Computers and Chemical Engineering 133 (2020) 106636

Contents lists available at ScienceDirect

Computers and Chemical Engineering

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

Scheduling of gasoline blending and distribution using graphical genetic algorithm

Feleke Bayu[a], Debashish Panda[a], Munawar A. Shaik[a,b], Manojkumar Ramteke[a,*]

[a] Department of Chemical Engineering, Indian Institute of Technology Delhi, India
[b] Department of Chemical and Petroleum Engineering, UAE University, Al Ain, United Arab Emirates

ARTICLE INFO

Article history:
Received 16 August 2019
Revised 31 October 2019
Accepted 5 November 2019
Available online 7 November 2019

Keywords:
Genetic algorithm
Gasoline blending
Scheduling
Multi-objective optimization

ABSTRACT

Scheduling of gasoline blending, and distribution (SGBD) involves allocating resources and sequencing the operations to give gasoline a high economic potential without compromising its quality and the customers' demand. The existence of nonlinearity and the need for multi-objective optimization makes SGBD complex. In this study, a graphical genetic algorithm (GGA) model involving a discrete-time representation is developed for both single- and multi-objective SGBD. In the single-objective formulation, the production cost is minimized, whereas in the multi-objective formulation, the sum of the square of fluctuation in inter-period blending rate is additionally minimized. The efficacy of the proposed model is checked by solving three industrial problems which involve the production of 20, 35 and 45 orders of different gasoline grades, respectively over the time-horizon of 8 days. The proposed model gives lower production cost compared to MINLP formulation and the reduction found to be increasing with the increase in problem size.

1. Introduction

Gasoline accounts for 43% of the total crude oil products and more than 70% of the total crude oil products' profit margin (World Energy Outlook, 2018). The world's energy future mainly depends on world population size and gross domestic product (GDP). By 2040, the world population estimated to be around 9.2 billion and both the GDP and the middle-class population will likely be double (Rocha et al., 2014; World Energy Outlook, 2018). Associated with this growth, the International Energy Agency's (IEA) has estimated (Rocha et al., 2014) that the total crude oil demand will increase from 95 Mb/d to 115 Mb/d between the year 2016 and 2040. Crude oil, being the main energy source (about 25% of the total world's energy demand) today and in the coming decades, optimizing its production process is an extremely relevant problem.

A crude oil refinery is a complex process (Jia et al., 2003) which involves both physical and chemical processes. It mainly consists of separation, conversion and treatment sections. In the separation section, a desalted, dehydrated and heated crude mix enters distillation and breaks into gases, light naphtha, heavy naphtha, jet fuel, kerosene, diesel oil, gas oil and residual-based on their cut point ranges. Following the separation, in the conversion stage, heavy and low-value distillation products are further processed to lighter and higher value products such as gasoline. Cracking, alkylation, isomerization and reformation are the main processes in the conversion section. In the cracking unit, the heavy hydrocarbon molecules break into lighter one either catalytically or thermally. In alkylation, isobutane and low-molecular-weight alkenes are converted to a high-octane gasoline components. In reforming and isomerization process, straight-chain hydrocarbon molecules are transformed into high-octane rating molecules such as branched-chain alkanes, cycloalkanes and aromatic hydrocarbons. Once the components (e.g. butane, light gasoline, coker and cracked gasoline, alkylate, reformate and isomerate) are produced from different conversion units and distillation, they are temporarily stored in different component tanks. The treatment section mainly involves the gasoline blending in which the components are blended in different proportion based on the product specifications and other requirements as shown in Fig. 1 to produce different grades of gasoline. Since gasoline produced must have an even combustion pattern, must satisfy prevailing environmental regulations and should have an easy starting of combustion in cold weather, blending becomes a crucial operation. Further, adding blending agents and additives (List of Registered Gasoline Additives, 2019) enhances the gasoline property indices [i.e. octane number (ON), Reid vapour pressure (RVP), anti-nocking and stability specifications, sulphur content, ASTM distillation point and flash point].

* Corresponding author.
E-mail address: mcramteke@chemical.iitd.ac.in (M. Ramteke).

https://doi.org/10.1016/j.compchemeng.2019.106636
0098-1354/

Nomenclature

Sets

U	units
I	component tanks
N	blenders
J	product tanks
K	periods
P	product types
O	orders
P_I	properties index

Positive variables

$b(n,i,k)$	amount of component i delivered to blender n at the period k in *kbbl*
$ct(n,k)$	changeover on blender n at the end of period k
$ct(j,k)$	changeover on product tank j at the end of period k
$x(n,k)$	a product flowrate from blender n to a product tank at the end of period k
$f(n,i,k)$	amount of component transferred to a blender n from component tank i at the end of period k
$f(j,o,k)$	amount of product delivered to an order o from product tank j at the end of period k
$cc(i,k)$	an inventory level in a component tank i at the end of period k
$cb(n,k)$	a blending rate in a blender n at the end of period k
$cpt(j,k)$	an inventory level in a product tank i at the end of period k
$T(o)$	tardiness of an order o
$Od(o,k)$	amount of product delivered to order o at the end of period k

Binary variable

$z(n,p,k)$	for assigning of a product p on a blender b during a period k
$x(u_1,u_2,k)$	for assigning of a material (components or products) transfer from unit u_1 to unit u_2 (units or orders) during a period k

Parameters

$c(i)$	unit cost of component i *($/kbbl)*
$cbr(n)$	changeover cost of in blender n *($)*
$cp(j)$	changeover cost of in product tank j *($)*
cr	tardiness cost for orders
$C(i)^{\max}$, $C(i)^{\min}$	maximum and minimum allowable inventory level in a component tank i, respectively
$C(j)^{\max}$, $C(j)^{\min}$	maximum and minimum allowable inventory level in a product tank j, respectively
$C(n)^{\max}$, $C(n)^{\min}$	maximum and minimum allowable blending rate in a blender n, respectively
$f(i)^{\max}$, $f(i)^{\min}$	maximum and minimum allowable flow rate from a component tank i, respectively
$f(o)^{\max}$, $f(o)^{\min}$	maximum and minimum allowable flow rate from a blender o, respectively
$px(p,p_i)^{\max}$, $px(p,p_i)^{\min}$	a product p's maximum and minimum property index p_i, respectively
ℓ_{str}	number of binaries used for representing the weight corresponding to each edge in CGA
$y(i)^{\max}$, $y(p)^{\min}$	a product p's maximum and minimum allowable compositions from each component tank i, respectively

After blending, gasoline has to be distributed in a way it fulfils the customer demands in time and quantity. Having good SGBD is an indispensable tool to make a refining plant profitable and competent. Additionally, maintaining the consistency of the product quality is equally crucial as improving profitability. Maintaining the product consistency not only enables the reduction of the high cost incurred due to quality giveaway but also increases the operating ability and controllability of the process. Thus, multiple objectives are naturally involved in the SGBD problem. Further, the mixing of components in the tanks leads to the nonlinear formulation. For such non-linear and multi-objective optimization formulation, the use of stochastic algorithms such genetic algorithm is more suitable compared to traditional MINLP approaches. The detailed literature review on these approaches is given in the next section.

In this study, a GGA, an extension of structure adapted genetic algorithm (SAGA) (Ramteke and Srinivasan, 2012) is used for optimizing the SGBD problem. The proposed model uses discrete-time representation (period-wise) where the scheduling horizon is divided into several equal time (6 h each) intervals. The cost of SGBD is minimized in a single objective optimization which is then extended for multi-objective optimization to additionally minimize the inter-period blending rate fluctuation in the consecutive periods for better operability and control. For such multi-objective optimization of SGBD, although the stochastic methods such as GA are the most suitable choice, the research in this direction is explored in a limited manner in the literature.

2. Literature review

Recently, solving gasoline blending has received significant attention. Both discrete- and continuous-time formulations were used for modeling the gasoline blending. Among these, the discrete-time formulation is easy to formulate and simple to represent but it leads to the larger formulation size compared to the continuous-time formulation. Though the continuous-time formulation requires a lower number of event points, it generally leads to higher complexity. Glismann and Gruhn (2000) developed a mixed-integer nonlinear programming (MINLP) model that sequentially integrates recipe planning with gasoline scheduling. Jia et al., (2003) and Jia and Ierapetritou (2003) proposed MILP formulations based on a continuous-time event for SGBD. Although they have addressed different operational features, their models are limited only to identical parallel blenders, constant recipes and product specifications. Some of these limitations such as handling variable recipes and product specifications were resolved by Carlos et al. (2006). They proposed both discrete-time and slot based continuous-time MILP models. It uses an iterative algorithm to linearize the nonlinearities of the gasoline properties. To incorporate many operating features and policies such as constant piecewise component flows and qualities, multiple delivery dates for multi-product orders, constant blending rates in a run, minimum lengths for blending runs, multipurpose product tanks and so on which prevail in real life, Li et al. (2010) and Li and Karimi (2011) presented a slot based continuous-time MILP models. To linearize the inherently nonlinear properties, they used

ingenious schedule adjustment procedure. Recently, Cerdá et al., 2016) have proposed a floating slot based continuous-time MINLP model for solving SGBD. Similarly, Castillo et al., (2017) presented piecewise relaxation and normalization techniques to compute the global optimum for SGBD.

The above literature review suggests that the inherent nonlinearity is the major factor that increases the computational difficulty while solving the SGBD. Further, the procedures used to handle this nonlinearity adds complexity to the computation and results in a giveaway to the global optimum. As a result, for the large size problems, solving becomes difficult. Lastly, optimizing multiple objectives in SGBD is difficult using mathematical programming approaches.

To optimize such complex applications, the stochastic approaches are more relevant due to their abilities to handle nonlinear and multi-objective formulations easily. Recently, the stochastic approaches, particularly applying a genetic algorithm (GA) to large industrial problems has got significant attention (Deb et al., 2002; Kumar Koratiya et al., 2010; Ramteke and Srinivasan, 2012; Sarkar and Modak, 2005). Though only the limited studies are reported on refinery processing and gasoline blending using GA (Ahsan, 2015; Ivanov and Ray, 2014; Khosla et al., 2007), several of these are used (Almeida, 2011; Bornapour et al., 2017; Hou et al., 2017; Kasat and Gupta, 2003; Khosravi et al., 2015; Simao et al., 2007; Panda and Ramteke, 2019, 2018) for other scheduling operations. Ramteke and Srinivasan (2012) developed a structured adapted genetic algorithm (SAGA) which is a population-based optimization algorithm and applied it to solve crude oil scheduling problem in marine access refineries. Thereafter, Panda and Ramteke (2019, 2018) further extended the SAGA procedure to reactive crude oil scheduling under multiple uncertainties and preventive crude oil scheduling under demand uncertainty.

3. Problem definition

To produce gasoline with pre-specified properties, the optimizer must determine the amount of input from each component tank to a blender in such a way that it minimizes the cost without compromising the quality of the product and operational requirements. It also determines the schedule of feeding the product tanks and order delivery during their delivery windows. A detailed description of gasoline blending, and distribution problem (Cerdá et al., 2016b) is stated as follows.

Given information:

1. Initial inventory in tanks (component and product) and the capacity of each tank.
2. Flow rates of the streams (from component tanks to blenders, blenders to product tanks and product tanks to orders).
3. Scheduling time and order delivery windows.
4. A unit cost of each component, tardiness cost and product changeover costs in blenders and product tanks.
5. Maximum and minimum product composition and blending rate limits.
6. Property indexes of each component and product.
7. The dedicated products for each product tank.
8. Constituent product for each order and their required amounts.

Operating rules

1. At a time, a component tank can supply to multiple blenders and multiple component tanks can cater to a blender.
2. At a time, a product tank can receive a product from multiple blenders.
3. At a time, a product tank can deliver a product to multiple orders and order can receive a product from multiple product tanks.
4. All orders must be delivered during the scheduling horizon.

Determine:

1. The product produced by the blenders and their blending rates in each period.
2. The amount of each component the blenders receive in each period.
3. The type and amount of product a product tank receives from blender (s) in each period.
4. The amount of product delivered to each order from product tanks in each period.
5. The inventory level in the tanks at the end of every period.

Constraints

1. During the scheduling horizon, a blender can deliver to multiple product tanks but only to one at a time.
2. At a time, a blender can process at most one product.
3. For a product tank, simultaneous product receiving and delivering is not allowed.
4. Inventory level in tanks, as well as the blending rate in blenders, must be maintained within the maximum (C_i^{max}) and minimum (C_i^{min}) limits.
5. Flow rates of streams must be within the maximum (f_i^{max}) and minimum(f_i^{min}) limits.
6. The demand (D^o) amounts should be fulfiled.
7. For each product, the required product property indices must be within the maximum ($^{P}{}_i^{max}$) and minimum ($^{P}{}_i^{min}$) limit.
8. For each product, the required product composition must be within the maximum ($y(p)^{max}$) and minimum ($y(p)^{min}$) limit.
9. In each unit (tanks and blenders) and for all orders, material balance equations must be fulfiled.
10. In the above list of constraints, constraints 1–3 are the operational constraints.

Assumptions

1. Every blending run should happen within a period.
2. Perfect mixing in a blender.
3. The product changeover costs in blenders and product tanks are sequence dependent.
4. The changeover time in blenders is incorporated in processing time whereas the same for product tanks assumed to be negligible.
5. Product certification does not require additional time.

Objectives

In a single-objective optimization, the objective is to minimize the production cost which is the sum of component costs, transition costs (in blenders and product tanks) and tardiness costs during the schedule horizon whereas an additional objective of minimizing the blending rate fluctuation between successive periods is used as a second objective in multi-objective optimization.

The following section presents the mathematical formulations used to define the above description.

4. Mathematical formulation

As mentioned in Section 3, a gasoline blending, and distribution scheduling model is formulated for both single and multi-objective formulation with a discrete-time (period-wise) representation where the time horizon is divided into several equal time intervals. Each time interval is of 6 h. During a single objective optimization, minimizing the production cost (PC) is the objective.

$$\min PC = \begin{bmatrix} \sum_n \sum_i \sum_k f(n,i,k) \times \Delta t \times c(i) + \sum_n \sum_k ct(n,k) \times cbr(n) \\ + \sum_j \sum_k ct(j,k) \times cp(j) + \sum_o T(o) \times Cr \end{bmatrix} \quad (1)$$

The objective function (Eq. (1)) is the summation of component costs, product changeover costs in blenders, product changeover costs in product tanks and tardiness costs due to delay in order delivery. The tardiness of the older delivery is obtained using Eq. (2).

$$T(o) = \max[0, (T_{dd}(o) - T_{uw}(o))] \forall o \in O \quad (2)$$

Further, in multi-objective optimization, an additional objective of minimizing the sum of the square of successive period blending rate fluctuation (δ^2) is used as an additional objective. (Eqs. (1) and (3) represents the two objectives)

$$\min \delta^2 = \sum_n \sum_k [x(n,k) - x(n,k-1)]^2 \forall k > 1 \quad (3)$$

The above objective functions are subjected to constraints 1–9 listed in Section 3. Among these, the first three constraints determine the connections between the units and are readily satisfied in the GGA procedure. Therefore, the mathematical formulations of these are described in Section 5.1. in the context of GGA. However, the remaining constraints 4–9 are handled either using the dynamic bounds or the penalty function approach in the GGA procedure (discussed in Section 5.2) and their mathematical formulations are given next.

Capacity constraints (constraint 4)**:** The inventory level in component and product tanks must be kept within the allowable limits. Further, the blending amount per period in the blenders has to be within the allowable range. These constraints are given mathematically as follows:

$$C(i)^{\min} \le Cc(i,k) \le C(i)^{\max} \forall i \in I, k \in K \quad (4)$$

$$C(j)^{\min} \le Cpt(j,k) \le C(j)^{\max} \forall j \in J, k \in K \quad (5)$$

$$C(n)^{\min} \le Cb(n,k) \le C(n)^{\max} \forall n \in N, k \in K \quad (6)$$

Flow rate constraints (constraint 5)**:** The component tanks can feed a blender within allowable feeding rate range. Similarly, product delivery is restricted within the allowable flow rate range to constitute an order. The mathematical expressions are given as follows:

$$F(i)^{\min} \le \sum_b f(n,i,k) \le F(i)^{\max} \forall n \in N, k \in K \quad (7)$$

$$F(o)^{\min} \le \sum_j f(j,o,k) \le F(o)^{\max} \forall o \in O, k \in K \quad (8)$$

Demand constraint (constraint 6)**:** The total amount of product delivered to an order o must fulfil the demand as given by the following equation.

$$D(o) = \sum_k Od(o,k) \forall o \in O \quad (9)$$

Property index constraints (constraint 7)**:** The following equation maintains the product property indices within the bounds.

$$cb(n,k) \times px(p,p_i)^{\min} \le \sum_i px(i,p_i) \times f(n,i,k) \times \Delta t \le cb(n,k) \times px(p,p_i)^{\max} \quad \forall n \in N, p \in P, p_i \in P_l, k \in K \quad (10)$$

Product composition constraints (constraint 8)**:** For each period k, the following constraint maintains the composition of a product in each blender within its bounds.

$$\sum_i f(n,i,k) \times y(p)^{\min} \times \Delta t \le cb(n,k) \le \sum_i f(n,i,k) \times y(p)^{\max} \times \Delta t \; \forall p \in P, n \in N, k \in K \quad (11)$$

Material balance for blenders (constraint 9)**:** The material balance in a blender n during a period k is given as follows:

$$cb(n,k) = cb(n,k-1) + \sum_{\rho(i,p)>0} \sum_i f(n,i,k) \times \Delta t - \sum_{\rho(n,p)<0} x(n,k) \forall n \in N, p \in P, k > 1 \quad (12)$$

$$cb(n,k) = \sum_{\rho(i,p)>0} \sum_i f(n,i,k) \times \Delta t \forall n \in N, p \in P, k = 1 \quad (13)$$

The Eqs. (12) and (13) determine the amount of product inventory present in a blender n during a period k by deducting the amount transferred to a product tank from the sum of the amount already existing in the blender and the amount transferred to the blender from component tanks.

Material balance for product tanks (constraint 9)**:** The material balance in a product tank j at the end of a period k is given as follows:

$$cpt(j,k) = cpt(j,k-1) - \sum_{\rho(i,p)<0} \sum_o f(j,o,k) \times \Delta t + \sum_{\rho(j,p)>0} x(j,k) \forall j \in J, p \in P, k > 1 \quad (14)$$

$$cpt(j,k) = cpt(j,0) - \sum_{\rho(i,p)<0} \sum_o f(j,o,k) \times \Delta t \forall j \in J, p \in P, k = 1 \quad (15)$$

The Eqs. (14) and (15) determine the inventory level in a product tank j at the end of a period k by subtracting the amount of product delivered to the orders from the cumulative of the already stored amount and the amount received in that period from the blender.

Material balance for orders delivered (constraint 9)**:** The amount of order delivered at the end of period k builds upon the existing one by adding the amount delivered during period k as defined by the equation below.

$$Od(o,k) = Od(o,k-1) + \sum_j f(j,o,k) \times \Delta t \forall o \in O, k \in K \quad (16)$$

5. Methodology

SGBD is one of the complex problems which involve several units working together in parallel as well as in a sequencial manner. Here, to solve SGBD, a GGA method is used. As Ramteke and Srinivasan (2012) illustrated, the conventional GA (NSGA-II) (Ashish Gujrathi, 2016; Deb et al., 2002; Rangaiah, 2009) has two major drawbacks for solving the large-scale scheduling problems. First, the use of penalty functions for constraints dilutes the ability of the fitness function to move in its optimum search direction. This makes the conventional GA inefficient in the presence of a large number of constraints. Second, the chromosome is comprised of both zero and nonzero valued variables accounting the combinatorial search space which leads to a large chromosome size. Such large chromosome size, in turn, leads to an increase in problem complexity and makes the algorithm memory inefficient. These drawbacks are mitigated in the graphical-based genetic algorithm.

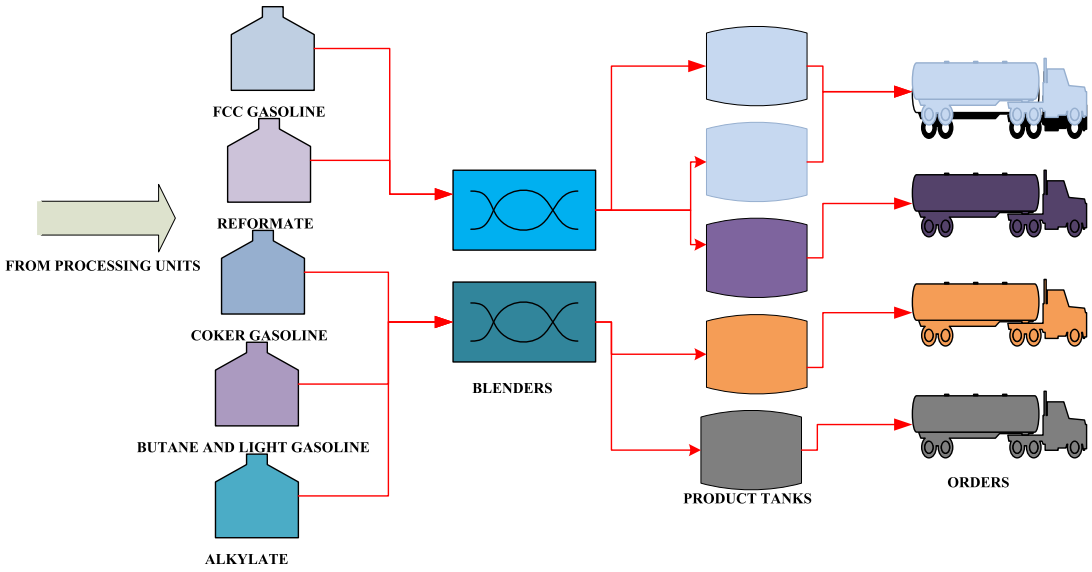

Fig. 1. Schematic of gasoline blending units.

5.1. Concept of size reduction in GGA

GGA uses the graphical facets of the SGBD network to circumvent the above limitations of the conventional GA. A flowchart of the method is given in Fig. 2. In GGA, the unit is represented as a *vertex* and the connection/link (material flow) as an *edge*. Therefore, the graphical representation encompasses a set of circles for vertices connected by lines for the edges. A source is a vertex whose all edges directed outward, whereas a sink has all edges directed inward. Depth is the layer of vertices from the source; the source has zero depth and the depth increases while moving toward the sink. The procedure used for the GGA will be elaborated using illustrative sample problem given in Fig. 3(A). It involves three components tanks (CT_1–CT_3), a blender (B_1), two product tanks (PT_1, PT_2), four orders (O_1–O_4) and two gasoline product types (P_1 & P_2: represented by green and violate colors). Order three and four (O_3 and O_4) require P_1, whereas the order one and two (O_1 and O_2) require P_2. As shown in Fig. 3(B), the superstructure of the problem consists of at most 23 [(number of component tanks × number of blenders) + +(number of blenders × number of product tanks × number of product types) + +(number of product tanks × number of product orders × number of product types) = =(3 × 1) + +(1 × 2 × 2) + +(2 × 2 × 4)] edges without considering for feasibility (constraints) in the structure. In the parenthesis, the first term represents the total number of connections (e_1, e_2, e_3) between three-component tanks and one blender. The second term represents the total number of connections (e_4, e_5, e_{14}, e_{15}) between one blender and two product tanks for two types of products. Similarly, the third term represents the total number of connections (e_7, e_9, e_{11}, e_{13}, e_{17}, e_{19}, e_{21}, e_{23}, e_6, e_8, e_{10}, e_{12}, e_{16}, e_{18}, e_{20}, and e_{22}) between two product tanks and four orders for two product types.

The proposed model uses discrete-time representation (period-wise) where the time horizon is divided into several equal time intervals. However, to satisfy the given structural constraints (1–3) and operating rules, some of the edges of the super-graph of SGBD network to necessarily have a zero-flow rate in each period. The super-graph omitted with such zero-flow rate edges in each period gives a corresponding sub-graph. A period-wise concatenating of such sub-graphs represents a structural schedule. The mathematical formulation of constraints 1–3 which are used to generate sub-graphs is given below.

According to constraint 2, the blender can supply the product to one tank in a time period. The constraint is represented in terms of edges as follows:

$$\sum_{j} e_{n,j,k} \leq 1, \forall n \in N, k \in K \tag{17}$$

where, $e_{n,j,k}$ is a binary variable representing an edge between a blender n and a product tank j at period k. The above equation assures that a blender at period k at most can feed one product tank.

According to constraint 2, at most one product can be processed at a time in a blender. The mathematical formulation is given below:

$$\sum_{p} e_{n,p,k} \leq 1, \forall n \in N, k \in K \tag{18}$$

where, $e_{n,p,k}$ is a binary variable which represents the blender n is producing the product type p at any period k.

According to constraint 3, simultaneous product receiving and delivering is not allowed for a product tank. The mathematical description is given as follows:

$$\left(\left[\sum_{o} e_{j,o,k}\right] \times \left[\sum_{n} e_{n,j,k}\right]\right) = 0, \forall j \in J, k \in K \tag{19}$$

where, $e_{j,o,k}$ represents an edge delivering order o from product tank j at period k. Similarly, $e_{n,j,k}$ represents an edge from blender n to the same product tank j at period k.

5.2. Initial feasible schedules generation

The GGA starts with an initialization step in which feasible (i.e. satisfying the structural constraints and operating rules) Np structural sub-chromosomes and their corresponding Np weight sub-chromosomes are randomly generated. For structural sub-chromosomes, the non-zero edges present in the SGBD network are generated randomly for each period. A periodwise arrangement of such edges in a string represents a structural sub-chromosome. Further, to *avoid* variable-length chromosomes, the length of the

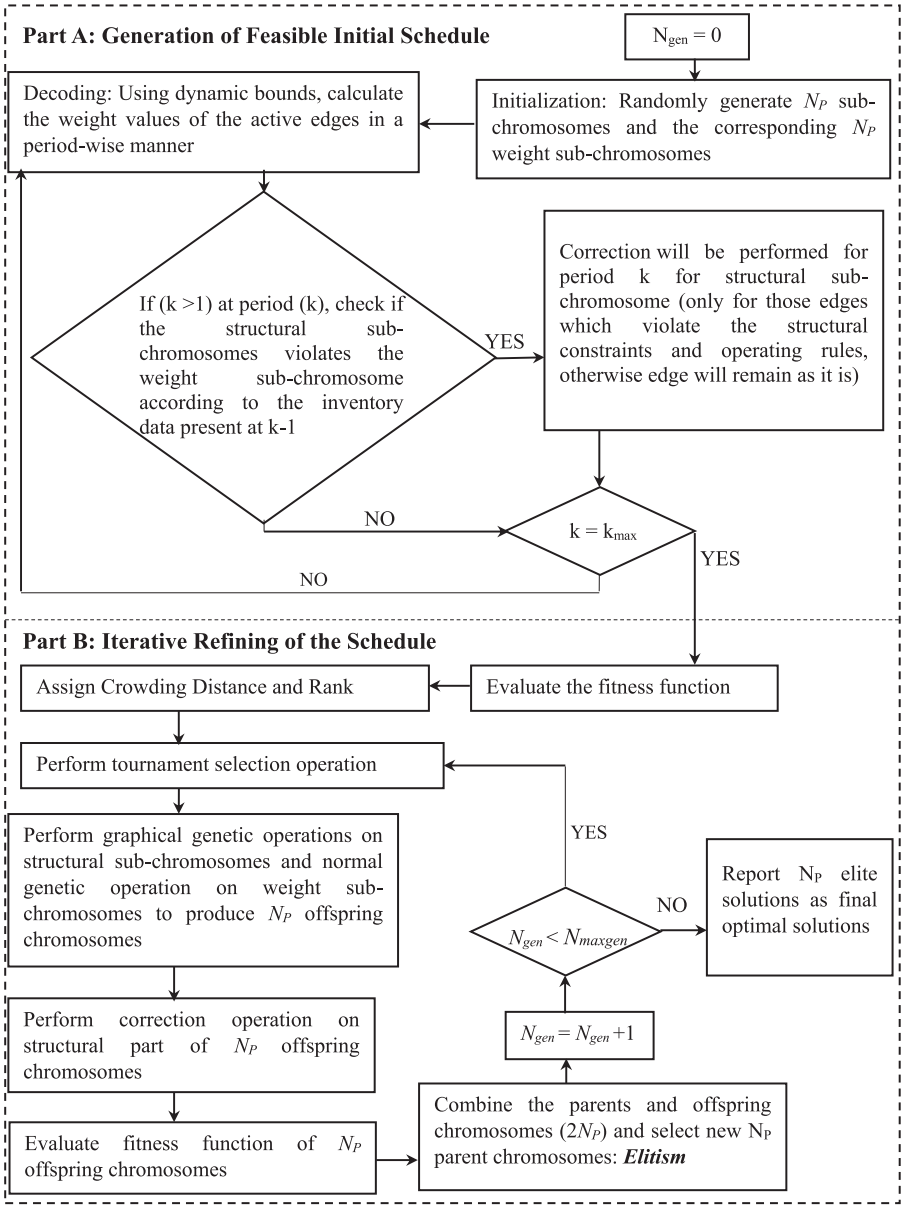

Fig. 2. Flowchart of the graphical GA for solving SGBD problem.

structural sub-chromosome is fixed to a *maximum* size of the feasible sub-graph (i.e. the maximum size of a sub-graph that satisfies the structural constraints) multiplied by the number of periods. Since the actual number of non-zero edges may be lower than the maximum number of slots fixed, empty spaces may present in the structural sub-chromosome as some of the slots may remain empty. Consider a feasible structure with zero flow rate edges omitted from the SGBD network of the illustrative problem as shown in Fig. 3(C). Here, the blender can receive component from all three component tanks and hence e_1, e_2 and e_3 can be present in one time period. For the connections between the blender and the product tanks, two cases arise depending on whether the product delivery to tanks will be present or not. In case (a), blender is not supplying the product to the product tanks. Therefore, the product tanks are available for catering the product orders (constraint 3). Here, between the product tanks and product orders, maximum two active edges can be present from each set of edges {e_7, e_9, e_{11}, e_{13}, e_{17}, e_{19}, e_{21}, e_{23}} and {e_6, e_8, e_{10}, e_{12}, e_{16}, e_{18}, e_{20}, e_{22}} since product tank can have either product one or two which can cater to maximum two product orders as per problem description (operating rule 3). Therefore, maximum four active edges can be present for two product tanks. Further, the edges connecting blender to product tanks e_4, e_5, e_{14} and e_{15} have to be absent (constraint 3). In case (b), blender is supplying the product to the product tank, only one edge will be active from the set of {e_4, e_5, e_{14} and e_{15}} since a blender can produce one product at a time and can supply the product to one product tank (constraint 1 and 2). This product tank cannot deliver the order (constraint 3) i.e. maximum two edges from either sets {e_7, e_9, e_{11}, e_{13}, e_{17}, e_{19}, e_{21}, e_{23}} or {e_6, e_8, e_{10}, e_{12}, e_{16}, e_{18}, e_{20}, e_{22}} will be present. Therefore, for the illustrative problem, the maximum possible size of structural sub-graph at any period is 7 (3 + 4) after considering the constraints mentioned in Section 3. Hence, to get the size of structural sub-chromosome, the sub-graph size must be multiplied by the number of periods; $7 \times 3 = 21$. The size of a chromosome will be twice the size of the structural sub-chromosome (the sum of variables on the structural and weight sub-chromosomes). Therefore, the size of the chromosome in the GGA becomes 42. On the other

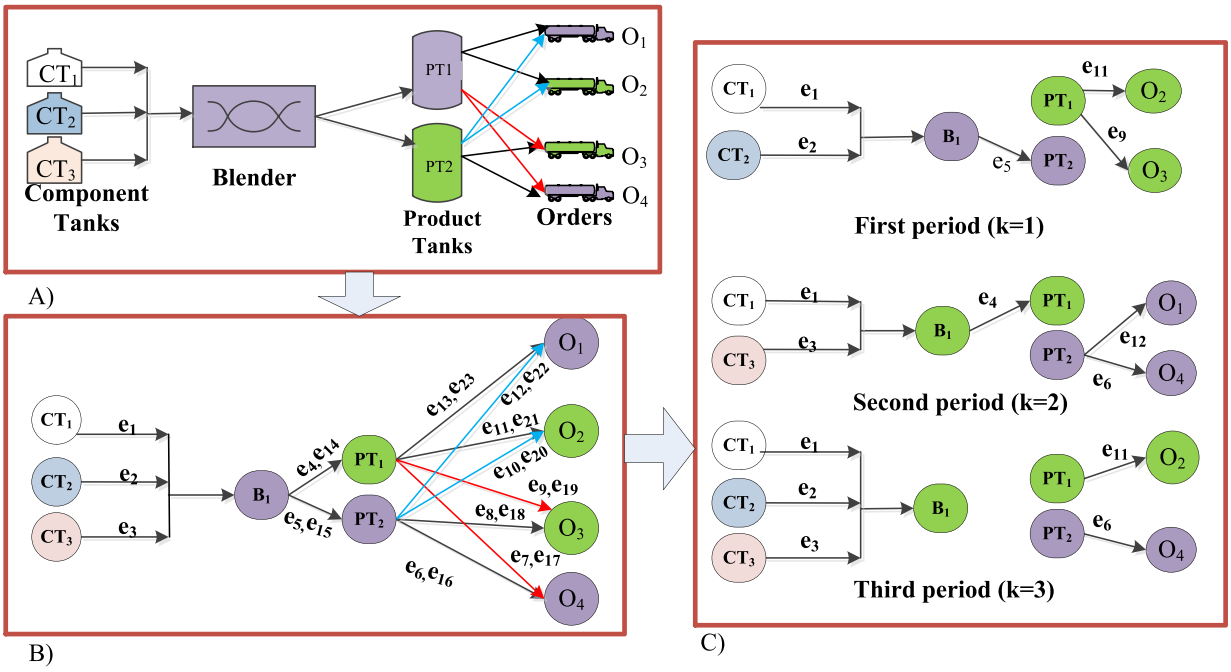

Fig. 3. Graphical representation for a sample scheduling problem. (For interpretation of the references to color in text, the reader is referred to the web version of this article.)

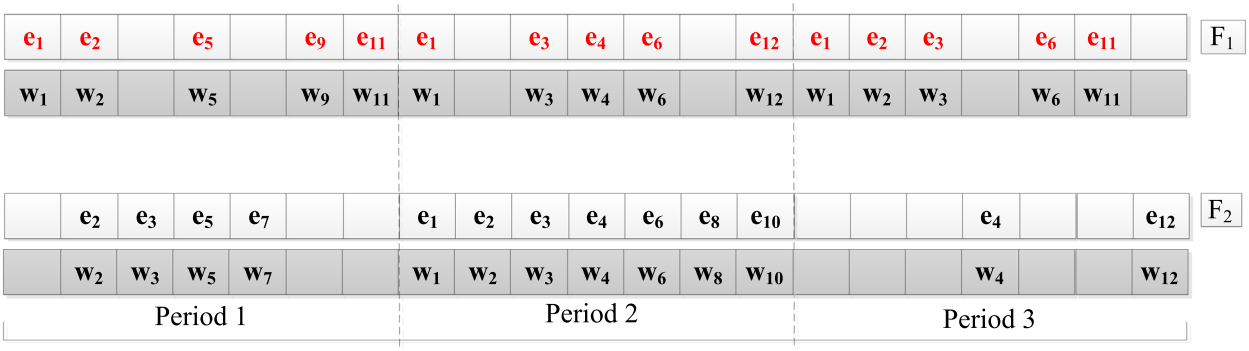

Fig. 4. Two possible Chromosomes (F_1 *and* F_2) for the illustrative problem.

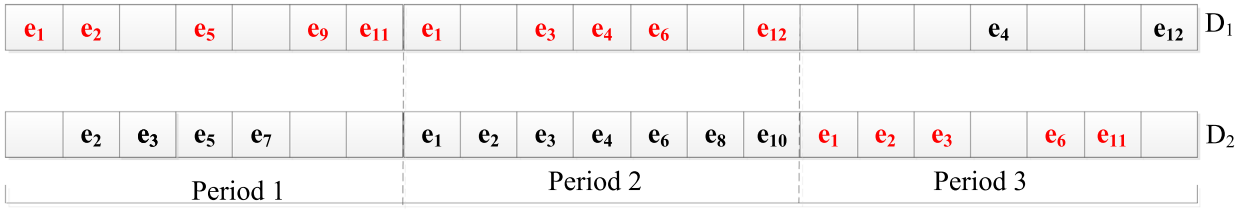

Fig. 5. Structural crossover.

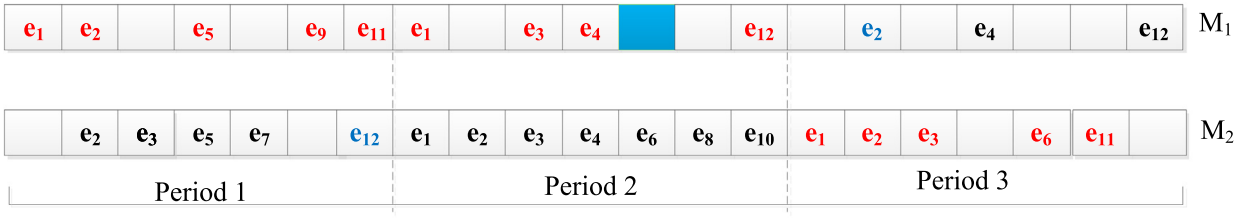

Fig. 6. Structural mutation (the new edges added are shown by blue color whereas the deleted edges are represented by empty blue colored boxes). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

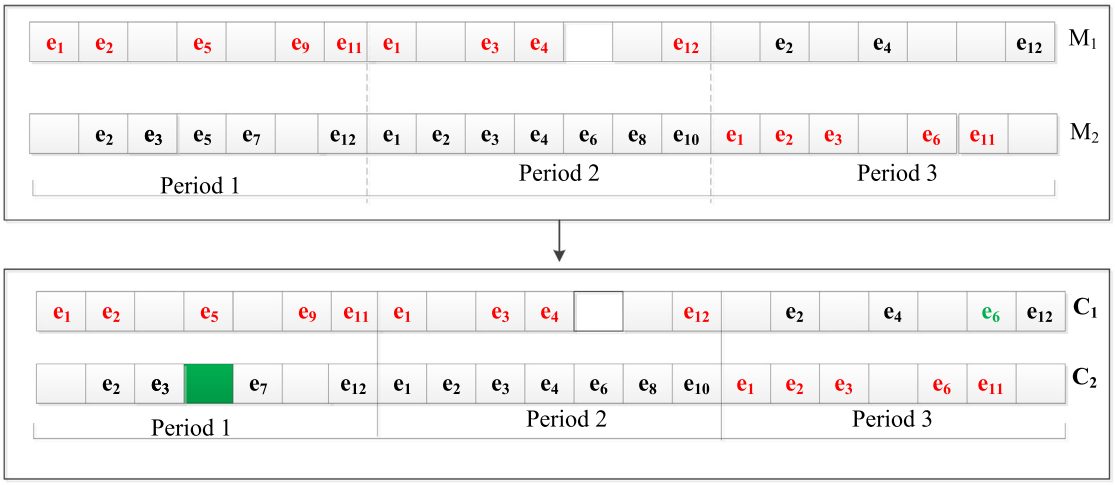

Fig. 7. GGA correction for the illustrative problem (the new edges added are shown by green color whereas the deleted edges are represented by empty green colored boxes). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

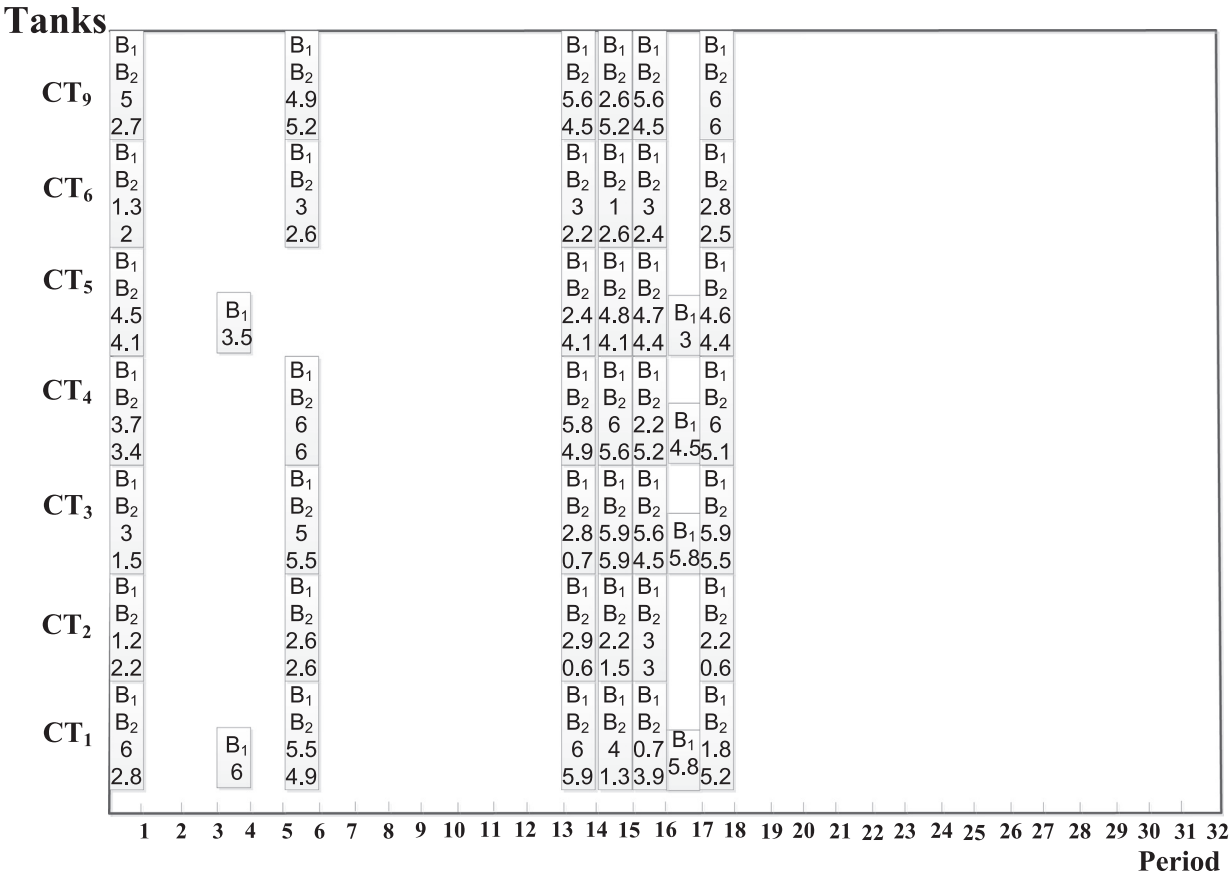

Fig. 8. Feeding schedule of the blenders from the component tanks for the first problem (inside the boxes, the notation at the top denotes the blender number whereas the numerical values at the bottom denote the amount of components transferred from the component tanks to the blender).

hand, for the conventional GA, the chromosome size becomes 69 (23 × 3). Clearly, the graphical GA reduces the chromosome size by 39.1% for the illustrative example.

During the initialization stage, weight sub-chromosomes are generated using the usual procedure of conventional GA, where, l_{str}, set of binaries are generated randomly for the weight of each edge. Next, the decoding step is used where the binaries are converted to real values using dynamic bound generation. In this operation, the weights of the edges are calculated in a period-wise manner. While assigning the dynamic bounds, the lower and upper bounds of the weight variables are changed as per the material balance is given in Eq. (20).

$$\begin{aligned} O(o,k)^{high} &= D(o) - Od(o,k) \\ O(o,k)^{low} &= f(o)^{\min} \end{aligned} \tag{20}$$

Here, $O(o, k)^{high}$ and $O(o, k)^{low}$ are the upper and lower bounds on order delivery rate for order o in period k. $D(o)$ is the demand for order o, $Od(o,k)$ is the order o that has been delivered up to period k, $f(o)^{\min}$ is the minimum allowable flow rate for order o. Therefore, at the end of decoding, all N_P chromosomes carrying

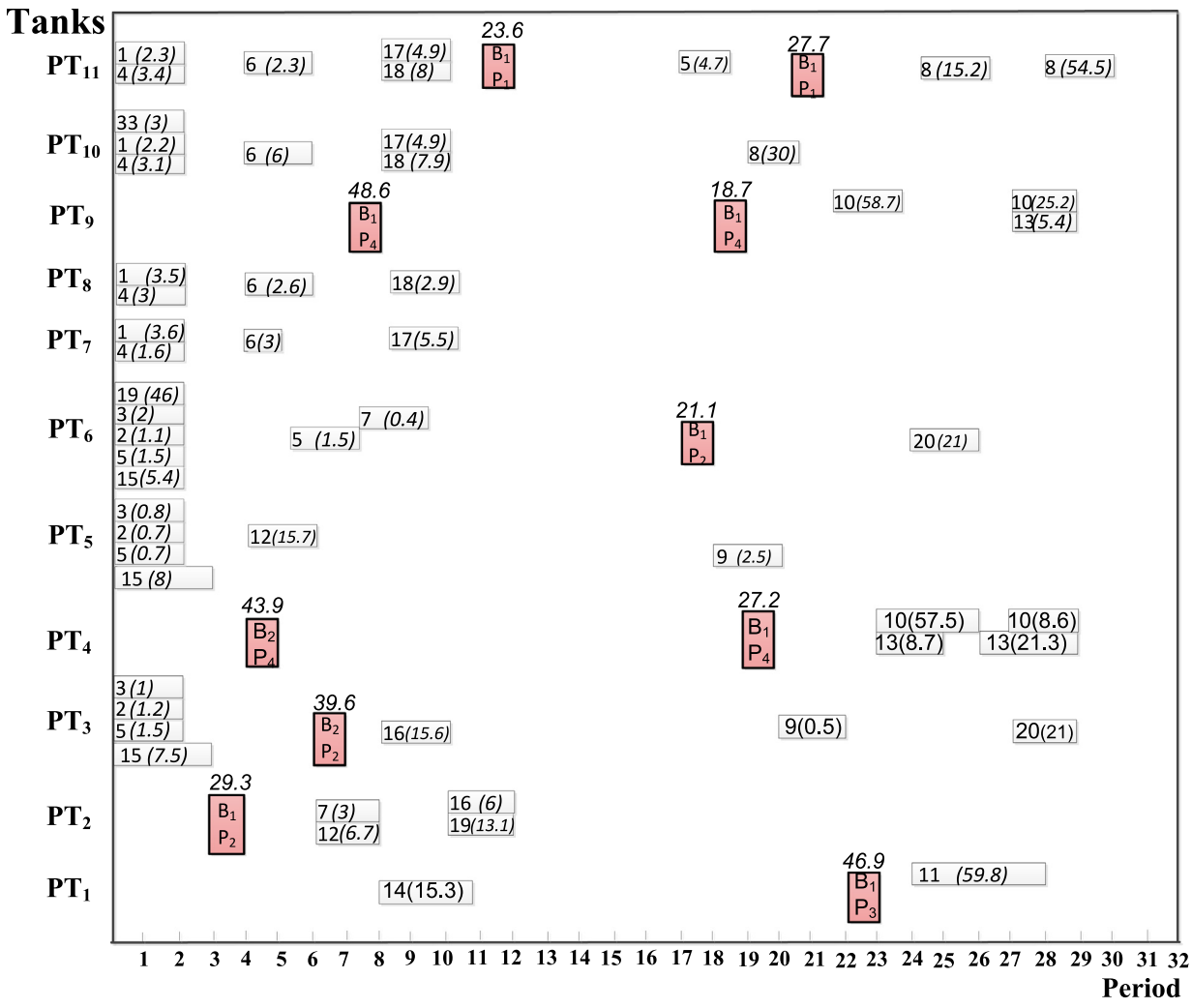

Fig. 9. Schedule for blending and order delivery of the first problem [inside the uncolored boxes, the two numbers represent order and the amount of product delivered to the order, respectively, whereas inside the colored boxes, the characters denote the blender number and the product type and the values above the boxes denote the amount of the product transferred from blender to the product tank]. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

information about both the active edges and their corresponding weight variables at each period are generated. Even though the initialization stage gives a feasible structure in terms of structural constraints, it might fail to satisfy the weight constraints, particularly for non-dedicated units (product tanks). For example, a product tank contains P_1 then it can only cater the product orders corresponding to P_1. However, if the structural schedule has the edges corresponding to P_2 then, these have to be corrected based on the above inventory information. To eliminate such violations, a correction operation is used period-wise. Since decoding is a period-wise process, based on the inventory present in the tanks in the previous period ($(k-1)$th period), the present active edge in the kth period is checked and corrected. The correction is needed only for the edges (incoming and outgoing) linked to product tanks.

Following the decoding and correction operations, the flow values corresponding to the edges present in each chromosome are used to evaluate the objective function(s). Subsequently, the objective functions are penalized using bracketed penalties (Deb et al., 2002) for the weight constraint violations (e.g. property index constraints and product composition constraints) to obtain fitness functions. After, evaluating the fitness functions the crowding distance (CD) and the rank is assigned (Deb et al., 2002) to every chromosome. Next, the tournament selection operation (Deb et al., 2002) is performed where the better chromosomes are copied into a mating pool. This mating pool population then undergo genetic operations.

5.3. Iterative refining of the schedule

GGA has adapted the graphical genetic operations, crossover and mutation (Ramteke and Srinivasan, 2012), particularly for structural sub-chromosome. To illustrate the genetic operations, two feasible chromosomes (F_1 *and* F_2) for the illustrative problem are shown in the Fig. 4. Each chromosome comprises of a structural sub-chromosome and its corresponding weight sub-chromosome. The structural sub-chromosome represents the active edges, e_i (i.e. $i = 1, 23$), of the feasible SGBD network. Likewise, the weight sub-chromosome represents the corresponding weight values of the active edges with w_i (i.e. $i = 1, 23$). The schedule representation for the chromosome F_1 is given Fig. S1 in the supplementary material. The details of graphical genetic operations in structural sub-chromosomes will be discussed below.

In graphical crossover operation (Ramteke and Srinivasan, 2012), two structural sub-chromosomes are selected randomly, and the crossover site is identified in a period-wise manner. At the identified period, their parts are swapped. Subse-

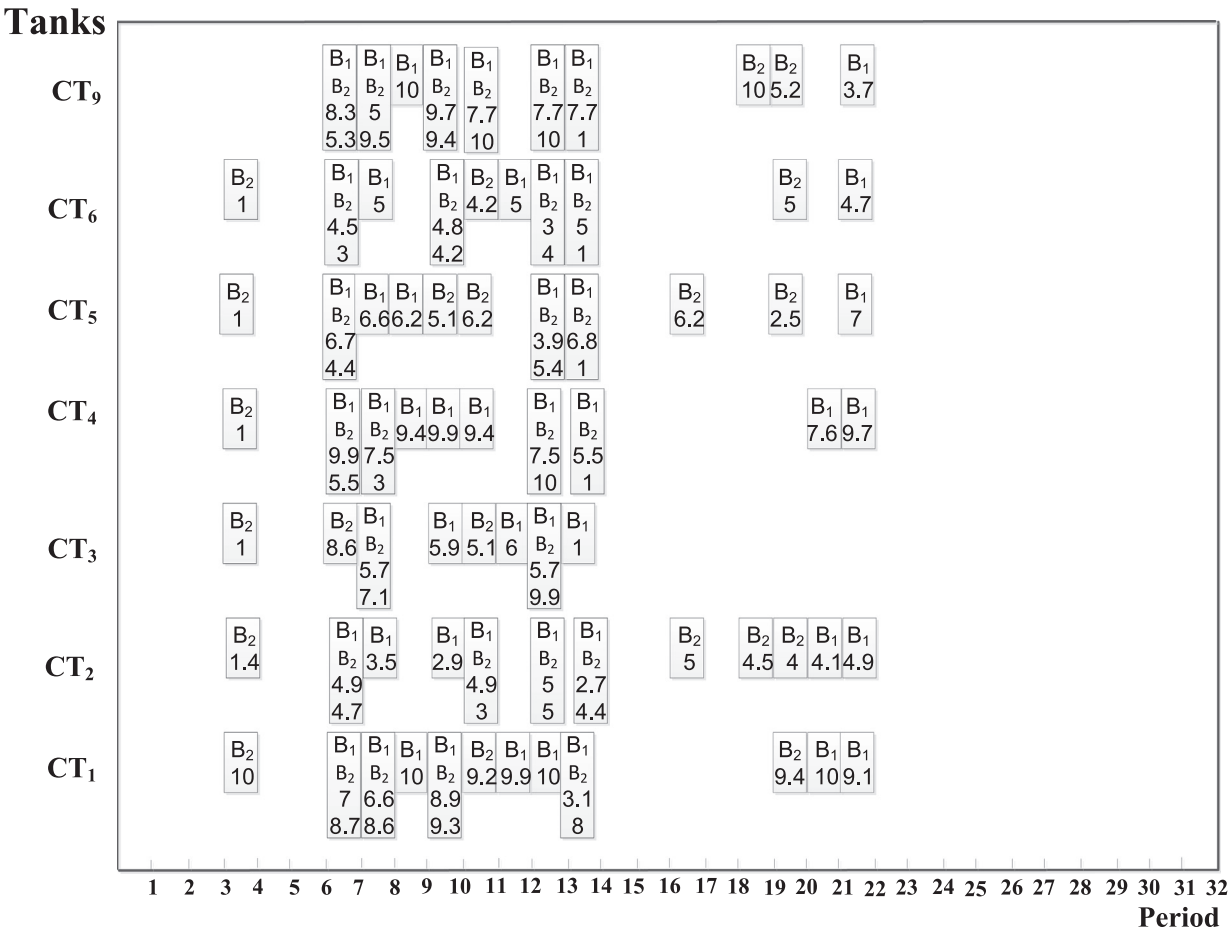

Fig. 10. Feeding schedule of the blenders from the component tanks for the second problem (inside the boxes, the notation at the top denotes the blender number whereas the numerical values at the bottom denote the amount of components transferred from the component tanks to the blender).

quently, a graphical mutation (Ramteke and Srinivasan, 2012) is performed by stochastically adding–for empty edge and deleting or replacing–for an infeasible non-empty edge. Graphical crossover and mutation for the sample problem are shown in Figs. 5 and 6, respectively. As shown in Fig. 5, the graphical crossover-site is at the end of period 2, where their parts are swapped. During mutation, for the first offspring chromosome (D_1), the e_6 on slot 12 gets deleted and the empty edge on slot 16 became active with edge e_2 whereas, for the second chromosome (D_2), the 7th slot is activated with edge e_{12}. The mutated offspring's structural sub-chromosomes (M_1 *and* M_2) are given in Fig. 6.

After a structural mutation, checking for both intra- and inter-period constraints violation is performed. Whenever violation exists, correction is made in both period-and depth-wise manner by deleting, adding or replacing the edges. For each period, correction proceeds from the source to a sink. Correction on lower depth may result in infeasibility on the higher depth vertices. So, depth-wise correction helps to detect the induced infeasibilities and rectify them. However, it may have a change build up– effect on the deeper edges– correcting the lower depth may lead the deeper edges to change more. To reduce this effect, a gradually decreasing mutation probability rate from the source to the sink is used. In the above illustrative example, after crossover, the second offspring violates the operating rule 4 as the order O_1 is not delivered during the time schedule. Similarly, during graphical mutation for the second offspring, M_2, deletion of e_6 violates the operating rule 4 as O_4 will not be delivered and addition of e_{12} violates the constraint 3 as it leads to simultaneous product inflow and outflow from a product tank. The structural sub-chromosomes after the correction operation are depicted in Fig. 7. The edge e_5 on the fourth slot of the second mutated offspring, M_2, gets deleted and the edge e_6 becomes activated on the 20th slot in the first offspring, M_1 to produce the structurally feasible offspring sub-chromosomes C_2 and C_1, respectively.

In weight sub-chromosome, the genetic operations are identical to conventional GA except for crossover site selection. The crossover site for weight sub-chromosomes is selected to be the same as that used in corresponding graphical crossover. Unlike structural sub-chromosome, genetic operation in weight sub-chromosome handles most of the constraint violations using the dynamic bounds. The dynamic bound generation ensures the fulfilment of the univariate (only one active edge entering or leaving a vertex) weight constraints. Further, for the multivariate constraints such as a unit receiving from and/or delivering to multiple units, static bounds such as capacity constraints are handled using penalty function approach.

The population of offspring chromosomes obtained after the genetic operations undergo the elitism operation (Deb et al., 2002). In this operation, the offspring chromosomes are mixed with the parent chromosomes and the best N_P chromosomes from this pool of $2N_P$ chromosomes are selected using the nondominated ranking and crowding distance procedure. These elite N_P chromosomes become parents for the next generation and iteration continuous repeating all the steps until the user-specified criterion becomes fulfiled (i.e. the counter for generations, N_{gen} = maximum number of user-specified generations, N_{maxgen}).

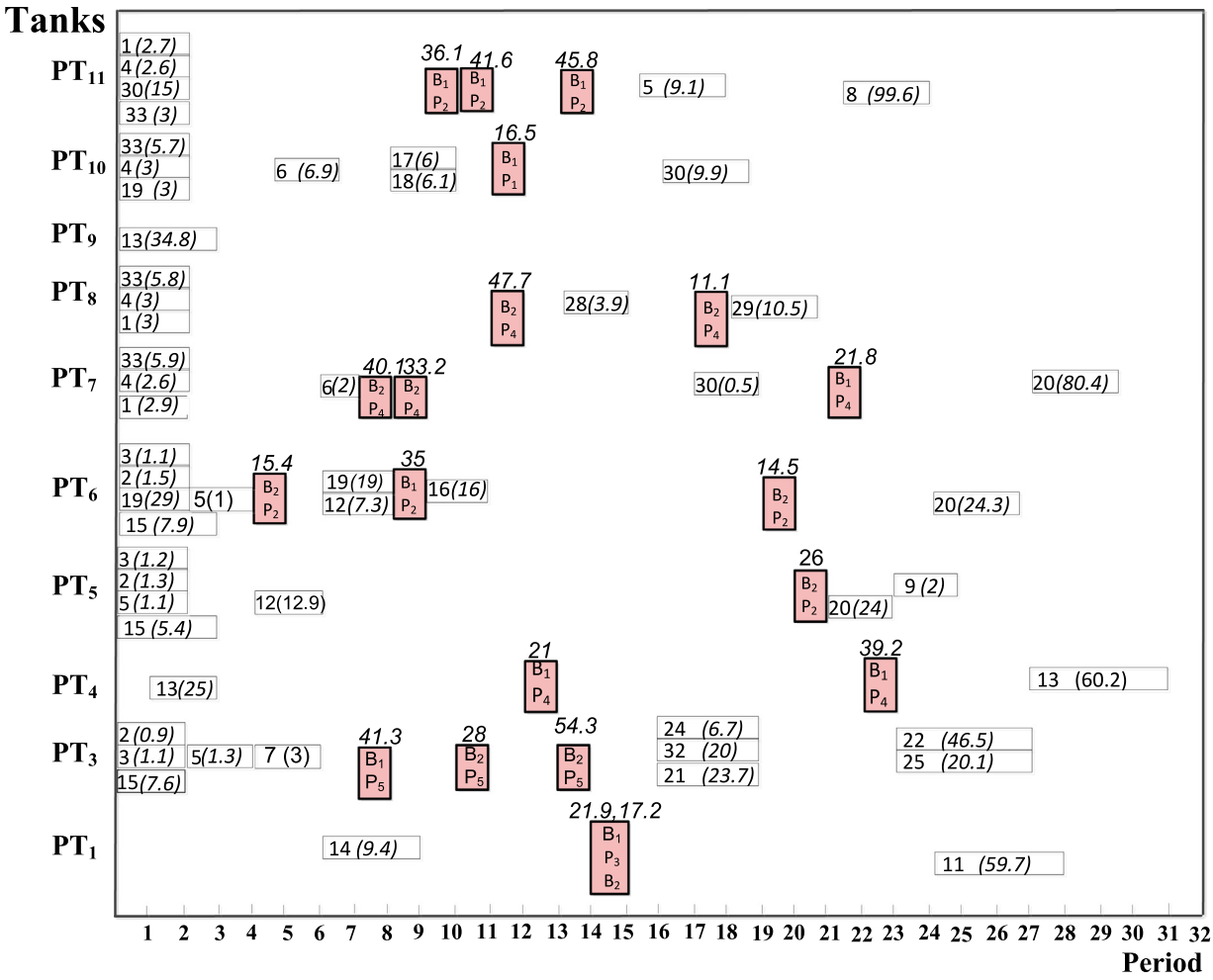

Fig. 11. Schedule for blending and orders delivery for the second problem [inside the uncolored boxes, the two numbers represent order and the amount of product delivered to the order, respectively, whereas inside the colored boxes, the characters denote the blender number and the product type and the values above the boxes denote the amount of the product transferred from blender to the product tank]. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6. Results and discussion

A GGA is executed in Fortran to solve three benchmark SGBD problems taken from (Cerdá et al., 2016). The problems are different in the number of blenders and orders, the demands and due date of orders and by the required product types. The first two problems have two blenders while the third one has three. Problems one to three involves three, four and five product types and 20, 35 and 45 orders, respectively. Each order has its own quantity and due date which can be different for each problem. The first two problems involve the processing of four types of product (P_1-P_4) while the third problem involves the processing of five product types (P_1-P_5). Apart from their differences, the three problems have the following common features. They receive the component from nine component tanks and the products from blender are temporarily stored in 11 product tanks. Moreover, the scheduling horizon for each problem is 192 h and the quality of the gasoline is checked using nine property indices. The complete details (Cerdá et al., 2016b) of the problems are given in Tables S1–S5 of the supplementary material. These problems are solved on a desktop computer (3.60 GHz processor, 32.00GB RAM and 64-bit operating system). Also, the GA parameters used in this study are given in Table S6 of the supplementary material. The solution statistics and Gantt charts are presented in Table 6 and Figs. 8–15. The Gantt charts are given in the period base, which is equivalent to 6 h duration and the amount of material (components or products) transferred or blended or stored are given in kilo barrels (*kbbl*).

6.1. Single objective optimization (SOO)

For the first problem, the SOO schedules are given in Figs. 8 and 9. Fig. 8 represents the amount of component transferred from the component tanks (CT_1–CT_9) to the blenders (B_1–B_2) at each period. For example, in period 4, the first blender (B_1) has received 6 and 3.5 kbbl of components from CT_1 and CT_5, respectively. Likewise, the blenders feed at each period from every component tank can be identified. Fig. 9 shows the amount and types of product the product tanks (PT_1-PT_{11}) will receive from blender (inside the colored box) in each period. In addition, it also provides the type and the amount of product delivered to the orders as given in the uncolored box. For instance, as shown in the colored box, the product tank PT_4 will be receiving 24.7 and 30 kbbl of the product P_4 at period 17, from the blender B_1 and then from the blender B_2, respectively. Similarly, as shown in the uncolored box, the same tank will deliver 41 kbbl of P_4 to the order O_{10} during 23–24. It will also deliver 39 kbbl of the P_4 to the order O_{13} during the period of 24–25. Following the SOO Gantt charts for the first problem, Figs. 10–11 provide the SOO schedule for the second problem and Figs. 12 and 13 provide the SOO schedule for the second problem. In Figs. 8–13, the boxes give the information about the active

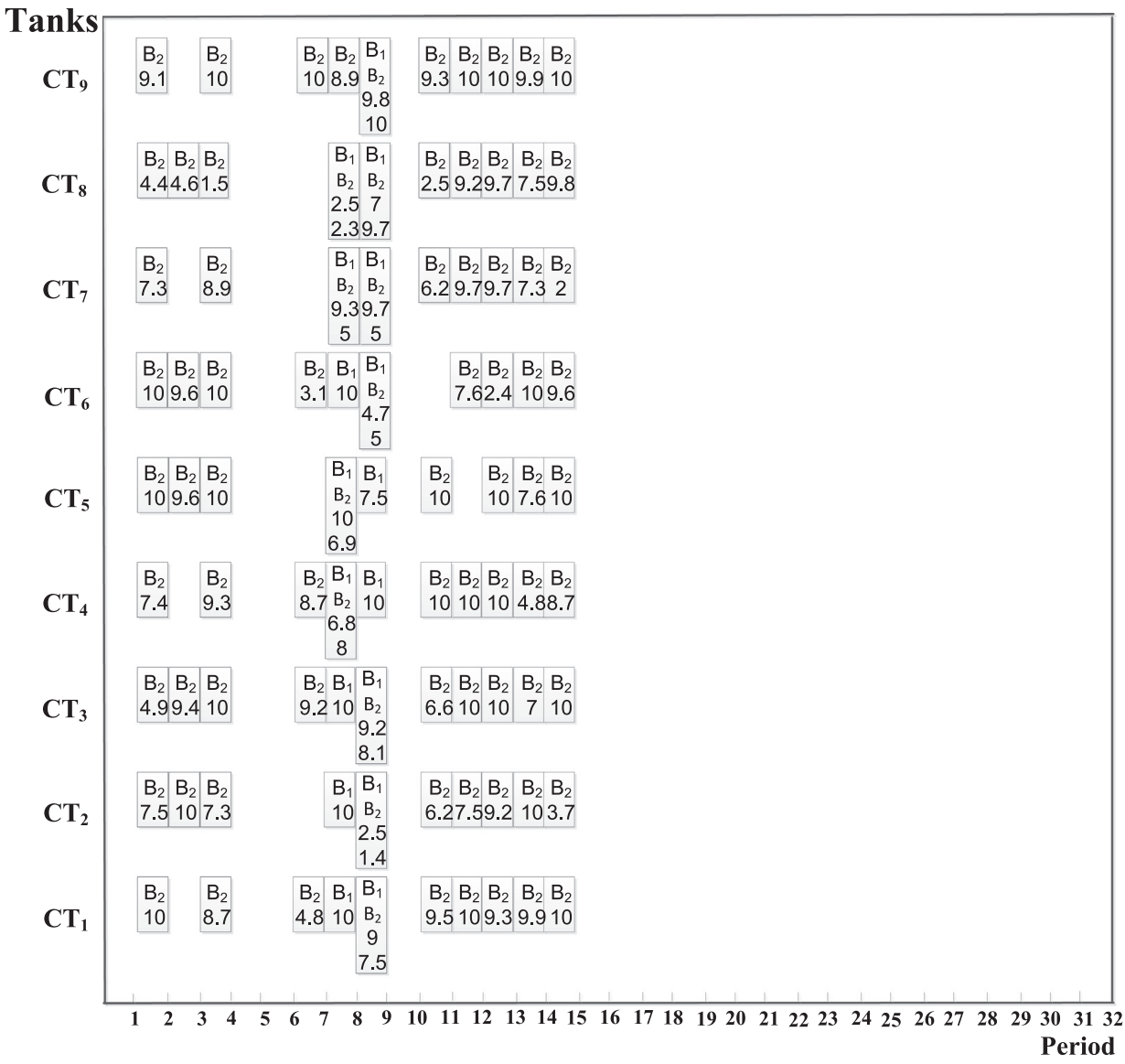

Fig. 12. Feeding schedule of the blenders from the component tanks for the third problem (inside the boxes, the notation at the top denotes the blender number whereas the numerical values at the bottom denote the number of components transferred from the component tanks to the blender).

edges and the corresponding weights similar to that represented in Fig. 4. For example in Fig. 8, the box corresponding to CT_1 for period 1 represents the edges from CT_1 to B_1 and B_2 with corresponding weights of 6 and 2.8 kbbl, respectively.

The SOO Gantt charts shown in Figs. 8–13 illustrates that the increase in problem size increases the number of material flow (edges) during each period. Hence, at a time a product tank will deliver to a greater number of orders for a larger problem size compared to the smaller one. Therefore, on top of the problem size increments, the combinatorial complexity increases which make solving the bigger size problem in conventional GA and mathematical programming difficult. On the other hand, none of the schedules has product transition in the product tanks. This enables the cost for product changeover in the product tanks to be null which leads to operating cost reduction.

The statistical comparison of the proposed approach with the MINLP approach of Cerdá et al., 2016a) is given in Table 1. Cerdá et al. (2016b) have used Core i7 3632QM 2.20 GHz one-processor PC with 12GB RAM and four cores, whereas the developed formulation is solved using 3.60 GHz processor, 32.00GB RAM and 64-bit operating system. The result shows that the proposed approach gives the savings of 80,200, 170,340 and 1,257,380 dollars for problems 1 to 3, respectively over the operating cost reported in Cerdá et al., 2016a). It is to be noted that the proposed approach used the exact formulation without any relaxation whereas Cerdá et al. (2016b) relaxed the MINLP formulation to NLP by fixing the integer variables to their MILP optimal. This could be the reason for a giveaway from the actual optimal value. The results are also compared with those reported by Castillo et al. (2017). However, the formulation used by Castillo et al. (2017) is slightly simplified as it involved just two property indices compared to nine used in the present study. Further, their formulation involved the relaxation of bilinear term in the model. The comparison shows the lower cost using GGA for all three problems. This could be attributed to relaxation and simplification used by Castillo et al. (2017). Further, the proposed approach gives a higher operating cost reduction for the bigger size problems which assures the applicability of the proposed model for large size problems. At the same time, it is shown that the proposed approach has a significant model size reduction compared to the conventional GA and MINLP model. The model

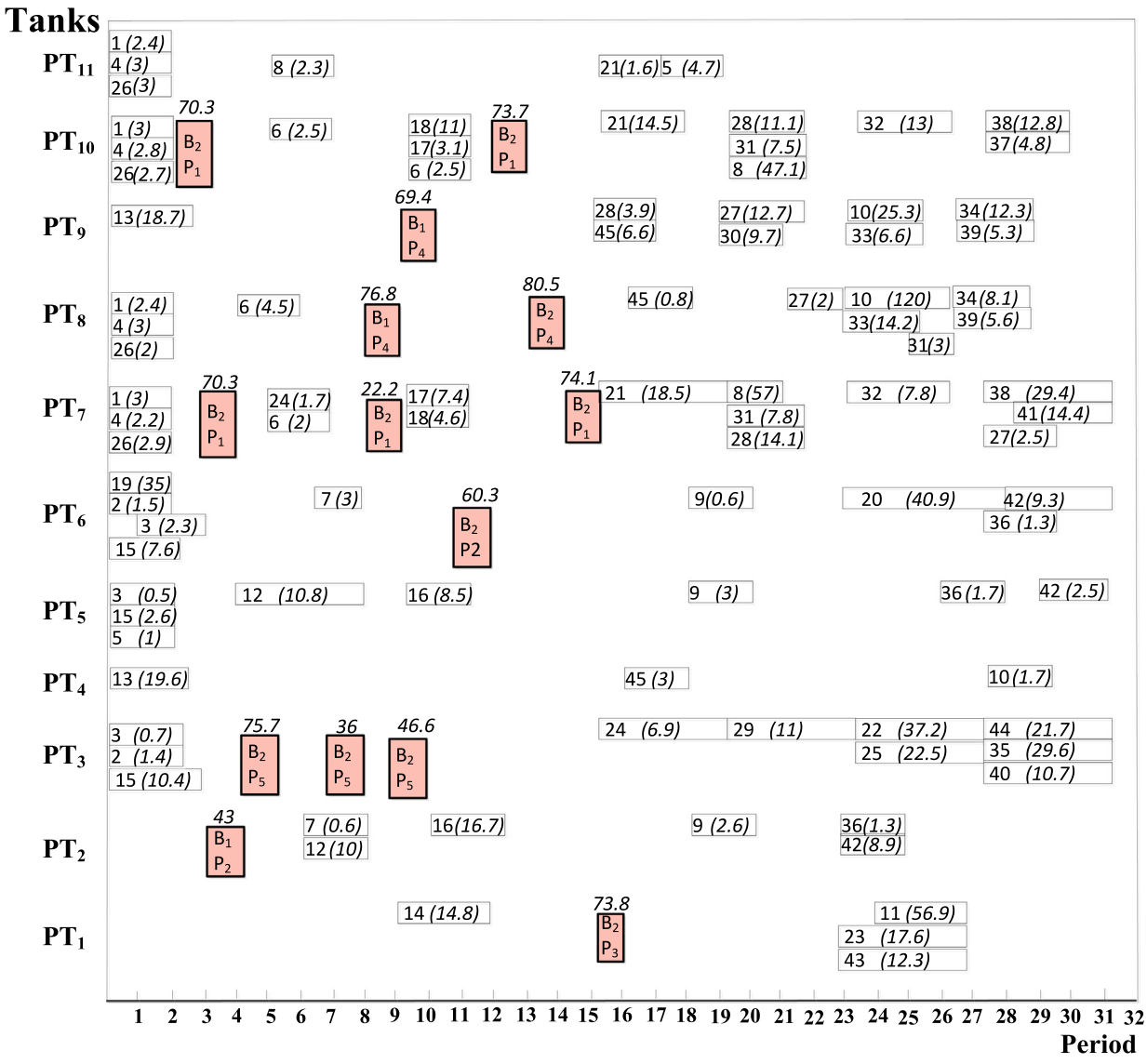

Fig. 13. Schedule for blending and orders delivery for the third [inside the uncolored boxes, the two numbers represent order and the amount of product delivered to the order, respectively, whereas inside the colored boxes, the characters denote the blender number and the product type and the values above the boxes denote the amount of the product transferred from blender to the product tank]. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Model statistics.

Problem	Number of variables				Number of			Costs (k$)			CPU (s)#	
	Cerdá et al. (2016)	ConventionalGA (CGA)	GGA	% Reduction for GGA	Constraints for Cerdá et al., 2016a)	Penalty function GGA	% Reduction	Cerdá et al. (2016b)	*Castillo et al. (2017)	GGA	GGA	Cerdá et al., 2016)
1	31,552	31,552	6784	78.5	2143	215	90	8080.35	8 206	8000.15	15.7	2.7
2	65,696	65,696	7808	88.1	7231	230	96.8	15,221.74	15 384	15,051.40	32.3	23.0
3	85,344	85,344	11,008	87.1	11,541	309	97.3	21,101.43	21 270	19,844.05	39.4	134.0

* Castillo et al. (2017) used only two property indices in their formulation compared to nine used in the present work and Cerdá et al., 2016). Therefore, the number of variables and number of constraints are not compared.

Cerdá et al. (2016b) have used Core i7 3632QM 2.20 GHz one-processor PC with 12GB RAM and four cores, whereas 3.60 GHz processor, 32.00GB RAM and 64-bit operating system is used in the current study.

size reduction is higher for the bigger size problems and has a phenomenal significance in computational efficiency. Also, the trend of CPU time is polynomial in nature with increase in problem size. Further, the effect of randomness of stochastic approach on the stability of gasoline operation is checked by performing three random runs using different random seeds for all three problems. The results are given in Table. S7 (supplementary material) which shows that the optimal values obtained are nearly the same for different random runs. Also, the corresponding schedules are compared in Figs. S4–S9 of the supplementary material for each problem. The comparison shows that the schedules are structurally the same (i.e. the sequences of blender feeding, product tank feeding, and or-

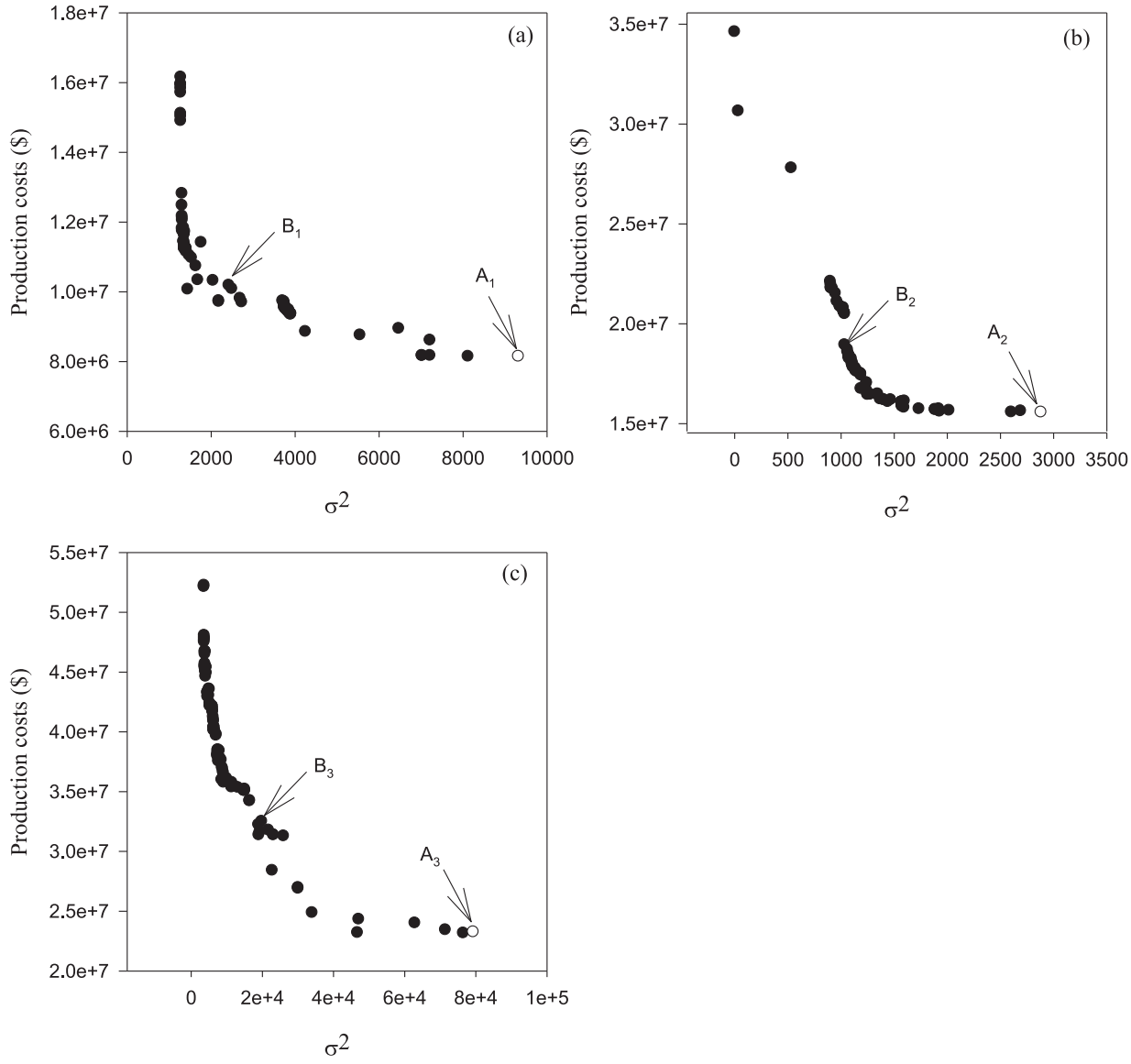

Fig. 14. Pareto front for the three problems 1–3 (B_1, B_2 and B_3 are the selected operating points on the Pareto optimal fronts of problems 1 −3 whereas A_1, A_2 and A_3 are the corresponding optimal solutions of SOO) .

der delivery are the same) for all random runs. Further, the corresponding flow rates are the same except for one or two places that validate the stability of the formulation. It is to be noted that such small fluctuations around the optimal solution with change in the random seed number are common for stochastic algorithms such as GA as these converge asymptotically. However, the deterministic algorithms are immune to such fluctuations as these do not require a random seed number.

6.2. Multi-objective optimization (MOO)

In gasoline blending, maintaining the consistency of the product quality is a very crucial aspect. Maintaining the product consistency not only enables the reduction of the high cost incurred due to quality giveaway but also increases the operating ability and controllability of the process. During blending, the main drive is to maximize the low-cost components in a way quality giveaway is reduced. Many of the gasoline index properties are non-linear, so estimating the final blend properties from fluctuating component feed flow rates become quite complex, especially for commercially available linearization-based programming models. To overcome this challenge, minimizing the fluctuation in blending flow rate becomes the best alternative. It is to be noted that, the second objective can be combined into the first objective using the cost factor to give single-objective formulation. However, the cost factors for blending rate fluctuation are usually planted specific. In contrast with this, the multi-objective formulation represents the generalized case in which the optimization gives the operator multiple operational options for the operating point. The operator can select a suitable operating point from the set of optimal solutions according to the plant requirement.

For the first problem, the Pareto optimal plot for MOO is given in Fig. 14(a). It gives relative trade-off of production cost against the sum of the square of the blending rate fluctuation in the consecutive periods (δ^2). Depending on the subjective preference or other decision philosophies, a decision-maker can find the best-preferred Pareto optimal solution out of the non-dominated Pareto optimal solutions. The optimum point for a single objective optimization (point A_1) is given in the figure for reference, which represents the corner solution of the Pareto front. The decision-maker

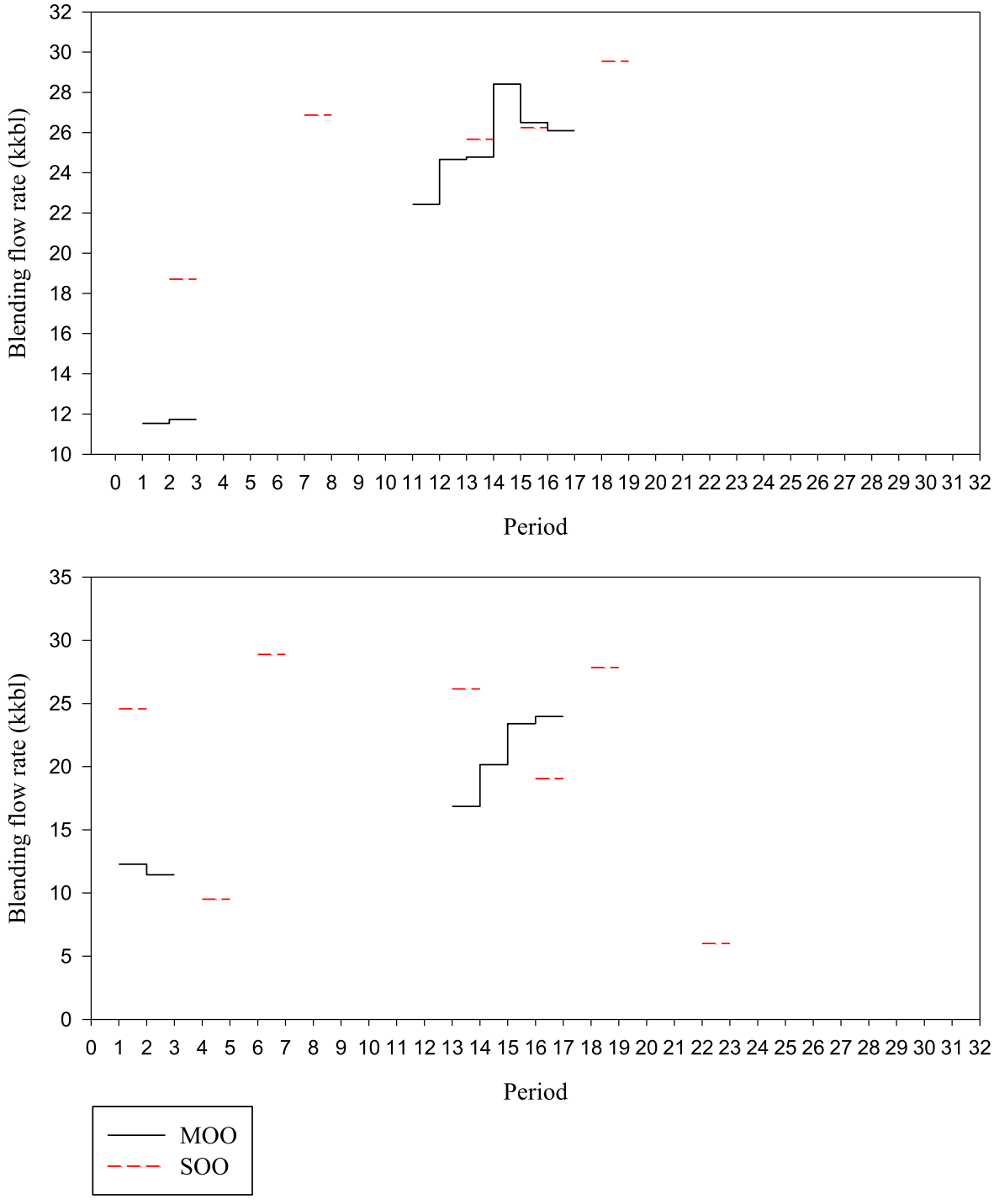

Fig. 15. Comparison of blending rate of (a) blender 1 (b) blender 2 for single and multiple objectives for problem 1 corresponding to points A_1 and B_1 shown in Fig. 14(a).

can pick an optimal solution out of the set of equally good solutions represented in a Fig. 14(a). Similarity, the Pareto optimal plots for MOO of second and third problems are shown in Fig. 14(b) and (c). Also, the corresponding single objective optimization solutions (points A_2 *and* A_3) are identified on the Pareto optimal plots for reference.

The blending rate comparison between single objective solutions (points A_1) and a multi-objective solution picked somewhat at the middle (points B_1) of problem 1 is given in Fig. 15. It shows that the blending rate fluctuation is high for SOO compared to MOO. In the case of SOO, the operation of the blenders is not continuous for more than one period, which leads to high blending rate fluctuations. However, for MOO, once the blender starts, it works continuously at least for two periods and the blending flow rate changes slightly in the consecutive periods. The maximum fluctuation for MOO is for blender 2, which is less than 25%. Therefore, the MOO has significantly improved consistency of blending flowrate which assures a stable process operation. A similar comparison of blending rates for problems 2 and 3 are given in Figs. S2 and S3 of the supplementary material.

7. Conclusion

In this work, a GGA is developed for SGBD. Three industrial problems are solved using the proposed model for both single and multi-objective formulations. Minimizing the production cost is the first objective for SOO, whereas minimizing the sum of the square of fluctuation in inter-period blending rate is the additional objective for MOO.

For the SOO formulation, the proposed model gives about 6% production cost reduction compared to the MINLP result given in the literature Cerdá et al. (2016a). The production cost reduction increases as the problem size increases, which assures the applicability of the proposed model for big size problems. At the same time, it is shown that the proposed approach has more than 80% model size reduction compared to the conventional GA.

In the MOO formulation, to smooth the blending process and to improves the controllability, minimizing the sum of the square of fluctuation in inter-period blending rate is added as the second objective. It has reduced the flow rate fluctuation which goes up to 100% for SOO to a maximum of 25% for MOO. Moreover, the Pareto front which gives the trade-off between production cost and blending rate fluctuation helps the operator to choose the operating condition from the set of non-dominated optimal solutions based on subjective preference information.

Declaration of Competing Interest

The authors do not have any conflicts of interest to declare.

Acknowledgement

We would like to dedicate this paper to Prof. Roger Sargent.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.compchemeng.2019.106636.

References

Ahsan, M., 2015. Prediction of gasoline yield in a fluid catalytic cracking (FCC) riser using k-epsilon turbulence and 4-lump kinetic models : a computational fluid dynamics (CFD) approach. J. King Saud Univ. Eng. Sci. 27, 130–136.

Oliveira, F., Hamacher, S., Almeida, M.R., 2011. Process industry scheduling optimization using genetic algorithm and mathematical programming. J. Intell. Manuf. 22 801–813.

Bornapour, M., Hooshmand, R., Khodabakhshian, A., Parastegari, M., 2017. Optimal stochastic scheduling of CHP-PEMFC, WT, PV units and hydrogen storage in reconfigurable micro grids considering reliability enhancement. Energy Convers. Manag. 150, 725–741.

Carlos, A.M., Grossmann, I.E., Harjunkoski, I., Kabor, P., 2006. A simultaneous optimization approach for off-line blending and scheduling of oil-refinery operations. Comput. Chem. Eng. 30, 614–634.

Castillo, P.A.C., Castro, P.M., Mahalec, V., 2017. Global optimization of nonlinear blend-scheduling problems. Engineering 3, 188–201.

Cerdá, J., Pautasso, P.C., Cafaro, D.C., 2016b. Optimizing gasoline recipes and blending operations using nonlinear blend models. Ind. Eng. Chem. Res. 55, 7782–7800.

Cerdá, J., Pautasso, P.C., Cafaro, D.C., Conicet, I.U.N.L., 2016a. A cost-effective model for the gasoline blend optimization problem. AIChE J. 62, 3002–3019.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE 6, 182–197.

Gujarathi, A., Babu, B.V., 2016. Evolutionary Computation: Techniques and Applications, first ed. CRC press.

Glismann, K., Gruhn, G., 2000. Short-term scheduling and recipe optimization of blending processes. Comput. Chem. Eng. 8, 1099–1104.

Hou, Y., Wu, N., Member, S., Zhou, M., Li, Z., 2017. Pareto-optimization for scheduling of crude oil operations in refinery via genetic algorithm. IEEE 47, 517–530.

Ivanov, S.Y., Ray, A.K., 2014. Multiobjective optimization of industrial petroleum processing units using genetic algorithms. In: Procedia Chem.. Elsevier Ltd, pp. 7–14.

Jia, Z., Ierapetritou, M., 2003. Mixed-integer linear programming model for gasoline blending and distribution scheduling. Ind. Eng. Chem. Res. 42, 825–835.

Jia, Z., Ierapetritou, M., Kelly, J.D., 2003. Refinery short-term scheduling using continuous time formulation: crude-oil operations. Ind. Eng. Chem. Res. 42, 3085–3097.

Kasat, R.B., Gupta, S.K., 2003. Multi-objective optimization of an industrial fluidized-bed catalytic cracking unit (FCCU) using genetic algorithm (GA) with the jumping genes operator. Comput. Chem. Eng. 27, 1785–1800.

Khosla, D.K., Gupta, S.K., Saraf, D.N., 2007. Multi-objective optimization of fuel oil blending using the jumping gene adaptation of genetic algorithm. Fuel Process. Technol. 88, 51–63.

Khosravi, R., Khosravi, A., Nahavandi, S., Hajabdollahi, H., 2015. Effectiveness of evolutionary algorithms for optimization of heat exchangers. Energy Convers. Manag. 89, 281–288.

Kumar Koratiya, V., Sunil, K., Sinha, S., 2010. Modeling, simulation and optimization of FCC dower reactor. Petrol. Coal 52.

Li, J., Karimi, I.A., 2011. Scheduling gasoline blending operations from recipe determination to shipping using unit slots. Ind. Eng. Chem. Res. 50, 9156–9174.

Li, J., Karimi, I.A., Srinivasan, R., 2010. Recipe determination and scheduling of gasoline blending operations. AIChE J. 56, 441–465.

List of Registered Gasoline Additives [WWW Document] 2019. https://www.iea.org/weo/.

Simao, L.M., Dias, D., Pacheco, M., 2007. Refinery scheduling optimization using genetic algorithms and cooperative coevolution. In: 2007 IEEE Symposium on Computational Intelligence in Scheduling, pp. 151–158. doi:10.1109/SCIS.2007.367683.

Panda, D., Ramteke, M., 2019. Preventive crude oil scheduling under demand uncertainty using structure adapted genetic algorithm. Appl. Energy 235, 68–82.

Panda, D., Ramteke, M., 2018. Reactive scheduling of crude oil using structure adapted genetic algorithm under multiple uncertainties. Comput. Chem. Eng. 116, 333–351.

Ramteke, M., Srinivasan, R., 2012. Large-scale refinery crude oil scheduling by integrating graph representation and genetic algorithm. Ind. Eng. Chem. Res. 51, 5256–5272.

Rangaiah, G.P., 2009. Multi-Objective Optimization: Techniques and Applications in Chemical Engineering. World Scientific..

Rocha, E.W., Huet, P., Mohatarem, G.M., 2014. The 2040 economy: long-term growth determinants. Global Economics & Country Risk Conference.

Sarkar, D., Modak, J.M., 2005. Pareto-optimal solutions for multi-objective optimization of fed-batch bioreactors using nondominated sorting genetic algorithm. Chem. Sci. Eng. 60, 481–492.

World Energy Outlook 2018 [WWW Document] 2018. https://www.iea.org/weo/. URL https://www.iea.org/weo2018/fuels/ (accessed 5.29.19).