


Please cite the Published Version

Li, SL, Huang, Z, Han, L  and Jiang, C (2018) A genetic algorithm enhanced automatic data flow management solution for facilitating data intensive applications in the cloud. *Concurrency and Computation: Practice and Experience*, 30 (23). e4844. ISSN 1532-0626

DOI: <https://doi.org/10.1002/cpe.4844>

Publisher: Wiley online

Version: Accepted Version

Downloaded from: <https://e-space.mmu.ac.uk/621405/>

Additional Information: This is an Author Accepted Manuscript of a paper accepted for publication in *Concurrency and Computation: Practice and Experience*, published by and copyright Wiley.

Enquiries:

If you have questions about this document, contact openresearch@mmu.ac.uk. Please include the URL of the record in e-space. If you believe that your, or a third party's rights have been compromised through this document please see our Take Down policy (available from <https://www.mmu.ac.uk/library/using-the-library/policies-and-guidelines>)

**A Genetic Algorithm Enhanced Automatic Data Flow
Management Solution for Facilitating Data Intensive
Applications in the Cloud**

| | |
|-------------------------------|---|
| Journal: | <i>Concurrency and Computation: Practice and Experience</i> |
| Manuscript ID | CPE-18-0414.R1 |
| Editor Selection: | Special Issue Submission |
| Wiley - Manuscript type: | Special Issue Paper |
| Date Submitted by the Author: | 07-Jun-2018 |
| Complete List of Authors: | Li, Siguang Huang, Zhengwen; Brunel University, Han, Liangxiu; Manchester Metropolitan University, School of Computing, Mathematics and Digital Technology |
| Keywords: | cloud computing, genetic algorithm, Petri Net, data flows, computation cost optimization |
| | |

SCHOLARONE™
Manuscripts

A Genetic Algorithm Enhanced Automatic Data Flow Management Solution for Facilitating Data Intensive Applications in the Cloud

Siguang Li^{1,2}, Zhengwen Huang³, Liangxiu Han⁴ and Changjun Jiang¹

¹The Key Laboratory of Embedded Systems and Service Computing, Tongji University, Shanghai, China

²College of Electrical Engineering, Binzhou University, Binzhou, Shandong, China.

³Department of Electronic and Computer Engineering, Brunel University London, Uxbridge, UB8 3PH, UK.

⁴School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, M1 5GD, UK.

Abstract

The past few years have witnessed a rapid deployment of computing infrastructures in the cloud in support of data intensive applications. The effort of the existing works is mainly focused on data reusing mechanisms without considering data processing routes which can significantly affect the computation costs when exchanging data among the computing node in the cloud. This paper presents a genetic algorithm enhanced Automatic Data Flow Management Solution (ADFMS) which facilitates automatic routing function and a self-adjustable intermediate data management mechanism to achieve an efficient data processing structure of cloud computing. Experimental results show that ADFMS optimizes costs in managing intermediate data in the cloud.

Keywords – cloud computing, genetic algorithm, Petri Net, data flows, computation cost optimization.

1. Introduction

With the rapid development of information technology in the past decades, the complexity of data processing work on the cloud network has increased significantly. Due to the limited computational capability of a target cloud network, the delivery efficiency of data intensive application significantly relies on the management of processing cost during the delivery procedure of a computing task [1-6]. A high delivery efficiency might be achieved if the total cost of data storage and data generation is well maintained at certain time during the data processing procedure. The data storage cost and the data generation cost are mainly restricted by the setting of data processing route and the intermediate data reuse. The data processing route is the topology of all involved computation stations for a given target data intensive application in the cloud network. The intermediate data reuse is about storing the intermediate data which are generated during the execution of current data intensive application for the execution of next task. As a result, balancing the cost generated by the change of data processing route and the cost generated by the deletion, preservation and regeneration of intermediate data in order to optimize the delivery efficiency of data intensive application on cloud network becomes a cutting-edge research challenge.

Because of the directed graphical presentation structure Petri Net [7] is employed to optimize the delivery efficiency of data processing task on cloud network. It is designed to provide a graphical work-flow-presentation solution which illustrates the delivery progress of a target job [8]. It has been widely applied for the modelling and optimization works of concurrency computation field. In [9] a Timed Petri Net

(TPN) was proposed to model the time related activities at workstation level. The manufacturing process of a target item is analyzed with the time spent on its corresponding component production steps. The model provides an adjustable platform for the time management of production process. In [10] an adjustable management mechanism is equipped with price parameters to create a Priced Timed Petri Net (PTPN). It considers event timing, real-time constraints, and computation costing together. This characteristic makes it particular suitable for the optimization work of data intensive application in cloud computing area. In work [11], [12] the concept of PTPN model is further extended to investigate the cost of large-scale scientific workflows in cloud network. The obtained results indicate that the efficiency of intermediate data storage mechanism could significantly influence the delivery efficiency of the target data intensive application on cloud platform. Based on that result, Augmented Petri Nets Cost Model [11] was introduced for the optimization of large bioinformatics work flows. This work well solves the performance optimization problem of cloud network with a well-designed data reusing mechanism.

However, data processing routes are neglected when the majority of attention has been only put on building the data reusing mechanisms in the previous mentioned works. The selection of route could reflect to the performance of intermediate data reuse. In other words, the selection of route may significantly limit the potential improvement space of delivery efficiency for data intensive application on a target cloud network. In precious works, the processing route and the reuse of intermediate data are not considered simultaneously to ensure the cloud network working in an efficient manner. The static implementation of route selection limits the potential optimization achievement which could be provided by a good management of intermediate data. In order to address this problem, this paper presents an Genetic Algorithm(GA) [13] enhanced Automatic Data Flow Management Solution (ADFMS). ADFMS facilitates automatic routing function and a self-adjustable intermediate data management mechanism to achieve an efficient dynamic data processing structure of cloud computing. The data processing route and intermediate data reuse are considered simultaneously in proposed solution. In order to achieve the max level of efficiency, a dynamic balance between processing route and cost can be maintained by the GA enhanced ADFMS. The contributions of this paper are listed below.

- It presents a novel GA bitstream like route representation structure for facilitating data intensive applications in a cloud network. It maintains the topology of routes and the setting of each involved node. It is used to construct and determine the best route of a given task on a cloud network.
- It provides a dynamical balanced data flow management mechanism which consider the route selection and intermediate data reuse simultaneously.
- It further proposes an optimizable data flow management solution which is driven by a Petri Net based fitness evaluation mechanism. This solution can provide simultaneous adjustment of cost contribution provided by the change of target processing route and the cost contribution generated by the deletion, preservation and regeneration of intermediate data.

The rest of this paper is organized as follows. Section 2 provides a general review of previous related work. Section 3 introduces a Genetic Algorithms enhanced Automatic Data Flow Management Solution. Section 4 illustrates an experiment which verifies the performance of proposed GA enhanced ADFMS. Section 5 concludes the paper and points out some potential future investigation directions.

2. Related Work

The original GA uses a fixed length string to represent the solution to a given problem. The simplest representation of GA chromosome is a fixed stream of zeros and ones (0 and 1). The chromosomes are used to maintain the information generated from a target system. With a well-structured exploring mechanism of searching space, GA has been widely applied for optimization problems [14]. The solution of optimization problem is considered as a point in a X-dimensions space. The X represents the number of involved factors in a target problem. A number of bit stream like chromosomes are created to represent a group of points in such space. Each chromosome maintains the coordinate information of a point in the space. At the end of the evolution procedure, the best chromosome containing the coordinate of the near or best point is found. The best solution to the target problem is then further extracted from such best chromosome.

The GA based optimization solutions mainly use chromosome to maintain the obtained coordinate information and to perform searching operations. The focus of search is set on (a) specific value(s) of involved factor(s). In work [15], GA was also applied to evolve the structure of Petri Nets itself. With the inhibitor arcs, this extended version of Petri Nets even was further developed as a Genetic Programming (GP) [16] like platform. The traditional parse tree of GP is replaced with the directed graph format representation of Petri Nets. This work combines the evolutionary algorithm concept into the Petri Net and empowers it with self-adapt ability. This kind of combination inspires us that the GA can also be employed to further explore the target space in aspect of route selection.

In this paper we take a GA bitstream like chromosome to represent the processing route of a dataset in a target cloud network. During the evolution process the processing route is refined by accumulating subnets which provide positive feedback to the target requirement (i.e. time or money-oriented cost fitness function). At the end of the evolution process, the information of an optimal route can be generated from the best chromosome to build a near ideal route.

2.1 Augmented Petri Nets Cost Model

Petri Nets was employed in many previous works [17-20] to build a representation platform for work optimization of cloud network. In our previous works [11] [12], the Augmented Petri Nets Cost Model (APNCM) was proposed to provide an evaluation and optimization platform for the reuse of intermediate data in cloud network. Our objective was focused on data management work of data intensive application (DIA) in cloud network. A DIA was modeled with the directed graph representation structure of Petri Net. Two types of nodes, "Place" nodes and "Transition" nodes, were employed to create a data processing application route. The "Transition" and the "Place" nodes present the processing elements of cloud network and their input/outputs respectively. The "Edge" was used to connect the two types of nodes. It is worth noting that only different types of nodes can be connected with a "Edge". The "Edge" also defines the direction of a transmission. Based these components, a Petri Nets can be created to represent the work flow of a DIA graphically if a processing sequence of such DIA task is given.

The cost model part [12] was designed to minimal the storage cost and computing cost. Based on the route presented with Petri Net, the total cost for a data intensive application in a time duration from t_0 to t_n can be calculated with (1).

$$Cost_{DIA} = \int_{t_0}^{t_n} \sum_{i=1}^n CostRateD_i * d_t \quad (1)$$

$CostRateD_i$ is the unit cost of the i^{th} dataset. The $Cost_{DIA}$ includes contributions from the storage cost of dataset and the computing cost of dataset generation work. For the dataset storage cost rate, $CostRateD_i$ can be represented as

$$CostRateD_i = D_{size_i} * CostS \quad (2)$$

where D_{size_i} is the size of the i^{th} Dataset, the $CostS$ is the price for storing dataset.

For the generation cost rate, $CostRateD_i$ is expressed with

$$CostRateD_i = GCost_i / UseRateD_i \quad (3)$$

where $UseRateD_i$ is the time window between the usage of the intermediate dataset i . When the $GCost_i$ is used for the regeneration cost of an intermediate dataset, it is interpreted as

$$GCost_i = (T_i + \sum_{k=1}^q T_k) * CostC + \sum_{r=1}^u D_{size_r} * CostS \quad (4)$$

Where

- T_i is the i^{th} transition time cost.
- q is the number of detected intermediate datasets.
- $\sum_{k=1}^q T_k$ represents the total cost of regenerating intermediate datasets, the predecessors of the i^{th} dataset.
- $CostC$ is the price for computing cost.
- u is the number of stored intermediate dataset.
- $\sum_{r=1}^u D_{size_r} * CostS$ represents the total storage cost of the predecessors of the i^{th} dataset.

Considering (1), (2) and (3), the total cost of a DIA can be represented with (5)

$$Cost_{DIA} = \begin{cases} \int_{t_0}^{t_n} \sum_{i=1}^n (D_{size_i} * CostS) * d_t \\ \int_{t_0}^{t_n} \sum_{i=1}^n (GCost_i / UseRateD_i) * d_t \end{cases} \quad (5)$$

Eq. (5) can be further applied as an objective function of data processing optimization work. The cost of deletion and regeneration of intermediate datasets are adjusted with the output of (5) in order to achieve a minimal cost in terms of money or time.

The APNCM provides an ideal candidate of fitness function for the evolution procedure of GA part in the proposed GA enhanced ADFMS. The route extracted from the bitstream like chromosome mentioned in II.A can be further evaluated with the APNCM. The result of evaluation provides a feedback to the evolution process of GA enhanced ADFMS. Such feedback is used to guide the direction of evolution progress efficiently. The detail of proposed GA enhanced solution is presented in next section.

3. GA Enhanced Automatic Data Flow Management Solution

The GA enhanced Automatic Data Flow Management Solution are developed with GA and APNCM. A novel GA bitstream like chromosome is created to represent the route of target date processing task and the setting of each involved nodes simultaneously. The APNCM is employed to construct fitness function of evolution process. The four core parameters of our solution are introduced first.

3.1 Core components of Augmented Petri Nets Cost Model

There are four components are employed from original Petri Net model to create data processing route.

- Transition - The transition is a node where the dataset is processed. In a transition node the computation task is performed on dataset.
- Place - The place is a node where the data is stored. In a place node some in/output dataset(s) of different transmission is(are) stored.
- Arc - The Arc represents the direction of data flow. It is used to connect two nodes (transition and place node) in the Petri Net.
- Settings of nodes - The settings of node are profiles information of the nodes. The format of settings are numerical values.

If the detail information of a data processing work flow is given, with the above core components, a complete Petri Net can be constructed. In order to represent a data processing route in a cloud network we developed a GA bitstream like chromosome structure.

3.2 Chromosome Structure

In this section we introduce the chromosome structure of the GA enhanced ADFMS. In order to maintain and refine the detail construction information of a data processing route which is presented with a Petri Net, we employ a GA bitstream like structure. In the GA bitstream like chromosome structure, each component of a Petri Net is assigned with a unique genetic code which is used to represent its appearance on a Petri Net. Figure 1 shows an example of chromosome structure employed in our approach. This chromosome contains two functionality parts. The former part is designed to represent the topology of Petri Net and the latter maintains the settings of each parameter which are constant values. The details of each part are discussed below.

| | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|-----|-----|-----|-----|-----|-----|-----|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | ... | ... | ... | ... | ... | ... | ... | C | C | C | C | C | C | C | C |
|---|---|---|---|---|---|---|---|-----|-----|-----|-----|-----|-----|-----|---|---|---|---|---|---|---|---|

Figure 1. An example of chromosome.

• Transition Segment

Transition node in Petri Net model is a procedure which takes data input(s) and generates data output(s). In this paper we propose a binary fragment to represent a transition node. A transition fragment is identified by the number of its input(s)/output(s) and its connected space nodes. The number of input(s) and output(s) are represented with the same number of bits. The fragments in Figure 2 show an example

representing a transition that has up to four input(s) and four output(s). The first bitstream (4 bits) of the fragment represents the input(s) part of transition. An enabled connection to a place is represented with '1'. If there is no connection is established, the corresponding bit is set to '0'. The last 4 bits represents the output(s) part of transition. And the connection is set with same way as input(s) part. The transition segment of chromosome consists of a number of such single transition binary fragments.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

Figure 2. An example of the Transition binary fragment.

• Place Segment

Place node in Petri Net model is a platform where the input and out data(s) are stored. In this paper we consider place as an intermediate exchange platform (between two transitions). The appearance of a place depends on its neighbor transitions (a neighbor transition takes input from or sends output to such place). It is worth noting that the starting/ending place is a special case which only has one type of transition (only input/output) node connected.

Based on the size of cloud network, a place node is indexed with an ID number. The ID number of a place node is the allocation connector of transition nodes. Two transition nodes sharing same place node (same ID number) can also be allocated in the host cloud network with the ID number of their connector place node. The whole topology of data processing route can be further constructed with the location information provided by ID numbers. For example, if a transition node has an input from place node A and an output to place node B then we can locate the transition node between place node A and B. The ID numbers of place nodes are further coded into a binary fragment and put on the later part (just after transition segment) of the bitstream like chromosome. The number of bits needed in each binary fragment is depended on the size of target cloud. Figure 3 shows an example of place segment which contains two place binary fragments for two place nodes. The former one has four bits to represent a place node index with ID number 13 ($8 + 4 + 0 + 1$); the later four bits fragment represents a place node with ID number 4 ($0 + 4 + 0 + 0$).

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

Figure 3. An example of Place binary fragment.

• Arcs Segment

Arc in Petri Net model is a notation representing the movement direction of a dataset in the cloud network. It provides a connection between transition node and place node. In this paper we use the sequence of transition binary fragments and ID number of place node to represent the direction of a dataset movement. A transition fragment appears on the position which is located closer to the first position on the transition segment is always triggered before the later one. That means the directional function of arc can be represented with such sequence mechanism. As a result, the arc acutely becomes a virtual component of

our solution and does not need to appear on the chromosome physically. The connection functionality is achieved by using the information in the transition segment and the place segment corporately.

- **Arguments segment**

In an Augmented Petri Net Cost Model (APNCM), the efficiency of a cloud computing platform is considered with two cost contribution factors, the computation cost and the storage cost. The computation cost depends on the computing time and the rent needed to finish a target task. The storage cost is based on the leasing time and the price of the cloud storage to finish a target data processing job.

In the proposed GA enhanced ADFMS, we take four cost sensitive factors into the account of the total cost. The four factors of a transition node on a target cloud network are listed below:

- (1) the computing times

The computing time is the CPU time needed to finish a given target computation job. If it is set for route detection purpose, its value depends on the complexity of the given task. If it is set for optimization purpose, the value of computing time depends on the hardware specification of target computation platform. It is also can be defined by user as a configuration parameter.

- (2) the unit price of computation platform

The unit price of computation platform is the price needed for leasing the target computation platform. Its value is can be defined by user as a configuration parameter of target cloud network.

- (3) the unit price of cloud storage

This unit price is the cost paid for the incurred intermediate cloud storage. Its value is also can be defined by user as a configuration parameter of target cloud network.

- (4) the intermediate data size

The intermediate data size is the size of dataset which is generated with transition nodes on a cloud network. Its value depends on the task which is under processing. If it is set for route detection purpose, the value of intermediate data size depends on requirement of data processing task. If it is set for optimization purpose, its value depends on the bandwidth of target computation platform. It is also can be defined by user as a configuration parameter of target cloud network.

Since these four factors are constant values, following our previous work [21], instead of binary fragment we create a GEP [22] style constant segment to represent the corresponding settings of transition nodes. The arguments segment of chromosome consists of a number of constant fragments. A fragment consists of eleven constant elements. First three constant elements represent one setting of the first three factors mentioned above, the computing time, the unit price of computation platform, and the unit price of cloud storage respectively. The last eight elements represent the settings of the corresponding intermediate datasets of a transition node. In Figure 4, an arguments segment which contains one transition node is provided. The T1 part contains the first three factors. The T2 part contains eight constant values for intermediate datasets.

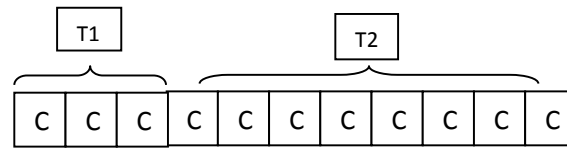


Figure 4. An example of Arguments segment.

It is worth noting that only transition node has its corresponding fragment in the arguments segment. The number of fragments in arguments segment is same as the number of transition nodes contained in the whole chromosome. A full chromosome of our solution is provided in Figure 5. The transition segment, the places segment and the arguments segment are linked sequentially. The genetic operator of ordinary GA, crossover and mutation are applied to provide variation on the transition and place segment of chromosome. The constant segment is operated with the classical GEP constant operator.

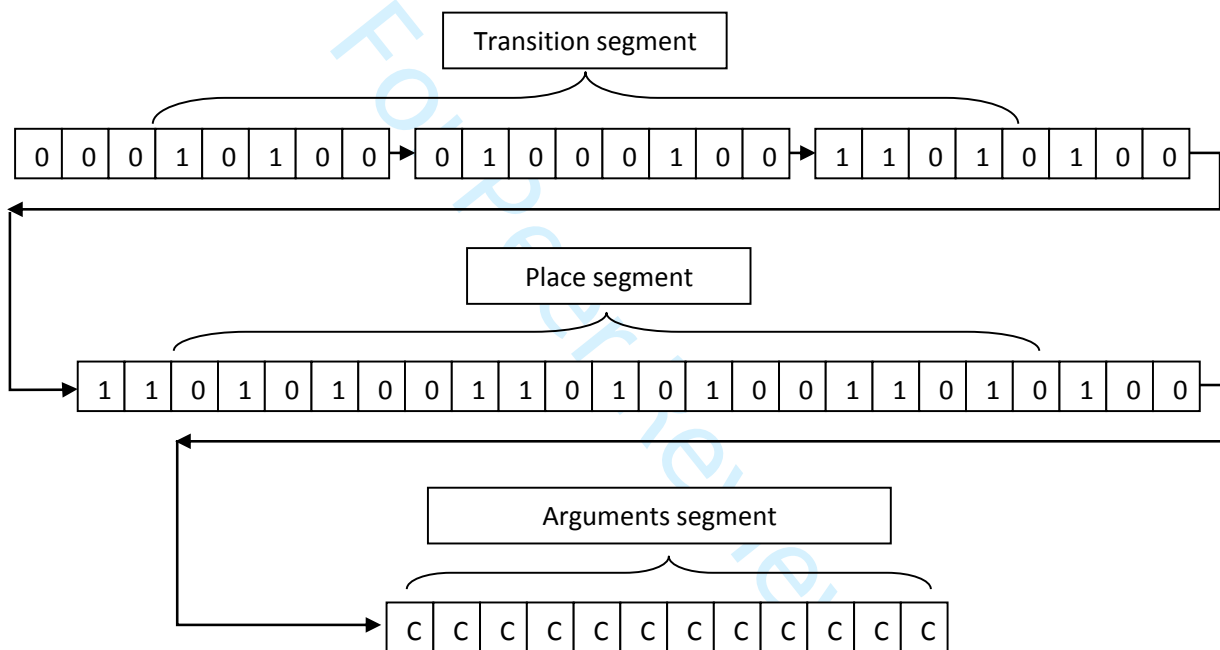


Figure 5. Full chromosome structure.

3.3 Decoding Mechanism

In ADFMS we use GA bitstream like chromosome to maintain an image of route of data processing task in a cloud network. This image contains the topology of a solution route and its corresponding constant argument settings. In order to generate such image, a decoding mechanism which extracts the information from chromosome to build a solution of a given data processing task in cloud network was developed. With such mechanism a complete Petri Net model which contains topology and corresponding constant argument setting of every node of the solution route can be extracted from the chromosome. The detail algorithm of decoding process is presented in Algorithm 1.

Input: A chromosome;

Output: An image of route of data processing task in a cloud network;

```

1: FOR x=1 TO number of transitions DO
2:   Get a transition fragment transition_x;
2:   Check sender(s) Segment information of the transition_x;
3:   Set sender Place(s) list;
4:   Check receiver(s) Segment information of transition_x;
5:   Set receiver Place(s) list;
6:   Get the constant fragment of transition_x to extract the constant value(s) to fill sender place(s) list and receiver place(s) list;
7:   Build node transition_x on petri net model;
8:   x++;
9: ENDFOR;
10: Return a complete petri net;

```

Algorithm 1: Decoding process implementation.

3.4 Fitness Function Design

The fitness function of GA enhanced ADFMS is based on the Augmented Petri Nets Cost Model. It uses the same calculation mechanism to evaluate the performance of a solution provided by GA part of ADFMS. The performance evaluation can be delivered with different target objectives. The route detection oriented fitness function is designed to provide a feasible path of target data processing task. The cost oriented fitness function gives user the cheapest solution (in terms of money or time spent on data processing task) by balancing the cost on computing and generation of the intermediate dataset.

For route detection purpose:

$$\begin{aligned}
 & \text{FitneesValue} = \\
 & \sum_{i=1}^n \left(\frac{\left(\begin{array}{l} \text{the no. of fitted sequences} \\ \text{the no. of transitions, correct work seuqence} \\ 0, invalide node} \end{array} \right)}{\left(\begin{array}{l} \text{the no. of transitions, correct work seuqence} \\ 0, invalide node} \end{array} \right)} \right)
 \end{aligned} \tag{6}$$

For optimization purpose:

$$\text{FitneesValue}_{time} = \sum_{i=0}^n Tcomp_i \tag{7}$$

$$\text{FitneesValue}_{cost} = \left\{ \begin{array}{l} \int_{t_0}^{t_n} \sum_{i=1}^n (D_{s_i} * CostS) * d_t \\ \int_{t_0}^{t_n} \sum_{i=1}^n (GCost_i / UseRateD_i) * d_t \end{array} \right. \tag{8}$$

3.5 Evolution Procedure

The evolution procedure of GA enhanced ADFMS is designed to generate a solution providing a near ideal performance. With the chromosome structure mentioned in section 3.2, many versions of the delivery route and the setting of all involved nodes for a data intensive application (i.e. chromosome) are encoded into bitstream to create the initial generation of evolution. During the evolution process, by accumulating bitstream segments which provide positive feedback to the target requirement (i.e. the fitness functions presented in 3.4), chromosomes are evolved generation by generation. At the end of procedure, based on the selection of fitness function a near ideal candidate solution can be generated from the best chromosome. The solution includes a best route or the most cost-efficient setting of a data intensive application in a target cloud network.

4 Performance Evaluation

To evaluate the performance of the GA enhanced Automatic Data Flow Management Solution, a number of experiments were designed to validate its chromosome structure for the presentation of Petri Net model and its optimization performance on a data intensive application case.

4.1 Dataset

A real data intensive application task biomedical application [23] [24] [25] is employed as our benchmark input task. The detail of the input task is provided in Table 1.

Table 1. A data intensive application task information.

| Task id | Dataset size(GBs) | Computing time(hours) | Task description |
|---------|-------------------|-----------------------|---|
| 1 | 78 | 1.9 | Image Processing |
| 2 | 60 | 1.2 | Feature Generation |
| 3 | 30 | 0.7 | Feature Selection and Extraction |
| 4 | 30 | 0.6 | |
| 5 | 0.1 | 0.8 | Classifier Construction and prediction evaluation |
| 6 | 5 | 0.5 | |
| 7 | 5 | 0.5 | |
| 8 | 0.4 | 0.1 | Cost Controlling feedback |

Figure 6 shows a Petri Net representation of the input task presented in our previous works [23][24]. The Petri Net is selected as a target route to validate the route selection functionality of our solution.

In Figure 6, the PE_x is the x^{th} transition node and the O_x represents x^{th} place node. This data processing task is investigated with two aspects, route detection and optimization for cost related problem.

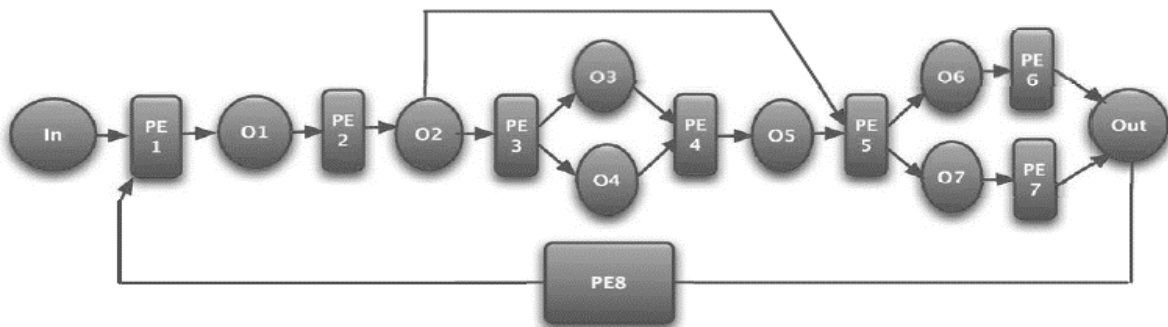


Figure 6. The target Petri Net representation.

4.2 Parameter Settings

The settings of GA evolution process are listed in Table 2. Due to the GEP style constant operation is involved, the parameters were set with the classical values used for a traditional GEP.

Table 2: GA evolution parameter settings.

| Parameters | Values | |
|-----------------------|------------------------|-------|
| Population size | 1000 | |
| Number of Generation | 20000 | |
| Genetic modifications | Cross rate | 30% |
| | Mutation | 0.44% |
| | Constant Mutation rate | 10% |
| Constant | High boundary | -1000 |
| | Low boundary | +1000 |

Since the purpose of evaluation experiment is to verify the concept oriented performance of our GA enhanced ADFMS, we select a PC with an ordinary specification to perform evolution experiments. The specification of benchmark platform is listed in Table 3.

Table 3: The benchmark platform.

| | | |
|------------------|------------------|----------------------|
| CPU | Model | Intel core i7-4700mq |
| | No. of Cores | 4 |
| | No. of Threads | 8 |
| | Frequency | 2.4G |
| Memory | 8 GB | |
| Operation system | Ubuntu 16.04 LTS | |

4.3 Route Detection Analysis

In order to evaluate the performance of the proposed GA bitstream like chromosome for route detection purpose. Ten executions of the GA enhanced ADFMS were conducted with the input data intensive application. With the data size, the sequence of task, the computing time of task and the route detection

fitness function, the GA enhanced ADFMS generate a chromosome containing the route information. Since the size of chromosome is too big to be demonstrated, only part of the best chromosome discovered in the evolution process is listed in Figure 7 showing the first three transition fragments and their corresponding place fragments. Based on the representation mechanism of the GA enhanced ADFMS, we can generate a part of Petri Net from the above chromosome segment as shown in Figure 8. T_x represents Transition node x , P_x is Place node x and $P0$ is the start Place node.

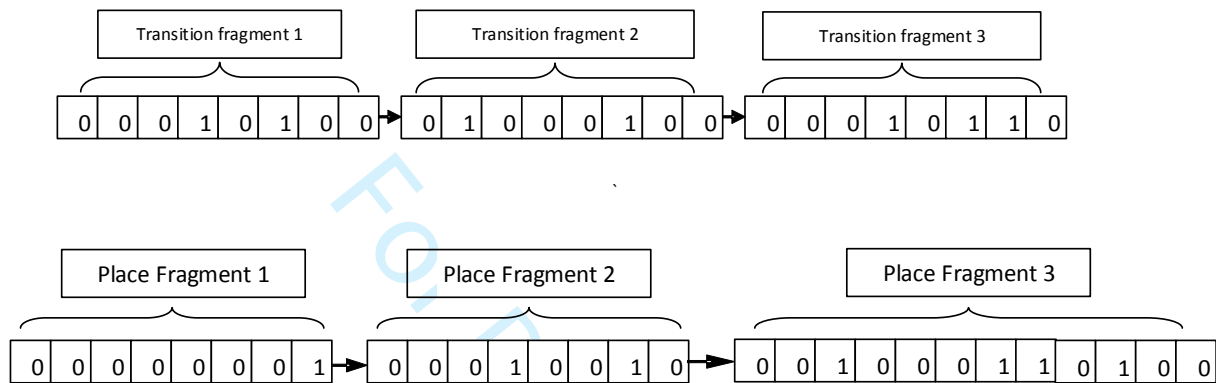


Figure 7. Part of the transition segment and place segment.

As shown in Figure 8, an identical part of the original Petri Net shown in Figure 7 can be extracted from the segment of chromosome generated with the GA enhanced ADFMS. It is worth noting that the constant segment was set to default value (integer value 1). The reason is that the evolution procedure was set for route detection purpose.

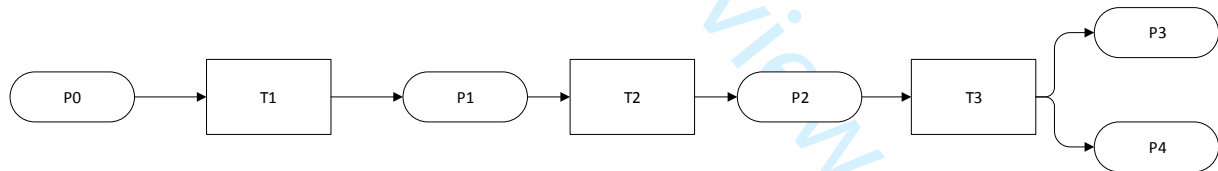


Figure 8. Part of the Petri Net.

With the setting of input dataset, the GA enhanced ADFMS can generate a route of target data intensive application automatically. Following the same setting, the routing work in previous works were done in a static manner. The processing route is predefined before the optimization work can be applied. That limits the potential space of optimization process. The dynamic solution of the GA enhanced ADFMS provides an adjustable route presentation structure. During the evolution process, this adjustable structure can refine the route for different target cloud networks setting scenarios in order to optimize the delivery efficiency of a data intensive application.

4.4 Optimization Results

In order to evaluate the performance of the proposed the GA enhanced ADFMS for optimization purpose, we follow the work [12] to investigate the intermediate data reuse optimization problem in a cloud network. Three scenarios, keeping all intermediate dataset, regenerating all intermediate dataset and dynamically regenerating intermediate dataset, were considered. Ten executions of ADFMS were conducted for the three scenarios. The minimalization of the cost caused by intermediate dataset was set as fitness function for the scenario dynamic regenerate intermediate dataset. The best results are listed in Table 4.

Table 4. Optimization results of ADFMS.

| Time (executions) | Keep all cost (\$) | Regenerate all cost (\$) | Optimization applied cost (\$) |
|-------------------|--------------------|--------------------------|--------------------------------|
| 1 | 0.875838889 | 0.875839 | 0.875838889 |
| 2 | 1.163636111 | 1.631839 | 0.892508897 |
| 3 | 1.619391667 | 2.387839 | 1.101186333 |
| 4 | 2.243105556 | 3.143839 | 1.480449667 |
| 5 | 3.034777778 | 3.899839 | 1.972605556 |
| 6 | 3.994408333 | 4.655839 | 2.476533167 |
| 7 | 5.121997222 | 5.411839 | 3.073198333 |
| 8 | 6.417544444 | 6.167839 | 3.669864139 |
| 9 | 7.88105 | 6.923839 | 3.704253806 |
| 10 | 9.512513889 | 7.679839 | 3.916717833 |

The optimization performance of the GA enhanced ADFMS was also compared with the Automatic Data Reuse Model with Petri Net (ADRMPN) [11]. A time unit (a period which is longer than a complete single execution of given data intensive application task) was selected as a checkpoint to observe the optimization performance. On each check point the speed up ratio of two algorithms are compared. As shown in Figure 9, the GA enhanced ADFMS performs similarly to the ADRMPN.

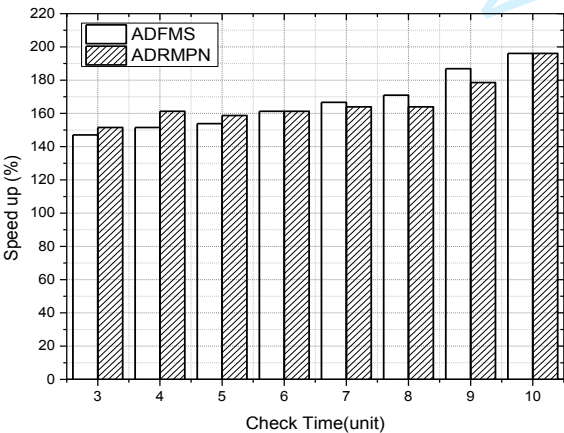


Figure 9. Optimization performance comparison.

It is worth noting that with the elapsed time, the speed up ratio increases, which means the more data requests received on the cloud network, the better optimization performance can be expected. Following the same setting of input dataset, the GA enhanced ADFMS still can achieve similar performance as previous works without a predefined route which are used in previous works. As mentioned in Section 4.3, the adjustable route presentation can maintain more potential space for optimization work. This is due to the interactive connection between the setting of cloud network and the selection of route is created with the GA enhanced ADFMS. Rather than optimizing on a predefined route, a dynamic balance between speed and cost is achievable with the proposed solution in this study.

5. Conclusion

In this paper we propose a novel Genetic Algorithm enhanced Automatic Data Flow Management solution for accelerating data intensive applications in cloud network. Instead of using a static predefined route which is used in previous works, the GA enhanced ADFMS operate a bitstream like chromosome to dynamically maintain the delivery route of a DIA in order to achieve a near maximal level of efficiency. It provides an automatic route detection solution for complex data intensive application. It also can be used to balance the cost generated by the change of target processing route and the cost generated by the deletion, preservation and regeneration of intermediate data in order to optimize the performance of data intensive application on the cloud computing platform.

Since the combination of GA and GEP chromosome structure is newly introduced in this work, the genetic operation efficiency of the proposed GA system is not well tuned. The application of genetic operation was based on classical GA or GEP implementation cases. Its performance can still be further improved to generate a faster and more accurate result at the convergence stage of evolution process.

Acknowledgment

This research is partially supported by the National Basic Research Program (973) of China under grant 2014CB340404 and the Science and Technology Commission of Shanghai Municipality under grant 16JC1401300.

References

- [1] J. Cao, D. P. Spooner, S. A. Jarvis, and G. R. Nudd, "Grid Load Balancing Using Intelligent Agents," *Future Generation Computer Systems*, vol. 21, Issue 1, January 2005.
- [2] C.A. Petri, "Kommunikation mit Automaten," Ph.D. thesis, University of Bonn, 1962.
- [3] C. Jie, D. Zhu, and B. Zhu. "Improved algorithms for intermediate dataset storage in a cloud-based dataflow." *Theoretical Computer Science* 657 (2017): 48-53.
- [4] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: the montage example," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, 2008, pp. 1-12.
- [5] D. Yuan, Y. Yang, X. Liu, G. Zhang, and J. Chen, "A data dependency based strategy for intermediate data storage in scientific cloud workflow systems," *CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE*, vol. 24, pp. 956-976, 2012.

- [6] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: the montage example," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, 2008, pp. 1–12.
- [7] J.L. Peterson, "Petri Nets," *Computing Surveys* 9(3), 1977, pp. 221–252.
- [8] M. Tadao. "Petri nets: Properties, analysis and applications." *Proceedings of the IEEE* 77.4 (1989): 541-580.
- [9] M. Gonzalo. "Timed Petri net modeling and optimization with heuristic search for flexible manufacturing workstations." *Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA'03. IEEE Conference*. Vol. 1. IEEE, 2003.
- [10] P. A. Abdulla, R. Mayr, "Petri Nets with Time and Cost (Tutorial)," *Proceedings 14th International Workshop on Verification of Infinite-State Systems*, Paris, France, 27th August 2012, pp. 9–24.
- [11] L. Han, Z. Xie, and B. Richard, "Automatic data reuse for accelerating data intensive applications in the Cloud." *Internet Technology and Secured Transactions (ICITST), 2013 8th International Conference for*. IEEE, 2013.
- [12] Z. Xie, L. Han, and B. Richard, "Augmented Petri Net Cost Model for Optimisation of Large Bioinformatics Workflows Using Cloud." *Modelling Symposium (EMS), 2013 European*. IEEE, 2013.
- [13] JH. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, U Michigan Press, 1975.
- [14] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, 1996.
- [15] M. Holger. "Evolving Petri nets with a genetic algorithm." *Genetic and Evolutionary Computation Conference*. Springer, Berlin, Heidelberg, 2003.
- [16] J. R. Koza, "Genetic Programming as a Means for Programming Computers by Natural Selection", *Stat. Comput.*, vol. 4, no. 2, pp. 87–112, 1994.
- [17] M. Gonzalo, et al. "Petri nets and genetic algorithms for complex manufacturing systems scheduling." *International Journal of Production Research* 50.3 (2012): 791-803.
- [18] N. Odrey, Y. Ma, "A Multi-Level Multi-Layer Petri Net Based Approach for Manufacturing Systems Control". *Proceedings of the 11th FAIM International Conference*, July 16-18, 2001. Dublin, Ireland.
- [19] D. Yuan, Y. Yang, X. Liu, G. Zhang, and J. Chen, "A data dependency based strategy for intermediate data storage in scientific cloud workflow systems," *CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE*, vol. 24, pp. 956–976, 2012.
- [20] Simon S. Woo, and M. Jelena, "Optimal application allocation on multiple public clouds." *Computer Networks* 68 (2014): 138-148.
- [21] Z. Huang, M. Li, C. Chousidis, A. Mousavi, & C. Jiang, "Schema Theory Based Data Engineering in Gene Expression Programming for Big Data Analytics," *IEEE Transactions on Evolutionary Computation*. DOI:10.1109/TEVC.2017.2771445, 2017.
- [22] C. Ferreira, "Gene Expression Programming: a New Adaptive Algorithm for Solving Problems", *Complex Systems*, vol.13, no.2, pp.22, 2001.
- [23] L. Han, J. I. van Hemert, and R. Baldock, "Automatically identifying and annotating mouse embryo gene expression patterns," *Bioinformatics*, vol. 27, no. 8, pp. 1101–1107, 2011.
- [24] L. Han and H.-Y. Ong, "Accelerating biomedical data-intensive applications using mapreduce," in *2012 ACM/IEEE 13th International Conference on Grid Computing (GRID)*, 2012, pp. 49 – 57.
- [25] Z. Xie, L. Han, and R. Baldock, "Enhancing parallelism of data-intensive bioinformatics applications," in *Proceedings of 8th EUROSIM Congress on Modelling and Simulation*, IEEE, 2013, pp. 519–524.

A Genetic Algorithm Enhanced Automatic Data Flow Management Solution for Facilitating Data Intensive Applications in the Cloud

Siguang Li^{1,2}, Zhengwen Huang³, Liangxiu Han⁴ and Changjun Jiang¹

¹The Key Laboratory of Embedded Systems and Service Computing, Tongji University, Shanghai, China

²College of Electrical Engineering, Binzhou University, Binzhou, Shandong, China.

³Department of Electronic and Computer Engineering, Brunel University London, Uxbridge, UB8 3PH, UK.

⁴School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, M1 5GD, UK.

Abstract

The past few years have witnessed a rapid deployment of computing infrastructures in the cloud in support of data intensive applications. The effort of the existing works is mainly focused on data reusing mechanisms without considering data processing routes which can significantly affect the computation costs when exchanging data among the computing node in the cloud. This paper presents a genetic algorithm enhanced Automatic Data Flow Management Solution (ADFMS) which facilitates automatic routing function and a self-adjustable intermediate data management mechanism to achieve an efficient data processing structure of cloud computing. Experimental results show that ADFMS optimizes costs in managing intermediate data in the cloud.

Keywords – cloud computing, genetic algorithm, Petri Net, data flows, computation cost optimization.

1. Introduction

With the rapid development of information technology in the past decades, the complexity of data processing work on the cloud network has increased significantly. Due to the limited computational capability of a target cloud network, the delivery efficiency of data intensive application significantly relies on the management of processing cost during the delivery procedure of a computing task [1-6]. A high delivery efficiency might be achieved if the total cost of data storage and data generation is well maintained at certain time during the data processing procedure. The data storage cost and the data generation cost are mainly restricted by the setting of data processing route and the intermediate data reuse. The data processing route is the topology of all involved computation stations for a given target data intensive application in the cloud network. The intermediate data reuse is about storing the intermediate data which are generated during the execution of current data intensive application for the execution of next task. As a result, balancing the cost generated by the change of data processing route and the cost generated by the deletion, preservation and regeneration of intermediate data in order to optimize the delivery efficiency of data intensive application on cloud network becomes a cutting-edge research challenge.

Because of the directed graphical presentation structure Petri Net [7] is employed to optimize the delivery efficiency of data processing task on cloud network. It is designed to provide a graphical work-flow-presentation solution which illustrates the delivery progress of a target job [8]. It has been widely applied for the modelling and optimization works of concurrency computation field. In [9] a Timed Petri Net

(TPN) was proposed to model the time related activities at workstation level. The manufacturing process of a target item is analyzed with the time spent on its corresponding component production steps. The model provides an adjustable platform for the time management of production process. In [10] an adjustable management mechanism is equipped with price parameters to create a Priced Timed Petri Net (PTPN). It considers event timing, real-time constraints, and computation costing together. This characteristic makes it particular suitable for the optimization work of data intensive application in cloud computing area. In work [11], [12] the concept of PTPN model is further extended to investigate the cost of large-scale scientific workflows in cloud network. The obtained results indicate that the efficiency of intermediate data storage mechanism could significantly influence the delivery efficiency of the target data intensive application on cloud platform. Based on that result, Augmented Petri Nets Cost Model [11] was introduced for the optimization of large bioinformatics work flows. This work well solves the performance optimization problem of cloud network with a well-designed data reusing mechanism.

However, data processing routes are neglected when the majority of attention has been only put on building the data reusing mechanisms in the previous mentioned works. The selection of route could reflect to the performance of intermediate data reuse. In other words, the selection of route may significantly limit the potential improvement space of delivery efficiency for data intensive application on a target cloud network. In precious works, the processing route and the reuse of intermediate data are not considered simultaneously to ensure the cloud network working in an efficient manner. The static implementation of route selection limits the potential optimization achievement which could be provided by a good management of intermediate data. In order to address this problem, this paper presents an Genetic Algorithm(GA) [13] enhanced Automatic Data Flow Management Solution (ADFMS). ADFMS facilitates automatic routing function and a self-adjustable intermediate data management mechanism to achieve an efficient dynamic data processing structure of cloud computing. The data processing route and intermediate data reuse are considered simultaneously in proposed solution. In order to achieve the max level of efficiency, a dynamic balance between processing route and cost can be maintained by the GA enhanced ADFMS. The contributions of this paper are listed below.

- It presents a novel GA bitstream like route representation structure for facilitating data intensive applications in a cloud network. It maintains the topology of routes and the setting of each involved node. It is used to construct and determine the best route of a given task on a cloud network.
- It provides a dynamical balanced data flow management mechanism which consider the route selection and intermediate data reuse simultaneously.
- It further proposes an optimizable data flow management solution which is driven by a Petri Net based fitness evaluation mechanism. This solution can provide simultaneous adjustment of cost contribution provided by the change of target processing route and the cost contribution generated by the deletion, preservation and regeneration of intermediate data.

The rest of this paper is organized as follows. Section 2 provides a general review of previous related work. Section 3 introduces a Genetic Algorithms enhanced Automatic Data Flow Management Solution. Section 4 illustrates an experiment which verifies the performance of proposed GA enhanced ADFMS. Section 5 concludes the paper and points out some potential future investigation directions.

2. Related Work

The original GA uses a fixed length string to represent the solution to a given problem. The simplest representation of GA chromosome is a fixed stream of zeros and ones (0 and 1). The chromosomes are used to maintain the information generated from a target system. With a well-structured exploring mechanism of searching space, GA has been widely applied for optimization problems [14]. The solution of optimization problem is considered as a point in a X-dimensions space. The X represents the number of involved factors in a target problem. A number of bit stream like chromosomes are created to represent a group of points in such space. Each chromosome maintains the coordinate information of a point in the space. At the end of the evolution procedure, the best chromosome containing the coordinate of the near or best point is found. The best solution to the target problem is then further extracted from such best chromosome.

The GA based optimization solutions mainly use chromosome to maintain the obtained coordinate information and to perform searching operations. The focus of search is set on (a) specific value(s) of involved factor(s). In work [15], GA was also applied to evolve the structure of Petri Nets itself. With the inhibitor arcs, this extended version of Petri Nets even was further developed as a Genetic Programming(GP) [16] like platform. The traditional parse tree of GP is replaced with the directed graph format representation of Petri Nets. This work combines the evolutionary algorithm concept into the Petri Net and empowers it with self-adapt ability. This kind of combination inspires us that the GA can also be employed to further explore the target space in aspect of route selection.

In this paper we take a GA bitstream like chromosome to represent the processing route of a dataset in a target cloud network. During the evolution process the processing route is refined by accumulating subnets which provide positive feedback to the target requirement (i.e. time or money-oriented cost fitness function). At the end of the evolution process, the information of an optimal route can be generated from the best chromosome to build a near ideal route.

2.1 Augmented Petri Nets Cost Model

Petri Nets was employed in many previous works [17-20] to build a representation platform for work optimization of cloud network. In our previous works [11] [12], the Augmented Petri Nets Cost Model (APNCM) was proposed to provide an evaluation and optimization platform for the reuse of intermediate data in cloud network. Our objective was focused on data management work of data intensive application (DIA) in cloud network. A DIA was modeled with the directed graph representation structure of Petri Net. Two types of nodes, “Place” nodes and “Transition” nodes, were employed to create a data processing application route. The “Transition” and the “Place” nodes present the processing elements of cloud network and their input/outputs respectively. The “Edge” was used to connect the two types of nodes. It is worth noting that only different types of nodes can be connected with a “Edge”. The “Edge” also defines the direction of a transmission. Based these components, a Petri Nets can be created to represent the work flow of a DIA graphically if a processing sequence of such DIA task is given.

The cost model part [12] was designed to minimal the storage cost and computing cost. Based on the route presented with Petri Net, the total cost for a data intensive application in a time duration from t_0 to t_n can be calculated with (1).

$$Cost_{DIA} = \int_{t_0}^{t_n} \sum_{i=1}^n CostRateD_i * d_t \quad (1)$$

$CostRateD_i$ is the unit cost of the i^{th} dataset. The $Cost_{DIA}$ includes contributions from the storage cost of dataset and the computing cost of dataset generation work. For the dataset storage cost rate, $CostRateD_i$ can be represented as

$$CostRateD_i = D_{size_i} * CostS \quad (2)$$

where D_{size_i} is the size of the i^{th} Dataset, the $CostS$ is the price for storing dataset.

For the generation cost rate, $CostRateD_i$ is expressed with

$$CostRateD_i = GCost_i / UseRateD_i \quad (3)$$

where $UseRateD_i$ is the time window between the usage of the intermediate dataset i . When the $GCost_i$ is used for the regeneration cost of an intermediate dataset, it is interpreted as

$$GCost_i = (T_i + \sum_{k=1}^q T_k) * CostC + \sum_{r=1}^u D_{size_r} * CostS \quad (4)$$

Where

- T_i is the i^{th} transition time cost.
- q is the number of detected intermediate datasets.
- $\sum_{k=1}^q T_k$ represents the total cost of regenerating intermediate datasets, the predecessors of the i^{th} dataset.
- $CostC$ is the price for computing cost.
- u is the number of stored intermediate dataset.
- $\sum_{r=1}^u D_{size_r} * CostS$ represents the total storage cost of the predecessors of the i^{th} dataset.

Considering (1), (2) and (3), the total cost of a DIA can be represented with (5)

$$Cost_{DIA} = \left\{ \begin{array}{l} \int_{t_0}^{t_n} \sum_{i=1}^n (D_{size_i} * CostS) * d_t \\ \int_{t_0}^{t_n} \sum_{i=1}^n (GCost_i / UseRateD_i) * d_t \end{array} \right. \quad (5)$$

Eq. (5) can be further applied as an objective function of data processing optimization work. The cost of deletion and regeneration of intermediate datasets are adjusted with the output of (5) in order to achieve a minimal cost in terms of money or time.

The APNCM provides an ideal candidate of fitness function for the evolution procedure of GA part in the proposed GA enhanced ADFMS. The route extracted from the bitstream like chromosome mentioned in II.A can be further evaluated with the APNCM. The result of evaluation provides a feedback to the evolution process of GA enhanced ADFMS. Such feedback is used to guide the direction of evolution progress efficiently. The detail of proposed GA enhanced solution is presented in next section.

The GA enhanced Automatic Data Flow Management Solution are developed with GA and APNCM. A novel GA bitstream like chromosome is created to represent the route of target date processing task and the setting of each involved nodes simultaneously. The APNCM is employed to construct fitness function of evolution process. The four core parameters of our solution are introduced first.

There are four components are employed from original Petri Net model to create data processing route.

- If the detail information of a data processing work flow is given, with the above core components, a complete Petri Net can be constructed. In order to represent a data processing route in a cloud network we developed a GA bitstream like chromosome structure.

In this section we introduce the chromosome structure of the GA enhanced ADFMS. In order to maintain and refine the detail construction information of a data processing route which is presented with a Petri Net, we employ a GA bitstream like structure. In the GA bitstream like chromosome structure, each component of a Petri Net is assigned with a unique genetic code which is used to represent its appearance on a Petri Net. Figure 1 shows an example of chromosome structure employed in our approach. This chromosome contains two functionality parts. The former part is designed to represent the topology of Petri Net and the latter maintains the settings of each parameter which are constant values. The details of each part are discussed below.

Figure 1. An example of chromosome.

Transition node in Petri Net model is a procedure which takes data input(s) and generates data output(s). In this paper we propose a binary fragment to represent a transition node. A transition fragment is identified by the number of its input(s)/output(s) and its connected space nodes. The number of input(s) and output(s) are represented with the same number of bits. The fragments in Figure 2 show an example

representing a transition that has up to four input(s) and four output(s). The first bitstream (4 bits) of the fragment represents the input(s) part of transition. An enabled connection to a place is represented with '1'. If there is no connection is established, the corresponding bit is set to '0'. The last 4 bits represents the output(s) part of transition. And the connection is set with same way as input(s) part. The transition segment of chromosome consists of a number of such single transition binary fragments.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

Figure 2. An example of the Transition binary fragment.

• **Place Segment**

Place node in Petri Net model is a platform where the input and out data(s) are stored. In this paper we consider place as an intermediate exchange platform (between two transitions). The appearance of a place depends on its neighbor transitions (a neighbor transition takes input from or sends output to such place). It is worth noting that the starting/ending place is a special case which only has one type of transition (only input/output) node connected.

Based on the size of cloud network, a place node is indexed with an ID number. The ID number of a place node is the allocation connector of transition nodes. Two transition nodes sharing same place node (same ID number) can also be allocated in the host cloud network with the ID number of their connector place node. The whole topology of data processing route can be further constructed with the location information provided by ID numbers. For example, if a transition node has an input from place node A and an output to place node B then we can locate the transition node between place node A and B. The ID numbers of place nodes are further coded into a binary fragment and put on the later part (just after transition segment) of the bitstream like chromosome. The number of bits needed in each binary fragment is depended on the size of target cloud. Figure 3 shows an example of place segment which contains two place binary fragments for two place nodes. The former one has four bits to represent a place node index with ID number 13 ($8 + 4 + 0 + 1$); the later four bits fragment represents a place node with ID number 4 ($0 + 4 + 0 + 0$).

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

Figure 3. An example of Place binary fragment.

• **Arcs Segment**

Arc in Petri Net model is a notation representing the movement direction of a dataset in the cloud network. It provides a connection between transition node and place node. In this paper we use the sequence of transition binary fragments and ID number of place node to represent the direction of a dataset movement. A transition fragment appears on the position which is located closer to the first position on the transition segment is always triggered before the later one. That means the directional function of arc can be represented with such sequence mechanism. As a result, the arc acutely becomes a virtual component of

our solution and does not need to appear on the chromosome physically. The connection functionality is achieved by using the information in the transition segment and the place segment corporately.

- **Arguments segment**

In an Augmented Petri Net Cost Model (APNCM), the efficiency of a cloud computing platform is considered with two cost contribution factors, the computation cost and the storage cost. The computation cost depends on the computing time and the rent needed to finish a target task. The storage cost is based on the leasing time and the price of the cloud storage to finish a target data processing job.

In the proposed GA enhanced ADFMS, we take four cost sensitive factors into the account of the total cost. The four factors of a transition node on a target cloud network are listed below:

(1) the computing times

The computing time is the CPU time needed to finish a given target computation job. If it is set for route detection purpose, its value depends on the complexity of the given task. If it is set for optimization purpose, the value of computing time depends on the hardware specification of target computation platform. It is also can be defined by user as a configuration parameter.

(2) the unit price of computation platform

The unit price of computation platform is the price needed for leasing the target computation platform. Its value is can be defined by user as a configuration parameter of target cloud network.

(3) the unit price of cloud storage

This unit price is the cost paid for the incurred intermediate cloud storage. Its value is also can be defined by user as a configuration parameter of target cloud network.

(4) the intermediate data size

The intermediate data size is the size of dataset which is generated with transition nodes on a cloud network. Its value depends on the task which is under processing. If it is set for route detection purpose, the value of intermediate data size depends on requirement of data processing task. If it is set for optimization purpose, its value depends on the bandwidth of target computation platform. It is also can be defined by user as a configuration parameter of target cloud network.

Since these four factors are constant values, following our previous work [21], instead of binary fragment we create a GEP [22] style constant segment to represent the corresponding settings of transition nodes. The arguments segment of chromosome consists of a number of constant fragments. A fragment consists of eleven constant elements. First three constant elements represent one setting of the first three factors mentioned above, the computing time, the unit price of computation platform, and the unit price of cloud storage respectively. The last eight elements represent the settings of the corresponding intermediate datasets of a transition node. In Figure 4, an arguments segment which contains one transition node is provided. The T1 part contains the first three factors. The T2 part contains eight constant values for intermediate datasets.

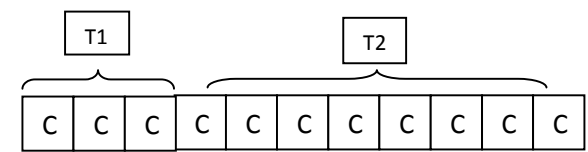


Figure 4. An example of Arguments segment.

It is worth noting that only transition node has its corresponding fragment in the arguments segment. The number of fragments in arguments segment is same as the number of transition nodes contained in the whole chromosome. A full chromosome of our solution is provided in Figure 5. The transition segment, the places segment and the arguments segment are linked sequentially. The genetic operator of ordinary GA, crossover and mutation are applied to provide variation on the transition and place segment of chromosome. The constant segment is operated with the classical GEP constant operator.

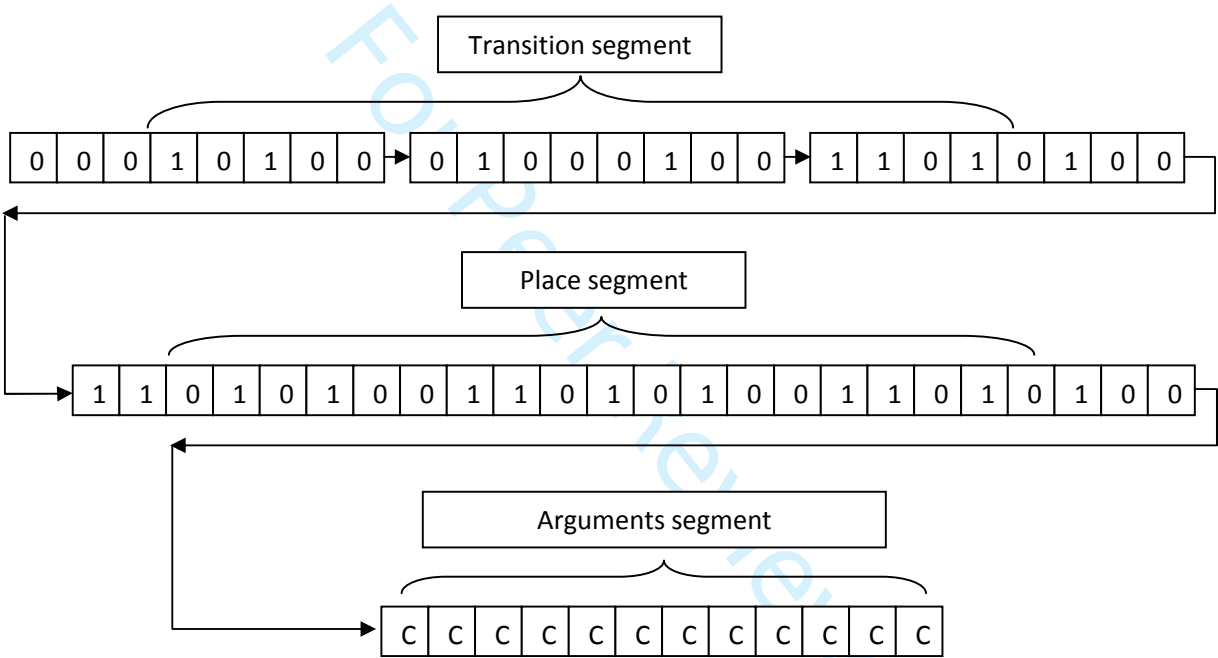


Figure 5. Full chromosome structure.

3.3 Decoding Mechanism

In ADFMS we use GA bitstream like chromosome to maintain an image of route of data processing task in a cloud network. This image contains the topology of a solution route and its corresponding constant argument settings. In order to generate such image, a decoding mechanism which extracts the information from chromosome to build a solution of a given data processing task in cloud network was developed. With such mechanism a complete Petri Net model which contains topology and corresponding constant argument setting of every node of the solution route can be extracted from the chromosome. The detail algorithm of decoding process is presented in Algorithm 1.

Input: A chromosome;

Output: An image of route of data processing task in a cloud network;

```

1: FOR x=1 TO number of transitions DO
2:   Get a transition fragment transition_x;
2:   Check sender(s) Segment information of the transition_x;
3:   Set sender Place(s) list;
4:   Check receiver(s) Segment information of transition_x;
5:   Set receiver Place(s) list;
6:   Get the constant fragment of transition_x to extract the constant value(s) to fill sender place(s)
   list and receiver place(s) list;
7:   Build node transition_x on petri net model;
8:   x++;
9: ENDFOR;
10: Return a complete petri net;

```

Algorithm 1: Decoding process implementation.

3.4 Fitness Function Design

The fitness function of GA enhanced ADFMS is based on the Augmented Petri Nets Cost Model. It uses the same calculation mechanism to evaluate the performance of a solution provided by GA part of ADFMS. The performance evaluation can be delivered with different target objectives. The route detection oriented fitness function is designed to provide a feasible path of target data processing task. The cost oriented fitness function gives user the cheapest solution (in terms of money or time spent on data processing task) by balancing the cost on computing and generation of the intermediate dataset.

For route detection purpose:

$$\begin{aligned}
 & \text{FitneesValue} = \\
 & \sum_{i=1}^n \left(\begin{cases} \text{the no. of fitted sequences} / \text{the no. of transitions, correct work seuquence} \\ 0, \text{ invalide node} \end{cases} \right)
 \end{aligned} \tag{6}$$

For optimization purpose:

$$\text{FitneesValue}_{time} = \sum_{i=0}^n Tcomp_i \tag{7}$$

$$\text{FitneesValue}_{cost} = \begin{cases} \int_{t_0}^{t_n} \sum_{i=1}^n (D_{s_i} * CostS) * d_t \\ \int_{t_0}^{t_n} \sum_{i=1}^n (GCost_i / UseRateD_i) * d_t \end{cases} \tag{8}$$

3.5 Evolution Procedure

The evolution procedure of GA enhanced ADFMS is designed to generate a solution providing a near ideal performance. With the chromosome structure mentioned in section 3.2, many versions of the delivery route and the setting of all involved nodes for a data intensive application (i.e. chromosome) are encoded into bitstream to create the initial generation of evolution. During the evolution process, by accumulating bitstream segments which provide positive feedback to the target requirement (i.e. the fitness functions presented in 3.4), chromosomes are evolved generation by generation. At the end of procedure, based on the selection of fitness function a near ideal candidate solution can be generated from the best chromosome. The solution includes a best route or the most cost-efficient setting of a data intensive application in a target cloud network.

4 Performance Evaluation

To evaluate the performance of the GA enhanced Automatic Data Flow Management Solution, a number of experiments were designed to validate its chromosome structure for the presentation of Petri Net model and its optimization performance on a data intensive application case.

4.1 Dataset

A real data intensive application task biomedical application [23] [24] [25] is employed as our benchmark input task. The detail of the input task is provided in Table 1.

Table 1. A data intensive application task information.

| Task id | Dataset size(GBs) | Computing time(hours) | Task description |
|---------|-------------------|-----------------------|---|
| 1 | 78 | 1.9 | Image Processing |
| 2 | 60 | 1.2 | Feature Generation |
| 3 | 30 | 0.7 | Feature Selection and Extraction |
| 4 | 30 | 0.6 | |
| 5 | 0.1 | 0.8 | Classifier Construction and prediction evaluation |
| 6 | 5 | 0.5 | |
| 7 | 5 | 0.5 | |
| 8 | 0.4 | 0.1 | Cost Controlling feedback |

Figure 6 shows a Petri Net representation of the input task presented in our previous works [23][24]. The Petri Net is selected as a target route to validate the route selection functionality of our solution.

In Figure 6, the PE_x is the x^{th} transition node and the O_x represents x^{th} place node. This data processing task is investigated with two aspects, route detection and optimization for cost related problem.

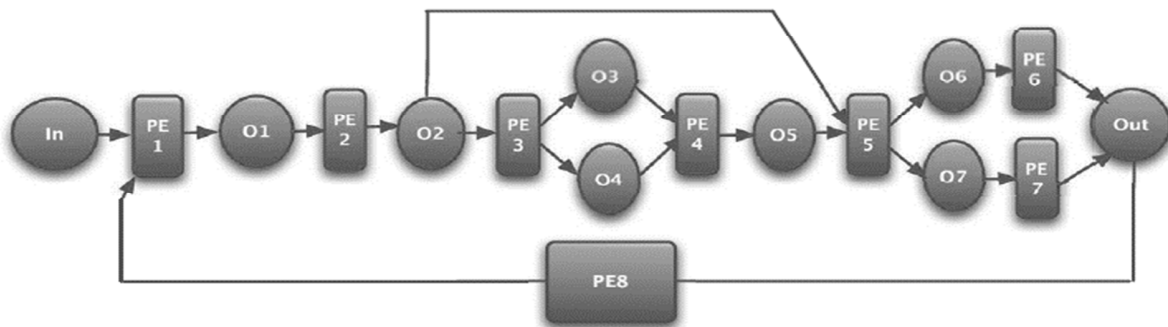


Figure 6. The target Petri Net representation.

4.2 Parameter Settings

The settings of GA evolution process are listed in Table 2. Due to the GEP style constant operation is involved, the parameters were set with the classical values used for a traditional GEP.

Table 2: GA evolution parameter settings.

| Parameters | Values | |
|-----------------------|------------------------|-------|
| Population size | 1000 | |
| Number of Generation | 20000 | |
| Genetic modifications | Cross rate | 30% |
| | Mutation | 0.44% |
| | Constant Mutation rate | 10% |
| Constant | High boundary | -1000 |
| | Low boundary | +1000 |

Since the purpose of evaluation experiment is to verify the concept oriented performance of our GA enhanced ADFMS, we select a PC with an ordinary specification to perform evolution experiments. The specification of benchmark platform is listed in Table 3.

Table 3: The benchmark platform.

| | | |
|------------------|------------------|----------------------|
| CPU | Model | Intel core i7-4700mq |
| | No. of Cores | 4 |
| | No. of Threads | 8 |
| | Frequency | 2.4G |
| Memory | 8 GB | |
| Operation system | Ubuntu 16.04 LTS | |

4.3 Route Detection Analysis

In order to evaluate the performance of the proposed GA bitstream like chromosome for route detection purpose. Ten executions of the GA enhanced ADFMS were conducted with the input data intensive application. With the data size, the sequence of task, the computing time of task and the route detection

fitness function, the GA enhanced ADFMS generate a chromosome containing the route information. Since the size of chromosome is too big to be demonstrated, only part of the best chromosome discovered in the evolution process is listed in Figure 7 showing the first three transition fragments and their corresponding place fragments. Based on the representation mechanism of the GA enhanced ADFMS, we can generate a part of Petri Net from the above chromosome segment as shown in Figure 8. T_x represents Transition node x , P_x is Place node x and P_0 is the start Place node.

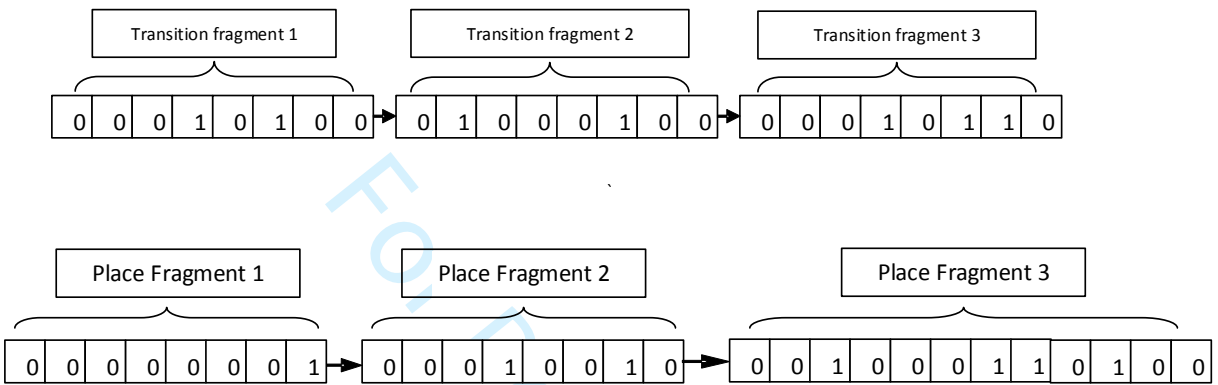


Figure 7. Part of the transition segment and place segment.

As shown in Figure 8, an identical part of the original Petri Net shown in Figure 7 can be extracted from the segment of chromosome generated with the GA enhanced ADFMS. It is worth noting that the constant segment was set to default value (integer value 1). The reason is that the evolution procedure was set for route detection purpose.

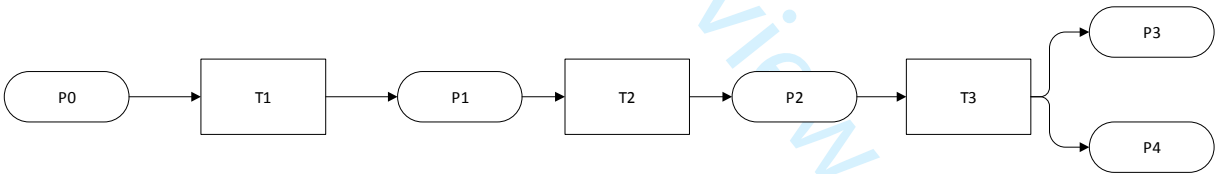


Figure 8. Part of the Petri Net.

With the setting of input dataset, the GA enhanced ADFMS can generate a route of target data intensive application automatically. Following the same setting, the routing work in previous works were done in a static manner. The processing route is predefined before the optimization work can be applied. That limits the potential space of optimization process. The dynamic solution of the GA enhanced ADFMS provides an adjustable route presentation structure. During the evolution process, this adjustable structure can refine the route for different target cloud networks setting scenarios in order to optimize the delivery efficiency of a data intensive application.

4.4 Optimization Results

In order to evaluate the performance of the proposed the GA enhanced ADFMS for optimization purpose, we follow the work [12] to investigate the intermediate data reuse optimization problem in a cloud network. Three scenarios, keeping all intermediate dataset, regenerating all intermediate dataset and dynamically regenerating intermediate dataset, were considered. Ten executions of ADFMS were conducted for the three scenarios. The minimalization of the cost caused by intermediate dataset was set as fitness function for the scenario dynamic regenerate intermediate dataset. The best results are listed in Table 4.

Table 4. Optimization results of ADFMS.

| Time (executions) | Keep all cost (\$) | Regenerate all cost (\$) | Optimization applied cost (\$) |
|-------------------|--------------------|--------------------------|--------------------------------|
| 1 | 0.875838889 | 0.875839 | 0.875838889 |
| 2 | 1.163636111 | 1.631839 | 0.892508897 |
| 3 | 1.619391667 | 2.387839 | 1.101186333 |
| 4 | 2.243105556 | 3.143839 | 1.480449667 |
| 5 | 3.034777778 | 3.899839 | 1.972605556 |
| 6 | 3.994408333 | 4.655839 | 2.476533167 |
| 7 | 5.121997222 | 5.411839 | 3.073198333 |
| 8 | 6.417544444 | 6.167839 | 3.669864139 |
| 9 | 7.88105 | 6.923839 | 3.704253806 |
| 10 | 9.512513889 | 7.679839 | 3.916717833 |

The optimization performance of the GA enhanced ADFMS was also compared with the Automatic Data Reuse Model with Petri Net (ADRMPN) [11]. A time unit (a period which is longer than a complete single execution of given data intensive application task) was selected as a checkpoint to observe the optimization performance. On each check point the speed up ratio of two algorithms are compared. As shown in Figure 9, the GA enhanced ADFMS performs similarly to the ADRMPN.

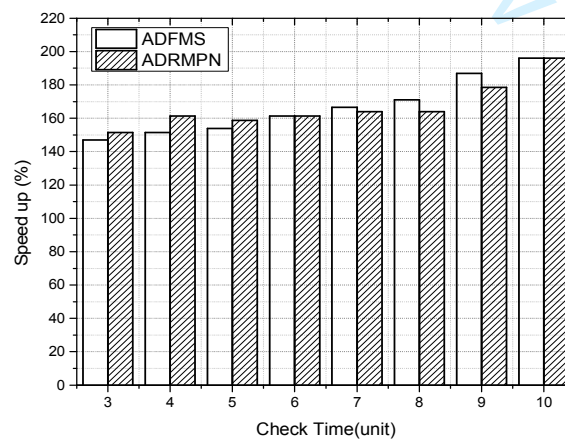


Figure 9. Optimization performance comparison.

It is worth noting that with the elapsed time, the speed up ratio increases, which means the more data requests received on the cloud network, the better optimization performance can be expected. Following the same setting of input dataset, the GA enhanced ADFMS still can achieve similar performance as previous works without a predefined route which are used in previous works. As mentioned in Section 4.3, the adjustable route presentation can maintain more potential space for optimization work. This is due to the interactive connection between the setting of cloud network and the selection of route is created with the GA enhanced ADFMS. Rather than optimizing on a predefined route, a dynamic balance between speed and cost is achievable with the proposed solution in this study.

5. Conclusion

In this paper we propose a novel Genetic Algorithm enhanced Automatic Data Flow Management solution for accelerating data intensive applications in cloud network. Instead of using a static predefined route which is used in previous works, the GA enhanced ADFMS operate a bitstream like chromosome to dynamically maintain the delivery route of a DIA in order to achieve a near maximal level of efficiency. It provides an automatic route detection solution for complex data intensive application. It also can be used to balance the cost generated by the change of target processing route and the cost generated by the deletion, preservation and regeneration of intermediate data in order to optimize the performance of data intensive application on the cloud computing platform.

Since the combination of GA and GEP chromosome structure is newly introduced in this work, the genetic operation efficiency of the proposed GA system is not well tuned. The application of genetic operation was based on classical GA or GEP implementation cases. Its performance can still be further improved to generate a faster and more accurate result at the convergence stage of evolution process.

Acknowledgment

This research is partially supported by the National Basic Research Program (973) of China under grant 2014CB340404 and the Science and Technology Commission of Shanghai Municipality under grant 16JC1401300.

References

- [1] J. Cao, D. P. Spooner, S. A. Jarvis, and G. R. Nudd, "Grid Load Balancing Using Intelligent Agents," *Future Generation Computer Systems*, vol. 21, Issue 1, January 2005.
- [2] C.A. Petri, "Kommunikation mit Automaten," Ph.D. thesis, University of Bonn, 1962.
- [3] C. Jie, D. Zhu, and B. Zhu. "Improved algorithms for intermediate dataset storage in a cloud-based dataflow." *Theoretical Computer Science* 657 (2017): 48-53.
- [4] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: the montage example," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, 2008, pp. 1–12.
- [5] D. Yuan, Y. Yang, X. Liu, G. Zhang, and J. Chen, "A data dependency based strategy for intermediate data storage in scientific cloud workflow systems," *CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE*, vol. 24, pp. 956–976, 2012.

- [6] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: the montage example," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, 2008, pp. 1–12.
- [7] J.L. Peterson, "Petri Nets," *Computing Surveys* 9(3), 1977, pp. 221–252.
- [8] M. Tadao. "Petri nets: Properties, analysis and applications." *Proceedings of the IEEE* 77.4 (1989): 541-580.
- [9] M. Gonzalo. "Timed Petri net modeling and optimization with heuristic search for flexible manufacturing workstations." *Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA'03. IEEE Conference*. Vol. 1. IEEE, 2003.
- [10] P. A. Abdulla, R. Mayr, "Petri Nets with Time and Cost (Tutorial)," *Proceedings 14th International Workshop on Verification of Infinite-State Systems*, Paris, France, 27th August 2012, pp. 9–24.
- [11] L. Han, Z. Xie, and B. Richard, "Automatic data reuse for accelerating data intensive applications in the Cloud." *Internet Technology and Secured Transactions (ICITST), 2013 8th International Conference for*. IEEE, 2013.
- [12] Z. Xie, L. Han, and B. Richard, "Augmented Petri Net Cost Model for Optimisation of Large Bioinformatics Workflows Using Cloud." *Modelling Symposium (EMS), 2013 European*. IEEE, 2013.
- [13] JH. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, U Michigan Press, 1975.
- [14] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, 1996.
- [15] M. Holger. "Evolving Petri nets with a genetic algorithm." *Genetic and Evolutionary Computation Conference*. Springer, Berlin, Heidelberg, 2003.
- [16] J. R. Koza, "Genetic Programming as a Means for Programming Computers by Natural Selection", *Stat. Comput.*, vol. 4, no. 2, pp. 87–112, 1994.
- [17] M. Gonzalo, et al. "Petri nets and genetic algorithms for complex manufacturing systems scheduling." *International Journal of Production Research* 50.3 (2012): 791-803.
- [18] N. Odrey, Y. Ma, "A Multi-Level Multi-Layer Petri Net Based Approach for Manufacturing Systems Control". *Proceedings of the 11th FAIM International Conference*, July 16-18, 2001. Dublin, Ireland.
- [19] D. Yuan, Y. Yang, X. Liu, G. Zhang, and J. Chen, "A data dependency based strategy for intermediate data storage in scientific cloud workflow systems," *CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE*, vol. 24, pp. 956–976, 2012.
- [20] Simon S. Woo, and M. Jelena, "Optimal application allocation on multiple public clouds." *Computer Networks* 68 (2014): 138-148.
- [21] Z. Huang, M. Li, C. Chousidis, A. Mousavi, & C. Jiang, "Schema Theory Based Data Engineering in Gene Expression Programming for Big Data Analytics," *IEEE Transactions on Evolutionary Computation*. DOI:10.1109/TEVC.2017.2771445, 2017.
- [22] C. Ferreira, "Gene Expression Programming: a New Adaptive Algorithm for Solving Problems", *Complex Systems*, vol.13, no.2, pp.22, 2001.
- [23] L. Han, J. I. van Hemert, and R. Baldock, "Automatically identifying and annotating mouse embryo gene expression patterns," *Bioinformatics*, vol. 27, no. 8, pp. 1101–1107, 2011.
- [24] L. Han and H.-Y. Ong, "Accelerating biomedical data-intensive applications using mapreduce," in *2012 ACM/IEEE 13th International Conference on Grid Computing (GRID)*, 2012, pp. 49 – 57.
- [25] Z. Xie, L. Han, and R. Baldock, "Enhancing parallelism of data-intensive bioinformatics applications," in *Proceedings of 8th EUROSIM Congress on Modelling and Simulation*, IEEE, 2013, pp. 519–524.