# A Hierarchical Framework for Optimal and Scalable Process Scheduling in Plant Operations

Ajit Umesh Deshpande Mayank Baranwal

***Abstract*— Process scheduling problems are often modeled as mixed-integer nonlinear programs (MINLPs) with a large number of constraints. While meta-heuristics, such as the simulated annealing (SA) algorithm or the genetic algorithm (GA) have been extensively employed to obtain high-quality solutions to MINLPs, their capabilities are limited by the large number of combinatorial constraints and the time required to obtain these solutions. In view of these limitations, this paper presents a hierarchical approach that leverages the capabilities of the satisfiability modulo theory (SMT) for constraint satisfaction and the relative CPU-time competitiveness of the SA algorithm in configuring meta-heuristics for optimal process scheduling subjected to static and temporal constraints. The framework has access to a high-fidelity simulator of the plant, but not the mathematical model. Besides addressing the "hard" operational constraints, our framework also accommodates for "soft" constraints, such as generating schedules that are contiguous and avoid frequent switching between two operational modes.**

## I. Introduction

Optimal scheduling in process industries is of vital importance given the competing nature of different activities over time and extremely tight profit margins [1], [2], [3]. Interplay between physical (operational) and economic constraints further aggravate the complexity of such optimization problems. The problem entails determining the most efficient way to produce a set of products from the raw materials given the input receiving rates, physical constraints involving flow of materials in and out of a receiving/processing unit, costs incurred for material flow and production, and quality-dependent profit generation. Of course, it would be ideal to produce only the highest quality products, however, processing constraints require that only a fraction of total demand may be of the highest quality, while the low-quality material is produced simultaneously in order to avoid wastage.

Recent advances in mathematical modeling and rapidly growing computational power has witnessed Mixed Integer Non-Linear Programming (MINLP) as one of the most widely explored methods for process scheduling problems [4], [5]. Despite their flexibility, MINLP models are computationally difficult to be solved exactly. This is partially addressed by resorting to linear models and consequently using state-of-the-art Mixed Integer Linear Programming (MILP) solvers, or using meta-heuristics, such as the simulated annealing (SA) algorithm [6], [7] and the genetic algorithm (GA) [8], [9]. The vanilla SA algorithm is incapable of handling a large number of constraints, since the constraints need to be added as a penalty to the true objective function. On the other hand, the GA algorithm is inherently computationally extensive. Recently, a Reinforcement Learning (RL) framework [10] was developed to determine optimal ship out policies for process scheduling. While the inference with RL framework is extremely fast, the RL agent needs to be trained every time the parameter of the models change, thereby limiting the capability of the machine-learning models.

In this paper, we aim to leverage the competitive advantage of SA algorithm over other methods in producing solutions at a much faster rate especially in situations involving larger time horizon. While the temporal constraints still need to be addressed by augmenting them to the true objective function, the time-invariant constraints are handled using the satisfiability modulo theory (SMT) solver. The SMT solvers are extremely efficient at producing feasible solutions and thus significantly narrow down the search space for the SA algorithm in our proposed hierarchical framework. We keep our focus restricted to workflow paths which are acyclic in nature. We further consider exact nonlinear constraints and costs incurred while determining a feasible solution. The main task of the proposed hierarchical SA algorithm is to suggest a work schedule for operating the manufacturing units without violating any constraints while maximizing the productivity of the entire manufacturing plant. We also present a delivery schedule involving a nonlinear delivery cost for the finished product using a greedy approach and verify its correctness to ensure maximum profit. Additionally, we take into consideration the idea of contiguous scheduling of the machines since frequent switching is considered detrimental to the machine performance over time.

## II. Prior Work

Process scheduling problems and their variants have been extensively studied using constraint reformulation in MILPs/MINLPs, such as the Reformulation Linearization Technique (RLT) [11], or through meta-heuristics, such as the SA algorithm [6], [7], the GA [8], [9] and more recently the RL algorithm [10]. Our work is primarily based on the plant model described in [10], which essentially concerns with process scheduling in an oil refinery [12]. The motive of the work in [12] is to maximize the utilization of the refinery units subjected to different types of constraints, including but not limited to resource constraints, constraints pertaining to

A. Deshpande is with the Department of Electrical & Computer Engineering Engineering at the University ofCalifornia, San Diego, CA 92093: deshpande.ajit@alumni.iitgn.ac.in.

M. Baranwal is with the Division of Data & Decision Sciences, Tata Consultancy Services Research, Mumbai, 400607 India e-mail: baranwal.mayank@tcs.com.

the time allowed between overhauling of a particular unit, capacity constraints and contiguity constraints.

In this paper, we propose a hierarchical framework involving the SMT solver, the SA algorithm and a greedy ship out heuristic to produce contiguous and feasible schedules for maximizing the utilization (and the revenue thereof) of the refinery units for the plant model described in [10]. The SA algorithm [13] was initially formulated to tackle classical optimization problems such as the travelling salesman problem. Various implementations of the SA algorithm have been described in [14]. Extensive work on constrained optimization using penalty in the SA algorithm has later been described in [15]. The approach of combining the SMT solver with the SA algorithm for Web Service Composition planning has been proposed in [16]. Due to the competitive edge of the SA algorithm in terms of its computational overhead, the SA algorithm was extensively employed for scheduling problems and its variants. The job shop scheduling problem, which aims to minimize the time needed to complete a task involving a set of jobs to be completed serially in the least amount of time has been tackled using the SA algorithm by [17]. The problem considered in this paper, on the other hand, consists of an acyclic graph as the plant model but the graph does have fairly complex connections amongst nodes especially considering that the materials used have different degree in terms of their quality. Moreover, it is required to maximize the reward of production over a fixed time horizon.

## III. Problem Formulation and Plant Model for Simulation

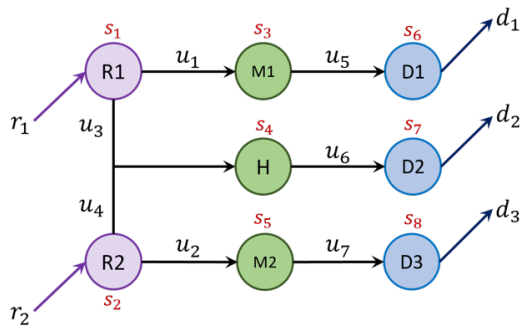



Fig. 1. Flow diagram of the plant.

The manufacturing plant model considered for the simulation is shown in Figure 1. We aim to optimize the process schedule for this simulation to maximize the total profit. This model was also used in [10] for analysis of the simulation results. The flow diagram consists of two receiving stations $R_1$ and $R_2$. Raw materials of two different qualities are received at these stations. The simulator consists of three machines used for processing of these raw materials. $M_1$ processes material from $R_1$ and $M_2$ from $R_2$. The machine $H$ is a hybrid unit and can process a mixture of materials from both $R_1$ and $R_2$. Raw materials processed by these machines are stored at dispatch units $D_1$, $D_2$ and $D_3$. The finished products are shipped from the dispatch units according to the shipping schedule with an objective to maximize the total profit.

The flow quantities in the simulation model are updated using dynamic mass-balance equations with each node (unit) being described by a state corresponding to the amount of material contained in that unit. We use $\{s_i[t]\}$ to denote the amount of material present at various nodes at any instant $t$, whereas $r_1[t]$ and $r_2[t]$ depict the amount of material received at the nodes $R_1$ and $R_2$, respectively, at the $t^{\text{th}}$ instant. Finally, the ship out quantities at the $t^{\text{th}}$ instant are denoted by $d_1[t], d_2[t]$ and $d_3[t]$ corresponding to the units $D_1, D_2$ and $D_3$, respectively. For the receiving node $R_1$, the update equation at time instant $t$ for the states in the simulation model is given by:

$$s_1[t] = s_1[t-1] + r_1[t-1] - u_1[t-1] - u_3[t-1] \quad (1)$$

The state-update equation for the receiving unit $R_2$ can be similarly obtained. For machine $M_1$, and shipping node $D_1$ the update equations are described in (2) and (3) respectively.

$$s_3[t] = s_3[t-1] + u_1[t-1] - u_5[t-1] \quad (2)$$
$$s_6[t] = s_6[t-1] + u_5[t-1] - d_1[t-1] \quad (3)$$

The state update equations for the remaining nodes can also be derived using the principle of mass balance. The remaining nodes in the simulation model will have the following state update equations

$$s_2[t] = s_2[t-1] + r_2[t-1] - u_2[t-1] - u_4[t-1] \quad (4)$$
$$s_4[t] = s_4[t-1] + u_3[t-1] + u_4[t-1] - u_6[t-1] \quad (5)$$
$$s_7[t] = s_7[t-1] + u_6[t-1] - d_2[t-1] \quad (6)$$
$$s_5[t] = s_5[t-1] + u_2[t-1] - u_7[t-1] \quad (7)$$
$$s_8[t] = s_8[t-1] + u_7[t-1] - d_3[t-1] \quad (8)$$

### A. Constraints and Production Costs

Each edge in the simulation model that connects two different units represents transfer of materials through these channels. Correspondingly, the edge weights $\{u_i[t]\}$ represent whether the channel is active (1) or closed (0) at the $t^{\text{th}}$ instant. The model is subjected to following operational constraints:

- There cannot be a simultaneous inflow and outflow from the machines, i.e., $u_1[t] \cdot u_5[t] = u_2[t] \cdot u_7[t] = \max(u_3[t], u_4[t]) \cdot u_6[t] = 0$ at each time instant $t$.
- Each area or node can hold at most $S_{\max}$ number of units at any instant. Any overflow from the edges or nodes is considered as a loss to the units.
- Ship out quantities from each shipping node at any instant is an integer belonging to the set $\{0, 1, \ldots, d_{\max}\}$ where $d_{\max}$ is the maximum holding constraint at an instant in time. The integrality constraint is synonymous with the fact that partially finished products are not acceptable.
- To establish contiguity in the operation of the machines, we force the edge values to be a certain value (0 or 1) for a randomly generated duration of time before it switches again. This helps in reducing the frequent switching of

machines which eventually contributes to the longevity of the plant.

The costs incurred in the production are listed below

- Raw material cost of both high and low quality material
- Cost of running the machine and transferring the material is $a \cdot u_i[t]$ for instant $t$ and edge $i$.
- Shipping cost is non-linearly related to the number of units shipped together. The nonlinear dependence ensures that it is profitable for a plant to ship quantities out in large batches. Let $d[t]$ be the number of units shipped together, the cost of shipping assumes the form

$$\beta_0 + \beta_1 \cdot \left(1 - e^{-\beta_2 d[t]}\right), \tag{9}$$

where $\beta_0, \beta_1$ and $\beta_2$ are appropriate cost coefficients.

The revenue earned per unit is $g_1$ for pure high quality product, $g_3$ for pure low quality product and $g_2$ for any mixture.

## IV. Preliminaries

### A. Annealing Algorithm

Our approach is based on the SA algorithm, which was essentially developed for modeling a thermodynamic process. In an annealing algorithm, the temperature of a system is gradually decreased from an initial temperature $T_0$. In a surrounding of decreasing temperatures starting from $T_0$, the system reaches equilibrium at an energy $E$ at temperature $T$ with probability given by the Boltzmann distribution

$$P(E = k) = \frac{1}{Z(T)} \exp\left(-\frac{k}{k_b T}\right), \tag{10}$$

where *Z(T)* is a normalization function and $k_b$ is the Boltzmann constant. The algorithm proposed in [18] (also known as the Metropolis algorithm) used the probabilistic distribution to simulate attainment of equilibrium of atoms at a temperature. In this algorithm, given the current energy state $E_0$, a random energy state $E$ is chosen as: If $E < E_0$, the system moves to the new state. If, however, $E \geq E_0$, the new state is still chosen with a probability proportional to

$$\exp\left\{-\frac{E - E_0}{k_b T}\right\}. \tag{11}$$

By repeating this procedure over multiple iterations, the system reaches equilibrium. Later, [13] described a process to further reduce the temperature each time the equilibrium is attained at a particular temperature, and continue the Metropolis algorithm at successively reduced temperatures until the system reaches the lowest temperature. The algorithm is referred to as the simulated annealing algorithm. The rate at which the initial temperature is successively reduced to lower temperatures is known as the annealing schedule.

### B. General Implementation of Simulated Annealing

The simulated annealing algorithm can be implemented for optimization of a large class of functions (possibly discontinuous) using the following steps:

1) Initialize the temperature to $T_0$ and the system to a random state in the search space where the function value is $E_0$.
2) Select another point in the search space and evaluate the value of the function. Let this value be $E$.
3) Compare the two function values and let $E$ - $E_0$ = $\Delta E$.
4) Choose a random real number $P$ in the range (0,1). If $P \leq exp\left(-\frac{\Delta E}{T}\right)$ (where $T$ is the current temperature), move the system to the new state $E$.
5) Repeat steps 2, 3 and 4 until the system reaches a state where the function value satisfies the constraints or the termination criterion has reached.

### C. Satisfiability Tests

The Boolean satisfiability problem (SAT) is used in our work to map feasible regions of the process scheduling problem to form a neighbourhood that is utilized by the SA algorithm. Satisfiability tests are designed to accept Boolean constraints in the conjunctive normal form and return all the acceptable values of the variables as a solution set. This solution set is generated using an exhaustive search method or more efficient algorithms such as the Schöning's random walk. In our work, we use these satisfiability tests to evaluate the search space for the SA algorithm.

## V. Proposed Hybrid Approach

### A. Constraint Modeling using SMT

Let $M$ be the length of the planning horizon. Our objective is to obtain an operation schedule and a shipping schedule such that they do not violate any constraints and generate as large a profit as possible. The operation schedule can be fully explained using the set of arrays $u_i$'s for each edge in the model, whereas the shipping schedule is represented using an array $d_i$ for $i \in \{1,2,3\}$ corresponding to each shipping node. The primary constraints in the plant model are simultaneous inflow/outflow constraints and the maximum edge occupancy constraints. For instance, the constraint that $u_1$ and $u_5$ cannot both be active (1) simultaneously is a simultaneous inflow/outflow constraint. This constraint is formulated as:

$$u_1[i] \cdot u_5[i] = 0, \qquad \forall i \in 1,2,\ldots,M. \tag{12}$$

The constraint that the total material transported through the edges $u_1$ and $u_2$ should not exceed the material received at $r_1$ is a maximum edge occupancy constraint. The maximum edge occupancy constraint is formulated as:

$$\sum_{t=1}^{M} (r_1[t] - (u_1[t] + u_3[t])) \geq 0. \tag{13}$$

The maximum edge occupancy constraints corresponding to other nodes can be modeled in a similar manner. Simultaneous inflow/outflow constraints remain consistent for any input sequence and threshold limits, since they are only dependent on the plant design. Hence, we can reduce the search space of the SA algorithm by evaluating a set of possible $u_i$'s apriori which do not violate the simultaneous inflow/outflow constraints.

We determine these possible $u_i$'s using the SMT solver [19]. These possible $u_i$'s would be the search space out of which we will be choosing different points as neighbours during the implementation of the SA algorithm. Subsequently, we describe an objective function that the SA algorithm will minimize. This function is defined as

$$\text{Objective} = -\text{Reward} + \text{EdgeCost} + \text{ShippingCost} + \text{Penalty} \tag{14}$$

This is expressed mathematically as follows

$$P = \sum_{t=1}^{M} \Big[ -(g3 \cdot d_1[t] + g2 \cdot d_2[t] + g1 \cdot d_3[t]) + \sum_{k=1}^{7} u_k[t] + \sum_{k=1}^{3} \left(\beta_0 + \beta_1 (1 - e^{-\beta_2 d_k[t]}\right) \Big] + y \cdot \text{count}$$

where $y$ is the penalty for each constraint violation and "count" is the number of constraint violations using a particular set of $u_i$ for that iteration.

We use the penalty as a hyperparameter for tuning and proceed to find the value of the penalty that gives a minimum value of the objective function. We also keep a track of the number of constraint violations during each iteration to confirm that a feasible solution is attainable for some value of $y$. For evaluation of the objective in equation (6), the shipping policy is described in the next section.

### B. Heuristic for Optimal Shipout

The sequences $u_5$, $u_6$ and $u_7$ represent the number of processed units reaching the nodes $D_1$, $D_2$ and $D_3$, respectively. Each shipping unit is subjected to a maximum holding constraint of $d_{\max}$ units, while it is profitable to ship products out in large batches due to the exponential nature of the reward function. Hence, we resort to a greedy approach for determining the optimal ship out policy. The greedy approach also ensures that the SA algorithm is concerned with optimizing just the $u_i$'s while the sequences $d_1$, $d_2$ and $d_3$ get fixed accordingly. The greedy ship out policy for the plant model can be described as stated below:

- For each shipping node, keep a count of the number of units produced after each time instant.
- As soon as the total units reaches $d_{\max}$ for any node at time instant $k$, set $d_i[k] = d_{\max}$ for that particular shipping node $i$. The default value for $d_i[j]$ is 0 for all time instants $j$ unless it is specifically set using the policy.
- At the end of the planning horizon, if the number of units stored at a shipping unit is nonzero, the units are shipped all at once at the end of the horizon.

In order to verify that our greedy policy for ship out is indeed optimal and yields the minimum shipping cost for a given sequence of $u_i$'s, we compare it with the the Genetic Algorithm in Section VI. It is observed that the batch-sizes of shipped out quantities are the same using both the greedy approach, as well as the GA algorithm, though the time instants at which these batches are shipped may vary. Moreover, since the shipping cost in (9) is time-invariant, it can be concluded that the greedy policy of shipping is indeed optimal.

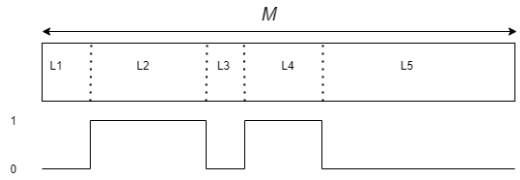



Fig. 2. A sample sequence for $u_i$ with contiguous scheduling.

### C. Designing Contiguous Schedule

For the purpose of contiguous scheduling of the machines and to reduce their switching frequency, we restrict the number of switching instants to at most four for each $u_i$. Consequently, each $u_i$ spanning over a time-horizon of length $M$ is fragmented into contiguous segments of lengths $L_1$, $L_2$, $L_3$, $L_4$ and $L_5$, such that (a) there would be no switching of operations during a contiguous interval spanned by $\{L_k\}$, and (b) the continuous segments sum up to the length of the planning horizon, i.e.,

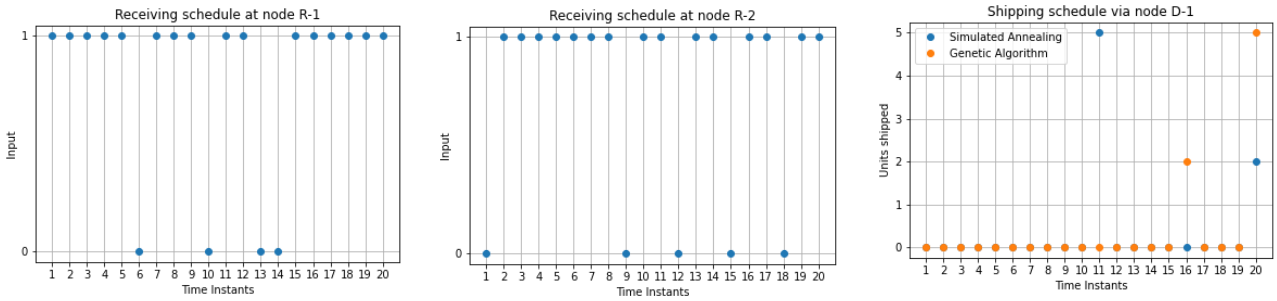
$$L_1 + L_2 + L_3 + L_4 + L_5 = M. \tag{15}$$

We use a random generator to generate $\{L_k\}$ with sum of elements equal to $M$. Next, we choose a feasible point in the search space suggested by the SMT solver and repeat it over a contiguous segment of length $L_1$. This process is repeated until the sequence of $u_i$'s over all $L_k$'s, $1 \leq k \leq 5$ gets fixed. The process of contiguous scheduling is illustrated in Figure 2. The above fragmentation scheme results in a feasible solution with a fairly contiguous nature which is then iteratively optimized using the SA algorithm. The overall algorithm is described in Algorithm 1.

## VI. Results

We now benchmark the proposed SMT+SA approach against the vanilla SA algorithm, the reinforcement learning algorithm, and the surrogate optimization (`surrogateopt`) framework in MATLAB for solving general MINLPs. All our algorithms are implemented on an Intel $i7-7700$HQ @ 2.80GHz machine. We use the receiving schedule and simulation model plant parameters, as suggested in [10]. The receiving schedule is designed to operate the plant for 20 days. Hence, the length of the receiving unit arrays at both the nodes $R_1$ and $R_2$ is 20. Moreover, $d_{\max}$ and $S_{\max}$ are set to 5. The cost coefficients are chosen as $g_1 = 1.8$, $g_2 = 1.2$, $g_3 = 1$, $\beta_0 = 1$, $\beta_1 = 0.5$ and $\beta_2 = 1$, respectively. The receiving schedules of the two receiving stations are shown in Figures 3a and 3b. We assume, with no loss of generality, that the node $R_1$ receives the low quality material, while $R_2$ receives the high quality raw material.

The search space for the SA algorithm is established using the SMT solver. For the constraints specified in Section III-A, the SMT solver produces 72 distinct feasible points of the form $(u_1, u_2, \ldots, u_7)$, where each $u_i$ is a 0 or 1, as opposed to 128 $(2^7)$ possible combinations. Besides narrowing down the




(a) (b) (c)

Fig. 3. Receiving schedule at nodes (a) $R_1$ and (b) $R_2$. Greedy shipout policy at node $D_1$ (c).

**Algorithm 1:** SA for Process Scheduling

**Data:** Receiving schedule $R$, Selling price of produced units, Maximum inventory $d_{\max}$ and holding capacity $S_{\max}$

- Initialize: the starting temperature $T_0$, $iter$ and $\alpha$.
- Use SMT solver for hard constraints to determine the neighbourhood of possible solutions $\mathcal{N}$.
- Select a random point from $\mathcal{N}$ and evaluate the initial objective function value.

**while** *iteration* $\leq$ *iter* **do**

- Select another point in $\mathcal{N}$.
- Evaluate the objective value $P$ for this point

**if** $P \geq$ *old objective* **then**
Change optimal point $S$ to the new point.
**else**

- Evaluate the probability $p = \exp\left(-\frac{\Delta E}{T}\right)$, where $\Delta E$ is the difference in objective
- Choose a random real $n$ between 0 and 1.

**if** $n \geq p$ **then**
Change the optimal point $S$ to the new point
**else**
Remain with the old point as the optimal point $S$

$T = T_0 - \alpha T_0$
iteration = iteration + 1
Determine the greedy ship out policy for the optimal point $S$

search space, the SMT solver also benefits the SA algorithm by making it explore the space of only feasible solutions. After tuning the coefficient for the penalty term in the SA objective, the proposed algorithm achieves the best reward of 28.6 with zero constraint violations. In comparison, the reward obtained in [10] is around 29, however, the RL algorithm in [10] does not account for contiguous operation of machines. We further compare the greedy ship out policy with the optimum shipping policy obtained using the GA in order to minimize the shipping costs for fixed $\{u_i\}$'s. The shipping schedules for both the algorithms are shown in Figure 3c for the nodes $D_1$. The accumulation of reward over

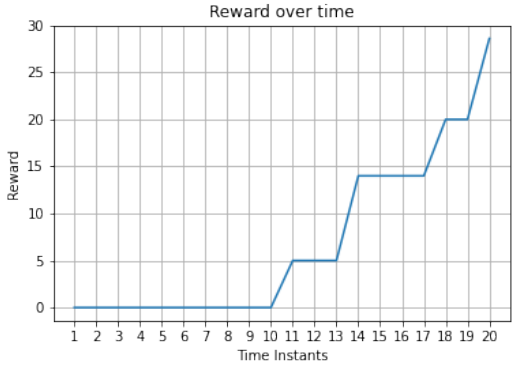
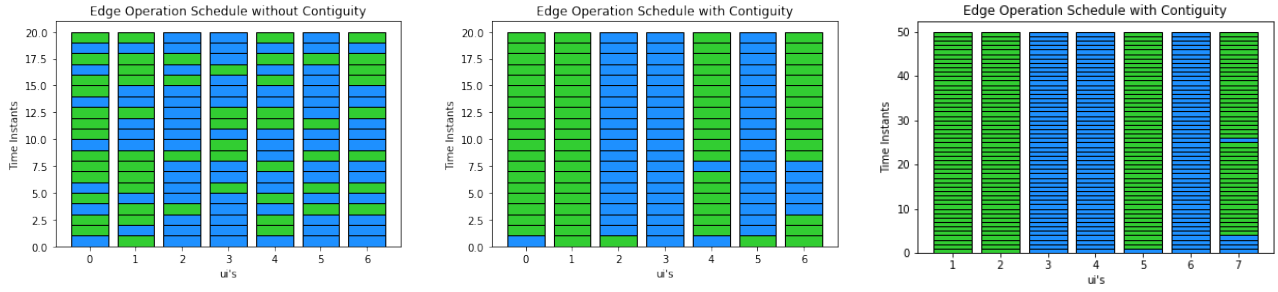



Fig. 4. Accumulation of reward over time.

time is plotted in Figure 4. We also compared our approach with the vanilla SA algorithm (without SMT reduction). After suitable tuning of the cost function coefficients, the vanilla SA algorithm resulted in a reward of 24.8, however, at the expense of discontiguous schedules and nearly 1.7x increase in runtime. The increase in runtime is attributed to the fact that the vanilla SA algorithm does not benefit from working with the otherwise reduced search space generated by the SMT solver. Figures 5a and 5b represent the schedules generated by the vanilla SA algorithm and the SMT+SA (ours). It can be seen that our approach produces contiguous schedule for smooth operation of machines. Interestingly, the surrogate optimization based MINLP solver `surrogateopt` could not return a feasible solution, let alone a sub-optimal solution, despite being executed for more than 5k epochs. The results are summarized in Table I.

A shipping batch distribution of 2 and 5 is suggested by both the algorithms. Our algorithm suggests a batch of 5 to be shipped at time instant 11 while the genetic algorithm suggests the same at time instant 20. Likewise, the batch of 2 is shipped at instants 20 and 16 for the two algorithms. Since the shipping cost is time-invariant, we can conclude that the shipping policy is the same according to both the algorithms. In a similar way, we can observe the shipping schedules from node $D_2$ and $D_3$. For both these shipping nodes, the batches suggested for shipping are the same by both the algorithms.

To exhibit the effect of contiguity on machine operations, we compare the machine schedules for a receiving schedule




(a) (b) (c)

Fig. 5. Machine operation for 20 days generated by (a) vanilla SA: Discontiguous, (b) SMT+SA: Contiguous. (c) Contiguous operation for 50 days.

TABLE I
COMPARISON OF SEVERAL APPROACHES

| Method | Run-time | Cost | Contiguous |
|---|---|---|---|
| Simulated Annealing | 165.10s | 24.8 | No |
| Reinforcement Learning | Unknown | 29 | No |
| SA+SMT (Ours) | 96.34s | 28.6 | Yes |
| `surrogateopt` | Infeasible | Infeasible | Infeasible |

lasting 50 time instants. The machine schedules with and without contiguous scheduling are shown in Figures 5b and 5c. For comparison, the reward with contiguous scheduling is 112.2 and without contiguous scheduling is 124.8. The number of switches overall in the plant pipelines are 4 with contiguous scheduling but increases to 27 without contiguous scheduling. The difference in the two rewards is around 12.5. Both contiguous and non-contiguous solutions can be utilized according to our need. The advantage of using the proposed SMT+SA framework is that we can obtain solutions for both scenarios without having to compromise on the computational advantage of the SA algorithm.

## VII. CONCLUSION

The hierarchical SA algorithm used in this paper is shown to be effective for optimizing process scheduling, especially with problems involving multiple nonlinear constraints. The technique involved in this work uses a SMT solver to satisfy the time-invariant constraints which are fixed for a particular plant design. The dynamic constraints are handled directly by the SA algorithm. The proposed approach, through its inherent construct, also results in contiguous operation of the machines and thus, avoids incessant switching. The proposed algorithm is able to nearly achieve the benchmark solution provided by the RL algorithm despite the additional constraint of generating a contiguous machine schedule. In future, we aim to extend this approach to a broader class of combinatorial optimization problems.

## REFERENCES


[1] C. Vassiliadis and E. Pistikopoulos, "Maintenance scheduling and process optimization under uncertainty," *Computers & Chemical Engineering*, vol. 25, no. 2-3, pp. 217–236, 2001.

[2] A. Legarretaetxebarria, M. Quartulli, I. Olaizola, and M. Serrano, "Optimal scheduling of manufacturing processes across multiple production lines by polynomial optimization and bagged bounded binary knapsack," *International Journal on Interactive Design and Manufacturing (IJIDeM)*, vol. 11, no. 1, pp. 83–91, 2017.

[3] G. P. Georgiadis, A. P. Elekidis, and M. C. Georgiadis, "Optimization-based scheduling for the process industries: from theory to real-life industrial applications," *Processes*, vol. 7, no. 7, p. 438, 2019.

[4] N. Sahinidis and I. E. Grossmann, "Minlp model for cyclic multiproduct scheduling on continuous parallel lines," *Computers & chemical engineering*, vol. 15, no. 2, pp. 85–103, 1991.

[5] C. A. Floudas and X. Lin, "Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications," *Annals of Operations Research*, vol. 139, no. 1, pp. 131–162, 2005.

[6] M. DiNatale and J. A. Stankovic, "Applicability of simulated annealing methods to real-time scheduling and jitter control," in *Proceedings 16th IEEE Real-Time Systems Symposium*. IEEE, 1995, pp. 190–199.

[7] C. Gallo and V. Capozzi, "A simulated annealing algorithm for scheduling problems," *Journal of Applied Mathematics and Physics*, vol. 7, no. 11, pp. 2579–2594, 2019.

[8] P.-C. Wang and W. Korfhage, "Process scheduling using genetic algorithms," in *Proceedings. Seventh IEEE Symposium on Parallel and Distributed Processing*. IEEE, 1995, pp. 638–641.

[9] I. A. Chaudhry and M. Usman, "Integrated process planning and scheduling using genetic algorithms," *Tehnički Vjesnik–Technical Gazette*, vol. 24, no. 5, pp. 1401–1409, 2017.

[10] S. Wagle and A. A. Paranjape, "Use of simulation-aided reinforcement learning for optimal scheduling of operations in industrial plants," in *2020 Winter Simulation Conference (WSC)*. IEEE, 2020, pp. 572–583.

[11] H. D. Sherali and W. P. Adams, "A reformulation-linearization technique (rlt) for semi-infinite and convex programs under mixed 0-1 and general discrete restrictions," *Discrete Applied Mathematics*, vol. 157, no. 6, pp. 1319–1333, 2009.

[12] T. M. Alkhamis and J. Yellen, "Refinery units maintenance scheduling using integer programming," *Applied Mathematical Modelling*, vol. 19, no. 9, pp. 543–549, 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0307904X9500032F

[13] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983. [Online]. Available: https://science.sciencemag.org/content/220/4598/671

[14] S. P. Brooks and B. J. Morgan, "Optimization using simulated annealing," *Journal of the Royal Statistical Society: Series D (The Statistician)*, vol. 44, no. 2, pp. 241–257, 1995.

[15] B. W. Wah, Y. Chen, and T. Wang, "Simulated annealing with asymptotic convergence for nonlinear constrained optimization," *Journal of Global Optimization*, vol. 39, no. 1, pp. 1–37, 2007.

[16] J. Skaruz, A. Niewiadomski, and W. Penczek, "Combining smt and simulated annealing into a hybrid planning method," *Studia Informatica. System and information technology*, vol. 19, no. 1-2, p. 43–48, May 2019. [Online]. Available: https://czasopisma.uph.edu.pl/studiainformatica/article/view/231

[17] P. J. M. van Laarhoven, E. H. L. Aarts, and J. K. Lenstra, "Job shop scheduling by simulated annealing," *Oper. Res.*, vol. 40, pp. 113–125, 1992.

[18] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[19] L. d. Moura and N. Bjørner, "Z3: An efficient smt solver," in *International conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008, pp. 337–340.