Journal Homepage: -www.journalijar.com

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)

Article DOI:10.21474/IJAR01/12881
DOI URL: http://dx.doi.org/10.21474/IJAR01/12881

ISSN NO. 2320-5407

RESEARCH ARTICLE

DELIVERY MANAGEMENT ALGORITHMS AND THEIR APPLICATIONS

M.Chumburidze[1], G.Chachua[2], T. Sakhelashvili[2] and E. Bitsadze[2]
1. Professor, Department of Computer Technology, Akaki Tsereteli State University.
2. Asisstant Professor, Department of Computer Technology, Akaki Tsereteli State University.

Manuscript Info

Manuscript History
Received: 20 March 2021
Final Accepted: 24 April 2021
Published: May 2021

Key words:-
Product Delivery, Optimization Problem, Dynamical Programming

Abstract

In this article optimization problem of product delivery problems in modern business has been investigated.Desition-making problems to manage network sales strategy is solved.The dynamical model of multistage graph has been constructed. The algorithms to find optimal plan of products salsment is performed. The tools applied in this development based on the generalized dynamical programming methods and graph theory applications.Javascript programming language for software implementation is used.

Introduction:-

One of the important factors in the development of the country is the appearance of special support for businesses and enterprises, which contributes to the production of new products or increase in imports from abroad. Consequently, the number of companies engaged in distribution activities is increasing. Distribution is fundamentally concerned with ensuring that products reach target customers in the most direct and cost efficient manner. The paper aims to help companies optimally redistribute their resources and make maximum profits.

This article is devoted to develop optimization algorithm of sailing workflow of goods to improve delivery operations to the next level with advanced route optimization and powerful analytics in supply chain. Decisions making software of sale management of ordered products with suggestions for best profits routes of deliveries has been performed. Some specific techniques for innovative decisions making in available alternative that can be applied for sales flows of goods in the future has been examined. In general, the decision making process helps managers and other business professionals solve problems by examining alternative choices and deciding on the best route to take. Using a step-by-step approach is an efficient way to make thoughtful, informed decisions that have a positive impact on the organizations short and long terms goals.

Statement Problem

In this section product delivery problems by estimation profits of requirements is considered. Decision making problems on accepting and delivering goods is solved. Graph-creating algorithm to modeling of dynamical process of goods delivering is used. Dynamical programming (**DP)** approaches to describe of multistage plan of delivery has been investigated. Priority queueing implementation by use stacks-techniques is used.

Introduce the following notations:

$(C_{ij})_{nxm}$ - matrix requirements product quantity,$(R_{ij})_{nxm}$- matrix of corresponding benefits (i=1,n;j=1,m), $(X_{ji})_{mxn}$ – Boolean matrix,S- total quantity of products, n- number of stages, m-number of projects.

Corresponding Author:- M. Chumburidze
Address:- #59 Str.Tamar Mepe, Kutaisi, Georgia, 4600.

Problem. It is required to manage product delivery process (PDP) in the following conditions:

$$\sum_{i=1}^{n}\sum_{j=1}^{m}(C_{ij}X_{ji}) \le S$$

$$\textstyle\sum_{i=1}^{n}\sum_{j=1}^{m}(R_{ij}X_{ji}) \to max.$$

Let us construct multistage graph of PDP by use the forward method **of DP.**

The vertices of graph stores the information about quantities of delivering products(Key) with corresponding benefits of current stages. The edges describes networks of delivering. For programing implementationlistAdgs-array to store verteces avalebled from currently node and listParents- arrayto store predict vertices of currently node are used. The pseudocode of initialisation of empty graph is described as follows:

```
function initEmptyGraph()
{           graph = new Array(n + 1)
for (let i = 1; I < graph.length – 1; i++) {
                        graph[i] = new Array(s + 1)
for (let j = 0; j < graph[i].length; j++)      //filing of vertices of graph
{
graph[i][j] = {
i: i,
j : j,
listAdjs: [],
listParents: [],
max_r: 0,
checked: false,
}}}
//filing of source vertex of graph
            Graph[0] = [{
            i: 0,
            j: 0,
            listAdgs: [],
            listParents: [],
            max_r: 0,
            checked: false
            }]
            //filing of end vertex of graph
checked = true
            Graph[n] = [{
            i: n,
            j: 0,
            listAdgs: [],
            listParents: [],
            max_r: 0,
            checked: true
                        }]}
```

Let us construct the the function which find the maximum of the products supplied for each of them :

```
function find_shopMaxes()
{listShops.forEach(shop => //considering the list of shops
{let max_c = 0// Find the maximum of products given for each store:
Object.keys(shop).forEach(key => {
if (key > max_c)
max_c = key}
shop.max_c = max_c
}}
```

the forward approches algorithm of dynamical programing to build information model of delivery in the set connections is constructed. For implemetation makeGraph() finction by usestack-technices is used:

```
function makeGraph(){
initEmptyGraph()// initialization of empty graph
find_shopMaxes() //defin for each stor max quantities of delivering produqts (max_c)
let stack = [ graph[0][0] ] // save the first node in stack
while (stack.length) {
let top = stack.shift()
let shop = listShops[top.i] // considering shop with index top.i
let adjsCount = top.i + 1 === n ? 0 : s
let j = top.i + 1 === n ? 0 : top.j
for (j; j <= adjsCount; j++) {  //comput the quantitie of delivering products
let c = j – top.j
if (adjsCount === 0) {
c = s – top.j
}  //compute of profities
let r = -1
//chack connection beetwen topandgraph[top.i + 1][j]
let hasEdge = true
if (shop[c] != undefined) {
r = shop[c]
}
else {
if (c > shop.max_c) {
r = shop[shop.max_c]
}
else
hasEdge = false
}
let adjNode = graph[top.i + 1][j]
if (hasEdge) {
adjNode.max_r = Math.max(top.max_r + r, adjNode.max_r)
top.listAdjs.push({
i: adjNode.i, // adjNode with index i
j: adjNode.j, // adjNode with index j
C: c,       //products quantitie add  to topfor move in nextf adjNode
R: r, //delivering productsprofits
})
//add the topin the listParents of adjNode
adjNode.listParents.push({
i: top.i,
j: top.j,
R: r
})
if (!adjNode.checked) {
stack.push(adjNode)
adjNode.checked = true
} } } }//construct of stack-data structure to investigation of connection set of PDP.
listRoads = [] // to feeling optimal paths
        find_roads(graph[n][0)// find roads of PDP by use the bacward approche of DP
property var listTempRoad: []// temporery save alternativ roads
function find_roads(node) // optimal path find function listRoads
{       listTempRoad.unshift(node.i === n ? s : node.j)
//condition to  chack finish of each path
if (node.i === 0) {
```

```
listRoads.push(listTempRoad)
listTempRoad.shift()
                    return
}
```

//The pseudocode to find in listParentsof optimal benefit:

```
          let max_r = -1
          let i = 0
          let parent = {}
          for (i; i < node.listParents.length; i++)
          {
parent = node.listParents[i]
var parentNode = graph[parent.i][parent.j] // define parent of node
if (parentNode.max_r + parent.R > max_r) {
          max_r = parentNode.max_r + parent.R   // to find of optimal benefit.
                    } }
```

//The pseudocode to select all path of benefit:

```
          for (i = 0; i < node.listParents.length; i++) {
          parent = node.listParents[i]
          let parentNode = graph[parent.i][parent.j]
                    if (parentNode.max_r + parent.R === max_r) //condition to perform recursion
                    {
                    find_roads(parentNode)
                    }}
          listTempRoad.shift()
}
```

Result of optimal benefits is presented in **graph[n][0]) max_r** and corresponding path of graph is saved in

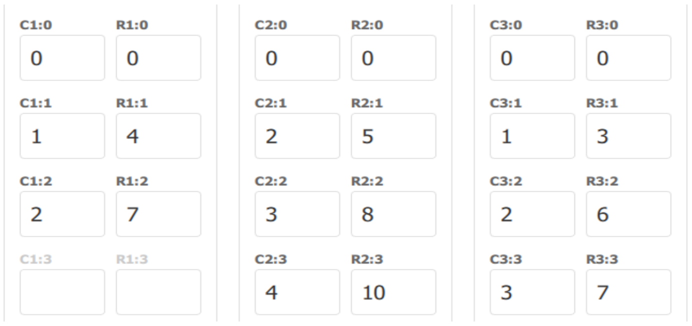
listRoads.

For simplicity consider particular example, when

n = 3, m = 4, s = 5, $C_{ij}(i = 1,...,3; j = 1,...,4)$, $R_{ij}(i = 1,...,3; j = 1,...,4)$

Let us consider list of shops:

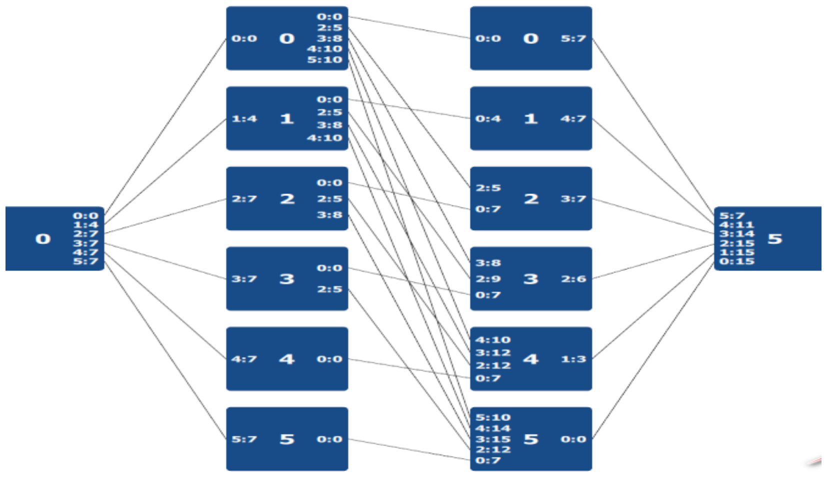
```
property int s = 5
property int n = 3
property int m = 4
property var listShops: [ //information about the shops
          {0: 0, 1: 4, 2: 7},
          {0: 0, 2: 5, 3: 8, 4: 10},
          {0: 0, 1: 3, 2: 6, 3: 7}
] (see pict.1).
```



C1:0	R1:0	C2:0	R2:0	C3:0	R3:0
0	0	0	0	0	0
C1:1	R1:1	C2:1	R2:1	C3:1	R3:1
1	4	2	5	1	3
C1:2	R1:2	C2:2	R2:2	C3:2	R3:2
2	7	3	8	2	6
C1:3	R1:3	C2:3	R2:3	C3:3	R3:3
		4	10	3	7

Picture 1:- Dates table.

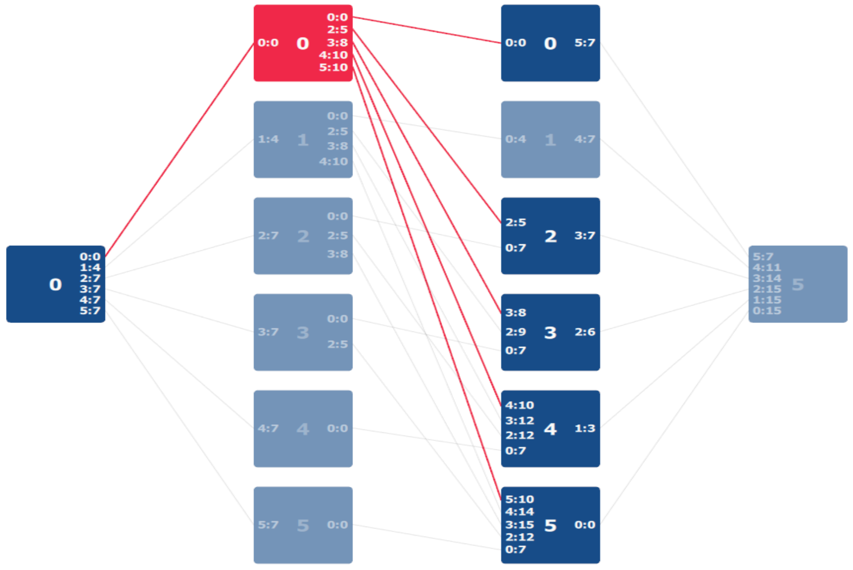
function **makeGraph()** get to result vizualized on the picture (see pict.2).


Picture 2:- PDP Model.

listShops is a list of indexsing stors (nodes of graph) included information about quantities of delivered products (Key) and corresponding benefits. For example consider node with index 1(second stor) ({0: 0, 2: 5, 3: 8, 4: 10})(see. Pic 3)

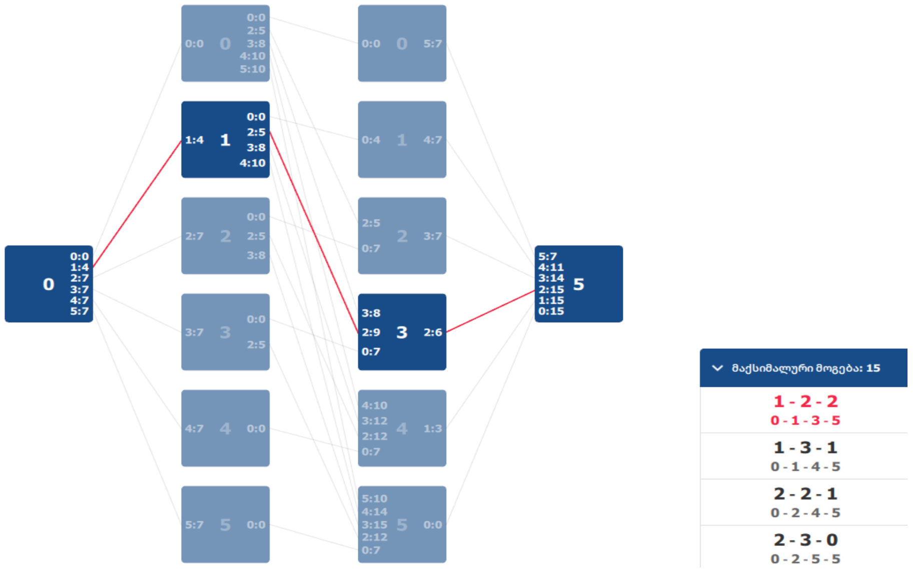
The target node in Pic 3 will be presented as follows:

```
{i: 1,
j: 0,
listAdjs:[
{i: 2, j: 0, C: 0, R: 0},
{i: 2, j: 2, C: 2, R: 5},
{i: 2, j: 3, C: 3, R: 8},
{i: 2, j: 4, C: 4, R: 10},
{i: 2, j: 5, C: 5, R: 10}
],
listParents:[
{i: 0, j: 0, R: 0}
],
max_r: 0,
checked: true,}//check a given node has already been considered during the construction of the graph
```


Picture 3:- PDP for target node.

Accordingly algorithms described above the results get to four alternative plans of optimal delivering with a value 15(Picture 4).

The orders of nodes *0 – 1 – 3 – 5*(Picture 4) indicate on the path (own of alternative plans of optimal delivery) delivering tour and *1-2-2* indicate number of delivered products with a **totalsum** of profits: 15.



Picture 4:- Optimal path of PDP.

Conclusion:-

In this work optimization problem of sale management of ordered products by the generalized DP approaches to maximize profits of products deliveries has been solved. Decisions making software of optimization management hes been performed. The particular example of stores selection problem within corporate network of delivering process has been discussed.
Multistage graph-model to describe of sales with a corresponding profit-probability has been constructed. The algorithms to obtain optimal incomes have been obtained.

This investigation has a many advantages: forecasting and analyzing of sales incomes in any time; fluently make decision in the planning stage to select of candidates; monitoring and managing product delivery process; optimize products flow; efficient in terms of time complexity; the results enable to be applied in decision making problems to optimize solution.

References:-

1. Chumburidze, M. et al. Dynamic Programming and Greedy Algorithm Strategy for Solving Several Classes of Graph Optimization Problems. *Broad Research in Artificial Intelligence and Neuroscience. 10 (vol. 1), Feb 2019, pp.101-107.*
2. Bennett, Nicholas. Introduction to Algorithms and Pseudocode, Working paper in Project "Exploring Modelling and Computation", August 2015 (19 p.), DOI: 10.13140/RG.2.2.28657.28008
3. Chumburidze, M. et al. About the Algorithms of Strategic Management. *The AMMCS 2019 Conference Book of Abstracts. AMMCS 2019 Publishers, Waterloo, Canada, August 18–23, 2019, pp. 100-101.*
4. Chumburidze, M. et al. The Complexity of Algorithms for Optimization Problems. *Computer Science & Telecommunications, 2 (vol.54). 2018, pp.125-130.*
5. Chumburidze, M. et al. Cash Flows Management Algorithm And Their Applications. *International Journal on Information Technologies &Security. 2020, Vol. 12 Issue 2, p15-24. 10p.*