity. For a

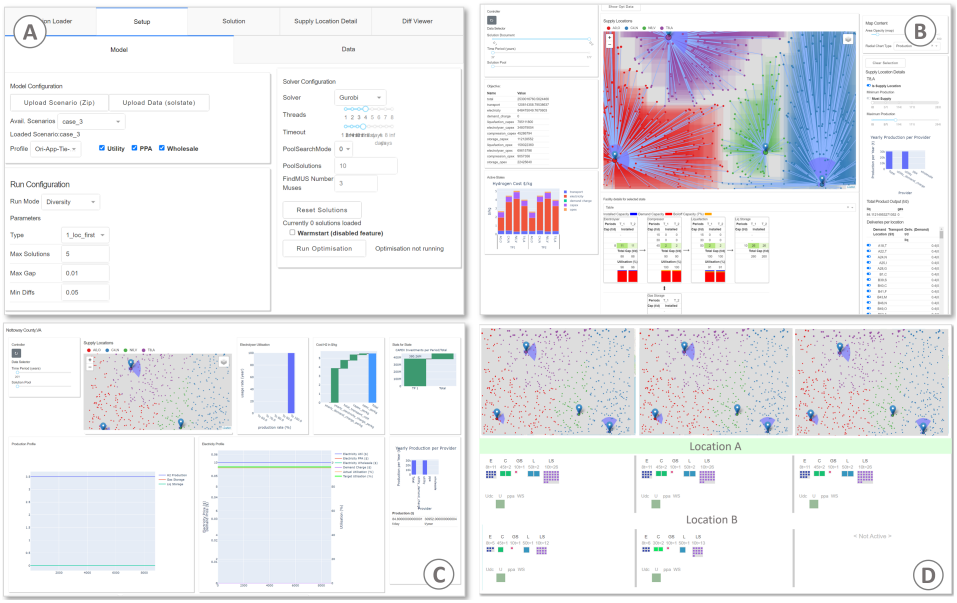

Figure 7 UI's four main tabs: A) input tab, which allows detailed configuration of the model and data before execution; B) supply network as an interactive map with details of selected supply location; C) operational details of each supply location; and D) comparison view of multiple solutions.

selected supply location, interactive elements allow changes to maximum and minimum daily production amount, disabling that particular supply location, or force a non-active location to be active. A re-solve of an instance containing the new data can then be triggered.

Operation Tab. For a supply location selected from the map card, this tab (Figure 7(C)) shows further details with focus on the operation side. The two cards in the middle show for each hour of the year (8760 hours) component usage and gas storage levels on the left, and electricity source utilisation (Utility, PPA, Wholesale) on the right. Other cards show the overall hydrolyser utilisation and waterfall charts of the investment costs. On the top left the same controller card is shown to select specific solutions and time periods.

5.3 Comparing solutions

Fig 7(D) shows part of the solution comparison pane for three solutions (one per column). For each solution, the pane shows a summary of the configuration of the run that created it, a breakdown of its objective value, and many other useful information, including the two kinds shown in the figure: a map of the plants it built and their hardware as a box-matrix plot (one plant per row). Each box represents one SKU of a particular type of hardware (identified by its colour) and capacity (identified by its size). This enables easy visual comparison of significant changes in plant locations and in their associated hardware. If a location appears in all solutions it is highlighted with a light green colour. Users can use the loading manager to compare any set of solutions obtained, for example, by selecting different model profiles, available sources, and diversity definitions; by modifying the robustness parameter, or by forcing a supply location to produce a given product.

5.4 Detecting and resolving conflict

It is not uncommon for users to create infeasible instances while exploring solutions by, for example, setting the minimum daily production for a plant to a higher value than its maximum production. This is a significant roadblock for the usability of any optimisation system. To address it, we use the conflict resolution method of [17], which allows users to obtain a visual representation of the conflicts that is easy to understand, select which of the conflicts to relax for a solution to be found, and obtain a solution to this relaxed instance that quantifies the minimum changes needed to restore feasibility for the original instance.

To implement this method the model needs to be changed to a) provide information about any constraint that can cause conflict, and b) soften these constraints to quantify the minimum changes needed. Both can be achieved in MiniZinc: a) by adding an annotation to each constraint that gives meaningful names to the constraint and its objects, and that tracks its values; b) by adding a slack variable to each constraint that quantifies the minimum changes (see [17] for details). The method also requires computing Minimum Unsatisfiable Sets (MUSes), where a MUS is a set of infeasible constraints in the instance such that removal of any one of the constraints makes the set feasible, and/or Minimum Correction Sets (MCS), where an MCS is a minimum set of constraints that if eliminated from the infeasible instance makes it feasible. Note that any minimum set that intersects all MUSes is an MCS. We have currently implemented one of the pathways proposed in [17] which, upon failure, uses FindMUS (a MUS enumerator available in MiniZinc) to obtain all MUSes, shows them to the user in different formats, and allows them to manually select an MCS, thus ensuring there is a solution if the constraints in the MCS are relaxed. Currently, users must directly modify these constraints to obtain a feasible instance. We are in the process of implementing the automatic relaxation and resolution process of [17].

Figure 8 shows two of the formats proposed in [17] to show MUSes, which we have implemented. The pane on the left shows the MUS-graph, a compact way of showing the conflicting constraints (left of graph), the MUSes they appear in (centre), and the objects they involve (right). Each circle on the left of the graph represents one conflicting constraint, whose colour connects it to the name shown in the legend on the top right of the graph. For the example shown in the figure these include *minimum production* and *maximum production*. Constraints in the same box appear in the same MUSes. The links connect each group of constraints to the MUSes they appear in, and each MUS to the objects in any of its constraints, which in this case are particular supply and demand locations. If users select a constraint on the left of the graph, the constraint and all the MUSes in which it appears are highlighted with a red frame. Users know they have selected an MCS if the selected constraints highlight all MUSes. A geo-network view of the conflicts is given by showing them on a map (Figure 8 right) of the supply/ demand locations. We overlay conflicts specific to a location with a frame in the colour of the conflict (e.g., light blue to show that this location is involved in a maximum production conflict), and with a coloured link for those involving a supply and demand location. Conflict overlays disappear when any of its constraints is selected in the MUS-graph, thus showing no conflict overlays once an MCS is selected.

6 Literature review

The problem of hydrogen production facility location and supply chain optimisation is widely studied; see Riera, Lima and Knio [16] for a recent survey of the field, both in terms of modelling the problem itself, and of optimisation strategies used. Our key takeaway from this survey is that there are myriads of different contexts in which to study the problem, from the

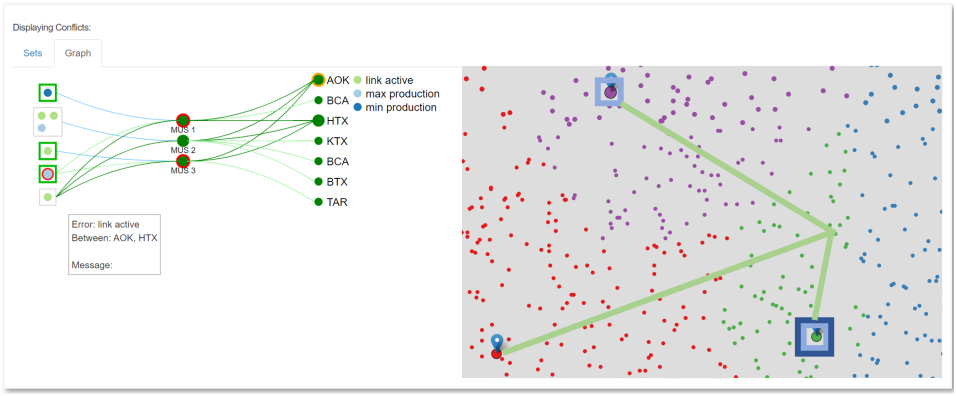

Figure 8 Two conflict visualisations: MUS-graph on the left, Geo-network on the right.

"micro" perspective of a single plant, to the "macro" decisions around the energy delivery system for entire countries (including hydrogen as one of many types of renewable fuels). As such, it is difficult to identify any two papers that study exactly the same mathematical model (e.g., for optimisation benchmarking purposes).

Nevertheless, several papers stand out as closely related to the problem studied here: Ingason, Ingolfsson and Jensson [10] study a electrolysis-based facility location problem for Iceland, although they ignore the operational scheduling of the plant thanks to the assumption of constant-rate renewable sources of electricity (hydro and geothermal). Likewise, Štádlerová et al. [20] study a facility location problem in Norway including uncertainty in the demand and operational considerations such as minimum turn-down rates, although not at an hourly resolution. Demirhan et al. [4] study a multi-fuel, multi-chemical (hydrogen, methanol and ammonia) facility location problem with hourly resolution on the operation of the production plant to capture variability in renewable energy generation from wind and solar. Finally, Kim and Kim [13] also study a joint facility location and hourly operation problem for green hydrogen, although they consider only one time period for facility location without planning the facility's expansion pathway. All these papers, however, focus mostly on the modelling expert's view and ignore the domain expert's one.

7 Conclusions and Future Work

This paper describes a Hydrogen Network Optimisation System (`HyNetOS`) designed to support energy companies in solving the complex combinatorial optimisation problem of producing and supplying hydrogen at minimum cost. `HyNetOS` integrates two supply network and facility operation models implemented in the high-level constraint modelling language MiniZinc, which natively supplies a range of tools to support the *human* decision-making process, such as finding a diverse sets of near-optimal solutions to present possible alternatives, and conflict resolution technology to explain infeasible instances. Further, `HyNetOS` supports our industry partner's decision-making process by means of a powerful and interactive user interface that allows them to view, change, and compare solutions, and rapidly iterate on "what-if" scenarios with efficiently solvable approximate versions of our models. To increase confidence in the quality of the model, we applied a strategy of redundancy and rapid prototyping of model changes to identify the most efficient formulation, and provide robustness measures, to produce resilient solutions against hydrogen production failures.

The HyNetOS system was quantitatively and qualitatively evaluated against an alternative implementation using a direct modelling approach. It was selected as the preferred system based mainly on its significantly higher scores in qualitative criteria such as maintainability, extensibility, user interface, code quality and technology stack, as well as its additional features (diversity and robustness).

While deployed, the system is in constant evolution with many avenues for future work. One of the most pressing and complex is extending the system to produce robust solutions against uncertainties in electricity prices and availability of resources using techniques such as stochastic programming [19], sensitivity analysis [2] and Predict+Optimise [14]. Others include the integration of extra functionality, such as the production of ammonia or the consideration of carbon intensity, the integration of a simulation system for detailed operational modelling on a high resolution time scale, and the integration of a plant layout optimisation system such as [1] to generate optimal hydrogen facility layouts.

References

1 Gleb Belov, Tobias Czauderna, Maria Garcia de la Banda, Matthias Klapperstueck, Ilankaikone Senthooran, Mitch Smith, Michael Wybrow, and Mark Wallace. Process Plant Layout Optimization: Equipment Allocation. In John Hooker, editor, *Principles and Practice of Constraint Programming*, pages 473–489. Springer, 2018. doi:10.1007/978-3-319-98334-9_31.

2 M. W. Dawande and J. N. Hooker. Inference-based sensitivity analysis for mixed integer/linear programming. *Operations Research*, 48(4):505–660, 2000. doi:10.1287/opre.48.4.623.12420.

3 Jip J. Dekker. MiniZinc Python, 2023. URL: https://minizinc-python.readthedocs.io/en/latest/.

4 C. Doga Demirhan, William W. Tso, Joseph B. Powell, and Efstratios N. Pistikopoulos. A multi-scale energy systems engineering approach towards integrated multi-product network optimization. *Applied Energy*, 281:116020, 2021. doi:10.1016/j.apenergy.2020.116020.

5 Ibrahim Dincer and Canan Acar. Review and evaluation of hydrogen production methods for better sustainability. *International Journal of Hydrogen Energy*, 40(34):11094–11111, 2015.

6 Graeme Gange, Jeremias Berg, Emir Demirović, and Peter J. Stuckey. Core-guided and core-boosted search for CP. In Emmanuel Hebrard and Nysret Musliu, editors, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 205–221, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-58942-4_14.

7 Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL: https://www.gurobi.com.

8 Q. Huangfu and J. A. J. Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, March 2018. doi:10.1007/s12532-017-0130-5.

9 IEA. Net zero by 2050. Technical report, IEA, Paris, May 2021. URL: https://www.iea.org/reports/net-zero-by-2050.

10 Helgi Thor Ingason, Hjalti Pall Ingolfsson, and Pall Jensson. Optimizing site selection for hydrogen production in Iceland. *International Journal of Hydrogen Energy*, 33(14):3632–3643, 2008. TMS07: Symposium on Materials in Clean Power Systems. doi:10.1016/j.ijhydene.2008.04.046.

11 Linnea Ingmar, Maria Garcia de la Banda, Peter J Stuckey, and Guido Tack. Modelling diversity of solutions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1528–1535, 2020.

12 IPCC. Climate change 2022: Impacts, adaptation, and vulnerability. Technical report, Cambridge University Press, Cambridge, UK and New York, NY, USA, 2022. Contribution of Working Group II to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change. doi:10.1017/9781009325844.

13 Minsoo Kim and Jiyong Kim. Optimization model for the design and analysis of an integrated renewable hydrogen supply (IRHS) system: Application to Korea's hydrogen economy. *International Journal of Hydrogen Energy*, 41(38):16613–16626, 2016. doi:10.1016/j.ijhydene.2016.07.079.

14 Jayanta Mandi, Emir Demiroviç, Peter J. Stuckey, and Tias Guns. Smart predict-and-optimize for hard combinatorial optimization problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1603–1610, 2020. doi:10.1609/aaai.v34i02.5521.

15 Nicholas Nethercote, Peter J Stuckey, Ralph Becket, Sebastian Brand, Gregory J Duck, and Guido Tack. MiniZinc: Towards a standard CP modelling language. In *Principles and Practice of Constraint Programming–CP 2007: 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007. Proceedings 13*, pages 529–543. Springer, 2007.

16 Jefferson A. Riera, Ricardo M. Lima, and Omar M. Knio. A review of hydrogen production and supply chain modeling and optimization. *International Journal of Hydrogen Energy*, 48(37):13731–13755, 2023. doi:10.1016/j.ijhydene.2022.12.242.

17 Ilankaikone Senthooran, Matthias Klapperstueck, Gleb Belov, Tobias Czauderna, Kevin Leo, Mark Wallace, Michael Wybrow, and Maria Garcia de la Banda. Human-Centred Feasibility Restoration in Practice. *Constraints*, 2023. (in publication). doi:10.1007/s10601-023-09344-5.

18 Shammamah Hossain. Visualization of Bioinformatics Data with Dash Bio. In Chris Calloway, David Lippa, Dillon Niederhut, and David Shupe, editors, *Proceedings of the 18th Python in Science Conference*, pages 126–133, 2019. doi:10.25080/Majora-7ddc1dd1-012.

19 Lawrence V. Snyder. Facility location under uncertainty: a review. *IIE Transactions*, 38(7):547–564, 2006. doi:10.1080/07408170500216480.

20 Šárka Štádlerová, Trygve Magnus Aglen, Andreas Hofstad, and Peter Schütz. Locating hydrogen production in Norway under uncertainty. In Jesica de Armas, Helena Ramalhinho, and Stefan Voß, editors, *Computational Logistics*, pages 306–321, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-031-16579-5_21.

21 Mark van den Brand and Jan Friso Groote. Software engineering: Redundancy is key. *Science of Computer Programming*, 97:75–81, 2015. Special Issue on New Ideas and Emerging Results in Understanding Software. doi:10.1016/j.scico.2013.11.020.

22 H. Paul Williams. *Model building in mathematical programming.* John Wiley & Sons, 5th edition, 2013.