Crude Oil Refinery Scheduling: Addressing a Real-World Multiobjective Problem through Genetic Programming and Dominance-based Approaches

Cristiane S Pereira
Pontifical Catholic University - RJ
Petrobras
Rio de Janeiro, Brazil
cristiane.salgado@petrobras.com.br

Douglas M Dias
Rio de Janeiro State University
Rio de Janeiro, Brazil
douglas.dias@uerj.com

Marley M B R Vellasco
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Brazil
marley@ele.puc-rio.br

Francisco Henrique F Viana
Celso Suckow da Fonseca Federal Educational Center
Rio de Janeiro, Brazil
henrique.viana@gmail.com

Luis Martí
Universidade Federal Fluminense
Niterói, RJ, Brazil
lmarti@ic.uff.br

ABSTRACT

This study presents the crude oil scheduling problem with four objectives divided in two different levels of importance. It comes from a real refinery where the scheduling starts on the offloading of ships, encompasses terminal and refinery tanks, a crude pipeline, and finishes on the output streams of the crude distillation units. We propose a new approach for the Quantum-Inspired Grammar-based Linear Genetic Programming (QIGLGP) evolutionary algorithm to handle the multiple objectives of the problem using the non-dominance concept. The modifications are concentrated on the population updating and sorting steps of QIGLGP. We tackle difference of importance among the objectives using the principle of violation of constraints. The problem constraints define if an instruction will or not be executed but do not affect the violation equation of the objectives. The individuals which have objective values under a pre-defined upper limit are better ranked. Results from five scenarios showed that the proposed model was able to significantly increase the percentage of runs with acceptable solutions, achieving success ratio of 100% in 3 cases and over 70% in 2 other ones. They also show that the Pareto front of these accepted runs contains a set of non-dominated solutions that could be analyzed by the decision maker for his *a posteriori* decision.

CCS CONCEPTS

• **Applied computing** → **Multi-criterion optimization and decision-making**; • **Computing methodologies** → *Planning and scheduling*; Genetic programming;

KEYWORDS

Evolutionary Multiobjective Optimization Algorithm; Quantum-Inspired Genetic Programming; Crude oil refinery scheduling;

ACM Reference Format:
Cristiane S Pereira, Douglas M Dias, Marley M B R Vellasco, Francisco Henrique F Viana, and Luis Martí. 2018. Crude Oil Refinery Scheduling: Addressing a Real-World Multiobjective Problem through Genetic Programming and Dominance-based Approaches. In *GECCO '18 Companion: Genetic and Evolutionary Computation Conference Companion, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3205651.3208291

1 INTRODUCTION

Crude oil refineries can be considered the most important example of continuous process systems that generate multiple products – due to the volume and value of these products–, handling billions of dollars every year [18].

Refinery scheduling is the activity responsible for linking the production planning (focus on profit and account weeks ahead) to the process plant operations (focus on short-term activities). In [1], scheduling is defined as the specification of what each stage of production is supposed to do over some short period from several shifts to several days. Usually, the refinery scheduling is shared between the crude oil and product's schedulers. The first one encompasses from the crude oil receiving until the output of crude distillation units (CDUs) while the second one completes the refinery's process areas up the final products.

Scheduling problems have been proved to be NP-Hard [15]. Crude oil scheduling is a mixed-integer nonlinear (MINLP) problem due to the resource allocation (integer), transfer volumes (continuous) and blend properties (nonlinear) characteristics [3, 17]. Some researchers applied mathematical and metaheuristics approaches to optimize solutions for the scheduling problem [3, 19, 25, 27], developing relevant models. However, most of works uses single-objective function, like maximizing profit [28], or aggregation techniques to convert multiple objectives into one expression [20].

We propose a model to tackle a crude oil scheduling problem with four objectives that do not have the same level of importance.

GECCO '18 Companion, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5764-7/18/07…$15.00
https://doi.org/10.1145/3205651.3208291

Two of them impact the processing capacity (CDUs throughput variation and delay on crude oil batches receive) while the others are related to a smooth operation (pipeline operation and tank switchovers). We use Quantum-Inspired Grammar-based Linear Genetic Programming (QIGLGP) [21] as the basic evolutionary algorithm which was modified to handle the multiple objectives of the problem using the non-dominance concept.

The QIGLGP uses a linear structure where each gene represents one crude oil schedule instruction. A domain specific language and a grammar are used to guarantee that those instructions are topologically valid. We modified the sorting step of the algorithm to replace the user preference based approach by a methodology where a nondominated sort, guided by a set of reference points, is used to update the distribution of probabilities of each schedule instruction on each gene. We apply the concept of handling constraints - from multiobjective algorithms - to the objectives evaluation to influence the nondominated sort, which means, two objectives have minimum values that, while not achieved, the correspondent individuals evaluation is like a constrained solution.

In Section 2 we present a background of the three foundations of this work: evolutionary multiobjective optimization algorithms, crude oil refinery scheduling problem, and quantum-inspired grammar-based linear genetic programming. In Section 3 we describe the details of the proposed methodology using many-objective concepts. Section 4 presents the case study and discusses the results obtained. Section 5 consolidates the most important observations about the methodology and the results.

2 BACKGROUND

The concept of evolutionary algorithms (EA) fits with the idea of solving multiobjective problems due to their population-based nature and their less susceptibility to the shape or continuity of the Pareto front [2]. Evolutionary Multiobjective Algorithms (EMOAs) can obtain multiple Pareto optimal solutions in a single simulation run, and their biggest challenge is to achieve a set of solutions under convergence (solutions as close as possible to the Pareto Front - PF) and diversity (well-distributed solutions) criteria [4].

A historical view of the literature shows that the EMOAs were proposed under three categories: Dominance-based, Indicator-based, and Decomposition-based [26], being the first one the most referenced, mainly through the well-known Nondominated Sort Genetic Algorithm II (NSGA-II) [7], Strength Pareto Evolutionary Algorithm 2 (SPEA2) [30], and their variations-based [2]. EMOAs that use the Pareto dominance relation have obtained favorable results when applied to multiobjective optimization problems (MOPs) with up to three objectives [23]. However, if the number of objectives is bigger, the high dimensionality of the objective space diminishes the probability of a solution to be dominated by another one in the population. As a consequence, the dominance criterion cannot impose preferences among solutions. The diminution in the dominance differentiation capability weakens the evolutionary pressure to the Pareto front for many objectives and so the convergence performance is mitigated [11, 13]. Classical EMOAs have been modified and new ones have been proposed to deal with this class of problem, so called many-objective optimization (MaOP).

Multiobjective Evolutionary Algorithm based on Decomposition (MOEA/D) [29] was the first algorithm to reduce the loss of selective pressure by the decomposition of the problem in subproblems. In [16] the authors propose an integration between decomposition and user preference methods where the weighting vector initialization and updating classical approach were modified using a uniform random number generator to remove the coupling between dimension and the population size but providing a set of evenly distributed weight vectors. In the same line, [24] proposes a method where the solutions are ranked by calculating Pareto partial dominance among solutions using r objective functions selected from m objective functions to induce appropriate selection pressure in the evolution process of the EMOA. The authors switch r objective functions, among $\binom{m}{r}$ possible combinations, every g generations to optimize all of the objective functions throughout the entire evolutionary process. Every g generations the Pareto dominance is evaluated considering all the m objectives; then the evolution continues with the next set of r objectives.

The use of predefined target points is the approach used by the Nondominated Sort Genetic Algorithm-III (NSGA-III) [6] to deal with the diversity issue in many-objective algorithms. Initially, a set of well-distributed reference points, which can be user-defined or automatically calculated, is provided to the algorithm. During the evolutionary process, these reference points are responsible for keeping the solutions spread along the Pareto front. The selection operator was modified in such a way that nondominated population members which are close to reference points are emphasized. This algorithm was extended to deal with constraints through the modification of the selection operator so that solutions that do not violate constraints are better ranked than solutions that violate [12]. This constrained NSGA-III also proposes a method to identify non-useful reference points, excluding them and including new ones aiming to keep the set of reference points a good representation of the Pareto optimal solutions. The algorithm EliteNSGA-III [5] was proposed to preserve, through the generations, the best individuals associated with each closest reference point as an indication of a well-distributed Pareto front. The selection operator was modified in order to split the probability that the next generation parents come from the current one or from the archive that preserves the elitist individuals.

Most real-world problems have more than three conflicting objectives and, usually, it is not necessary to know the true Pareto Front, but a good approximation is enough [16]. The crude oil schedule problem is considered here as a MaOP – with four operational objectives – which aims to find a set of good tradeoff solutions.

2.1 Crude Oil Refinery Scheduling Description

Refinery scheduling deals with the forecasted timing of the movements, thus capturing the dynamic nature of actual operations. It starts with the current vision of the refinery. Time and operations move continuously from the beginning to the end of the period. It is always subject to the variability that occurs in the real world, and that is why schedulers must be able to project consequences of disruptions and changes and properly respond to them in a short period of time [14].

Refineries have different scopes of decisions depending on their logistics, geographical location, company owner and so on. The bigger the scope of decisions, the higher the complexity of the scheduling. We analyse a crude oil scope that includes refinery and terminal, which greatly increases the type of activities involved. These activities are:

- CDU feeding: in most cases, each CDU is fed by one or two tanks (injection tank configuration). Schedulers must decide, for every task, if one or two tanks will be used, the flow rate of each tank and the volume consumed on each one. Sequencing this type of task builds the CDUs' schedule.
- Pipeline transferring: this activity transfers crude oil from the terminal tankage to the refinery tankage through a big pipeline. For every task, the scheduler must decide the source tank and the volume that will be pumped into the pipeline and also how this total volume will be received at the terminal tankage (in one or more refinery tanks with the respective tanks' name and volume ratio). This is a coordinated task because the pipeline is always fulfilled and operates in a First-In-First-Out mode. Its operation can be guided by the need for oil in the refinery tankage or to free space on the terminal tankage. The sequencing of this task builds the pipeline schedule.
- Ship offloading: for every ship, it must be completed in a predefined time window. Schedulers have to define which crude oil blend will be first offloaded (if the ship contains more than one), destination tanks and their respective volumes. For many reasons, schedulers may not be able to respect the ship's time window but will incur paying the penalty as a consequence.

To build the schedule, schedulers know, at the beginning of the horizon: the plant (topology, equipments' capacities and tanks' heels), the CDUs (target flow rate and operational mode), properties of all crude oils and their distilled products, tankage inventory (volume and composition), the ships (arrival, time window, flow rate offloading, volume and blend composition) and the constraints (related to properties or operational issues).

Scheduled activities are highly connected because the availability of each equipment for one of them depends on the tasks it was realized before. The sequencing of all these tasks aims four objectives: to keep the CDUs operating with the total flow rate decided by the scheduler; to execute ship offloading inside the time window; to keep the pipeline operating uniformly and to minimize operational transitions made by tanks switchovers. A more detailed discussion of the objectives is presented in section 3.

2.2 Quantum Inspired Grammar-based Linear Genetic Programming

Quantum-Inspired Grammar-based Linear Genetic Programming (QIGLGP) [21] is the basic evolutionary algorithm used in this work but modified to handle multiple objectives using concepts from NSGA-III. Its fundamental entities are a Domain Specific Language (DSL) [8], Quantum gene and Quantum individual, Classical gene and Classical individual, and a Quantum evolutionary operator.

The DSL represents the translation of the most important crude schedule activities into instructions that can be used by the Quantum-inspired genetic programming algorithm to create the program that effectively represents a scheduling solution. The process of generating each instruction follows a predefined grammar that guarantees that none of them will be topologically invalid, e.g., a "feedCDU" instruction will not have a terminal tank as a possible argument. In order to represent all activities, the grammar makes available:

- four instructions for feeding CDU, in a combination of using one or two refinery tanks and the maximum possible volume for the resource(s) or a proportion of this volume;
- two instructions for pipeline transferring, using the maximum possible volume for this task (considering the volume in the terminal tank and the ullage in the refinery tank) or a proportion of this volume;
- two instructions for ship offloading, defining the crude blend that will be offloaded (if the ship has more than one) and the volume (or a proportion of it) of each task considering the ship and terminal tank capacities;
- an instruction that represents doing nothing.

A Quantum Gene (QG) represents the superposition of all possible instructions under the predefined search space. Its basic information unit, *qudit*, can be described by a state vector within a quantum mechanics system of d levels, where d is the number of states on which the qudit may be measured, or observed [10]. Its implementation is equivalent to a tree composed of accumulated probabilities vectors.

A Quantum Individual (QI) is a list of quantum genes. Every quantum gene has the same structure, but the distribution of probabilities will be different depending on the evolutionary process. The Quantum Population (QP) is composed of a set of QIs.

A Classical Gene (CG) is the result of all observations of a quantum gene that are necessary to define the main function and its arguments. For instance, given the main function of pipeline transferring with a proportional volume, the arguments are the terminal tank, refinery tank, and the ratio of the maximum volume that will be transferred. Each parameter, which means, the equipment's names and the volume proportion were defined by the observation of the only one correspondent quantum-gene. Therefore, the classical gene is the complete information, i.e., the instruction with the name of the terminal tank pumping to the pipeline, the name of the refinery tank receiving the crude oil batch and the proportion of the maximum possible volume that is pumped.

A Classical Individual (CI) is a linear disposition of all classical genes. It has the same number of genes as QI, but an instruction that does nothing (named NOp) reduces its effective size. The Classical Population (CP) is made by a set of CIs. Each scheduling program is built by the execution of each CG in a sequenced order defined by its position in the correspondent CI. Despite every CG is a grammatically and topologically valid instruction, as the problem is constrained, it doesn't always represent a feasible instruction. If an instruction violates one or more problem constraint (product quality, minimum duration for a task, etc.), then it is not executed. This instruction is ignored by the simulator which calculates the inventory levels and compositions of the equipment after each task. The set of valid instructions represents the scheduling solution

itself. The search space is proportional to the number of: quantum genes, refinery tanks, terminal tanks, CDUs and the volume ratio parameters of each type of instruction.

The evolutionary process occurs through the Quantum Evolutionary Operator by updating the probability distribution of each quantum gene from the quantum individuals, whose observations resulted in the corresponding best classical individuals. This update is an increment in the probabilities of observing the same elements (function and arguments) that result in that classical gene.[9].

The original QIGLGP algorithm deals with the multiple objectives of the problem following a user predefined hierarchization of them. The decision maker establishes which is the most important objective, followed by the second most important, the third one and the last one. It means that in a comparison between two individuals, the one with the minimum value for the first objective will be the winner. In case of a tie, the second objective will be evaluated to determine the winner and so on, following the hierarchical row of objectives [21]. This hierarchy defines the sorting step of the algorithm. In this work, the objectives are not handled as a pre-defined hierarchical list. The QIGLGP algorithm is modified to tackle the objectives following the Pareto dominance principals. Therefore, at the end of the evolutionary process, the scheduler has a set of useful solution to *a posteriori* decision, selecting which one is more appropriate for a particular occasion.

3 PROPOSED METHODOLOGY

This work uses the same four operational objectives of the original QIGLGP study case: CDU downtime, ship offload, pipeline downtime and tank switchover but with a new perspective. The evolutionary process works to minimize these objectives since they represent undesired conditions.

CDU stoppage is the most critical occurrence in the refinery: if the unit is not processing crude oil, the first intermediate streams are not produced and, as soon as the intermediate stocks go to low-level limits, all the other process units will end up without feedstream. Therefore the entire refinery would stop. The CDU feeding flow rate target is defined by the planning process, but considering the variability of refinery environment, the scheduler has the flexibility to set a value every day, since he still keeps in mind the planning target for the month. CDU downtime objective measures the total time that the CDUs were not operating. Its minimum value is 0 (there was no stopping in the CDU operation) and the maximum value is the horizon multiplied by the number of CDUs.

Although 0 is the ideal value, a 2% of the number of hours of the horizon is acceptable as a good solution because operational variations will absorb this forecasted difference over the next few days. It is important to say that all acceptable solution assume that a 2% deviation does not incur in a real CDU downtime.

An inappropriate ship offloading that ends up in delay to undock the ship will incur in the payment of demurrage which has two aspects of penalty. The first one is economical when the refinery has to pay a tax for occupying the ship beyond the timespan contracted (the value is a function of the size, nationality, and owner of the ship). The second aspect is operational, since an offloading delay may compromise the availability of crude oil in the refinery or terminal and, as a consequence, the refinery may struggle to keep its production level. In this objective, like in CDU downtime, 0 is the ideal value, but again, there is tolerance where small deviations from this ideal can be ignored in real-world problems. Any delay up to one hour is considered an acceptable solution.

Pipeline downtime is related to a non-operation of the pipeline. Pipeline transfer is the operation that guarantees crude oil in the refinery. The scheduler pumps all possible crude oil to the refinery tanks because it diminishes the risk of compromising CDU operation. This rule of thumb is based on the concept that the crude oil at the refinery guarantees the volume and the quality needed to make the blends that achieve property specifications for being processed. It is not rare that pipeline tasks sum a lesser number of hours compared to the entire horizon but, usually, it operates uninterruptedly in its reduced horizon. The pressure along the equipment keeps the batches compacted, avoiding blending among them and the information about the composition of this new blend be unknown. To consider these two aspects (the total operation time and the number of stoppages and restarts in the horizon), the pipeline downtime objective was modeled with two terms. The integer part of the objective value represents the total number of hours the pipeline did not operate, and the fractional part represents the number of times that the pipeline stopped and restarted during the horizon. There is no ideal value for this specific objective, and it is less important than the CDU downtime or the ship offload.

Unnecessary tank switchovers represent inappropriate schedule decision since any operational movement brings a transient state to the resources. For instance, every tank in the refinery has different properties; so, when a process unit has its feeding task changed, it will face a transient operation. The bigger the difference between the properties of the tanks, the more important may be this transient state. Another aspect is related to the schedule itself because whenever a refinery tank receives crude oil from the pipeline, it must wait a settling time of some hours while brine (that comes mixed with the crude oil batch) is decanted. After that, if the feeding unit task does not empty the tank and receive another crude oil batch, a new settling time must be waited so that a new feeding task can be done. In practical terms, the tank will stay more hours unavailable for the same volume fed. As it happened with the pipeline downtime, the tank switchover objective does not have a desired or even estimated value. Good scheduling solutions may come from some different number of tank switchover and, certainly, this objective is less important than the CDU downtime and the ship offloading, but it can be considered at the same level of the pipeline downtime.

When a solution has the CDU downtime less than 2% of the total horizon and Ship offloading lesser than 1 hour, it is considered an acceptable solution, which means a solution that represents a feasible schedule.

The results presented in [22] showed that, in different runs, QIGLGP produced different acceptable schedules using a user-preference based approach. However, it is not always easy to define a hierarchical sequence among the objectives. This strict need of definition is the main limitation of the method. For instance, a small downtime in the CDU operation probably is a better solution if the ship offloading is appropriate when compared to another solution without CDU downtime but a significant ship offloading delay. This work proposes an improvement in the original algorithm to handle

better with the tradeoffs of the objectives. It is a many-objective approach in which the evolutionary process searches for the best nondominated solutions that meet the conditions under which these solutions are operationally acceptable.

The many-objective concepts adopted in this work are inspired by [6], where a set of points is used as a reference in the population sort stage to help the evolutionary algorithm to achieve a set of well-distributed solutions along the Pareto front. From [12] comes to the second inspiration, where the approach adopted to handle constraints is adapted to deal with the different importance of the objectives. Algorithm 1 presents the new approach for QIGLGP where the steps 8 up to 21 of algorithm replaced the original hierarchical sorting step of the population. In the modified QIGLGP, the population is sorted based on the non-dominance of individuals, considering that individuals which meet the two critical objectives are better ranked than the ones which do not.

In Algorithm 1, NI is the number of individuals in each population, g is the current generation, nGP is the total number of generations (stop criteria), $horizon$ is the number of hours of the scenario, FPf is the current Pareto front index, K is the number of individuals to complete CP_g.

Equation (1) presents the violation measure that relates the two critical objectives with the maximum value they could assume in the worst case. On equation 1 $CDUdwt$ and $ShipOffld$ are the values of the critical objectives, $nCDU$ is the total number of CDUs and $VolShip$ is the sum of volume from every ship that was not completely offloaded at the end of the scenario.

$$\frac{CDUdwt}{nCDUs \times horizon} + \frac{ShipOffld}{\sum_{n=1}^{nShips} VolShip_n}, \tag{1}$$

4 RESULTS

The case study comes from a real Brazilian refinery, whose crude oil area contemplates docking the ship in a dedicated terminal, pumping the crude oil through a large pipeline, feeding the CDUs and ends with intermediate streams produced by the distillation units. The model could be applied to any scenario from this refinery or others topologically similar refineries. We decided to use the same five scenarios (CEN01, CEN02, CEN03, CEN04, and CEN05) from [22], to allow a comparative analysis of the proposed many-objective and the predefined user-preference based approaches. These scenarios have different levels of complexity to allow a better evaluation of the model. The proposed approach deals with *a posteriori* evaluation of the decision maker, while [22] is based on *a priori* analysis done by this same decision maker.

CEN01 and CEN02 are the least complex ones, with a shorter scheduling horizon (192 h). The first one is in a more comfortable condition of stock level and has only one ship to offload. The second one brings a little more complexity to the problem since its initial inventory is lower and has two ships to offload (i.e. more tasks to be scheduled but helpful with the inventory issue). CEN03 brings more complexity through its higher horizon (216 h) and only one ship to be offloaded with the equivalent volume of the two ships in CEN02. This scenario demands more attention from the scheduler, to guarantee enough space in the refinery to make possible pumping from the terminal. It is necessary to free space on terminal tanks to receive the massive amount of oil in a small time window.

Algorithm 1 QIGLGP based on nondominance of individuals influenced by acceptance criteria.

1: Initialize NI individuals QP and CP for generation $g = 0$
2: Calculate the fitness functions for $\mathrm{CP}_{g=0}$
3: $g \leftarrow g + 1$
4: Calculate the set of reference points
5: **while** $g \leq nGP$ **do**
6: Create auxiliary classical population (ACP_g) based on the observation of QP_{g-1}
7: Calculate the fitness functions for CP_g
8: Nondominated sort of individuals from $\mathrm{CP}_{g-1} \cup \mathrm{ACP}_g$ influenced by acceptance criteria, which means that if the two individuals being compared have CDUdowntime lesser then ($0.02 \times horizon$) and ShipOffloading lesser then 1 hour, then the basic dominance comparison will be applied to rank them. Otherwise, if only one of the individuals meets the acceptance criteria, this one dominates the other one. The last condition is that if both individuals do not meet the acceptance criteria, the one with lesser violation will dominate the other individual.
9: $FPf \leftarrow 0$
10: $|PC_g| \leftarrow 0$
11: **while** $|PC_g| + |PC_{FPf}| \leq NI$ **do**
12: $PC_g \leftarrow (PC_g) \cup (PC_{\mathrm{FPf}})$
13: $FPf \leftarrow FPf + 1$
14: **end while**
15: **if** $|PC_g| < NI$ **then**
16: Number of individuals to fill CP_g (K), coming from PC_{FPf}: $K = (NI - |PC_g|)$
17: Normalize each objective.
18: Associate every individual from PC_g to the respective nearest reference point.
19: Select K individuals from PC_{FPf} through a mapping to those closest to the reference points with smaller number of PC_g individuals associated.
20: Add these K individuals to PC_g.
21: **end if**
22: Apply Quantum operator (QOp) to the NI quantum individuals according to the CI correspondent
23: **if** Fitness function of all individuals of $\mathrm{CP_g}$ and $\mathrm{ACP_g}$ have the same value **then**
24: Reinitialize QP (similar steps 1 and 2)
25: **end if**
26: $g \leftarrow g + 1$
27: **end while**

CEN04 has the same standards of CEN03 but is harder because the inventory level in the terminal and refinery is much higher, which makes the transference coordination more challenging. CEN05 and CEN04 have the largest horizon (240h), but they are opposite in the inventory level issue. In CEN05, the refinery tankage and the terminal tankage are below what is considered a safe inventory level, which means there is a real risk to compromise CDUs operation. Two big ships must be offloaded, what is helpful volumetrically speaking but one must keep in mind that a terminal tank cannot receive from a ship and pump to pipeline simultaneously. So, the

Table 1: Percentage of runs with Acceptable Solutions.

Approach	CEN01	CEN02	CEN03	CEN04	CEN05
Manyobjective	100	100	100	94	76
User-preference	98	100	100	76	36

first tank will start to pump only when it is filled (some hours later). Then, after the refinery tank receives crude oil, there is still a settling time of 24 h before the tank can feed a CDU. For all these issues, CEN05 is considered the most complex scenario.

Each scenario was run 50 times with different seed for the random number generator as evaluation criteria of the model. The population size was defined as 56, following the methodology used by NSGA-III [6] to calculate the number of reference points in a problem with four objectives. In this particular case, 56 is the minimum possible value considering the parameterization of the methodology. Other QIGLGP parameters were 0.008 for the quantum operator step and 0.8 for the initial ratio of the instruction NOp (do nothing).

Two aspects are considered in the model analysis. The first one is the capacity of the model to generate the Pareto front with, at least, one acceptable individual considering the critical objectives. The second one is the diversity of different scheduling solutions present on the Pareto front.

At the end of the evolutionary process, the Pareto front should be constituted by nondominated individuals with lesser than 2% of CDU downtime and lesser than one hour of ship offloading or by the individual with the least violation. The total of runs (among 50) that results in the first case represents the percentage of runs with acceptable solutions. Table 1 presents the results of the many-objective and the user-preference based approaches.

Results on Table 1 show an increase in the ability of the proposed algorithm to find solutions that attend most critical objectives, especially in the hardest scenarios. CEN04 improved in 23% and CEN05 in 111% their performance. This Table 1 is concerned with one accepted individual and does not provide any information about diversity. In order to provide the scheduler conditions to evaluate and take *a posteriori* decision, it is necessary to expose the different solutions that compose the Pareto front of these accepted runs.

Table 2 presents, for those runs which obtained acceptable solutions, the mean, minimum and maximum number of individuals on the Pareto front. For these individuals, the same table also shows the statistics for the values of the objectives.

An analysis of Table 2 shows average values from 3 up to 5 individuals on the Pareto front, depending on the scenario complexity. A human being can differentiate and analyze this number of alternatives to select the most appropriate one considering the conditions at the moment of the decision. If many solutions are in the Pareto front, it would be recommended to identify the most different ones to be presented to the scheduler. A methodology to identify them is a suggestion for future work. Table 2 also presents high values for standard deviation compared to the mean value for three objectives. This behavior suggests the samples are distributed throughout the boundaries of the objectives values. For instance, CEN03 has found a solution with only 24 hours of pipeline downtime, and also it has found a solution with 121 hours of pipeline downtime. CEN04 was scheduled with 15 and also with 35 tanks switchovers. All of them are feasible solutions but represent different schedules.

Table 2: Mean, minimum and maximum number of individuals, and their objectives' values, on the Pareto Front for the runs with acceptable solutions.

		CEN01	CEN02	CEN03	CEN04	CEN05
Number of solutions on Pareto front	Min	1	2	1	1	1
	Mean	4.78	4.72	3.56	3.47	2.84
	Max	12	12	13	11	10
CDU downtime	Min	0	0	0	0	0
	Mean	1.32	1.67	1.21	1.50	1.64
	stDev	1.21	1.57	1.42	1.60	1.37
	Max	3.73	3.80	4.16	4.69	4.78
Ship offloading	Min	0	0	0	0	0
	Mean	0.004	0.027	0.020	0.053	0.013
	stDev	0.063	0.149	0.114	0.192	0.126
	Max	0.969	0.852	0.759	0.959	0.934
Pipeline downtime	Min	0.001	24.001	14.002	3.001	18.001
	Mean	19.166	34.777	40.076	28.539	49.207
	stDev	14.608	15.382	23.880	28.371	25.858
	Max	106.004	140.002	121.004	120.003	128.004
Tank switchovers	Min	12	7	11	15	16
	Mean	15.6	11.1	17.8	20.8	23.2
	stDev	2.1	1.9	3.2	4.7	3.0
	Max	26	18	27	35	32

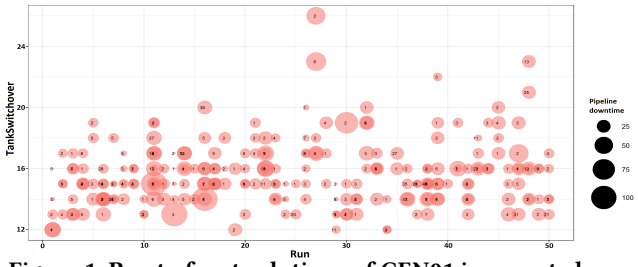

Figure 1: Pareto front solutions of CEN01 in accepted runs.

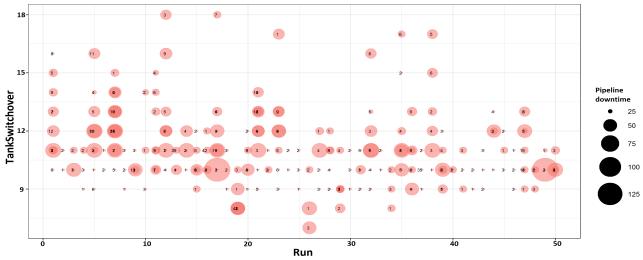

Figure 2: Pareto front solutions of CEN02 in accepted runs.

Figures 1 to 5 show the solutions of the Pareto front for each accepted run for the five scenarios. The CDUdowntime and ShipOffload objectives are not plotted in the chart to make the visualization easier but, certainly, they are under the acceptance conditions for these individuals. The abscissa axis shows the run ID, and the ordinate axis shows the number of tank switchovers. The size of the bubble indicates the range of pipeline downtime and the number inside the bubble indicates the index of the individual. These are the points used in the results presented in Table 2

Figures 1 to 3 show that CEN01, CEN02, and CEN03 have solutions concentrated in a smaller range when compared to CEN04 and CEN05. For instance, CEN01, CEN02, and CEN03 have tank switchover objective lower than 25, while CEN04, which is 24% hours longer, goes up to 35, which means 40% more.

Figure 6 shows the hypervolume metric along the evolutionary process. The reference point adopted is an upper limit over the

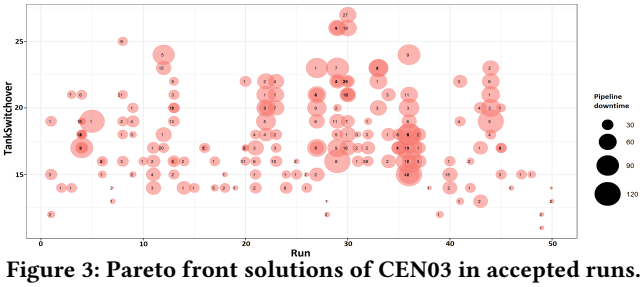

Figure 3: Pareto front solutions of CEN03 in accepted runs.

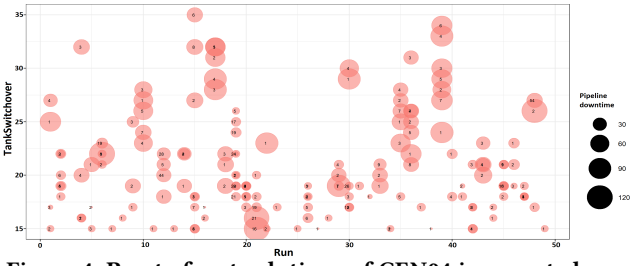

Figure 4: Pareto front solutions of CEN04 in accepted runs.

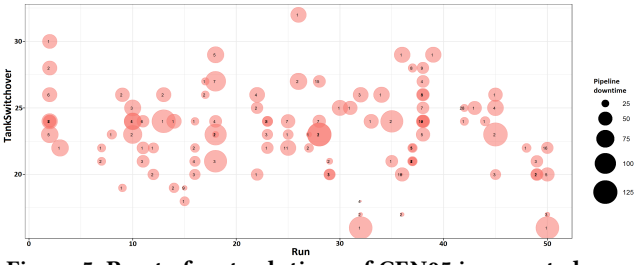

Figure 5: Pareto front solutions of CEN05 in accepted runs.

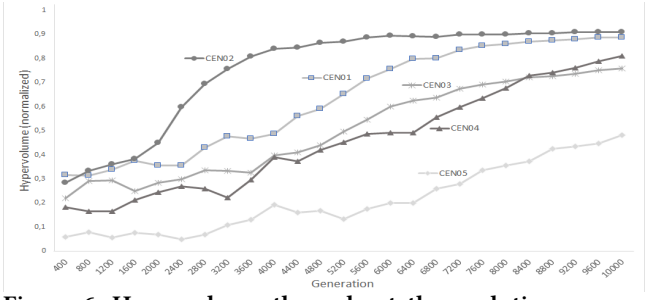

Figure 6: Hypervolume throughout the evolutionary process for the five scenarios.

search space, so, whenever hypervolume increases it means the evolution is progressing. The hypervolume was calculated considering the mean of individuals on the Pareto front of the runs with acceptable solutions.We can note that CEN02 is the fastest to achieve the convergence of the evolutionary process, with a significant increase in the metric near the generation 2,000. CEN01, CEN03, and CEN04 have similar behavior, with a significant increase near the generation 4,500. However, CEN01 and CEN03 apparently achieve the convergence, while CEN04 probably would get better results if the stop criteria were a bigger number of generations. CEN05 is under its evolutionary process because its hypervolume is still increasing and there is no signal reaching a plateau.

We analyzed the number of individuals in Pareto front throughout the generations. We could prove that constraint violation concept adopted made the evolutionary process be similar a user-preference based approach while all individuals are over the upper limit of the objectives. Based on the violation equation, while the ships are not entirely offloaded, this will be the most significant component of the equation. After that, CDU downtime will be the responsible for defining the individuals ranking and, the last one component will be the ships offloading delay. Only after the evolutionary process produces feasible individuals, the non-dominance concept is used.

Figure 7 presents two examples of schedules generated by CEN03. A bar with a label 'Receive' means that the equipment is receiving from one source, and a bar with a label 'Send' means that the equipment is sending to one destiny. A bar with a label 'Recieve-2' means that the equipment is receiving from two sources. A light gray bar means the settling time that is needed whenever a refinery tank will feed a CDU after being the destiny of a pipeline transfer. The dark gray bars are related to the pipeline transfers whatever be the one that connects terminal and refinery or the subsea pipeline.

We can note that both solutions have the same value for the Ship offloading objective, but one of them has a smaller CDU downtime and pipeline downtime. On the other hand, the other schedule does more transferences, and it has a bigger value for the tank switchover objective. Both of them are feasible solutions and can be adopted by the scheduler after *a posteriori* evaluation. Then, he can consider aspects like the shift, or the day of the week when some movements will be done, as well as the final blend in tanks.

5 CONCLUSION

The evolutionary algorithm proposed - QIGLGP modified - has proved to be able to create crude oil scheduling solutions for the five studied scenarios. We used two metrics for this analysis: the success ratio of the algorithm to produce at least one feasible solution, and the number of different solutions on the Pareto front.

The results showed the algorithm good performance under the first metric perspective since three scenarios were always able to create feasible solutions; one scenario did that in 94% of runs, while the hardest one did that in 76% of runs. Considering the second metric, Figures 1 to 5 showed that the algorithm was able to create the Pareto front solutions distributed along the objective space. A comparative analysis among the cases shows that the easiest scenarios are closer to the convergence since their hypervolume metric is reaching a plateau as presented on Figure 6.

The many-objective approach of QIGLGP can find an average set of 3 to 5 different scheduling solutions, which can be presented to the scheduler so that he can take *a posteriori* decision after analyzing the nondominated alternatives. Figure 7 shows two examples of schedules generated.

As future work, we consider the use of elitism concept inspired on [5] to develop a methodology to preserve the most diverse solutions to be presented to the scheduler.

6 ACKNOWLEDGMENTS

We thank PETROBRAS for sponsoring this study and give us all the information and support needed for its development.

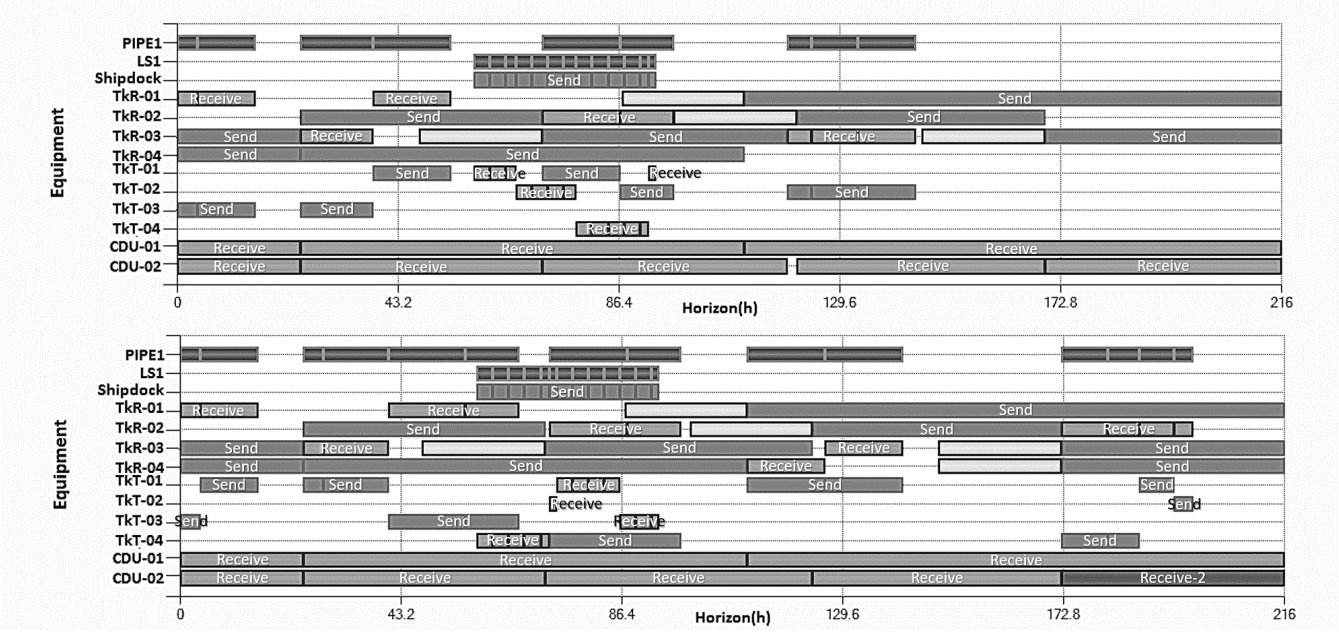

Figure 7: Two examples of schedule solutions for CEN03.

REFERENCES

[1] C. E. Bodington. 1995. *Planning, scheduling and control integration in the process industries* (5th ed.). McGraw-Hill, USA.

[2] C. A. C. Coello. 2006. Evolutionary multi-objective optimization: a historical view of the field. *IEEE computational intelligence magazine* 1 (2006), 28–36.

[3] D. D. S. Cruz. 2007. *Refinery Scheduling Using Genetic Algorithms: A Study for Crude Oil Scheduling Case.* Master thesis. COPPE/UFRJ, Rio de Janeiro.

[4] K. Deb. 2001. *Multi-objective Optimization using Evolutionary Algorithms* (1st ed.). John Wiley & Sons, England.

[5] K. Deb, A. Ibrahim, M. V. Martin, and S. Rahnamayan. 2016. Elite NSGA-III: An Improved Evolutionary Many-Objective Optimization Algorithm. *IEEE Congress on Evolutionary Computation* 63 (2016), 973–982.

[6] K. Deb and H. Jain. 2014. An Evolutionary Many-Objective Optimization Algorithm Using Reference-point Based Non-dominated Sorting Approach, Part I: Solving Problems with Box Constraints. *IEEE Transactions on Evolutionary Computation* 18 (2014), 577–601.

[7] K. Deb, A. Pratap, S. Agarwal, and T Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6 (2002), 182–197.

[8] A. V. Deursen and P. Klint. 2002. Domain Specific Language Design Requires Feature Descriptions. *Journal of Computing and Information Technology* 10, n.1 (2002), 1–17.

[9] D.M. Dias and M. A. C. Pacheco. 2009. Toward a Quantum-Inspired Linear Genetic Programming Model. *Evolutionary Computation Conference* 1 (2009), 1691–1698.

[10] Douglas Mota Dias and Marco Aurélio Cavalcanti Pacheco. 2013. Quantum-Inspired Linear Genetic Programming as a Knowledge Management System. *Comput. J.* 56, 9 (2013), 1043–1062.

[11] E. M. N. Figueiredo, T. B. Ludermir, and C. J. A. Bastos Filho. 2016. Many Objective Particle Swarm Optimization. *Information Science* 53 (2016), 1689–1700.

[12] H. Jain and K. Deb. 2014. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, Part II: Handling constraints and extending to an adaptive approach. *IEEE Transactions on Evolutionary Computation* 18 (2014), 602–622.

[13] B. Li, J. Li, K. Tang, and X Yao. 2015. Many-objective evolutionary algorithms: a survey. *Comput. Surveys* 48 (2015), 13:1–13:35.

[14] M. V. O. Magalhães. 2004. *Refinery Scheduling.* Doctorate dissertation. Department of Chemical Engineering and Chemical Technology, Imperial College, London.

[15] A. Masood, Y. Mei, G Chen, and M. Zhang. 2016. Many-Objective Genetic Programming for Job-Shop Scheduling. *IEEE Congress on Evolutionary Computation* 63 (2016), 219–216.

[16] A. Mohammadi, M. N. Omidvar, X. Li, and K. Deb. 2014. Integrating user preferences and decomposition methods for many-objective optimization. *IEEE Congress on Evolutionary Computation (CEC)* 1 (2014), 421–428.

[17] Lincoln FL Moro and José M Pinto. 2004. Mixed-integer programming approach for short-term crude oil scheduling. *Industrial & engineering chemistry research* 43, 1 (2004), 85–94.

[18] L. F. L. Moro. 2000. *Técnicas de Otimização Mista-Inteira para o Planejamento e Programação de produção em Refinarias de Petróleo.* Doctorate dissertation. Escola Politecnica, Universidade de Sao Paulo, Sao Paulo.

[19] S. Mouret, I. Grossmann, and P. Pestiaux. 2011. A new Lagrangian decomposition approach applied to the integration of refinery planning and crude-oil scheduling. *Computers and Chemical Engineering* 35 (2011), 2750–2766.

[20] F. Oliveira, M. R. Almeida, and S. Hamacher. 2008. Genetic Algorithms Applied to Scheduling and Optimization of Refinery Operations. *Proceedings of the VI ALIO/EURO Workshop on Applied Combinatorial Optimization* 6 (2008), 182–197.

[21] C. S. Pereira. 2012. *Petroleum Scheduling Multiobjective Optimization for Refinery by Genetic Programming using Domain Specific Language.* Master Thesis - in Portuguese. Department of Electrical Engineering, Pontificia Universidade Catolica do Rio de Janeiro, Rio de Janeiro, Brazil.

[22] C. S. Pereira, D. M. Dias, M. Vellasco, E. H. Hollmann, A. V. A. da Cruz, and M. A. C. Pacheco. [n. d.]. Crude oil scheduling of a refinery by Genetic Programming in Domain-Specific Language. *submitted to Engineering Applications of Artificial Inteligence* ([n. d.]).

[23] P. J. Purshouse, R. C.; Fleming. 2007. On the evolutionary optimization of many conflicting objectives. *IEEE Transactions on Evolutionary Computation* 11 (2007), 770–784.

[24] H. Sato, H. E. Aguirre, and K. Tanaka. 2010. Pareto partial dominance MOEA in many-objective optimization. *IEEE Congress on Evolutionary Computation (CEC)* 1 (2010), 1–8.

[25] N. K. Shah and M. G. Ierapetritou. 2015. Lagrangian decomposition approach to scheduling large-scale refinery operations. *Computers and Chemical Engineering* 79 (2015), 1–29.

[26] A. Trivedi, D. Srinivasan, K. Sanyal, and A. Ghosh. 2016. A Survey of Multiobjective Evolutionary Algorithms based on Decomposition. *IEEE computational intelligence magazine* 21 (2016), 440–462.

[27] F. Wu, N.and Chu, C. Chu, and M. Zhou. 2009. Short-Term Schedulability Analysis of Multiple Distiller Crude Oil Operations in Refinery With Oil. *IEEE Transactions on Systems, Man and Cybernetics* 39 (2009), 1–16.

[28] J. Xu, S. Zhang, J. Zhang, S. Wang, and Q. Xu. 2017. Simultaneous scheduling of front-end crude transfer and refinery processing. *Computers and Chemical Engineering* 96 (2017), 212–236.

[29] Q. Zhang and H. Li. 2007. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation* 11 (2007), 712–731.

[30] E. Zitzler, M. Laumanns, and L. Thiele. 2001. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems* 1 (2001), 2–21.