World Academy of Science, Engineering and Technology
International Journal of Social and Business Sciences
Vol:18, No:7, 2024

About the Case Portfolio Management Algorithms and Their Applications

M. Chumburidze, N. Salia, T. Namchevadze

***Abstract*—**This work deals with case processing problems in business. The task of strategic credit requirements management of cases portfolio is discussed. The information model of credit requirements in a binary tree diagram is considered. The algorithms to solve issues of prioritizing clusters of cases in business have been investigated. An implementation of priority queues to support case management operations has been presented. The corresponding pseudo codes for the programming application have been constructed. The tools applied in this development are based on binary tree ordering algorithms, optimization theory, and business management methods.

***Keywords*—**Credit network, case portfolio, binary tree, priority queue, stack.

I. Introduction

RECENT technological advances show that process management in various fields of science, including business and industrial development, is based on the integration of fundamental and applied research areas, from modelling and creating algorithms to the development and implementation of software platforms. In the modern world, the issues of modelling case management problems and the development of strategic algorithms for their solution are becoming relevant, ensuring effective software implementation by minimizing temporal and spatial difficulties.

Case management is a fairly complex and time-consuming process that comes with quite a lot of complexities, but effective time scheduling algorithms can reduce problems that show when cases will be selected and performed in available [1].

Goals and objectives of the Case Management: improving the forms and methods of working with documents, organizing effective document management and office work in accordance with the requirements of the current legislation and regulations, the formation and preservation of the documentary fund, acquisition, accounting, storage and use of archival documents, development of regulatory documents business organization. A case is a set of activities that are interrelated in some way but the logic of the activity flow is not necessarily an orchestration and combination of steps and rules [2]. So, case management is really more about collaboration than a process-driven workflow and these give a flexibility for people to add and remove steps, together accomplish work that is not ideal for a traditional process automation or workflow. It is not necessary to define a path of claim management. An effective business case is one of the biggest keys to digital transformation success. Business cases are one of those things that's very important. The first step to an effective business to create a case management algorithm of workflows that place a "case" at the center of the workflow, with rules governing what can happen next with that particular "case." This contrasts with a more sequential workflow. *Case* management is the process of storing original documents of various forms, which directly record the main information of the process. The use of computer technology has greatly improved effective digital *case* management. However, the issue of managing personal *cases* is still relevant for managers of the organization [3]. Time is a precious resource. Yet somehow — even when managers are frantically searching for tracking and organizing important *cases*, chances to spending more time wading through piles of paper than it should be overlook how much time is wasted as a result of disorganization. For example, the archiving process in microfinance organizations requires a lot of effort and labor. The *cases* of several branches with large portfolios are sent to the archive for processing at the same time, which causes the case to be delayed and *cases* cannot be sent to partner organizations in time. All this may harm the image of the company and bring financial losses. But effective management can reduce problems and increase efficiency.

This work deals to create an efficient algorithm to manage of personal e-portfolio with facilities own exploring strategies and priorities of cases in the several type of credit organization to explore the credit history of clients. In particular, *case* circulation itself includes archive activity, which is divided into credit and non-credit direction. In particular, the process of archiving credit agreements of clients has been discussed. Strategy management of personal credit agreement is based to build priority queue of credit documents for consideration to perform personal work timely and safely, without errors, respond to clients quickly and effectively [4].

II. Notations and Definitive Concepts

Time management helps a person or company to plan time and save resources. For example, if you are overwhelmed with work, and you do not know what to take on first, you should prioritize which tasks are urgent and important, and which are just distractions [5].

In this section case portfolio management problem is considered. Case selection algorithms to solve of optimization

M. C. is with the Department of Computer Technology, Akaki Tsereteli State University, Georgia (corresponding author, phone: 995-593-90169; e-mail: manana.chumburidze@atsu.edu.ge).

N. S. is Deputy director, Georgian Lawyers High School, Georgia (e-mail: n.salia@bk.ru).

T. N. is with the Department of Computer Technology, Akaki Tsereteli State University, Georgia (e-mail: tsatsa.namchevadze@atsu.edu.ge).

problem of time and case complexity have been considered. Binary-tree model for design and applications to case movement is constructed. The Algorithms for a risk management approach for the design of building portfolios are considered.

In this work we are considering the algorithms of prioritization, planning and structuring of case management [10]:

1. *Prioritization:* To complete a task, you need to determine how urgent, complex and important it is, and only then proceed with its implementation.
2. *Planning:* To complete a task, you need to understand when it should be done and how long it will take.
3. *Structuring:* To complete a task, you need to understand how to track its progress and results.

Let us consider the management of credit processes in a bank or any credit organization. The credit risk management begins with a loan file well documented. It is expected that all credit processes go to the credit committee with requires of credit agreement. A credit case folder is a record that contains the set of data used to conduct a credit review of a customer or customer account.

Statement problem. It is required to optimize case portfolio management to maximize the portfolio income with the criteria to minimize the credit risk.

Solution of this problem is conducted to determine the process of implementing management in minimizing risk of cases. The research method is based on the descriptive algorithms with a qualitative approach. In particular, case requirements selection algorithm to optimize of portfolio incomes is built.

The cases that will be issued a credit requirement are listed in the following content of algorithms [6]:

- To build credit network between departments of banks
- To create data-structure to organize all case requirements in one place
- To consider case requirements of linear ordering algorithms in alternative approaches
- To create priority-queue of cases with respect to key field of credit requirements
- To sort records of credit requirements and store in personal cases portfolio
- To build the clusters of cases and they prioritize.

Our goal is to maximize long-term of case portfolio to construct algorithm with risk awareness to protect the company under the worst-case scenarios.

For consideration, a binary tree data structure to build on network of credit requirements is used. A binary tree is a recursive structure where each vertex has no more than two children [6]. The root of this tree present of credit administration in bank. The nodes are using to describe of collaboration of neighbor departments in credit process. It is a record of the department and the data provided by it, and also contains information about the departments in relational connections. The binary tree determines how positions are departments in hierarchical structure.

In the next stage the order algorithms use to create a track of documents flow to complete a task.

III. The Effective Algorithms and Their Applications

In this section the cases stacking algorithms which are based on priority queues analysis are proposed.

There are binary tree ordering algorithms with Depth-First Search (DFS) and Breadth-First Search (BFS) strategies [7].

- Pre-order Traversal algorithms
- Post-order Traversal algorithms
- In-order Traversal algorithms
- Level Order Traversal algorithms

For the software application, nodes are described in the following pseudocodes:

```
struct node
{ int data;
char department;
node* left;
node* right;
    node(int data,char department)
{this->data=data;
this->> department = department;
this->left=NULL;
this->right=NULL;
} };
```

Allow us to consider simple example of hierarchically organize credit process in bank (see Fig. 1).

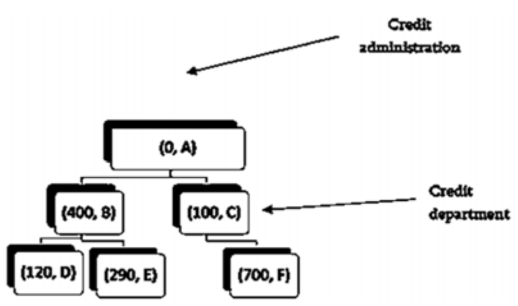

Fig. 1 Topology of credit network

In this work the preorder algorithm [7] to store the credit-requirements documents in one place of cases-portfolio has been constructed. On the basis of these algorithms, it becomes possible to obtain a different sequence of data movement between departments.

The Recursive function applications for Pre-order, post-order and In-order algorithms have the forms correspondingly:

```
  void preorder( node* root)
  {
  if (root==NULL) return;

     cout << root->data << " ";
preorder(  root->left);
  preorder( root->right);

  };

  void postorder( node* root)
```

```
{
if (root==NULL) return;

postorder(  root->left);

postorder( root->right);
cout << root->data << " ";

};

void inorder( node* root)
{
if (root==NULL) return;

inorder (  root->left);
 cout << root->data << " ";
inorder ( root->right);

};
```

For software implementation, the priority queues and stacks data structures to improve the time-complexity of recursive algorithms in the binary trees are used.

In-order traversal is the most commonly used option for traversing a DFS tree. As DFS suggests, we will first focus on the depth of the selected node and then move on to the width at that level. It uses a symmetric addressing rule for vertices. The efficient ordering is important for optimizing the algorithms that require input data to be in sorted lists. The Stack-data structure [8] is used to improve of time and space complexity of an algorithm. The pseudo code to perform this algorithm has a form:

```
void iteration-inorder(node* root)
{stack<node*> s;
// create pair of different data types to combine together two values:
cost of case and mark of case
pair<int, char> adjs[99];
node* curr=root;
int i=0;
while(curr!=NULL  || s.empty()==false)
{
// create a stack of cases route
while(curr!=NULL)
{s.push(curr);
curr=curr->left;}
curr=s.top();
s.pop();
adjs[i].first=curr->data;
adjs[i].second=curr->piliali;
i=i+1;
curr=curr->right;
}
```

Allow us to introduce the vector of credits requirement cases (see Table I). The elements of vector are record type and include symbol of department with a credit requirement:

```
pair<int, char> adjs[] =  {make_pair(120, D), make_pair(400,
B),make_pair(290, E),make_pair(0, A), make_pair(100,
C),make_pair(700, F)}
```

Similarly, we can use pre-order algorithm that will give us different sequences of data than in-order algorithm. Pre-order traversal is also the variant of DFS traversal algorithm. It uses a parent-child addressing rule for vertices. The Stack-data structure is used to improve of time and space complexity of an algorithm. The Pre-order algorithms result is in Table II.

TABLE I
THE VECTOR OF CASES

0	*1*	*2*	*3*	*4*	*5*
(120, D)	(400, B)	(290, E)	(0, A)	(100, C)	(700, F)

TABLE II
THE VECTOR OF RESULTS

0	*1*	*2*	*3*	*4*	*5*
(0, A)	(400, B)	(120, D)	(290, E)	(100, C)	(700, F)

Next variant of DFS traversal algorithm is a sost-order algorithm. It uses a child-parent visiting rule for vertices. The Stack-data structure is used to improve of time and space complexity of an algorithm. The post-order algorithm result is in Table III.

TABLE III
THE VECTOR OF ORDER RESULTS

0	*1*	*2*	*3*	*4*	*5*
(120, D)	(290, E)	(400, B)	(700, F)	(100, C)	(0, A)

Let us consider BFS algorithm for ordering binary tree. The queue-data structure [9] is used to save visited nodes of tree for next exploration for build the track of document flow along of departments. The queue-data structure is used to improve of time and space complexity of an algorithm. The pseudo code to perform this algorithm has a form:

```
void iteration-bfsorder(node* root)
{queue<node*> q;
// create pair of different data types to combine together two values:
cost of case and mark of case
pair<int, char> adjs[99];
node* curr=root;
int i=0;
while(curr!=NULL  || q.empty()==false)
{
// create a stack of cases route
while(curr!=NULL)
{q.push(curr);
curr=curr->left;}
curr=s.top();
q.pop();
adjs[i].first=curr->data;
adjs[i].second=curr->piliali;
i=i+1;
curr=curr->right;
}
```

The BFS algorithms result is in Table IV.

TABLE IV
THE VECTOR OF BFS Results

0	*1*	*2*	*3*	*4*	*5*
(0, A)	(400, B)	(100, C)	(120, D)	(290, E)	(700, F)

Let us consider Table I to build the order of priorities of the departments according to the data. The priority-queuing approach [8] to sort of cases in increase by the *key* of credits requirement is used (see Table V). The following pseudo code is performed:

```
//introduce data structure- priority_queue;
priority_queue<pair<int, char> > p1;
//loop to sort cases in array
for (int k=0; k<i; k++)
    p1.push(adjs[k]);
// introduce data structure- stack
    stack<pair<int, char>> S[99];
    int k=0;
    while(!p1.empty())
        {cout<<p1.top().first<<" "<<p1.top().second<<endl;
          adjs[k]=p1.top();
                p1.pop();
                k=k+1;  }
```

TABLE V
THE ARRAY OF CASES

0	1	2	3	4	5
(0, A)	(100, C)	(120, D)	(290, E)	(400, B)	(700, F)

On the next stage step by step to execute the extract operation of terminal elements from array (see Fig. 2) to create clusters stacks. How to set up the data to create a cluster stack array is as follows:

```
for (int k=0; k<i/2; k++)
    // build stacks of cases
{   S[k].push(adjs[i-k-1]);
    S[k].push(adjs[k]);}
    for (int k=0; k<i/2; k++)
        {cout<<endl<<k<<": ";
    while(!S[k].empty())
            {cout<<S[k].top().first<<"- "<<S[k].top().second<<" ; ";
            S[k].pop();
               }}}
```

As a result cases clusters stacks is obtained (see Fig. 2).

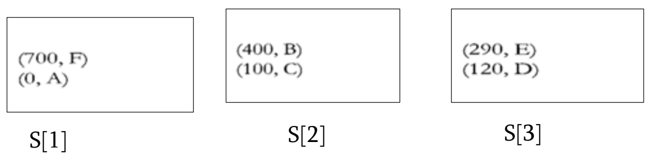

Fig. 2 Stacks of cases

IV. CONCLUSION

In this paper strategic case management problem has been solved. Decision making problem by selecting out different credit requirements cases with a potential outcomes has been decided. The hierarchical model of credit *requirements* in binary tree-diagram is constructed. Case selection algorithm across one time period consideration by the build of cases clusters stacks [10] is developed. The alternative tracks of documents flows are investigated by use the In-order, Pre-order, Post-order and Level Order Traversal (BFS) algorithms.

An implementation of priority queues to support the case management operations is delivered. The features and effectiveness of the cluster approach for the documents to solve case management problem have been investigated. The algorithms for clustering cases in priority queues have been obtained. The cases stacking algorithms on the base of priority queues analysis have been constructed. This approach increases effectiveness of strategic management of cases-portfolio. Optimizing a case portfolio application has been involved minimize default risk.

The corresponding pseudo codes for programing application have been constructed. The tools applied in this development are based on the binary tree data algorithms and dynamic structural analysis in C++ Object-Oriented Programming. For software implementation priority queue and stacks data structures technique are used.

The proposed algorithms and their application in this development reduce the time and space complexity of software implementation of problem.

REFERENCES

[1] Powell, Suzanne K.; Colleagues at Mayo Clinic Hospital, Arizona Work–Life Balance: How Some Case Managers Do It! *Professional Case Management.* 23(5):235-239, September/October 2018

[2] Valenčík, Radim, Červenka, Jan. Analysis Tools of Connecting Investment Opportunities and Investment Means in the Area of Small and Medium-Sized *Enterprises European Research Studies* Vol. XIX, Issue 4, 2016 pp. 130- 139

[3] Gritsyuk, M. Katherina, Gritsyuk I.Vera, Technologies and Tools Used for Prediction of Characteristics of Activity of Industrial Enterprises. *International Journal on Information Technologies and Security*, No.4 (vol. 12), 2020, pp. 47-62

[4] Skiena, Steven. *Sorting and Searching. The Algorithm Design Manual. Springer, 2008, p. 480. doi:10.1007/978-1-84800-070-4_4. ISBN 978-1-84800-069-8.*

[5] Cheryshov, O, N. Choporov, A. P. Preobrazhenskiy, O. Ja. Kravets The Development of Optimization Model and Algorithm for Support of Resources Management in Organizational System, *International Journal on Information Technologies and Security,* No.2 (vol. 12), 2020, pp. 25-36.

[6] Chumburidze, Manana, Mzia Kiknadze, Nino Topuria, and Elza Bitsadze. "About the Algorithms of Strategic Management." In *International Conference on Applied Mathematics, Modeling and Computational Science*, pp. 647-654. Springer, Cham, 2019, DOI: https://doi.org/10.1007/978-3-030-63591-6_59.

[7] Chumburidze, M. et Beradze, T. Cash Flows Management Algorithm and Their Applications. *International Journal on Information Technologies & Security. 2020, Vol. 12 Issue 2, p15-24. 10p*

[8] Bennett, Nicholas. Introduction to Algorithms and Pseudo code, working paper in Project *Exploring Modelling and Computation*, August 2015 (19 p.), DOI: 10.13140/RG.2.2.28657.28008

[9] Chumburidze, M., & Shonia, N. (2022). The Algorithms of Strategic Financial Management. *International Journal on Information Technologies & Security*, *14*(1).

[10] Chumburidze, M., Chahua, G. and Sakhelashvili, T. (2021) Delivery Management Algorithms and Their Applications. *International Journal of Advanced Research* 9(5): pp596-601 DOI: http://dx.doi.org/10.21474/IJAR01/12881