Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

Optimal schedule generation for single-channel crude transfer using a multi-model approach

Aditya A. Paranjape[a,1], Mayank Baranwal[b,*], Satyavrat Wagle[b], Rushi Lotti[b], Sushanta Majumder[c], Anne-Laure Bullière[d]

[a] *Department of Aeronautics, Imperial College London, UK*
[b] *Tata Consultancy Services Research, Tata Consultancy Services Ltd, India*
[c] *Engineering and Industrial Services, Tata Consultancy Services Ltd, India*
[d] *TotalEnergies, France*

ARTICLE INFO

Article history:
Received 11 October 2021
Revised 10 January 2022
Accepted 13 February 2022
Available online 16 February 2022

Keywords:
Crude transfer scheduling
Reinforcement learning
Optimal control
Mixed-integer linear programming

ABSTRACT

This paper presents three techniques for scheduling for crude transfer between a port and a refinery on a single pipeline in the presence of stringent flow constraints. The three techniques are based on metaheuristics (business rules), mixed integer linear programming and reinforcement learning. In addition to comparing the three techniques, we also demonstrate how knowledge gleaned from one technique (in our case, the metaheuristics) can be used to design an algorithm based on another technique (in our case, reinforcement learning). A novel feature of our approach to reinforcement learning, in particular, is the use of low-fidelity, reduced-order simulators for training the scheduler and supporting it with a post-processor based either on business rules or on integer programming for ensuring compatibility with the constraints. The set of constraints considered here includes temporal restrictions on the use of the pipeline, flow constraints in the tanks that feed the column distillation units in the refinery, and the need to ensure a certain minimum residence time for crude in a given tank for dewatering.

1. Introduction

In this paper, we present algorithms for scheduling the movement of crude from an upstream port to a refinery. Crude comes in a variety of grades based on criteria such as sulfur and metal content. The challenge of scheduling the movement in the setting of this paper stems from a combination of four factors: the presence of a single pipeline serving all individual grades of crude, volume transfer constraints, particular facets of the production plan, and the allocation of tanks to individual crude grades. Although this paper considers a very specific refinery model, the set of challenges listed here could apply equally to other refineries. Therefore, the techniques designed in this paper can be applied or scaled readily to other refineries.

The problem addressed in the paper can be viewed as a special case of the larger problem of optimizing the operations of a network of machines. Unlike more traditional problems which seek to maximize the net productivity of the plant or other performance metrics (Wagle and Paranjape, 2020; Hubbs et al., 2020), our problem primarily concerns feasibility due to the tight volume transfer constraints. Nonetheless, most of the techniques developed in this paper would work equally well for more traditional problems where the set of feasible solutions affords further scope for optimization. Our algorithms can function as stand-alone schedulers in their own right or act as high-performance recommender systems to human schedulers.

1.1. Literature overview

The problem considered in the paper shares common ground with well-known problems in decision theory, such as supply chain optimization, flow-shop optimization Pinedo (2012) and dynamic resource allocation problems. Applications include manufacturing plants, fuel refineries, defence resource allocation Bertsekas et al. (2000) and even epidemics Brandeau et al. (2003). Most of the techniques developed for these problems, especially at a theoretically rigorous level, deal with the time of completion of a process (the makespan). These problems can also be viewed as combinatorial optimization problems, which lead one to explore programming techniques such as genetic algorithms or linear programming (LP) and their variants. When the plant is poorly char-

* Corresponding author.
E-mail addresses: a.paranjape@imperial.ac.uk (A.A. Paranjape), baranwal.mayank@tcs.com (M. Baranwal).
[1] On sabbatical from TCS Research.

https://doi.org/10.1016/j.compchemeng.2022.107732
0098-1354/© 2022 Elsevier Ltd. All rights reserved.

acterized or too complex to control analytically or using combinatorial optimization techniques, approaches based on approximate dynamic programming, or reinforcement learning (RL), can be invoked Zhang and Dietterich (1995, 1996); Dalal and Mannor (2015).

Although LP-based techniques (Pinto et al., 2000; Mendez et al., 2006) as well as model predictive control (MPC) (Al-Othman et al., 2008; Brandeau et al., 2003) have been applied successfully to schedule optimization problems, these techniques rely on knowing the plant model reasonably accurately (if not exactly). In our setting, on the contrary, we assume that the plant model is not known and needs to be learned in a suitable sense through trial-and-error. This places our problem firmly in the remit of approximate dynamic programming, or reinforcement learning. RL-based techniques have been proposed for scheduling problems in the context of the usual makespan minimization (Zhang and Dietterich, 1995; 1996) and even missile defence systems (Bertsekas et al., 2000).

MPC is a widely-used control technique for industrial plant operations. It consists of two essential elements: a receding horizon (RH) framework and a solver for the optimal control problem over a finite horizon. Traditional MPC uses the resulting control inputs in an *open-loop* fashion. Robustness is derived from the receding horizon framework. In contrast, RL provides a closed-loop *policy* for each time instant (Ernst et al., 2009) and derives robustness from exposure to "uncertainties" in the training regime.

Recent attempts to bring together MPC and learning have involved either learning an explicit system model that can be exploited by MPC (Williams et al., 2017), or using MPC to generate training sets for neural network-based policy generators which operate at a fraction of the computational cost of MPC (Zhang et al., 2016). RL techniques can be used to augment typical MPC algorithms (Negenborn et al., 2005), wherein MPC is used (on an approximate model) until the value function estimates required by RL are sufficiently reliable. Thereafter, the computationally simpler RL-generated policy can replace the one generated by MPC. Alternately, RL can be used to obtain a policy for the finite-horizon window so that the control inputs in each window can be computed in a closed-loop manner rather than being computed a priori and used in an open-loop manner (Wagle and Paranjape, 2020; Hubbs et al., 2020).

A related problem is representing information about extraneous variables, such as the production plan, for the purpose of training the policy. In this paper, we assume that the extraneous variables can be represented in terms of known "primitives." An alternate approach is to use *temporal moments*, as illustrated in our prior work (Wagle and Paranjape, 2020), to reduce the dimensionality of the training set for learning algorithms.

1.2. Contributions

In this paper, we develop three families of controllers for optimal scheduling of the flow channels in a refinery characterized by a single dominant flow channel between the port and the refinery. The three families are based on meta-heuristics (business rules), reinforcement learning (RL) and mixed-integer linear programming (MILP).

A key requirement is that the controller be able to work with a large class of crude receipt schedules and be robust to disruptions in crude receipt or downstream processing. We deal with these problems by using a receding horizon framework, and we train our RL algorithm to work with a large class of production plans which can be written as a concatenation of primitive strings.

Another contribution of our work is the use of domain-driven multimodel framework for deriving the optimal control laws. The multimodel framework consists of nested models, where the inner model is obtained by a simplifying abstraction of the outer model. These abstractions may be spatial, temporal or both. The simplest model lends itself to RL, especially by providing an appropriate state space. Intermediate models are suitable for deriving the heuristics and for training the RL agent. The outermost model, which is also the most detailed, is used for deriving an MILP-based standalone optimizer as well as an MILP-based post-processor for the RL agent.

The rest of the paper is organized as follows. We present the technical preliminaries in Section 2. The multimodel framework is presented in Section 3. The optimizers are presented in Section 4. Finally, numerical case studies are presented in Section 5.

2. Preliminaries: Receding horizon control

We are interested in sequential decision-making problems over time horizons $T \gg 1$. Let $\boldsymbol{s} \in \mathcal{S}$ and $\boldsymbol{u} \in \mathcal{U}$ denote the state and action variables. The spaces $\mathcal{S}$ and $\mathcal{U}$ are typically subsets of multidimensional real or integer spaces. The class of problems that our techniques can address are of the form

$$\begin{aligned} \text{Objective:}\quad & \max \sum_{k=1}^{T} f(\boldsymbol{s}[k], \boldsymbol{u}[k]) \quad \text{s.t.} \\ \text{Dynamics:}\quad & \boldsymbol{s}[k+1] = A\boldsymbol{s}[k] + \sigma_{\mathcal{T}}[k]B\boldsymbol{u}[k] + \boldsymbol{d}[k] \\ \text{Control:}\quad & \boldsymbol{u} = [u_1, \ldots, u_p]^\top, \quad u_i \in \{0,\ 1\} \\ \text{Admissibility:}\quad & \boldsymbol{c}_{\min} \le C\boldsymbol{s} + D\boldsymbol{u} \le \boldsymbol{c}_{\max} \end{aligned} \tag{1}$$

where $\mathcal{T} \subset \mathbb{R}$ consists of disjoint temporal intervals, $\boldsymbol{d}[\cdot]$ denotes fully or partially known external signals and $\sigma_{\mathcal{T}}[\cdot]$ is the indicator function

$$\sigma_{\mathcal{T}}[t] = \begin{cases} 1, & t \in \mathcal{T} \\ 0, & \text{otherwise} \end{cases}$$

The objective function $f(\cdot,\cdot) : \mathcal{S} \times \mathcal{U} \to \mathbb{R}_{\ge 0}$ may be nonlinear.

When information about $\boldsymbol{d}[k]$ is available over a finite time horizon (i.e., over a horizon $h \ll T$), a receding horizon approach is a natural way to solve the problem. We iterate over the following steps starting at time $t = 0$:

1. Solve the problem (1) with T set to h.
2. Take the computed actions for some time $\tau_h < h$.
3. Return to the first step.

It is assumed that, after τ_h steps, information about $\boldsymbol{d}$ is available for a further h steps.

Next, we briefly recall the relevant features of mixed-integer linear programming (MILP) and reinforcement learning (RL) in the context of solving Problem (1) with $T = h$. These approaches yield two different types of solution:

- The use of MILP leads to an *iterative solver* which must be run at the start of each planning horizon, and as many time as necessary thereafter.
- The use of offline RL involves one-off training that results in a *non-iterative formula* (also called a policy) parametrized in terms of $\boldsymbol{d}[k_0 : k_0 + h]$ (where k_0 denotes the start of the planning horizon). Depending on how it is employed, this policy may be invoked either once or on multiple instances until time τ_h has elapsed; its parameters are then adjusted to account for any new information about $\boldsymbol{d}$.

2.1. Mixed-integer linear programming

Linear programs and mixed-integer linear programs (MILPs) have long been used in the context of refinery management and scheduling optimization (Koenig, 1963; Baker and Lasdon, 1985; Gill, 1995; Göthe-Lundgren et al., 2002; Khor and Varvarezos, 2017). These include scheduling of crude oil unloading at the port (Lee et al., 1996), building piecewise linear models for optimal crude blending (Jia and Ierapetritou, 2003; Gao et al., 2015), optimizing energy efficiency (Wu et al., 2017), optimal production planning (Zhen et al., 2008; Uribe-Rodriguez et al., 2020), and short-term logistics optimization (Taherkhani et al., 2020; Misra et al., 2020). For a petroleum refinery aiming to optimize crude scheduling at a mode level, a zero-one programming technique that corresponds to binary selection of various crude modes can be modeled using linear constraints. Other constraints, such as, tank capacity constraints, simultaneous inflow/outflow constraints, and contiguity constraints can also be modeled as linear constraints. For instance, let $s[t]$ denote the state of a tank at instant t with a minimum and maximum capacities being 0 and $S_{\max}$, respectively. Let $r[t]$ be the amount of crude received in the tank at time t, while $u[t]$ is a binary variable, which indicates whether the crude is withdrawn at time t with a flow-rate of C units per instant. We describe the tank-state at the next instant as:

$$s[t+1] = s[t] + r[t] - C \cdot u[t],$$

which is linear in the decision variable $u[t]$, and thus the corresponding capacity constraints, $0 \le s[t+1] \le S_{\max}$ are also linear in the decision variable $u[t]$. Details on linear modeling of other constraints are described later in Section 4.2 in detail. Note that our MILP approach assumes a discrete-time representation of the refinery system. This is in contrast to event-triggered discrete-time formulation (Saharidis et al., 2009) where time between two events is assumed to evolve in a continuous manner. Discrete-time representation of plant dynamics is necessitated by the complexity of the underlying system (discussed in detail in Section 3), particularly associated with the inactivity of the shared pipeline for a prolonged duration, availability of the tanks for unloading (due to dewatering constraints), minimization of demurrage costs and other hydraulic constraints. A suitable choice of the associated objective function is to maximize the throughput or minimize the total shortfall in production at the CDUs, which can again be modeled using linear functions of the underlying decision variables. Linear modeling of constraints and objective function entails use of MILPs for optimizing mode level crude scheduling.

While the problem of optimal scheduling can be addressed through a suitable MILP formulation, solving large-scale MILPs can be computationally prohibitive if not modeled appropriately (Kannan and Monma, 1978; Papadimitriou, 1981; Phillips et al., 2015). The computational complexity of MILPs arising from exploring combinatorially many candidate solutions can be alleviated through approaches, such as, branch-and-bound (Ross and Soland, 1975), or Benders decomposition (Codato and Fischetti, 2006) to some extent; however, carefully formulating the underlying model is still the primary source of obtaining significant computational speed-up (Trick, 2005; Klotz and Newman, 2013). In this paper, we develop novel and computationally tractable linear formulations of optimal scheduling of refinery operations purely from an (a) MILP perspective, and (b) reinforcement learning (RL)-guided MILP framework. We further discuss the advantages and disadvantages of the two approaches. Additionally, an MILP formulation for generating optimal production plan that takes into account several crude-level constraints is also developed. It is shown that while MILP still continues to be the first line of attack for optimal scheduling in refinery systems, several nonlinear constraints and scale of the problem may prohibit the use of MILP algorithms, thus, requiring metaheurisitc approaches, such as, RL to guide efficient MILP exploration.

2.2. Reinforcement learning

Reinforcement learning (RL) techniques are an approximate way to solve problems of the form

$$\max \sum_{k=0}^{T-1} \gamma^k f(\boldsymbol{s}[k], \boldsymbol{u}[k]), \quad \boldsymbol{s}[k+1] = g(\boldsymbol{s}[k], \boldsymbol{u}[k]) \tag{2}$$

where $0 < \gamma \le 1$ is called the discount factor. The dynamics need not be Markov; however, a larger range of techniques apply to Markov systems, along with the accompanying theoretical guarantees. RL techniques attempt to solve the Bellman equation (or its action-value variant)

$$V(\boldsymbol{s}) = \max_{\boldsymbol{u}} E(f(\boldsymbol{s}, \boldsymbol{u}) + \gamma V(\boldsymbol{s}'))$$

where $\boldsymbol{s}' = g(\boldsymbol{s}, \boldsymbol{u})$. Here E denotes the expected value in a stochastic setting, and the actual value in a deterministic setting. The end-product is a policy $\pi : \mathcal{S} \to \mathcal{U}$ which yields the optimal control signal $\boldsymbol{u} = \pi(\boldsymbol{s})$ to be taken at any given state.

When the system in (2) depends on exogenous signals $\boldsymbol{d}$, it is clear that the policy π also depends on $\boldsymbol{d}$. This dependence can be captured, at least approximately, in a parametrized form which we denote as $\pi(\cdot\,; \boldsymbol{d})$. Notice that the policy is trained exactly once with a sufficiently rich combination of $(\boldsymbol{s}, \boldsymbol{d})$; thereafter, in a receding-horizon setting, only the parameters for $\boldsymbol{d}$ are adjusted when new information about $\boldsymbol{d}$ becomes available.

In this paper, we consider the case where the value function as well as the policy are encoded through dedicated neural networks; i.e., the so-called actor-critic architecture. A schematic is shown in Fig. 1.

2.2.1. Simulator-based training

Simulator-aided training is the primary mode for training a policy in an offline setting. Here, the RL agent has access to only a simulator, while relevant domain knowledge reflects in the choice of the variables that enter the neural network as inputs, as well as in the number and structure of hidden layers.

2.2.2. Supervised training

An alternate approach to train the RL agent, especially in the early stages of training, is to employ meta-heuristics or historical performance data to estimate the state trajectory and the value function for a given sequence of actions. Unlike the schematic of Fig. 1, which is relevant to simulator-based training, the policy network is trained by directly comparing the computed actions with those taken historically or recommended by the meta-heuristics. This “priming” can reduce the time required to train the network in a simulator-based model and also bias the network adequately towards well-accepted (albeit possibly non-optimal) solutions when there are multiple optimal candidates.

3. Multi-level modeling

3.1. Port-refinery complex

The port-refinery complex is illustrated in Fig. 2. It consists of tanks in the port area (labeled port) and a refinery complex, (labeled refinery), with its tanks and column distillation units (CDUs). The port and refinery areas are linked by a single pipeline, denoted by PRL. The capacity of each tank is marked in the figure.

The nomenclature employed for the tanks follows the following notation: tank ZGcn is in zone Z (port or refinery) and holds crude of grade G. The number cn denotes a unique identifier; in particular, for the refinery tanks, c denotes the numerical identifier of the CDU and n represents the order of the tank.

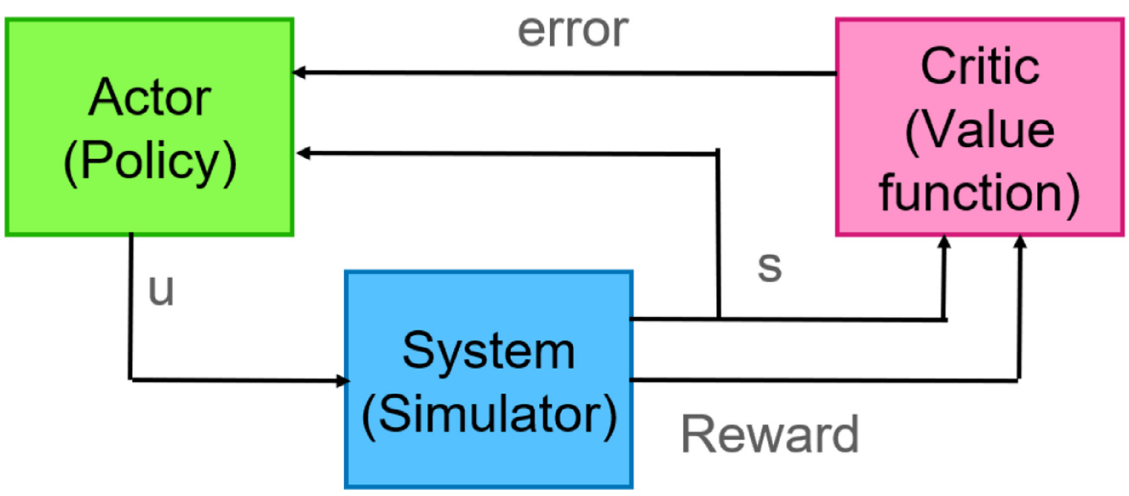


Fig. 1. A schematic of an actor-critic architecture.

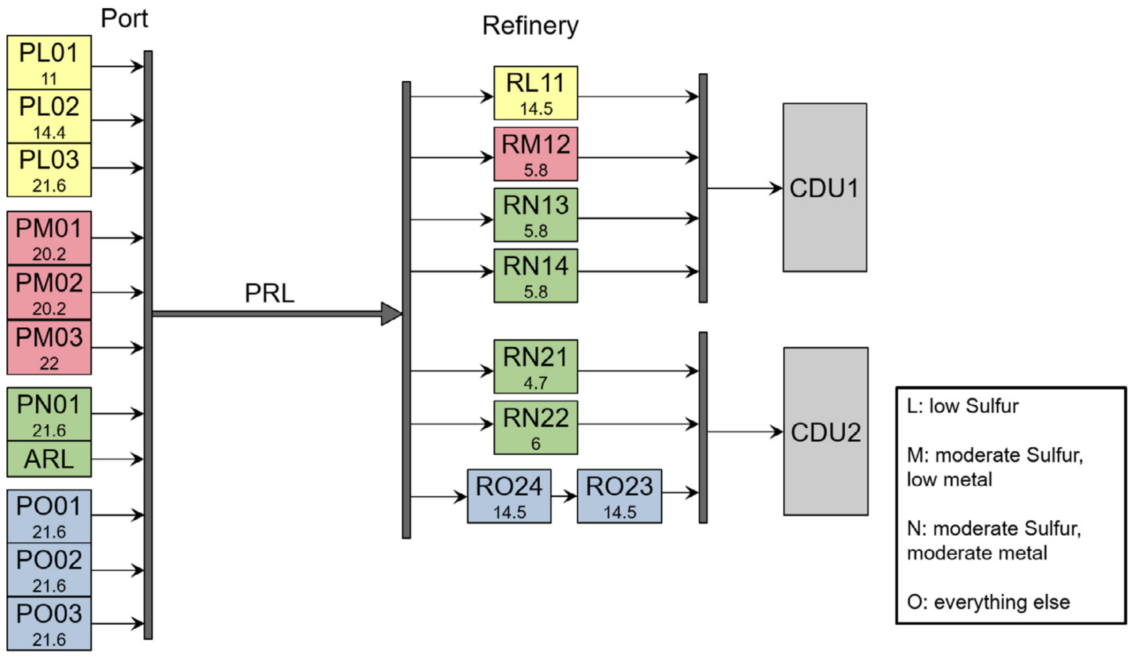


Fig. 2. A schematic map of the port-refinery complex considered in the paper. Additional restrictions (e.g., RN21 and RN22 feeding CDU2 only hold N-type crude) are listed in the text..

The refinery processes four grades of crude: L-type, M-type, N-type and O-type. There is further classification within these grades (or modes) based on the metal content and the specific weight of the crudes, but we do not consider these properties here. The tanks in port and refinery are colored based on the grades that they are allowed to hold. For instance, the tank RL11 to CDU1 is only allowed to hold L-type.

We note a key constraint regarding the tanks: other than tank RO23, no tank can be filled and drained simultaneously.

We measure crude in metric tonnes. Therefore, 1 unit of crude equals 1000 metric tonnes. The flow rate of a pipe is measured in metric tonnes per hour; i.e., units/hr in our notation. The CDUs, CDU1 and CDU2, are assumed to consume crude at their design capacity of 0.22 units/hr and 0.32 units/hr, respectively. In practice, the exact capacity can be adjusted based on flow constraints. Here, we do not permit as much.

The port area receives crude from ships, with the exception of ARL (which is of N-type grade). Crude received from ships is stored in a suitable tank, depending on the grade of the crude, and available for transfer to the refinery only after 24 hrs of dewatering. ARL, on the other hand, is sourced from an upstream provider, already dewatered, and can be drawn when needed in packets of 4.45 units.

PRL transfers crude at a rate of 1.045 units/hr. We split time into 38-hr blocks. PRL is available for our use only during the first 19.8 hrs of a 38 hr block; for the remaining 18.2 hrs, it is not available in any event. With this constraint, we can easily verify that the amount of crude transferred by the PRL in a 38 hr block (20.7 units) is equal to the amount processed by the two CDUs during the same time period.

This appears to pose a straight-forward replenishment problem as far as scheduling is concerned; however, the challenge stems from the processing of L-type crude. Note that RL11, with a capacity of 14.5 units, is the only tank in the refinery area for holding L-type. It is economically desirable to process L-type continuously over two successive days; in other words, it is drawn continuously over 48 hrs with no time for replenishment during this period. This is evident in the sample production plan introduced later in the paper (see Fig. 6).

3.2. Sim3: Master model for flow in the port-refinery complex

Let $t = 0$ denote an arbitrary starting time, and let 1 time unit correspond to 15 minutes. Let $s_{(\cdot)}[t] \in \mathbb{R}$ denote the level of a tank at a time instant t. The subscript is matched to the tank identifier in Fig. 2. Let $p_{(\cdot)}[t] \in \{0, 1\}$ denote whether its outgoing pipe is open (1) or closed (0), and let $u_{(\cdot)}$ denote likewise for its incoming pipe.

The dynamics of the flows in the port-refinery complex are described by the following equations:

$$\begin{aligned}
\text{Capacity constraints}: &\ u_{(\cdot)} = 0 \ \text{ if a tank is almost full} \\
\text{Capacity constraints}: &\ p_{(\cdot)} = 0 \ \text{ if a tank is almost empty} \\
\text{port tanks}: &\ s_{(\cdot)}[t+1] = s_{(\cdot)}[t] + 0.33u_{(\cdot)} - 0.26p_{(\cdot)} \\
\text{refinery tanks to CDU1}: &\ s_{(\cdot)}[t+1] = s_{(\cdot)}[t] + 0.26u_{(\cdot)} - 0.056p_{(\cdot)}
\end{aligned} \tag{3}$$

$$\text{T24}: \ s_{RO24}[t+1] = s_{RO24}[t] + 0.26u_{RO24} - 0.08p_{RO24} \tag{4}$$

$$\begin{aligned}
\text{RO23}: &\ s_{RO23}[t+1] = s_{RO23}[t] \\
&\ +0.08(u_{RO23} - p_{RO24}) \\
\text{RN21,RN22 to CDU2}: &\ s_{(\cdot)}[t+1] = s_{(\cdot)}[t] + 0.26u_{(\cdot)} - 0.08p_{(\cdot)} \\
\text{Filling and draining}: &\ p_i u_i = 0, \quad \forall i \neq \{\text{RO23}\} \\
&\ p_i p_j = \delta_{ij}, \quad \forall (i,j) \in \text{refinery} \\
\text{PRL on}: &\ p_i = u_j = 1, \\
&\ \text{iff PRL transfers from tank i to j}
\end{aligned} \tag{5}$$

where δ_{ij} is the Kronecker delta. By *almost full*, we mean that the tank can be filled in less than 15 minutes with the nominal in-flow rate, and likewise for almost empty.

3.3. Sim0, Sim1 and Sim2: Abstract models

An inspection of Sim3 reveals a large action space. If we constrain our attention to the PRL alone, it is clear that there are 79 actions to be chosen in each 38 hour block (4 decision points per hour for 19.8 hours). Over 30 calendar days, this translates to nearly 1496 decision points. To simplify the planning problem, we take the following steps:

1. We redefine the state and action spaces, as explained presently. In effect, the action space reduces to calculating just 5 decision variables in a 38 hour block. This yields an aggregate crude transfer schedule.
2. We post-process the aggregate transfer schedule using a higher-fidelity simulator to ensure that the schedule complies with tank capacity constraints at each instant in time.

In practice, this is achieved using two sets of abstractions.

1. We group tanks that hold the same grade of crude into *zones* in each area. The capacity of a zone is the sum of the capacities of individual tanks. For instance, RO23 and RO24 are grouped into the O-type zone, while tanks RL11 (L-type) and RM12 (M-type) are single-tank zones. This obvious simplification creates the following problem. Consider the N-type zone (RN13 and RN14) feeding CDU1. If one of them is full and the other is actively feeding CDU1, it might create the false impression of spare holding capacity in that zone and lead a controller to erroneously schedule the transfer of N-type crude to CDU1.
2. We re-calibrate the duration of a time instant, so that instants k and $k+1$ are separated by 38 hours. We additionally define the instant k' located 19.8 hrs between k and $k+1$. This means that there are just 19 planning instants in a 30 day period. This simplification also comes with a risk. Continuing with the example of the N-type zone feeding CDU1, suppose now that the CDU1 is fed N-type only for the first 18 hours of a 19.8 hour sub-block. Had the feeding tank been full at the start of the 19.8 hr block, it leaves space for nearly 18 units, whereas only 9 units can be actually transferred in the remaining 1.8 hrs. The controller should have some awareness of this "adjusted" spare capacity, but it is not obvious how this notion may be formalized once tanks are replaced with zones.

By using one or more of these abstractions, we design three simulators which are listed in Table 1. Of these, the governing equations for Sim1 and Sim2 are similar to those of Sim3 in Eq. (3), with the tanks replaced by zones and constraints of the form $p_{(\cdot)}u_{(\cdot)} = 0$ relaxed for all zones except RL11 and RM12. The zone-based abstraction is shown schematically in Fig. 3.

In Sim0 and Sim1, we have replaced port tanks with an infinite reservoir. This is done with the assumption that the crude receipt schedule (supplemented by ARL packets) is fundamentally adequate for meeting the production schedule. An RL agent trained using Sim0, therefore, needs a post-processor to ensure that tank capacities are respected and N-type crude flow is grouped into packets of 4.45 units when ARL is to be drawn.

The last column of Table 1 lists the primary (but not exclusive) utility of each specific model or simulator. The RL agent is trained using Sim0 and inherits its state and action space. Sim1 is used later to design the heuristic optimizer. Sim2 is used as part of the training of the RL agent, in order to evaluate the rewards and penalties accurately, and includes a post-processor to convert the actions recommended by the agent (which are based on Sim0) into those required in Sim2 (15 minute slots). Finally, Sim3 is used for the final validation of the RL agent, and for designing the MILP algorithm.

The dynamics for Sim0 are described by the following equations, with the understanding that 1 instant lasts 38 hours.

$$\begin{aligned}
\boldsymbol{s}[k'] &= \text{clip}(\boldsymbol{s}[k] + \boldsymbol{u}[k:k'] - \boldsymbol{p}[k:k']) \\
\boldsymbol{s}[k+1] &= \text{clip}(\boldsymbol{s}[k'] - \boldsymbol{p}[k:k'])
\end{aligned} \tag{6}$$

$$\sum_{i=1}^{5} u_i[k:k'] \leq 20.7 \tag{7}$$

where $\boldsymbol{s} \in \mathbb{R}^5$ denotes the states of the refinery *zones*. The control input $\boldsymbol{u}[k] \triangleq \boldsymbol{u}[k:k'] \in \mathbb{R}^5$ (with components $u_i[\cdot]$) denotes the amount of crude transferred to each of the five zones on the refinery side, while $\boldsymbol{p}[k:k'] \in \mathbb{R}^5$ denotes the amount of crude consumed by the CDUs during that same period. The 'clip' function limits the zone levels to between 0 and the maximum limit.

4. Optimization algorithms: Formulation

In this section, we will present three optimization algorithms. The first family consists of an algorithm based on heuristics (also referred to as business rules). The second consists of an algorithm based on mixed-integer linear programming (MILP), while the third consists of algorithms based on reinforcement learning (RL).

4.1. Heuristics (business rules)

Business rules may be viewed, generally, as a combination of analytical rules based on simplified models and common-sense

Table 1
Hierarchically-organized list of simulators .

Model/ Simulator	Time ticks	Port	Refinery	Utility
Sim0	38 hrs	∞	Zone	Design of RL agent
Sim1	15 min	∞	Zone	Design of heuristics-based optimizer
Sim2	15 min	Zone	Tank	Evaluation of the RL agent
Sim3	15 min	Tank	Tank	Design of MILP

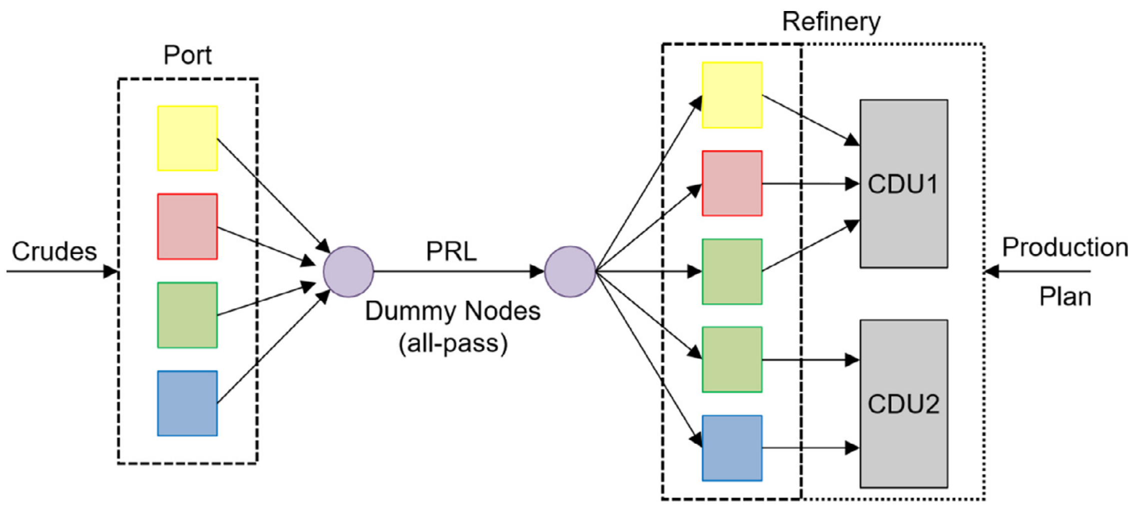


Fig. 3. The equivalent zone-based map of the refinery. We have represented the PRL by dummy nodes in this schematic. The four colors denote L-type (yellow), M-type (pink), N-type (green) and O-type (blue) crude.. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

rules grounded in experience. Business rules are, by design, effective and robust but not always optimal. In our setting, they serve two purposes: (i) they set a baseline for formal optimization algorithms, and (ii) they can be used for priming the RL algorithms, as explained earlier.

The business rules employed in the paper are based on Sim0 and Sim1: they draw on the state and action spaces of Sim0 and prescribe the volume of each crude to be transferred over a 19.8 hr block, and then use Sim1 to order the crude transfer slots and vacate individual slots to prevent overflows.

The algorithm based on business rules is detailed in Algorithm 1. We define the following quantities based on the production demand. Recall that a block of 38 hrs is split into 152 units of 15 min each; moreover, only the first 79 units are available for crude transfer. Let t_0 denote the starting time of a 38 hr block. Then, we define:

$$\begin{aligned} \boldsymbol{m}_{20} &= \boldsymbol{p}[t_0 : t_0 + 79] \\ \boldsymbol{m}^e_{38} &= \boldsymbol{p}[t_0 : t_0 + 152] + \text{additional terms} \\ \boldsymbol{m}_{38:76} &= \boldsymbol{p}[t_0 + 152 : t_0 + 304] \end{aligned} \tag{8}$$

where "additional terms" are included for L-type and M-type based on the following motivation. On any calendar day starting during a 38 block, the volumes required to meet the requirement for the entire window of production (typically 24 hrs for M-type and 48 hrs for L-type) must be provided before the 19.8 hr slot ends. The additional volume is added to obtain $\boldsymbol{m}^e_{38}$ (where the superscript 'e' stands for extended). We clarify that no such terms are needed for N-type and O-type.

Next, we derive a few upper limits on the amount of crude that can be transferred over the 19.8 hrs:

$$\begin{aligned} \text{Necessary for preventing overflows}: u_i &\le s_{i,\max} + m_{20,i} - s_i \\ \text{Crudes with a single refinery tank}: u_i &\le 1.045\,\text{clip}(19.8 - \tau_i) \end{aligned} \tag{9}$$

Algorithm 1 Business rules for scheduling.

```
1: Initialize: bids $\boldsymbol{b} \leftarrow 0$
2: Compute $\boldsymbol{m}_{20}$, $\boldsymbol{m}^e_{38}$ and $\boldsymbol{m}_{38:76}$ using (7)
3: $\boldsymbol{b} \leftarrow \text{clip}(\boldsymbol{m}^e_{38} - \boldsymbol{s})$, maximumvalues given by (8)
4: if $\Sigma_{\boldsymbol{b}} = \sum_{i=1}^{5} b_i > 20.7$ then
5:     Normalize: $\boldsymbol{b} \leftarrow 20.7(\boldsymbol{b}/\Sigma_{\boldsymbol{b}})$; update$\Sigma_{\boldsymbol{b}} \leftarrow 20.7$
6: end if
7: Compute remaining PRL capacity: $\sigma = 20.7 - \Sigma_{\boldsymbol{b}}$
8: Compute projected level of each zone: $\boldsymbol{s}_{38} = \boldsymbol{s} + \boldsymbol{b} - \boldsymbol{p}[t_0 : t_0 + 152]$
9: Compute projected spare capacity: $\boldsymbol{e} = \boldsymbol{s}_{\max} - \boldsymbol{s}_{38}$
   {Allocate some spare capacity to M-type and L-type (experience-based heuristic)}
10: Compute amount to be transferred: $a = \min(\boldsymbol{e}_{38,M-type}, m_{38:76,M-type}, \sigma)$
11: Update: $b_{M-type} \leftarrow b_{M-type} + a$, $\sigma \leftarrow \sigma - a$, $e_{38,M-type} \leftarrow e_{38,M-type} - a$
12: Repeat for L-type
13: While $\sigma > 0$: repeat for other crudes in descending order of $\boldsymbol{m}_{38:76}$;update $\boldsymbol{b}$ and $\boldsymbol{e}$
14: If $\sigma > 0$: fill M-type, N-type (CDU1), N-type(CDU2) O-type and L-typeuntil full (subject to bounds in (8) or while $\sigma > 0$; update $\boldsymbol{b}$
   {Likely if simulation starts with all tanks close to full:}
15: If $\sigma > 0$ remains: record possible unused capacity
16: Output: $\boldsymbol{u} = \boldsymbol{b} =$ amounts to be transferred during 19.8 hrs
17: Create a 15 min-level schedule using Sim1 or Sim2 (see Algo 2)
```

The first bound is applicable to all crudes and intended to prevent an overflow. Note that it is only a necessary condition for reasons which will become evident later.

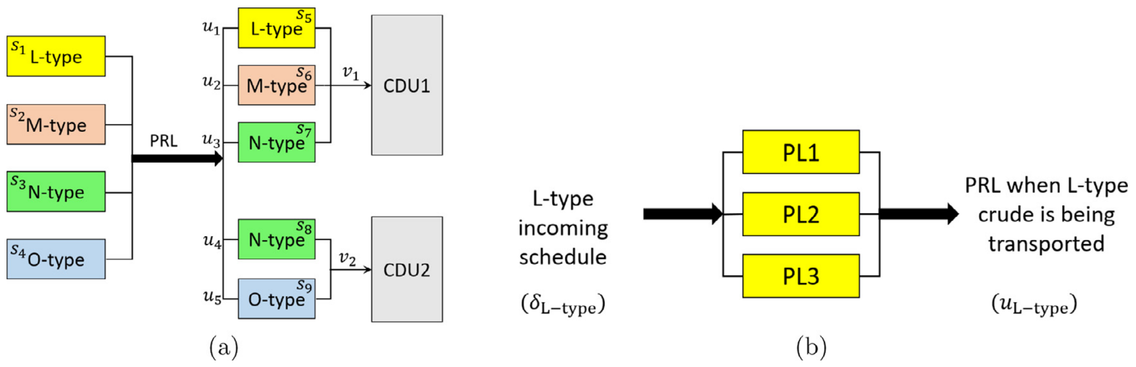


Fig. 4. Schematics of the proposed (a) primary, and (b) secondary MILP formulations. The primary MILP works at the zone-level and optimizes the schedule for crude transfer through the PRL. The secondary MILP builds upon the solution from primary MILP for optimal tank assignment.

The second constraint above applies only to crudes with a single holding tank in the refinery area, it is clear that we cannot transfer any volume while its tank is feeding the CDU. If $\tau_i \leq 19.8$ denote the total time during the 19.8 hr block when the i^{th} crude grade is processed by the CDU, it is evident that only $19.8 - \tau_i$ hrs are available for transferring that crude.

The crude volumes obtained using Sim0 need to be broken down into 15 min slots. We start with L-type, although the high-level ordering can be refined easily (e.g., order the grades as per their consumption on present day and the future). If feasible, up to 0.261 units of crude are transferred during a 15 min slot; else, the next grade is considered. If no grade is feasible, the slot is allowed to remain fallow. This is repeated every 15 min until the end of the 19.8 hr transfer block.

4.2. Mixed integer linear programming

In this section, we formulate a series of MILPs in an hierarchical fashion for scheduling transfer of crude oil in order to maximize the total throughput at the CDUs. However, before we lay out the exact MILP formulation, we must also look at the challenges that restrict us to formalize a single MILP in order to incorporate several feasibility constraints. The flows in and out of tanks are modeled in a discrete-time framework at a temporal resolution of 15 minutes. For a production plan that spans over 30 days, there is a total of nearly 19 blocks, each amounting to 38 hours. Additionally, the first 19.8 hours of a 38-hour block are used for transporting crude through the pipeline. For the chosen time-resolution, this amounts to a total of 79 sub-blocks within each 38-hour window, i.e., there is a total of $79 \times 19 = 1501$ time instants necessitating decision making by the operator. Recall that the port-refinery complex being addressed in this paper consists of 18 tanks (see Fig. 2). If the decisions are made individually at the tank level during each of the 1501 time instants, the simplest of the MILP formulations would require at least a total of $1501 \times 18 = 27018$ binary decision variables. Furthermore, modeling the tank capacity constraints alone introduce an aggregate of $2 \times 30 \times 24 \times 4 \times 18 = 103680$ linear constraints (2 for min and max capacity constraints, 30 days, 24 hours, 4 decision points per hour, and 18 tanks). Even the simplest of the MILPs comprising of nearly 27k decision variables and 100k linear constraints are practically intractable, let alone incorporating complex constraints, such as, dewatering and contiguity constraints. Thus, it is computationally prohibitive to formulate MILPs at the tank levels and we, therefore, restrict ourselves to the abstraction of Sim3.

Abstraction at the level of Sim3 significantly reduces the number of decision variables and linear constraints (easily by three fold), since the Sim3 abstraction works at the zone-level requiring decision making at the five refinery zones (see Fig. 3). However, the number of capacity constraints, along with imposing contiguity and dewatering constraints at the chosen granularity of fifteen minutes necessitates further decomposition of the optimization problem into hierarchically simpler MILPs, each of which can be solved efficiently on simpler compute resources. The proposed decomposition is based on the key observation that the original problem is inherently a feasibility problem, with an added objective of minimizing the shortfall at the CDUs. The constraints are decoupled into zone-level constraints (capacity constraints on zones) and tank-level constraints (capacity and dewatering constraints on tanks). The advantage is that solving the zone-level scheduling problem at the abstraction of Sim2 (or Sim3) fixes the schedule for crude transfer through PRL (see Fig. 2). Once the zone-level crude transfer decisions get locked, it simply remains to address the tank assignment problem with capacity and dewatering constraints. For instance, solving the zone-level optimization problem would fix the transfer schedule for the L-type mode. Once the transfer times for the L-type crude are known, it suffices to optimize crude unloading at tanks PL01, PL02 and PL03 upon arrival and withdrawal from them after wards such that the flow in and out of tanks matches the crude receipt schedule and PRL transfer schedule for L-type (obtained by solving the zone-level MILP). Thus, the problem of assigning tanks for accommodating crude receipt and withdrawal can be handled independently for different crudes, resulting in a set of decoupled and computationally tractable MILPs. Below we describe the proposed hierarchical nature of MILP formulations in detail.

4.2.1. Primary MILP

As discussed previously, the purpose of the primary MILP is to obtain optimal schedule for crude transfer through the PRL at the zone-level. The problem of tank assignments for storage and withdrawal are disregarded at this stage and addressed subsequently through secondary MILPs. To this end, we denote $s_i[t]$ to denote the current state of the i^{th}-zone with an overall maximum capacity of $S_{i_{\max}}$. The setup for the primary MILP is shown in Fig. 4a. The capacity of a zone is defined as the cumulative capacity of the corresponding tanks. For instance, the capacity of the L-type zone on the port side (Fig. 4a) is the cumulative capacity of the tanks PL01, PL02 and PL03 (Fig. 2). There is a total of nine zones (four on the port side, five on the refinery side). Let T be the total length of the planning horizon. Binary variables $v_1[t]$ and $v_2[t]$ are used to indicate activity of CDU1 and CDU2, respectively at time t. Finally, we use binary variables $u_i[t]$ to indicate transfer of crude to the corresponding refinery zones at time t (see Fig. 4a). (P-MILP) describes the associated MILP for optimizing schedule of crude transfer through the PRL.

$$
\begin{array}{ll}
\text{Minimize:} & \left(T-\sum_{t=1}^{T}\nu_1[t]\right)+\left(T-\sum_{t=1}^{T}\nu_2[t]\right) \\
\text{s.t., Capacity constraints:} & 0 \le s_i[t] \le S_{i_{\max}}, \quad \forall i \in \{1,\dots,9\} \\
\text{Mass-balance constraints:} & \begin{cases} s_i[t+1]=s_i[t]+r_i[t]-C_{\text{PRL}}\cdot u_i[t] & \forall i\in\{1,2\} \\ s_3[t+1]=s_3[t]+r_3[t]-C_{\text{PRL}}\cdot(u_3[t]+u_4[t]) & \\ s_4[t+1]=s_4[t]+r_4[t]-C_{\text{PRL}}\cdot u_5[t] & \\ s_{4+i}[t+1]=s_{4+i}[t]+C_{\text{PRL}}\cdot u_i[t]-C_1\eta_i[t]\cdot\nu_1[t] & \forall i\in\{1,2,3\} \\ s_{4+i}[t+1]=s_{4+i}[t]+C_{\text{PRL}}\cdot u_i[t]-C_2\eta_i[t]\cdot\nu_2[t] & \forall i\in\{4,5\} \end{cases} \\
\text{Exclusivity constraints:} & \sum_{i=1}^{5} u_i[t]=1 \\
\text{Min. up-time constraints:} & \sum_{\tau=t}^{\min\{t+\text{MUT}-1,T-1\}} u_i[\tau]-(u_i[t]-u_i[t-1])\ge 0, \quad \forall i\in\{1,\dots,5\} \\
\text{Min. down-time constraints:} & \sum_{\tau=t}^{\min\{t+\text{MDT}-1,T-1\}} 1-u_i[\tau]-(u_i[t-1]-u_i[t])\ge 0, \quad \forall i\in\{1,\dots,5\} \\
\text{Integer constraints:} & u_i[t]\in\{0,1\}, \quad \forall i\in\{1,\dots,5\}, \qquad \nu_1[t],\nu_2[t]\in\{0,1\}
\end{array}
\tag{P-MILP}
$$

Here C_{PRL}, C_1 and C_2 represent the flow rates through the PRL and CDUs, respectively. We use $\eta_i[t]$ to represent (binary) activity of the i^{th}-zone on a given day. For instance, a value of $\eta_2[t]=1$ indicates that CDU1 is required to process M-type crude (from zone-2) at the time instant t. Note that $\eta_i[t]$s are specifications and not the decision variables themselves. Similarly, crude incoming schedules are specified by quantities $\{r_i[t]\}$. The primary objective of the MILP in (P-MILP) is to minimize the total shortfall at the CDUs. Since the binary variables $\nu_1[t]$ and $\nu_2[t]$ denote the instantaneous status (activity) of the CDUs, the total number of events of inactivity is captured in the objective. Capacity constraints are easy to understand; the states $s_k[t]$ are obtained using simple mass-flow balance. Exclusivity constraints are added to ensure that only one kind of crude can be transported through the PRL pipeline at any instant. Minimum up/down-time constraints are commonly employed in the unit commitment literature (Kazarlis et al., 1996), and provide a simple and interesting way to ensure contiguity of flow. They can be interpreted as follows: Let us suppose that the i^{th} decision variable gets activated at time t, i.e., $u_i[t-1]=0$ and $u_i[t]=1$, then the minimum up-time constraints force the subsequent MUT number of variables to be unity. Thus, if the PRL pipeline begins to admit flow to one of the refinery tanks at any time t, then it must continue to do so at least for the remaining MUT time instants. The minimum down-time constraints can be interpreted similarly.

4.2.2. Secondary MILP

Solving the primary MILP addresses the optimal scheduling of crude transfer through the PRL pipeline at the zone-level. Once the optimal schedules for the transfer of different crude types becomes known, the problem of optimal tank assignment gets decoupled based on the corresponding crude type. As such, the objective of the secondary MILPs is to assign receipt and withdrawal of each crude type to the corresponding tanks taking into account their capacities and dewatering constraints. The withdrawal schedule from port tanks should be in accordance with the optimal crude transfer schedule through the PRL obtained by solving the primary MILP. Similarly, the arrival schedule to refinery tanks should match up with the corresponding crude transfer schedule through the PRL. This is depicted in Fig. 4b for the L-type tanks located at the port side. Binary variables $e_l[t]$ are used to indicate whether the crude receipt at time t is being allocated to the l^{th}-tank. Similarly, binary variables $w_l[t]$ indicate if the l^{th}-tank is being emptied at time t. The formulation below describes the associated secondary MILP for the feasible tank assignment problem.

$$
\begin{array}{ll}
\text{Minimize:} & 0 \\
\text{s.t., Capacity constraints:} & 0 \le s_{\text{tank}_l}[t] \le S_{\max_{\text{tank}_l}}, \qquad \forall l \in \{1,\dots,n_i\} \\
\text{Mass-balance constraints:} & s_{\text{tank}_l}[t+1]=s_{\text{tank}_l}[t]+C_{\text{in}}\cdot e_l[t]-C_{\text{PRL}}\cdot w_l[t], \qquad \forall l\in\{1,\dots,n_i\} \\
\text{Exclusivity constraints:} & \sum_{l=1}^{n_i} e_l[t]=\delta_i[t], \qquad \sum_{l=1}^{n_i} w_l[t]=u_i[t] \\
\text{Dewatering constraints:} & (1-w_l[\tau])-e_l[t]\ge 0, \qquad \forall\tau\in\{t,t+1,\dots,t+\Delta-1\}, \quad \forall l \\
\text{Min. up-time constraints:} & \sum_{\tau=t}^{\min\{t+\text{MUT}-1,T-1\}} z_l[\tau]-(z_l[t]-z_l[t-1])\ge 0, \qquad z_l\in\{e_l,w_l\} \\
\text{Min. down-time constraints:} & \sum_{\tau=t}^{\min\{t+\text{MDT}-1,T-1\}} 1-z_l[\tau]-(z_l[t-1]-z_l[t])\ge 0, \qquad z_l\in\{e_l,w_l\} \\
\text{Integer constraints:} & e_l[t], w_l[t]\in\{0,1\}
\end{array}
\tag{S-MILP}
$$

It is worth noting that the secondary MILP is a feasibility problem (and not an optimization problem). Consequently, the optimization objective in (10) is simply set to a constant. The number of tanks associated with the i^{th} crude-type is depicted by n_i. The exclusivity constraints ensure that at least one of the tanks is always available for crude receipt (when $\delta_i[t]=1$) and withdrawal (when $u_i[t]=1$). Dewatering constraints ensure that if the crude is received at the l^{th}-tank at time t, then there is no withdrawal from it at least for the next Δ time instants, so as to allow for the removal of water. Here, Δ is chosen to be 96 (24×4). The minimum up/down-time constraints ensure that the assignment generated by the secondary MILP is contiguous with minimal switching between the tanks.

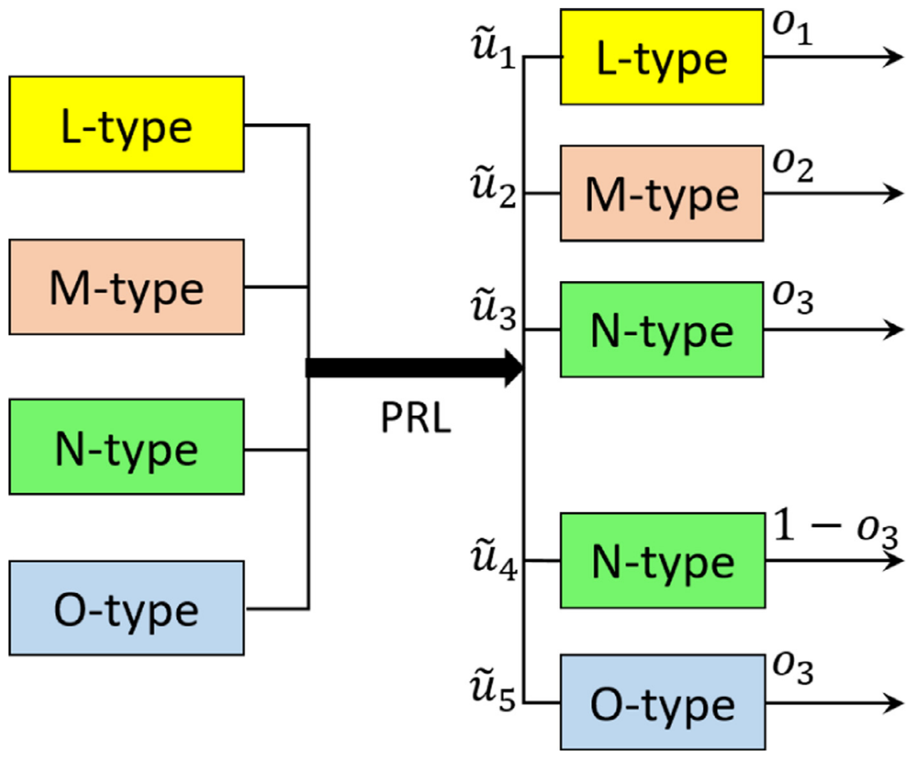


Fig. 5. Schematic of the proposed MILP framework for generation of optimal production plan.

4.2.3. Optimal production plan

Through out this paper, we assume that the daily production plan at the CDUs is known a priori. These production plans are in fact generated carefully keeping in mind several factors, the primary being meeting the monthly targeted production demand. Other key factors affecting the daily production plan include generating fairly contiguous processing schedule, avoiding processing low-sulphur crude, such as the L-type along with a low-sulphur crude type, such as N-type. The program below provides an MILP formulation for generating optimal production plan at the abstraction of Sim0. Fig. 5 shows the schematic of the refinery system with real variables $\tilde{u}_i[d]$ representing the flow values during the d^{th}-day, while $o_l[d]$, $l \in \{1,2,3\}$ is a binary variable indicating the crude-type being processed on the d^{th}-day. For instance, if $o_3[d]=1$ indicates that CDU1 is required to process N-type, while O-type will be processed by CDU2 on the d^{th}-day.

$$
\begin{array}{ll}
\text{Minimize:} & \sum_{d=1}^{D-1} q[d] \\
\text{s.t.,Capacity constraints:} & 0 \le s_i[d] \le S_{i_{\max}}, \qquad \forall i \in \{1,\dots,9\} \\
\text{Mass-balance constraints:} & \begin{cases} s_i[d+1] = s_i[d] + \tilde{r}_i[d] - \tilde{u}_i[d] & \forall i \in \{1,2\} \\ s_3[d+1] = s_3[d] + \tilde{r}_3[d] - (\tilde{u}_3[d] + \tilde{u}_4[d]) & \\ s_4[d+1] = s_4[d] + \tilde{r}_4[d] - \tilde{u}_5[d] & \\ s_{4+i}[d+1] = s_{4+i}[d] + \tilde{u}_i[d] - \tilde{C}_1 \cdot o_i[d] & \forall i \in \{1,2,3\} \\ s_8[d+1] = s_8[d] + \tilde{u}_4[d] - \tilde{C}_2 \cdot (1-o_3[d]) & \\ s_9[d+1] = s_9[d] + \tilde{u}_5[d] - \tilde{C}_2 \cdot o_3[d] & \end{cases} \\
\text{Exclusivity constraints:} & \sum_{l=1}^{3} o_l[d] = 1 \\
\text{PRL constraints:} & \sum_{i=1}^{5} \tilde{u}_i[d] = F[d] \\
\text{Dewatering constraints:} & o_i[d] = 1 \Rightarrow \tilde{u}_i[d-1] = 0, \qquad i \in \{1,2\} \\
\text{Crude-quality constraints:} & q[d] \ge o_1[d] + o_3[d-1] - 1, \quad 1 \ge o_1[d+1] + o_3[d] \\
\text{Inflow/outflow constraints:} & o_i[d] = 1 \Rightarrow \tilde{u}_i[d] = 0 \\
\text{Min. up-time constraints:} & \sum_{\tau=d}^{\min\{d+\text{MUT}-1,D-1\}} o_l[\tau] - (o_l[d] - o_l[d-1]) \ge 0 \\
\text{Min. down-time constraints:} & \sum_{\tau=d}^{\min\{d+\text{MDT}-1,D-1\}} 1 - o_l[\tau] - (o_l[d-1] - o_l[d]) \ge 0 \\
\text{Integer constraints:} & o_l[d], q[d] \in \{0,1\}
\end{array}
\qquad \text{(Sopt-MILP)}
$$

Here, the binary variable q represents the instance when the high-sulphur content crude (N-type) is processed the day before low-sulphur content crude (L-type). The objective is to minimize such instances, since processing L-type after N-type reduces the quality of the low-sulphur content crude; however, the reverse is acceptable, i.e., processing of a low-sulphur content crude can precede the processing of a high-sulphur content crude. Because, L-type and M-type can only be stored in single tanks on the refinery side, dewatering requirements necessitate that if crude is received at a given day, it is not available for processing until the next day is over. Quantities $\tilde{r}_i[d]$'s denote the daily receipt of crudes, while $\tilde{C}_1$ and $\tilde{C}_2$ represent the daily crude processing rates at the CDUs.

4.3. Reinforcement learning

4.3.1. Volume of each crude to be transferred

We use the deep deterministic policy gradient (DDPG) architecture as descrived in Lillicrap et al. (2015) and implement it using the DDPG obtained from OpenAI's Spinning Up repository.[2]. DDPG is a variant of the actor-critic architecture in Fig. 1. The RL algorithm solves for the volume of each crude to be transferred during a 19.8 hr block.

We draw upon the state space of Sim0, presented in Eq. (7), to design the input layer of the neural networks that encode the actor as well the critic. These actor as well as the critic are called at the start of 38 hr block. The input layer consists of the following states:

- Time: day and time (at the start of the 38 hr slot).
- Current refinery zone levels (recall that Sim0 works with abstract zones rather than tanks)
- Production plan; i.e., the volume of each crude grade to be produced on the current day and over each of the next four days. A moment-based representation can be used as an alternative to the day-by-day production plan (Wagle and Paranjape, 2020).
- Maximum volumes of L-type and M-type that can be transferred during the next 19.8 hrs; these are found by determining the number of hours for which the two grades are not produced during the 19.8 hr transfer block.

The reward structure, in this particular problem, consists purely of penalties so that the desired reward is zero. The penalty function is a sum-of-squares, and the individual terms (before being normalized and squared) are calculated as follows.

1. Overflow after 19.8 hrs: for each crude, the overflow $o_{i+} = \mathrm{clip}(s_i[k] + u_i[k:k'] - p_i[k:k'] - s_{i,\max})$, where the clipping function bounds o_{i+} below by zero.
2. Shortfall over 38 hrs: for each crude, the shortfall $o_{i-} = \mathrm{clip}(p_i[k:k+1] - (s_i[k] + u_i[k:k']))$, where the clipping function bounds o_{i-} below by zero.
3. Transferring L-type/M-type beyond permissible limits: the excess amount transferred is given by $l_i = \mathrm{clip}(u_i[k:k'] - u_{i,\max}[k:k'])$, where $i \in \{L-type, M-type\}$ and the clipping function bounds below by zero.

Notice that no crude is transferred to the refinery tanks during the last 18.2 hrs of a 38 hr block. Therefore, the excess needs to be computed at the end of 19.8 hours. Moreover, the extra bounds for L-type and M-type arise because of the fact that there is just one tank for each of these grades in the refinery zone.

The net step reward is given by

$$r[k] = \sum_{i=1}^{5} \left(\left(\frac{o_{i+}}{s_{i,\max}} \right)^2 + \left(\frac{o_{i-}}{s_{i,\max}} \right)^2 \right) + \sum_{i=1}^{2} \left(\frac{l_i}{0.12 + u_{i,\max}} \right)^2 \tag{10}$$

In the above formula, the numerical labels 1 : 5 correspond to L-type, M-type, N-type (CDU1), N-type (CDU2) and O-type, respectively. The additional 0.5 in the denominator of the last term keeps the term finite when $u_{i,\max} = 0$.

The output of the RL algorithm is a vector $\boldsymbol{b} \in [0,1]^5$ satisfying $\sum_{i=1}^{5} b_i = 1$. Notice that the notion of $\boldsymbol{b}$ here is not the same as that in Section 4.1. We get the volume of crude to be transferred during the interval $k:k'$ as

$$\boldsymbol{u}[k:k'] = 20.7\,\boldsymbol{b}, \quad \sum_{i=1}^{5} b_i = 1 \tag{11}$$

4.3.2. Post-processing and refinement

The approach detailed so far only prescribes the volume of each crude to be transferred during a 19.8 hr block. The ordering of the crude supply (i.e., the crude grade to be transferred during individual 1 min slots) has yet to be solved for.

To appreciate the issues that can arise in the absence of post-processing, consider the following pathological case wherein all tanks are full at the start of a 38 hr block. Suppose that only N-type and O-type crude are to be processed by CDU1 and CDU2, respectively, for the foreseeable future. It is reasonable to expect that the RL algorithm would recommend a non-zero top-up only for N-type and O-type. Due to the normalization and multiplication by 20.7 in Eq. (12), it is inevitable that at least one refinery tank would experience an overflow. To see this, note that the PRL can transfer 20.7 units over 19.8 hrs, while the two CDUs put together consume only 12.13 units during the same period.

While formulating the business heuristics, we avoided this problem by permitting fallow slots (i.e., slots where no crude is transferred) explicitly. In Algorithm 1, notice how slots are allocated only against demand or spare capacity.

The objective of a post-processor is to solve the following problem: determine the crude grade to be transferred during any 15 min slot subject to *an upper limit* obtained from Eq. (12). We consider two candidate approaches to solve the ordering problem. The first approach, based on secondary MILP, has been described in Section 4.2.2. The second approach, based on meta-heuristics, is described next.

Meta-heuristics

We start by creating a nominal schedule which consists of five contiguous blocks, one for each crude and with the total volume equal to that obtained from Eq. (12). These five blocks are assumed to be ordered randomly. Thereafter, we run the following steps for each 15 min block. we check if crude from the first block in the list can be transferred; else, we move to the next crude block. If no crude can be transferred, we declare the slot as fallow. This approach is described in Algorithm 2. Note that Sim1 and Sim2 can both be used for post-processing. We note the following difference in how they are employed:

[2] https://spinningup.openai.com/en/latest/

Algorithm 2 Post-processing using Sim1/Sim2.

```
1: Initialize: $k_0 \leq 5$ contiguous crude blocks obtainedfrom Eq. (10) after eliminating crude blockswith near-zero volume
2: Initialize: state $\boldsymbol{s}$
3: Order the blocks randomly: permuted labels = $[r_1, \ldots, r_{k_0}]$; volumes = $[\mathrm{vol}(r_1), \ldots, \mathrm{vol}(r_{k_0})]$
4: Initialize: $\boldsymbol{v}$ =zero array of size $79 \times 2$
5: Initialize: $k = 0$ (number of 15 min blocks assigned)
6: while $k < 79$ and $k_0 > 0$: do
7:    Internal initialization: $j = 1$; flag $= 0$
8:    while flag $= 0$ do
9:       if Crude from $r_j$ blockcan be transferred { depends on whether Sim1 or Sim2 is used} then
10:         $v[k, 1] = r_j$, $v[k, 2] = \min(\mathrm{vol}(r_j), 0.26)$; $\mathrm{vol}(r_j) \leftarrow \mathrm{vol}(r_j) - v[k, 2]$
11:         if $\mathrm{vol}(r_j) = 0$ then
12:            Eliminate the block $r_j$; $k_0 \leftarrow k_0 - 1$
13:         end if
14:         $k \leftarrow k + 1$, flag $= 1$
15:      else
16:         j = j + 1
17:         if $j > k_0$ {no blocks left to check} then
18:            flag $= 0$; $k = k + 1$ {fallow slot}
19:         end if
20:      end if
21:   end while
22:   Map $\boldsymbol{v}[k]$ to $\boldsymbol{u}${exact form depends on Sim1/Sim2}
23:   Update Sim1 state: $\boldsymbol{s} \leftarrow \mathrm{clip}(\boldsymbol{s} + \boldsymbol{u} - \boldsymbol{p})$
24: end while
25: Optional feature: fill fallow slots using the logic of line 14of Algorithm 1.
26: Output: $\boldsymbol{v}$ = ordered array of crude to be transferred during each 15 min slot
```

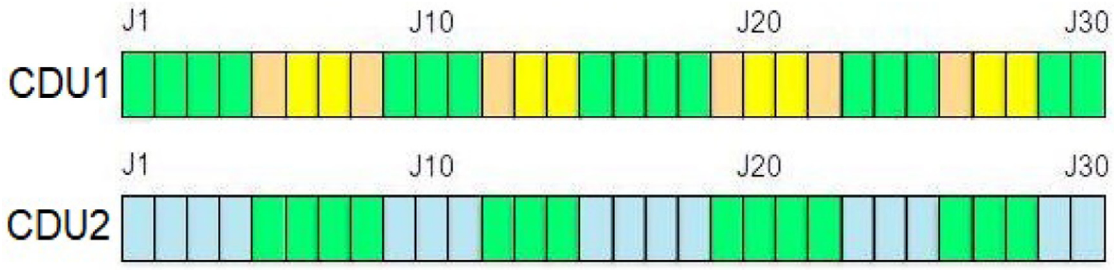


Fig. 6. The monthly production plan considered in this sequel. J10 is the tenth day, and so on.

- Sim1: crude can be transferred as long as there is free space in the refinery zone. For L-type and M-type, an additional constraint that must be port is that CDU1 should not be drawing from the corresponding tanks.
- Sim2: crude can be transferred to a tank as long as it has free space and it isn't feeding the CDUs (except RO23, which can be filled and drained simultaneously). For N-type crude, we try to drain a tank to the extent possible before refilling it. However, if necessary, we use the following logic. If a tank has been drained to less than 75% of its capacity and the other tank for the same grade is full, we start draining the second tank so that the first tank can be refilled.

5. Numerical case studies

For numerical simulation, we use the production plan in Fig. 6. We initialize the tank levels to 60% of their capacity.

5.1. Business rules

The actual production over the first 30 days, predicted using Sim-1, is shown in Fig. 7. Business rules meet 100% of the production requirement in Fig. 6.

5.2. MILP

The MILPs described in Section 4.2 were implemented using the Gurobi optimizer Bixby (2007) on an i7-7700HQ CPU @ 2.8GHz and 16GB RAM. Fig. 8 depicts the optimal PRL transfer schedule generated by the primary MILP. The computational time required is about five minutes and thus illustrates the power of the proposed decomposition method. The solutions generated by the primary MILP are fed to the a series of secondary MILPs for optimal tank assignments. Fig. 9 depicts the crude arrival and withdrawal schedule for the tank PO01. The results for other tanks follow a similar trend. It can be observed that both the tank capacity constraints, as well as the dewatering

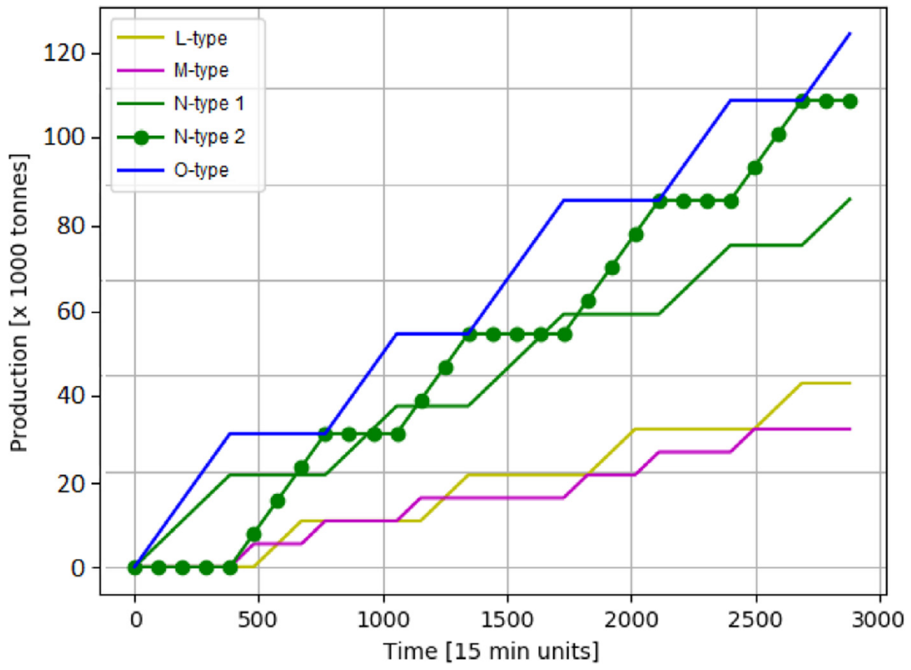


Fig. 7. Production over 15 days for the plan in Fig. 6.

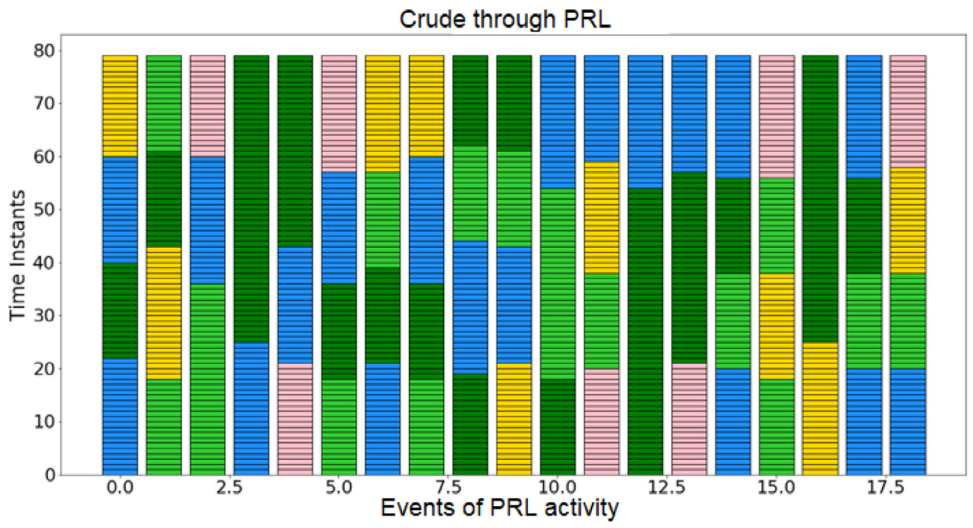


Fig. 8. Optimal crude transfer schedule generated by solving the primary MILP. The schedule is fairly contiguous and meets the targeted demand.

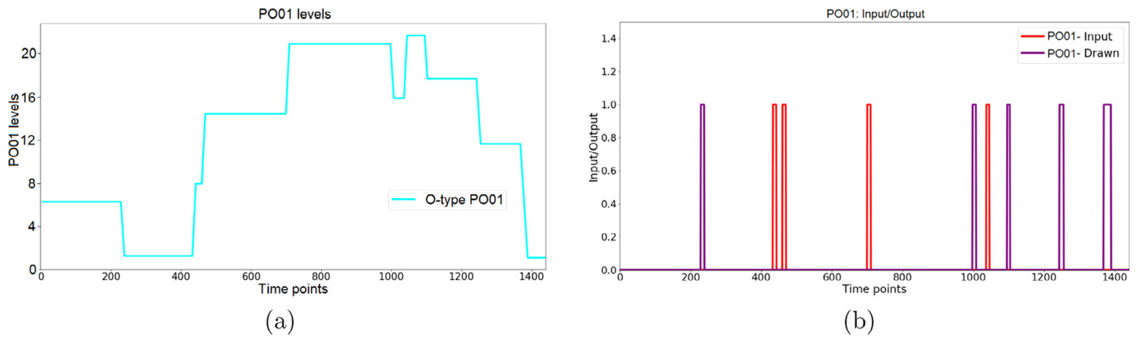


Fig. 9. Illustration of validity of the dewatering constraints for tank PO01. (a) State of the tank at different time points. Each time point depicts a 15-minute window. (b) Red and purple bars indicate the instances of crude receipt and withdrawal, respectively. Observe that a receipt event preceding an event of withdrawal must be separated by at least 96 time points (1 day). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

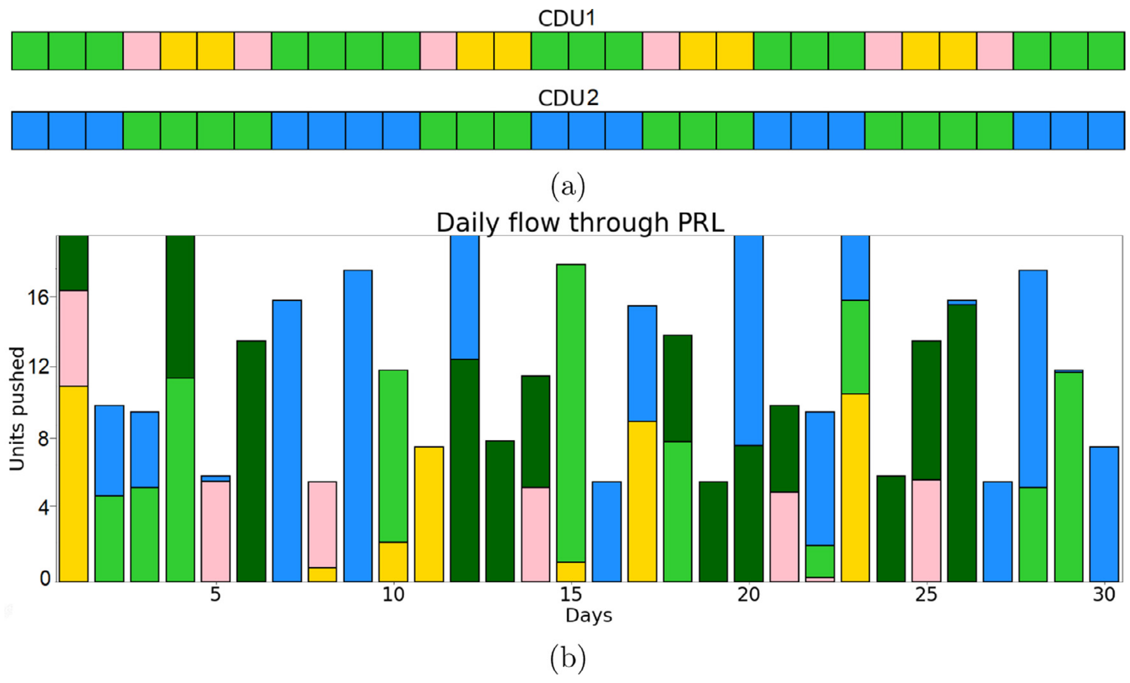


Fig. 10. Optimal daily production plan produced by (Sopt-MILP). (a) Daily production schedules, (b) Daily targeted crude transfer through PRL ($\{\tilde{u}_i\}$). The yellow, pink, green and blue colors represent the crude types L-type, M-type, N-type and O-type, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

constraints are being port. The computational time required for solving the secondary MILPs is only about 30 seconds. Thus, the entire decomposition approach presents itself as an extremely tractable approach for optimizing crude transfer in a refinery system.

We further demonstrate the advantages of the proposed MILP framework for generating optimal production plan using the targeted production demand. Fig. 10a depicts the daily production plan generated by the (Sopt-MILP). It can be seen that in the optimal production plan, there are no instances of N-type being processed the day before the day L-type is scheduled to be processed, as mandated by the crude quality constraints. Fig. 10b shows the daily crude transfer through PRL, that can further be used to refine the search for the primary MILP. The computation time required for generating the optimnal production plan is only 0.5 seconds.

5.3. Reinforcement learning

The two algorithms used in simulation and optimization using reinforcement learning are the Deep Deterministic Policy Gradient (DDPG) Silver et al. (2014); Lillicrap et al. (2015) and the Soft Actor Critic (SAC) as described in Haarnoja et al. (2018). The DDPG was modified to include explicit exploration by the addition of random noise to the DDPG policy outputs which reduces exponentially as the training session progresses.

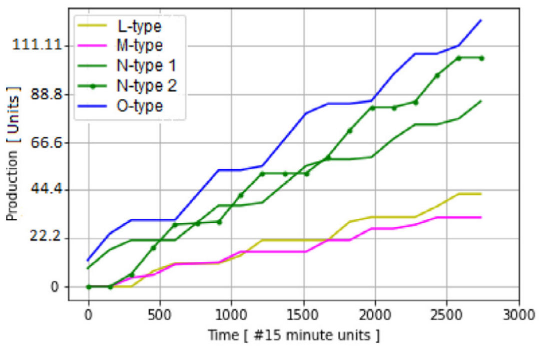
We consider three cases for numerical simulation:

1. Case 1: DDPG on Sim0, with Sim0 additionally used for computing the rewards. The schedule is post-processed using the Sim2 meta-heuristics in Algo 2, with the optional feature that fallow slots are replaced with a feasible crude grade. The resulting production is shown in Fig. 11 a. Note that production takes place per the plan in Fig. 6. When the upstream refinery tank is empty, production is stopped, but it does not move to a different crude grade until the appointed time.
2. Case 2: DDPG on Sim0 with Sim2 used for computing the rewards. This case is split into two, as described below. The rewards are plotted in Fig. 11 b and Fig. 11 d.
3. Case 3: SAC on Sim0 with Sim2 used for computing the rewards. The rewards are plotted in Fig. 11 c.

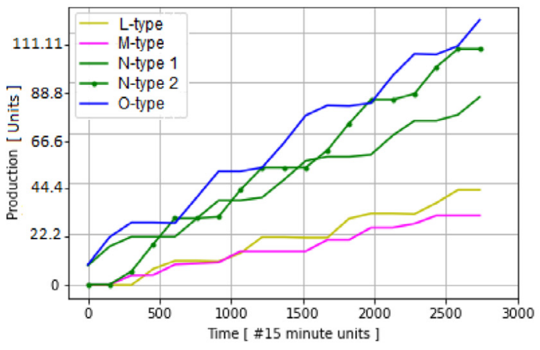
Case 2 is split further into two, to study a variant of the algorithm wherein we shuffle the production schedule. We view the production plan as a concatenation of “primitive” plans described presently. We shuffle the order in which the concatenation is done. The three primitive production considered here are defined for CDU1; the production plans for CDU2 are complementary: CDU2 processes N-type when CDU1 does not; else it processes O-type. The three primitives for CDU1 are: (N-type, N-type, N-type), (N-type, N-type, N-type, N-type) and (M-type, L-type, L-type, M-type).

The variation in the rewards during training for each of these cases is shown in Fig. 12. We note here that the reward shown in Fig. 12(a) is obtained before the post-processing based on Algorithm 2. Interestingly, this performs better than when Sim2 is used to compute the rewards which, one would expect, should weed out infeasible solutions that do not register on Sim0.

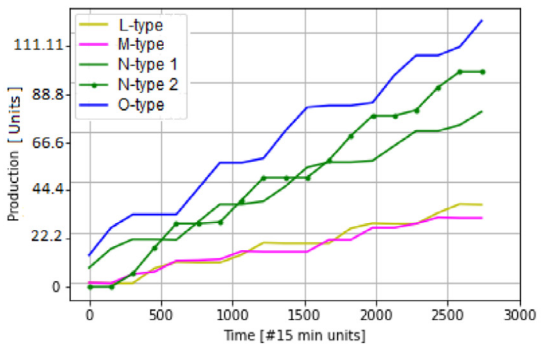
From Fig. 11, we observe that the base DDPG algorithm with exploratory noise performs better than its variations. Notice, for instance, the conspicuous shortfall in production of all but O-type when the policy is trained using SAC. However, with a shuffled schedule, the policy is more robust against changes in the production plan. The time required for training is similar (within 10%) for all four cases. The performance of each algorithm, in terms of the deficit, is listed in Table 2. It can be enhanced significantly using an MILP-based post-processor, such as the one in Section 4.2.2.



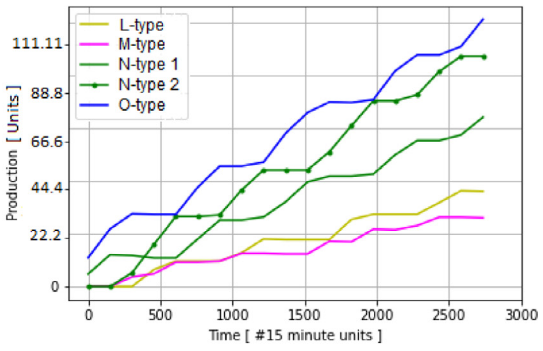
(a) DDPG on Sim 0



(b) DDPG on Sim 2

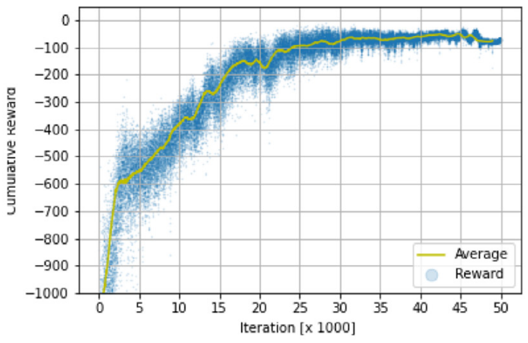


(c) SAC Sim 2

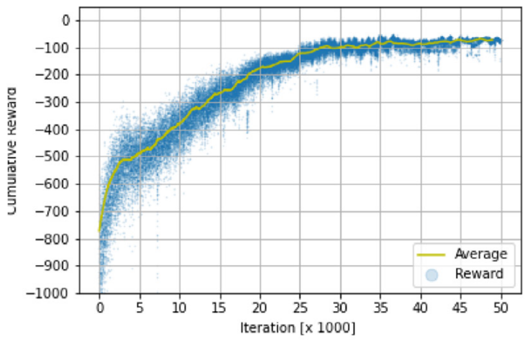


(d) DDPG on Sim 2 with shuffled schedules

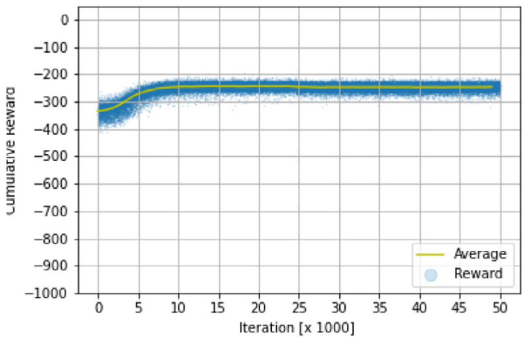
Fig. 11. Production time history over 30 days when the refinery is refilled using RL-based schedule.



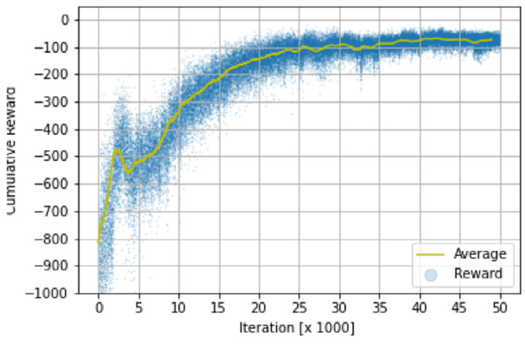
(a) DDPG on Sim 0 (before post-processing)



(b) DDPG on Sim 2



(c) SAC Sim 2



(d) DDPG on Sim 2 with shuffled schedules

Fig. 12. The variation in the reward during training for each of the four cases listed above.

Table 2
Comparison of RL algorithms .

Algorithm	L-type	M-type	N-type 3	N-type 4	O-type
DDPG Sim0	0%	−0.45%	0%	−1.52%	0%
DDPG Sim2	−2.45%	−5.33%	−2.96%	−2.98%	−3.88%
SAC Sim2	−18.97%	−10.61%	−11.31%	−12.94%	−5.4%
DDPG Sim2 Shuffled	−6.12%	−9.55%	−16.47%	−9.53%	−8.1%

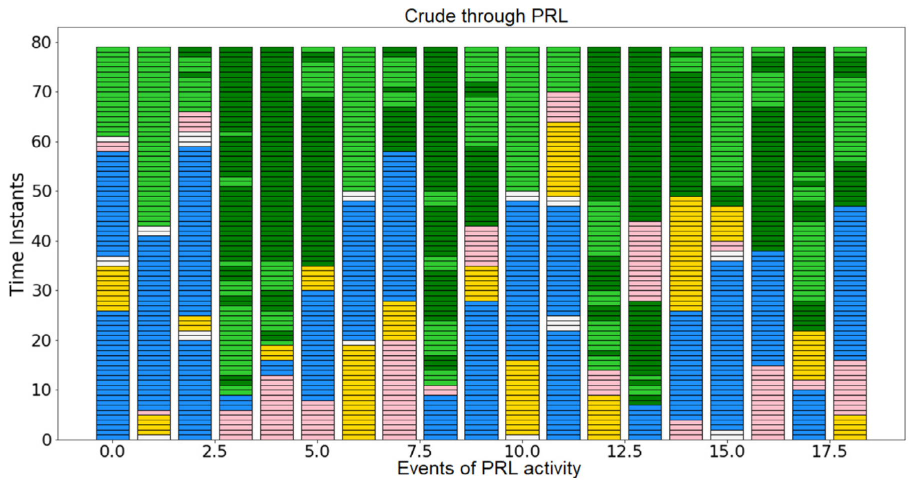


Fig. 13. Optimal crude transfer schedule generated by the RL algorithm and post-processed using the MILP. The white colored bars indicate fallow slots.

We finally demonstrate how the optimal policy generated by the RL agorithm can be leveraged to assist the primary MILP for generating the optimal schedule for crude transfer through the PRL. The RL algorithm outputs the amount of crude for each crude-type to be transferred during each PRL session. The primary MILP can be modified to search for schedules, that match the solutions generated by the RL algorithm. The performance of the hybrid approach is demonstrated in Fig. 13.

6. Conclusion

In this paper, we presented three optimum scheduling techniques, based on a multimodel framework, for transferring crude to the Antwerp refinery from the upstream port. We derived four nested models, with each nesting a consequence of spatio-temporal abstractions. Each of these models presents features that aids a specific optimizer. For instance, the optimal scheduler based on business rules was derived using Sim1, while the state space of Sim0 was found to be ideally suited for designing the RL algorithm. Sim2 was used for post-processing the schedules generated using Sim0 and Sim1. Finally, Sim3 was found to be amenable to optimization using MILP. We also developed an MILP-based post-processor for the schedules generated by the business rules and RL.

The problem addressed in the paper could be linearized, which allowed us to derive an ideal solution using MILP. Business rules could reproduce this solution (up to a few units of crude being exchanged between successive days), but could not address the dewatering constraint adequately. RL came to within 97% of the baseline MILP solution and could match it with post-processing using a secondary MILP solver.

It is worth noting, in defence of RL, that the problem formulation was perhaps not appropriate for RL given that the set of feasible solutions was singular for all practical purposes because of the production plan being prescribed a priori. However, we expect the RL architecture developed here, and in our prior paper (Wagle and Paranjape, 2020), to solve two follow-on problems which were not addressed in the paper: crude-blending and production planning. The former arises when one considers material properties of individual crude such as density, metal content and viscosity. The production plan can be introduced as a solvable variable by demanding that is satisfy a weaker set of constraints and maximize some prescribed performance metrics. While RL is well-suited to solving this advanced problems, solutions found using meta-heuristics or MILP (based on a linearized problem) can help pre-train the RL policy.

Author Contributions

A.P., M.B., S.M. and A-L.B. conceived the study. A. P., S.W. and R.L. implemented the Reinforcement Learning models. M.B. implemented the Mixed-Integer Linear Programming models. A.P., M.B., S.W. and R.L. wrote and revised the paper. S.M. and A-L.B. carried out the feasibility analysis.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Al-Othman, W.B.E., Lababidi, H., Alatiqi, I.M., Al-Shayji, K., 2008. Supply chain optimization of petroleum organization under uncertainty in market demands and prices. Eur J Oper Res 189 (3), 822–840.

Baker, T.E., Lasdon, L.S., 1985. Successive linear programming at exxon. Manage Sci 31 (3), 264–274.

Bertsekas, D.P., Homer, M.L., Logan, D.A., Patek, S.D., Sandell, N.R., 2000. Missile defense and interceptor allocation by neuro-dynamic programming. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 30 (1), 42–51.

Bixby, B., 2007. The gurobi optimizer. Transp. Re-search Part B 41 (2), 159–178.

Brandeau, M.L., Zaric, G.S., Richter, A., 2003. Resource allocation for control of infectious diseases in multiple independent populations: beyond cost-effectiveness analysis. J Health Econ 22 (4), 575–598.

Codato, G., Fischetti, M., 2006. Combinatorial benders' cuts for mixed-integer linear programming. Oper Res 54 (4), 756–766.

Dalal, G., Mannor, S., 2015. Reinforcement learning for the unit commitment problem. In: 2015 IEEE eindhoven powertech, pp. 1–6.

Ernst, D., Glavic, M., Capitanescu, F., Wehenkel, L., 2009. Reinforcement learning versus model predictive control: a comparison on a power system problem. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 39 (2), 517–529.

Gao, X., Jiang, Y., Chen, T., Huang, D., 2015. Optimizing scheduling of refinery operations based on piecewise linear models. Computers &amp; Chemical Engineering 75, 105–119.

Gill, G., 1995. Linear programming as a tool for refinery planning. In: Proceedings of the 31st Annual Conference of the Operational Research Society of New Zealand, Wellington, New Zealand, August, vol. 3, pp. 103–111.

Göthe-Lundgren, M., Lundgren, J.T., Persson, J.A., 2002. An optimization model for refinery production scheduling. Int. J. Prod. Econ. 78 (3), 255–270.

Haarnoja, T., Zhou, A., Abbeel, P., Levine, S., 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Dy, J., Krause, A. (Eds.), Proceedings of the 35th International Conference on Machine Learning. In: Proceedings of Machine Learning Research, vol. 80. PMLR, pp. 1861–1870.

Hubbs, C.D., Li, C., Sahinidis, N.V., Grossmann, I.E., Wassick, J.M., 2020. A deep reinforcement learning approach for chemical production scheduling. Computers & Chemical Engineering 141, 106982.

Jia, Z., Ierapetritou, M., 2003. Mixed-integer linear programming model for gasoline blending and distribution scheduling. Industrial &amp; Engineering Chemistry Research 42 (4), 825–835.

Kannan, R., Monma, C.L., 1978. On the computational complexity of integer programming problems. In: Optimization and Operations Research. Springer, pp. 161–172.

Kazarlis, S.A., Bakirtzis, A.G., Petridis, V., 1996. A genetic algorithm solution to the unit commitment problem. IEEE Trans. Power Syst. 11 (1), 83–92.

Khor, C.S., Varvarezos, D., 2017. Petroleum refinery optimization. Optimization and engineering 18 (4), 943–989.

Klotz, E., Newman, A.M., 2013. Practical guidelines for solving difficult mixed integer linear programs. Surveys in Operations Research and Management Science 18 (1–2), 18–32.

Koenig, W.J., 1963. The application of computers for refinery simulation (refinery planning by linear programming). 6th World Petroleum Congress. OnePetro.

Lee, H., Pinto, J.M., Grossmann, I.E., Park, S., 1996. Mixed-integer linear programming model for refinery short-term scheduling of crude oil unloading with inventory management. Industrial &amp; Engineering Chemistry Research 35 (5), 1630–1641.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

Mendez, C.A., Grossmann, I.E., Harjunkoski, I., Kaboré, P., 2006. A simultaneous optimization approach for off-line blending and scheduling of oil-refinery operations. Computers & Chemical Engineering 30 (4), 614–634.

Misra, S., Kapadi, M., Gudi, R.D., 2020. A multi grid discrete time based framework for maritime distribution logistics &amp; inventory planning for refinery products. Computers &amp; Industrial Engineering 146, 106568.

Negenborn, R.R., De Schutter, B., Wiering, M.A., Hellendoorn, H., 2005. Learning-based model predictive control for Markov decision processes. IFAC Proceedings Volumes 38 (1), 354–359.

Papadimitriou, C.H., 1981. On the complexity of integer programming. Journal of the ACM (JACM) 28 (4), 765–768.

Phillips, A.E., Waterer, H., Ehrgott, M., Ryan, D.M., 2015. Integer programming methods for large-scale practical classroom assignment problems. Computers &amp; Operations Research 53, 42–53.

Pinedo, M., 2012. Scheduling - Theory, Algorithm, and Systems, 5th Springer, Cham, Switzerland.

Pinto, J.M., Joly, M., Moro, L.F.L., 2000. Planning and scheduling models for refinery operation. Comput. Chem. Eng. 24 (9–10), 2259–2276.

Ross, G.T., Soland, R.M., 1975. A branch and bound algorithm for the generalized assignment problem. Math Program 8 (1), 91–103.

Saharidis, G.K.D., Minoux, M., Dallery, Y., 2009. Scheduling of loading and unloading of crude oil in a refinery using event-based discrete time formulation. Computers & Chemical Engineering 33 (8), 1413–1426.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M., 2014. Deterministic policy gradient algorithms. 31st International Conference on Machine Learning, ICML 2014 1.

Taherkhani, M., Seifbarghy, M., Tavakkoli-Moghaddam, R., Fattahi, P., 2020. Mixed-integer linear programming model for tree-like pipeline scheduling problem with intermediate due dates on demands. Operational Research 20 (1), 399–425.

Trick, M., 2005. Formulations and reformulations in integer programming. In: International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming. Springer, pp. 366–379.

Uribe-Rodriguez, A., Castro, P.M., Gonzalo, G.-G., Chachuat, B., 2020. Global optimization of large-scale MIQCQPs via cluster decomposition: application to short-term planning of an integrated refinery-petrochemical complex. Computers &amp; Chemical Engineering 140, 106883.

Wagle, S., Paranjape, A.A., 2020. Use of simulation-aided reinforcement learning for optimal scheduling of operations in industrial plants. In: 2020 Winter Simulation Conference (WSC), pp. 572–583. doi:10.1109/WSC48552.2020.9383893.

Williams, G., Wagener, N., Goldfain, B., Drews, P., Rehg, J.M., Boots, B., Theodorou, E.A., 2017. Information theoretic MPC for model-based reinforcement learning. In: 2017 IEEE international conference on robotics and automation (ICRA), may 29th - june 3rd, singapore, pp. 1714–1721.

Wu, N., Li, Z., Qu, T., 2017. Energy efficiency optimization in scheduling crude oil operations of refinery based on linear programming. J Clean Prod 166, 49–57.

Zhang, T., Kahn, G., Levine, S., Abbeel, P., 2016. Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search. In: 2016 IEEE international conference on robotics and automation (ICRA), may 16th - 21st, stockholm, sweden, pp. 528–535.

Zhang, W., Dietterich, T.G., 1995. A reinforcement learning approach to job-shop scheduling. In: Proceedings of the 14th international joint conference on artificial intelligence - volume 2. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 1114–1120.

Zhang, W., Dietterich, T.G., 1996. High-performance job-shop scheduling with a time-delay TD λ network. In: Touretzsky, D.S., Mozer, M.C., Hasselmo, M.E. (Eds.), Advances in neural information processing systems 8. MIT Press, Cambridge, MA, pp. 1024–1030.

Zhen, G., Lixin, T., Hui, J., Nannan, X.U., 2008. An optimization model for the production planning of overall refinery. Chin. J. Chem. Eng. 16 (1), 67–70.