ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტი
ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტი
კომპიუტერულ მეცნიერებათა დეპარტამენტი

ნანა სანიკიძე

მარაგთა მართვის ერთი ამოცანის სიმულაციური მოდელირება
Simulation modeling of one task on inventory management

სამაგისტრო პროგრამა: ინფორმაციული სისტემები

სამაგისტრო ნაშრომი შესრულებულია ინფორმაციულ სისტემებში
მეცნიერების მაგისტრის აკადემიური ხარისხის მოსაპოვებლად

ხელმძღვანელი: **ფრიდონ დვალიშვილი**
ფიზ. -მათ. მეცნიერებათა კანდიდატი,
ასოცირებული პროფესორი

თბილისი
2017

ანოტაცია

ნაშრომში მოყვანილია მარაგთა მართვის დისკრეტულ-ალბათური მოდელი. ჩამოყალიბებულია მარაგთა მართვის ერთი კონკრეტული ამოცანა და მისი გადაჭრის სტრატეგია სიმულაციებით. მოყვანილია მოდელის ძირითადი პარამეტრები, მათი განსაზღვრის წესი, ის ალბათური განაწილებები, რომელთა გენერირებითაც ხდება პარამეტრების მნიშვნელობების მიღება. მოყვანილია პროგრამის ორგანიზაცია და ლოგიკა, ძირითადი ხდომილებების ბლოკ-სქემები.

Annotation

The paper describes discrete probability model of inventory management. It covers one concrete task of inventory management and simulations of its solution strategy. Here are also described basic parameters of the model, rules of their determination and the probability distributions, which generate the values of parameters. The program organization and logic, as well as flowcharts (diagrams) of main events are presented in this document.

სარჩევი

შესავალი

სისტემა წარმოადგენს ობიექტების, მაგალითად ადამიანების ან მექანიზმების ერთობლიობას, რომლებიც ფუნქციონირებენ და ერთმანეთთან ურთიერთქმედებენ გარკვეული მიზნის მისაღწევად. სისტემის შესწავლისა თუ ფუნქციონირების დასადგენად ჩვენ ვაკეთებთ გარკვეულ დაშვებებს. ეს დაშვებები ჩაიწერება მათემატიკურ-ლოგიკურ მიმართებებში, რომლებიც ყალიბდება გარკვეულ სტრუქტურაში - მოდელში. ზოგიერთ შემთხვევაში მიზნის მისაღწევად საკმარისია მათემატიკურ-ანალიზური მეთოდებით სარგებლობა, თუმცა უმრავლესობა სისტემებისა რთულია, და მათთვის შეუძლებელია ისეთი რელური მოდელის შექმნა, რომლის რეალიზაცია, ამოხსნა, ჩაიწერება ანალიზურად. ასეთი სისტემებისთვის ფართოდ გამოიყენება სიმულაციური მოდელირება.

მოდელირება და სიმულაცია გამოყენებული შეძლება იყოს ადამიანის საქმიანობის სხვადასხვა სფეროში. იგი განსაკუთრებით ეფექტურია შემდეგი პრობლემების გადაწყვეტაში:

- წარმოების და ბიზნეს სისტემების პროექტირება და ანალიზი;
- სატრანსპორტო სისტემების პროექტირება და მუშაობის ანალიზი. მაგალითად: აეროპორტები, ავტომაგისტრალები, ქალაქის საკომუნიკაციო-საავტომობილო სისტემები, მეტროპოლიტენი და სხვა.
- სხვადასხვა მასობრივი მომსახურების ორგანიზაციების შექმნის პროექტის შეფასება. მაგალითად: შეკვეთების დამუშავების ცენტრი, სწრაფი კვების სისტემა, საავადმყოფოების ჰოსპიტალების ქსელი, სხვადასხვა მომსახურეობის, თუ კავშირის ქსელები, პროვაიდერები და სხვა.
- სამხედრო შეიარაღების სხვადასხვა სისტემების შეფასება და მოთხოვნების დაგეგმვა მათი მატერიალურ-ტექნიკური უზრუნველყოფისათვის;
- კავშირის ქსელების პროტოკოლებისა და მოწყობილობებისთვის მოთხოვნის განსაზღვრა;
- სხვადასხვა კომპიუტერული სისტემების პროგრამული უზრუნველყოფისა და მოწყობილობებზე მოთხოვნის განსაზღვრა;
- საქმიან სფეროში სხვადასხვა პროცესების მოდელირება;
- რესურსების მართის სისტემებში პოლიტიკის განსაზღვრა;

- ფინანსური და ეკონომიკური სისტემების წინასწარი ანალიზი;
- ეკო-სისტემები და გლობალური დათბობის პროცესების სიმულაციური მოდელირება და სხვა;

რაც დრო გადის სისტემები უფრო და უფრო რთულდება და ანალიზური მეთოდები პრაქტიკულად გამოუყენებელი ხდება. ბოლო პერიოდში სიმულაციური მოდელირების გამოყენება და გავრცელება უკავშირდება რამდენიმე ფაქტორ-პრობლემის ფაქტიურ დაძლევას:

1) მოდელირებისთვის, რომლებიც ბოლო პერიოდში რთულ სისტემებსა და პროცესებს იკვლევენ, გართულებულია კომპიუტერული პროგრამების დაწერა, თუმცა უკანასკნელ წლებში ეს პრობლემა იხსნება ძლიერი პროდუქტების გამოჩენასთან ერთად;
2) რთული სისტემების მოდელირება ხშირად ითხოვს დიდ მანქანურ დროს. თუმცა უკანასკნელი ძლიერი კომპიუტერის სწრაფქმედების ზრდითა და თვითღირებულების შემცირებით ეს პრობლემაც თანდათანობით იხსნება;
3) კომპიუტერულ მოდელირებაზე ხშირად იქმნდება არასწორი შთაბეჭდილება, როგორც დაპროგრამების სავარჯიშოზე, მიუხედავად იმისა, როგორი რთული არ უნდა იყოს იგი. ამ შეხედულებების საფუძველზე კვლევები სისტემებზე მოდელირებით სრულდება, როგორც ევრისტიკული მოდელის დაპროგრამება, მისი შემდგომი ერთჯერადი „გაშვებით“ პროგრამაზე „პასუხების“ მისაღებად. ასეთი მიდგომა აზრს კარგავს სისტემების სირთულის ზრდასთან ერთად, რასაც თვითონ დაპროგრამების მსოფლიოს წამყვანი ჯგუფებიც ადასტურებენ.

სიმულაციური მოდელირება აღიქმება, როგორც „უკანასკნელი იმედს მეთოდი“ და ამაში არის ჭეშმარიტება. თუმცა რელობაში ძალიან მალე მივდივართ იმ აზრამდე, რომ სიმულაცია და მოდელირება ხშირად ერთადერთი ინსტრუმენტია რთული სისტემის შესწავლისა და კვლევისათვის.

სიმულაციური მოდელების კლასიფიკაცია ხდება ქვემოთ ჩამოთვლილი სამი ასპექტის მიხედვით:

1) სტატიკური თუ დინამიკური? სტატიკური სიმულაციური მოდელი - ეს არის სისტემა დროის განსაზღვრულ მომენტში. სისტემა, რომელშიც დრო არ თამაშობს არავითარ როლს. დინამიკური სიმულაციური მოდელი კი წარმოადგენს სისტემას, რომელიც იცვლება დროში, როდესაც სისტემის მდგომარეობა ევოლუციას განიცდის.
2) დეტერმინისტული თუ სტოქასტული? თუ სიმულაციური მოდელი არ შეიცავს ალბათურ-სტატისტიკურ კომპონენტებს, მას დეტერმნისტული ეწოდება. ასეთ მოდელში შედეგი შეიძლება მივიღოთ როდესაც მისთვის მოცემულია ყველა შემავალი ცვლადები და დამოკიდებულებანი. ძალიან ბევრი სისტემა მოდელირდება რამოდენიმე შემთხვევითი შემავალი კომპონენტების მონაცემებით. შედეგად იქმნება სტოქასტული სიმულაციური მოდელი.
3) უწყვეტია თუ დისკრეტული? უწყვეტი მოდელირება - სისტემის დროში მოდელირება, რომლის წარმოდგენაში მდგომარეობის ცვლადები იცვლებიან უწყვეტად დროსთან მიმართებაში. დისკრეტული სიმულაციური მოდელების დინამიკური ბუნება ითხოვს, რომ თვალყური ვადევნოთ სიმულაციური დროის მიმდინარე დისკრეტულ მნიშვნელობებს მოდელის ფუნქციონირების გასწვრივ.

სიმულაციურ მოდელში მნიშვნელოვანია სიმულაციური დროის მიმდინარეობის მექანიზმის არსებობა. მიუხედავად იმისა, რომ მოდელირება გამოიყენება სხვადასხვა რეალური სისტემების ანალიზისათვის, ყველა დისკრეტულ-ხდომილებით სიმულაციურ მოდელებს გააჩნიათ საერთო კომპონენტები. დისკრეტულ-ხდომილებითი სიმულაციური მოდელი, რომელიც იყენებს ხდომილებიდან ხდომილებაზე გადასვლის მოდელური დროის საათის(მდს) მექანიზმს და დაწერილია რომელიმე უნივერსალურ დაპროგრამების ენაზე, შეიცავს შემდეგ კომპონენტებს:

- სისტემის მდგომარეობა - მდგომარეობათა ცვლადების ერთობლიობა, აუცილებელი სისტემის აღწერისათვის დროის განსაზღვრულ მომენტში;
- მოდელური დროის საათი - ცვლადი, რომელიც შეიცავს ყოველი შემდეგი განსხვავებული ტიპის ხდომილების დადგომის დროს;
- სტატისტიკური მთვლელები - ცვლადები, რომლებშიც ინახება სისტემის მახასიათებლებზე სტატისტიკური ინფორმაცია;

- ინიციალიზაციის პროგრამა - ქვეპროგრამა, რომელსაც სიმულაციური მოდელი მოყავს საწყის, ნულოვან მდგომარეობაში;
- სინქრონიზაციის პროგრამა - ქვეპროგრამა, რომელიც ხდმილებათა სიაში ეძებს შემდეგ ხდომილებას და მოდელური დროის საათის ჩვენება გადაყავს ახალი ხდომილების დადგომის დროზე;
- ხდომილებათა დამუშავების პროგრამა - ქვეპროგრამა, რომელიც ანახლებს სისტემის მდგომარეობას, როდესაც დგება განსაზღვრული ტიპის ხდომილება. ყოველი ტიპის ხდომილებისთვის არსებობს ხდომილების დამუშავების ცალკე ქვეპროგრამა;
- ბიბლიოთეკური პროგრამა - ქვეპროგრამათა კრებული, რომელიც გამოიყენება შემთხვევით დაკვირვებათა გენერაციისათვის(რომლებიც გამოიყენებიან სიმულაციურ მოდელში);
- ანგარიშების გენერატორი - ქვეპროგრამა, რომელიც ითვლის სტატისტიკური მთვლელებიდან სისტემის მუშაობის კრიტერიუმების შეფასებებს და იძლევა მოდელირების დასრულების ანგარიშს;
- ძირითადი პროგრამა - ქვეპროგრამა, რომელიც იძახებს სინქრონიზაციის პროგრამას, რათა განისაზღვროს შემდეგი ხდომილება. შემდეგ მართვას გადასცემს შესაბამისი ხდომილების დამუშავებს პროგრამას, რათა მოხდეს სისტემის მდგომარეობის განახლება. ძირითად პროგრამას ასევე შეუძლია შეასრულოს მოდელირების შეწყვეტის კონტროლი და გამოიძახოს მოდელირების დასრულების ანგარიშის გენერატორიც.

1. მარაგთა მართვის სისტემის სიმულაციური მოდელირება

განვიხილოთ მარაგთა მართვის ერთი ამოცანა და მისი სიმულაციური მოდელირება.

1.1. ამოცანის დასმა

კომპანია ყიდის გარკვეული ერთი სახის პროდუქციას. აუცილებელია განისაზღვროს მომდევნო n თვის განმავლობაში თუ რა რაოდენობით უნდა ჰქონდეს მას გასაყიდი პროდუქციის მარაგი ყოველი მომდევნო თვისთვის. მოთხოვნის დადგომებს შორის დროის შუალედები არის დამოუკიდებელი, ერთნაირად განაწილებული შემთხვევითი სიდიდეები, საშუალო მნიშვნელობით t_0. მოთხოვნის მოცულობა D ასევე წარმოადგენს დამოუკიდებელ(ისინი არ არიან დამოკიდებული იმაზე თუ როდის დადგება მოთხოვნა), ერთნაირად განაწილებულ შემთხვევით სიდიდეებს. ჩვენ განვიხილავთ შემდეგი სახის კონკრეტულ შემთხვევას D -სთვის:

$$D = \begin{cases} 1, \text{ალბათობით } \frac{1}{6} \\ 2, \text{ალბათობით } \frac{1}{3} \\ 3, \text{ალბათობით } \frac{1}{3} \\ 4, \text{ალბათობით } \frac{1}{6} \end{cases}$$

ყოველი თვის დასაწყისში , მარაგის დონის შესაბამისად, კომპანიამ უნდა გადაწყვიტოს თუ რა რაოდენობის პროდუქცია უნდა შეუკვეთოს მომწოდებელს.

ვთქვათ, Z არის შეკვეთილი პროდუქციის რაოდენობა, მაშინ შეკვეთის ხარჯები იქნება:

$$K + iZ$$

სადაც K არის შეკვეთილი პროდუქციის შესასყიდი ფასი, i - კი დამატებითი ხარჯი ერთეულ პროდუქციაზე (მაგ: ტრანსპორტირების ხარჯი და ა.შ.).

ცხადია, რომ თუ Z = 0, მაშინ შეკვეთის ხარჯი ნულის ტოლია. შეკვეთის გაფორმების შემდეგ, შეკვეთის მიღების დრო წარმოადგენს [a, b] შუალედზე თანაბრად განაწილებულ შემთხვევით სიდიდეს.

კომპანიის მარაგების მართვის სტრატეგია (s, S), რომელსაც ის მუდმივად იყენებს, შემდეგია:

$$Z = \begin{cases} S - I & \text{, თუ } I < s \\ 0 & \text{, თუ } I \geq s \end{cases}$$

სადაც I, S, s - არიან მარაგების დონე, შესაბამისად თვის დასაწყისში, შეკვეთის მიღების მომენტში და კრიტიკული.

თუ მარაგის დონე არანაკლებია მოთხოვნაზე, მოთხოვნა დაუყონებლივ კმაყოფილდება. თუ მოთხოვნა მეტია მარაგზე, განსხვავება გადაიდება და იფარება შემდგომი შევსების შემდეგ. შეკვეთის მიღების დროს, თავდაპირველად იფარება გადადებული მოთხოვნები. დარჩენილი ნაშთით (თუ ასეთი იქნება) ივსება მარაგი.

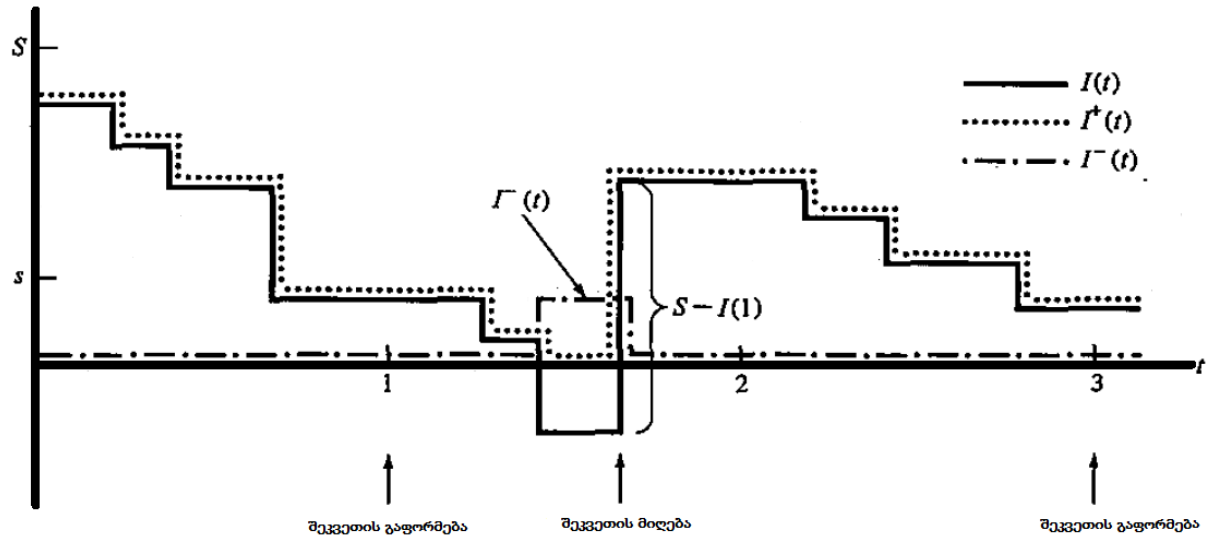
აღვნიშნოთ, რომ რეალურ სისტემებში გვაქვს არა მარტო შეკვეთის ხარჯები, არამედ შენახვის ხარჯები და დეფიციტით მიღებული დანაკარგი. შემოვიღოთ აღნიშვნები:

ვთქვათ, I(t) არის მარაგის დონე t მომენტში (I(t) შეიძლება იყოს დადებითი, უარყოფითი ან 0),

$I^+(t) = \max\{I(t), 0\}$ t მომენტში მარაგის დონე ($I^+(t) >= 0$).

$I^-(t) = \max\{-I(t), 0\}$ პროდუქციის ის რაოდენობა , რომელიც დარეზერვდა მომავლისთვის t მომენტში.

I(t), $I^+(t)$ და $I^-(t)$ შესაძლო მნიშვნელობები ნაჩვენებია ნახაზ 1-ზე.



ნახ. 1. პროდუქციის რაოდეობების I(t), $I^+(t)$ და $I^-(t)$ ცვლილება დროში.

ჩვენ ვიგულისხმებთ, რომ შენავის ხარჯი (ერთ თვეში) არის h. შენახვის ხარჯებში შეიძლება ვიგულისხმოთ: საწყობის, დაზღვევის, მომსახურების, სხვადასხვა ტიპის გადასახადები, ფარული ხარჯები (მაგ: მარაგებში დახარჯული კაპიტალის ინვესტირება სხვა აქტივებში).

თუ $I^+(t)$ წარმოადგენს t მომენტში მარაგის რაოდენობას, მაშინ n თვის განმავლობაში საშუალო მარაგი შეგვიძლია ჩავწეროთ შემდეგნაირად:

$$\bar{I}^+ = \frac{\int_0^n I^+(t)\,dt}{n}$$

მაშასადამე, საშუალო შენახვის ხარჯი n თვეში იქნება $h\bar{I}^+$.

ვთქვათ დეფიციტის ხარჯია π ერთეულ პროდუქციაზე ერთ თვეში. დეფიციტის ხარჯებში შეგვიაძლია ვიგულისხმოთ ის შემოსავალი, რომელიც შეიძლებოდა მიეღო კომპანიას მარაგის არსებობის შემთხვევაში, ასევე დეფიციტის გამო კომპანიის იმიჯზე მიყენებული ზარალი. საშუალო დეფიციტური ხარჯი განვსაზღვროთ შემდეგნაირად:

$$\bar{I}^{-} = \frac{\int_0^n I^{-}(t)\,dt}{n}$$

მაშასადამე, საშუალო დეფიციტური ხარჯი ერთ თვეში იქნება - $\pi\bar{I}^{-}$.

განვიხილოთ კონკრეტული შემთხვევა. ვთქვათ, მარაგის საწყისი დონეა I(0) = 60. მოდელირებას მოვახდენთ n = 120 თვე, ხოლო მოყვანილი ამოცანის პარამეტრებია:

t_0 = 0,1 თვე; K = \$32; i = \$3; [a, b] = [0.5, 1]; h = \$1; π = \$5.

განვიხილოთ და ერთმანეთს შევადაროთ შემდეგი 9 სტრატეგია:

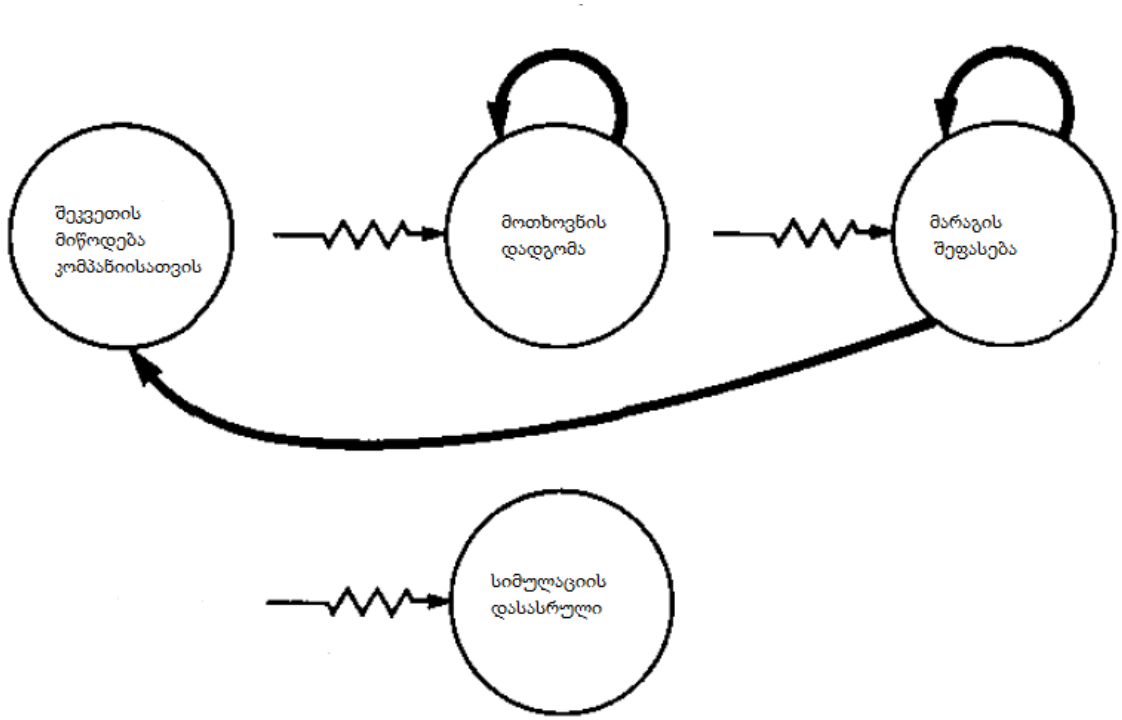
s	20	20	20	20	40	40	40	60	60
S	40	60	80	100	60	80	100	80	100

1.2. პროგრამის ორგანიზება და ლოგიკა

მარაგთა მართვის სისტემის ეს მოდელი იყენებს 4 განსხვავებული ტიპის ხდომილობებს:

ხდომილება	ხდომილების ტიპი
მიმწოდებლიდან შეკვეთის მიწოდება კომპანიისთვის	1
მყიდველისაგან პროდუქციაზე მოთხოვნა	2
სიმულაციის დასასრული n თვის შემდეგ	3
თვის დასაწყისში მარაგის შეფასება	4

ხდომილებების გრაფი მოცემულია ნახაზ 2-ზე.



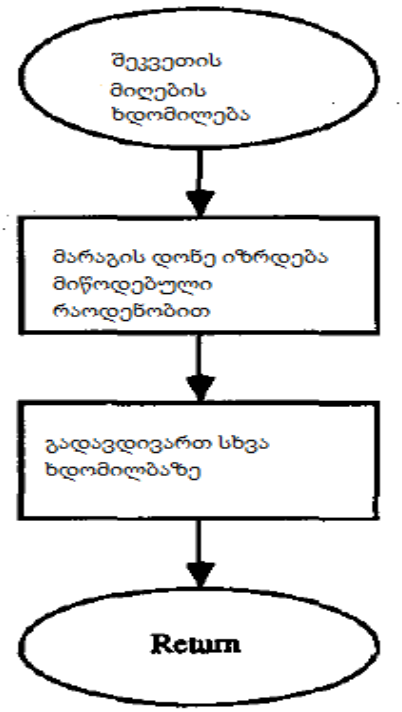
ნახ. 2. ხდომილებათა გრაფი.

ამ სისტემის მოდელირებისათვის საჭიროა სამი ტიპის შემთხვევითი სიდიდე:

1. მოთხოვნის დადგომის მომენტებს შორის დროის შუალედი, რომლებისთვისაც შეიძლება გამოვიყენოთ ექსპონენციალური განაწილება;
2. მოთხოვნის მოცულობის დადგენისათვის შეიძლება გამოვიყენოთ დისკეტული შემთხვევითი სიდიდე, რომელიც უკვე აღწერილია ზემოთ. მისი მოდელირება შეიძლება მოხდეს შესაბამისი წესით .
3. შეკვეთის მიღების დრო არის თანაბრად განაწილებულ შემთხვევითი სიდიდე [a, b] შუალედზე. მისი მოდელირება შეიძლება მოხდეს შესაბამისი გენერირების წესებით.

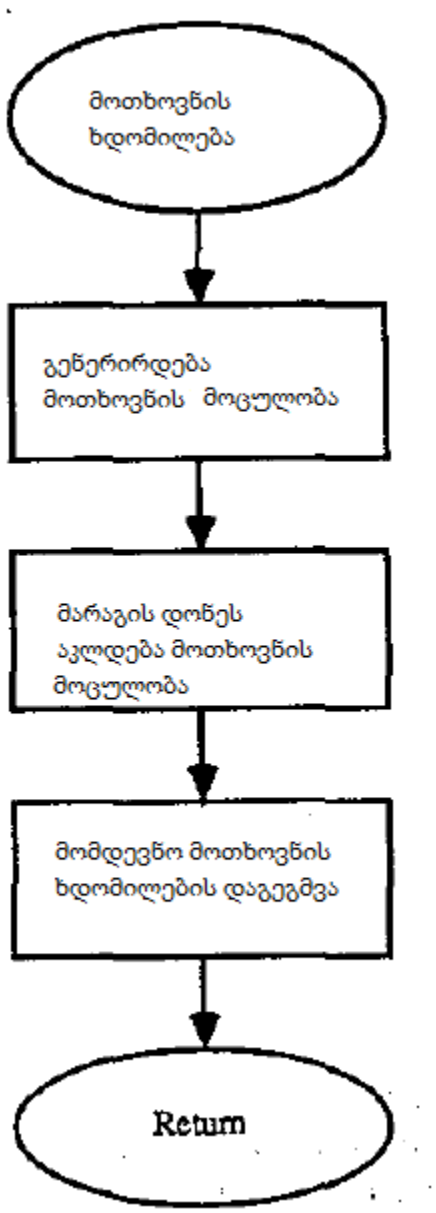
განვიხილოთ მარაგთა მართვის ამოცანის ხდომილებათა გრაფის თითოეული ხდომილება.

შეკვეთის მიღების ბლოკ-სქემა ნაჩვენებია ნახაზ 3-ზე . შეკვეთის მიწოდებისთანავე უნდა აისახოს შესაბამისი ცვლილებები: მარაგის დონე იზრდება მოწოდებული რაოდენობით და მარაგის შევსებისთანავე გადავდივართ სხვა ხდომილებაზე.



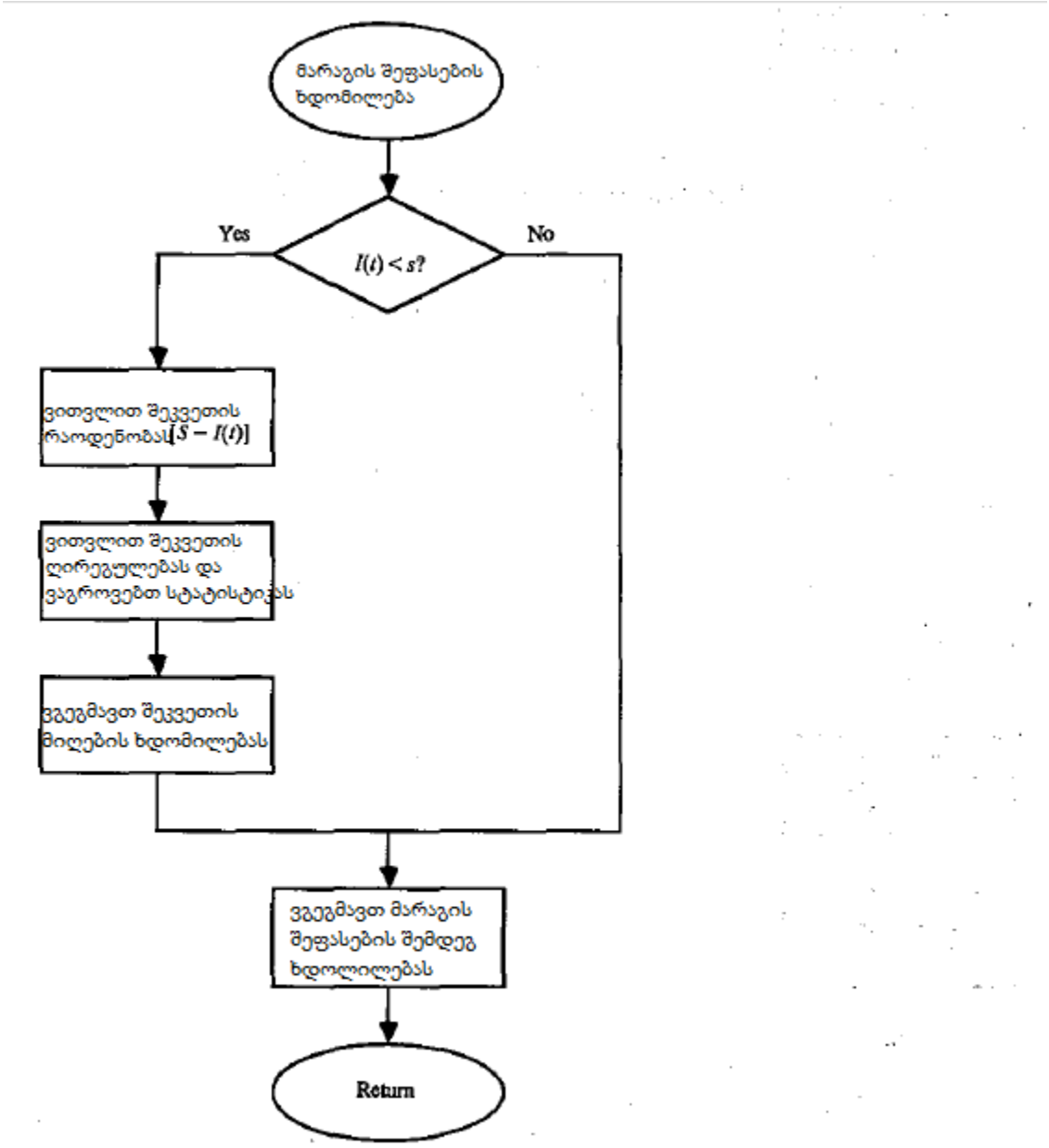
ნახ. 3. შეკვეთის მიღების ბლოკ-სქემა.

პროდუქციის მოთხოვნის ხდომილება მოცემულია ნახაზ 4-ზე. პროდუქტის მოთხოვნის დროს პირველ რიგში ვაგენერირებთ მოთხოვნის მოცულობას, მარაგს ვამცირებთ ამ რაოდენობით, საბოლოოდ კი ვგეგმავთ შემდეგი მოთხოვნის დროს. ამ დროს მარაგის შეიძლება გახდეს უარყოფითიც.



ნახ. 4. მოთხოვნის დადგომის ხდომილების ბლოკ-სქემა.

მარაგის შეფასება აღწერილია ნახაზ 5-ზე. თუ ამ მომენტში მარაგის დონე (I(t)) მეტია საკმარისი მარაგის მინიმალურ მნიშვნელობაზე(s-ზე), ეს იმას ნიშნავს რომ არსებული მარაგი საკმარისი, საჭირო არ არის პროდუქტის შეკვეთა და გადავდივართ შემდეგ ხდომილებაზე. თუ I(t) < s (კრიტიკული რაოდენობა), საჭიროა პროდუქციის შეკვეთა S - I(t) რაოდენობაზე.



ნახ. 5. მარაგის შეფასების ხდომილების პროგრამის ბლოკ-სქემა.

2. C# პროგრამა

2.1. პროგრამული კოდის აღწერა

1 ცხრილში მოცემულია ქვეპროგრამები, რომლებსაც იყენებს ჩვენს მიერ აღწერილი სიმულაციური მოდელი.

ქვეპროგრამა	ქვეპროგრამის აღწერა
Initialize	ინიციალიზაციის პროგრამა
Timing	სინქრონიზაციის პროგრამა
OrderArrival	1 ხდომილების დამუშავების პროგრამა
Demand	2 ხდომილების დამუშავების პროგრამა
Report	3 ხდომილების დამუშავების პროგრამა
Evaluate	4 ხდომილების დამუშავების პროგრამა
UpdateTimeAvgStates	სტატისტიკური მთვლელების განახლების პროგრამა
Expon	ექსპონენციალური შემთხვევითი რიცხვის გენერირების პროგრამა
Uniform	[a, b] ინტერვალიდან შემთხვევითი რიცხვის გენერირების პროგრამა
Rand	[0, 1] ინტერვალიდან შემთხვევითი რიცხვის გენერირების პროგრამა

ცხრილი 1. სიმულაციურ მოდელში გამოყენებული ქვეპროგრამები.

მომხმარებლისთვის განკუთვნილი ინტერფეისიდან ხდება ჩვენი სიმულაციური მოდელის პარამეტრების განსაზღვრა. ეს პარამეტრებია:

- საწყისი მარაგის დონე;
- სიმულაციის სიგრძე;
- პროდუქტის შეკვეთის რეგულარული ღირებულება;
- ერთეული პროდუქტის ღირებულება;
- შენახვის გადასახადი ერთეულ პროდუქტზე;
- დეფიციტის გადასახადი ერთეულ პროდუქტზე.

ძირითადი პროგრამა ყოველი (s, S) წყვილისთვის იძახებს ინიციალიზაციის პროგრამას. ამის შემდეგ ხდება სინქრონიზაციის პროგრამის გამოძახება, რათა განისაზღვროს შემდეგი ხდომილება. შემდეგ მართვას გადასცემს შესაბამის ხდომილების დამუშავების ფუნქციას, რათა მოხდეს სისტემის მდგომარეობის განახლება. ძირითადი პროგრამა ასევე ასრულებს მოდელირების შეწყვეტის კონტროლს. მოდელირების შეწყვეტა ხდება მესამე ტიპის ხლომილებაზე(სიმულაციის დასასრული) გადასვლის დროს. მესამე ტიპის ხდომილება ითვლის სტატისტიკური მთვლელებიდან სისტემის მუშაობის კრიტერიუმების შეფასებეს და იძლევა მოდელირების დასრულების ანგარიშს.

მოვიყვანოთ ძირითადი პროგრამა :

```
static void Main(string[] args)
{
            //სიმულაციურ მოდელში ხდომილებების რაოდენობა
            int numEvents = 4;

            // საწყისი მარაგის დონე
            int inventoryLevel = 60;

            //მოთხოვნის დადგომებს შორის დროის შუალედების საშუალო მნიშვნელობა
            double interdemanTime = 0.1;

            //სიმულაციის სიგრძე(რამდენი თვე მოვახდენთ მოდელირებას)
            int simulationLength = 120;
```

```
//შეკვეთის მიღების დრო
double deliveryRangeMin = 0.5;
double deliveryRangeMax = 1;

//დამატებითი ხარჯი ერთეულ პროდუქციაზე
double incrementalCost = 3.0;

//შეკვეთილი პროდუქციის შესასყიდი ფასი
double setupCost = 32.0;

//დეფიციტით მიღებული დანაკარგი ერთეულ პროდუქტზე
double holdingCost = 1.0;

//შენახვის ხარჯი ერთეულ პროდუქტზე
double shortageCost = 5.0;

//მოთხვნის სიგრძე
double[] probDistrib = { 0, 0.167, 0.5, 0.833, 1 };

//სტრატეგიების  რაოდენობა
int numberOfPolicies = 9;

// s და S მნიშვნელობები თითოეული სტრატეგიისთვის
List<int> s = new List<int>();
s.Add(20);
s.Add(20);
s.Add(20);
s.Add(20);
s.Add(40);
s.Add(40);
s.Add(40);
s.Add(60);
s.Add(60);

List<int> S = new List<int>();
S.Add(40);
S.Add(60);
```

```
S.Add(80);
S.Add(100);
S.Add(60);
S.Add(80);
S.Add(100);
S.Add(80);
S.Add(100);

//სიმულაციური მოდელირების დასაწყისი თითოეული სტრატეგიისთვის
for (int i = 0; i < numberOfPolicies; i++)
{
    //ინიციალიზაციის პროგრამა
    Initialize
    int nextEventType = 0;
    do
    {
        //სინქრონიზაციის პროგრამა
        Program m = new Program();
        m.Timing(timeNextEvent, numEvents, out time, out nextEventType);

        //სტატისტიკური მთვლელების განახლების პროგრამა
        UpdateTimeAvgStats
        int amount = 0;
        switch (nextEventType)
        {
            case 1:
                //1 ხდომილების დამუშავების პროგრამა
                OrderArrival
                break;
            case 2:
                //2 ხდომილების დამუშავების პროგრამა
                Demand
                break;
            case 3:
                //3 ხდომილების დამუშავების პროგრამა
                Simulation
                break;
            case 4:
```

```
                        //4 ხდომილების დამუშავების პროგრამა
                        Evaluate
                        break;
                    }
                } while (nextEventType != 3);
            }
            Console.ReadKey();
        }
```

სურ. 1. ძირითადი პროგრამა C#-ზე.

ინიციალიზაციის პროგრამა - ქვეპროგრამა, რომლსაც სიმულაციური მოდელი მოყავს საწყის, ნულოვან მდგომარეობაში:

```
Initialize
   // სიმულაციური საათის ინიციალიზაცია
   double time = 0.0;

   // მდგომარეობის ცვლადების ინიციალიზაცია
  int intLevel = inventoryLevel;
  double timeLastEvent = 0.0;

  //სტატიკური მთვლელების ინიციალიზაცია
  double totalOrderingCost = 0.0;
  double areaShortage = 0.0;
  double areaHolding = 0.0;

   //ხდომილებს შორის დროის ინიციალიზაცია
   double[] timeNextEvent = { 0.0,
                           Math.Pow(10, 30),
                           time + Expon(interdemanTime),
                           simulationLength,
                           0.0
                            };
```

სურ. 2. ინიციალიზაციის პროგრამის C#-ის კოდი.

სინქრონიზაციის პროგრამა - ქვეპროგრამა, რომელიც ხდომილებათა სიაში ეძებს შემდეგ ხდომილებას და მოდელური დროის საათის ჩვენება გადაყავს ახალი ხდომილების დადგომის დროზე:

```
public void Timing(double[] timeNextEvent, int numEvents, out double time, out int
nextEventType)
        {

            double minTimeNextEvent = Math.Pow(10, 29);
            nextEventType = 0;

            //შემდეგი ხდომილების განსაზღვრა
            for (int i = 1; i <= numEvents; i++)
            {
                if (timeNextEvent[i] < minTimeNextEvent)
                {
                    minTimeNextEvent = timeNextEvent[i];
                    nextEventType = i;
                }
            }


            //მოდელური დროის საათის ჩვენების გადაყვანა ახალი ხდომილების დადგომის დროზე
            time = minTimeNextEvent;
        }
```

სურ. 3. სინქრონიზაციის პროგრამის C#-ის კოდი.

სტატისტიკური მთვლელების განახლების პროგრამა:

UpdateTimeAvgStats

```
double timeSinceLastEvent;
timeSinceLastEvent = time - timeLastEvent;
timeLastEvent = time;

//შენახვის ხარჯის სტატისტიკური მთვლელის განახლება
if (intLevel < 0)
{
   areaShortage -= intLevel * timeSinceLastEvent;
}

//დეფიციტით მიღებული დანაკარგის სტატისტიკური მთვლელის განახლება
else if (intLevel > 0)
{
   areaHolding += intLevel * timeSinceLastEvent;
}
```

სურ. 4. სტატისტიკური მთვლელების განახლების პროგრამის C#-ის კოდი.

1 ხდომილების დამუშავების პროგრამა - მიმწოდებლიდან შეკვეთის მიწოდება კომპანიისთვის:

OrderArrival

```
//მიწოდებული რაოდენობით მარაგის დონის შევსება
intLevel += amount;

//მომდევნო ხდომილების დაგეგმვა
timeNextEvent[1] = Math.Pow(10, 30);
```

სურ. 5. მიმწოდებლისგან კომპანიისთვის შეკვეთის მიწოდების პროგრამის C#-ის კოდი.

2 ხდომილების დამუშავების პროგრამა - კომპანიისგან პროდუქციაზე მოთხოვნა:

```
Demand
    int sizeDemand;

    //მოთხოვნის სიგრძის განსაზღვრა
    sizeDemand = RandomInteger(probDistrib);

    //მარაგის დონის შემცირება
    intLevel -= sizeDemand;

    //მომდევნო მოთხოვნის ხდომილების დაგეგმვა
    timeNextEvent[2] = time + Expon(interdemanTime);
```

სურ. 5. კომპანიისგან პროდუქციაზე მოთხოვნის პროგრამის C#-ის კოდი.

4 ხდომილების დამუშავების პროგრამა - თვის დასაწყისში მარაგის შეფასება:

```
Evaluate
      //მარაგის შეფასება
      if (intLevel < s[i])
      {
          //შეკვეთის რაოდენობა
          amount = S[i] - intLevel;

          //შეკვეთის ღირებულება
          totalOrderingCost = totalOrderingCost + setupCost + incrementalCost * amount;

          //შეკვეთის მიღების დაგეგმვა
          timeNextEvent[1] = time + Uniform(deliveryRangeMin, deliveryRangeMax);
       }

       //მაარგის შეფასების შემდეგი ხდომილების დაგეგმვა
      timeNextEvent[4] = time + 1.0;
```

სურ. 6. თვის დასაწყისში მარაგის შეფასების პროგრამის C#-ის კოდი.

3 ხდომილების დამუშავების პროგრამა - სიმულაციის დასასრული(ანგარიშების გენერატორი) - ქვეპროგრამა, რომელიც ითვლის სტატისტიკური მთვლელებიდან სისტემის მუშაობის კრიტერიუმების შეფასებებს და იძლევა მოდელირების დასრულების ანგარიშს:

```
Simulation
    double avgOrderingCost, avgHolgingCost, avgShortageCost, avgTotalCost;

    avgOrderingCost = Math.Round(totalOrderingCost / simulationLength, 2);
    avgHolgingCost = Math.Round(holdingCost * areaHolding / simulationLength, 2);
    avgShortageCost = Math.Round(shortageCost * areaShortage / simulationLength, 2);

    avgTotalCost = Math.Round(avgOrderingCost + avgHolgingCost + avgShortageCost, 2);
```

სურ. 7. ანგარიშების გენერატორის C#-ის კოდი.

შეკვეთის მიღების დრო, თანაბრად განაწილებული შემთხვევითი სიდიდე [a, b] შუალედზე:

```
public static double Uniform(double a, double b)
{
    double U;
    Random r = new Random();
    U = r.Next(1, 10) / 10;
    return a + U * (b - a);

}
```

სურ. 8. შეკვეთის მიღების დროის გენერაციის პროგრამის C#-ის კოდი.

მოთხოვნის დადგომის მომენტებს შორის დროის შუალედების განსაძღვრა, რომლისთვისაც ვიყენებთ ექსპონენციალურ განაწილებას:

```
public static double Expon(double meanInterdemand)
{
     double U;

     // U(0,1) ინტერვალიდან შემთხვევითი რიცხვის გენერაცია
     Random r = new Random();
     U = r.Next(1, 10);

     // დავაბრუნოთ ექპონენციალური შემთხვევითი რიცხვი
     return -meanInterdemand * Math.Log(U / 10);

}
```

სურ. 9. Expon ფუნქციის C#-ის კოდი.

მოთხოვნის მოცულობის განსაზღვრა, ვიყენებთ დისკრეტულ შემთხვევით სიდიდეს:

```
public static int RandomInteger(double[] probDistrib)
{
   double U;
   Random r = new Random();
   U = r.Next(1, 10) / 10.1;
   int i = 1;
   while (U >= probDistrib[i])
   {
       i++;
   }
   return i;

}
```

სურ. 10. მოთხოვნის მოცულობის განსაზღვრის ფუნქციის C#-ის კოდი.

2.2. სიმულაციური მოდელირების შედეგები და მათი ანალიზი

სურათ 11-ზე მოცემულია სიმულაციური მოდელირების შედეგი:

მარაგთა მართვის ერთი ამოცანის სიმულაციური მოდელირება

საწყისი მარაგის დონე	60 ერთეული
მოთხოვნის დადგომებს შორის დროის საშუალო მნიშვნელობა	0,1
სიმულაციის სიგრძე	120 თვე
შეკვეთის მიღების დრო	0,5-დან 1 თვემდე
შეკვეთილი პროდუქციის შესასყიდი ფასი	32$
დამატებითი ხარჯი ერთეულ პროდუქციაზე	3$
შენახვის ხარჯი ერთეულ პროდუქტზე	1$
დეფიციტით მიღებული დანაკარგი ერთეულ პროდუქტზე	5$
სტრატეგიების რაოდენობა	9

სტრატეგიები	საშუალო ჯამური ღირებულება	საშუალო შეკვეთის ღირებულება	საშუალო შენახვის ღირებულება	საშუალო დეფიციტის ღირებულება
(20,40)	123,99	97,78	10,65	15,56
(20,60)	119,97	90,56	16,78	12,63
(20,80)	122,74	86,45	25,24	11,05
(20,100)	129,13	82,56	37,98	8,59
(40,60)	121,24	97,23	22,78	1,23
(40,80)	124,49	88,65	33,72	2,12
(40,100)	133,4	83,45	48,42	1,53
(60,80)	141,07	99,12	40,78	1,17
(60,100)	142,46	89,78	51,56	1,12

სურ. 11. სიმულაციური მოდელირების შედეგი.

თვიური საშუალო ჯამური ღირებულება არის განსხვავებული ყოველი (s, S) წყვილისთვის. მაგ: s=20 -ისთვის, როცა S იცვლება 40-დან 100-მდე შენახვის გადასახადი იზრდება $10.65-დან $37.98-მდე საშუალოდ ერთ თვეში, მაშინ როცა დეფიციტის გადასახადის კლებულობს. მსგავსად ამისა, ფიქსირებული S = 100 და s იზრდება 20-დან 60მდე, დეფიციტის ღირეულება კლებულიბს ($8.59, $1.53 , $1.12) და იზრდება შენახვის გადასახადი ($37.98, $48.42, $51.56).

საბოლოოდ, ვითვლით ამ სამი კომპინენტის ჯამურ ღირებულებას , რომელიც განსხვავებულია ყოველი (s, S) წყვილისთვის. თუ გადავხედავთ ამ კრიტერიუმის მნიშვნელობებს , მარტივად შევამჩნევთ , რომ (20, 60) შემთხვევა არის საუკეთესო. საშუალო ჯამური ღირებულება ერთ თვეში არის $119.97.

დასკვნა

სისტემის შესწავლისა თუ ფუნქციონირების დასადგენად ჩვენ ვაკეთებთ გარკვეულ დაშვებებს. ეს დაშვებები ჩაიწერება მათემატიკურ-ლოგიკურ მიმართებებში, რომლებიც ყალიბდება გარკვეულ სტრუქტურაში - მოდელში. ზოგიერთ შემთხვევაში მიზნის მისაღწევად საკმარისია მათემატიკურ-ანალიზური მეთოდებით სარგებლობა, თუმცა უმრავლესობა სისტემებისა რთულია და მათთვის შეუძლებელია ისეთი რელური მოდელის შექმნა, რომლის რეალიზაცია, ამოხსნა, ჩაიწერება ანალიზურად. ასეთი სისტემებისთვის ფართოდ გამოიყენება სიმულაციური მოდელირება.

ნაშრომში მოყვანილია მარაგთა მართვის ერთი კონკრეტული დისკრეტულ-ალბათური მოდელი. ამ ამოცანაში მიზნის მისაღწევად საკმარისი არ არის მატემატიკურ-ანალიზური მედოდების გამოყენება, ამიტომ ამ სისტემისათვის ვიყენებთ სიმულაციურ მოდელირებას. სიმულაციურ მოდელს ვქმნით იმის გასარკვევად, თუ რა რაოდენობის პროდუქტი უნდა ჰქონდეს კომპანის თვის დასაწყისში, რომ კომპანიის ჯამური დანახარჯები იყოს მინიმალური. ჯამურ დანახარჯებში იგულისხმება პროდუქციის შეკვეთის ღირებულება, შენახვის ხარჯები და დეფიციტით მიღებული დანაკარგი.

სიმულაციას ვახდენთ 120 თვის განმავლობაში. შესაკვეთი პროდუქციის რაოდენობის განსაზღვრისათვის ვიყენებთ (s, S) სტრატეგიას, სადაც s არის მარაგის კრიტიკული რაოდენობა, ხოლო S - თვის დასაწყისში მარაგის რაოდენობა. ვიყენებთ 9 განსხვავებულ სტრატეგიას და თითოეული სტრატეგიისთვის ვითვლით შეკვეთის, შენახვის და დეფიციტით მიღებული ხარჯების საშუალო ღირებულებას ერთ თვეში. საბოლოოდ კი, ვიღებთ ხარჯების ჯამურ ღირებულებას. და ბოლოს, ვარჩევთ საუკეთესო (s, S) სტრატეგიას მინიმალური საშუალო ჯამური ღირებულების მიხედვით.

გამოყენებული ლიტერატურა:

[1] **Averill M. Law and w. David Kelton –** Simulation Modeling & Analysis.