A Petri Nets Based Generic Genetic Algorithm Framework for Resource Optimization in Business Processes

Yain-Whar Si[1,*], Veng-Ian Chan[1], Marlon Dumas[2], Defu Zhang[3]

[1] *Department of Computer and Information Science, University of Macau, Macau*

[2] *University of Tartu, Estonia*

[3]*Department of Computer Science, Xiamen University, China*

fstasp@umac.mo, debbyc@icm.gov.mo, marlon.dumas@ut.ee, dfzhang@xmu.edu.cn

Abstract

Business process simulation (BPS) enables detailed analysis of resource allocation schemes prior to actually deploying and executing the processes. Although BPS has been widely researched in recent years, less attention has been devoted to intelligent optimization of resource allocation in business processes by exploiting simulation outputs. This paper endeavors to combine the power of a genetic algorithm (GA) in finding optimum resource allocation scheme and the benefits of the process simulation. Although GA has been successfully used for finding optimal resource allocation schemes in manufacturing processes, in this previous work the design of these algorithms is ad hoc, meaning that the chromosomes, crossover and selection operators, and fitness functions need to be manually tailored for each problem. In this research, we pioneer to design and implement a Petri Nets based Generic Genetic Algorithm (GGA) framework that can be used to optimize any given business processes which are modelled in Color Petri Nets (CPN). Specifically, the proposed GGA framework is capable of producing an optimized resource allocation scheme for any CPN process model, its task execution times, and the constraints on available resources. The effectiveness of the proposed framework was evaluated on archive management workflow at Macau Historical Archives and an insurance claim workflow from an Australian insurance company. In both case studies, the framework identified significantly improved resource allocation scheme relative to the one that existed when the data for the case studies were collected.

Keywords: Colored Petri Nets, genetic algorithm, resource optimization, business process simulation.

1. INTRODUCTION

Business Process Simulation (BPS) [1] is an established technology for quantitative analysis of business processes. BPS has been widely used for analyzing business processes in a wide range of domains. For

*Corresponding author
Yain-Whar Si (fstasp@umac.mo), Faculty of Science and Technology, University of Macau, E11-4022, Avenida da Universidade, Taipa, Macau, China, Tel.: +853 8822-4454

instance, BPS is used to redesign a public administration process in Italy to improve its efficiency in providing services to the local community [2]. BPS is also used in supply chain management to investigate the adoption of RFID systems [3] as well as to improve the internal processes accounting firms in Netherlands [4].

The essential benefit of BPS is that it enables analysts to estimate key performance indicators (KPI), such as *cycle time* and *cost-per-execution*, prior to actually deploying and executing the processes in a real environment. BPS is typically used to analyze and compare multiple resource allocation schemes for a business process in order to select the one that strikes the best tradeoff between multiple KPIs. By *resource allocation scheme*, we mean an assignment of resources to each task in the process model.

However, while current BPS technology allows analysts to manually explore resource allocation schemes, it leaves them the burden of finding the right scheme for their needs. This manual search for an optimal scheme can be quite time-consuming, especially in the case of large and complex process models. In this setting, the contribution of this paper is a Petri Nets based generic framework to automatically optimize resource allocation in business processes. In a nutshell, the proposed framework combines the evolutionary optimization capabilities embodied by Genetic Algorithms (GA) [5] with BPS technology in order to seek resource allocation schemes that improve cost or execution time.

In previous work, GA has already been successfully used for deriving optimal resource allocation schemes for credit requirement management [6], dynamic execution planning for web-services [7], and manufacturing processes [8, 9]. However, in this previous work the design of these algorithms is ad hoc, meaning that the chromosomes, crossover and selection operators, and fitness functions need to be manually tailored for each BPS problem. In contrast, this paper puts forward a Colored Petri Nets (CPN) [10] based Generic Genetic Algorithm (GGA) framework that can be used to optimize any given business process.

Colored Petri Nets (CPN) is a well-established language for modeling and simulating business processes [11]. CPN is based on strong mathematical foundation and they are extensively used for capturing and analyzing models from a wide variety of applications. These applications include finding resource allocation scheme and operation schedule of manufacturing systems [12], scheduling for the semiconductor manufacturing environment [13], controlling traffic lights in congested urban areas [14] [15], finding optimal schedule in Flexible Manufacturing Systems (FMS) [16], diagnosing and managing shop floor for Computer Aided Manufacturing (CAM) [17], and optimization of the execution time for

tasks Grid computing [18]. These applications highlight the distinct advantage offered by CPN in modelling complex processes.

The proposed GGA framework is capable of extracting information regarding available resources, task execution times, and costs from the given business process model to apply the optimization algorithm on those inputs. This capability is achieved by encoding the GA components in a generic Colored Petri Nets (CPN), into which a process model can be plugged in. Specifically, GGA framework is directly implemented in Colored Petri Nets. Although GA has been implemented in various programming languages, to the best of our knowledge, no one has ever implemented GA components directly in CPN. Our work is the first-of-its kinds in business process simulation area. The overview of the interaction between GGA framework and Integrated Development Environment (IDE) for Colored Petri Nets (CPN Tools [19]) is depicted in Figure 1. Note that although IDE for CPN provides functions such as editing, verification, simulating, and performance analysis, we will mainly focus on the latter two functions. In this interaction, GGA framework which is developed in CPN can automatically extracts resource information from the business process which are also modelled in CPN. The extraction of resource information is enabled by an interface which is also implemented as a CPN model. Once the resource information is extracted, the relevant business process is annotated with a particular resource assignment scheme by the GGA framework according to outcomes of the genetic algorithm (GA). The business process together with a particular resource assignment scheme is then transferred to the IDE for simulation. Once the simulation is completed, the results of the simulations is transferred back into GGA for fitness calculation and ranking. This process is repeated for all members of the population in GA. The members of the population are then ranked according to their fitness and after that a new generation is created within the GGA framework based on selection, mutation, and cross-over operators. This process of generating new populations and simulation for fitness calculation will continue until a predefined stopping criterion is reached. At the point, GGA framework returns the business process with the desired optimal resource allocation scheme.

The advantages of directly implementing GA components in CPN are as follows:

- Integrated Development Environments (IDE) for CPN are readily available. For instance, any business processes modeled in CPN can be simulated in CPN Tools (cpntools.org) which is an integrated environment for editing, simulating, and analyzing Colored Petri nets. The key functions provided by CPN Tools for analyzing Colored Petri Nets are depicted in the right-hand area of Figure 1.

- By implementing GA components in CPN instead of using conventional programming languages, process simulation and genetic algorithm can be tightly integrated and executed under the same IDE.
- Since both GA components and process models are implemented in CPN, they can seamlessly interact with each other to share resource information. This situation is depicted in the left-hand area of Figure 1. Since GA components can directly extract resource information from the process model, unlike other BPS approaches, the proposed GA framework can be used to optimize any processes without having to tailor the design of the chromosomes, crossover and selection operators, and fitness functions. In previous applications [6, 7] of GA in BPS, these genetic operators are manually tailored for each business process.
- Process simulation is usually an interactive procedure comprising the steps such as process modeling, selection of appropriate parameters, simulating the configured processes, and the post-analysis of the simulation result involving human experts. By controlling the simulation with a genetic algorithm, we can fully automate the finding of optimum resource allocation scheme without ever leaving the CPN environment.

The main contribution of our work is the design of a Petri Nets based generic Genetic Algorithm framework into which any process models (also encoded in CPNs) can be plugged in for resource optimization. This situation is depicted in the left-bottom corner of Figure 1. In this illustration, any process models designed in CPN can be directly plugged in to the GGA framework for resource optimization.

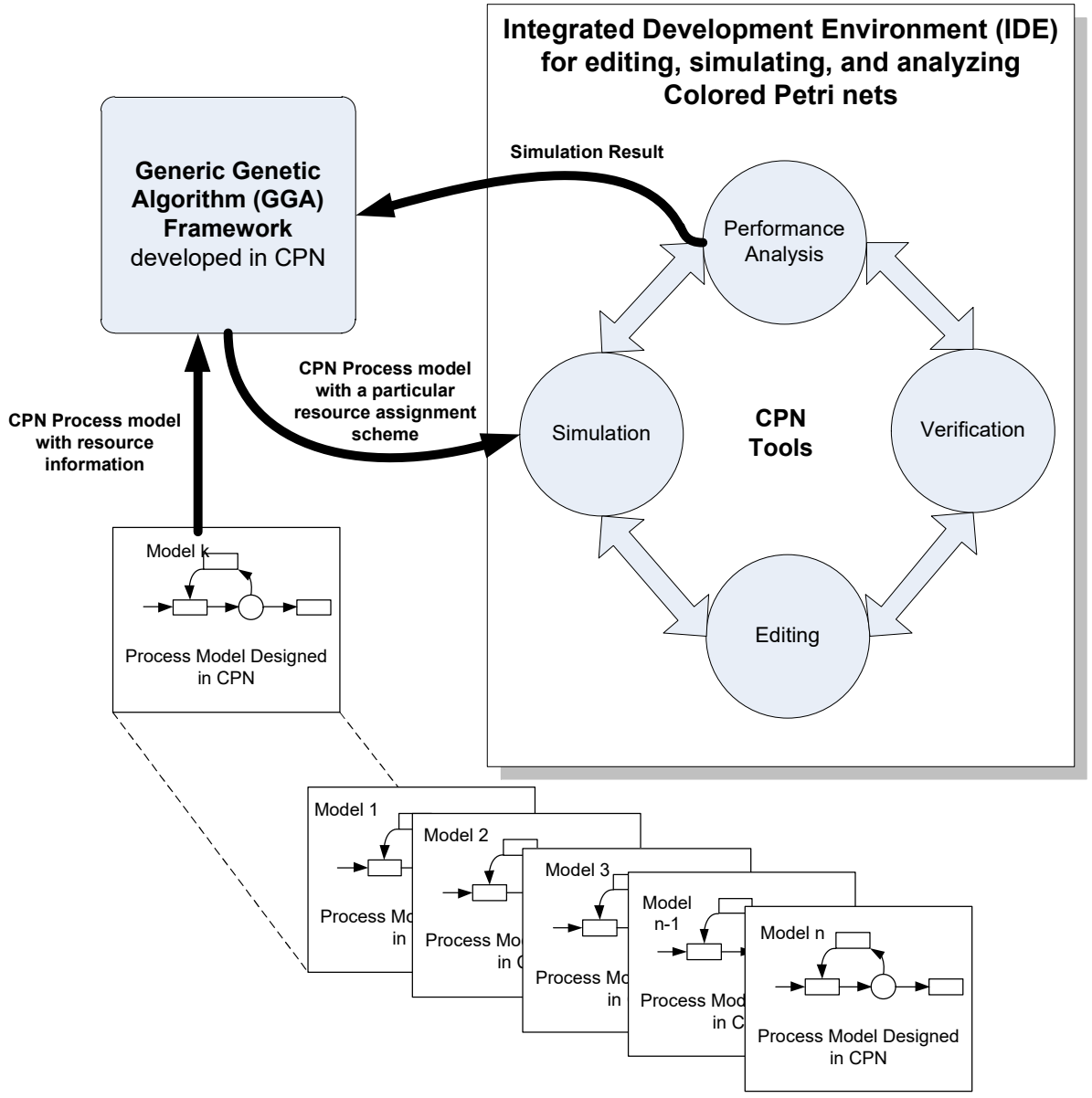

Figure 1. Interaction between Genetic Algorithm (GA) components which are directly implemented in CPN and Integrated Development Environment (IDE) for Colored Petri nets (CPN Tools)

To achieve this capability, we design four critical components in Color Petri Nets; (a) the whole genetic algorithm, (b) tokens for modeling chromosomes, (c) tokens for carrying the parameters for controlling the simulation, and (d) an interface based on places and transitions. The interface is designed in such a way that it allows:

- automatic extraction of task execution times and the constraints on available resources from the CPN process models,

- accepting of tokens as chromosomes carrying parameters for various resource assignment schemes for simulation,
- accepting of tokens which include the parameters for controlling the simulation, and
- transferring of simulation results to the genetic algorithm for fitness evaluation in each generation.

This paper is the extended version of [20]. With respect to [20], we have included an additional larger-scale case study (the insurance case study) that provides further evidence of the potential applicability and benefits of the proposed framework. The paper is structured as follows. In Section 2, we review related work. In Section 3, the proposed generic framework is introduced. In Sections 4 and 5, a case study from a Macau public organization is presented. In Section 5, and 7, a second case study of a "teleclaims" process from an Australian insurance company is detailed. In section 8, we conclude the paper and discuss future work.

2. RELATED WORK

Simulation-based analyses of business processes have been reported in [1, 21-23]. BPS is not only a cost-effective way of analyzing and improving business process, but also a promising approach for verifying the correctness and efficiency of the process models. Specifically, BPS is a well-established approach to predict the impact of certain changes in design of the process models [24]. In [21], for example, a virtual supply chain simulation model is used to analyze the impact on performance for the strategic decisions such as number of plants, the transportation modes and the relocation of warehouses. In [22], Bisogno et al. demonstrated that BPS can be used to analyze a standardized patient arrival and treatment process in an orthopedic-emergency room of a public hospital. In the context of Enterprise Engineering, BPS is used to support business process reengineering (BPR) [1]. BPS is also used for the design of sustainable Product Service Systems (PSS) in which both services and products are offered to the customers [23].

BPS has also been shown to be an effective technique for analyzing the effectiveness of escalation strategies aimed at avoiding or mitigating deadline violations in business processes [25-27]. Examples of escalation actions are for example to give higher priority to a case in order to speed-up its execution, or to negotiate an extended deadline with the customer. In [26], van der Aalst et al. analyze several deadline escalation strategies using BPS. In their approach, escalation strategies are evaluated from three perspectives: the process perspective of using alternative path selection, the data perspective of using data degradation, and the resource perspective of using resource redeployment. In [28], four escalation

strategies from [25] and [26] are evaluated from both a temporal (cycle time) and a cost perspectives. Similar study on seven escalations strategies is also reported in [27].

The above are just a sample of a larger body of studies on the use of BPS to manually analyze alternative process models, or alternative resource allocation schemes for a given process model. This paper however goes beyond the simulation of specific alternatives, aiming instead at automatically identifying optimal or near-optimal allocation schemes for a given process model based on simulation. In this respect, the research reported in this paper is closer to [6, 29], where GA and simulation technology are jointly used to derive optimal resource allocation schemes for business processes. However, these algorithms need to be manually tailored for each BPS problem, whereas in this paper we aim at providing a generic framework. The research reported here is also related to work on designing scheduling systems for manufacturing processes, control systems, and grid computing. During the design phase, the performance and correctness of these systems are often tested with Petri Nets-based simulation models. Petri Nets allow modelling of concurrency and synchronization in manufacturing systems. Petri Nets provide diagrammatic tools similar to the state transition diagrams in modelling system behaviors. For instance, in [30], Li et al. used transition–timed Petri Nets (TTPN) and an admissible heuristics function to find optimal scheduling strategies for flexible manufacturing systems (FMS). The proposed heuristic function considers available time of shared resources such as subparts and machines during scheduling. The heuristic functions allow exploration of only the necessary part of the reachability tree generated by the TTPN. However, scheduling with TTPN with admissible heuristic function proposed in [30] assumes that each machine can be used for only one transition in a job. In TTPN, transitions are used to represent operations (tasks) in each job. In contrast to their approach, the proposed GGA framework designed in Colored Petri Nets (CPN) allows complex assignment of resources to the transitions.

Petri Nets and GA are used to solve various optimization problems. For instance, Timed Petri Net (TPN) model and GA are used to improve the efficiency of manufacturing processes [31]. In [32], a Petri Net is used as a fitness function for GA to optimize the urban traffic based on vehicles positions and relevant roads' infrastructures. Petri Nets and GA are also used to optimize production scheduling in Job Shop Manufacturing Systems [33-36]. Scholastic Petri Nets and GA are also used to optimize the maintenance scheduling problems [37]. In wafer fabrication, Queueing-Petri Net (Q-PN) and GA are used to find the optimal scheduling policy in a semiconductor manufacturing system [38]. In these works, GA components are implemented in programming languages which are different from the ones used for Petri Nets.

Colored Petri Nets (CPN) and GA are also combined to solve a number of optimization problems. For example, in [12], a system based on CPN and a GA is used to find resource allocation scheme and operation schedule of manufacturing systems. In [13], CPN and GA are used for solving the generalized scheduling problems arising from the semiconductor manufacturing environment. Li et al. [39] combined Petri Nets and GA to solve the hull balance decision problem in damaged ships. In their approach, Petri nets are used to model the ship anti-flooding decision process and the heuristic color genetic algorithm to generate anti-flooding decision plans. Dezani et al. [14] proposed a method by combining CPN and GA to control traffic lights in congested urban areas. Their objective is to reduce the number of vehicles in the congested areas. In [16], GA and CPN are combined to find the optimal schedule in Flexible Manufacturing Systems. A decision support system based on fuzzy CPN with GA was also proposed in [17]. The proposed system was designed to solve the problem of diagnosing and managing shop floor for Computer Aided Manufacturing. In contrast to our approach, the GA components used in [12-14, 16, 17, 39] are not directly implemented in Colored Petri Nets.

CPN and other machine learning methods are also combined to solve a number of engineering problems. In the context of transportation engineering, Barzegar et al. [15] combined CPN with Fuzzy logic to efficiently control the traffic signals. In their approach, a variable-structure Learning Automata (LA) [40] is used to adjust the membership functions of the fuzzy reasoning system. In Grid computing, Shojafar et al. [18] also proposed an algorithm to optimize the execution time for tasks based on stochastic Petri Nets and S-model Learning Automata. LA is a kind of reinforcement learning where the decision-making unit selects the current action based on the past interactions with the environment. In other words, the LA is capable of adaptively learning the optimal action from a random environment when probability or rewards are unknown. Due to its adaptive decision making capability in unknown random environment, LA and GA are combined to avoid falling into local optimal solutions in NP problems [41].

3. Petri Nets based Generic Genetic Algorithm Framework for Resource Optimization

An overview of the proposed GGA framework is given in Figure 1. Note that the entire GGA framework given in Figure 1 was modelled Color Petri Nets (CPN). First, the user imports the CPN process model to the GGA framework for analysis. Next, the GGA framework uses the resource information of the model to form the members of the population for the 1st generation. Each chromosome represents a potential resource allocation scheme, that is, a function that assigns each task in the model to a set of resources. For each chromosome, one round of simulation is performed to calculate the degree of fitness. Chromosome fitness is calculated based on the workflow completion time and the total cost. These chromosomes are

then ordered according to the fitness and genetic operators such as selection, crossover, and mutation are applied to the chromosomes to form new members for the next generation. The overall process iterates until the change in the fitness of the best members in the population for several consecutive generations is less than a predefined threshold. The resource allocation scheme encoded in the fittest chromosome from the last generation is then returned as the best resource allocation scheme for the process model.

We would like to emphasize that the entire GGA framework and the input process models in Figure 1 are both encoded as Color Petri Nets models. In this way, the framework can be loaded into CPN Tools [19] and, once a process model has been attached to it, the optimization is performed by executing the framework's CPN. In other words, the GGA framework acts as a "wrapper" for the process model itself, feeding it with different resource allocation schemes until an optimal scheme is found. To better illustrate the proposed idea, each Color Petri Nets model of the GGA framework are translated into corresponding Data Flow Diagrams (DFD) and they are depicted in Appendix A. In the following sections, we describe each component of the GGA framework.

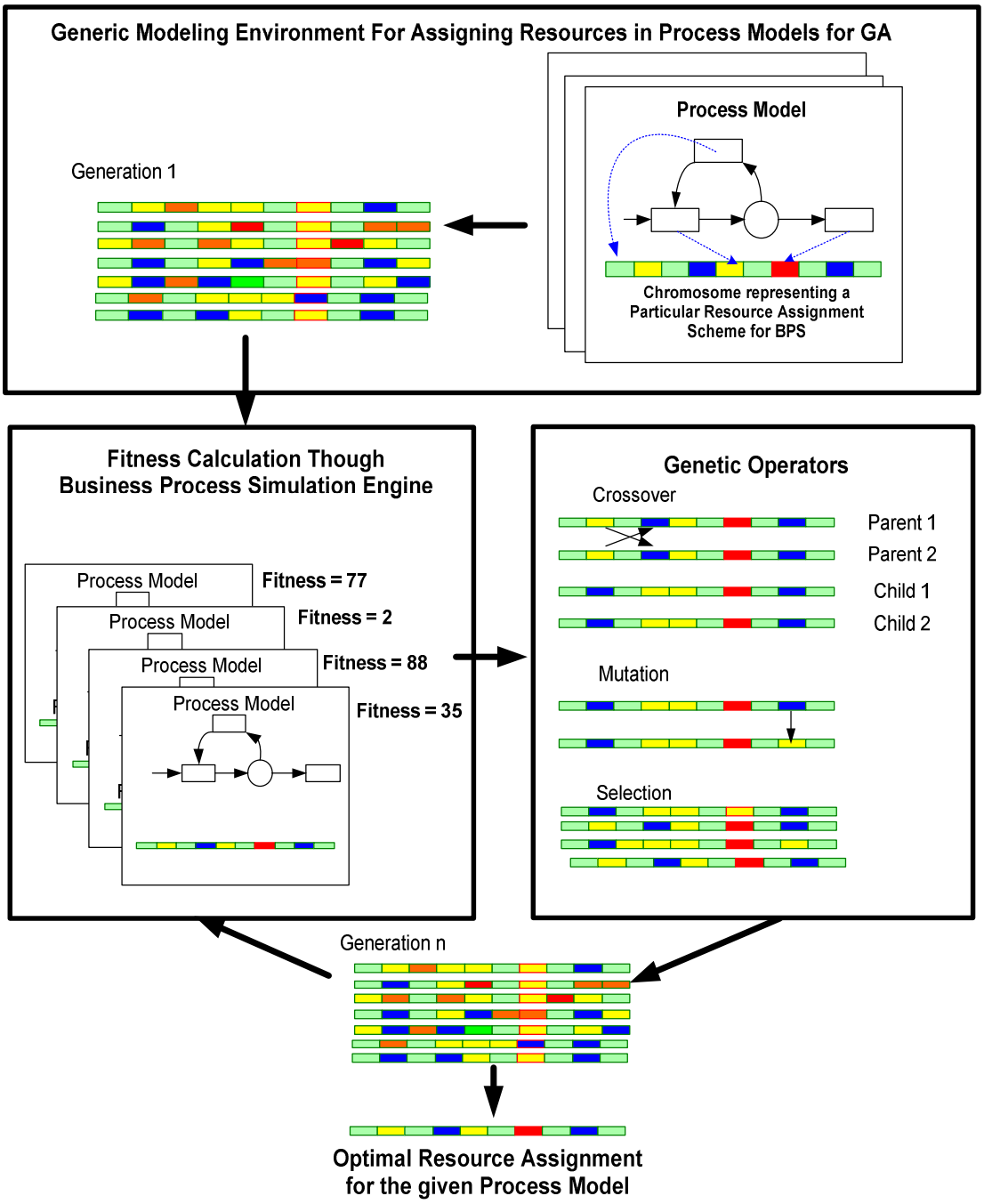

Figure 2. The overall scheme of the Generic Genetic Algorithm (GGA) modeling framework for resource optimization in business process simulation (The level-0 DFD diagram of GGA framework is depicted in Figure 29 in Appendix A.)

3.1 Chromosome generation

First, the GGA framework accepts inputs from the user to configure the Genetic Algorithm. For instance, the user can configure the population size, the number of resource types, the number generations needed for the GA, the selection rate, and the elimination rate. Next, the algorithm randomly generates chromosomes based on the population size, tasks to be assigned, and the resource bounds associated to each resource type in the process model. In the proposed model, there is no limit on the number of resource types for the simulation. The CPN models of chromosome generation functions are depicted in Figure 2 and 3. The corresponding DFD diagram is depicted in Figure 30 in Appendix A.

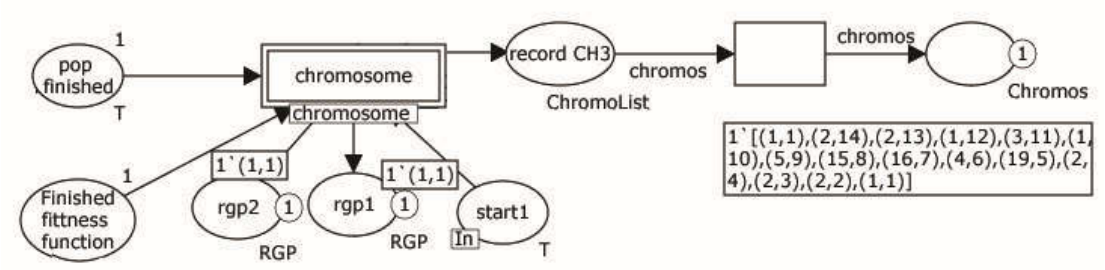

Figure 2. Chromosome function (Level 0)

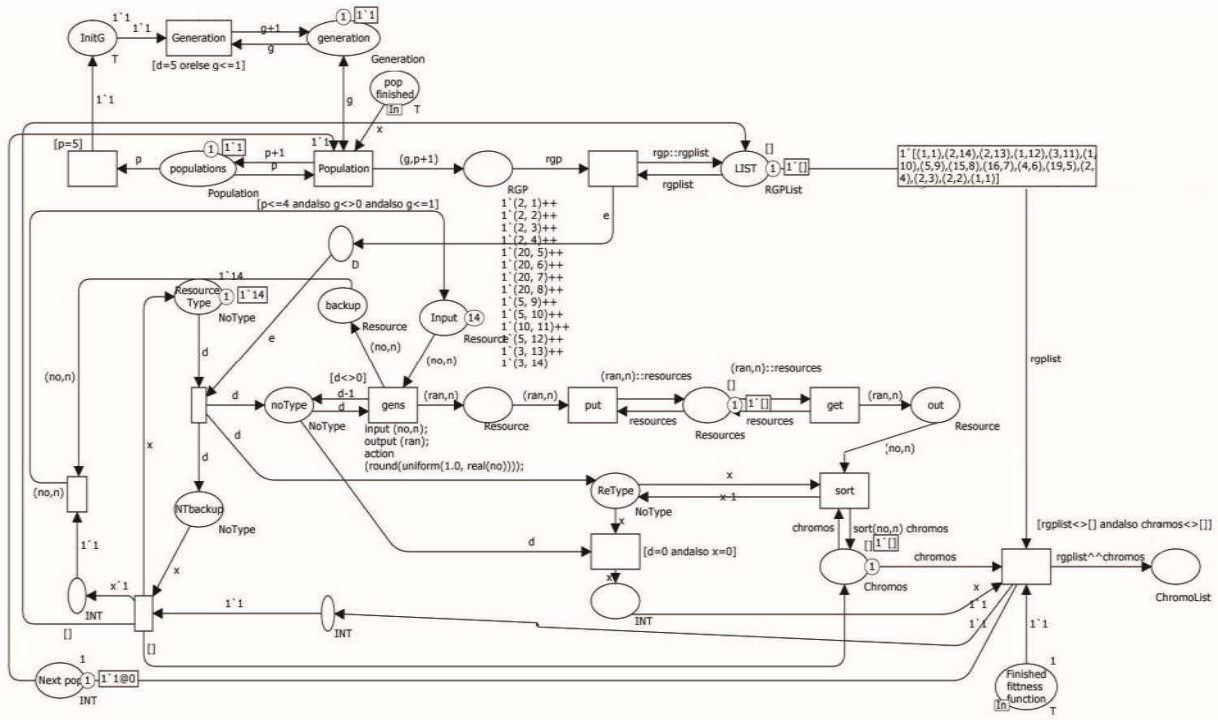

Figure 3. Chromosome function (Level 1)

In our approach, valid resource assignment schemes for the business process are constructed as chromosomes. The chromosome with the highest fitness value is expected to produce the best resource assignment scheme for the business process. The generation of valid chromosomes in our framework can be illustrated based on a sample business process (see Figure 4). Suppose that there are nine tasks and six different types of resources in the process. Tasks are depicted in green and resources are shown in yellow. The lower and upper bound of the available resources are shown under the resource name in brackets.

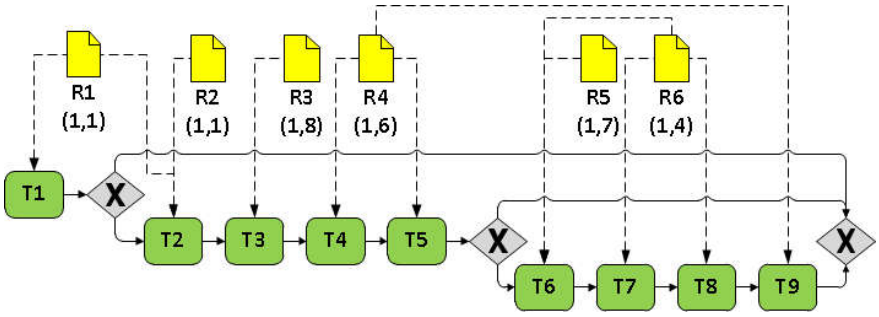

Figure 4. Resource assignment in sample business process

Based on the resource assignment constraints from Table 1, we can randomly generate a number of valid chromosomes. An example of a valid chromosome can be generated as follows: ((T1,R1,1),(T2,R1,1),(T2,R2,1),(T3,R3,3),(T4,R4,2),(T5,R4,2),(T6,R5,4),(T6,R6,1),(T7,R6,2),(T8,R6,1),(T9,R4,2)).

Table 1. Constraints on resource assignment scheme of a sample business process

Task ID	Resource Types	Lower Bound	Upper Bound
T1	R1	1	1
T2	R1	1	1
	R2	1	1
T3	R3	1	8
T4	R4	1	6
T5	R4	1	6
T6	R5	1	7
	R6	1	4
T7	R6	1	4
T8	R6	1	4
T9	R4	1	6

Population size depends on the nature of the problem and chromosomes are randomly generated to cover the range of possible solutions. After the chromosomes are generated, they are used as resource allocation schemes by process models for simulation. Once the process simulation is completed, the fitness of each chromosome (resource allocation scheme) can be calculated. Each chromosome is then sorted based on the fitness value. The new generation of chromosomes are generated after performing selection, crossover and mutation operations on the selected chromosomes.

3.2 Fitness function

In the proposed framework, we define a fitness function *f(c,g)* (in equation 1) for calculating the fitness of chromosome *c* from generation *g* based on the total workflow completion time and total cost of all tasks *n* in the workflow. The CPN models of fitness functions are depicted in Figure 5 and 6. The corresponding DFD diagram is depicted in Figure 31 in Appendix A.

$$f(c,g) = \frac{1}{\sum_1^n time_i^{c,g}} + \frac{1}{\sum_1^n cost_i^{c,g}} \quad (1)$$

In Figure 5, the "Fitness function" transition will be fired when there is a result available in the "results" place. By firing the transition, the result will be passed into the level 1 CPN model of fitness function (Figure 6) to evaluate the fitness. The higher fitness result represents further improvement of the related parameters. In our model, we compare the fittest chromosome of current generation with the previous generation's fittest chromosome to check whether they are similar in value. This checking is used to confirm whether the stopping criteria is met (see Section 3.5 Termination). After that, the tokens are passed into the "sorting" transition to do to extract the best results. The tokens are sorted in descending order according to their fitness. The tokens with high fitness values are stored for next generation.

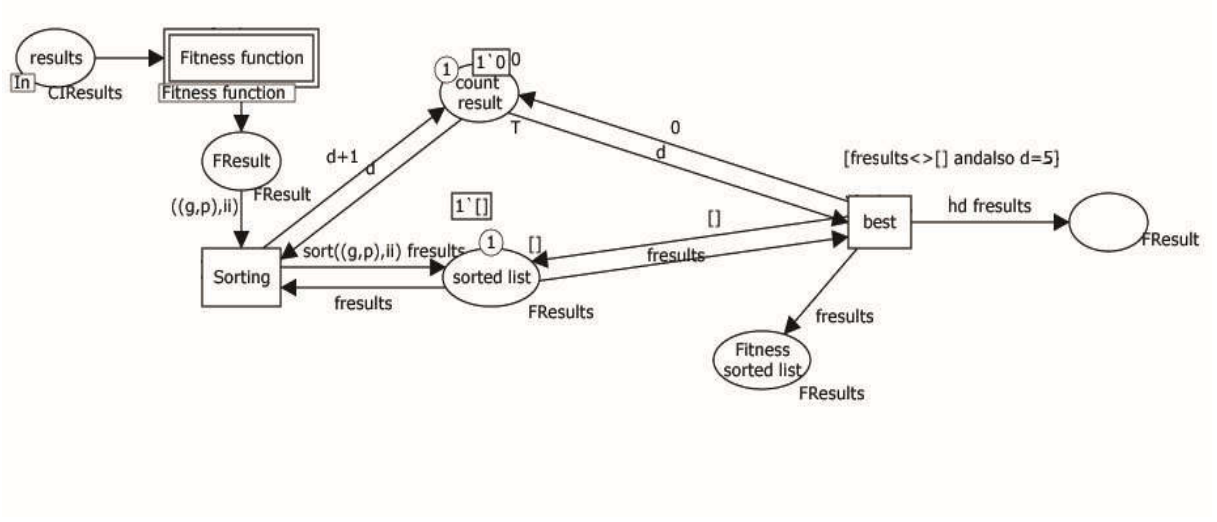

Figure 5. Fitness function (Level 0)

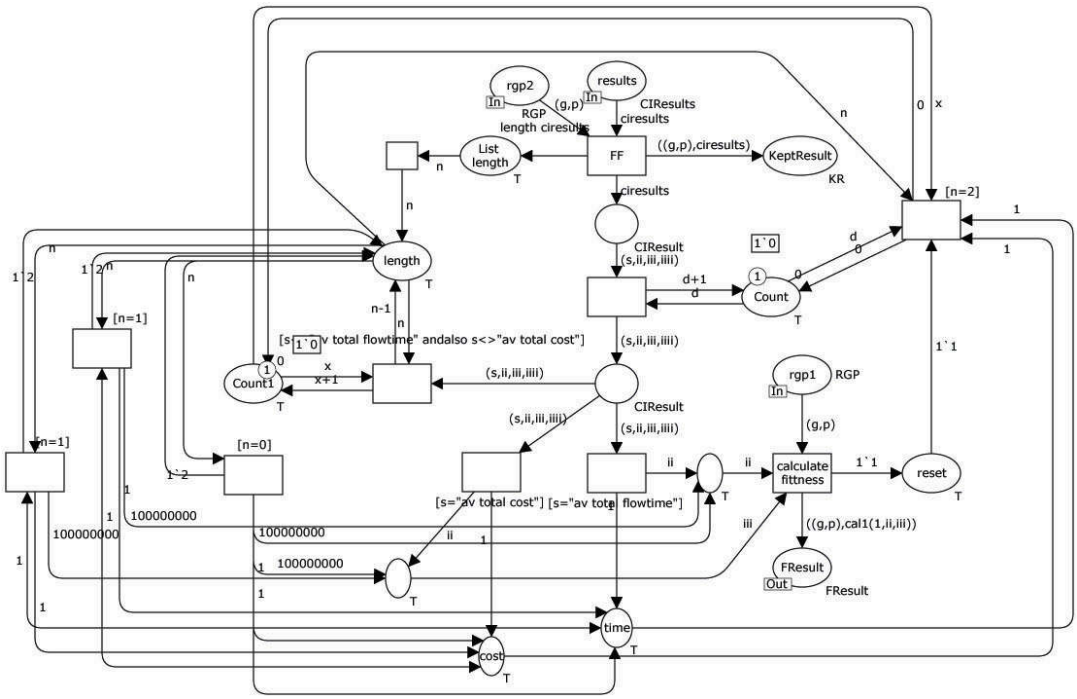

Figure 6. Fitness function (Level 1)

3.3 Selection operation

Selection is a genetic operator that chooses a chromosome from the current generation's population for inclusion in the next generation's population. At the later stage, selected chromosomes will be passed into the crossover and mutation operators to produce the offspring chromosomes for the next generation. Selection rate is important in deciding how many chromosomes can be maintained in the next generation. In our model, a predefined percentage of the non-fit chromosomes will be eliminated and the parents with high fitness value will be chosen to produce new chromosomes. In our case, 90% of the selection rate can ensure the chromosomes remain high variety. The CPN model of the selection operator is depicted in Figure 7. The corresponding DFD diagram of Figure 7 is depicted in Figure 32 in Appendix A.

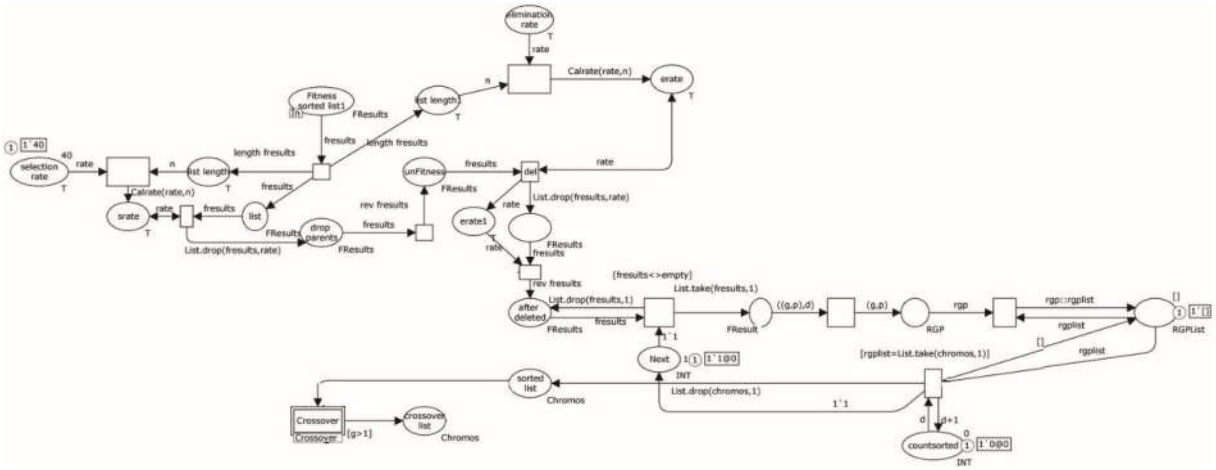

Figure 7. Selection operation (Level 0)

3.3 Crossover operation

The CPN models of a crossover operator are depicted in Figure 8 and 9. The corresponding DFD diagram is given Figure 33 in Appendix A. To generate chromosomes for the next generation, crossover operator is applied to selected chromosomes from the current generation. In this operator, a pair of chromosome is selected for breeding and the crossover point is randomly selected. Each parent is divided into 2 parts (genes) based on the crossover point. After that, genes from the parents are swapped to form 2 new chromosomes. In each iteration, new parents are selected for crossover and the process continues until the required number of offspring is generated. Crossover operation is designed to produce next generation of chromosomes that is different from the current generation. Crossover operator also aims to increase the average fitness of the population since only the fittest members from each generation are selected for breeding.

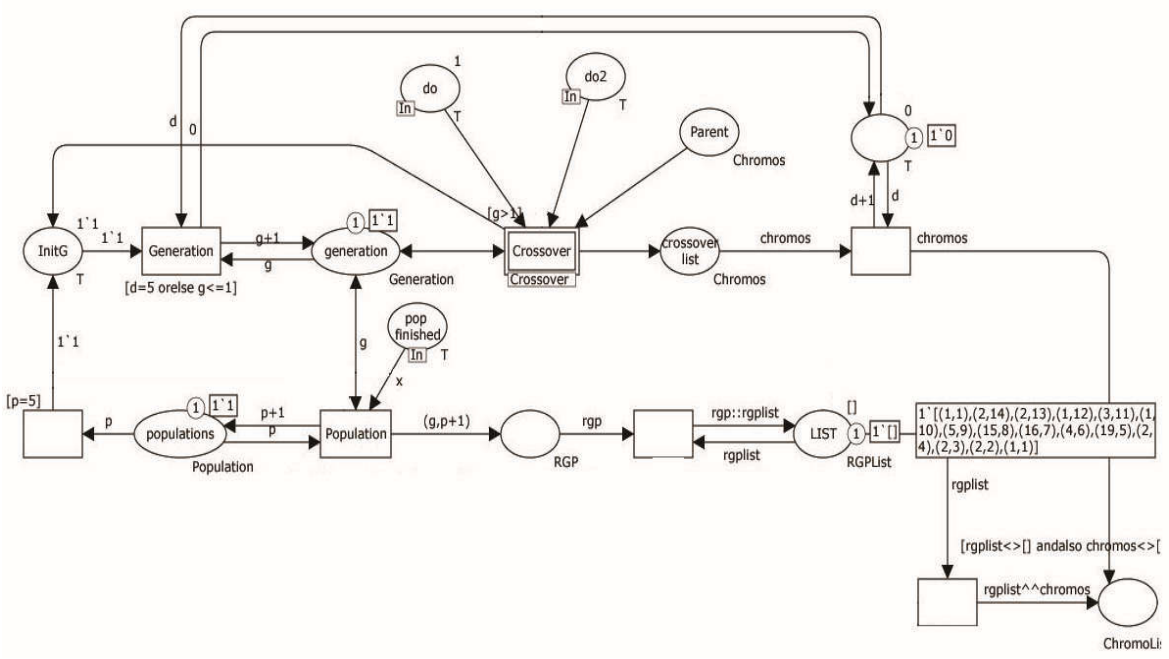

Figure 8. Crossover operator (Level 0)

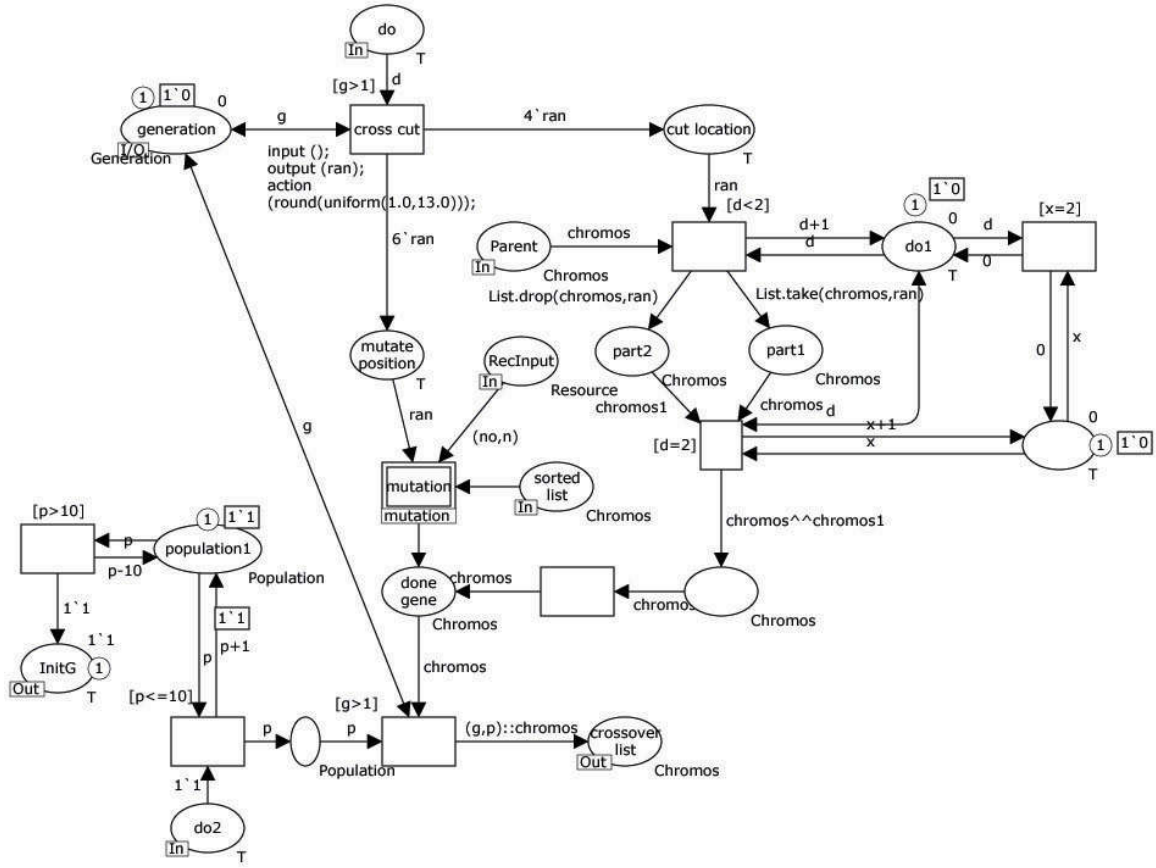

Figure 9. Crossover operator (Level 1)

3.4 Mutation operation

The CPN model of mutation operator is depicted in Figure 10. The corresponding DFD diagram is depicted in Figure 34 in Appendix A. Mutation operation is used to diversify the genetic property of genes in the chromosomes for the next generation. Mutation is an important operator for evolving into better solutions as well as preventing the current population from trapping at local optima. In the proposed framework, we use the single point mutation and the probability of mutation is set to $^1/_L$ where L is the length of the chromosome. In this operator, the mutation point is randomly chosen and each chromosome is divided into 2 parts except the mutation point. After that the genes are extracted and their values are adjusted randomly according to the lower and upper limits of the resource type. Finally, all genes are combined to produce a new chromosome which is stored in a list.

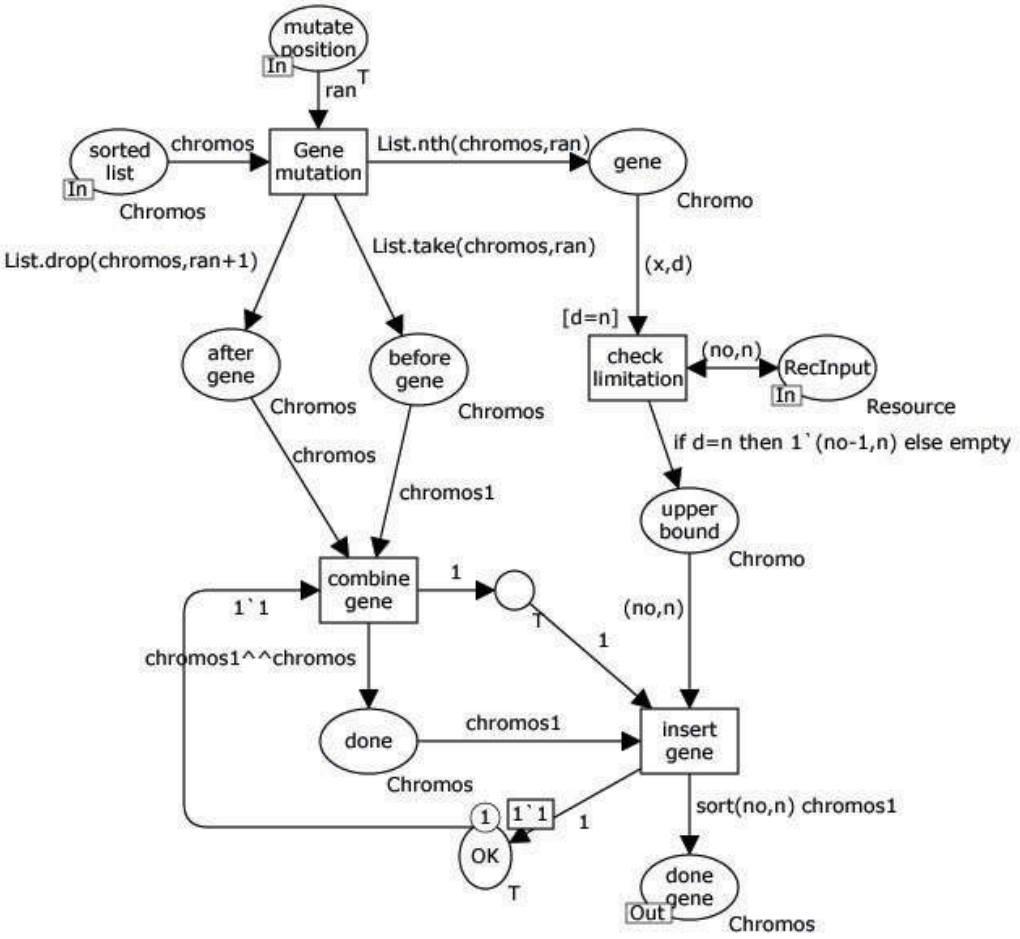

Figure 10. Mutation operator (Level 0)

3.5 Termination

The evolution process of genetic algorithm stops when a termination condition has been reached. In our case, the simulation will stop when the change in the fitness of several generations is less than a predefined threshold. In the proposed model, the threshold is set to the range of -3% to 3%. The genetic algorithm then returns the fittest chromosome from the final generation. The resulting chromosome represents the best possible resource allocation scheme for the process model. The CPN model of termination function is depicted in Figure 11.

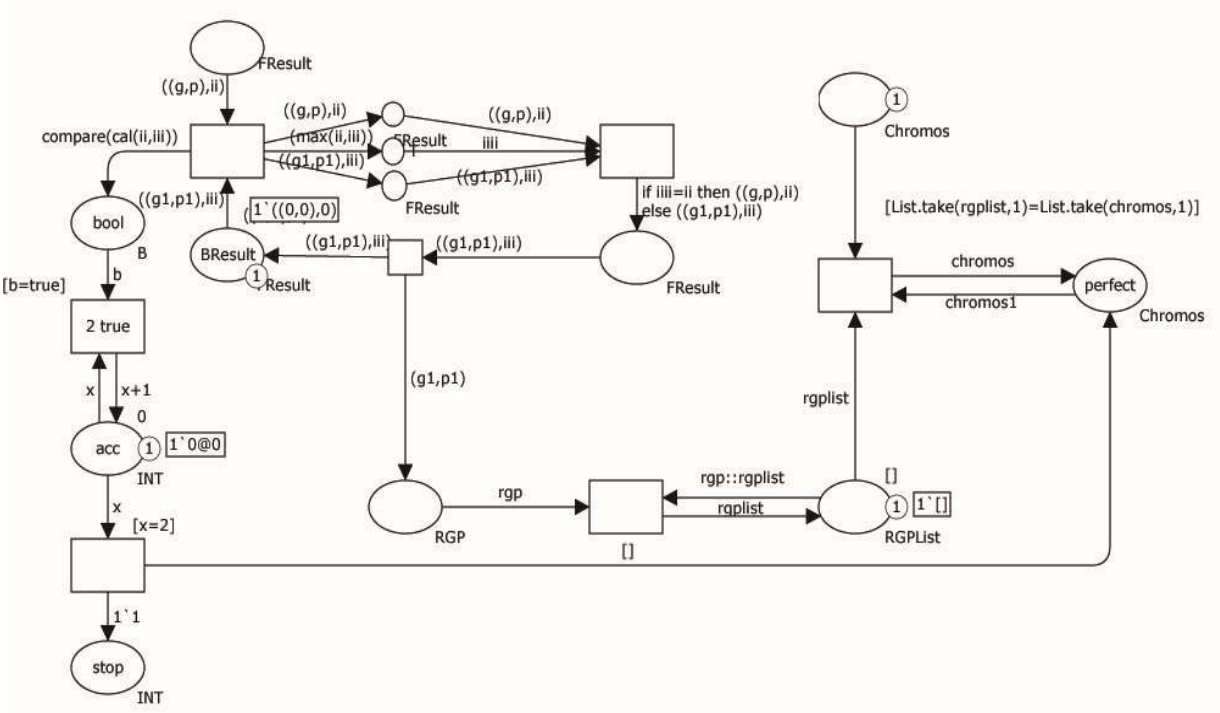

Figure 11. Termination Function

The pseudo-code of the genetic algorithm from the GGA framework is shown in Algorithm 1.

procedure Genetic algorithm for resource optmization
begin
set generation $g := 1$;
set counter $i := 1$;
set population size p;
\\Generation of initial population
While $i \leq p$ **then Do**

```
        Randomly generate a valid chromosome i based on the resource bounds
        extracted from the business process;
        i := i+1;
    end
    Set i := 1;

    Repeat
        \\Fitness calculation
        While i ≤ p then Do
            Run the simulation based on resource assignment scheme from
            chromosome i;
            Derive the fitness of chromosome i based on simulation result.
            i: = i+1;
        end
        \\Generation of offspring for next generation
        Sort all the chromosomes by fitness;
        Select 40% of parents from the population;
        Crossover to produce offspring from these pairs;
        Mutate the remaining 50% of the candidates from the population;
        Randomly generate 10% candidates for the new population;
        Replace the weakest candidate of population with these new offspring;
        Set g:= g+1;
        Set i:= 1;
    Until change in degree of fitness <= predefined threshold
    return the chromosome with best fitness result;
end
```

Algorithm 1. Pseudo-code of genetic algorithm for CPN models

Although we have used some default settings for the simulation, they can be changed by the user to meet their requirements. In our prototype, the population size is set to 10 in each generation and there is no limit on the number of resource types for the process model. The lower and upper bound of a resource can be defined in the chromosome structure and the selection and elimination rate is set to 90% (40% for crossover and 50% for mutation) and 10% respectively. The crossover point is randomly generated by the

GGA and the mutation rate is *1/L* where *L* is the length of the chromosome. All these settings are stored in the places in the GGA framework and can be altered according to the user's input.

4. ARCHIVE MANAGEMENT WORKFLOW CASE STUDY

In this section, we demonstrate the application of the proposed framework to an archive management workflow at Macau Historical Archives. An archive [42] is a collection of records and documents that are conserved as an invaluable asset. The level-0 CPN model of Archive Management Workflow analyzed in this paper is depicted in Figure 12. Due to the limited space, only the highest level of CPN model is shown in the article.

We would like to emphasize that the process model (cf. upper-right-hand corner of Figure 1 from Section 3) can be any CPN model. Since the remaining modules of the GGA framework in Figure 1 are also implemented in CPNs, these modules can be seamlessly integrated with any process model to be optimized (e.g. Archival Management process in Figure 12). This means that we can leverage the expressiveness of CPNs for representing process models. The GGA framework which acts as a wrapper is depicted as the transitions named "GA" and "Pass in" in the shaded colored rectangle areas in Figure 12. We claim that the proposed GA framework is generic since the wrapper can be plugged into any process models encoded in CPNs. For instance, the remaining parts outside of the shaded area in Figure 12 and the corresponding level-0 process model of archive management workflow which are encoded in CPNs.

The high level processes "Pass in" and "GA" – depicted as rectangles with double border lines in the highlighted area of Figure 12 – are implemented as separate CPN models. The "GA" sub-process encompass all the modules of the GGA framework discussed in previous sections. The "Pass in" sub-process integrates the business process with the CPN models of the GGA framework. To enable this integration, we define an interface between the GGA framework and the embedded process model based on a set of designated *places* in these CPN models. These places enable the transfer of chromosomes from the business process model to the GA algorithm and vice-versa. Chromosomes are encoded as *tokens* in the Petri Nets model. The level-0 CPN model of "Pass in" sub-process is depicted in Figure 13.

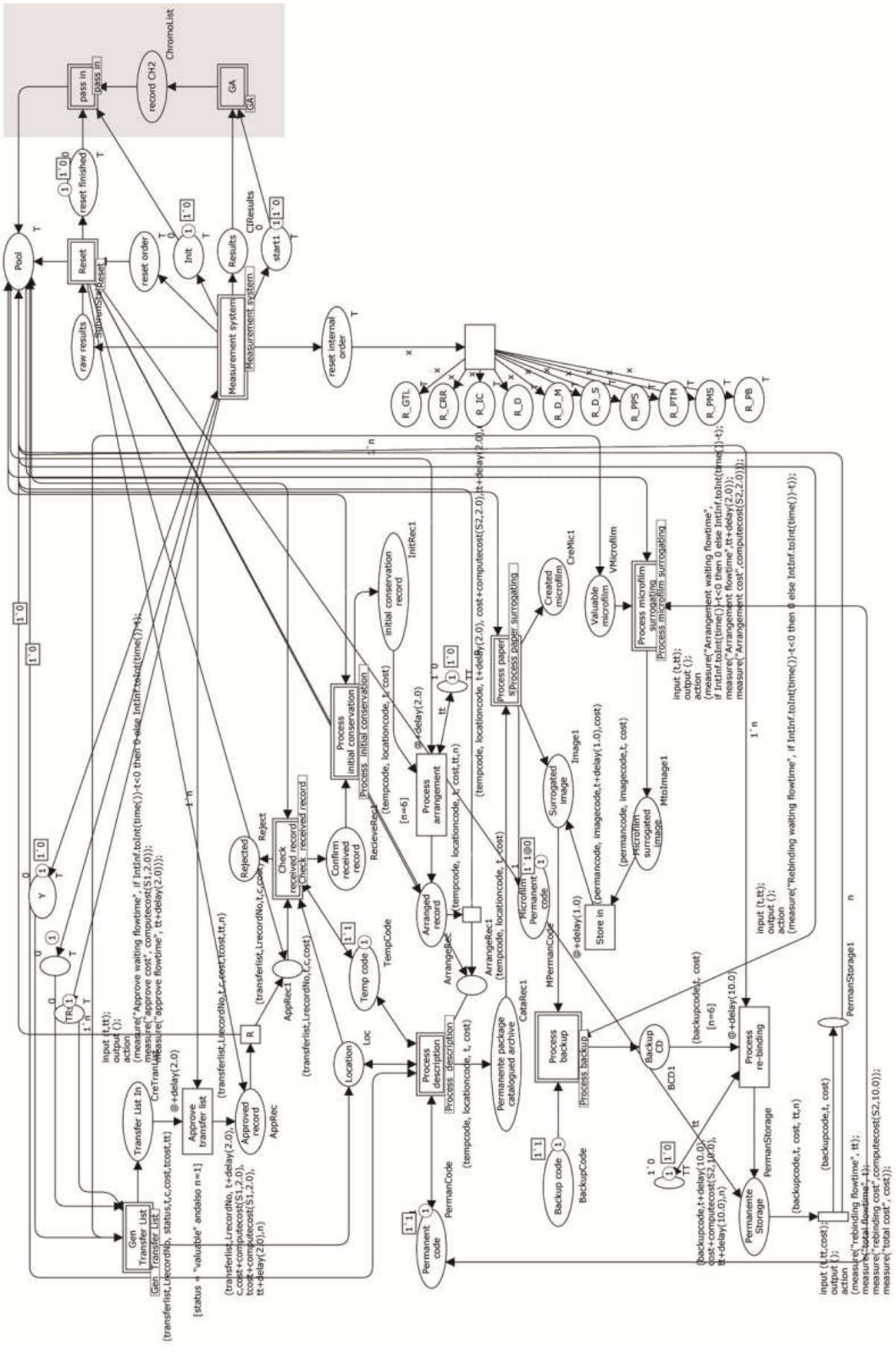

Figure 12. Level – 0 Color Petri Nets model of Archival Management Workflow

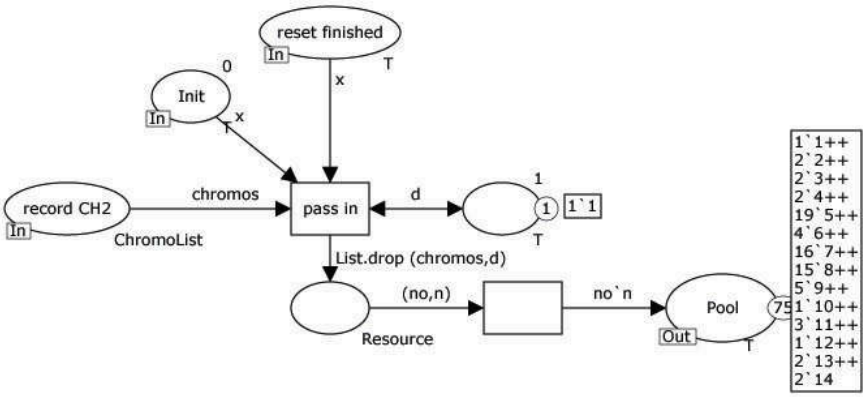

Figure 13. Pass In Function (Level 0)

The "Pass in" module communicate with the business process model – encoded by the transitions and places on the left side of Figure 12 – via the designated places called "Pool" and "Results". The place "Pool" contains tokens for the simulation. These tokens are referenced during the simulation by the respective tasks in the process model. After a simulation completes, the result is returned to the "GA" modules from the process "Measurement system" via the place "Results". In our prototype, all transitions in Figure 12 are implemented as separate CPN models.

For convenience, we conceptualize the level-0 workflow model of Macau Historical Archives in Figure 14 using the Event-driven Process Chains (EPC) notation – a notation in which a process model is represented as a directed graph consisting of: functions (rounded rectangles) representing tasks; events (hexagons) representing for example outcomes of decisions; connectors (circles) representing for example points of choice; and resource types (ellipses).

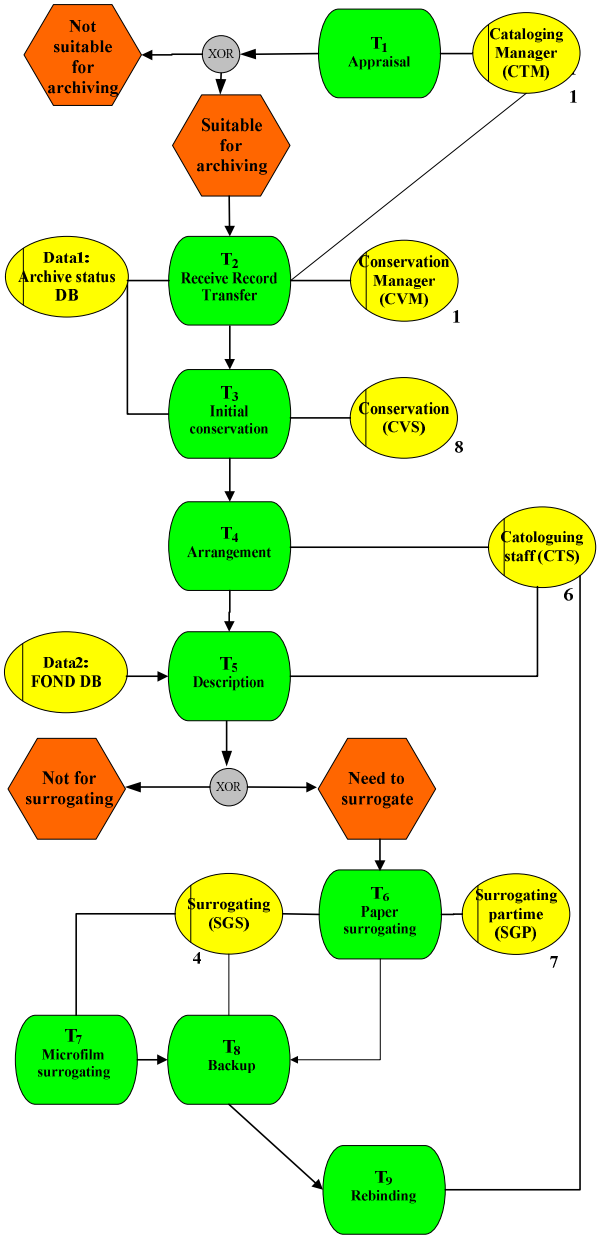

Figure 14. Archival management workflow (Level 0)

First, records for archiving are evaluated in the "Appraisal" (T1) task. Appraisal is the act of assessing whether these records have sufficient value to warrant acquisition by an archival institution. After appraisal, the records are formally accepted by the Historical Archives in the "Receive record transfer" (T2) task. Next, in the "Initial conservation" (T3) task, basic cleaning of the records is performed before they are grouped in the "Arrangement" (T4) task. Then the "Description" (T5) task is carried out to organize and record details of the archive based on international description standards. After descriptions

are added, the “Paper surrogating” (T6) task is carried out to create digital archives from the physical format. The next steps in the process involve creating backups and rebinding archives. In the “Backup” (T8) task, the original files are copied into storage media so that it can be restored if the original data is deleted or damaged. The “Rebinding” (T9) task is performed for repackaging and associating related meta-information prior to permanent storage. The “Microfilm surrogating” (T7) task captures and stores images of the archives in microfilm formats. Finally records are stored at the permanent storage. Depending of the nature of the archived material, periodic maintenance tasks (i.e. preservation) are also scheduled at the end of the process.

At the beginning of each year, transfer lists are sent to the Macau Historical Archives by various organizations. A transfer list contains approximately 225 records for archiving. In normal condition, approximately two transfer lists and a box of microfilm are received per year. Each box of microfilm contains approximately 2000 images. According to the recent statistics in Macau Historical Archives, approximately 14% microfilm pages are delayed and the bottleneck was located at T4, T5 and T6 since relatively high number of assigned records cannot be processed on time. We also find that delays in these tasks have a ripple effect on the whole process.

In the CPN process model, each task is assigned with an “estimated average completion time” for processing a job (see Table 2). These values are calculated from recent statistical data from Macau Historical Archives. We can see that the total completion time of the workflow is approximately 1580 minutes.

Table 2. Average completion time of each task in the workflow

Task	**Task Description**	**Estimated Average Task Cost (per record)**	**Estimated Average Completion Time (per record)**
T1	Appraisal	$5.28	2 minutes
T2	Received record transfer	$9.5	4 minutes 25 seconds
T3	Initial conservation	$182.83	85 minutes
T4	Arrangement	$4.3	2 minutes
T5	Description	$3810.15	24 Hours
T6	Paper surrogating	$37.23	19 Minutes 30 seconds
T7	Microfilm surrogating	$12.84	6 Minutes
T8	Backup	$17.83	8 Minutes 20 seconds

T9	Rebinding	$18.43	10 Minutes
Total (per record)		**$4098.39**	**1577 Minutes 15 seconds**

For better accuracy, each simulation experiment is run for a 4 years period and average values are calculated for comparison. In the Macau Historical Archives, there are 14 types of resources including 6 types of resources described in level-0 workflow model from Figure 12. We consider human resources and equipment in our experiments. There are 4 managers (senior technician level) assigned for each department. There are also a number of deposit, conservation, surrogating, and cataloguing managers. Table 3 shows the cost of each type of human resource and the departments to which these resources belong.

In addition to these human resources, there are several pieces of resources. Two scanners (MOP $106,000 & MOP $40,000 per each.) and one eclipse scanner (MOP $680,000) are used for scanning microfilm to images. One DSV 300 scanner (MOP $270,000) is used for editing microfilms and one DAW writer (MOP $430,000) is used for writing microfilms. Two disinfection machines (MOP $170,000) and two dusting disposal machines (MOP $150,000) are also used.

Table 3. Human Resources in Macau Historical Archives

Position	**Conservation Department**	**Cataloguing Department**	**Surrogating Department**	**Deposit Department**
Senior technician MOP $25370/month	1	1	1	1
Professional technician MOP $20650/month	8	6	4	--
Part-time worker MOP $35/Hour	--	--	7	--
Total cost per month (MOP $)	$190570	$149270	$118470	$25370

The resource assignment scheme that is currently used in Macau Historical Archives is depicted in Table 4. For illustration, we denote this scheme as “original as-is model”. There is however some flexibility within the Historical Archives that allows the number of human resources to be varied. The resource bounds (upper and lower number of resources that can be allocated to each task) are those shown in Table 1.

Table 4. Resource allocation scheme in original as-is model

Resource Types	Description	Resource usage	Task
R1	Cataloguing manager	1	T1, T2, T5
R2	Deposit manager	1	T2, T5
R3	Conservation manager	1	T2
R4	Surrogating manager	1	T6, T7
R5	Conservation	8	T2, T3, T5,T6,T8
R6	Cataloguing	6	T4, T5, T9
R7	Surrogating	4	T6, T7, T8
R8	Surrogating part time	7	T6
R9	DSV 300	1	T7
R10	m-scanner	1	T7
R11	Paper scanning	1	T6
R12	Microfilm writer	1	T6
R13	Disinfection machine	2	T3
R14	Dusting machine	2	T3

5. ARCHIVE MANAGEMENT WORKFLOW EXPERIMENTAL RESULTS

The results of running the GGA framework on the Archive Management workflow are shown in Figure 15, Figure 16, and Figure 17. These figures show per-task KPIs for the original "as is" resource allocation scheme and for the best resource allocation schemes found by the genetic algorithm in the last five generations (denoted by G31, G32, G33, G34, and G35 respectively). The fittest chromosome found by GA in 35^{th} generation is depicted in Table 5. From these graphs, we can first of all observe that the original "as-is" allocation yields relatively long average waiting time per task. It also has the longest task completion time and largest total cost when compared to the resource allocation schemes generated during the application of the GGA framework. In Figure 15, we can see that T4, T5, T6 and T8 have particularly high average waiting time.

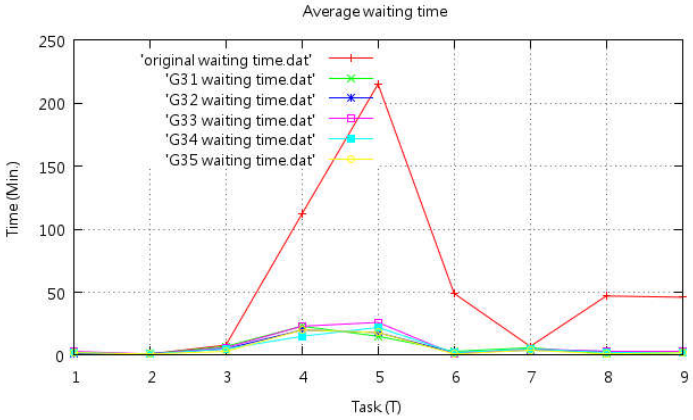

Figure 15. Average waiting time

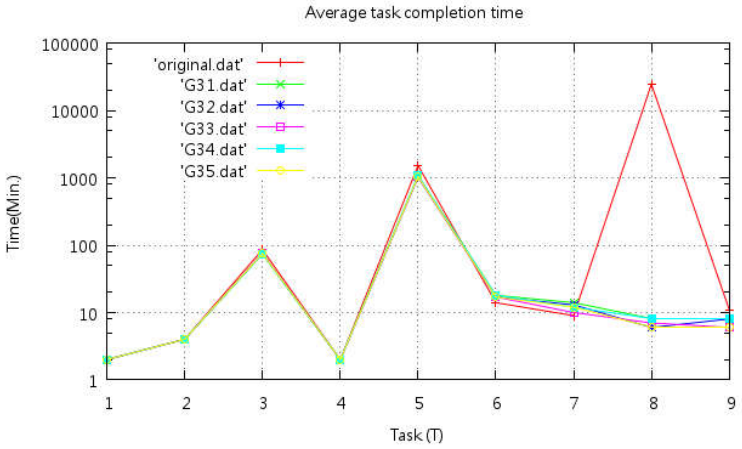

Figure 16. Average task completion time

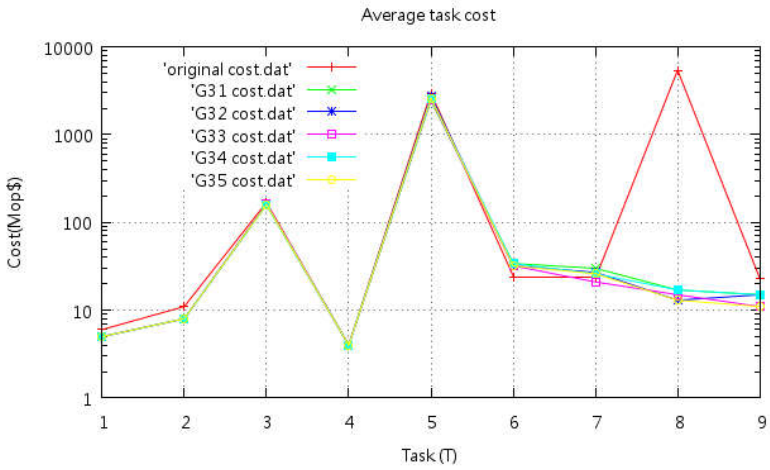

Figure 17. Average task cost

Table 5. Resource allocation scheme found in GA approach

Resource Types	Description	Resource usage	Task
R1	Cataloguing manager	1	T1, T2, T5
R2	Deposit manager	1	T2, T5
R3	Conservation manager	1	T2
R4	Surrogating manager	1	T6, T7
R5	Conservation	7	T2, T3, T5,T6,T8
R6	Cataloguing	6	T4, T5, T9
R7	Surrogating	3	T6, T7, T8
R8	Surrogating part time	7	T6
R9	DSV 300	1	T7
R10	m-scanner	1	T7
R11	Paper scanning	2	T6
R12	Microfilm writer	1	T6
R13	Disinfection machine	2	T3
R14	Dusting machine	2	T3

Figure 18 shows the evolution of the average total workflow time and average total workflow cost over time. The figure shows a gradual decrease in these two KPIs over the later generations.

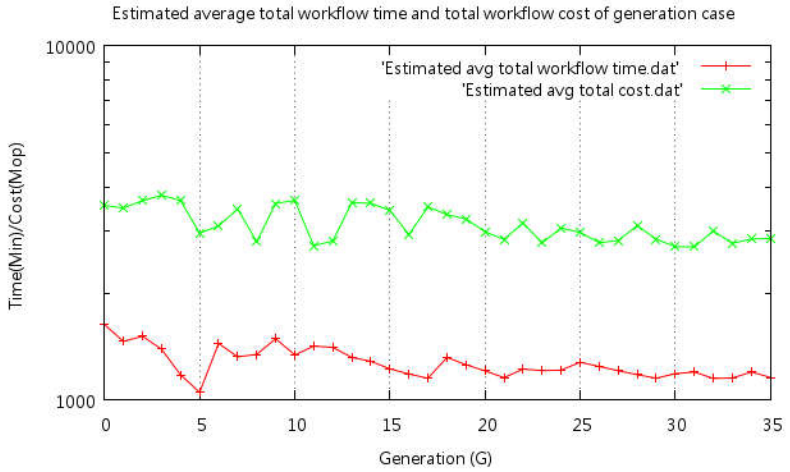

Figure 18. Estimated average total workflow time and average total workflow cost for each generation

From the above experiments, we find that the GA-based approach significantly reduces workflow time and the total cost relative to the original resource allocation scheme. We can see that original as-is model has the longest workflow time among all experimental models. Specifically, the GA-based resource allocation in generation 35 achieves 95% reduction in total workflow time and 55% reduction in total workflow cost.

6. INSURANCE CLAIMS WORKFLOW CASE STUDY

In this section, we apply our GA modeling framework to find an optimized resource allocation scheme for an insurance claim workflow [28]. The case study is based on the "teleclaims" process of a large Australian insurance company. The level-0 CPN representation of this model is given in Figure 19. Due to the limited space, only the highest level of CPN model is shown in the article. To better illustrate the idea, we also conceptualize the level-0 workflow model of "teleclaims" process in Figure 20 using the Event-driven Process Chains (EPC) notation.

The workflow handles inbound phone calls for lodging different types of insurance claims. Three sub-models describe this process: a back office model, a Brisbane call center model, and a Sydney call center model. Both centers have the same structure and are similar in terms of incoming call volume, average call handling time, and number of call center agents. There are two tasks in each call center: "check if sufficient information is available" and "register claim". Call Center Agents handle both tasks. Five tasks are performed in the back-office: (1) determine the likelihood of claim, (2) assess claim, (3) initiate payment, (4) advise claimant on reimbursement, and (5) close claim. An insurance claim will be rejected in the call center or in the back office if the claim is not qualified for reimbursement. Each case is handled by a single call center agent in the call center and then handled by a single claims handler in the back-end. Two scenarios are considered in this case study: a "normal" scenario and a "stormy season" scenario. In the normal scenario, approximately 9000 calls are received in each of the two call centers. In the stormy season, approximately 20000 calls are received in each call center. All claims lodged in the Brisbane and in the Sydney call centers are funneled to a single back-office where the claims are processed until their resolution.

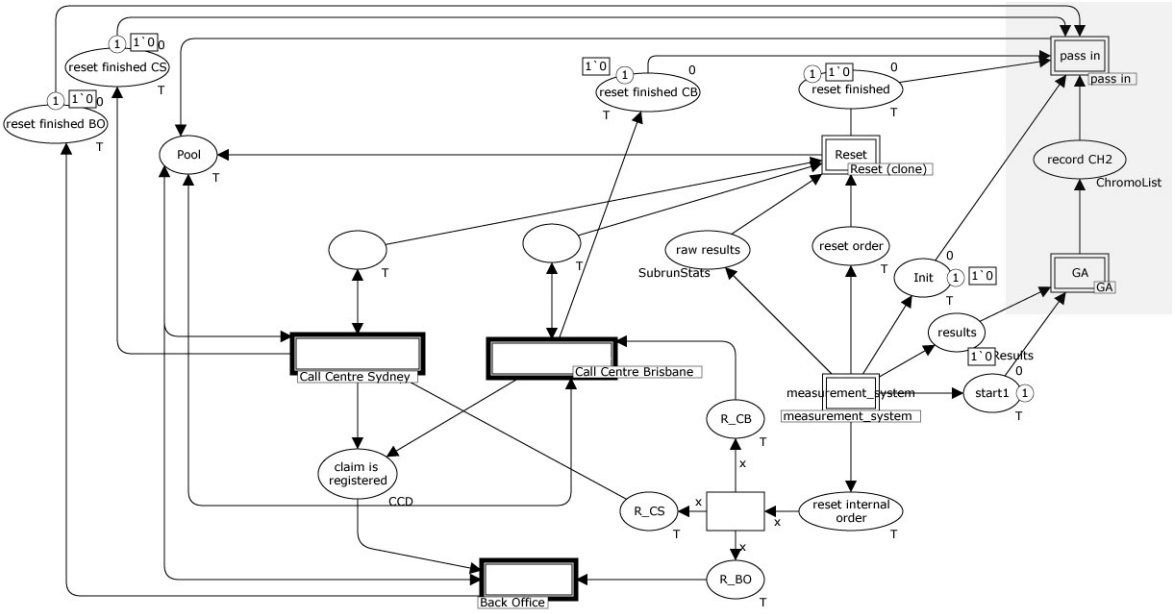

Figure 19. Level – 0 Color Petri-Nets Model of Insurance Claiming Workflow

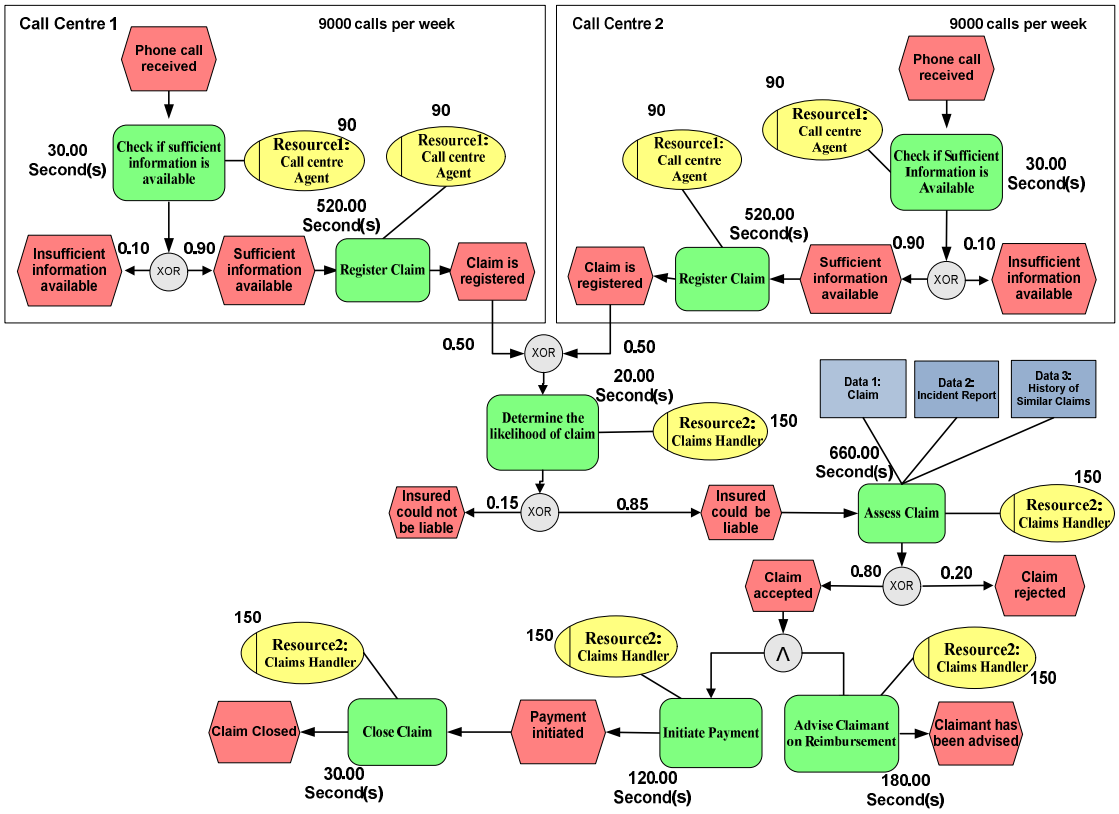

Figure 20. Insurance Claiming Workflow Model in EPC (Level 0)

In Table 6, we define initial values for the cost and time attributes of each task. These attributes are defined as follows:

- Average execution cost: Each task is assigned with a cost representing the portion of IT and office resources required to complete the task (including office supplies and postage cost when applicable). In addition to this cost attached to each task, there are compensation costs and resource usage costs as detailed below.
- Average completion time: Each task is assigned with an average completion time.

Table 6. The values of attributes for each task in experiment model

Task (T_i)	**Description**	**Average execution cost**	**Average completion time**
TB1	Check if sufficient information is available	10	30
TB2	Register Claim	20	520
TS1	Check if sufficient information is available	10	30
TS2	Register Claim	20	520
TO1	Determine the likelihood of claim	20	20
TO2	Access claims	30	660
TO3	Initiate payment	17	120
TO4	Advise claimant on reimbursement	10	180
TO5	Close claim	5	30

Each insurance claim requested from customers is denoted as a case and the following attributes are defined for each case in the experiment:

- ID of a case: Unique identifier of a case.
- Arrival time of a case: The time when the phone call for an insurance claim is received at one of the call centers.
- Compensation cost of a case: When a case misses its deadline, the client is compensated by the insurance company for violation of the service level agreement. To avoid paying compensation to clients, the insurance company must process insurance claims within their deadlines. In the experiment, compensation cost is assigned to a case when the insurance claim is made at one of the call centers. The range of compensation cost in the experiment is set from 120 to 160.
- Deadline of a case: All incoming cases are assigned with the same deadline. This deadline is calculated based on execution time of tasks along the longest path of the workflow.

Each resource has three attributes: (a) role: which used to describe the responsibility of an employee in the workflow (b) amount: number of resources; and (c) cost: wage of a resource. The wage of a call center agent is 4000 (per week) and the wage of a claim handler is 6000 (per week). When a task is executed for a given case, a resource cost is incurred proportional to the amount of time the task is executed. The number of available resources is shown in Table 7. These numbers are the same for both seasons.

Table 7. Existing Resource Allocation Scheme (both seasons)

Resource Type	Description	# of Resources	Tasks
R1	Sydney call agent	90	TS1, TS2
R2	Brisbane call agent	90	TB1, TB2
R3	Back office claim handler	150	TO1, TO2, TO3, TO4, TO5

To better understand the performance of the insurance claim workflow in normal and stormy seasons, we conducted an initial simulation experiment. Table 8 shows the results of this simulation for stormy season and for normal season. The results show that the current allocation is suitable for normal condition but not for stormy season condition. From Table 8, we can see that the waiting time in Back Office is significantly higher than the waiting time in Call Centers. It indicates that a bottleneck exists in Back Office during stormy season. In order to find better resource assignment schemes, we apply our generic GA framework first on the normal season and then on the stormy season. The results of these experiments are detailed in following section.

Table 8. Results in Stormy Season and in Normal Season

Description		**Normal condition (9000 cases per weeks)**	**Stormy season condition (20000 case Per week)**
TIME	Average workflow time	1213	35951
	Waiting time at Brisbane	0	0
	Waiting time at Sydney	0	0
	Waiting time at Back Office	0	40446
COST	Average workflow cost (per two weeks)	1566000	3480000
	Resource cost(per two weeks)	1620000	1620000
	Average compensation cost (per two weeks)	756000	4720000

	Total cost (per two weeks)	3942000	9820000

7. INSURANCE CLAIMS WORKFLOW EXPERIMENTAL RESULTS

Normal Season: The results of applying the GGA framework on the normal season are shown in Figure 21, Figure 22, and Figure 23. We observe that resource allocation scheme from 105th generation achieves similar average waiting time, 1% reduction of average task completion time, and 18% reduction in average total workflow cost. These results reflect the fact that the workflow is over-provisioned during normal season and thus substantial cost savings can be achieved by reducing the number of resources. Figure 24 compares the resource allocation schemes used in the normal season against the best allocation found by the GA framework. This figure confirms that the GA-based approach significantly reduces average workflow cost and resource cost compared to the original allocation scheme. The fittest chromosome found by the GA-based resource allocation in 105th generation is depicted in Table 9.

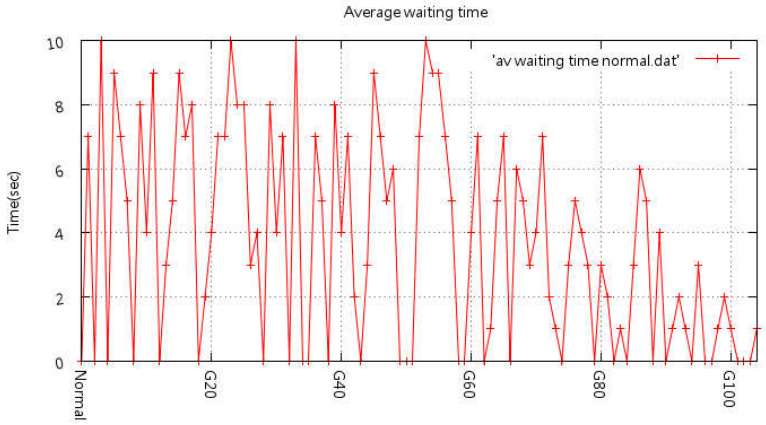

Figure 21. Average waiting time

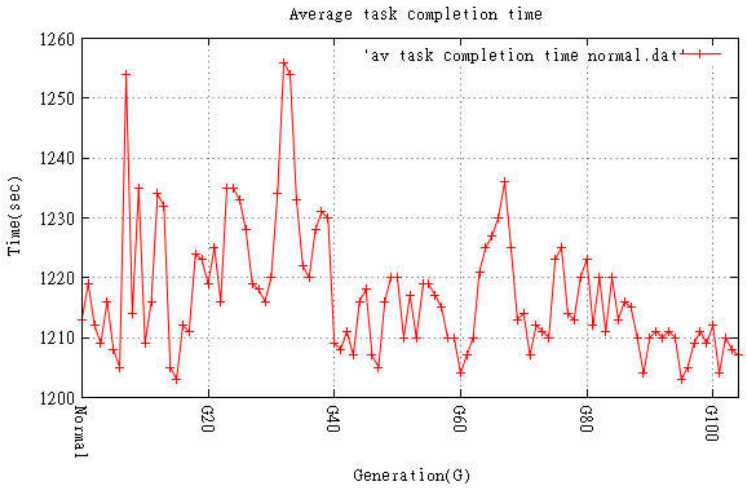

Figure 22. Average task completion time

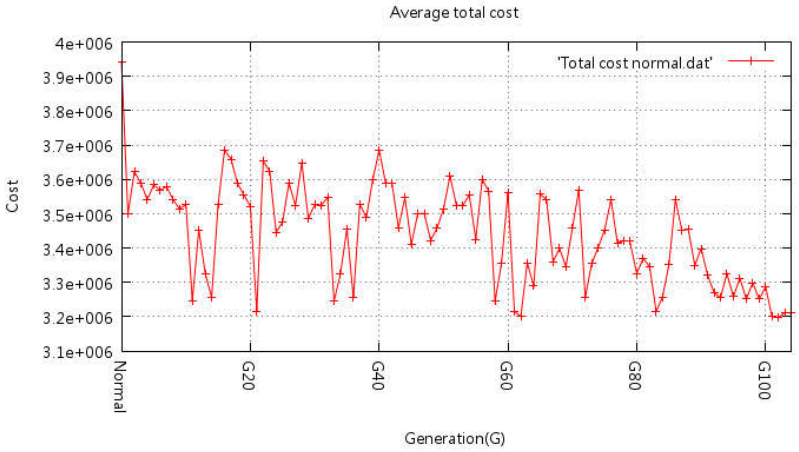

Figure 23. Average total cost

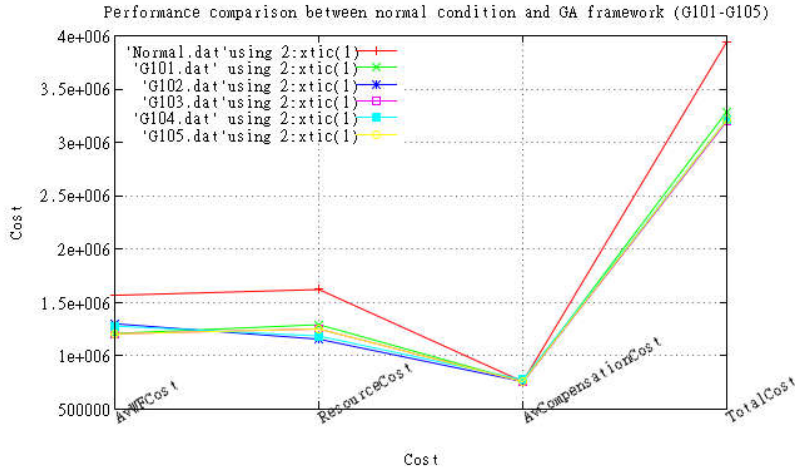

Figure 24. Performance comparison between original allocation for normal season and allocation found by GA framework at iterations G101-G105

Table 9. The fittest chromosome found by the GA in 105th generation

Resource Types	Description	Resource usage	Task
R1	Sydney call agent	65	TS1, TS2
R2	Brisbane call agent	50	TB1, TB2
R3	Back office claim handler	145	TO1, TO2, TO3, TO4, TO5

***Stormy Season**:* The experiment results for stormy season are given in Figure 25, Figure 26, and Figure 27. The experimental results show that the best resource allocation found after the 118th generation achieves 1% increase in average waiting time, 8% increase in average task completion time, and 5% reduction in total workflow cost. The fittest chromosome found by the GA-based resource allocation in 118th generation is depicted in Table 10. We can also observe that there is a slight increase in average waiting time and average task completion time since the resource allocation found by the GA-based approach uses fewer resources compared to the original resource allocation scheme (cf. Table 7).

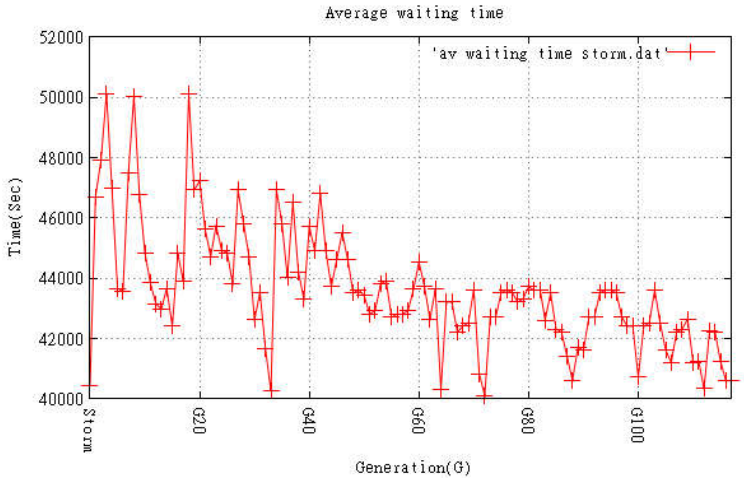

Figure 25. Average waiting time

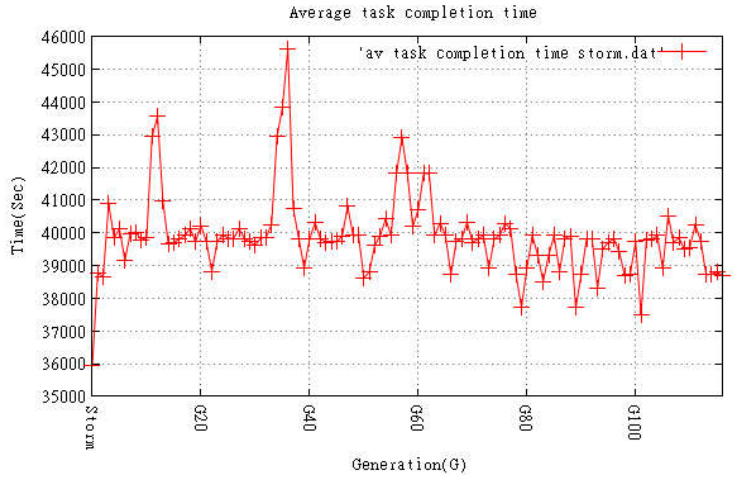

Figure 26. Average task completion time

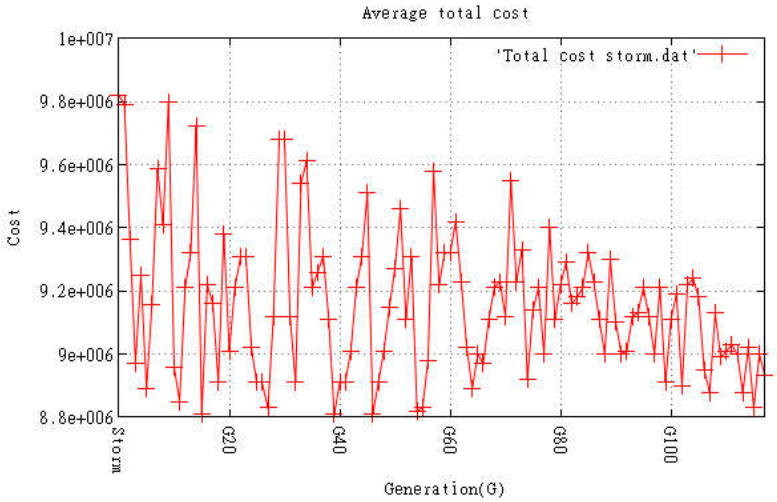

Figure 27. Average total cost

A more detailed performance comparison between the existing stormy season allocation and the ones found by the GA framework is shown in Figure 28. From this figure, we find that GA-based approach reduces average workflow cost, resource cost and total cost.

Table 10. The fittest chromosome found by the GA in 118th generation

Resource Types	Description	Resource usage	Task
R1	Sydney call agent	88	TS1, TS2
R2	Brisbane call agent	83	TB1, TB2
R3	Back office claim handler	145	TO1, TO2, TO3, TO4, TO5

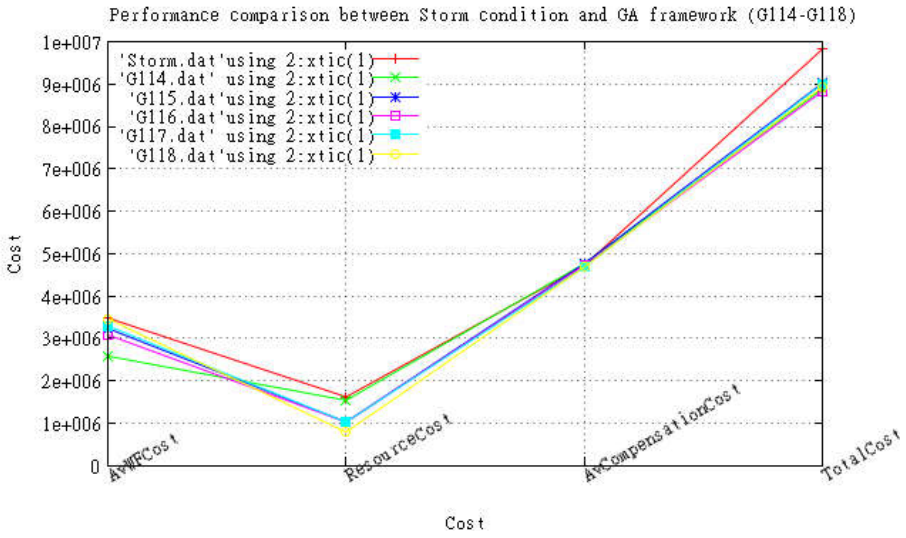

Figure 28. Performance comparison between original allocation in stormy season and allocation found by GA framework at iterations G114-G118

8. CONCLUSION

In this paper, we presented a Petri Nets based Generic Genetic Algorithm (GGA) Framework for resource optimization in business processes. In this framework, a genetic algorithm is entirely modelled in Color Petri Nets (CPNs) that can be used to optimize the allocation of resources in business processes by means of simulation. Four critical components of the framework are modelled in Color Petri Nets; (a) the whole genetic algorithm, (b) tokens for modeling chromosomes, (c) tokens for carrying the parameters for controlling the simulation, and (d) an interface based on places and transitions. To the best of our knowledge, no one has ever implemented genetic algorithm components directly in CPNs. Our work is the first-of-its kinds in business process simulation area.

The proposed framework is generic since it allows analysts to plug-in any process model captured as a CPN. Since all modules of the GGA framework are implemented in CPNs, we can leverage the expressiveness of CPNs for representing process models. The GGA framework takes as input a process model, information about available resources and execution times of tasks, and produces as output an optimized resource allocation scheme. By implementing GA components as well as the process models in CPN, genetic algorithm and process simulation can be tightly integrated and executed under the same IDE.

The applicability of the framework was evaluated via two case studies: one stemming from the Macau Historical Archive and another from an Australian insurance company. In both case studies, the framework identified significantly improved resource allocation scheme relative to the one that existed when the data for the case studies were collected.

So far, the proposed generic evolutionary framework can be used to simulate process models defined in CPNs. As for the future work, we plan to extend our framework for simulating workflows defined in other process modeling notations. Another avenue for future work is to extend the framework in order to handle other attributes besides time and cost-related ones, like for example error rates (i.e. reliability).

9. ACKNOWLEDGEMENTS

This research was funded by the Research Committee of University of Macau, grant MYRG2016-00148-FST and MYRG2017-00029-FST.

10. REFERENCES

1. Liu, Y. and J. Iijima, *Business process simulation in the context of enterprise engineering.* Journal of Simulation, 2015. **9**(3): p. 206-222.
2. Rinaldi, M., R. Montanari, and E. Bottani, *Improving the efficiency of public administrations through business process reengineering and simulation A case study.* Business Process Management Journal, 2015. **21**(2): p. 419-462.
3. Jakkhupan, W., S. Arch-int, and Y.F. Li, *Business process analysis and simulation for the RFID and EPCglobal Network enabled supply chain: A proof-of-concept approach.* Journal of Network and Computer Applications, 2011. **34**(3): p. 949-957.
4. Kloos, M., et al., *XBRL-Driven Business Process Improvement: A Simulation Study in the Accounting Domain*, in *Software Engineering and Formal Methods*, S. Counsell and M. Nunez, Editors. 2014, Springer Int Publishing Ag: Cham. p. 288-305.
5. Holland, J.H., *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence*. 1975, Ann Arbor: University of Michigan Press. viii, 183 p.
6. Djedovic, A., et al., *Optimization of business processes by automatic reallocation of resources using the genetic algorithm*, in *2016 11th International Symposium on Telecommunications (Bihtel)*. 2016: Sarajevo. p. 1-7.
7. Oh, J., et al., *Fault-tolerant execution planning for collaborative business processes based on genetic algorithms.* International Journal of Innovative Computing Information and Control, 2012. **8**(7B): p. 5265-5275.

8. Lee, H. and S.S. Kim, *Integration of process planning and scheduling using simulation based genetic algorithms.* International Journal of Advanced Manufacturing Technology, 2001. **18**(8): p. 586-590.
9. Hegazy, T. and M. Kassab, *Resource optimization using combined simulation and genetic algorithms.* Journal of Construction Engineering and Management-Asce, 2003. **129**(6): p. 698-705.
10. Jensen, K., *Coloured Petri nets (2nd ed.): basic concepts, analysis methods and practical use: volume 1*. 1996: Springer-Verlag. 234.
11. Aalst, W.v.d. and C. Stahl, *Modeling Business Processes: A Petri Net-Oriented Approach*. 2011: The MIT Press. 386.
12. Saitou, K., Malpathak, S., Qvam, H., *Robust Design of FMSs using CPN and Genetic Algorithm.* Journal of Intelligent Manufacturing, 2002(13): p. 339–351.
13. Chien, C.F. and C.H. Chen, *Using genetic algorithms (GA) and a coloured timed Petri net (CTPN) for modelling the optimization-based schedule generator of a generic production scheduling system.* International Journal of Production Research, 2007. **45**(8): p. 1763-1789.
14. Dezani, H., et al., *Controlling Traffic Jams on Urban Roads Modeled in Coloured Petri Net using Genetic Algorithm*, in *38th Annual Conference on Ieee Industrial Electronics Society*. 2012. p. 3043-3048.
15. Barzegar, S., et al. *Traffic Signal Control with Adaptive Fuzzy Coloured Petri Net Based on Learning Automata*. in *2010 Annual Meeting of the North American Fuzzy Information Processing Society*. 2010.
16. Napalkova, L., G. Merkuryeva, and M.A. Piera, *Development of Genetic Algorithm for Solving Scheduling Tasks in FMS with Coloured Petri Nets*. International Mediterranean Modelling Multiconference 2006, ed. A.G. Bruzzone, et al. 2006. 135-140.
17. Lin, C.P. and Ieee, *Decision support in computer integrated manufacturing using Fuzzy Colored Petri Nets with genetic algorithm*, in *Smc '97 Conference Proceedings - 1997 Ieee International Conference on Systems, Man, and Cybernetics, Vols 1-5: Conference Theme: Computational Cybernetics and Simulation*. 1997. p. 82-87.
18. Shojafar, M., et al., *ALATO: An efficient intelligent algorithm for time optimization in an economic grid based on adaptive stochastic Petri net.* Journal of Intelligent Manufacturing, 2015. **26**(4): p. 641-658.
19. CPN Group. *CPN Tools*. 2012 [cited 2017 4th September]; Available from: http://cpntools.org/.
20. Chan, V.-I. and Y.-W. Si, *Generic Evolutionary Framework for Simulating Business Processes U- and E-Service, Science and Technology*, T.-h. Kim, et al., Editors. 2011, Springer Berlin Heidelberg. p. 31-41.
21. Yoo, T., *Supply Chain Simulation using Business Process Modeling in Service Oriented Architecture.* International Journal of Information Systems and Supply Chain Management, 2015. **8**(4): p. 30-43.
22. Bisogno, S., et al., *Combining modelling and simulation approaches: How to measure performance of business processes.* Business Process Management Journal, 2016. **22**(1): p. 56-74.

23. Rondini, A., et al., *Business Process Simulation for the Design of Sustainable Product Service Systems (PSS)*, in *Advances in Production Management Systems: Innovative Production Management Towards Sustainable Growth*, S. Umeda, et al., Editors. 2015, Springer-Verlag Berlin: Berlin. p. 646-653.
24. Heinrich, R., et al., *Integrating business process simulation and information system simulation for performance prediction.* Software and Systems Modeling, 2017. **16**(1): p. 257-277.
25. Panagos, E. and M. Rabinovich, *Reducing Escalation-Related Costs in WFMSs Workflow Management Systems and Interoperability*, A. Doğaç, et al., Editors. 1998, Springer Berlin Heidelberg. p. 107-128.
26. Aalst, W.M.P.v.d., M. Rosemann, and M. Dumas, *Deadline-based escalation in process-aware information systems.* Decis. Support Syst., 2007. **43**(2): p. 492-511.
27. Si, Y.-W., M. Dumas, and K.-L. Chan, *Evaluation of trade-offs between workflow escalation strategies.* Concurrent Engineering, 2014. **22**(1): p. 77-88.
28. Chan, K.-L., Y.-W. Si, and M. Dumas, *Simulation-Based Evaluation of Workflow Escalation Strategies*, in *Proceedings of the 2009 IEEE International Conference on e-Business Engineering*. 2009, IEEE Computer Society. p. 75-82.
29. Hagen, C.R.V., D. Ratz, and W. Stucky, *Simulation of business process improvement by a customised genetic algorithm*. Modelling and Simulation 2004, ed. C. Bobeanu. 2004. 252-256.
30. Li, C., et al., *Scheduling FMS problems with heuristic search function and transition-timed Petri nets.* Journal of Intelligent Manufacturing, 2015. **26**(5): p. 933-944.
31. Li, T.P., et al., *Optimization for Manufacturing Process Based on Timed Petri Net and Genetic Algorithm*, in *Proceedings of the 2016 4th International Conference on Machinery, Materials and Computing Technology*, J. Zhu and G. Yao, Editors. 2016. p. 638-645.
32. Dezani, H., et al., *Optimizing urban traffic flow using Genetic Algorithm with Petri net analysis as fitness function.* Neurocomputing, 2014. **124**: p. 162-167.
33. Yao, A.W.L. and Y.M. Pan, *A Petri Nets and Genetic Algorithm Based Optimal Scheduling for Job Shop Manufacturing Systems*, in *Ieee International Conference on System Science and Engineering*, A. Szakal, Editor. 2013. p. 99-104.
34. Caballero-Villalobos, J.P., G.E. Mejia-Delgadillo, and R.G. Garcia-Caceres, *Scheduling of complex manufacturing systems with Petri nets and genetic algorithms: a case on plastic injection moulds.* International Journal of Advanced Manufacturing Technology, 2013. **69**(9-12): p. 2773-2786.
35. Mejia, G., et al., *Petri nets and genetic algorithms for complex manufacturing systems scheduling.* International Journal of Production Research, 2012. **50**(3): p. 791-803.
36. Sharda, B. and A. Banerjee, *Robust manufacturing system design using multi objective genetic algorithms, Petri nets and Bayesian uncertainty representation.* Journal of Manufacturing Systems, 2013. **32**(2): p. 315-324.

37. Zhang, T., et al., *Maintenance Scheduling for multi-unit system: a Stochastic Petri-net and genetic algorithm based approach* Eksploatacja I Niezawodnosc-Maintenance and Reliability, 2012. **14**(3): p. 256-264.
38. Hung, W., et al., *Modeling, scheduling, and prediction in wafer fabrication systems using Queueing Petri Net and Genetic Algorithm*, in *2001 Ieee International Conference on Robotics and Automation, Vols I-Iv, Proceedings*. 2001. p. 3559-3564.
39. Ang, L. and J.Y. Pu, *Damaged Ship Anti-flooding Decision Plan Intelligent Generation System Based on Petri Net and Heuristic Color Genetic Algorithm*, in *Mechatronics and Intelligent Materials Iii, Pts 1-3*, R. Chen, W.P. Sung, and J.C.M. Kao, Editors. 2013. p. 1866-1870.
40. Narendra, K.S. and M.A.L. Thathachar, *Learning Automata - A Survey.* IEEE Transactions on Systems, Man, and Cybernetics, 1974. **SMC-4**(4): p. 323-334.
41. Safari Mamaghani, A., K. Asghari, and M.R. Meybodi, *Designing a New Structure Based on Learning Automaton to Improve Evolutionary Algorithms (With Considering Some Case Study Problems).* Journal of Advances in Computer Research, 2013. **4**(3): p. 1-24.
42. Pearce-Moses, R. *SAA: A Glossary of Archival and Records Terminology, The Society of American Archivists*. 2012 [cited 2017 4th September]; Available from: http://www.archivists.org/glossary/index.asp.

11. APPENDIX A

In the level-0 DFD depicted in Figure 29, we use different colors to distinguish the functions designed for the GA framework;

- Yellow: Parameters to be defined by the user.
- Green: Chromosome generation.
- Blue: Fitness function.
- Orange: Selection.
- Light blue: Crossover.
- Grey: Mutation.

First, the program accepts inputs from the user to configure the GA. For instance, the user can configure the population size, the number of resource types, the number generations needed for the GA, the selection rate, and the elimination rate. Next, the algorithm randomly generates chromosomes based on the population size, tasks to be assigned, and the resource bounds associated to each resource type in the process model. In the proposed model, there is no limit on the number of resource types for the simulation. After the chromosomes are generated, they are used as resource allocation schemes by process models for simulation. Once the process simulation is completed, the fitness of each chromosome (resource allocation scheme) can be calculated. Each chromosome is then sorted based on the fitness value. The new generation of chromosomes are generated after performing selection, crossover and mutation operations on the selected chromosomes.

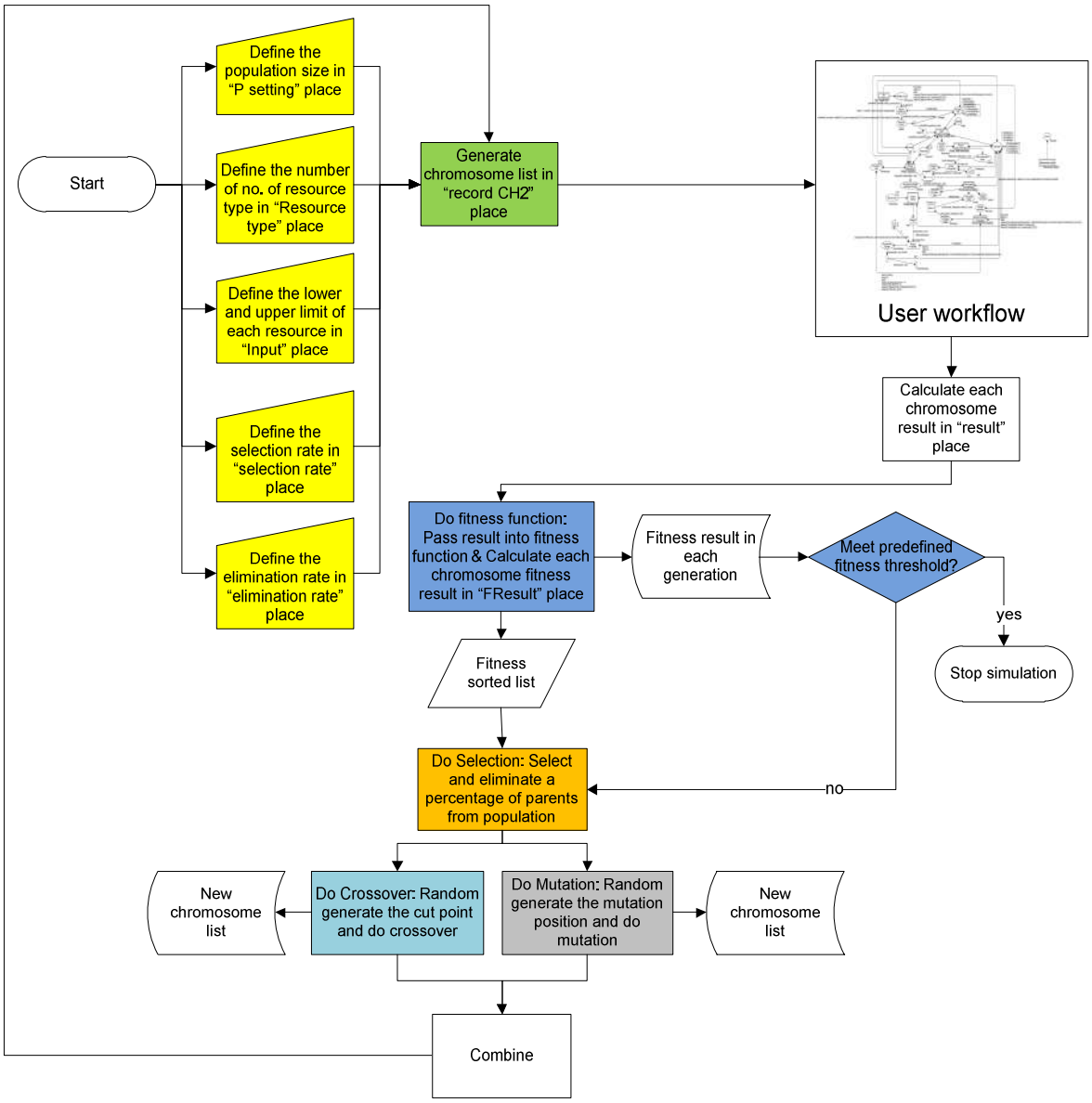

Figure 29. The Data Flow Diagram (DFD) representation of the GGA framework

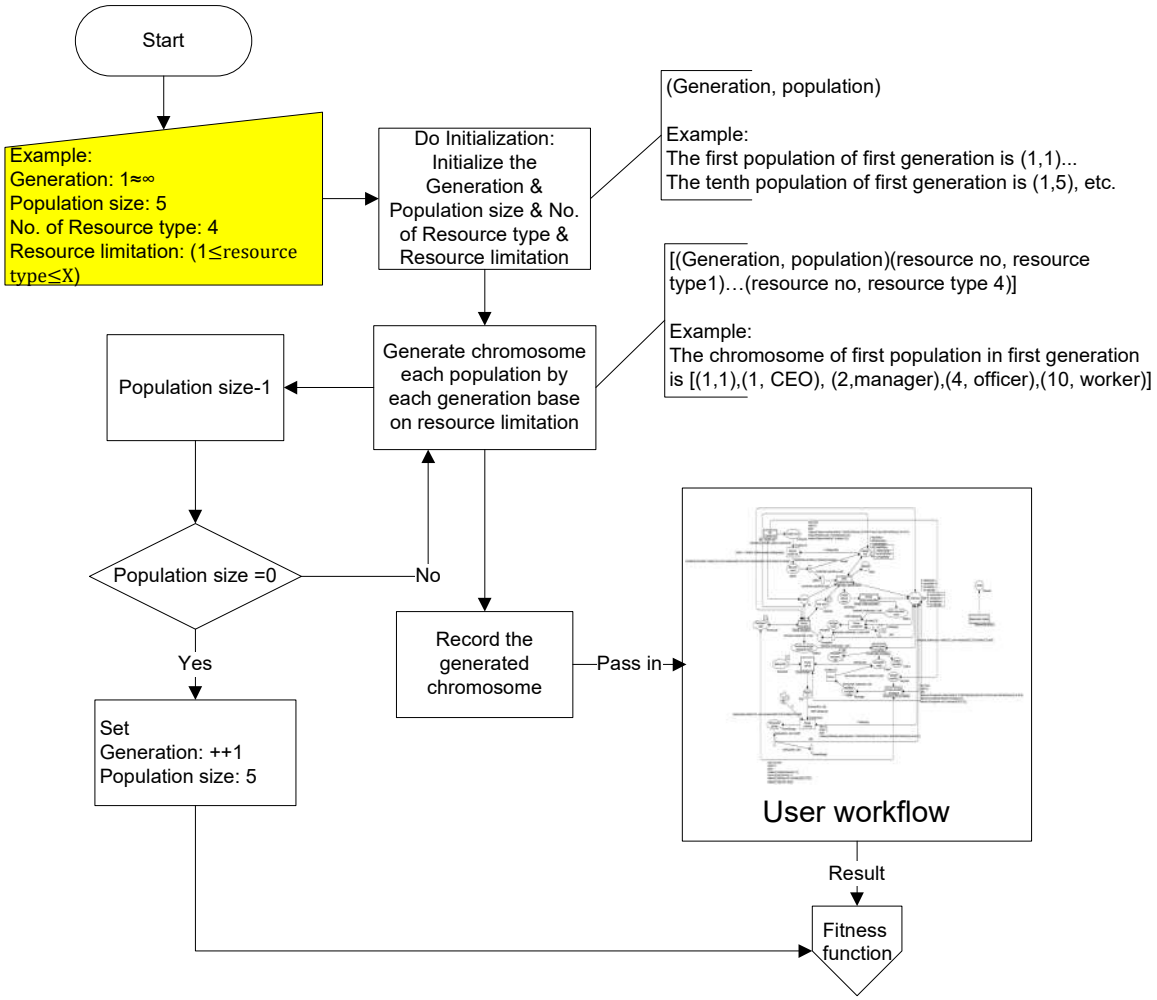

Figure 30. Chromosome Generation

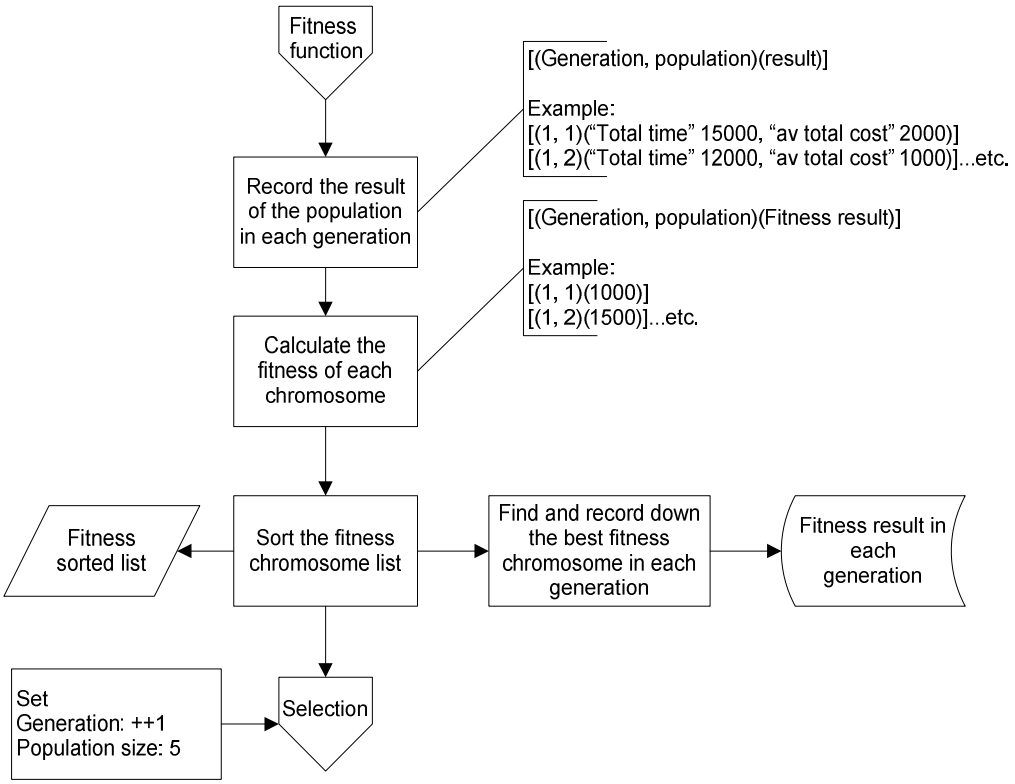

Figure 31. Fitness function

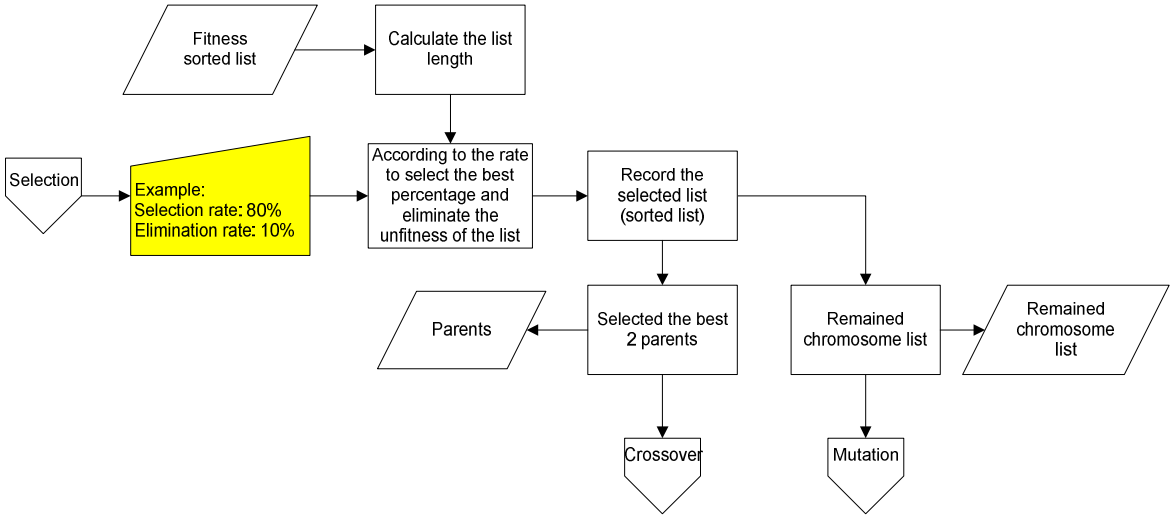

Figure 32. Selection operation

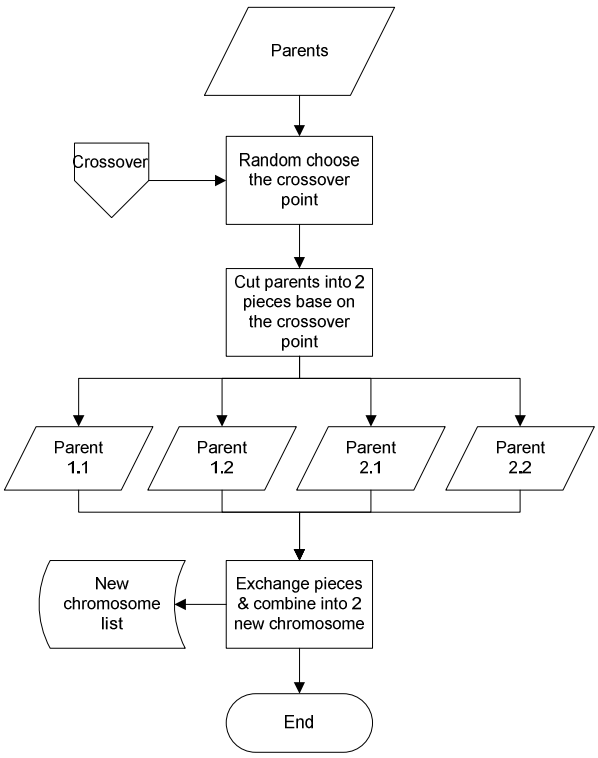

Figure 33. Crossover operation

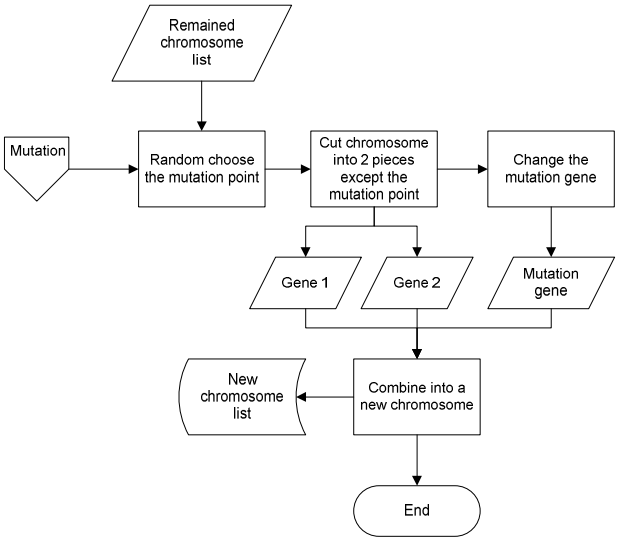

Figure 34. Mutation operation