Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 748141, 11 pages
http://dx.doi.org/10.1155/2014/748141

Research Article

A Novel Algorithm Combining Finite State Method and Genetic Algorithm for Solving Crude Oil Scheduling Problem

Qian-Qian Duan, Gen-Ke Yang, and Chang-Chun Pan

Department of Automation and Key Laboratory of System Control and Information Processing, Shanghai Jiao Tong University, Ministry of Education of China, Shanghai 200240, China

Correspondence should be addressed to Gen-Ke Yang; sjtu1019@163.com and Chang-Chun Pan; 390635304@qq.com

Received 4 October 2013; Accepted 4 December 2013; Published 18 February 2014

Academic Editors: Q. Cheng and J. Yang

A hybrid optimization algorithm combining finite state method (FSM) and genetic algorithm (GA) is proposed to solve the crude oil scheduling problem. The FSM and GA are combined to take the advantage of each method and compensate deficiencies of individual methods. In the proposed algorithm, the finite state method makes up for the weakness of GA which is poor at local searching ability. The heuristic returned by the FSM can guide the GA algorithm towards good solutions. The idea behind this is that we can generate promising substructure or partial solution by using FSM. Furthermore, the FSM can guarantee that the entire solution space is uniformly covered. Therefore, the combination of the two algorithms has better global performance than the existing GA or FSM which is operated individually. Finally, a real-life crude oil scheduling problem from the literature is used for conducting simulation. The experimental results validate that the proposed method outperforms the state-of-art GA method.

1. Introduction

In recent years refineries have to explore all potential cost-saving strategies due to intense competition arising from fluctuating product demands and ever-changing crude prices. Scheduling of crude oil operations is a critical task in the overall refinery operations [1–3]. Basically, the optimization of crude oil scheduling operations consists of three parts [4]. The first part involves the crude oil unloading, mixing, transferring, and multilevel crude oil inventory control process. The second part deals with fractionation, reaction scheduling, and a variety of intermediate product tanks control. The third part involves the finished product blending and distributing process. In this paper, we focus on the first part, as it is a critical component for refinery scheduling operations. Scheduling of crude oil problem is often formulated as mixed integer nonlinear programming (MINLP) models [2, 5, 6]. The solution approaches for solving MINLP can be roughly divided into two categories [7]: deterministic approaches and stochastic approaches. Some deterministic methods have been available for many years [8]. These methods require the prior step of identification and elimination of nonconvexity and decompose the MINLP models into relevant nonlinear programming (NLP) and mixed integer linear programming (MILP) and then these subproblems have to be iteratively solved. The most common algorithms are branch and bound [9], outer-approximation [10], generalized benders decomposition [11], and so forth. Also, some commercial MINLP solvers have been developed for solving the problem at hand optimally [12]. However, the commercial solver can only handle MINLPs with special properties. The other stream of global optimization is the stochastic algorithms, for example, simulated annealing (SA), GA, and their variants [7]. GA proposed by Holland [13], because of their simple concept, easy scheme, and the global search capability independent of gradient information, have been developed rapidly. Much other attention is given to the development of GA for MINLP. For instance, Yokota et al. developed a penalty function that is suitable for solving MINLP problems [14]. Costa and Oliveira also implemented another type of penalty function to solve various MINLP problems, including industrial-scale problems [15]. They also noted that the evolutionary approach is efficient, in terms of the number of function evaluations, and is very suitable to handle the difficulties of the nonconvexity. Going one step further, some mixed coding methods were proposed, which include mixed-coding genetic algorithm [15] and information-guided genetic algorithm (IGA). Ponce-Ortega et al. [16] proposed a two-level approach based on GA to optimize the heat exchanger networks (HENs). The outer level is used to perform the structural optimization, for which a binary GA is used. Björk and Nordman [17] showed that the GA is very suitable to solve a large-scale heat exchanger network.

Obviously, the two different approaches previously discussed have their own advantages and disadvantages. On the one hand, a deterministic approach usually involves considerable algebra and undeviating analysis to the problem itself, whereas the evolutionary approach does not have this property. On the other hand, some deterministic approaches, such as mathematical programming, usually cannot provide practical solutions in reasonable time, whereas the evolutionary approach can generate satisfying solutions. In this work, a novel genetic algorithm which combined the finite state method and GA is proposed to solve crude oil scheduling problem. A MINLP model is formulated based on the single-operation sequencing (SOS) time representation. A deterministic finite automation (DFA) model which captures valid possible schedule sequences is constructed based on the sequencing rules. The initialization and mutation operation of GA is based on the model which builds legal schedules complying with sequencing rules and operation condition. Thus, the search space of the algorithm is substantially reduced as only legal sequence is explored. The rest of the paper is organized as follows: the MINLP model is specified in Section 2. Section 3 reviews the background of finite state theory. In Section 4, a novel genetic algorithm which combined the finite state method and GA is proposed to solve the MINLP model. A test problem is studied to verify our approach in Section 5. In the last section, conclusive remarks are given.

2. Mathematic Model

In this section, the MINLP model of refinery crude oil scheduling problem is described [18]. This problem has been widely studied from the optimization viewpoint since the work of Lee et al. [19]. It consists of crude oil unloading from marine vessels to storage tanks, transfer and blending between tanks, and distillation of crude mixtures. The goal is to maximize profit and meet distillation demands for each type of crude blend (e.g., low sulfur or high sulfur blends), while satisfying unloading and transfer logistics constraints, inventory capacity limitations, and property specifications for each blend. The logistics constraints involve nonoverlapping constraints between crude oil transfer operations.

2.1. Sets. The following sets will be used in the model.

(i) $T = \{1, \dots, n\}$ is the set of priority-slots;

(ii) W is the set of all operations: $W \triangleq W_U \cup W_T \cup W_D$;

(iii) $W_U \subset W$ is the set of unloading operations;

(iv) $W_T \subset W$ is the set of tank-to-tank transfer operations;

(v) $W_D \subset W$ is the set of distillation operations;

(vi) R is the set of all operations: $R = R_V \cup R_S \cup R_C \cup R_D$;

(vii) $R_V \subset R$ is the set of vessels;

(viii) $R_S \subset R$ is the set of storage tanks;

(ix) $R_C \subset R$ is the set of charging tanks;

(x) $R_D \subset R$ is the set of distillation units;

(xi) $I_r \subset W$ is the set of inlet transfer operations on resource r;

(xii) $O_r \subset W$ is the set of outlet transfer operations on resource r;

(xiii) C is the set of products (i.e., crudes);

(xiv) K is the set of product properties (e.g., crude sulfur concentration).

2.2. Parameters. Parameters used in the paper are defined below:

(i) H is the scheduling horizon;

(ii) $[\underline{V_v^t}, \overline{V_v^t}]$ are bounds on the total volume transferred during transfer operation V; in all instances, $\underline{V_v^t} = 0$ for all operations except unloading for which $\underline{V_v^t} = \overline{V_v^t}$ is the volume of crude in the marine vessel;

(iii) $[\underline{N_D}, \overline{N_D}]$ are the bounds on the number of distillations;

(iv) $[\underline{FR_v}, \overline{FR_v}]$ are flow rate limitations for transfer operation v;

(v) S_r is the arrival time of vessel r;

(vi) $[\underline{x_{vk}}, \overline{x_{vk}}]$ are the limits of property k of the blended products transferred during operation v;

(vii) x_{ck} is the value of the property k of crude c;

(viii) $[\underline{L_r^t}, \overline{L_r^t}]$ are the capacity limits of tank r;

(ix) $[\underline{D_r}, \overline{D_r}]$ are the bounds of the demand on products to be transferred out of the charging tank r during the scheduling horizon;

(x) G_c is the gross margin of crude c.

2.3. Variables

2.3.1. Assignment Variables

$$Z_{iv} \in \{0, 1\}, \quad i \in T, \ v \in W. \tag{1}$$

$Z_{iv} = 1$ if operation v is assigned to priority-slot i; $Z_{iv} = 0$ otherwise.

2.3.2. Time Variables

$$S_{iv} \geq 0, \quad D_{iv} \geq 0, \quad i \in T, \ v \in W. \tag{2}$$

S_{iv} is the start time of operation v if it is assigned to priority slot i; $S_{iv} = 0$ otherwise.

D_{iv} is the duration of operation v if it is assigned to priority slot i; $D_{iv} = 0$ otherwise.

2.3.3. Operation Variables

$$V_{iv}^{t} \geq 0, \quad V_{ivc} \geq 0, \quad i \in T, \ v \in W, \ c \in C. \tag{3}$$

V_{iv}^{t} is the total volume of crude transferred during operation v if it is assigned to priority slot i; $V_{iv}^{t} = 0$ otherwise.

V_{ivc} is the volume of crude c transferred during operation v if it is assigned to priority slot i; $V_{ivc} = 0$ otherwise.

2.3.4. Resource Variables

$$L_{ir}^{t}, L_{irc}, \quad i \in T, \ r \in R, \ c \in C. \tag{4}$$

L_{ir}^{t} is the total accumulated level of crude in tank $r \in R_S \cup R_C$ before the operation was assigned to priority-slot i.

L_{irc} is the accumulated level of crude c in tank $r \in R_S \cup R_C$ before the operation was assigned to priority-slot i.

2.4. Objective Function. The objective is to maximize the gross margins of the distilled crude blends. Let G_c be the individual gross margin of crude c,

$$\max \sum_{i \in T} \sum_{r \in R_D} \sum_{v \in I_r} \sum_{c \in C} G_c \cdot V_{ivc}. \tag{5}$$

2.5. General Constraints. It should be noted that the crude composition of blends in tanks is tracked instead of their properties. The distillation specifications are later enforced by calculating a posteriori the properties of the blend in terms of its composition. For instance, in the problem, a blend composed of 50% of crude A and 50% of crude B has a sulfur concentration of 0.035 which does not meet the specification for crude mix X nor for crude mix Y.

2.5.1. Assignment Constraints. In the SOS model, exactly one operation has to be assigned to each priority slot,

$$\sum_{v \in W} Z_{iv} = 1, \quad i \in T. \tag{6}$$

2.5.2. Variable Constraints. Variable constraints are given by their definitions. Start time, duration, and global volume variables are defined with big-M constraints,

$$\begin{gathered} S_{iv} + D_{iv} \leq H \cdot Z_{iv}, \quad i \in T, \ v \in W, \\ V_{iv}^{t} \leq \overline{V_{v}^{t}} \cdot Z_{iv}, \quad i \in T, \ v \in W, \\ V_{iv}^{t} \geq \underline{V_{v}^{t}} \cdot Z_{iv}, \quad i \in T, \ v \in W. \end{gathered} \tag{7}$$

Crude volume variables are positive variables whose sum equals the corresponding total volume variable,

$$\sum_{c \in C} V_{ivc} = V_{iv}^{t}. \tag{8}$$

Total and crude level variables are defined by adding to the initial level in the tank all inlet and outlet transfer volumes of operations of higher priority than the considered priority slot,

$$L_{ir}^{t} = L_{0r}^{t} + \sum_{j \in T, j<i} \sum_{v \in I_r} V_{iv}^{t} - \sum_{j \in T, j<i} \sum_{v \in O_r} V_{iv}^{t}, \quad i \in T, \ r \in R, \tag{9}$$

$$L_{irc} = L_{orc} + \sum_{j \in T, j<i} \sum_{v \in I_r} V_{ivc} - \sum_{j \in T, j<i} \sum_{v \in O_r} V_{ivc}, \quad i \in T, \ r \in R, \ c \in C. \tag{10}$$

2.5.3. Sequencing Constraints. Sequencing constraints restrict the set of possible sequences of operations. Cardinality and unloading sequence constraints are specific cases of sequencing constraints. More complex sequencing constraints will also be discussed later.

2.5.4. Cardinality Constraint. Each crude oil marine vessel has to unload its content exactly once. $\sum_{i \in T} \sum_{v \in O_r} Z_{iv} = 1, r \in R_V$. The total number of distillation operations is bounded by $\underline{N_D}$ and $\overline{N_D}$ in order to reduce the cost of CDU switches,

$$\underline{N_D} \leq \sum_{i \in T} \sum_{v \in W_D} Z_{iv} \leq \overline{N_D}. \tag{11}$$

2.5.5. Unloading Sequence Constraint. Marine vessels have to unload in order of arrival to the refinery. Considering two vessels $r_1, r_2 \in R_V, r_1 < r_2$ signifies that r_1 unloads before r_2,

$$\sum_{j \in T, j<i} \sum_{v \in O_{r_2}} Z_{jv} + \sum_{j \in T, j \geq i} \sum_{v \in O_{r_1}} Z_{jv} \leq 1. \tag{12}$$

2.5.6. Scheduling Constraints. Scheduling constraints restrict the values taken by time variables according to logistics rules.

2.5.7. Nonoverlapping Constraint. A nonoverlapping constraint between two sets of operations $W_1 \subset W$ and $W_2 \subset W$ states that any pair of operations $(v_1, v_2) \subset W_1 \times W_2$ must not be executed simultaneously.

Unloading operations must not overlap,

$$\sum_{v \in W_U} (S_{iv} + D_{iv}) \leq \sum_{v \in W_U} S_{jv} + H \cdot \left(1 - \sum_{v \in W_U} Z_{jv}\right), \quad i, j \in T, \ i < j. \tag{13}$$

Inlet and outlet transfer operations on a tank must not overlap,

$$\begin{gathered}\sum_{v\in I_r}(S_{iv}+D_{iv})\le\sum_{v\in O_r}S_{jv}+H\cdot\left(1-\sum_{v\in O_r}Z_{jv}\right),\\ i,j\in T,\ i<j,\ r\in R_S\cup R_C,\\ \sum_{v\in O_r}(S_{iv}+D_{iv})\le\sum_{v\in I_r}S_{jv}+H\cdot\left(1-\sum_{v\in I_r}Z_{jv}\right),\\ i,j\in T,\ i<j,\ r\in R_S\cup R_C.\end{gathered}\tag{14}$$

Although we do not consider crude settling in storage tanks after vessel unloading, it could be included in the model with a modified version of constraint (14) taking into account transition times. We define TR_V as the transition time after unloading operation $v \in W_U$ and TR as the maximum transition time, $\mathrm{TR} = \max_{v\in W_U}\mathrm{TR}_V$

$$\begin{aligned}&\sum_{v\in I_r}(S_{iv}+D_{iv}+\mathrm{TR}_v\cdot Z_{iv})\\ &\quad\le\sum_{v\in O_r}S_{jv}+(H+\mathrm{TR})\cdot\left(1-\sum_{v\in O_r}Z_{jv}\right).\end{aligned}\tag{15}$$

Constraint (15) is valid in the four possible cases:

$$\begin{aligned}&\left(\exists v_1\in I_r, Z_{iv_1}=1\right)\\ &\quad\wedge\left(\exists v_2\in O_r, Z_{jv_2}=1\right)\Longrightarrow S_{iv}+D_{iv_1}+\mathrm{TR}_{v_1}\le S_{jv_2},\\ &\left(\exists v_1\in I_r, Z_{iv_1}=1\right)\\ &\quad\wedge\left(\bigvee v_2\in O_r, Z_{jv_2}=1\right)\Longrightarrow S_{iv}+D_{iv_1}\le H+\mathrm{TR}-\mathrm{TR}_{v_1},\\ &\left(\bigvee v_1\in I_r, Z_{iv_1}=0\right)\\ &\quad\wedge\left(\exists v_2\in O_r, Z_{jv_2}=1\right)\Longrightarrow 0\le S_{jv_2},\\ &\left(\bigvee v_1\in I_r, Z_{iv_1}=0\right)\\ &\quad\wedge\left(\bigvee v_2\in O_r, Z_{jv_2}=0\right)\Longrightarrow 0\le H+\mathrm{TR}.\end{aligned}\tag{16}$$

A tank may charge only one CDU at a time,

$$\begin{gathered}\sum_{v\in O_r}(S_{iv}+D_{iv})\le\sum_{v\in O_r}S_{jv}+H\cdot\left(1-\sum_{v\in O_r}Z_{jv}\right),\\ i,j\in T,\ i<j,\ r\in R_C.\end{gathered}\tag{17}$$

A CDU may be charged by only one tank at a time,

$$\begin{gathered}\sum_{v\in I_r}(S_{iv}+D_{iv})\le\sum_{v\in I_r}S_{jv}+H\cdot\left(1-\sum_{v\in I_r}Z_{jv}\right),\\ i,j\in T,\ i<j,\ r\in R_D.\end{gathered}\tag{18}$$

To avoid schedules in which a transfer is being performed twice at a time, thus possibly violating the flow rate limitations, constraint (19) is included in the model,

$$S_{iv}+D_{iv}\le S_{jv}+H\cdot\left(1-Z_{jv}\right),\quad i,j\in T,\ i<j,\ v\in W.\tag{19}$$

2.5.8. *Continuous Distillation Constraint.* It is required that CDUs operate without interruption. As CDUs perform only one operation at a time, the continuous operation constraint is defined by equating the sum of the duration of distillations to the time horizon,

$$\sum_{i\in T}\sum_{v\in I_r}D_{iv}=H,\quad r\in R_D.\tag{20}$$

2.5.9. *Resource Availability Constraint.* Unloading of crude oil vessels may start only after arrival to the refinery. Let S_r be the arrival time of vessel r,

$$S_{iv}\ge S_r\cdot Z_{iv},\quad i\in T,\ r\in R_v,\ v\in O_r.\tag{21}$$

2.5.10. *Operation Constraints.* Operation constraints restrict the values taken by operation and time variables according to operational rules.

2.5.11. *Flow Rate Constraint.* The flow rate of transfer operation v is bounded by $\underline{\mathrm{FR}_v}$ and $\overline{\mathrm{FR}_v}$

$$\underline{\mathrm{FR}_v}\cdot D_{iv}\le V_{iv}^t\le\overline{\mathrm{FR}_v}\cdot D_{iv},\quad i\in T,\ v\in W.\tag{22}$$

2.5.12. *Property Constraint.* The property k of the blended products transferred during operation v is bounded by $\underline{x_{vk}}$ and $\overline{x_{vk}}$. The property k of the blend is calculated from the property x_{ck} of crude c assuming that the mixing rule is linear,

$$\underline{x_{vk}}\cdot V_{iv}^t\le\sum_{c\in C}x_{ck}V_{ivc}\le\overline{x_{vk}}\cdot V_{iv}^t,\quad i\in T,\ v\in W,\ k\in K.\tag{23}$$

2.5.13. *Composition Constraint.* It has been shown that processes including both mixing and splitting of streams cannot be expressed as a linear model. Mixing occurs when two streams are used to fill a tank and is expressed linearly in constraint (10). Splitting occurs when partially discharging a tank, resulting in two parts: the remaining content of the tank and the transferred products. This constraint is nonlinear. The composition of the products transferred during a transfer operation must be identical to the composition of the origin tank,

$$\frac{L_{irc}}{L_{ir}^t}=\frac{V_{ivc}}{V_{iv}^t},\quad i\in T,\ r\in R,\ v\in O_r,\ c\in C.\tag{24}$$

Constraint (24) is reformulated as an equation involving bilinear terms,

$$V_{ivc}\cdot L_{ir}^t=L_{irc}\cdot V_{iv}^t,\quad i\in T,\ r\in R,\ v\in O_r,\ c\in C.\tag{25}$$

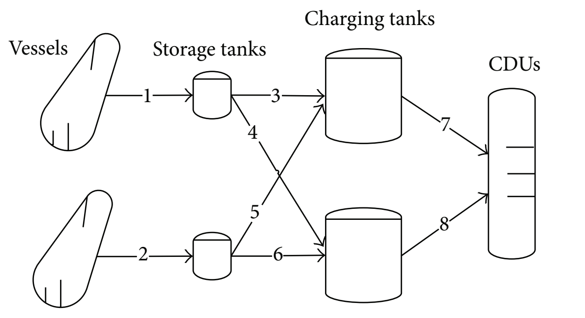


FIGURE 1: Crude oil operations system for the problem.

Note that constraint (25) is correct even when operation v is not assigned to priority-slot i, as then

$$V_{iv}^{t} = V_{ivc} = 0. \tag{26}$$

2.5.14. Resource Constraints. Resource constraints restrict the use of resources throughout the scheduling horizon.

2.5.15. Tank Capacity Constraint. The level of materials in the tank r must remain between minimum and maximum capacity limits $\underline{L_r^t}$ and $\overline{L_r^t}$, respectively. Let L_{0r}^t be the initial total level and let L_{0rc} be the initial level of crude c in the tank r. As simultaneous charging and discharging of tanks is forbidden, the following constraints are sufficient:

$$\begin{gathered}\underline{L_r^t} \le L_{ir}^t \le \overline{L_r^t}, \quad i \in T,\ r \in R_S \cup R_C,\\ 0 \le L_{irc} \le \overline{L_r^t}, \quad i \in T,\ r \in R_S \cup R_C,\ c \in C,\\ \underline{L_r^t} \le L_{0r}^t + \sum_{i\in T}\sum_{v\in I_r} V_{iv}^t - \sum_{i\in T}\sum_{v\in O_r} V_{iv}^t \le \overline{L_r^t},\\ r \in R_S \cup R_C,\\ 0 \le L_{0rc} + \sum_{i\in T}\sum_{v\in I_r} V_{ivc} - \sum_{i\in T}\sum_{v\in O_r} V_{ivc} \le \overline{L_r^t},\\ r \in R_S \cup R_C,\ c \in C.\end{gathered} \tag{27}$$

2.5.16. Demand Constraint. Demand constraints define lower and upper limits, $\underline{D_r}$ and $\overline{D_r}$, on total volume of products transferred out of each charging tank r during the scheduling horizon,

$$\underline{D_r} \le \sum_{i\in T}\sum_{v\in O_r} V_{iv}^t \le \overline{D_r}, \quad r \in R_C. \tag{28}$$

3. Finite State Theory

This section presents in a somewhat informal way those basic notions and definitions from formal language and finite state theories, which are relevant for the sections to follow. Related definitions are taken from literature [20, 21]. Readers, who are unfamiliar with formal language theory, are advised to consult the sources whenever necessary.

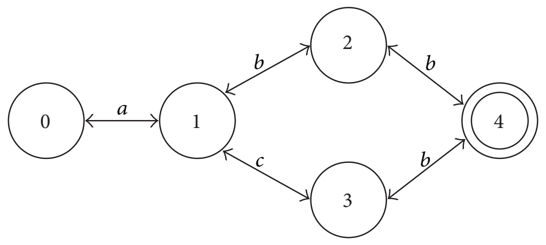


FIGURE 2: A deterministic finite state automaton (DFA).

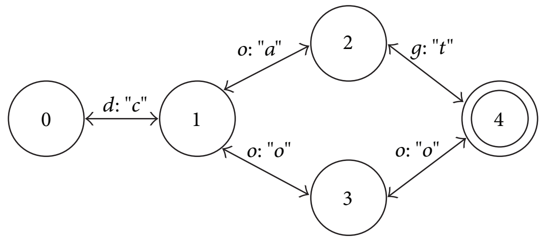


FIGURE 3: Finite state transducer encoding the relation {(dog, cat), (dog, cow)}.

3.1. Finite State Automata. A DFA is a 5-tuple $(Q, \Sigma, \delta, i, F)$, where Q is a set of states, Σ is an alphabet, i is the initial state, $F \subseteq Q$ is a set of final states, and δ is a transition function mapping $Q \times \Sigma$ to Q. That is, for each state u and symbol a, there is at most one state that can be reached from u by "following" a (Figure 2).

3.2. Finite State Transducers. A finite state transducer (FST) is a 6-tuple $(\Sigma_1, \Sigma_2, Q, \delta, i, F)$, where Q, i, and F are the same as for DFA, Σ_1 is input alphabet, Σ_2 is output alphabet, and δ is a function mapping $Q \times (\Sigma_1 \cup \{\varepsilon\}) \times (\Sigma_2 \cup \{\varepsilon\})$ to a subset of the power set of Q (Figure 3). Intuitively, an FST is much like an NFA except that transitions are made on strings instead of symbols and, in addition, they have outputs.

3.3. Finite State Calculus. As argued in Karttunen [22–25], many of the rules used can be analyzed as special cases of regular expressions. They extend the basic regular expression with new operators. These extensions make the finite state automation and finite state transducer become more suitable for particular applications. The system described below was implemented using FSA Utilities [26], a package for implementing and manipulating finite state automata, which provides possibilities for defining new regular expression operators. The part of FSAs built in regular expression syntax relevant to this paper is listed in Table 4.

One particular useful extension of the basic syntax of regular expressions is the replace-operator. Karttunen [22–25] argues that many phonological and morphological rules can be interpreted as rules which replace a certain portion of the input string. Although several implementations of the replace-operator are proposed, the most relevant case for our purposes is the so-called "leftmost longest-match" replacement. In case of overlapping rule targets in the input, this operator will replace the leftmost target, and in cases where

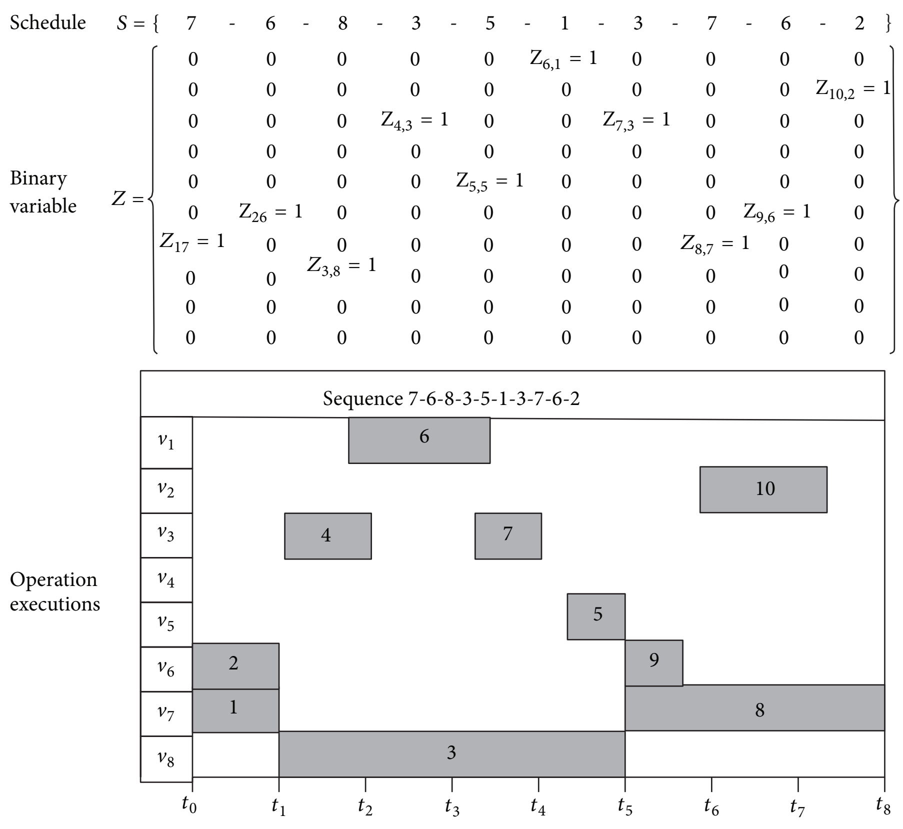


FIGURE 4: An example to indicate the relationship between binary variable and schedule.

a rule target contains a prefix which is also a potential target, the longer sequence will be replaced. Gerdemann and van Noord [27] implement leftmost longest-match replacement in FSA as the operator:

$$\text{replace}\ (\text{Target}, \text{LeftContext}, \text{RightContext}), \qquad (29)$$

where Target is a transducer defining the actual replacement and LeftContext and RightContext are regular expressions defining the left and right context of the rule, respectively. The segmentation task discussed in the mutation procedure makes crucial use of longest-match replacement.

4. The Hybrid Algorithm

From the point view of optimization efficiency and robustness, a novel two-level optimization framework based on finite state method and GA is proposed for the MINLP model in this section.

4.1. Two-Level Optimization Structure. As the foundation of the framework, a two-level optimization structure is introduced. Once all binary variables are fixed the original problem becomes a relatively simpler model with only continuous variable. Following this deal, we rewrite (5) as follows:

$$\max (J(\xi, z)) \Longleftrightarrow \max_{z} \left[\max_{\xi} J(\xi, \overline{z}) \right], \qquad (30)$$

where ξ and z represent continuous and binary variables, respectively. Equation (30) shows when z is fixed as $\overline{z}$, the submodel $J(\xi, \overline{z})$ can be solved optimally by continuous-optimization solvers in the inner level; then we update $\overline{z}$ towards the best binary solution z^* in the outer level.

We used an example in Figure 4 to show how binary solution can be mapped to a scheduling sequence. The schedule $S = [7683513762]$ where 7 stands for the specific operation 7 to assign to position 1 corresponding to the binary decisions $Z_{17} = 1$.

4.2. Initial Population. Based on the sequencing rules [18] and the extension to the regular expression calculus [22–25], a DFA model which builds legal schedules complying with sequencing rules and operation condition is constructed. The whole set of possible schedules is too huge to be processed at once. The DFA model of the schedule constitutes a reasonable framework, capturing all possible schedules and removing many redundant sequences of operations. Initial values of decision variables must satisfy the equality constraints and operation condition and therefore represent a feasible operating point.

Here, we still use the instance with 8 operations from Mouret et al. [18] to describe an efficient sequencing rule by

```
macro (procedure,
        segmentation, % segmentation of the input sequence into a set of sub-sequence
        mutation, % apply mutation rules
        clean up) % remove markers
```

Algorithm 1

using a regular expression. A feasible sequence $v_1 \cdots v_i \cdots v_n$ can be described by the following:

$$\begin{aligned} \text{sequence} &= (\varepsilon + L_a)(L_b \cdot L_a)^* (\varepsilon + L_b), \\ L_a &= 7(\varepsilon + 4)(\varepsilon + 6)(\varepsilon + 1 + 14)(\varepsilon + 2 + 26), \\ L_b &= 8(\varepsilon + 3)(\varepsilon + 5)(\varepsilon + 1 + 13)(\varepsilon + 2 + 25). \end{aligned} \tag{31}$$

However, this automation suffers from a serious problem of overgeneration. For example, the short length of the sequence may lead to infeasibility, while the long length of the sequence may result in an unsolvable model. It is an interesting challenge for finite state syntactic description to specify a sublanguage that contains all and only the sequences of valid length.

Our solution is to construct a suitable constraint for the sequences of valid length. The constraint expressions denote a language that admits sequences of valid length but excludes all others. We obtain the desired effect by intersecting the constraint language with the original language of sequence expressions. The intersection of the two languages contains all and only the valid dates:

$$\text{ValidSequence} = \text{Sequence} \cap \text{ValidLength}. \tag{32}$$

The ValidLength constraint is a language that includes all sequences of length n:

$$\text{ValidLength} = (1 + 2 + 3 + 4 + 5 + 6 + 7 + 8)^n. \tag{33}$$

We have now completed the task of describing the language of valid sequences from the set of possible sequence expressions. It is also possible to create an automation on the basis of the regular expression and ValidSequence and then generate all possible sequences $v_1 \cdots v_i \cdots v_n$ accepted by the automaton. The processes are implemented using FSA Utilities [26] that is a package for implementing and manipulating DFA and finite state transducer. In order to generate all possible sequences. When all possible sequences $v_1 \cdots v_i \cdots v_n$ accepted by the automaton are generated, and the population of the according possible binary decisions is generated. In the initial population stage of GA, the population size is the number of individuals. When the number of individuals is given, a population of candidate solutions is generated by randomly selecting from the population of the all possible binary decisions.

4.3. Rule-Based Mutation Approach.

In the mutation stage, we use a finite state transducer for this rule-based mutation process. The rule-based mutation strategy must obey the sequencing rule and the nonoverlapping constraint such that all involved solutions in GA are feasible.

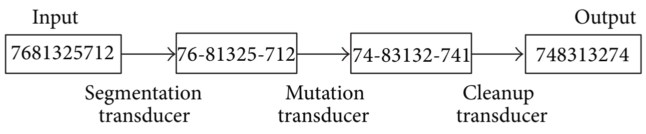


Figure 5: An example of mutation.

```
replace (
        [identity(SSequence),[]x-],[],]
        ).
```

Algorithm 2

The proposed mutation approach is a two-step procedure.

Step 1. Segmentation of the input sequence into a set of subsequences (i.e., the subsequence which belongs to the regular language L7 or L8).

Step 2. Mutation of the subsequences into others.

Formally, the rule-based mutation procedure is implemented as the composition of three transducers (see Algorithm 1).

An example of mutation including the intermediate steps is given for the sequence "7681325712" as shown in Figure 5.

4.3.1. Segmentation Transducer.

Segmentation transducer splits an input sequence into subsequences. The goal of segmentation is to provide a convenient representation level for the next mutation step.

Segmentation is defined as shown in Algorithm 2.

The macro "SSequence" defines the set of subsequences. The subsequences which belong to the regular language L7 and L8 are displayed in Tables 1 and 2. Segmentation attaches the marker "–" to each subsequence. The Targets are identified using leftmost longest-match, and thus at each point in the input, only the longest valid segment is marked.

4.3.2. The Mutation Rules.

In the GA process, the mutation rules are made by carefully considering nonoverlapping constraint between operations. A concrete instance for partially illustrating the mutation rules is given in Algorithm 3. Note that the final element of the left-context must be a marker and

```
marco (conversion, $L_7$_subsequence_rules)
                    ∘ $L_8$_subsequence_rules)
marco ($L_7$_subsequence_rules,
replace ({2, 4, 6} × 1, 7, —)
∘ replace ({14, 26, 41, 42, 46, 61, 62} × 12, 7, —)
∘ replace ({142, 412, 414, 426, 461, 462, 612, 614, 626} × 126, 7, —)
marco ($L_8$_subsequence_rules,
replace ({2, 3, 5} × 1, 8, —)
∘ replace ({13, 25, 31, 32, 35, 51, 52} × 12, 8,—)
∘ replace ({132, 312, 313, 325, 351, 352, 512, 513, 525} × 125, 8, —)
```

Algorithm 3: An example to demonstrate the mutation rules.

Table 1: Subsequence belonging to L_7.

Length	Sequences belonging to L_7									
1	7									
2	71	72	74	76						
3	712	714	726	741	742	746	761	762		
4	7126	7142	7412	7414	7426	7461	7462	7612	7614	7626
5	71426	74126	74142	74612	74614	74626	76126	76142		
6	741426	746126	746142	761426						
7	7461426									

the target itself ends in "–." This ensures that mutation rules cannot apply to the same subsequence.

5. Experimental Study

In this section, the same problem from the literature [18] is used for computational experiments. The proposed methodology is compared with existing promising algorithms, mixed-coding GA [15, 28]. Figure 1 depicts the refinery configuration for problem. The data involved in the problem are given in Table 3. The performance comparison with different computing times, such as 350 s, 500 s,. . ., 2400 s, is conducted. The objective value is used to statistically analyze the optimization results.

The performance comparison between the two methodologies used is illustrated in Figure 6, which shows that the hybrid optimization algorithm which combined the finite state method and GA will statistically outperform the mixed-coding counterpart. The genetic algorithm which combined the finite state method and GA finds feasible solutions very fast and is able to find better solutions in reasonable time.

In Figure 7, we compare the objective variance of each iteration in the two evolution processes of these two kinds of methodology. By tracking the evolution process, we find that the mixed-coding GA is easy to stick in a local minimal sequence solution. This situation only can be improved through increasing the mutation scaling factor. However, this may result in a hard convergence, unless sufficient iterations are implemented. As for the hybrid optimization algorithm, the optimization processes of binary variable and continuous variable are separated. The performance of the whole methodology mainly depends on the FSM

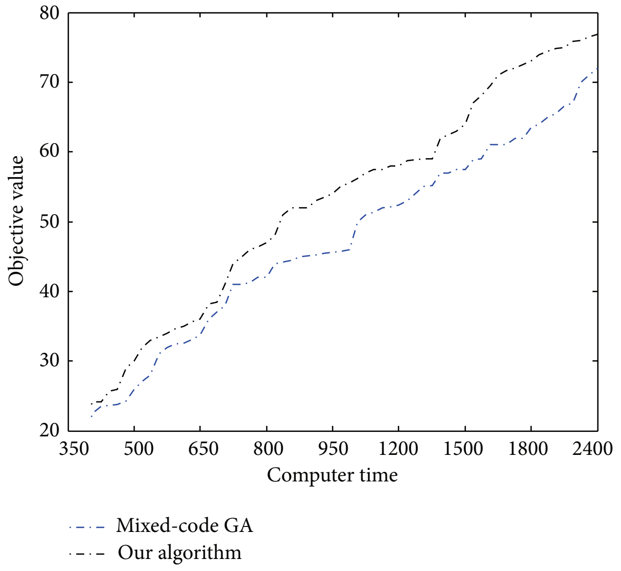


Figure 6: Average objective values of two methodologies.

which captures most promising schedules and removes many redundant sequences of operations, so that the user can use a small population size of corresponding discrete variables to obtain suboptimal solutions. From Figure 7, we see that the proposed method has converged at 350 iterations as opposed to 2400 iterations for the mixed-coding GA.

The success of the proposed algorithm lies in a comprehensive analysis of the region of the search space and its capacity to focus the search on the regions with the partial solution. One of the good merits of the hybrid algorithm is

Table 2: Subsequence belonging to L_8.

Length	Sequences belonging to L_8								
1	8								
2	81	82	83	85					
3	812	813	825	831	832	835	851	852		
4	8125	8132	8312	8313	8325	8351	8352	8512	8513	8525
5	81325	83125	83132	83512	83513	83525	85132	85125		
6	831325	835125	835132	851325						
7	8351325									

Table 3: Problem data.

	Scheduling horizon		8 days		
Vessels	Arrival time		Composition	Amount of crude	
Vessel 1	0		100% A	1000	
Vessel 2	4		100% B	1000	
Storage tanks	Capacity		Initial composition	Initial amount	
Tank 1	[0, 1000]		100% A	250	
Tank 2	[0, 1000]		100% B	750	
Charging tanks	Capacity		Initial composition	Initial amount	
Tank 1 (mix X)	[0, 1000]		100% C	500	
Tank 1 (mix X)	[0, 1000]		100% D	500	
Crudes	1	Gross margin	Crude mixtures	Property1	Demand
Crude A	0.01	9	Crude mix X	[0.015, 0.025]	[1000, 1000]
Crude B	0.06	4	Crude mix Y	[0.045, 0.055]	[1000, 1000]
Crude C	0.02	8	Unloading flow rate	[0, 500]	
Crude D	0.05	5	transfer flow rate	[0, 500]	

Table 4: A fragment of FSA regular expression syntax and U transducers, and R can be either.

[]:	The empty string
$[R_1, \ldots, R_n]$:	Concatenation
$\{R_1, \ldots, R_n\}$	Disjunction
$R^\wedge$:	Optionality
Identity (A):	Identity: the transducer which maps each element in A onto itself
$T \circ U$:	Composition of the transducers T and U
macro (Term, R):	Use term as an abbreviation for R

that each solution involved in the GA algorithm is guaranteed to be feasible by using the mutation rules generated by DFM method while in existing GA algorithms the procedure to generate feasible solution under complex process constraints is very time costive. The deterministic finite automata (DFA) can easily represent this kind of structure. Furthermore, the complex process constraints can be very difficult to express with mixed integer programming. Consequently, it is unfeasible to solve the industrial problem by using MIP solver.

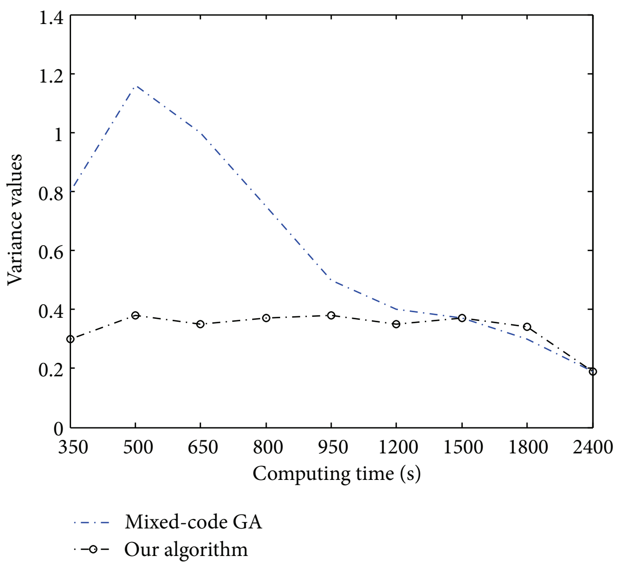


Figure 7: Variance values of two methodologies.

6. Conclusion

In this paper, a novel hybrid optimization algorithm which combined the finite state method and GA is proposed. The proposed algorithm constitutes a reasonable framework, capturing both the operating condition and sequencing rule of the schedule. The solution captures all possible schedules and removes many redundant sequences of operations. The algorithm is equivalent to introducing new structure information into the optimization process, which will help reduce the risk of trapping in a local minimal sequence solution. The hybrid optimization algorithm is an effective and robust tool to solve the crude oil scheduling problem in terms of efficiency and reliability. Algorithms only with the two properties are suitable for solving practical engineering application.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research is partially supported by the China National Natural Science Foundation under Grant 61203178, Grant 61304214, and Grant 61290323. The authors thank the financial funds from Shanghai Science and Technology Committee under Grant 12511501002 and Grant 13511501302.

References

[1] J. M. Pinto, M. Joly, and L. F. L. Moro, "Planning and scheduling models for refinery operations," *Computers and Chemical Engineering*, vol. 24, no. 9-10, pp. 2259–2276, 2000.

[2] J. Li, I. A. Karimi, and R. Srinivasan, "Recipe determination and scheduling of gasoline blending operations," *AIChE Journal*, vol. 56, no. 2, pp. 441–465, 2010.

[3] J. Li, R. Misener, and C. A. Floudas, "Continuous-time modeling and global optimization approach for scheduling of crude oil operations," *AIChE Journal*, vol. 58, no. 1, pp. 205–226, 2012.

[4] Z. Jia, M. Ierapetritou, and J. D. Kelly, "Refinery short-term scheduling using continuous time formulation: crude-oil operations," *Industrial and Engineering Chemistry Research*, vol. 42, no. 13, pp. 3085–3097, 2003.

[5] C. A. Méndez, I. E. Grossmann, I. Harjunkoski, and P. Kaboré, "A simultaneous optimization approach for off-line blending and scheduling of oil-refinery operations," *Computers and Chemical Engineering*, vol. 30, no. 4, pp. 614–634, 2006.

[6] M. Pan, X. Li, and Y. Qian, "New approach for scheduling crude oil operations," *Chemical Engineering Science*, vol. 64, no. 5, pp. 965–983, 2009.

[7] M. F. Cardoso, R. L. Salcedo, S. F. de Azevedo, and D. Barbosa, "A simulated annealing approach to the solution of minlp problems," *Computers and Chemical Engineering*, vol. 21, no. 12, pp. 1349–1364, 1997.

[8] I. E. Grossmann, "Review of nonlinear mixed-integer and disjunctive programming techniques," *Optimization and Engineering*, vol. 3, no. 3, pp. 227–252, 2002.

[9] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: a survey," *Operations Research*, vol. 14, no. 4, pp. 699–719, 1966.

[10] M. A. Duran and I. E. Grossmann, "An outer-approximation algorithm for a class of mixed-integer nonlinear programs," *Mathematical Programming*, vol. 36, no. 3, pp. 307–339, 1986.

[11] A. M. Geoffrion, "Generalized Benders decomposition," *Journal of Optimization Theory and Applications*, vol. 10, no. 4, pp. 237–260, 1972.

[12] C. D'Ambrosio and A. Lodi, "Mixed integer nonlinear programming tools: a practical overview," *4OR*, vol. 9, no. 4, pp. 329–349, 2011.

[13] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, The University of Michigan Press, Ann Arbor, Mich, USA, 1975.

[14] T. Yokota, M. Gen, and Y.-X. Li, "Genetic algorithm for non-linear mixed integer programming problems and its applications," *Computers and Industrial Engineering*, vol. 30, no. 4, pp. 905–917, 1996.

[15] L. Costa and P. Oliveira, "Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems," *Computers and Chemical Engineering*, vol. 25, no. 2-3, pp. 257–266, 2001.

[16] J. M. Ponce-Ortega, M. Serna-González, and A. Jiménez-Gutiérrez, "Heat exchanger network synthesis including detailed heat exchanger design using genetic algorithms," *Industrial and Engineering Chemistry Research*, vol. 46, no. 25, pp. 8767–8780, 2007.

[17] K.-M. Björk and R. Nordman, "Solving large-scale retrofit heat exchanger network synthesis problems with mathematical optimization methods," *Chemical Engineering and Processing: Process Intensification*, vol. 44, no. 8, pp. 869–876, 2005.

[18] S. Mouret, I. E. Grossmann, and P. Pestiaux, "A novel priority-slot based continuous-time formulation for crude-oil scheduling problems," *Industrial and Engineering Chemistry Research*, vol. 48, no. 18, pp. 8515–8528, 2009.

[19] H. Lee, J. M. Pinto, I. E. Grossmann, and S. Park, "Mixed-integer linear programming model for refinery short-term scheduling of crude oil unloading with inventory management," *Industrial and Engineering Chemistry Research*, vol. 35, no. 5, pp. 1630–1641, 1996.

[20] J. E. Hopcroft, *Introduction to Automata Theory, Languages, and Computation*, Pearson Education, India, New Delhi, India, 3rd edition, 2008.

[21] E. Roche and Y. Schabes, *Finite-State Language Processing*, The MIT Press, Cambridge, Mass, USA, 1997.

[22] L. Karttunen, "Constructing lexical transducers," in *Proceedings of the 15th conference on Computational Linguistics*, vol. 1, Association for Computational Linguistics, 1994.

[23] L. Karttunen, "The replace operator," in *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, 1995.

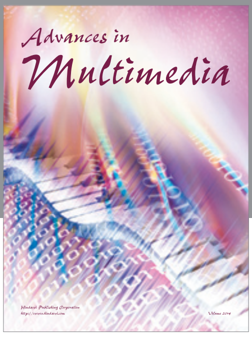
[24] L. Karttunen, "Directed replacement," in *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, 1996.

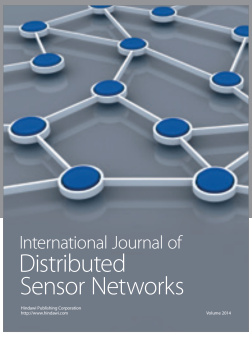
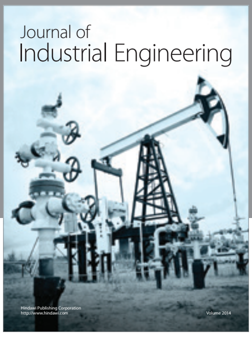
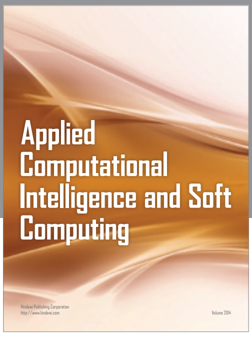
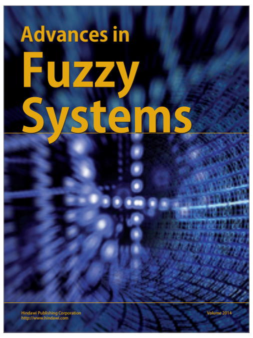
[25] L. Karttunen and K. R. Beesley, *Two-Level Rule Compiler*, Xerox Corporation, Palo Alto Research Center, 1992.

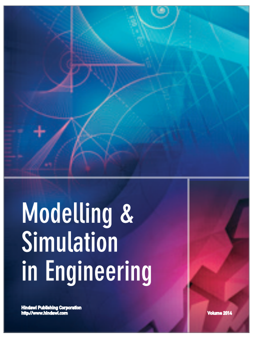
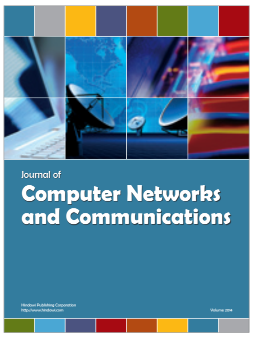
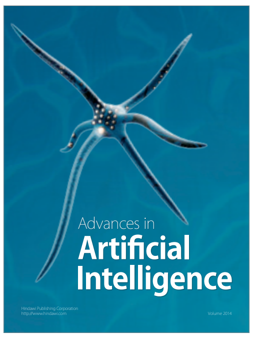
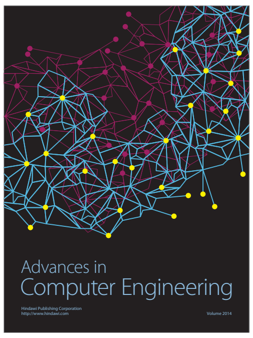
[26] G. van Noord, "FSA utilities: a toolbox to manipulate finite-state automata," in *Automata Implementation*, pp. 87–108, Springer, New York, NY, USA, 1997.

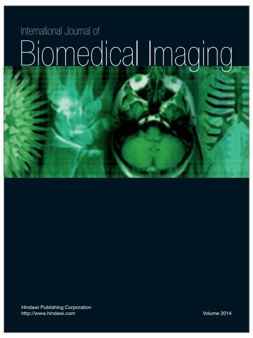
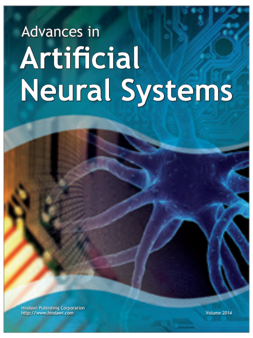
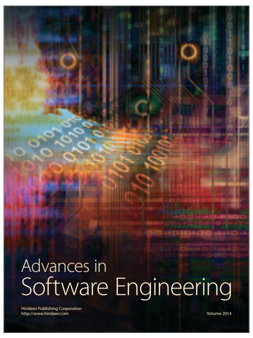
[27] D. Gerdemann and G. van Noord, "Transducers from rewrite rules with backreferences," in *Proceedings of the 9th Conference on European Chapter of the Association for Computational Linguistics*, Association for Computational Linguistics, 1999.

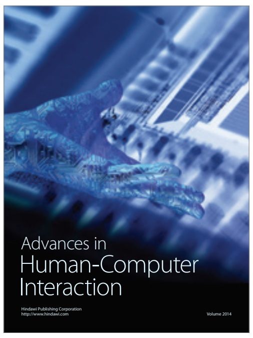
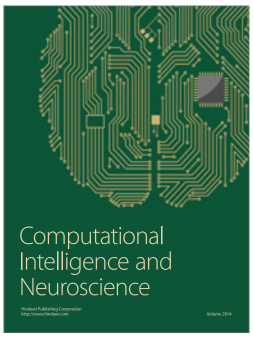
[28] Y.-C. Lin, K.-S. Hwang, and F.-S. Wang, "A mixed-coding scheme of evolutionary algorithms to solve mixed-integer non-linear programming problems," *Computers and Mathematics with Applications*, vol. 47, no. 8-9, pp. 1295–1307, 2004.


Advances in
Multimedia

The Scientific World Journal

International Journal of
Distributed Sensor Networks

Journal of
Industrial Engineering

Applied Computational Intelligence and Soft Computing

Advances in
Fuzzy Systems


Modelling & Simulation in Engineering

Journal of
Computer Networks and Communications

Advances in
Artificial Intelligence

Advances in
Computer Engineering

International Journal of
Computer Games Technology

International Journal of
Biomedical Imaging

Advances in
Artificial Neural Systems

Advances in
Software Engineering

Journal of
Robotics

Advances in
Human-Computer Interaction

Computational Intelligence and Neuroscience

International Journal of
Reconfigurable Computing

Journal of
Electrical and Computer Engineering