International Journal of Innovative Computing, Information and Control
Volume **8**, Number **2**, February **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 1463–1477

GENERATING VALID REFERENCE BUSINESS PROCESS MODEL USING GENETIC ALGORITHM

Bernardo Nugroho Yahya[1], Hyerim Bae[1,*], Joonsoo Bae[2] and Dongsoo Kim[3]

[1]Department of Industrial Engineering
Pusan National University
30-san, Jangjeon-dong, Geumjong-gu, Busan 609-735, South Korea
bernardo@pusan.ac.kr; *Corresponding author: hrbae@pusan.ac.kr

[2]Department of Industrial and Information Systems Engineering
Chonbuk National University
664-14 1-Ga Deokjin-dong, Jeonju, Jeonbuk 561-756, South Korea
jsbae@chonbuk.ac.kr

[3]Department of Industrial and Information Systems Engineering
Soongsil University
1-1 Sangdo-dong, Dongjak-gu, Seoul 156-743, South Korea
dskim@ssu.ac.kr

Received November 2010; revised April 2011

Abstract. *With the growth of the Business Process Management (BPM) market, customer requirements based process customization has emerged as a key to competitive advantage. Where there are numerous demands for customization, a great number of business process model variations result. The existence of numerous process variations leads to both process redundancy and process underutilization, which impact on business performance in negative ways. Thus, there is a need to create a process reference model that can identify and find a representative model without redundancy. This paper introduces a new, Genetic Algorithm (GA)-based approach to generation of a valid business process reference model from a process repository. Previous research using the integer programming (IP) problem produced a reference process model. However, there are problems, pertaining to presentational limitations and unguaranteed validation processes. The GA procedure uses the randomness of genetic processes to resolve both problems in IP formulation. Near the end of this paper, we show the experimental results of the proposed method, which is conveniently executed using business process structure properties and the proximity of activities. It is believed that this process reference model can help a novice process designer to create a new process based on existing processes.*

Keywords: Business process modeling, Process reference model, Genetic algorithm, Proximity score

1. Introduction.

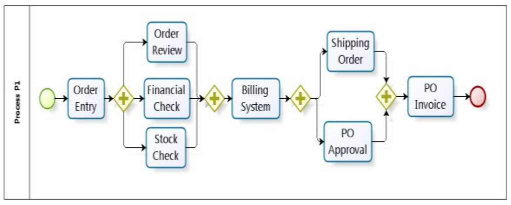
In dynamic business environments, there exists a commonality among process variants having common business goals but variance among process structures [1]. These variations can occur as a consequence of individual customer requirements and the different concerns of organizational units. Generally, process adaptations are needed for configuration purposes at build time. Adaptive process management technology has emerged in response to these needs. Accordingly, a significant number of process variants are created from the same process model, but those variants slightly differ from each other in their structures. There are only a few such approaches, which utilize information on these variants and the corresponding process adaptations [2-5].

Figure 1 illustrates the variation of business processes having common goals, showing seven process variants as examples, modeled using Business Process Modeling Notation (BPMN). Logistics processes are good examples to represent the various process variants. It displays the setup workflow categories through flexible user-defined codes based on customer numbers, customer types, quantity ordered, product categories, item numbers, various combinations thereof, and/or special prices [6, 7]. The characteristics of each process example have to be modeled for a specific goal (meet the requirements of each product type) in the logistics process. In the example of purchase order process, eight activities are initialized: order entry, order review, financial check, stock check, manager review, billing system, shipping order, purchase order (PO) approval and PO invoice. According to the given process context, the concerns of a certain organizational unit and specific customer requirements, different variations of a basic process are needed. Examples of these, again, are shown in Figure 1.

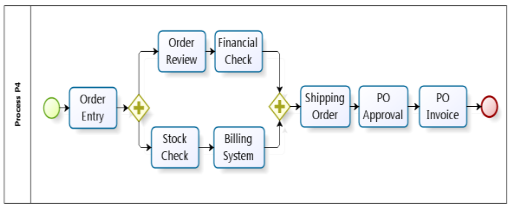
The generation of a generic process model, as a new reference model, is necessary for future adaptations and decreasing change costs. Industrial process model collections and reference process models such as ITIL, SCOR and eTOM exist, however, its high level reference model corresponds to process-guidelines to match different aspects of IT, industry, application, services, etc. without considering the level of implementation. Moreover, existing approaches only attempted to create reference models based on history of process configurations [2-4]. Thus, the present study developed a method for generating the best reference model from a large number of past business process variants without any history information of process configurations. The method can create a reference model with a combinatorial optimization problem by using distance measure and considering a process model's mathematical formulation for optimization of the activity distances in a process model [8]. However, in the model's application, there are two limitations. First, the math formulation has a semantic presentational limitation. A presentational limitation is claimed as no consideration with regard to activity properties, e.g., split and merge activity. Second, process may satisfy the safety property without any guarantee on process validation. The safety property, as a part of soundness property, mentioned that a business process will always satisfy a given property, e.g., it will always run to completion [9]. However, there is no guarantee when a process has safety property, and it also follow the soundness property as a validation approach.

To overcome these limitations, we provide a new, Genetic Algorithm (GA)-based method of generating a BP reference model from a BP repository. We adopted a measure to assess the proximity distance of activities in order to evaluate both integer programming (IP) and GA measurements. Measuring the proximity between activities is a means of extracting process structure knowledge from the process repository. A reference process is considered as a valid model whenever it follows soundness properties [9] and corresponds to the characteristics of existing process variants [5]. That is, by maximizing the proximity value, we can obtain a reference model that can be representative of process variants in a repository. The generated reference model can also be considered as process template/library in order for ease of use the user experience design of BP.

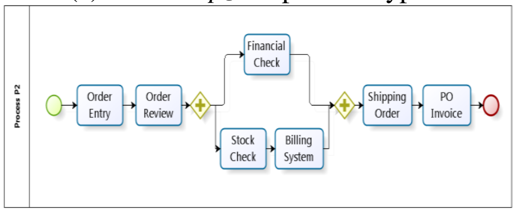
This paper proceeds as follows. In Section 2, we briefly review the literature on GA and graph theory in the Business Process Management (BPM) field. In Section 3, we incorporate the proposed proximity score measurement (PSM) method into the evaluation function of an IP problem. The GA method is proposed as a way to improve IP results. Genetic processes such as selection, crossover and mutation methods are also presented. Section 4 examines the experimental results using the IP problem and GA. Finally, Section 5 concludes this study.



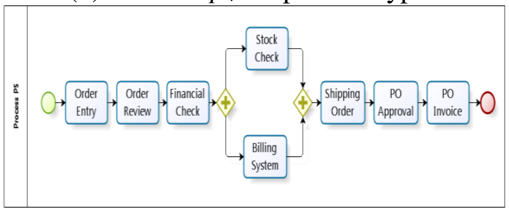
(a). Process p_1 for product type 1



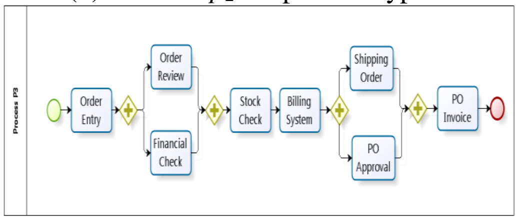
(d). Process p_4 for product type 4



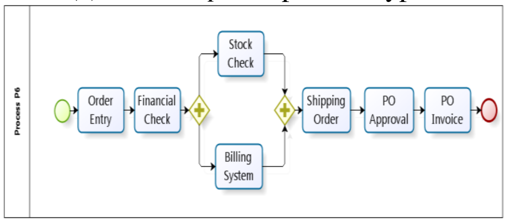
(b). Process p_2 for product type 2



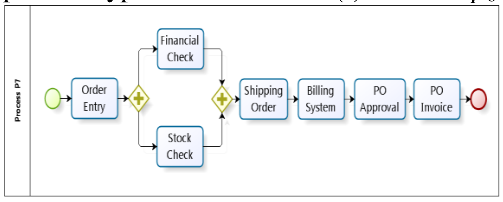
(e). Process p_5 for product type 5



(c). Process p_3 for product type 3



(f). Process p_6 for product type 6



(g). Process p_7 for product type 7

FIGURE 1. Process variants in logistics

2. **Related Work.** Although BP modeling recently has seen wide adoption among enterprises, BP design research is still in its developmental phase. There are several research works of GA in the domain of BP [10-15]. The term Genetic Process Mining (GPM) was proposed to retrieve process knowledge from logs [10]. The retrieval knowledge represented as the process model that reflects the process behavior in the log. Other similar domain, based on project management, dealt with GA approach in personnel assignment problem and network models and optimization [11, 12]. Additional attributes, e.g., resources, in the activity to determine the optimum solution of objective function are out of scope of this study. The design and model of GA approaches, encoding and decoding function, that were carried out are somewhat significant to this study [13]. Other GA approaches were used to solve subset sum problem and graph problem, which is similar to the domain of this study [14, 15]. However, there is no research attempt on finding reference model using GA.

There are some existing business-process-design-related research fields, which usually are titled business process modeling, workflow verification, process change and workflow patterns [16-19]. Process configurations using version management approach was discussed previously [16]. Research about pattern-based workflow design using Petri net was also proposed [17]. Kim and Kim [19] developed a process design tool with regard to fault-tolerant process debugger. Jung, Bae and Liu [18] discussed a method to find similar process by clustering stored processes in repository. All researches developed such approaches to improve new process disregard of reference models, and they become the basic of BP modeling by means of process reference models.

Any discussions of reference model issues are mostly pertinent to process variants [2-5]. Recently, a comprehensive heuristics-based approach to discovering new reference models by learning from past process configurations was discussed [2], and a mathematical programming approach was introduced [8]. The proposed heuristic [2] updates process configurations and produces new reference models based on a minimum edit distance from initial reference processes. However, most traditional process design tools have lack of functions to store process configurations. When there are a lot of processes already stored in the repository, it requires a special method to generate process reference model without any process configuration information. The IP-based mathematical model [8] was proposed to address the issue of creating reference processes without initial reference information or process reconfiguration. There remain problems in the presentational and validation aspects of the process using IP formulations, which this study attempted to overcome with a GA approach. The industry papers using refactoring operations [3] emerged as process configuration tool from AS-IS into TO-BE process models. The scenario of refactoring operations took into account an existing process models (AS-IS model) with a reference model where some parts of the existing model should be preserved and others should be replaced, named as process merging. Process merging differs from process configurations in the way that it usually requires to preserve certain parts of the AS-IS model and the underlying IT system. It is significant to find configured reference model from TO-BE process models that is less considerable on the underlying approach. Scenario-based analysis on application of reference process models in SOA [4] and survey results and classification [5] have all necessary concepts in regard to process reference models with less quantitative techniques. Thus, this proposed method attempt to enhance and overcome problems occurs on the previous works as a mean of creating a valid reference process from repositories without information of process configurations.

3. Proposed Model.

We measured the process structure distance using PSM. For a simpler distance measure, the process variants in Figure 1 were transformed into graph abstractions, as illustrated in Figure 2. In this approach, the transformation of process variant into graph abstraction used string edit distance [20]. We also assume that process variants are considered as directed acyclic graph. It means that we consider no loop (cyclic) activity in the measurement. A discussion of distance measurement using PSM and activity proximity score can be found in the following section.

3.1. Proximity score measurement (PSM).

This section describes some definitions as a prerequisite to convey about business process and activity proximity measurement.

Definition 3.1. *Process Model. We define a process model p^k, which means the k-th process in a process repository. It can be represented as a tuple of $\langle A^k, L^k \rangle$, each element of which is defined below.*

$A^k = \{a_i | i = 1, \ldots, I\}$ is a set of activities where a_i is the i-th activity of p_k and I is the total number of activities in p_k.

$L^k \subseteq \{l_{ij} = (a_i, a_j) | a_i, a_j \in A^k\}$ is a set of links where l_{ij} is the link between two activities a_i and a_j in the k-th process. The element (a_i, a_j) represents the fact that a_i immediately precedes a_j.

Definition 3.2. *Activity Proximity Score. We have to obtain the Activity Proximity Score (APS) for each process. The APS value, which is denoted by q^k_{ij}, is defined in Equation*

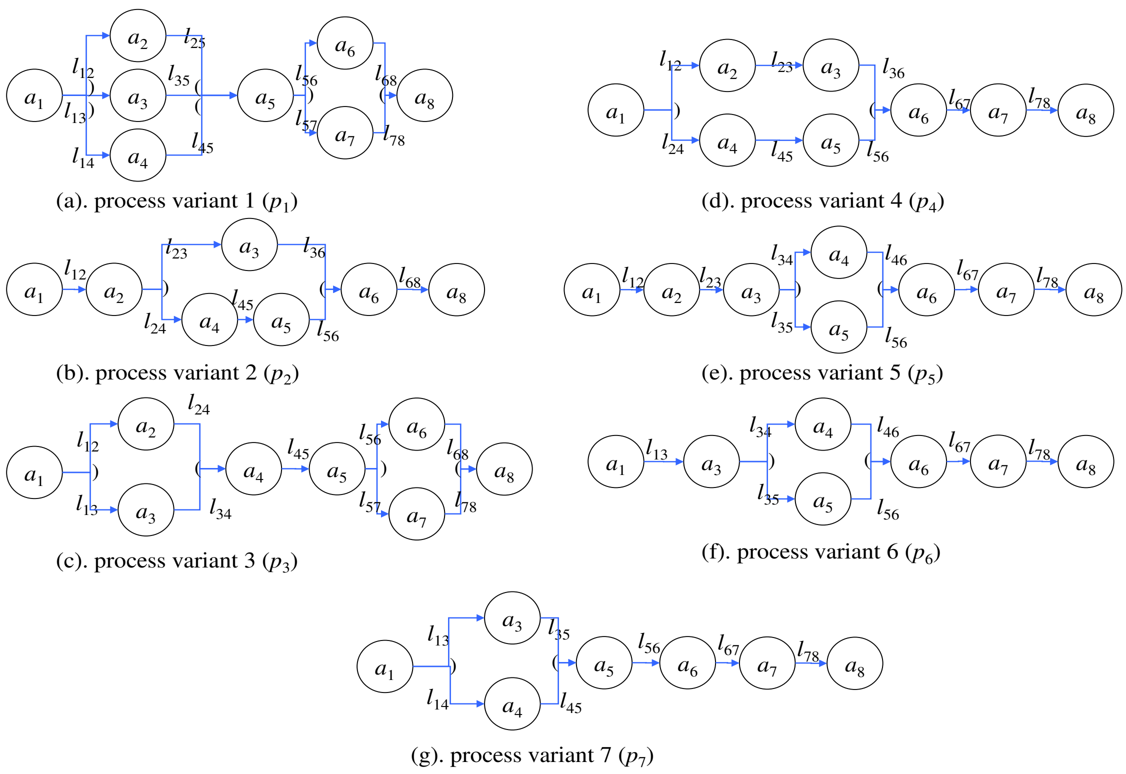

a_1 l_{12} a_2 l_{13} a_3 l_{14} a_4 l_{25} l_{35} l_{45} a_5 l_{56} a_6 l_{57} a_7 l_{68} l_{78} a_8
(a). process variant 1 (p_1)
a_1 l_{12} a_2 l_{23} a_3 l_{24} a_4 l_{45} a_5 l_{36} l_{56} a_6 l_{68} a_8
(b). process variant 2 (p_2)
a_1 l_{12} a_2 l_{13} a_3 l_{24} l_{34} a_4 l_{45} a_5 l_{56} a_6 l_{57} a_7 l_{68} l_{78} a_8
(c). process variant 3 (p_3)
a_1 l_{12} a_2 l_{23} a_3 l_{24} a_4 l_{45} a_5 l_{36} l_{56} a_6 l_{67} a_7 l_{78} a_8
(d). process variant 4 (p_4)
a_1 l_{12} a_2 l_{23} a_3 l_{34} a_4 l_{35} a_5 l_{46} l_{56} a_6 l_{67} a_7 l_{78} a_8
(e). process variant 5 (p_5)
a_1 l_{13} a_3 l_{34} a_4 l_{35} a_5 l_{46} l_{56} a_6 l_{67} a_7 l_{78} a_8
(f). process variant 6 (p_6)
a_1 l_{13} a_3 l_{14} a_4 l_{35} l_{45} a_5 l_{56} a_6 l_{67} a_7 l_{78} a_8
(g). process variant 7 (p_7)

FIGURE 2. Graph abstraction from Figure 1

(1) as

$$q_{ij}^{k} = \frac{h^k(i,j)}{d_{ij}^{k}} \tag{1}$$

where $h^k(i,j) = 1$ if $a_i \rightarrow a_j$ in the k-th process; 0, otherwise, and d_{ij}^k is the average path distance between activity a_i and a_j of the k-th process. Each pair of activities (a_i and a_j) has a single value of APS, $k = 1, 2, 3, \ldots, K$, where q_{ij}^k is the APS of the k-th process in a process repository, K is the total number of processes, and $a_i \rightarrow a_j$ denotes that activity a_j is reachable from a_i. Detailed distance calculations can be found in Yahya et al. [21].

Definition 3.3. *InDegree and OutDegree of activity. InDegree defines the number of edges incoming to an activity, and OutDegree defines the number of edges outgoing from an activity. We denote the InDegree and the OutDegree of the i-th activity as inDegree(a_i) and outDegree(a_i), respectively, and according to these concepts, we can define start/end activities and split/merge semantics.*

Start activity (a_S) is an activity with an empty set of preceding activities, inDegree(a_i) = 0. End activity (a_E) is an activity with an empty set of succeeding activities, outDegree(a_i) = 0.

For a split activity a_i such that outDegree(a_i) > 1, $f_s(a_i)$ = AND if all of the succeeding activities should be executed; otherwise, $f_s(a_i)$ = OR.

For a merge activity a_i such that inDegree(a_i) > 1, $f_m(a_i)$ = AND if all of the preceding activities should be executed; otherwise, $f_m(a_i)$ = OR.

In this study, a reference process is considered as a valid process (validity property) if its activity satisfy the split and merge property using the InDegree and OutDegree value.

3.2. **Integer programming mathematical formulation.** A process of automatic reference model creation finds an optimum reference process by maximizing the sum of proximity scores among the process variants in a process repository. The following notations, extended from [10], are used in the mathematical formulation of our problem. Notice that y_i, z_j and x_{ij} are decision variables.

i, j: activity index (i, j = 1, …, I), where I is the number of activities;
k: process variant index (k = 1, …, K), where K is the number of process variants;
y_i: 1, if the i-th activity is a start activity; 0, otherwise;
z_j: 1, if the j-th activity is an end activity; 0, otherwise;
x_{ij}: 1, if the i-th activity immediately precedes the j-th activity; 0, otherwise.
Mathematical Formulation

$$\min \sum_i \sum_j ((K - c_{ij}).x_{ij} + c_{ij}(1 - x_{ij})) \tag{2}$$

$$\sum_i^I y_i = 1 \tag{3}$$

$$\sum_j^I z_j = 1 \tag{4}$$

$$y_i + x_{ji} \leq 1 \quad \forall k, \quad q_{ji}^k = 1 \tag{5}$$

$$z_i + x_{ij} \leq 1 \quad \forall k, \quad q_{ij}^k = 1 \tag{6}$$

$$y_i + \sum_{(j:q_{ji}^k=1)}^I x_{ji} \leq 1 \quad i = 1, \ldots, I \tag{7}$$

$$z_i + \sum_{(j:q_{ij}^k=1)}^I x_{ij} \leq 1 \quad i = 1, \ldots, I \tag{8}$$

$$x_{ij} \in 0, 1 \quad \forall k, \quad q_{ij}^k = 1 \tag{9}$$

$$y_i, z_i \in 0, 1 \quad i = 1, \ldots, I \tag{10}$$

In this model, we link two activities based on the information from the existing process variants. The summation of the number of adjacent links among all of the process variants is denoted as c_{ij}, where $c_{ij} = \sum_{\left(k:q_{ij}^k=1\right)} q_{ij}^k$. This determines the cost of creating a link between a_i and a_j. When the constraints are satisfied, we minimize the multiplication of the integer values of possible link (x_{ij}) and negative cost ($-c_{ij}$) of links a_i and a_j. To avoid unexpected links, we multiply $(1 - x_{ij})$ by the cost. In other words, in order to maximize the sum of proximity scores among process variants, the objective functions have to be minimized (2). Constraints (3) and (4) impose the condition that there is only one start (y_i) activity and one end activity (z_i) in a process reference model. Constraints (5) and (6) guarantee that there are no immediate predecessors to the start activity and no immediate successors of the end activity, respectively. Constraints (7) and (8) determine that there should be at least one path following the start activity and preceding the end activity, respectively. The adjacent relationship of activity a_i and a_j is denoted as x_{ij} with a binary value element, as shown by constraint (9). Constraint (10) reflects the fact that the start/end activity has a binary value element.

By using the graph example in Figure 2 and applying the mathematical approach, we obtain the result from LINGO, shown in Figure 3. The process provides us with insight into the new reference process. The safety property, as a part of soundness property,

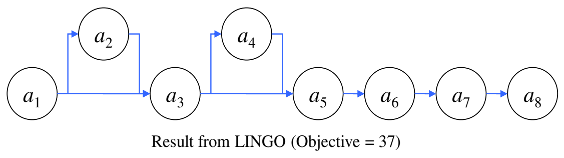
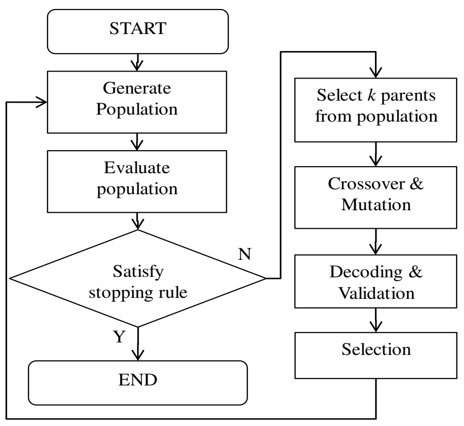


FIGURE 3. Result from LINGO

has been considered in the IP constraint. However, behavior in between start and end activities may hold some irrelevant properties. For example, the result is considered to be an invalid process, since activity a_3 (financial check activity) has never been experienced as a merge activity. Therefore, the present study set about solving the validity problem using a GA approach. The randomness of GA may aid the search for the best chromosome to represent a valid process that achieves the optimum solution.

3.3. **Genetic algorithm procedure.** The structure of the GA procedure is shown in Figure 4. Detailed descriptions of each step are provided thereafter.



Parameter	Value
Generations	100
Population size	10
Selection	Roulette wheel & elitist
Crossover	Uniform
Crossover rate	80%
Mutation	Swapping
Mutation rate	10 %

FIGURE 4. Genetic algorithm structure and parameters

3.3.1. *Initialization.* In order to create a random initial population, we first retrieve related process variants from the repositories. Afterwards, we collect the relevant information such as activity adjacent constraints and the number of ingoing and outgoing edges based on the existing process variants. Let us denote $\max_k(inDegree(a_i))$, $\min_k(inDegree(a_i))$, $\max_k(outDegree(a_i))$ and $\min_k(outDegree(a_i))$, where $k \in K$ is the maximum of the ingoing edges, the minimum of the ingoing edges, the maximum of the outgoing edges and the minimum of the outgoing edges of the i-th activity among the k-th process variants in the repository, respectively. This constraint limits an activity to have either ingoing or outgoing edges without any prior experience of it among process variants. A later Subsection 3.3.4 will discuss in more detail of the activity adjacent constraint and the maximum and minimum of the ingoing and outgoing edges on each activity. An initial population of processes can be randomly generated based on the constraint developed from the retrieval of process variants knowledge from the repositories. Each initialized process will be encoded as explained in the following section.

3.3.2. *Encoding.* Each solution is an individual in a population, and the encoding of each individual is based on the precedence relation between activities. We can see the structure of the chromosome in Figure 5. The first and the last part are the assignment of the start activity and the end activity, respectively. The middle part is the possible precedence relation between the two activities. The length of each chromosome is equal to two times the total number of activities ($2 \times I$), representing start and end activities, added to the combinatorial problem of the directed relations of two activities ($I \times (I-1)$).

An example of encoding from process variant (p_1) to an individual chromosome is presented in Figure 6. It is apparent that a_1 is a start activity and a_8 is an end activity. The middle part shows all of the adjacent relationships between the two activities in p_1.

y_1	...	y_I	x_{12}	x_{13}	…	x_{1I}	x_{21}	x_{23}	…	x_{2I}	…	…	$x_{(I-2)I}$	$x_{(I-1)I}$	z_1	...	z_I

FIGURE 5. Representation of chromosome

Individual chromosome:
100000001110000000100000010000001000000011000000010000001000000000000001

y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{21}	x_{23}	x_{24}	x_{25}	x_{26}	x_{27}	x_{28}	x_{31}	x_{32}	x_{34}	x_{35}	x_{36}	x_{37}	x_{38}
1	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0

x_{41}	x_{42}	x_{43}	x_{45}	x_{46}	x_{47}	x_{48}	x_{51}	x_{52}	x_{53}	x_{54}	x_{56}	x_{57}	x_{58}	x_{61}	x_{62}	x_{63}	x_{64}	x_{65}	x_{67}	x_{68}
0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1

x_{71}	x_{72}	x_{73}	x_{74}	x_{75}	x_{76}	x_{78}	x_{81}	x_{82}	x_{83}	x_{84}	x_{85}	x_{86}	x_{87}	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

FIGURE 6. Encoding from process variant p_1 (see Figure 1) to individual chromosome

3.3.3. *Crossover and mutation.* Here, we describe the genetic operators that are used in our GA procedure. First, the crossover operator is designed. At each step of the GA procedure, for further generation of offspring, we select parents from the population of the generation. We use a uniform crossover to combine two parents, which process is illustrated in Figure 7. We also design a swap mutation to modify the activity position in an individual for a process variant. A combination roulette-wheel-and-elitist method is adopted for the selection procedure.

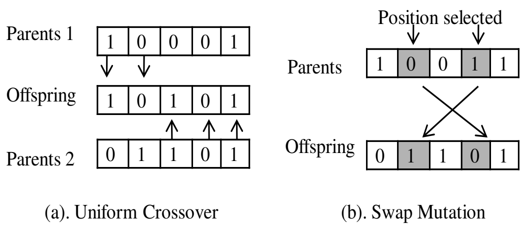


FIGURE 7. Crossover and mutation operation

3.3.4. *Decoding and validation.* The illustration in Figure 8 shows the method of decoding from a chromosome to a process. Due to the random genetic processes (crossover and mutation) that might produce an invalid result, it is necessary to verify the process for further generation. We apply the soundness properties of business processes to verify the process well-formedness. For a business process model to be sound, three properties are required [9]:

1) For every activity a_j that is reachable from a_i, there exists a sequence to the next activity until a preferred end activity a_E is reached.

2) A preferred end activity a_E is the activity that should be reachable from a start activity a_S.

3) There is no activity that is never processed in any execution of the model (there are no needless elements).

Individual chromosome:
100000001000000010000000110000001000000010000000100000001000000000000001

y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{21}	x_{23}	x_{24}	x_{25}	x_{26}	x_{27}	x_{28}	x_{31}	x_{32}	x_{34}	x_{35}	x_{36}	x_{37}	x_{38}
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0

x_{41}	x_{42}	x_{43}	x_{45}	x_{46}	x_{47}	x_{48}	x_{51}	x_{52}	x_{53}	x_{54}	x_{56}	x_{57}	x_{58}	x_{61}	x_{62}	x_{63}	x_{64}	x_{65}	x_{67}	x_{68}
0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0

x_{71}	x_{72}	x_{73}	x_{74}	x_{75}	x_{76}	x_{78}	x_{81}	x_{82}	x_{83}	x_{84}	x_{85}	x_{86}	x_{87}	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

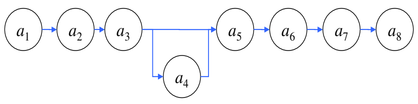


FIGURE 8. Decoding from chromosome to process variant

The Figure 8 illustration shows that any activity a_j that is reachable from a_i has a sequence to end activity a_E (a_8 is reachable from any earlier activities ($a_1 - a_5$)). There also exists a path from start activity a_S (a_1) to end activity a_E (a_8). Since the process model has no iterative path, we consider that each activity in the process has a chance to be executed. Since GA is based on randomness, we need a mechanism to guarantee the validation of a business process while we decode a chromosome into a business process. In order to satisfy the process validation, we use three boundaries: a precedence matrix, the maximum in/out degree and the minimum in/out degree (Figure 9). A precedence matrix is a binary value matrix where value 1 denotes that there exists at least one link among process variants in the repository. The maximum and minimum in/out degrees represent the maximum and minimum number of ingoing and outgoing edges among process variants in the repository, respectively. Both boundaries are important to the creation of a valid process.

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
a_1	0	1	1	1	0	0	0	0
a_2	0	0	1	1	1	0	0	0
a_3	0	0	0	1	1	1	0	0
a_4	0	0	0	0	1	1	0	0
a_5	0	0	0	0	0	1	1	0
a_6	0	0	0	0	0	0	1	1
a_7	0	0	0	0	0	0	0	1
a_8	0	0	0	0	0	0	0	0

Max Outdegree

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
3	2	2	1	2	1	1	0

Min Outdegree

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
1	0	1	1	0	1	0	0

Max Indegree

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
0	1	1	2	3	2	1	2

Min Indegree

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
0	0	1	1	0	1	0	1

FIGURE 9. Precedence relation matrix (left figure) for creating valid process and maximum and minimum of outdegree and indegree of each activity (right figure), based on Figure 1

The randomness of the genetic process might result in a process without a start/end activity or a start/end activity that is greater than one. There are some cases in which an invalid process can be created (see Figures 10(a) and 10(b)). To create a valid process, we developed the *start_end* algorithm to certify that there will be only one start/end activity that satisfies the constraint (*inDegree* = 0 and *outDegree* = 0 for the start and end activities, respectively). Figure 11 shows the *start_end* algorithm assigning and validating the start and end activities. Figure 10 shows some other cases of invalid processes. Case 3 illustrates a process that is invalid given the fact that there is a link from a_1 to a_5 that has no precedence relation (as shown in Figure 9). The other case, case 4, shows that there is a link from a_1 to a_3 with previous precedence relation information. However, it is also considered to be an invalid process, since a_3 has a maximum *inDegree* equal to 1, though a_1 has the possibility of being a split activity (the maximum *outDegree* is greater than 1). Thus, we need to delete the link that creates an invalid process.

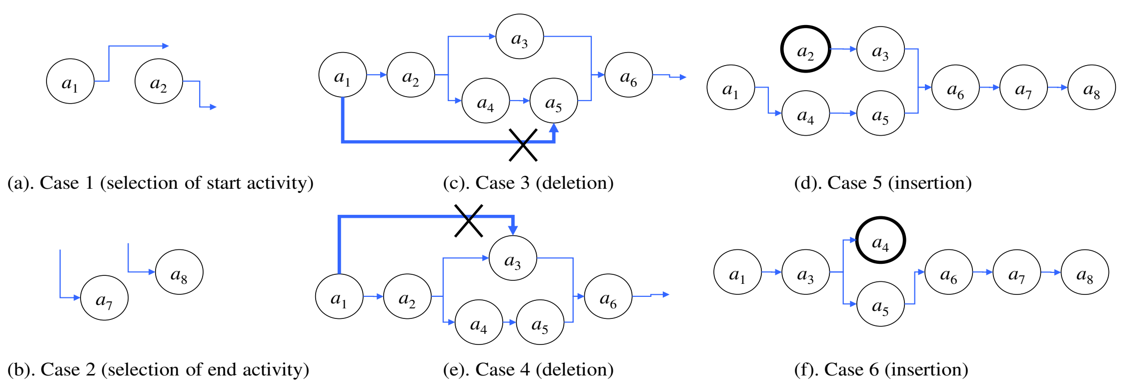


Figure 10. Randomness cases for genetic process

There can be other cases in which some activities are not connected. To establish a valid business process, we establish that there is only one start activity and one end activity, as denoted by the *start_end* algorithm. Therefore, case 5 and case 6 violate the business process soundness property, since there is more than 1 activity for which the in/out degree values are equal to 0. The *start_end* algorithm is only an attempt to assign the right start/end activity without considering any link activity with an in/out degree equal to 0. In case 5, both activities a_1 and a_2 have an *inDegree* equal to 0. Additionally, in case 6, both activities a_4 and a_8 have an *outDegree* equal to 0. With regard to this problem, our approach creates links based on the cost value to minimize the objective function.

The *valid_process* algorithm (Figure 12) was developed to generate a valid process based on the soundness properties. There are two functions included in the *valid_process* algorithm. The first is link deletion, and the second is link insertion. The deletion function deletes any link with no implication for the definition of a core process, such as when there is no respective link in the process repository or when the split-join behavior shows that either the ingoing or the outgoing link has more than the maximum constraint. Meanwhile, the insertion function connects any activity without either an ingoing or an outgoing edge.

The total time complexity of decode mechanism (both *start_end* and *valid_process* algorithm) of validity process is $O(N^2)$ where $N = |A|$. The time complexity of *start_end* algorithm is $O(N)$ since it performs the N times of total activities. To find a valid link between two activities, the *valid_process* algorithm has $O(N^2)$ in calculating each of activity link on both deletion and insertion process. Figure 13 shows the GA results. As explained previously, the result from LINGO is invalid (the objective value = 37) since

```
Algorithm start_end ($p^k$)
Input : a chromosome of a process $p^k$ ($y_1$, …, $y_I$ ; $z_1$, … $z_I$)
Output : $a_S$ and $a_E$ in $p^k$
Begin
/* Selection of start activity */
  int i, start ⟵ 0;
  WHILE ((start <1) || (i<=I))      /* select the first $y_i$ that is equal to 1 */
    IF ($y_i$=1) THEN {
      IF (inDegree($y_i$)=0) THEN
        start++; return $a_S$ ⟵ $y_i$ ;}
      i++;
  END WHILE
  IF ($a_S$ =null) THEN
      FOR each $a_i \in A^k$ DO    /* select the first $a_i$ for which inDegree($a_i$)=0 */
        IF (inDegree($a_i$)=0) THEN
          return $a_S$ ⟵ $y_i$ ;
      END FOR

/* Selection of end activity */
  int end ⟵ 0; i ⟵ I ;
  WHILE ((end <1) || (i >=1))      /* select the last $z_i$ that is equal to 1 */
    IF ($z_i$=1) THEN {
      IF (outDegree($z_i$)=0) THEN
        end++; return $a_E$ ⟵ $z_i$ ;}
      i--;
  END WHILE
  i ⟵ I ;
  IF ($a_E$ =null) THEN
      WHILE (i >= 1)    /*select the last $a_i$ for which outDegree($a_i$)=0 */
        IF (outDegree($a_i$)=0) THEN return $a_E$ ⟵ $z_i$ ;
        i-- ;
      END FOR
End.
```

FIGURE 11. *start_end* algorithm

activity a_3 has never been a merge activity. By using the *valid_process* algorithm to check the process path (the GA decoding verification method), we can produce the result shown in Figure 13, with the objective function greater than the result of LINGO (Figure 3). Although the GA results in a greater objective value, the process model fits with the soundness and validation of the process variants properties.

4. Experimental Results and Discussions.

We conducted experiments as presented in Section 4.1. This section demonstrates the experiment results with regard to best fitness value and execution time. Section 4.2 discusses qualitative analysis of experiment results, along with its limitation and directions for further research.

4.1. Experimental results.

In the present study, we conducted experiments in the mathematical model using LINGO combined with the Java programming language (Java-APS) [22]. The resulting Java-APS inputs an APS value into LINGO. That is, by using Java-APS, we retrieved topological information on the BP model and generated the APS value in a text file for LINGO computation. Afterward, LINGO computed the combinatorial optimization problem based on those APS values generated by Java-APS. With regard to GA, we ran the experiments using the parameters in Figure 4 and retrieved process variants from the repository with various numbers of activities, as shown in Table 1 (avg. # of activities). The objective value and execution time results for both LINGO and GA are listed in Table 1.

```
Algorithm valid_process ($p^k$)
Input : a chromosome of a process $p^k$
Output : valid process $p^k$
Begin
/* Link Deletion */
  FOR each $a_i \in A^k$ DO // $A^k$ is a set of activities in $p^k$
    FOR each $a_j \in A^k$, $a_j \neq a_i$
      IF ($q_{ji}^k = 1$) THEN
        IF (($c_{ji}$=0) || ($outDegree(a_j) > \max_k(outDegree(a_j))$)) THEN
          $L^k \leftarrow L^k - \{l_{ij}\}$; $outDegree(a_j)$--; $inDegree(a_i)$--; // $L^k$ is a set of links in $p^k$
      IF ($q_{ij}^k = 1$) THEN
        IF (($c_{ij}$=0) || ($inDegree(a_j) > \max_k(inDegree(a_j))$)) THEN
          $L^k \leftarrow L^k - \{l_{ij}\}$; $outDegree(a_i)$--; $inDegree(a_j)$--;

/* Link Insertion */
   FOR each $a_i \in A^k$ DO $link\_insert(a_i)$;
       IF (($inDegree(a_i)$==0) && ($a_i \neq a_S$))THEN
       FOR each $a_j \in A^k$ , $a_j \neq a_i \wedge (a_i ,a_j) \in L^k$
         IF (($outDegree(a_j)$>0) && ($outDegree(a_j) < \max_k(outDegree(a_j))$)) THEN
           IF ($c_{ji} >$ max_in($a_i$)) THEN max_in($a_i$) = $c_{ji}$;
       END FOR
       FOR each $a_j \in A^k$ , $a_j \neq a_i$
           IF ($c_{ji}$ = max_in($a_i$)) THEN $L^k \leftarrow L^k + \{l_{ij}\}$;
             $outDegree(a_j)$++; $inDegree(a_i)$++;
     END FOR
     IF (($outDegree(a_i)$==0) && ($a_i \neq a_E$)) THEN
       FOR each $a_j \in A^k$ , $a_j \neq a_i \wedge (a_i ,a_j) \in L^k$
         IF (($inDegree(a_j)$>0) && ($inDegree(a_j) < \max_k(inDegree(a_j))$)) THEN
           IF ($c_{ij} >$ max_out($a_i$)) THEN max_out($a_i$) = $c_{ij}$;
       END FOR
       FOR each $a_j \in A^k$ , $a_j \neq a_i$
         IF ($c_{ij}$ = max_out($a_i$)) THEN $L^k \leftarrow L^k + \{l_{ij}\}$;
         $outDegree(a_i)$++; $inDegree(a_j)$++;
       END FOR
   END FOR
  End.
```

FIGURE 12. *valid_process* algorithm

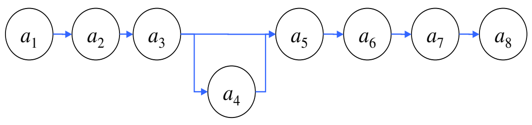


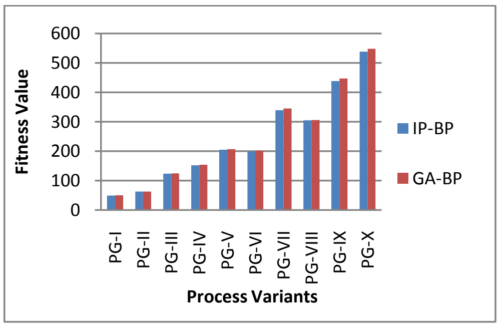
FIGURE 13. Reference process model as result of GA (fitness = 38)

4.2. **Discussions.** The IP problem, which has lesser computational time than GA approach, can result invalid BP reference model. First, it has no consideration with regard to activity properties, e.g., split and merge activity. Second, it guarantees only the safety property. The safety property always guarantees that a process will always run to completion. However, it does not guarantee that in-between start and end activity of BP reference model correspond to existing properties of process variants, referred to validity property. Thus, GA approach attempts to solve these problems. Existing approach of generating reference model [2-5] were ineffective when there is no information with regard to process configurations. Since it is too complex to collect information about the configurations of large numbers of process variants, it is particularly necessary to have functions as presented at this present study. By retrieving all process variants without any

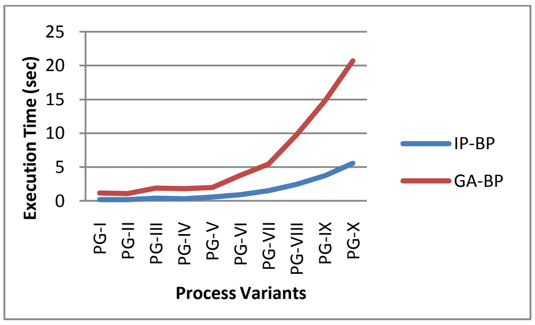
TABLE 1. Experimental results

Process Group	Avg. # of Act	IP-BP		GA-BP					
		Obj. Value	Exec. Time	Best	Avg. Fitness	St. Dev	Min.	Max.	Exec. Time
PG-I	6.1	49	0.17	50	50.18	0.064	50	79	1.19
PG-II	10.4	63	0.17	63	63.69	0.098	63	145	1.09
PG-III	14.8	124	0.39	125	125.44	0.055	125	176	1.89
PG-IV	19.0	152	0.31	154	155.05	0.104	154	274	1.81
PG-V	23.2	205	0.56	207	208.63	0.079	207	381	2.00
PG-VI	28.6	199	0.92	202	203.88	0.198	202	412	3.75
PG-VII	33.1	339	1.49	345	346.72	0.205	345	552	5.45
PG-VIII	38.3	305	2.45	306	309.40	0.330	306	696	9.77
PG-IX	43.0	438	3.72	447	449.95	0.245	447	779	14.75
PG-X	47.2	538	5.56	548	551.23	0.222	548	903	20.69

information of process configuration, it may help general organizations, which apply GA approach, to solve both problems mentioned above. The random search mechanism using GA can guarantee the soundness of BP reference model with some validation algorithms to rearrange the result of crossover and mutation process. The usage of three boundaries, a precedence matrix, the maximum in/out degree and the minimum in/out degree, is considered as a measure to achieve the soundness properties [9]. The fitness values of GA were slightly greater than the IP problem, however, it can represent a sound and a more compliant process. It is recommended that when the numbers of activities increase, such heuristics be applied to result a better process model with a rational computational time. We left this issue to our further research.



(a). Graphic of Fitness Functions comparison



(b). Graphic of Execution Time Comparison

FIGURE 14. Graphic of (a) fitness values comparison and (b) execution time comparison

There are two major deficiencies of our main results. First, this proposed method considered directed acyclic graph with exclusion of loop/iterative activity. The existence of loop/iterative activity may hinder the distance calculation between reachable activities. Thus, it is considerable for future work to make a distance measure specifically for loop activity with regard to the probability of execution. Second, indexing on graph abstraction activity mapping is simply based on string edit distance. Semantic accuracy can be obtained not only using topological information of BP but also by considering other attribute such as performers, documents and artifacts. We put both of these issues for our future works.

5. **Conclusions.**

This paper presents a new method of finding a process reference model. In order to generate a valid reference process, we enhanced the math formulation of the IP problem, which is a combinatorial optimization problem, with the GA procedure. Although the GA procedure has a higher cost on execution time and a greater fitness value than the IP problem, it resulted in a more valid process compared with the IP approach. We applied the GA procedure to large problems and successfully acquired reference models.

Problems associated with the use of IP formulation, such as the selection of the start/end activity and the selection of activity links, were resolved using the GA procedure with the *start_end* and *valid_process* algorithms. Selection of a start/end activity based on the in/out degree of links produced, in comparison with the IP approach, a valid reference process. The deletion and insertion of links during the decoding procedure, which are not parts of the IP approach, are considered to be important to generating a valid reference process. Even though the GA fitness value is greater than in the IP approach, the GA reference process satisfies the validity and soundness properties necessary to represent the corresponding process variants in the repository. In other words, the presentation limitations of the mathematical formulation and the possibility of an unguaranteed valid process were resolved by using the GA procedure. Due to the cost of GA execution time, there remain issues for further research. Other meta-heuristic approaches might hold some advantages for reducing the execution time.

The process reference model derived by our approach can be utilized for various purposes. First, it can be a process template for certain process variants. Second, it can deal with the process reuse issue. Hence, our approach can be a robust decision making tool for convenient process modeling by novice designers.

Acknowledgment. This work was supported by a Korea Research Foundation Grant funded by the Korean Government (MOEHRD) and the Regional Research Universities Program/Research Center for Logistic Information Technology.

REFERENCES

[1] A. Hallerbach, T. Bauer and M. Reichert, Managing process variants in the process lifecycle, *Proc. of the 10th International Conference on Enterprise Information Systems*, pp.154-161, 2008.

[2] C. Li, M. Reichert and A. Wombacher, Discovering reference model by mining process variants using a heuristic approach, *Proc. of BPM the 7th International Conference, LNCS*, Ulm, Germany, vol.5701, pp.344-362, 2009.

[3] J. M. Kuster, J. Koehler and K. Ryndina, Improving business process models with reference models in business-driven development, *BPM Workshops, LNCS*, vol.4103, pp.35-44, 2006.

[4] O. Holschke, P. Gelpke, P. Offermann and C. Schropfer, Business process improvement by applying reference process models in SOA – A scenario-based analysis, *Multikonferenz Wirtschaftsinformatik*, 2008.

[5] P. Fettke, P. Loos and J. Zwicker, Business process reference models: Survey and classification, *BPM Workshops, LNCS*, vol.3812, pp.469-483, 2006.

[6] D. Hong, H. Ling and Y. Li, A novel theory of business process virtualization on E-commerce, *ICIC Express Letters*, vol.4, no.2, pp.381-388, 2010.

[7] X. Wang, K. Zhang, D. Yang and X. Zhang, Simulation research on resource deployment of logistics distribution system under mobile business environment, *ICIC Express Letters*, vol.4, no.2, pp.435-440, 2010.

[8] J. Bae, T. Lee, H. Bae and K. Lee, Process reference model generation by using graph edit distance, *Korean Institute Industrial Engineering Conference*, Korean, 2010.

[9] W. M. P. van der Aalst, Workflow verification: Finding control-flow errors using petri-net-based techniques, *Business Process Management, LNCS*, no.1806, pp.161-183, 2000.

[10] A. L. Alves de Medeiros, *Genetic Process Mining*, Ph.D. Thesis, Eindhoven Technical University, 2006.

[11] C. K. Chang, M. J. Christensen and T. Zhang, Genetic algorithms for project management, *Annals of Software Engineering*, vol.11, pp.107-139, 2001.
[12] M. Gen, R. Cheng and L. Lin, *Network Models and Optimization Multiobjective Genetic Algorithm Approach*, Springer-Verlag, London, 2008.
[13] D. E. Goldberg, *Genetic Algorithm in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, 1989.
[14] R. L. Wang, A genetic algorithm for subset sum problem, *Neurocomputing*, vol.57, pp.463-468, 2004.
[15] Z. Q. Chen, R. Wang and K. Okazaki, An efficient genetic algorithm based approach for the minimum graph bisection problem, *International Journal of Computer Science and Network Security*, vol.8, no.6, pp.118-124, 2008.
[16] D. Kim, N. Lee, S. Kan, M. Cho and M. Kim, Business process version management based on process change patterns, *International Journal of Innovative Computing, Information and Control*, vol.6, no.2, pp.567-575, 2010.
[17] G. Zhou, Y. He and P. Yu, Modelling workflow patterns based on P/T nets, *International Journal of Innovative Computing, Information and Control*, vol.1, no.4, pp.673-684, 2005.
[18] J. Jung, J. Bae and L. Liu, Hierarchical clustering of business process models, *International Journal of Innovative Computing, Information and Control*, vol.5, no.12(A), pp.4501-4511, 2009.
[19] M. Kim and D. Kim, Fault-tolerant process debugger for business process design, *International Journal of Innovative Computing, Information and Control*, vol.6, no.4, pp.1679-1687, 2010.
[20] I. Levenshtein, Binary code capable of correcting deletions, insertions and reversal, *Cybernetics and Control Theory*, vol.10, no.8, pp.707-710, 1966.
[21] B. N. Yahya, H. Bae and J. Bae, Process design selection using proximity score measurement, *BPM 2009 Workshop, LNBIP*, Germany, vol.43, pp.330-341, 2009.
[22] R. Sedgewick, *Algorithm in Java, Part 5 Graph Algorithm*, Addison-Wesley, Pearson Education, 2004.