საქართველოს ტექნიკური უნივერსიტეტი

ი. როდონაია. ლ. თედეშვილი

საქმიანი პროცესების ანალიზი და დიზაინი

დამხმარე სახელმძღვანელო

თბილისი
2016

საქართველოს ტექნიკური უნივერსიტეტი

ი. როდონაია. ლ. თედეშვილი

საქმიანი პროცესების ანალიზი და დიზაინი

დამხმარე სახელმძღვანელო

თბილისი
2016

უაკ 338.22

რ-738

ი.როდონაია. ლ.თედეშვილის ნაშრომზე: **საქმიანი პროცესების ანალიზი და დიზაინი;** დამხმარე სახელმძღვანელო, 2016 წ.-64გვ.

ნაშრომში წარმოდგენილია საქმიანი პროცესების ანალიზის და დაგეგმარების ძირითადი საკითხები. ნაშრომის მთავარი მიზანია უზრუნველყოს სტუდენტები ანალიტიკური ინსტრუმენტების სიმრავლის ამომწურავი გაცნობიერებით, რომელიც შეიძლება გამოყენებულ იქნას საქმიანი პროცესების ანალიზისათვის, ადეკვატური შეფასებისთვის და, საბოლოოდ, მათი ეფექტური დიზაინისათვის.

დამხმარე სახელმძღვანელო „საქმიანი პროცესების ანალიზი და დიზაინი“ მნიშვნელოვან დახმარებას გაუწევს ინფორმატიკის სპეციალობის და ბაკალავრიატის მე-3 კურსის სტუდენტებს.

ნაშრომის მიმართ შენიშვნები არ გამაჩნია და მიმაჩნია, რომ დამხმარე სახელმძღვანელო „საქმიანი პროცესების ანალიზი და დიზაინი“ შეიძლება გამოქვეყნდეს საქართველოს ტექნიკურ უნივერსიტეტში.

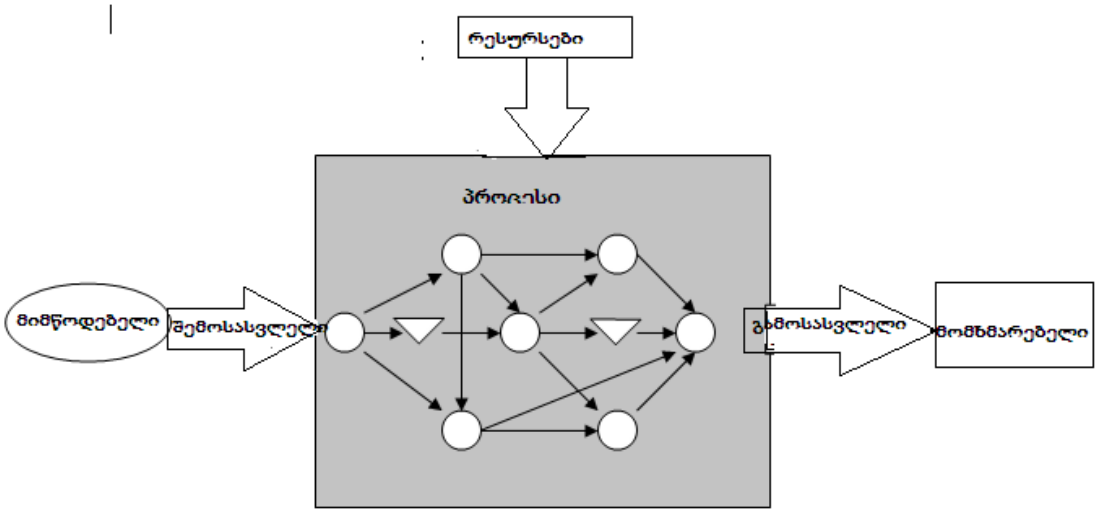
რეცენზენტი: საქართველოს ტექნიკური უნივერსიტეტის
ინფორმატიკისა და მართვის სისტემების ფაკულტეტის
სრული პროფესორი
თ .მაჭარაძე

საქართველოს ტექნიკური უნივერსიტეტი, თბილისი 2016
ISBN 978-9941-0-9432-3

შინაარსი

შესავალი

ბიზნეს პროცესი არის კარგად განსაზღვრული საზღვრების და თანმიმდევრობის კავშირების მქონე **აქტივობის და ბუფერების ქსელი, რომელიც** იყენებს რესურსებს იმისათვის, რომ გარდაქმნას სისტემის შემოსასვლელი სისტემის გამოსასვლელში მომხმარებლის მოთხოვნების დაკმაყოფილების მიზნით. (სურ.1)



სურ.1

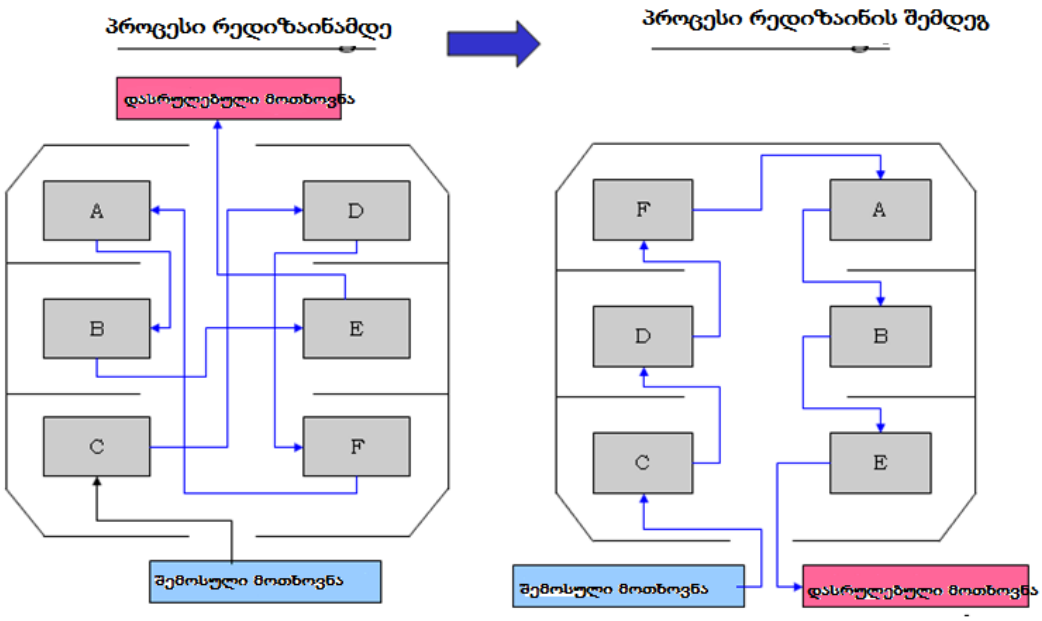
ბიზნეს პროცესების დიზაინისა და დაგეგმარების ძირითადი ინსტრუმენტები.

ზოგადი პროცესის ბარათები. (სურ.2)

აქტივობა	მიმდინარე პროცესი			ხელალხლა დამუშავებული პროცესი			სხვაობა	
	ნომერი	დრო	%	ნომერ	დრო	%	ნომერი	დრო
ოპერაციები	5	30	10	5	30	37.5	0	0
შემოწმება	3	60	20	1	20	25.0	-2	-40
ტრანსპორტირება	10	120	40	2	20	25.0	-8	-100
შენახვა	0	0	0	0	0	0	0	0
შეყოვნება	7	90	30	1	10	12.5	-6	-80
ჯამური	**25**	**300**	**100**	**9**	**80**	**100**	**-16**	**-220**

სურ.2

პროცესების ნაკადების დიაგრამები .სურ.3

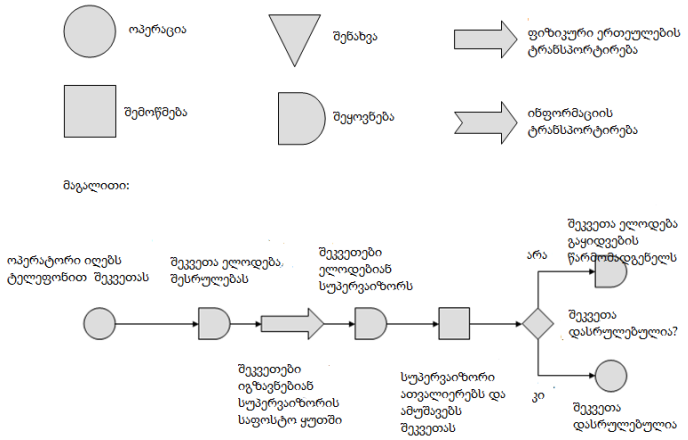


სურ.3

აქ გვაქვს პროცესი, ექვსი სამუშაო გუნდით (A-დან F-დე), ფიზიკურად ორგანიზებული როგორც ნაჩვენებია ზემოთ (მარცხენა მხარეს). ცხადია, ზედმეტი

ტრანსპორტირება ხდება სამუშაო ჯგუფების მიმდინარე განლაგების გამო. საქმიანობების თანმიმდევრობაა: C, D, F, A და E. სამუშაო ჯგუფების გადანაწილება ისე, როგორც ნაჩვენებია ნახაზის მარჯვენა მხარეს, გვაძლევს პროცესის უფრო ეფექტურ სქემასსურ. (სურ.4)

ნაკადების გრაფები (Flow charts)



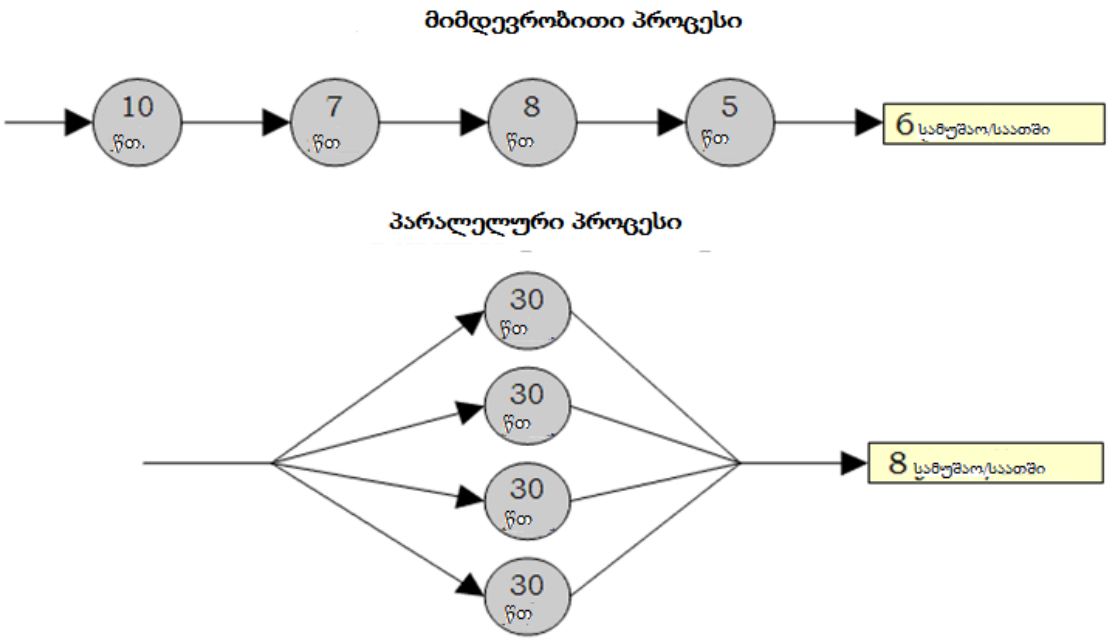
სურ.4

$$\text{საქმიანობის საშუალო დრო} = \frac{(\text{ერთეულის დამუშავების დრო})*(\text{პარტიის ზომა})+\text{მომზადების დრო}}{\text{ეფექტურობა}}$$

მაგალითი: ერთეულის შემოწმება (ინსპექცია) : 3 წუთი; ერთი პარტიის (batch, lot) რაოდენობა:10 ერთეული; 15 წუთი სჭირდება იმისათვის, რომ მომზადდეს ახალი პარტიის ინსპექცია; ინსპექტორი-მუშაკი გამოუცდელია და ინსპექციისთვის ხარჯავს 25% მეტ, ვიდრე გამოცდილი.

საქმიანობის საშუალო დრო=((3*10)+15)/0.75=60 წუთი

მიმდევრობითი პროცესების დამუშავების დროის მინიმიზაცია (სურ.5)



სურ.5

მიმდევრობითი პროცესის ყველაზე ხანგრლივი საქმიანობა -10 წუთი (ე.წ. ვიწრო ადგილი, **bottleneck**). პროცესის გამოსასვლელი (**output**) = 60 წუთი/10წუთი=6 სამუშაო საათში. პროცესის **ეფექტურობა=** (10+7+8+5)/(10*4)=75%

პარალელური პროცესი: თითოეული აქტივობა:30 წუთი. ახლა პროცესის გამოსასვლელი (**output**)=4*(60/30)=8სამუშაო საათში. პროცესის **ეფექტურობა** =30/30=100%

პროცესების და სამუშაოების ტერმინოლოგია

Makespan - პროცესის ყველა საქმიანობის ჯამური დრო. ბიზნეს-პროცესების ერთ-ერთი მნიშვნელოვანი ამოცანაა **makespan**-ის მინიმიზაცია.

Tardiness (დაგვიანება) - სამუშაოს დასრულების დაგვიანებული დრო (ანუ დრო, რომლითაც სამუშაო დასრულდა უფრო გვიან, ვიდრე იყო დაგეგმილი (**due date**). ბიზნეს-პროცესების ერთ-ერთი მნიშვნელოვანი ამოცანა აგრეთვე არის **tardiness** -ის მინიმიზაცია. მეორე ამოცანაა: დაგვიანებული სამუშაოების რაოდენობის მინიმიზაცია.

სამუშაოების შესრულების პრიორიტეტები:

First-In-First-Out (FIFO) - პირველი სრულდება ის სამუშაო, რომელიც დანარჩენებზე უფრო ადრე მოსულია(arrival time - მოსვლის დრო)

Earliest-Due-Date first (EDD) - პირველი სრულდება ის სამუშაო, რომლის დაწყების დრო (**due date**) არის ყველაზე ახლო მიმდინარე მომენტისთვის

Shortest Processing Time first (SPT) - პირველი სრულდება ის სამუშაო, რომლის შესრულების ხანგრძივლობის დრო (processing time) არის ყველაზე მოკლე.

ცნობილია, რომ **ერთი** მომსახურე მოწყობილობისთვის (**server**) EDD არის ყველაზე ეფექტური მაქსიმალური დაგვიანების დროის მინიმიზაციისთვის, ხოლო SPT ეფექტურია საშუალო შესრულების დროის მინიმიზაციისთვის.

პროცესის შესრულების დრო -გამტარუნარიანობა (Process Throughput)

პროცესის სამუშაოების (კლიენტების) შემოდინების (**inflow**) და გადინების (**outflow**) სიჩქარე (ინტენსივობა) (**rate**):

IN(t) - სამუშაოების (კლიენტების) შემოდინების ინტენსივობა დროის მომენტში *t*

OUT(t) - სამუშაოების (კლიენტების) გადინების ინტენსივობა დროის მომენტში *t*

IN - სამუშაოების (კლიენტების) შემოდინების **საშუალო** ინტენსივობა მთელი პროცესის შესრულების დროს

OUT - სამუშაოების (კლიენტების) გადინების **საშუალო** ინტენსივობა მთელი პროცესის შესრულების დროს

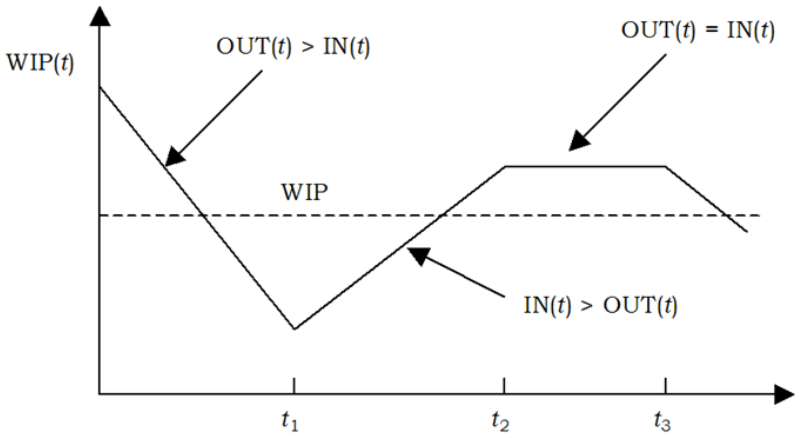
სტაბილურ სისტემაში: IN=OUT=λ

– λ = პროცესის სამუშაოების შემოდინების საშუალო ინტენსივობა

– **λ** =პროცესის შესრულების დრო - გამტარუნარიანობა (process throughput), იზომება დასრულებული სამუშაოების რაოდენობით დროის ერთეულში.

პროცესის დაუსრულებელი სამუშაოები (**Work-In-Process, WIP**)

ყველა ის სამუშაო, რომლის დამუშავება პროცესში დაწყებულია, მაგრამ არ არის ჯერჯერობით დასრულებული.



პროცესის ციკლის დრო (**Process Cycle Time).**

სამუშაოს შემოსვლის და დასრულების დროის სხვაობა (**cycle time**). აგრეთვე ცნობილია როგორც პროცესის შესრულების დრო (Process Throughput). ციკლის დროში შედის: დამუშავების დრო (Processing time), ინსპექციის დრო (Inspection time), ტრანსპორტირების დრო (Transportation time), შენახვის დრო (Storage time), ლოდინის დრო (Waiting time)

ლიტლის (**Little's**) **ფორმულა**

Little's Formula: WIP = λ·CT

სადაც: გამტარუნარიანობა (Throughput) (= λ) and ციკლის დრო (Cycle time) (=CT).

ზოგიერთი შედეგი: რაც მოკლეა ციკლის დრო, მით ნაკლებია WIP თუ λ იზრდება ⇒ იმისათვის, რომ WIP შევინარჩუნოთ ძველ დონეზე, აუცილებელია CT-ის შემცირება

ციკლის დროის ანალიზი.

აუცილებელია შემდეგი შემთხვევების განხილვა:

- სამუშაოს ხელმეორე შესრულება (აღმოჩენილი წუნის გამო) (**Rework)**
- მრავალგეზიანი შესრულება (**Multiple paths)**
- სამუშაოების პარალელური შესრულება (**Parallel activities**)

სამუშაოს ხელმეორე შესრულება - Rework

აღნიშვნები:

- T = ხელმეორე ციკლში შემავალი ყველა სამუშაოების დროების ჯამი
- r = იმ სამუშაოების, რომლებიც უნდა ხელმეორედ დამუშავდეს, პროცენტი (rejection rate)

დავუშვათ, რომ ხელმეორე დამუშავებაზე გაგზავნილი სამუშაო აღარ საჭიროებს დამუშავებას . მაშინ **საშუალო** ციკლის დრო :

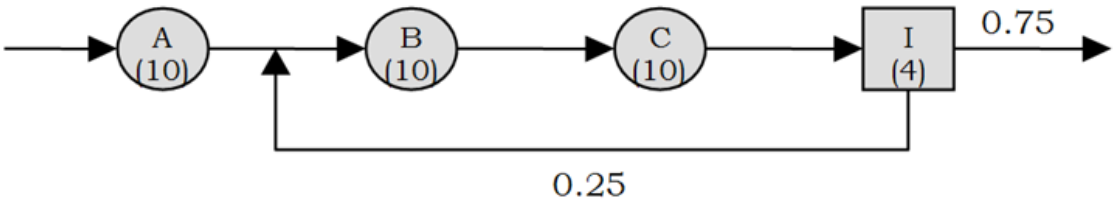
$$CT = (1+r)T$$

თუ ხელმეორე დამუშავებაზე გაგზავნილი სამუშაოსთვის კვლავ შეიძლება დასჭირდეს დამუშავება, მაშინ **საშუალო** ციკლის დრო:

$$CT = T/(1-r)$$

მაგალითი.

პროცესი შედგება 3 აქტივობისგან A, B & C, თითოეული სრულდება საშუალოდ 10 წუთში. შემოწმების (ინსპექციის) აქტივობა სრულდება 4 წუთში. აქტივობის X% იგზავნება განმეორებით დამუშავებაზე. (სურ.7)



სურ.7

გამოთვალეთ პროცესის საშუალო ციკლის დრო, თუ:

- არც ერთი სამუშაო იგზავნება ხელმეორე დამუშავებაზე
- სამუშაოების 25% იგზავნება ხელმეორე სამუშაოზე, მაგრამ მხოლოდ ერთხელ
- სამუშაოების 25% იგზავნება ხელმეორე სამუშაოზე, და ეს შეიძლება მოხდეს არაერთხელ

მრავალგეზიანი შესრულება (Multiple paths)

მაგალითად, სამუშაოები შეიძლება გაიყოს 'სწრაფ' და 'ნორმალურ' სამუშაოებზე

აღნიშვნები:

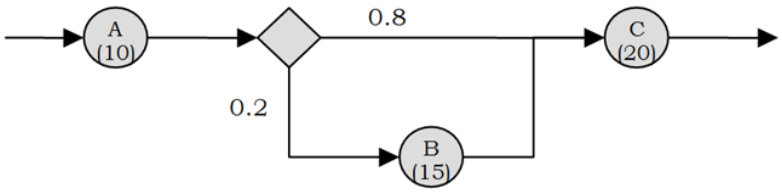
p_i - ალბათობა, რომ სამუშაო იქნება გაგზავნილი i -ურ გეზით

T_i - i -ურ გეზით გაგზავნილი სამუშაოს შესრულების დრო

მაშინ **საშუალო** ციკლის დრო: (სურ.8)

$$CT = p_1T_1+p_2T_2+\ldots+p_mT_m = \sum_{i=1}^{m} p_iT_i$$

მაგალითი.



სურ.8

გამოთვალეთ პროცესის საშუალო ციკლის დრო.

სამუშაოების პარალელური შესრულება (Parallel activities) (სურ.9)

აღნიშვნები:

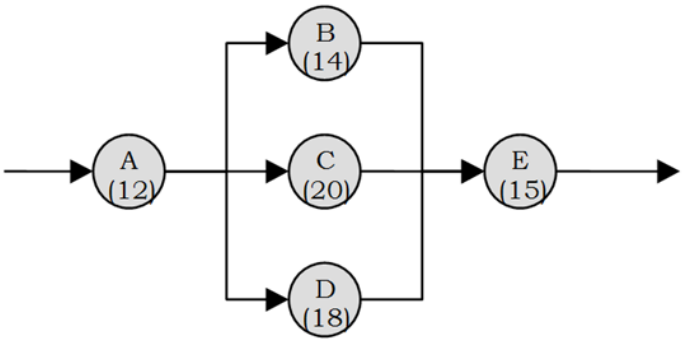
M სამუშაო სრულდება პარალელურად

T_i - i -ურ შტოში შესრულების დრო

მაშინ **საშუალო** ციკლის დრო:

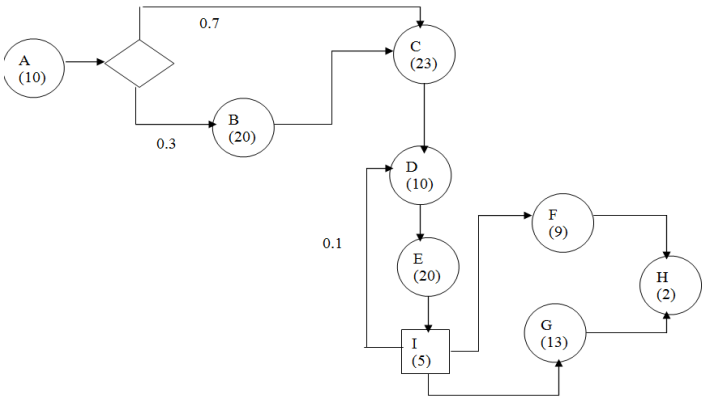
$$CT_{parallel} = Max\{T_1, T_2, \ldots, T_M\}$$

მაგალითი.



სურ.9

ამოცანა 1.1. (სურ.10)



სურ.10

გამოთვალეთ პროცესის შესრულების საშუალო დრო.

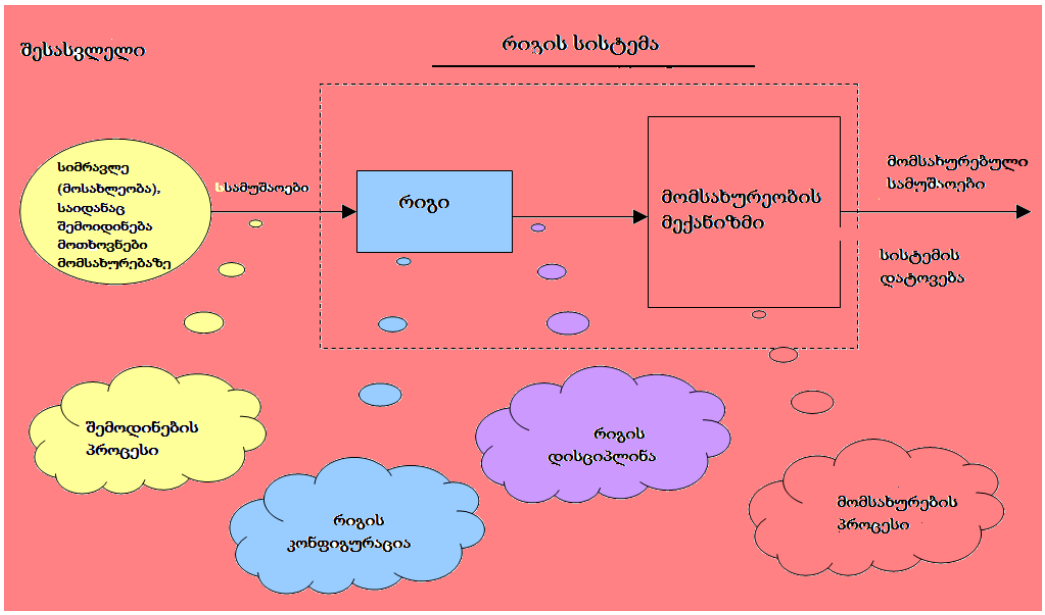
რიგების სისტემები

რეალური სამყაროდან აღებული რიგების სისტემების მაგალითები:

კომერციული რიგების სისტემები:

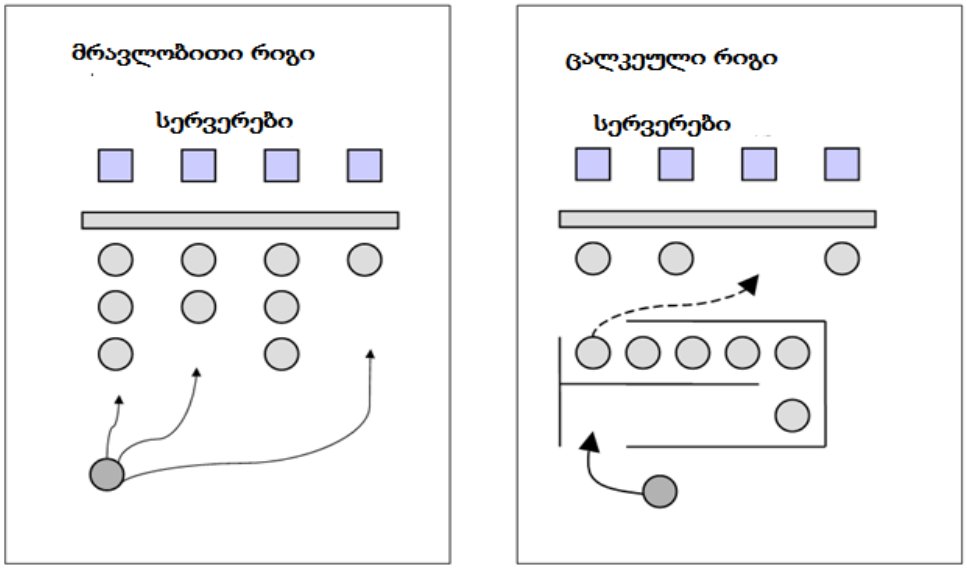
- კომერციული ორგანიზაციები, რომლებიც ემსახურებიან გარე მომხმარებელს
- მაგალითად, სტომატოლოგი, ბანკი, ბანკომატები, ბენზინგასამართი სადგურები, ავტოფარეხი .
- **სატრანსპორტო მოსამსახურე სისტემები**
- მანქანები, რომლებიც წარმოადგენენ კლიენტებს ან სერვერებს˜
- მაგალითად, მანქანები, რომლებიც დგანან შუქნიშნებთან, სატვირთო მანქანები ან გემები, რომლებიც ელოდებიან დატვირთვას, ტაქსები, სახანძრო მანქანები, ლიფტები, ავტობუსები
- ოციალური მომსახურების სისტემიბი
- მაგალითად, იურიდიული ფირმები, რენტგენის კაბინეტები საავადმყოფოებში,

ბაზისური რიგის პროცესის კომპონენტები: (სურ.11)



სურ.11

მაგალითი: ორი რიგის კონფიგურაციები



ზოგადი რიგების მოდელი.

მომსახურეობის (**service**) დრო და **შემოსვლებს შორის (interarrival)** დრო მიღებულია როგორც დამოუკიდებელი და იდენტურად განაწილებული.
ზოგადი აღნიშვნა: A/B/C
A - შემოსვლებს შორის დროის განაწილება
B - მომსახურეობის დროის განაწილება
C - პარალელური სერვერების რაოდენობა
განაწილებების ზოგადი აღნიშვნა
M - მარკოვის (ანუ ექსპონენციალური) განაწილება
D - დეტერმინისტური (ანუ არაშემთხვევითი) განაწილება
G -ზოგადი განაწილება
მაგალითი: M/M/2 - რიგების სისტემა, რომელშიც სამუშაოების (ანუ კლიენტების) შემოსვლებს შორის დრო არის ექსპონენციალურად განაწილებული, სამუშაოების დამუშავების დრო არის აგრეთვე ექსპონენციალურად განაწილებული; სისტემაში არის ორი სერვერი
მაგალითი. საბანკო მენეჯერს სურს გამოიყენოს რიგების მოდელის ანალიზი ბანკომატებთან რიგების შექმნის გასაანალიზებლად. რამდენიმე თანამშრომელმა გამოიყენა წამზომი შემოსვლებს შორის **(interarrival**) და მომსახურეობის დროების შესაგროვებლად

შემოსვლებს შორის **(interarrival)** გაზომილი დრო:

8	5	12	33	13	16	9	5	7
11	26	9	5	33	10	24	23	29
4	84	45	46	1	27	25	17	29
10	4	21	31	50	64	30	17	14
16	11	19	88	11	16	2	21	25
4	30	19	4	13	35	94	37	18
24	2	40	56	2	2	154	13	82
41	16	43	28	14	30	11	12	15
20	7	15	28	19	3	31	58	14
11	21	17	36	3	17	38	8	3
84	10	42	16	10	112	7	46	17
3	38	5						

შემოსვლებს შორის საშუალო დრო = 25.3

მომსახურეობის გაზომილი დრო:

11	10	15	20	27	63	16	13	33
8	18	18	12	20	47	17	16	19
41	33	16	7	36	16	19	15	14
43	21	32	21	17	46	10	7	27
26	50	16	7	19	15	23	27	23
18	14	10	7	7	21	16	15	17
20	12	12	18	17	17	19	11	6
18	33	34	14	16	11	40	48	9
16	28	44	46	13	11	21	14	16
8	28	30	23	18	24	30	57	31
62	19	59	26	16	22	27	7	40
17	22	11						

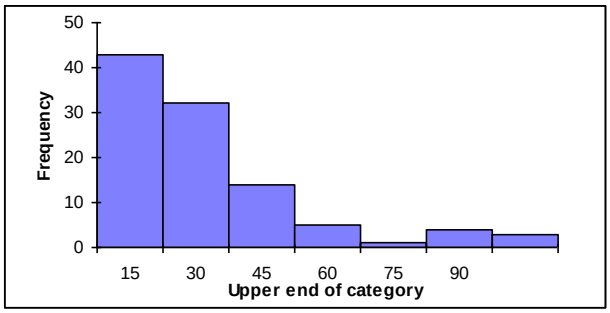
მომსახურეობის საშუალო დრო = 22.3

ბანკის მენეჯერს სურს იცოდეს (ამ მონაცემზე დაყრდნობით), იქნება თუ არა გონივრულად ვივარაუდოთ , რომ შემოსვლებს შორის დრო და მომსახურეობის დრო არის ექსპონენციალურად განაწილებული.

იმისათვის, რომ დავრწმუნდეთ, რომ ეს დროები შეესაბამება ექსპონენციალურ განაწილებას, ჩვენ შეგვიძლია ავაგოთ **ჰისტოგრამები** (**histograms**) შემოსვლებს შორის და მომსახურეობის დროებისათვის . ეს შეიძლება გაკეთდეს **Excel** -ის ინსტრუმენტით **Data Analysis** მენიუს გამოყენებით ჰისტოგრამის რეჟიმში. ამ რეჯიმში უნდა განვსაზღვროთ „bin range” (ინტერვალის სიგანე), მაგალითად, განვსაზღვროთ ინტერვალები 15, 30, 45, 60, 75, and 90. ჰისტოგრამა იქმნება იმით, რომ

დაითვლება დაკვირვების რაოდენობა, რომელიც ხვდება თითოეულ ინტერვალში (**bin**). შემოსვლებს შორის დროებისთვის მივიღებთ: (სურ.12)

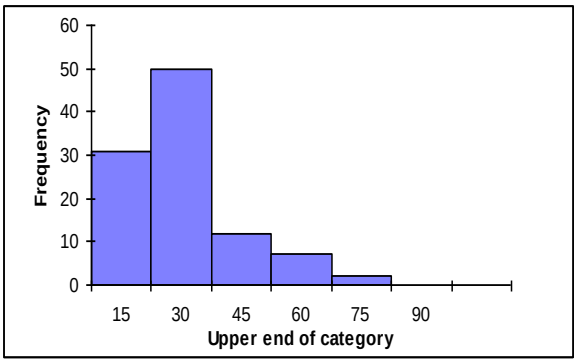
Bin	*Frequency*
15	43
30	32
45	14
60	5
75	1
90	4
More	3



სურ.12

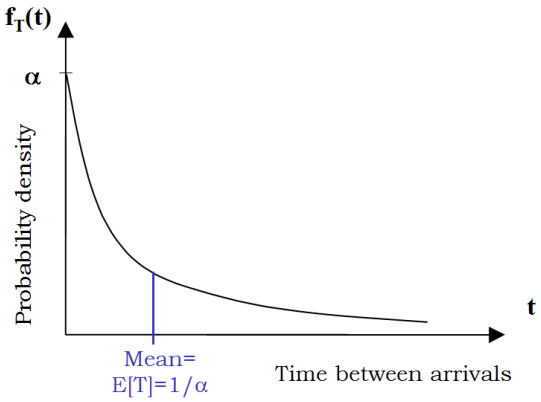
მომსახურეობის დროებისთვის მივიღებთ: (სურ.13)

Bin	*Frequency*
15	31
30	50
45	12
60	7
75	2
90	0
More	0



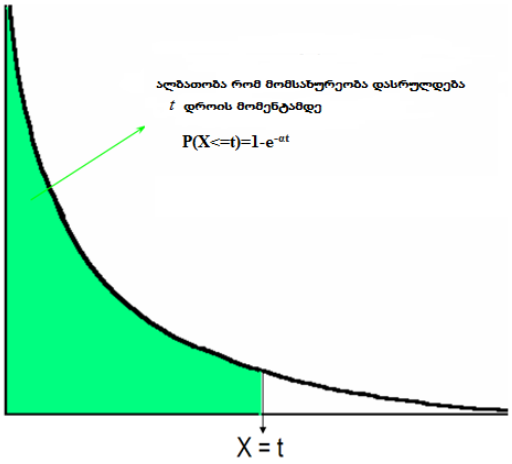
სურ.13

შემოსვლებს შორის დრო ხასითდება იმით, რომ თითოეული კლიენტის შემოსვლის დრო არაფრად არ არის დამოკიდებული წინა კლიენტის შემოსვლის დროზე. (ე.წ. memoryless (მეხსიერების არქონის) თვისება). თუ ჩვენ შევამცირებთ bin-ების სიგანეს (ვთქვათ, 5, 10, 15,.....), მაშინ მივიღებთ უფრო **გლუვ** მრუდეს (ე.წ. სიხშირის (სიმკვრივის (density)) ფუნქციას, რომლის საშუალოა (mean) E[t]=1/ α, ჩვენ შემთხვევაში E[t]= 25.3 და, შესაბამისად, პარამეტრი α=1/E[t]=1/25.3=0.04): (სურ 14)



სურ.14

იმის გამო, რომ სიხშირე (ალბათობის სიმჭიდროვის ფუნქცია) მცირდება მუდმივად მარცხნიდან მარჯვნივ, მისი მოდა (mode) (ყველაზე სავარაუდო მნიშვნელობა) არის 0. ანუ , შემოსვლებს შორის დრო **T** უფრო ახლოა 0-თან , ვიდრე ნებისმიერი სხვა დროის მნიშვნელობასთან. ყოველთვის არსებობს 63.2% შანსი იმისა, რომ შემოსვლებს შორის დრო ნაკლებია ან ტოლია საშუალო მნიშვნელობის E [T] = 1 / a. ამავე დროს, სიმკვრივის ფუნქციის გრძელი მარჯვენა კუდი მიუთითებს, რომ დიდი შემოსვლებს შორის დრო აგრეთვე შესაძლებელია. ეს ნიშნავს, რომ ექსპონენციალური განაწილება (სხვა განაწილების კანონებთან შედარებით) აღწერს ყველაზე შემთხვევით პროცესს. (სურ.15)



სურ.15

მეორეს მხრივ, მომსახურეობის დროებისთვის ჰისტოგრამა გარკვეულწილად განსხვავდება შემოსვლებს შორის დროების ექსპონენციალურ ალბათობის სიმჭიდროვის ფუნქციისაგან. მისი უმაღლესი სიმაღლის ბინი (bin) არ არის ყველაზე მარცხენა მხარეს, არამედ შეესაბამება ბინს ინტერვალში 30 დან 45 წამამდე. იმის გათვალისწინებით, თუ როგორ მუშაობენ ბანკომატები, ეს არ არის გასაკვირი. არსებობს გარკვეული მინიმალური დრო, რომელიც საჭიროა ნებისმიერ მომხმარებელის დასამუშავებლად (განურჩევლად ამოცანისა), ასე რომ, ყველაზე სავარაუდო დრო არ არის ნულთან ახლოს (ექსპონენციალური განაწილებისგან განსხვავებით).

მაგალითი. დავუშვათ, რომ ფოსტაში მომსახურეობის მოლოდინის დრო **T** არის ექსპონენციალურად განაწილებული 3 წუთის საშუალოს მნიშვნელობით.

თუ შევდივართ ფოსტაში უშუალოდ წინა სხვა მომხმარებელის შემდეგ, რა არის ალბათობა იმისა, რომ მოგვიწევს 5 წუთზე მეტი ლოდინი?

რადგან E(T)=1/α=3, მაშინ α =1/3. ამიტომ :
$P(T>5)=1-P(T<=5)=1-F(5)=1-(1-e^{-1/3 * 5})=e^{-5/3} \approx 0.189$
იმავე პირობებით, რა არის ალბათობა იმისა, რომ მოგვიწევს 2-დან და 4 წუთამდე შორის ლოდინი? აქ ჩვენ გამოვთვლით შემდეგნაირად:
$P(2<=T<=4)=F(4) - F(2)=(1- e^{-4/3}) – (1-e^{-1/3})=e^{-2/3} – e^{-4/3} \approx 0.250$

ტერმინოლოგია და აღნიშვნები.

- სისტემის მდგომარეობა = მომხმარებელთა რაოდენობა სისტემაში.
- რიგის სიგრძე = (სისტემის მდგომარეობა) - (მომხმარებელთა რაოდენობა, რომელთა მომსახურეობა ხდება ამჟამად)

N(t) = სისტემაში მომხმარებელთა / სამუშაოების რაოდენობა ***t*** დროში
$P_n(t)$ = ალბათობა იმისა, რომ არსებობს ***n*** მომხმარებელი / სამუშაო სისტემაში დროის ***t*** მომენტში
λ_n **=** სისტემაში შემოსვლების შორის საშუალო დროების ინტენსივობა (rate) (=მომხმარებლების / სამუშაოების მოსალოდნელი რაოდენობა დროის ერთეულში, როდესაც სისტემაში ***n*** მომხმარებელია)

μ_n = სისტემაში მომსახურეობის საშუალო დროების ინტენსივობა (rate) (=სისტემიდან გამოსული მომხმარებლების / სამუშაოების მოსალოდნელი რაოდენობა დროის ერთეულში, როდესაც სისტემაში ***n*** მომხმარებელია))

ρ = მომსახურების მოწყობილობის(სერვერის)გამოყენების ფაქტორი (=დროის მოსალოდნელი წილი, როდესაც ეს მომსახურების მოწყობილობა არის ხმარებაში)

მაგალითი. განვიხილოთ სატელეფონო ლაპარაკების კომუტატორი იურიდიულ ფირმაში. საშუალოდ, 20 ზარი საათში შემოდის კომუტატორში, და ოპერატორს სჭირდება საშუალოდ 2 წუთი იმისათვის, რომ გაიგოს , თუ რა უნდა მომხმარებელს (ზარის ავტორს) და გადაუგზავნოს ზარი საჭირო პიროვნებას.

ამ მაგალითისთვის გვაქვს: λ=20 ზარი/საათში, μ=60/2=30 ზარი/საათში, კომუტატორის გამოყენების ფაქტორი ρ = 20/30 = 67%.

სისტემის სტაბილური მდგომარეობის ანალიზი

ფორმულები და აღნიშვნები

- ალბათობა იმისა , რომ სისტემაში არ იმყოფება არც ერთი მომხმარებელი (სამუშაო):

$$P_0 = 1-\rho$$

- ალბათობა იმისა , რომ სისტემაში იმყოფება ზუსტად ***n*** მომხმარებელი (სამუშაო):

$$P_n = \rho^n(1- \rho)$$

- ალბათობა იმისა , რომ სისტემაში იმყოფება ***k*** ან მეტი მომხმარებელი (სამუშაო):

$$P(n\geq k) = \rho^k$$

- ალბათობა იმისა , რომ სისტემაში იმყოფება არანაკლებ ერთი მომხმარებელი (სამუშაო):

$$P(n\geq 1)=1-P_0=\rho$$

- სისტემაში მყოფი მოხმარებლების(სამუშაოების) საშუალო რაოდენობა:

$$L=\rho/(1- \rho)= \lambda/(\mu-\lambda)$$

- რიგებში მყოფი მოხმარებლების(სამუშაოების) საშუალო რაოდენობა:

$$L_q=L-\rho= \rho^2/(1- \rho)= (\rho \lambda)(\mu- \lambda)$$

- საშუალო დრო, რომელსაც მომხმარებელი (სამუშაო) ატარებს სისტემაში:

$$W=L/ \lambda = 1/(\mu- \lambda)$$

- საშუალო დრო, რომელსაც მომხმარებელი (სამუშაო) ატარებს რიგებში:

$W_q = L_q/\lambda = \rho/(\mu - \lambda)$

$$\mathbf{W_q = L_q/\lambda = \rho/(\mu - \lambda)}$$

მაგალითი. სადაზღვევო კომპანია ღებულობს საშუალოდ 40 მოთხოვნას კვირაში ახალი პოლისების გასაფორმებლად. მუშაკების ერთ გუნდს შეუძლია გააფორმოს საშუალოდ 50 მოთხოვნა კვირაში. რა არის იმის ალბათობა, რომ ახალ მოსულ მოთხოვნას მოუწიოს ლოდინი? ანუ, რა არის იმის ალბათობა, რომ გუნდი არის დაკავებული ახალი მოთხოვნის მოსვლის დროს?

ამოხსნა. რადგან $\lambda=40$ და $\mu=50$, მაშინ $\rho=40/50=0.8$ და $P(n>=1)= \rho^1 = 0.8^1 = 0.8$.

ალბათობა იმისა, რომ გუნდი არის უსაქმოდ: $P_0=1-\rho=1-0.8=0.2$

მოთხოვნების მოსალოდნელი რაოდენობა: $L= \lambda/(\mu - \lambda)=40/(50-40)=4$ მოთხოვნა

რიგში მყოფი მოთხოვნების მოსალოდნელი რაოდენობა: $L_q = (\rho\lambda)/(\mu - \lambda)=(0.8*40)/(50-40)=3.2$ მოთხოვნა

საშუალო დრო, რომელსაც მომხმარებელი ატარებს სისტემაში (ანუ ციკლის საშუალო დრო): $W=1/(\mu - \lambda) =1/(50-40) =0.1$ კვირა $=0.5$ days $=4$ საათი (მიღებულია, რომ კვირაში 5 დღეა და დღეში 8 სამუშაო საათი)

რიგში ყოფნის (ანუ მოთხოვნის ლოდინის) საშუალო დრო: $W_q = \rho/(\mu - \lambda)=0.8/(50-40)=0.08$ კვირა$=0.4$ days$=3.2$ საათი

ამ გამოთვლებიდან ჩანს, რომ გუნდი დაკავებულია დროის 80% . ციკლის საშუალო დრო 4 საათია, საიდანაც 3.2 საათი იხარჯება რიგში ყოფნაზე. ანუ ციკლის მხოლოდ 20% იხარჯება ფულის მომტან აქტივობაზე (value-adding activities), და 80% - ფულის არმომტან ლოდინის დროზე(non-value-adding waiting time).

მოდელი M/M/c (სერვერებისრაოდენობა c >=1)

- სერვერების გამოყენების ფაქტორი $\rho=\lambda/(c\mu)$
- ალბათობა იმისა , რომ სისტემაში არ იმყოფება არც ერთი მომხმარებელი (სამუშაო):

$$P_0 = \frac{1}{\sum_{n=0}^{c-1} \frac{(\lambda/\mu)^n}{n!} + \frac{(\lambda/\mu)^c}{c!(1-\rho)}}$$

- ალბათობა იმისა , რომ სისტემაში ყველა სერვერი დაკავებულია:

$$P(n \geq c) = \frac{(\lambda/\mu)^c}{c!(1-\rho)} P_0$$

- ალბათობა იმისა , რომ სისტემაში იმყოფება ზუსტად ***n*** მომხმარებელი (სამუშაო):

$$P_n = \begin{cases} \frac{(\lambda/\mu)^n}{n!} P_0 & \text{for } n = 1,2,\ldots c \\ \frac{(\lambda/\mu)^n}{c!c^{n-c}} P_0 & \text{forn} = c, c+1,\ldots.. \end{cases}$$

- სისტემაში მყოფი მოხმარებლების (სამუშაოების) საშუალო რაოდენობა:

$$L = \frac{(\lambda/\mu)^c \rho}{c!(1-\rho)^2} P_0 + \frac{\lambda}{\mu}$$

- რიგებში მყოფი მოხმარებლების(სამუშაოების)საშუალო რაოდენობა:

$$L_q = \frac{(\lambda/\mu)^c \rho}{c!(1-\rho)^2} P_0 = L - \frac{\lambda}{\mu}$$

- საშუალო დრო, რომელსაც მომხმარებელი (სამუშაო) ატარებს სისტემაში:

$$W = \frac{L}{\lambda} = \frac{L_q}{\lambda} + \frac{1}{\mu}$$

- საშუალო დრო, რომელსაც მომხმარებელი (სამუშაო) ატარებს რიგებში:

$$W_q = \frac{L_q}{\lambda}$$

მაგალითი. დავუშვათ,რომ წინა მაგალითის პირობებში სერვერების (მოსამსახურე გუნდების რაოდენობა) გახდა 2. გავიხსენოთ, რომ λ=40 მოთხოვნა კვირაში და და μ=50 მოთხოვნის დამუშავემა კვირაში. მაშინ გვაქვს სისტემა **M/M/2** და შემდეგი გამოთვლები:

$$\rho = \frac{\lambda}{c\mu} = \frac{40}{2*50} = 0.4$$

იმისათვის, რომ მივიღოთ პასუხი, უნდა გამოვთვალოთ ალბათობა, რომ სადაზღვევო კომპანიაში არ იმყოფება არც ერთი პოლისის გაფორმების მოთხოვნა:

$$P_0 = \frac{1}{\sum_{n=0}^{c-1} \frac{(\lambda/\mu)^n}{n!} + \frac{(\lambda/\mu)^c}{c!(1-\rho)}} = \frac{1}{\sum_{n=0}^{1} \frac{0.8^n}{n!} + \frac{0.8^2}{2!(1-0.4)}} = \frac{1}{1.8 + \frac{0.64}{1.2}} \approx 0.4286$$

შემდეგ, თუ c=2, მაშინ ალბათობა, რომ ორივე გუნდი დაკავებულია, არის:

$$P(n \geq c) = \frac{(\lambda/\mu)^c}{c!(1-\rho)} P_0 = \frac{0.8^2}{2!(1-0.4)} 0.4286 \approx 0.2286 \approx 0.23$$

მაგალითი. სადაზღვევო კომპანია წინა მაგალითიდან განიხილავს მეორე გუნდის შექმნის შესაძლებლობას. მაგრამ, გადაწყვეტილის მიღების წინ, მათ სურთ გაანალიზონ ამ გადაწყვეტილების ეფექტი სხავადასხვა ოპერატიულ მახასიათებლებზე. წინა ამოცანისგან განსხვავებით, ახალ პროცესში იქნება ორი სერვერი (გუნდი) და ახალი მოდელი წარმოადგენს M/M/2 მოდელს.

წინა მაგალითიდან ვიცით, რომ λ=40 მოთხოვნა კვირაში და μ=50 მოთხოვნა კვირაში.

ზემოდ მოყვანილი ფორმულებიდან ვიღებთ:

$$\rho = \frac{\lambda}{c\mu} = \frac{40}{2*50} = 0.4$$

იმისათვის, რომ გავიგოთ რა არის იმის ალბათობა, რომ ახალმოსულ მოთხოვნას მოუწიოს ლოდინი, ჯერ უნდა გამოვთვალოთ ალბათობა იმ მოვლენისა, რომ სადაზღვევო კომპანიაში არც ერთი მოთხოვნა არ შემოსულა.

$$P_0 = \frac{1}{\sum_{n=0}^{c-1} \frac{(\lambda/\mu)^n}{n!} + \frac{(\lambda/\mu)^c}{c!(1-\rho)}} = \frac{1}{\sum_{n=0}^{1} \frac{0.8^n}{n!} + \frac{0.8^2}{2!(1-0.4)}} = \frac{1}{1.8 + \frac{0.64}{1.2}} \approx 0.4286$$

შემდეგ უნდა გამოვთვალოთ იმის ალბათობა *P(n≥c)=P(n≥2),* რომ კომპანიაში შემოსულია ორი ან მეტი მოთხოვნა (ამ შემთხვევაში ორივე გუნდი ახალი მოთხოვნის შემოსვლის დროს დაკავებულია):

$$P(n \geq c) = \frac{(\lambda/\mu)^c}{c!(1-\rho)} P_0 = \frac{0.8^2}{2!(1-0.4)} 0.4286 \approx 0.2286 \approx 0.23$$

რა არის იმის ალბათობა, რომ ახალ შემოსულ მოთხოვნას შეხვდება, როგორც მინიმუმი, ერთი თავისუფალი გუნდი?

$$P(n < 2) = P_0 + P_1 = P_0 + \frac{(\lambda/\eta)^1}{1!} P_0 = 0.4286 + \frac{0.8^1}{1!} 0.4286 = 0.4286 * (1 + 0.8) \approx 0.77$$

რამდენია საშუალო WIP, ანუ რამდენია მოთხოვნების მოსალოდნელი რაოდენობა:

$$L = \frac{(\lambda/\mu)^c \rho}{c!(1-\rho)^2} P_0 + \frac{\lambda}{\mu} = \frac{0.8^2 \cdot 0.4}{2!(1-0.4)^2} 0.4286 + 0.8 \approx 0.95$$ მოთხოვნა

რამდენია რიგში მყოფი მოთხოვნების მოსალოდნელი რაოდენობა:

$L_q = L - (\lambda/\mu) = 0.95 - 0.8 = 0.15$ მოთხოვნა

რამდენია საშუალო ციკლის დრო (ანუ რამდენ დროს ატარებს საშუალოდ მოთხოვნა სადაზღვევო კომპანიაში):

$W = L/\lambda = L_q/\lambda + 1/\mu = 0.15/40 + 1/50 = 0.02375$ კვირა= 0.11875 დღე=0.95 დღე=57 წუთი

(აქ მიღებულია, რომ კვირაში 5 სამუშაო დღეა და დღეში 8 საათია)

რამდენია რიგში საშუალო ლოდინის დრო :

$Wq=L_q/\lambda$ =0.15/40=0.00375 კვირა = 0.01875 დღე==0.15 დღე=9 წუთი

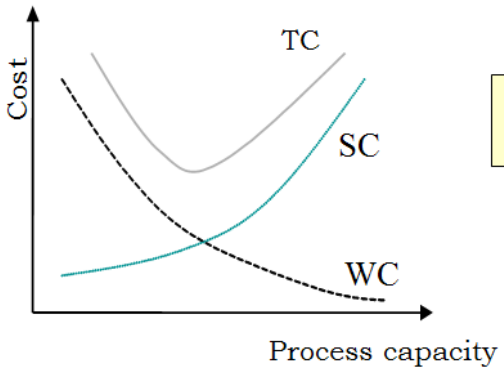
აქედან აშკარაა, რომ მოთხოვნისთვის (კლიენტისთვის) მეორე გუნდის შექმნა ხელსაყრელია. მაგრამ რამდენად ხელსაყრელია ეს სადაზღვევო კომპანიისთვის, ანუ რა ხარჯთანაა დაკავშირებული მეორე გუნდის ამუშავება?

ღირებულების (ხარჯების) ანალიზი.

განიხილება:

- WC = (რიგში) ლოდინის მოსალოდნელი ღირებულება (**shortage cost**) დროის ერთეულში (რამდენად არის შეფასებული კარგვა კლიენტის ლოდინის გამო)
- SC = მომსახურეობის მოსალოდნელი ღირებულება (ხარჯი) (**capacity cost**) დროის ერთეულში
- TC = ჯამური მოსალოდნელი ღირებულება დროის ერთეულში

ჩვენი მიზანია ჯამური ღირებულების მინიმიზირება (რიგში) (სურ.16)



Min TC = WC + SC

სურ.16

ლოდინის მოსალოდნელი ღირებულება არის სისტემაში მყოფი მომხმარებლების (სამუშაოების) რაოდენობის ფუნქცია (ანუ ლოდინის მოსალოდნელი ღირებულება განისაზღვრება სისტემაში მომხმარებლების (სამუშაოების) რაოდენობით)~

C_w= ლოდინის ღირებულება ერთ მომხმარებლისთვის დროის ერთეულში

C_wN = დროის ერთეულისთვის ლოდინის ღირებულება, როდესაც სისტემაში იმყოფება N მომხმარებელი (სამუშაო)

$$WC = C_w \sum_{n=0}^{\infty} nP_n = C_w L$$

(რიგში) ლოდინის მოსალოდნელი ღირებულება არის ფუნქცია რიგში მყოფი მომხმარებლების (სამუშაოების) რაოდენობისა

$$WC = C_w L_q$$

მომსახურეობის ღირებულების ანალიზი

დროის ერთეულში მომსახურეობის მოსალოდნელი ღირებულება, SC, დამოკიდებულია სერვერების რაოდენობაზე და მათ სიჩქარეზე (წარმადობაზე).

- c= სერვერების რაოდენობა
- μ= ერთი მომხმარებლის მომსახურეობის საშუალო ინტენსივობა (rate)
- $C_s(\mu)$ = ერთი სერვერისთვის დროის ერთეულში მოსალოდნელი ღირებულება როგორც μ-ს ფუნქცია

$$SC = c*C_S(\mu)$$

მიზანი: მთელი სისტემისთვის ჯამური ღირებულების მინიმიზირება დროის ერთეულისთვის:

$$\min_{\mu \in A, c=0,1,\ldots} TC = c \cdot C_s(\mu) + WC$$

მაგალითი. CopyCo არის მცირე ზომის ასლების გადაღების კომპანია. კომპანიას დაქირავებული აქვს ერთი Xerox-ის ტიპის მანქანა $50 დღეში. დანარჩენი ხარჯი (ხელფასები, დენის ღირებულება და ა.შ.) არის $150 დღეში. კომპანია განიხილავს მეორე მანქანის დაქირავების საშუალებას (იგივე ხარჯების პირობებით - $200 დღეში). წარსული გამოცდილებიდან ცნობილია, რომ CopyCo იღებს დღეში საშუალოდ 95 შეკვეთას, და კლიენტის მომსახურეობის საშუალო დრო არის 6 წუთი. CopyCo-ს სამუშაო დღე -10 საათი, ერთი მომხმარებლის რიგში ლოდინი ერთი საათის განმავლობაში შეფასებულია როგორც $5/საათში. **CopyCo** მფლობელს სურს ჯამური ხარჯების მინიმიზირება.

ოპტიმიზირების მოდელია:

$$\underset{c=\{1,2\}}{\text{Minimize}}\ TC = SC + WC = cC_s(\mu) + C_w L$$

ერთადერთი გადაწყვეტილების ცვლადი არის სერვერების რაოდენობა: c=1 ან c=2 ?

ამოცანის პირობიდან გამომდინარე გვაქვს:

λ = 95 შეკვეთა /დღე =95/10 = 9.5 შეკვეთა /საათი

μ = 60/6 = 10 შეკვეთა /საათი

$C_s(\mu)$=$200 დღეში= 200/10 = $20 საათში

C_w = $5 ერთი შეკვეთისთვის საათში.

M/M/1 სისტემა:

ρ =λ/μ =9.5/10 =0.95

L_1 =ρ/(1- ρ)=0/95/(1-0.95)=19 შეკვეთა

$TC_1 = SC_1 + WC_1$ = 1*20+5*19=$115 საათში.

M/M/2სისტემა:

ρ =λ/2μ =9.5/20 =0.475

$$P_0 = \frac{1}{\sum_{n=0}^{c-1} \frac{(\lambda/\mu)^n}{n!} + \frac{(\lambda/\mu)^c}{c!(1-\rho)}} = \quad = \frac{1}{1 + 0.95 + \frac{0.95^2}{2!(1-0.475)}} \approx 0.3559$$

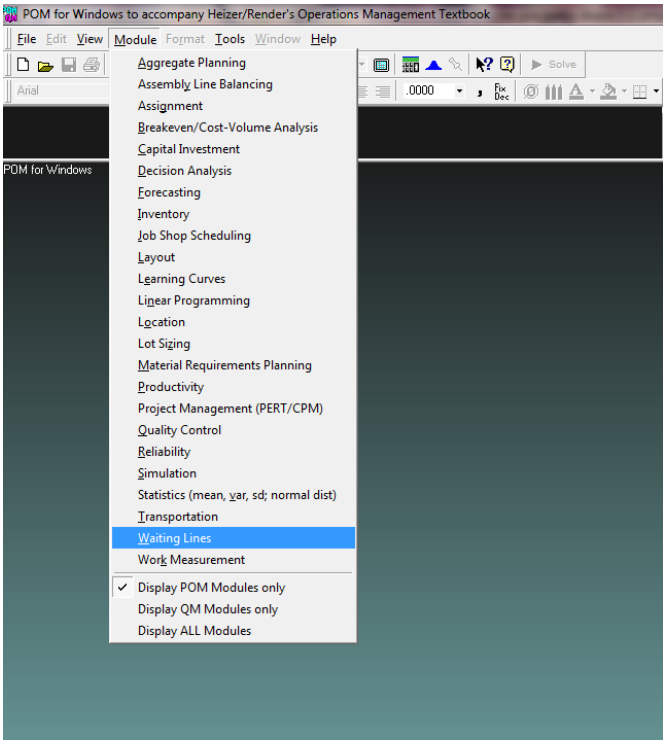
$$L_2 = \frac{(\lambda/\mu)^c \rho}{c!(1-\rho)^2} P_0 + \frac{\lambda}{\mu} = \frac{0.95^2 * 0.475}{2!(1-0.475)^2} 0.3559 + 0.95 \approx 1.23$$შეკვეთა

$TC_2=SC_2+WC_2$=2*20+5*1.23=$46.15 საათში.

რადგან $TC_2<TC_1$, გადაწყვეტილება ასეთია: დაქირავდეს მეორე Xerox-ი

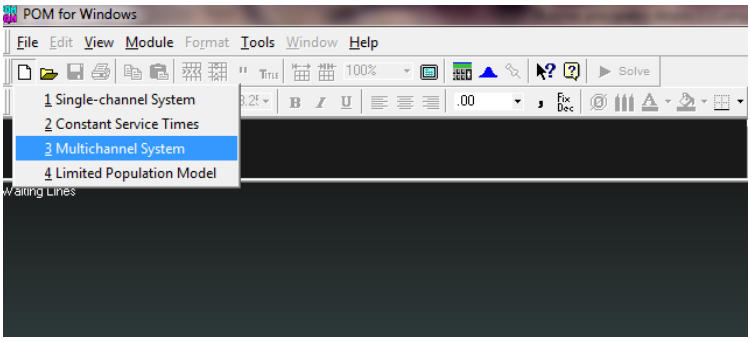
ამოცანის ამოხსნა POM-ის საშუალებით.

პროგრამა POM-ის გამოძახების შემდეგ ეკრანზე გამოჩნდება სურათი:(სურ.17)



სურ.17

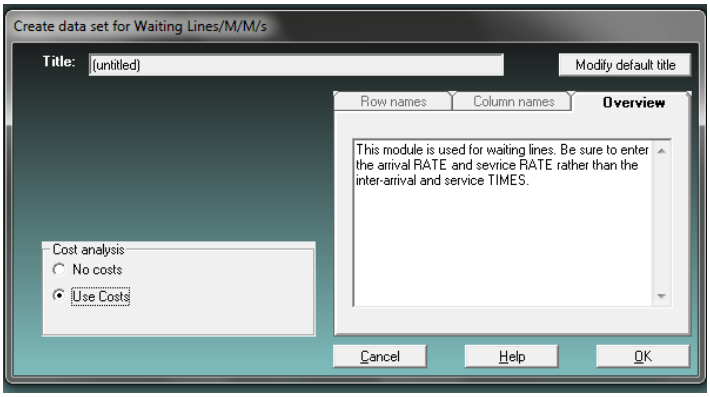
მენიუს პუნქტში Module უნდა გამოვიძახოთ Waiting Lines, შემდეგ გავხსნათ ახალი მოდელი, და შემდეგ ამოვირჩიოთ პუნქტი Multichannel System (მრავალსერვერიანი სისტემა):(სურ.18)



(სურ.18)

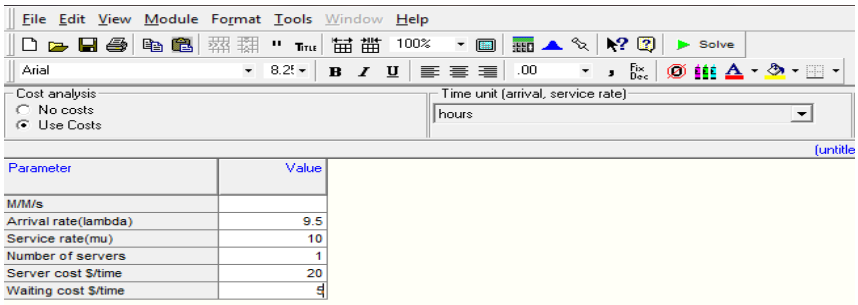
შემდეგ ფანჯარაში ამოვირჩიოთ რეჟიმი Use Cost (ღირებულებების გამოყენება):

(სურ.19)



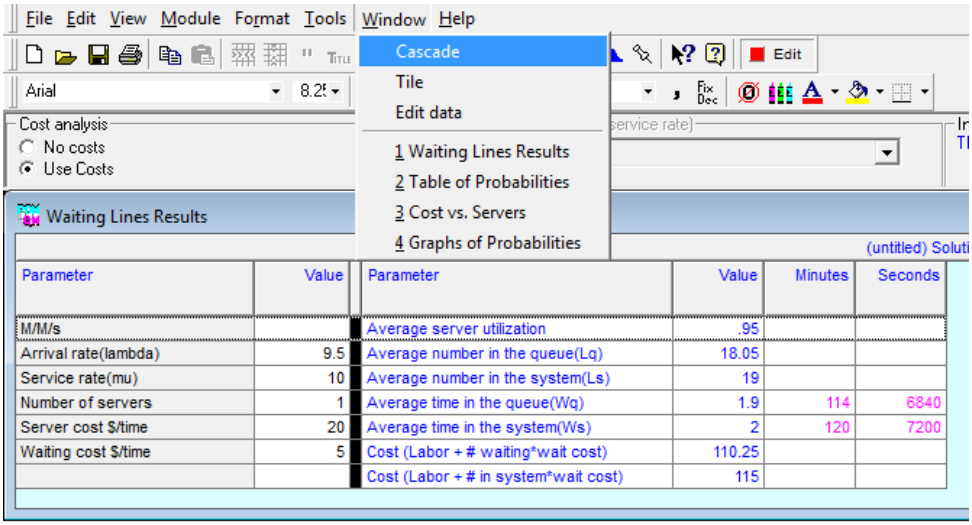
(სურ.19)

ამოცანის პირობების თანახმად შევიტანოთ შემდეგი მონაცემები (M/M/1 სისტემა, დროის ერთეული -საათი): (სურ.20)



(სურ.20)

დავაჭიროთ ღილაკს Solve და მივიღებთ შემდეგ პასუხს: (სურ.21)



(სურ.21)

მომხმარებლების საშუალო რაოდენობა სისტემაში

(Average number in the system (L_s)) =19;

მომხმარებლების საშუალო რაოდენობა რიგში

(Average number in the queue (W_q))=1.9;

მომხმარებლების რიგში ყოფნის საშუალო დრო

(Average time in the queue (W_q))=1.9 საათი =114 წუთი;

მომხმარებლების სისტემაში ყოფნის საშუალო დრო

(Average time in the system (W_s))=2 საათი =120 წუთი;

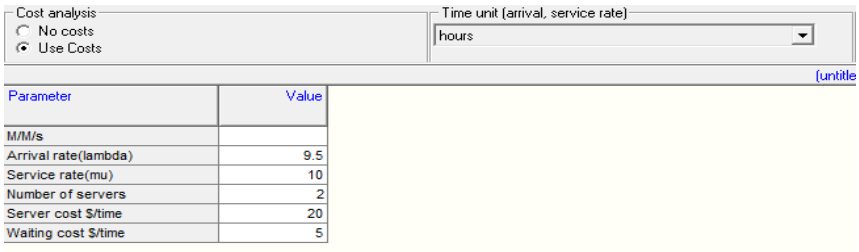
ჯამური ღირებულება

(Cost (Labor+# in system*wait cost))=$115

როგორც ვხედავთ, პასუხი ახლოსაა პასუხთან, რომელიც მიღებულია Excel-ის საშუალებით (M/M/1 სისტემისთვის,1 სერვერი).

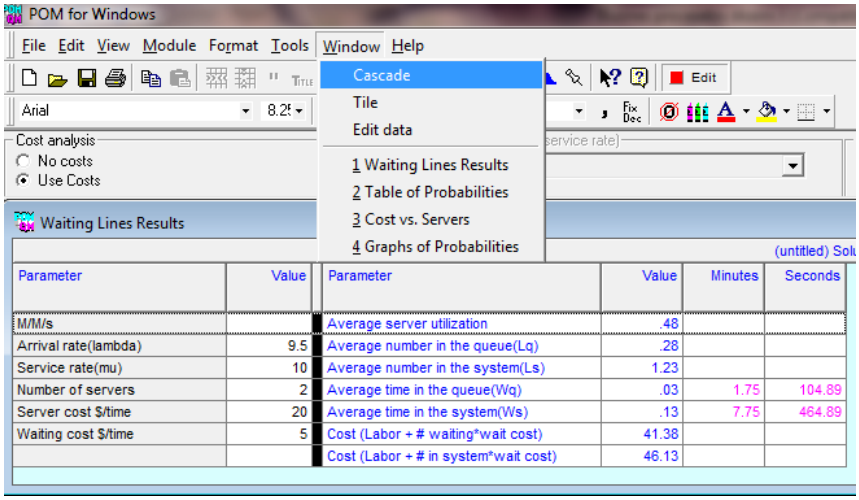
ახლა განვიხილოთ სისტემა M/M/2(ორი სერვერი):

საწყისი მონაცემების ეკრანი: (სურ.22)



(სურ.22)

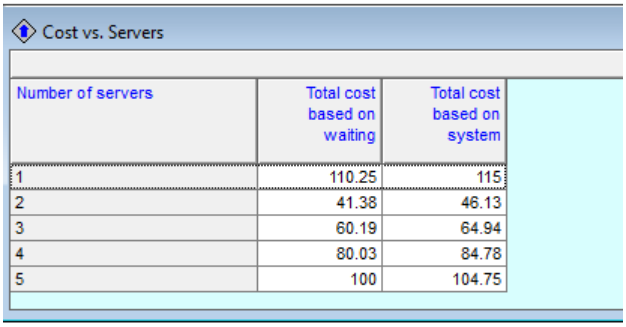
მიღებული პასუხი : (სურ.23)



სურ.23

აგრეთვე ეს პასუხი ახლოსაა პასუხთან, რომელიც მიღებულია Excel-ის საშუალებით (M/M/2(ორი სერვერისთვის)).

მენიუს პუნქტში Window ამოვირჩიოთ პუნქტი Costs vs. Servers, მივიღებთ:(სურ.24)



სურ.24

აქედან ჩანს, რომ სერვერების ოპტიმალური რაოდენობა არის 2 ($46.13 არის მინიმალური თანხა სხვებთან შედარებით)

უფრო სწრაფი სერვერის დაქირავების ვარიანტის განხილვა.

განიხილება Xerox-ის ახალი მოდელის დაქირავება, რომელიც ორჯერ უფრო სწრაფია (რაც ნიშნავს, რომ შეკვეთის შესრულების საშუალო დრო 3 წუთია (ძველ მოდელს ჰქონდა 6 წუთი). დაქირავების ფასი შეადგენს $150 დღეში. ახალი მოდელის ექსპლუატაცია მოითხოვს მეტ ხარჯებს პერსონალზე და ელექტროკვებაზე, რაც შეადგენს $250 დღეში. ამრიგად, ჯამური ხარჯია $400 დღეში, რაც უტოლდება ორი ძველი მოდელის ექსპლუატაციის ხარჯებს. საკითხავია, რომელი მოდელი უნდა იყოს დაქირავებული - ერთი ახალი თუ ორი ძველი?

უფრო სწრაფი ახალი მოდელისთვის გვაქვს შემდეგი M/M/1 რიგი:

$\rho = \lambda/\mu = 9.5/10 = 0.95$

C_w =$5 ერთ შეკვეთისთვის საათში

$\mu_{\text{ახალი}}=60/3=20$ შეკვეთა /საათი

$C_s(\mu_{\text{ახალი}})$ = $250 +$150 დღეში= 400/10 = $40 საათში

ჩვენი მიზანია: $\underset{c,\,\mu}{\text{Minimize}}$ $TC=SC+WC=cC_s(\mu) + C_wL$ (აქ მინიმიზირება ხდება ერთდროულად ორი პარამეტრით: c (სერვერების რაოდენობა) და μ (სერვერის სისწრაფე).

ანუ გადაწყვეტილების ცვლადებია: c=2 და μ=10 შეკვეთა/საათი (ძველი მოდელისთვის) ან c=1 და μ=20 შეკვეთა/საათი (ახალი მოდელისთვის). გარდა ამისა, ჯამური ხარჯი ძველი მოდელისთვის უკვე გვაქვს გამოთვლილი: TC_2 =$46.15 საათში.

ახალი მოდელისთვის გვაქვს შემდეგი M/M/1 სისტემა:

$\rho = \lambda/2\mu = 9.5/20 = 0.475$

$L_{\text{ახალი}} = \rho/(1-\rho)=0/475/(1-0.475)=0.9048$ შეკვეთა

$TC_{\text{ახალი}}=SC_{\text{ახალი}}+WC_{\text{ახალი}}=1*40+5*0.9048=\$44.52$ საათში.

ხარჯების შედარება გვიჩვენებს, რომ ახალი (უფრო სწრაფი მოდელის დაქირავება უფრო მომგებიანია, ვიდრე ორი ძველი მოდელის დაქირავება ($44.52<$46.14))

<u>ზოგადათ:</u> სწრაფი სერვერების ნაკლები რაოდენობა უფრო ჯობია, ვიდრე მეტი რაოდენობა ნელი, იგივე ჯამური წარმადობით სერვერების მეტი რაოდენობა.

ამოცანა. უნივერსიტეტი განიხილავს სუპერ-კომპიუტერის დაქირავების შესაძლებლობას. განიხილება ორი ვარიანტი:

- კომპიუტერი M, რომელიც უფრო ძვირია, სამაგიეროდ უფრო სწრაფია
- კომპიუტერი C, რომელიც უფრო იაფია, სამაგიეროდ უფრო ნელია

დამუშავების დრო და შემოსვლების შორის დრო (ორივე განაწილებულია ექსპონენციალური კანონით):

- $\lambda = 20$ სამუშაო/დღე
- $\mu_M = 30$ სამუშაო/დღე
- $\mu_C = 25$ სამუშაო/დღე

დაქირავების და რიგში ლოდინის ღირებულებები:

- დაქირავება: $C_M = \$500$ დღეში, $C_C = \$350$ დღეში
- ლოდინის ღირებულება: $50 დღეში ერთი სამუშაოსთვის

ბიზნეს პროცესების სიმულაცია პროგრამა ExtendLT-ს საშუალებით

მოდელი შედგება **ბლოკებისგან,** რომლებიც შეერთებულები არიან ერთმანეთთან. ბიბლიოთეკა (Library) არის ბლოკების საცავი.

ახალი მოდელის შექმნა იწყება ბიბლიოთეკების გახსნით. უნდა გაიხსნას ბიბლიოთეკები BPR და Discrete Event Simulation. ამისათვის ამოვირჩიეთ პუნქტი Open Library მენიუდან Library menu (ანუ Library>Open Library) და ამოირჩიეთ BPR.LRX ფოლდერიდან (საქაღალდიდან) Libraries. იგივე გააკეთეთ ბიბლიოთეკისთვის Discrete Event.LRX.

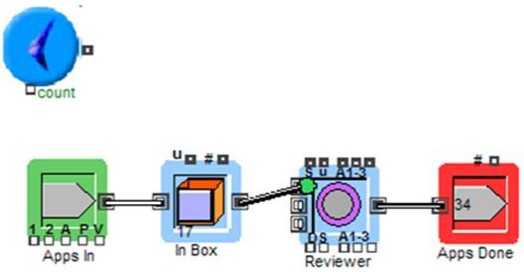
იმისათვის, რომ დავუმატოთ ბლოკი მოდელისთვის, უნდა:

- გავხსენათ მენიუ Library
- ამოირჩიოთ ის ბიბლიოთეკა , რომელიც მოიცავს სასურველ ბლოკს. გაიხსნება ბლოკების სახვადასხვა ტიპების იერარქიული მენიუ. როდესაც ტიპი შერჩეულია, ყველა ბლოკის სახელი გამოჩნდება მარჯვნივ ფანჯარაში.
- ამოირჩიოთ ტიპი თაგვის საშუალებით

Extend-ის ძირითადი ბლოკები (სურ.25)

სურ.25

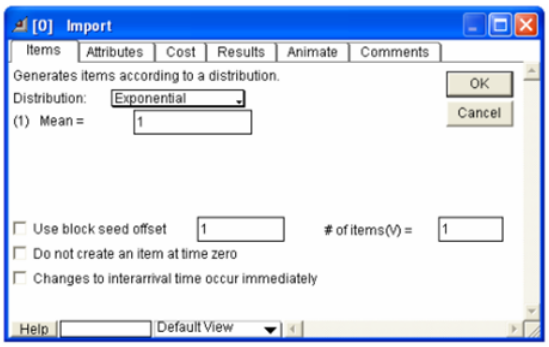
- ბლოკი **Executive** (ბიბლიოთეკიდან Discrete Event) არის სპეციალური ბლოკი, რომელიც უნდა იყოს ჩართული **ყველა** მოდელში! ეს ბლოკი უნდა იყოს მოთავსებული ყველა დანარჩენი ბლოკის მარცხნივ
- ბლოკი **Import** (**BPR**-ის ბიბლიოთეკის **Generators** ქვემენიუ, მოკლე ჩანაწერი: BPR>Generators) ქმნის (გენენირებს) მომხმარებლების (სამუშაოების) შემოდინების ნაკადს სისტემაში გარე სამყაროდან (იმპორტირება). ამ ბლოკში უნდა იყოს შეტანილი შემოსვლებს შორის საშუალო დრო და განაწილების კანონი (როგორ წესი, Exponential)
- ბლოკი **Repository** (BPR >Resources) წარმოადგენს რესურსების მარაგს (საცავს): მუშების, ნაწილების, მასალების და ა.შ.
- ბლოკი **Operation** (BPR>Activities) ასრულებს მომსახურეობას ან ოპერაციას. დამუშავების (მომსახურეობის) დრო (**processing time**) , რომელიც უნდა იყოს განსაზღვრული ამ ბლოკში, ყოველთვის არის მუდმივი. იმისათვის, რომ შევიტანოთ ცვლილება (შემთხვევითობა) დამუშავების (მომსახურეობის) დროში, ბლოკი **Input Random Number** (Generic>Inputs/Outputs) უნდა დაემატოს მოდელს.
- ბლოკი **Export** (BPR> Routing) გამოიყენება მომხმარებლების (სამუშაოების) სისტემიდან გამოსაყვანად (და დამუშავებული მომხმარებლების რაოდენობის დასათვლელად). რადგან ბიზნეს პროცესს შეიძლება ჰქონდეს რამდენიმე გამოსასვლელი, ამიტომ ერთზე მეტი ბლოკი **Export** შეიძლება იყოს ჩართული მოდელში.(სურ.26)



სურ.26

მაგალითი. გავიხსენოთ სადაზღვევო კომპანიის მაგალითი (გვერდი 2 დოკუმენტიდან 'Biznes procesebis dizaini 3'. ავაგოთ მოდელი პროცესის იმიტირებისათვის ერთი კვირის განმავლობაში (40 საათი). დავუშვათ, რომ საათი (**hours)** არის შერჩეული დროის ერთეულის სახით. თუ 40 მოთხოვნა კვირაში შემოდის კომპანიაში, მაშინ საშუალო დრო მოთხოვნების შემოსვლებს შორის არის 1 საათი. თუ კომპანიის მუშაკების გუნდი ამუშავებს 50 მოთხოვნას კვირაში, მაშინ თითოეული მოთხოვნის საშუალო დამუშავების დრო არის 0.8 საათი (ანუ 40/50). ავაგოთ Extend-ის სიმულაციური მოდელი.

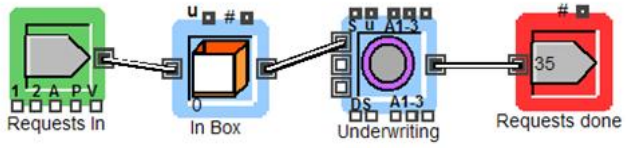
1. დავუმატოთ ბლოკი **Executive** (Discrete Event> Executive)
2. დავუმატოთ ბლოკი **Import** (BPR> Generators). ორჯერ დააწკაპუნეთ თაგვით ბლოკზე და გახსნილ ფანჯარაში შეცვალეთ განაწილება (Distribution) შეიტანეთ Exponential-ზე:



შეიტანეთ საშუალო დრო (**Mean**) 1 საათი.

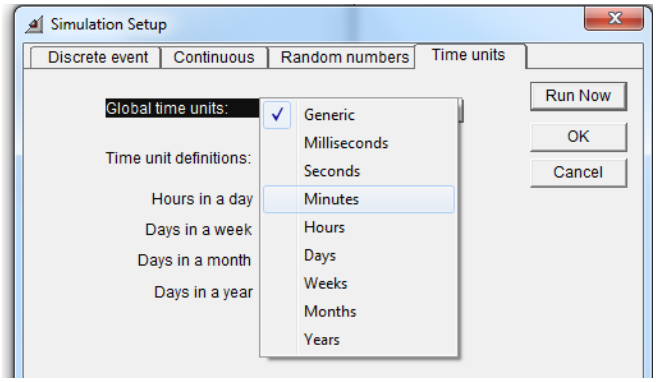
3. **Import** ბლოკის შემდეგ უნდა მოვათავსოთ ბლოკი Repository (ან ბლოკი **Stack**: BPR>Queus>Stack). ეს ბლოკი საჭიროა იმისათვის, რომ შემოსული მოთხოვნა არ დაიკარგოს იმ შემთხვევაში, თუ დამუშავების ბლოკი დაკავებულია წინა მოთხოვნის დამუშავებით.
4. დაუმატეთ ბლოკი Operation (BPR> Activities>Operation). ორჯერ დააწკაპუნეთ თაგვით ბლოკზე და შეიტანეთ დამუშავების დრო 0.8
5. დაუმატეთ ბლოკი Export

6. შეაერთეთ ბლოკები ერთმანეთთან. ამისათვის დააყენეთ თაგვის კურსორი ბლოკის კონექტორზე, დააჭირეთ თაგვის მარცხენა ღილაკს, და დაჭერილი თაგვის ღილაკით გადაათრიეთ თაგვი შემდეგი ბლოკის კონექტორამდე: სურ.27



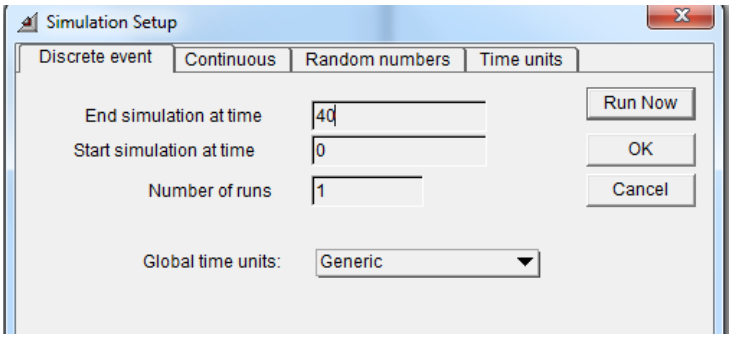
სურ.27

სიმულაციის დაწყებამდე უნდა ამოვირჩიოთ დროის ერთეული. ამოირჩიეთ Run>Simulation Setup, შემდეგ შეცვალეთ დრო საათებზე (hours): სურ. 28



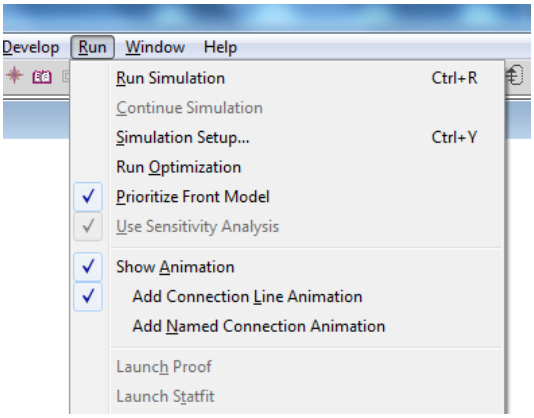
სურ.28

აგრეთვე შეცვალეთ სიმულაციის ხანგრძლივობის დრო 40 საათზე: სურ.29



სურ.29

აირჩიეთ აგრეთვე Show Animation და Add Connection Line Animation მენიუში Run: (სურ.30)



სურ.30

(ან დააწკაპუნეთ თაგვით შემდეგ სიმბოლოებზე (როგორც ნაჩვენებია სურათზე): (სურ.31)

მონაცემთა შეგროვება და სტატისტიკა

სურ.31

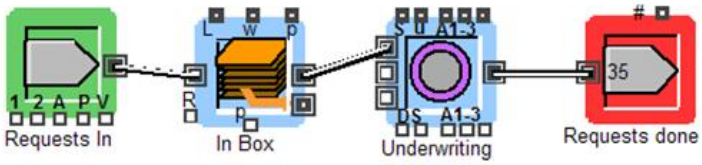
ბლოკები იმყოფებიან Discrete Event> Statistics -ში

ბლოკი **Costs Stats** აგროვებს ღირებულების (ხარჯების) მონაცემებს ყველა ბლოკიდან, სადაც ჩართულია ღირებულების გამოთვლის რეჟიმი (ღირებულება დროის ერთეულში

(cost per unit of time), ღირებულება ერთი მოთხოვნის დასრულების შემდეგ (cost per item), ჯამური ღირებულება (total cost) და ა.შ.)

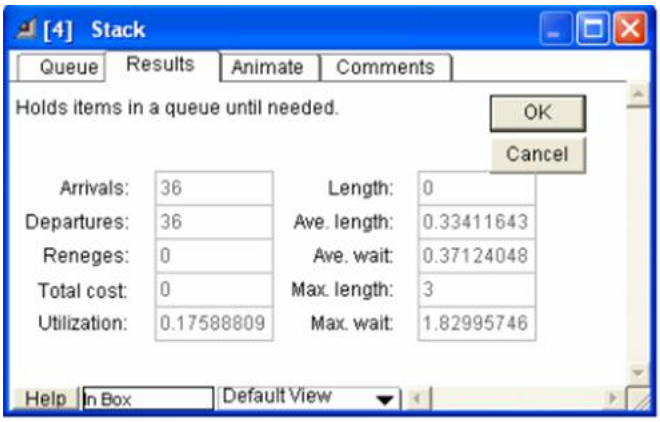
რიგების სტატისტიკა (სურ.32)

დავუშვათ, რომ ჩვენ შევქმენით შემდეგი მოდელი:



სურ32

(აქ წინა მოდელთან შედარებით ბლოკი Repository გამოცვლილია ბლოკით **Stack (**BPR> Queus>Stack), ეს ბლოკი უფრო მძლავრია , ვიდრე Repository)ამ მოდელის გაშვების შემდეგ, რიგის სტატისტიკა გამოიყურება (სხვა კომპიუტერებზე ეს ფანჯარა შეიძლება გამოიუყურებოდეს ოდნავსხვანაირად): სურ33

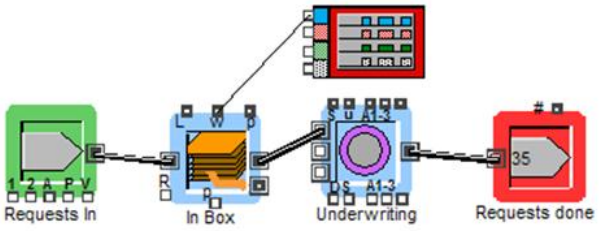


სურ33

თუმცა ეს სტატისტიკა, შეგროვებული სიმულაციის პერიოდის ბოლოს, მნიშვნელოვანია, ის არ გვიჩვენებს პროცესის დინამიკას სიმულაციის მთელ პერიოდის განმავლობაში. ამავე დროს საინტერესოა, როდის იყო სისტემა ყველაზე დატვირთული, როდის იყო რიგში ყველაზე მეტი მოთხოვნა და ა.შ. ამისათვის გამოიყენება ბლოკი Plotter (გახსენით ბიბლიოთეკა Plotter, შემდეგ ამოირჩიეთ Plotter, Discrete Event (Library>Plotter>Plotter, Discrete Event): (სურ34)

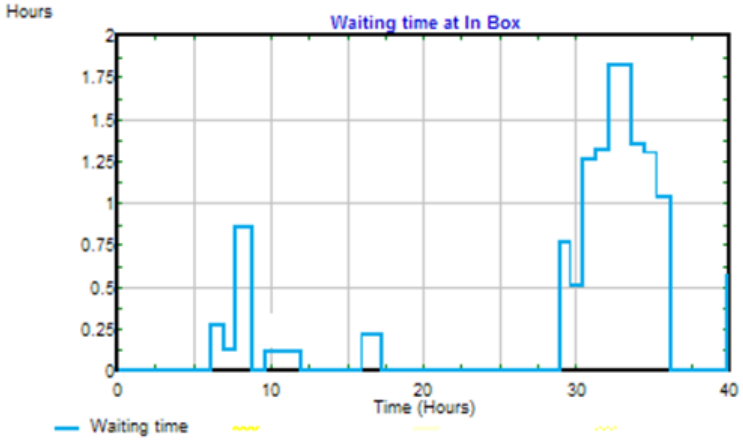
სურ34

შემდეგ შეაერთეთ ეს ბლოკი, როგორც ნაჩვენებია სურათზე:(სურ35)



სურ35

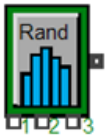
სიმულაციის გაშვების შემდეგ მივიღებთ: (სურ 36)



სურ 36

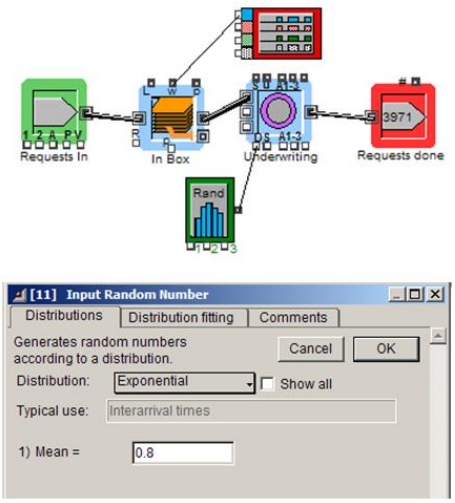
ამ გრაფიკიდან ჩანს, რომ ყველაზე დატვირთული დრო იყო სიმულაციის 32.5 საათის შემდეგ, როდესაც რიგში ლოდინის მაქსიმალური დრო იყო 1.83 საათი

ცვალებადობის (შემთხვევითობის) შეტანა დამუშავების ბლოკში.
ბლოკი **Input Random Number** (Generic> Inputs/Outputs): (სურ 37)



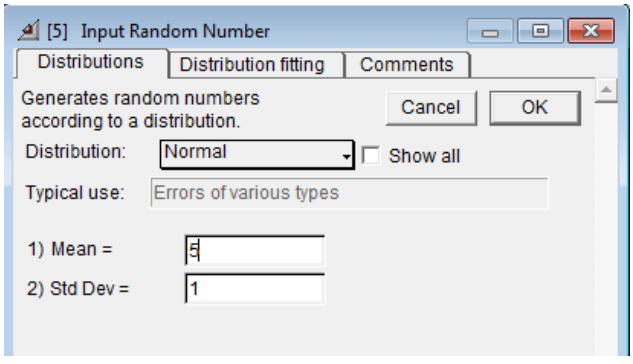
სურ 37

ეს ბლოკი გენერირებს შემთხვევით რიცხვებს, განაწილებულს სხვადასხვა კანონებით. შეცვალეთ მოდელი ისე, როგორც ნაჩვენებია სურათზე: (სურ 38)



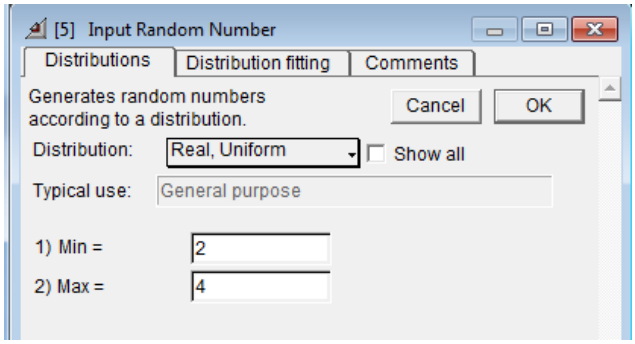
სურ 38

აქ დამუშავების დრო შეტანილია განაწილებული ექსპონენციალური კანონის სახით საშუალო დროით 0.8. მაგრამ დამუშავების პროცესებისთვის ეს არ არის დამახასიათებელი (რადგან მოთხოვნის დრო არის დამოკიდებული წინა მოთხოვნის დროზე, რაც არ არის ექსპონენციალური კანონის თვისება, სადაც მოთხოვნის შემოსვლის დრო არანაირად არ არის დომოკიდებული წინა მოთხოვნის შემოსვლის დროზე). ამიტომ დამუშავების დროისთვის გამოიყენება ალბათობის განაწილების ნორმალური, თანაბარი ან სხვა კანონი. მაგალითად, ნორმალური კანონი საშუალოთი (mean) 5 წუთი და სტანდარტული გადახრით (standard deviation) 1 წუთი ბლოკში Input Random Number გამოიყურება:(სურ 39)



(სურ 39)

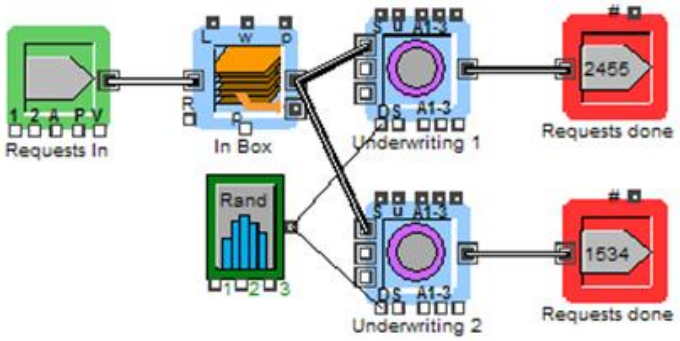
თანაბარი განაწილების კანონი მინიმალური დროით 2 წუთი და მაქსიმალური 4 წუთით (ანუ ნებისმიერი მნიშვნელობა ამ დიაპაზონში არის ერთნაირად მოსალოდნელი) ბლოკში Input Random Number გამოიყურება:(სურ 40)



(სურ 40)

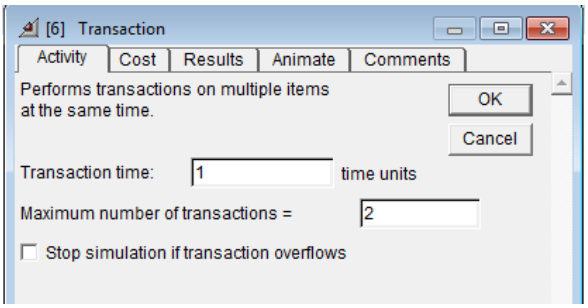
მეორე გუნდის დამატება

პირველი ვარიანტი:(სურ 41)



(სურ 41)

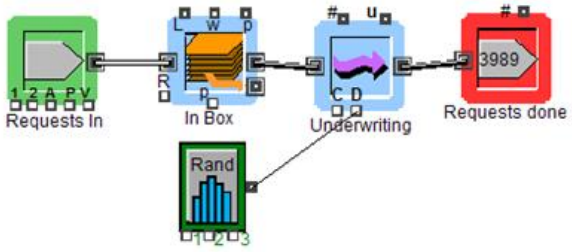
მეორე ვარიანტია ბლოკის **Transaction** (BPR>Activities) გამოყენება:(სურ 42)



(სურ 42)

აქ მაქსიმალური რაოდენობა ტრანზაქციების (maximum number of transactions) მიღებულია ორის ტოლი (მაქსიმალური რაოდენობა არის 1000 ტრანზაქცია). ანუ ორი გუნდი მუშაობს პარალელურად.

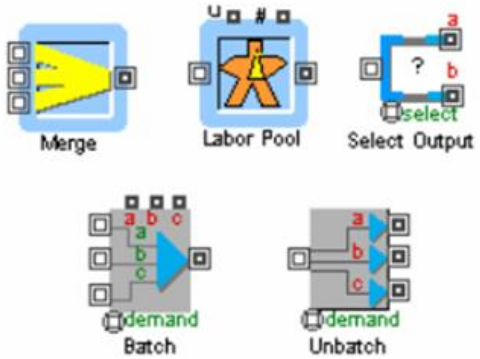
მოდიფიცირებული მოდელი:(სურ 43)



(სურ 43)

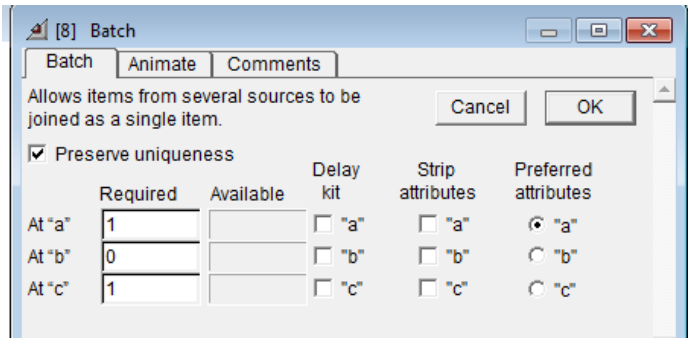
შრომითი რეზერვის - მარაგის (Labor Pool) დამატება

დავუშვათ, რომ სესხის გაცემის მოთხოვნა გადის პირველად განხილვას (review) (განხილვის დრო თანაბრად განაწილებულია დიაპაზონში 15-30 წუთი) და 20% შემთხვევაში მოთხოვნა უარყოფილია (მაგალითად, მომხმარებლის მიერ წარდგენილ დოკუმენტებს აკლია საჭირო ინფორმაცია). თუ მოთხოვნა არ არის უარყოფილი (80% შემთხვევაში), მაშინ ის გადადის შემდეგი დამუშავების ოპერაციაზე (სესხის გაფორმების დასრულება). თუ მოთხოვნა უარყოფილია, მაშინ ის გუნდი, რომელიც იყო დაკავებული ამ მოთხოვნის დამუშავებით, უნდა იყოს განთავისუფლებული შემდეგი მოთხოვნის დასამუშავებლად. იგივე ხდება, როდესაც გუნდი ასრულებს სესხის გაფორმებას. ამ მოდელისთვის ჩვენ დაგვჭირდება ახალი ბლოკები:(სურ 44)



(სურ 44)

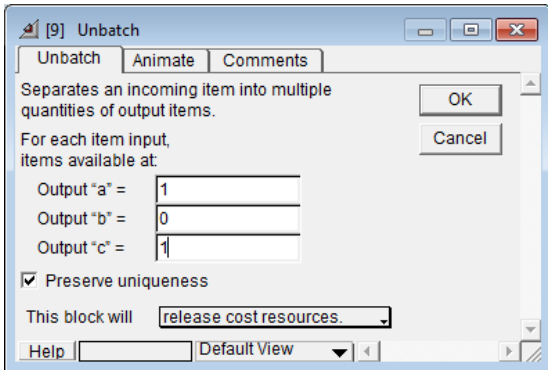
- ბლოკი **Labor Pool** (BPR>Resource) წარმოადგენს მუშაკებს (ან მუშაკების გუნდებს). საწყისი რაოდენობა ამ ბლოკში უნდა იყოს განსაზღვრული. ბლოკი Labor Pool აკონტროლებს მუშაკების გამოყენებას და განთავისუფლებას.
- ბლოკი **Batch** (Discrete Event>Batching) გამოიყენება იმისათვის, რომ გააერთიანოს 3-მდე შემოსასვლელი ერთ გამოსასვლელში. ამის შემდეგ ახალშექმნილი ერთობლიობა აგრძელებს მოგზაურობას სისტემაში. სესხების გაცემის მოთხოვნების მაგალითში მოთხოვნა შემოდის სისტემაში და რომელიმე გუნდი **Labor Pool**-დან უნდა იყოს მიმაგრებული ამ მოთხოვნაზე და ამის შემდეგ მოთხოვნა და გუნდი წარმოადგენს ერთობლიობას. აუცილებელია აგრეთვე მოინიშნოს შემდეგი პუნქტები: 'Preserve uniqness' (განმარტება იხილე შემდეგ პუნქტში **Unbatch**) და შეტანილი უნდა იქნას მოთხოვნების და გუნდების რაოდენობა შესაბამის შესასვლელებზე (ამ შემთხვევაში უნდა შეტანილ იქნას ერთიანი 1 შესასვლელზე At "a" და 1 შესასვლელზე At "c" : (სურ 45)



(სურ 45)

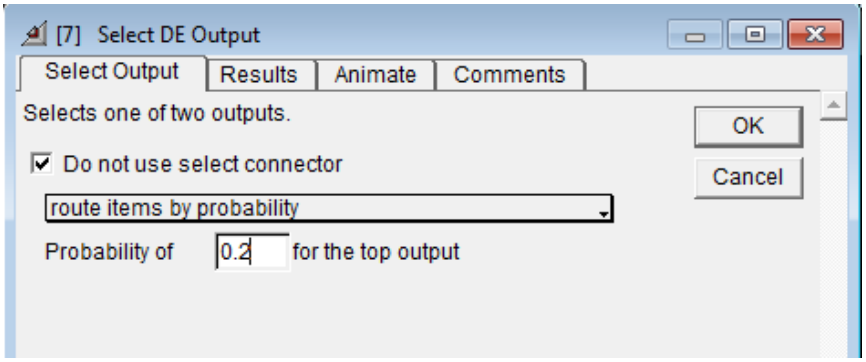
- ბლოკი **Unbatch** (Discrete Event>Batching) ასრულებს საპირისპირო ოპერაციას: მოსული გაერთიანებული ელემენტი იყოფა რამდენიმე ნაწილად. ჩვენ მაგალითში ბლოკ **Unbatch**-ში მოთხოვნა და გუნდი კვლავ განცალკევდებიან, მოთხოვნა გადის სისტემიდან და გუნდი ბრუნდება **Labor Pool**-ში. გარდა ამისა, აუცილებელია მოინიშნოს შემდეგი პუნქტები: 'Preserve uniqness' (იმისათვის, რომ გუნდებმა და მოთხოვნებმა განცალკევებულობის შემდეგ დაიბრუნონ თავისი საწყისი დანიშნულება, ანუ მოთხოვნა უნდა დარჩეს მოთხოვნად (და არ გადაიქცეს გუნდად) და პირიქით, გუნდი უნდა დარჩეს გუნდად და არ

გადაიქცეს მოთხოვნად) . აგრეთვე შეტანილ უნდა იქნას მოთხოვნების და გუნდების რაოდენობა შესაბამის შესასვლელებზე (ამ შემთხვევაში ერთიანი (1) გამოსასვლელზე Output “a” და ერთიანი (1) გამოსასვლელზე Output “c”) : (სურ.46)



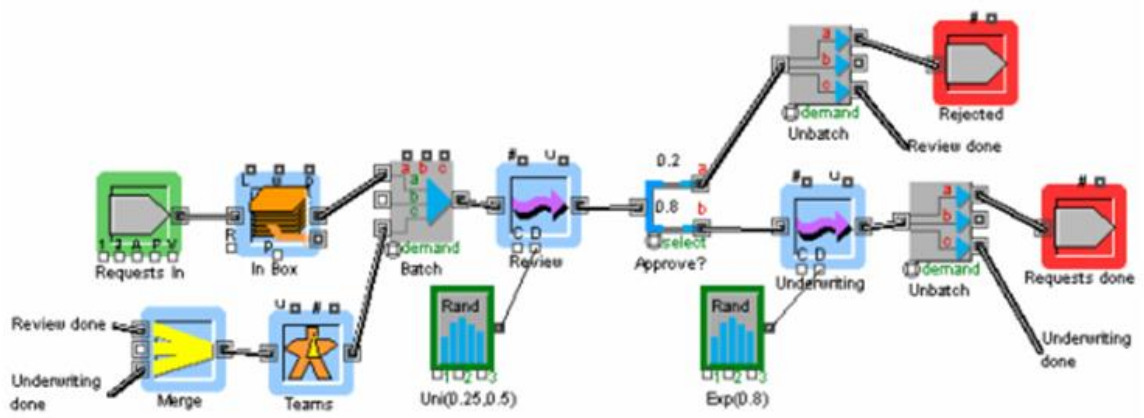
სურ.46

- ბლოკი **Merge** (BPR>Routing) იღებს ელემენტებს სხვადასხვა მიმართულებიდან და გზავნის მათ შემდეგ ეტაპზე.
- ბლოკი **Select DE Output** (Discrete Event> Routing) გზავნის ორ ელემენტს ორი სხვადასხვა მიმართულებით გარკვეული პირობების თანახმად. ამ ბლოკის გამოყენების ერთ-ერთი გზაა ელემენტების გაგზავნა ორი მიმართულებით მოცემული ალბათობის მიხედვით. ამისათვის უნდა მოინიშნოს პუნქტი ‘Do not use select connector’ და უნდა იყოს შეტანილი შესაბამისი ალბათობა:
  (სურ 47)



(სურ 47)

სრული მოდელი გამოიყურება: (სურ 48)



(სურ 48)

მოდელი შეიცავს ორ ‘დასახელებულ შეერთებას’ (named connections): Review Done (პირველადი განხილვა დასრულებულია მოთხოვნის უარყოფით)) და Underwriting Done (სესხის გაფორმება დასრულებულია). ‘დასახელებული შეერთებისთვის’ აუცილებელია შემდეგი: დააწკაპუნეთ ორჯერ თაგვის მარცხენა ღილაკს იქ, სადაც საჭიროა ‘დასახელებულ შეერთების“ შექმნა. გაიხსნება ტექსტური ველი. ჩაბეჭდეთ ველში სასურველი ტექსტი და დააწკაპუნეთ სადმე ტექსტური ველის გარეთ (ან დააჭირეთ კლავიატურის ღილაკს Enter). ახლა ტექსტური ველი შეიძლება შეერთებულ იქნას ნებისმიერი ბლოკის კონექტორთან. აიღეთ ასლი შექმნილი ტექსტის და ჩასვით მეორე ტექსტურ ველში იმ ბლოკთან, რომელთანაც საჭიროა ‘დასახელებული შეერთების“ შექმნა. ჩვენ მაგალითში გვაქვს ორი ‘დასახელებული შეერთება’: „Underwriting Done“

ტექსტურ ველში Unbatch-ის მეორე (ქვედა) ბლოკთან და ბლოკ Merge-თან და „Review Done“ ტექსტურ ველში Unbatch-ის პირველ (ზედა) ბლოკთან და ბლოკ Merge-თან.

ბიზნეს პროცესების წარმადობის ოპტიმიზაცია.

სიმულაციის მოდელებში ზოგიერთი ცვლადი შეიძლება იყოს კონტროლირებადი (მაგალითად, მუშების რაოდენობა საწარმოში ან კლიენტების პრიორიტეტების წესები სადაზღვევო კომპანიში). ასეთ ცვლადებს უწოდებენ **გადაწყვეტილების ცვლადებს (Decision variables).** ასეთი ცვლადების ოპტიმალური მნიშვნელობების (საუკეთესო მნიშვნელობების მომგებიანობის თვალსაზრისით) პოვნა წარმოადგენს მნიშვნელოვან ამოცანას სიმულაციის და გადაწყვეტილებათა მიღების თეორიაში.

ასეთი ოპტიმალური მნიშვნელობების პოვნა თხოულობს ძებნას განმეორებად (იტერატიულ) სტილში. ეს გულისხმობს მოდელის სიმულაციის გაშვებას გადაწყვეტილების ცვლადების მნიშვნელობების საწყისი სიმრავლისათვის, შედეგების ანალიზს, ერთი ან მეტი ცვლადის მნიშვნელობის შეცვლას, სიმულაციის ხელმეორედ გაშვებას, და ამ პროცესის გამეორებას იქმდე, ვიდრე მისაღები გადაწყვეტილება არ იქნება მიღებული. ეს პროცესი ძალზე მოსაწყენია და ხანგრძლივია და ამავე დროს არ არის ნათელი, რა წესით უნდა შეიცვალოს მნიშვნელობები ერთი სიმულაციიდან მეორემდე.

უფრო ფორმალური მიდგომაა ყველა შესაძლო ვარიანტის (ალტერნატივის) დანომვრა (მაგალითად, გადაწყვეტილების ცვლადების ყველა შესაძლო მნიშვნელობების დანომვრა). თუმცა ეს მიდგომა გარანტიას გვაძლევს, რომ ოპტიმალური გადაწყვეტილება იქნება მოძებნილი (მაგალითად, გადაწყვეტილების ცვლადების მნიშვნელობების ის კომბინაცია, რომელიც უზრუნველყოფს მაქსიმალურ მოგებას), მას აქვს შეზღუდული გამოყენება. წარმოიდგინეთ, რომ სიმულაციის მოდელში გამოყენებულია მხოლოდ ორი გადაწყვეტილების ცვლადი. თუ თითოეულ ცვლადს აქვს 10 შესაძლო მნიშვნელობა, ყველა კომბინაციის შემოწმება (მოდელის შესაბამისი მნიშვნელობებით გაშვება) მოითხოვს 100 (10^2 ალტერნატივა) სიმულაციის

გაშვებას. თუ თითოეულ გაშვებას სჭირდება, ვთქვათ, კომპიუტერული დროის 2 წამი, მაშინ მთლიანად სიმულაციის პროცესი დასრულდება დაახლოებით 3 წუთში.
მაგრამ , თუ 2 ცვლადის ნაცვლად გვაქვს 6 ცვლადი, მაშინ ყველა კომბინაციის შემოწმება (ექსპერიმენტის ჩატარება) მოითხოვს სიმულაციის 1 მილიონ (10^6) გაშვებას, ან დაახლოებით კომპიუტერული დროის 23 დღეს. ცხადია, რომ ეს არარეალისტური მოთხოვნაა.

იმისათვის, რომ ეს პრობლემა გადაილახოს, Extend-ს გააჩნია სპეციალური ბლოკი - **Evolutionary Optimizer (ევოლუციონალური ოპტიმაიზერი).** ოპტიმიზაციის პრობლემა შეიძლება აღწერილი იყოს ოპტიმაიზერის **ინტერფეისში,** და შემდეგ ეს ბლოკი ეძებს ოპტიმალურ გადაწყვეტილებას წინასწარ განსაზღვრული **მიზნის (objective)** გათვალისწინებით. თითქმის ყოველთვის ოპტიმაიზერი პოულობს ოპტიმალურ (საუკეთესო) ან სუბ-ოპტიმალურ (ოპტიმალურთან ახლოს მყოფ) გადაწყვეტილებას.
სახელწოდება 'ევოლუციონალური ოპტიმაიზერი' წარმოიშვება ხელოვნური სისტემების მიმართ **ბუნებრივი ევოლუციის** ბიოლოგიური პრინციპების გამოყენების შედეგად.
როგორც ზემოთ იყო ნათქვამი, ოპტიმიზაციის მოდელი შედგება 3 ძირითადი ელემენტისაგან:

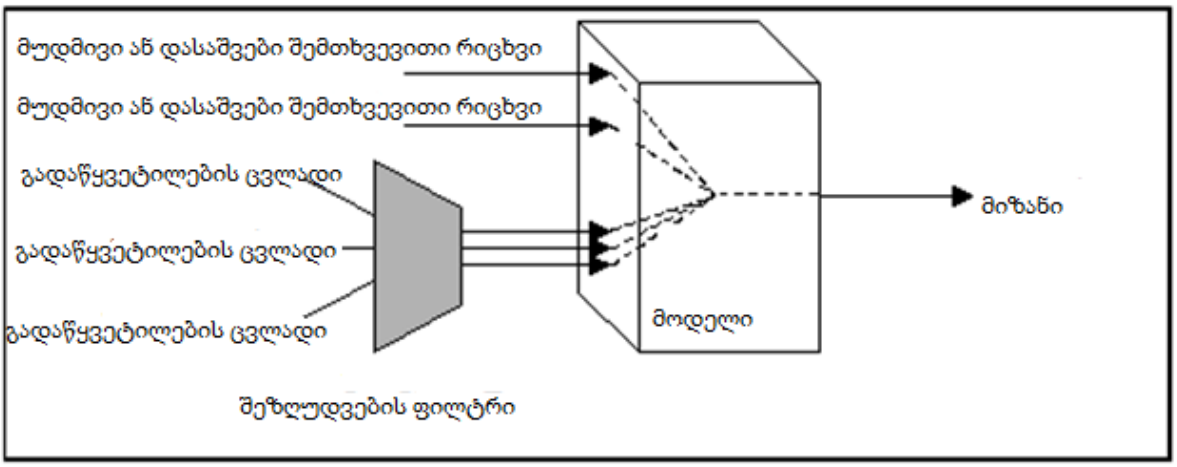
- *გადაწყვეტილებათა ცვლადები (Decision variables)*
- *შეზღუდვები (Constraints):* მაგალითად, თუ ხელფასების ჯამური ბიუჯეტი პროცესისთვის, რომელშიც გამოყენებულია ორი რესურსი (აღნიშნული, როგორც R1 და R2), არის $500000, მაშინ შემდეგი შეზღუდვა შეიძლება იყოს ჩამოყალიბებული:

$$20000 * R1 + 10000 * R2 \leq 500000$$

აქ იგულისხმება, რომ თითოეულ R1 რესურსის ღირებულება არის $20000 წელიწადში და თითოეულ R2 რესურსის ღირებულება არის $10000 წელიწადში. აგრეთვე იგულისხმება, რომ რესურსები R1 და R2 წარმოადგენენ საწყისს მნიშვნელობებს ბლოკიდან Labor Pool და ისინი იყვნენ *კლონირებული* (განმარტება იხილე შემდეგ) ბლოკ **Evolutionary Optimizer** დასამატებლად.

- *მიზანი* (*Objective)*: ოპტიმიზაციის მოდელის მიზნის მათემატიკური წარმოდგენა, მაგალითად, მოგების (შემოსავლის) მაქსიმიზაცია ან ხარჯების მინიმიზაცია გადაწყვეტილებათა ცვლადების შესაბამისი შეზღუდვების გათვალისწინებით

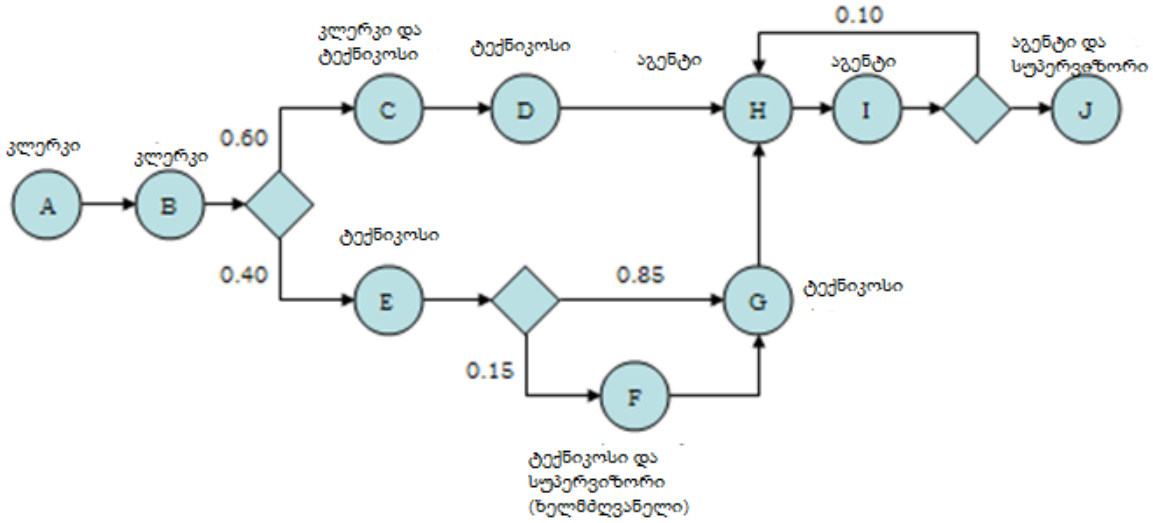
ოპტომაიზერის კონცეპტუალური მოდელი: (სურ 49)



სურ.49

პროცესების ოპტიმიზაცია.

განვიხილოთ შემდეგი ბიზნეს პროცესი: (სურ 50)



სურ 50

პროცესი შედგება 10 აქტივობისგან (სამუშაოებისგან). 4 ტიპის რესურსი ასრულებს ამ სამუშაოებს: კლერკები (საოფისო მუშაკები), აგენტები, ტექნიკოსები და სუპერვიზორები (ხელმძღვანელები). ზოგიერთი სამუშაო მოითხოვს ორ რესურს (მაგალითად, სამუშაო F სრულდება ორი ტიპის რესურსის მიერ: ტექნიკოსით და სუპერვიზორით).

პროცესის მენეჯერს სურს ციკლის დროის მინიმიზაცია (ანუ გახადოს იგი რაც შეიძლება ხანმოკლე). ეს შესაძლებელია სამუშაოებთან მიმაგრებული რესურსების რაოდენობის გაზრდით. მაგრამ ამავე დროს მენეჯერს არ სურს რესურსების დაბალი დატვირთვა. მაშასადამე, პროსესის მენეჯერს სურს დააწესოს შეზღუდვები როგორც თითოეული ტიპის რესურსის რაოდენობაზე, აგრეთვე რესურსების ჯამურ რაოდენობაზე (ეს აგრეთვე გამოწვეულია ხელფასების ფონდის ეკონომიის აუცილებლობით). ოპტიმიზაციის ამოცანა შეიძლება იყოს ჩამოყალიბებული შემდეგნაირად:

მინიმიზაცია : ციკლის საშუალო დრო

შეზღუდვები: კლერკები + აგენტები + ტექნიკოსები + სუპერვიზორები ≤ 12

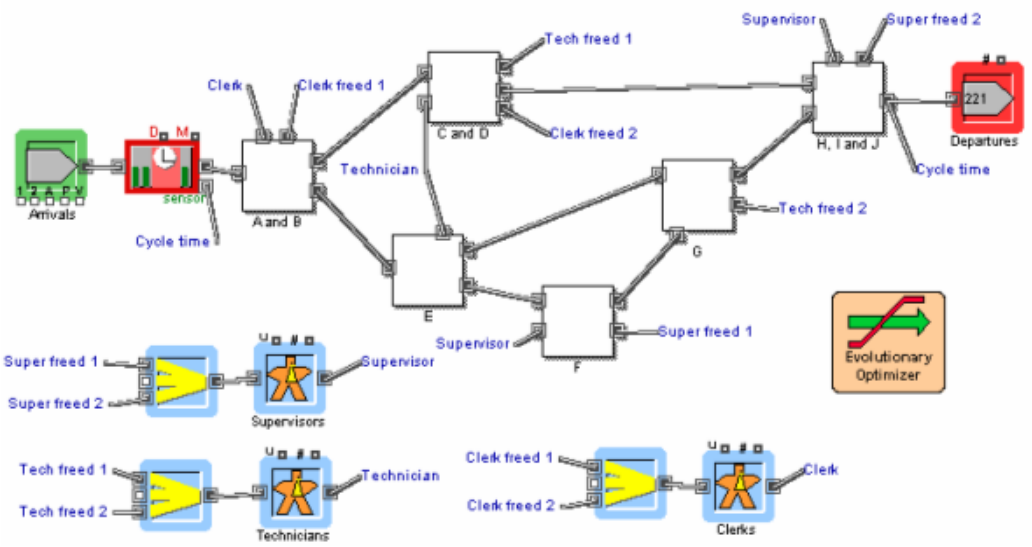
$$1 \le \text{კლერკები} \le 10$$

$$1 \le \text{ტექნიკოსები} \le 10$$

$$2 \le \text{სუპერვიზორები} \le 5$$

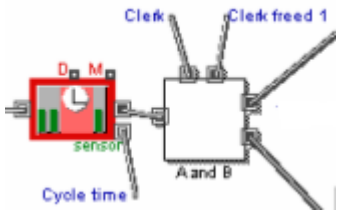
$$1 \le \text{აგენტები} \le 10$$

პროცესის Extend-ის მოდელი გამოიყურება ასე:(სურ 51)



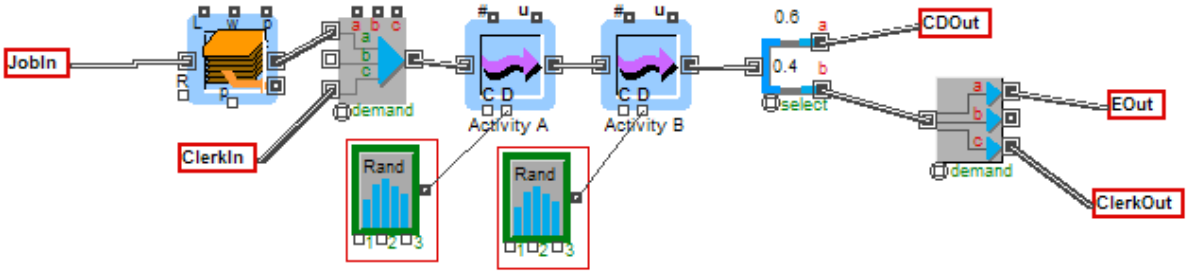
(სურ 51)

აქ თეთრი ფერის ბლოკები წარმოადგენენ „იერარქიულ: ბლოკებს, ანუ ბლოკებს, რომლების შეიცავენ რამდენიმე აქტივობას (სამუშაოებს). მაგალითად, ბლოკი : (სურ 52)



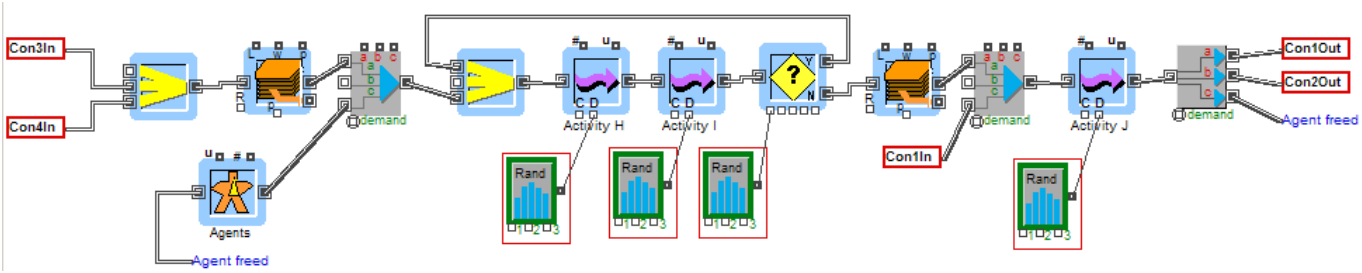
(სურ 52)

შეიცავს შიგნით შემდეგ მარტივ (არა-იერარქიულ) ბლოკებს: (სურ 53)



(სურ 53)

მოდელი იყენებს 4 რესურსის პულს(მარაგს): Clerks, Technicians, Supervisors, Agents. სურათზე ნაჩვენებია მხოლოდ 3 რესურსი, მეოთხე (აგენტების) პული მოთავსებულია იერარქიულ ბლოკში H, I, and J: (სურ 54)



(სურ 54)

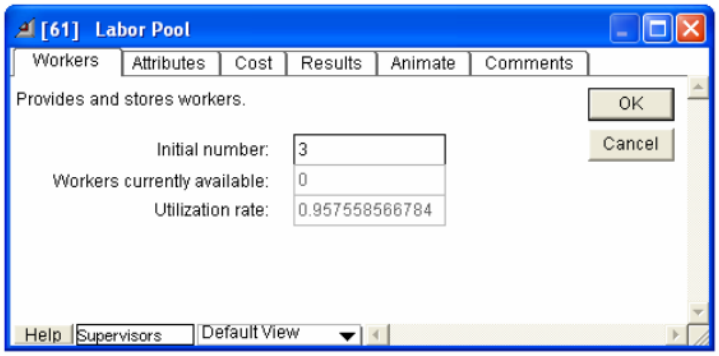
დავუშვათ, რომ მოდელი უკვე აგებულია და საჭიროა მოდელის ოპტიმიზაცია. პირველი ბიჯია ბლოკის Evolutionary Optimizer დამატება. ეს ბლოკი განლაგებულია ბიბლიოთეკის Generic-ის Optimization ქვემენიუში.

ოპტიმიზაციის მოდელი შეიცავს 4 გადაწყვეტილების ცვლადს (decision variables), მიზნობრივ ფუნქციას (objective function) და ერთ შეზღუდვას (constraint). შემდეგი ბიჯია გადაწყვეტილების ცვლადების ჩართვა ბლოკში Evolutionary Optimizer. ეს ხდება clone (ასლის შექმნა) layer ინსტრუმენტის გამოყენებით:(სურ 54)

(სურ 54)

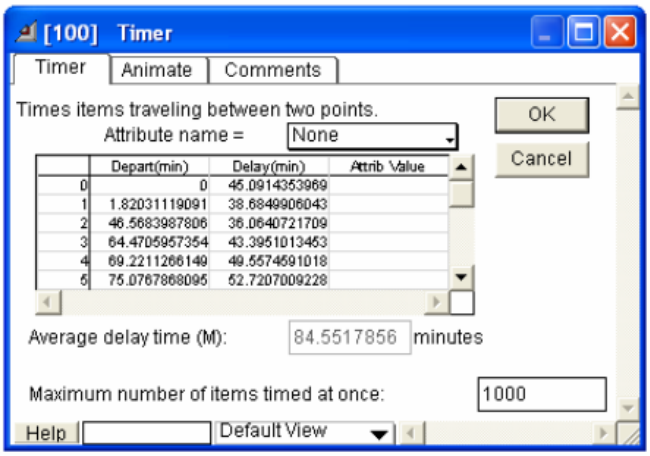
ინსტრუმენტი Clone layer ქმნის მოდელის რომელიმე ბლოკის შერჩეული ელემენტის ასლს და ათავსებს მას მოდელის სხვა ბლოკში. მაგალითად, ჩვენ შეგვიძლია შევქმნათ რესურსის პულის გამოყენების დონის (utilization rate) ასლი და მოვათავსოთ იგი სადმე მოდელის სხვა ბლოკში ან მოვათავსოთ უბრალოდ რესურსის პულის გვერდით. ახლა მოდელის გაშვების დროს ჩვენ შეგვიძლია დავინახოთ რესურსის პულის გამოყენების დონის ცვლილებები რესურსის პულის ბლოკის გახსნის გარეშე.

ინსტრუმენტი Clone layer გამოიყენება თითოეული რესურსის საწყისი მნიშვნელობის ასლის შესაქმნელად. მაგალითად, სუპერვიზორის დიალოგური ბლოკი:(სურ 55)



(სურ 55)

იმისათვის, რომ შევქმნათ ხელმისაწვდომი სუპერვიზორების საწყისი რაოდენობის ასლი (clone) და ჩავრთოდ იგი ბლოკში Evolutionary Optimizer, უნდა გავხსნათ შესაბამისი ბლოკის დიალოგური ფანჯარა. მაგალითად, გახსენით სუპერვიზორების პულის ფანჯარა, ამოირჩიეთ ფურცელი Workers, შემდეგ მენიუს ზოლიდან ამოირჩიეთ ინსტრუმენტი Clone Layer, მონიშნეთ ინსტრუმენტი Clone Layer-ის საშუალებით სუპერვიზორების საწყისი რაოდენობის ველი (სურათზე იგი 3-ის ტოლია), და გადაათრიეთ იგი თაგვით **დახურულ** ბლოკზე Evolutionary Optimizer. შეასრულეთ მსგავსი კლონირება 4 ჯერ კლერკების, აგენტების, ტექნიკოსების და სუპერვიზორების საწყისი მნიშვნელობებისთვის. შემდეგ, გახსენით ბლოკი Timer: (სურ 56)



(სურ 56)

და შემდეგ კვლავ ინსტრუმენტის Clone Layer საშუალებით მოახდინეთ ველის Average delay time (აქ ის 84.55 წუთის ტოლია) კლონირება და ჩართეთ იგი ბლოკში Evolutionary Optimizer.

შემდეგ, დააწკაპუნეთ ორჯერ ბლოკზე Evolutionary Optimizer. შეიტანეთ რაიმე თქვენთვის გასაგები სახელი თითოეულ რესურსისთვის (ავტომატურად შექმნილი სახელების Var1, Var2, და ა.შ. ნაცვლად). თქვენს მიერ შექმნილ სახელებში დაუშვებელია ხარვეზების (ცარიელი პოზიციის), სპეციალური სიმბოლოების გამოყენება, მაგრამ დასაშვებია ქვედა ტირეს („ _“) გამოყენება.

ახლა შეიტანეთ გადაწყვეტილების ცვლადების **საზღვრები** და ჩამოაყალიბეთ მიზნობრივი ფუნქცია. საზღვრები უნდა შეიტანოთ ფურცლის Set Cost-ის სვეტებში Min Limit და Max Limit :

მიზნობრივი ფუნქცია გამოიყურება როგორც:

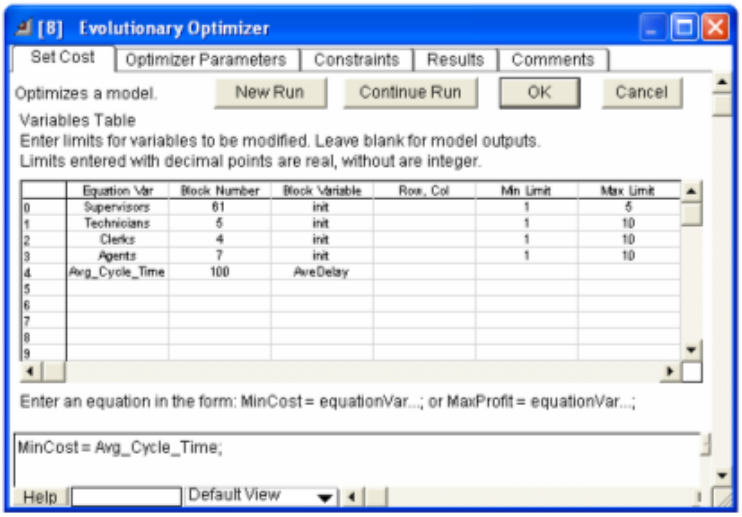
MinCost=Avg_Cycle_Time

მიზნობრივი ფუნქციის სხვა ფორმაა:

MaxProfit = equationVar….. (თუ საჭიროა მიზნობრივი ფუნქციის

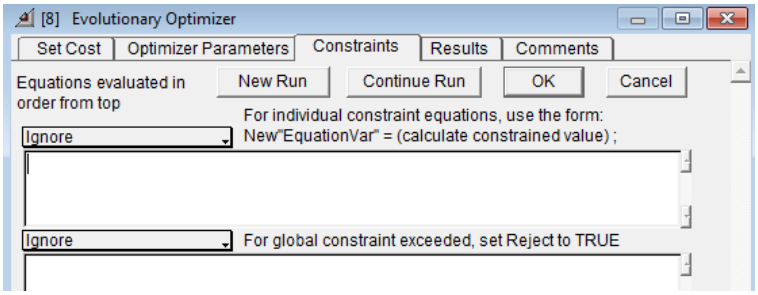
მაქსიმიზაცია)

დასრულებული დიალოგი გამოიყურება როგორც:(სურ 57)



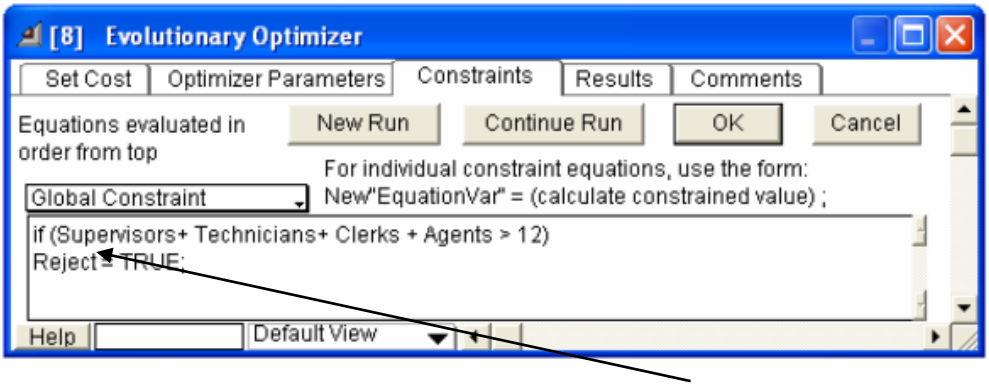
(სურ 57)

იმისათვის, რომ შევიტანოთ **გლობალური** შეზღუდვა (ე.ი. შეზღუდვა, რომელშიც შედის ერთზე მეტი ცვლადი), გადადით ფურცელზე Constraints ბლოკის Evolutionary Optimizer-ის დიალოგურ ფანჯარაში. შეზღუდვების ფანჯარაში:(სურ 58)



სურ 58

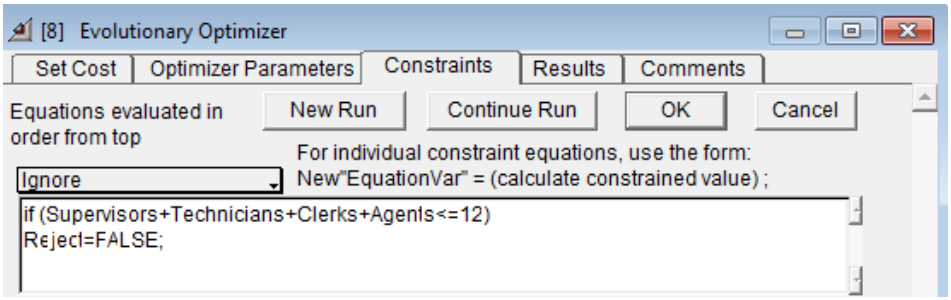
გახსენით ველი „Ignore” და ამოირჩიეთ პუნქტი „Global Constrains”:(სურ 59)



(სურ 59)

შეიტანეთ ველში გლობალური შეზღუდვის გამოსახულება (იხილე ზედა სურათი).

აქ ტერმინი Reject ნიშნავს, რომ ბლოკმა Evolutionary Optimizer უნდა მიიღოს გადაწყვეტილებების ცვლადების ის კომბინაცია, რომელშიც რესურსების ჯამური რაოდენობა 12-ზე მეტია (Reject=TRUE). ამავე მოთხოვნის მეორე ვარიანტია: სურ.60

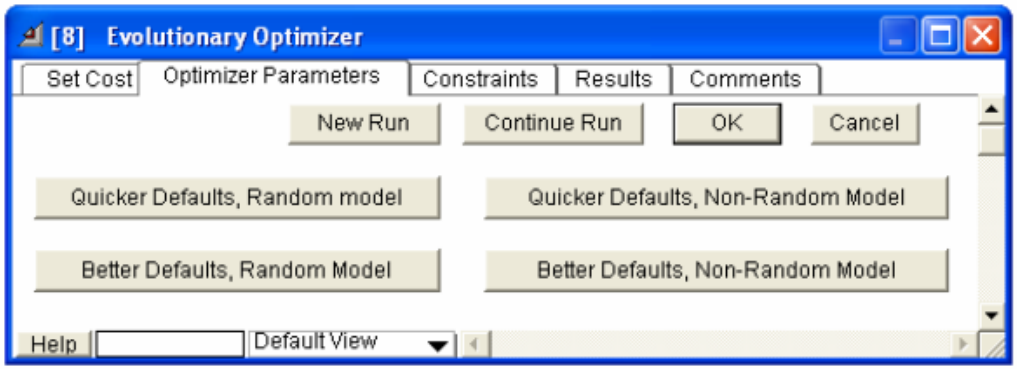


სურ 60

აქ გამოსახულება Reject=FALSE ნიშნავს, რომ ბლოკმა Evolutionary Optimizer უნდა მიიღოს გადაწყვეტილებების ცვლადების ის კომბინაცია, რომელშიც რესურსების ჯამური რაოდენობა არ აღემატება 12-ს.

ახლა ოპტიმიზაციის მოდელი დასრულებულია და შეგვიძლია დავიწყოთ გადაწყვეტილებების ცვლადების ოპტიმალური მნიშვნელობების ძებნა. არსებობს ძებნის 4 პარამეტრი:

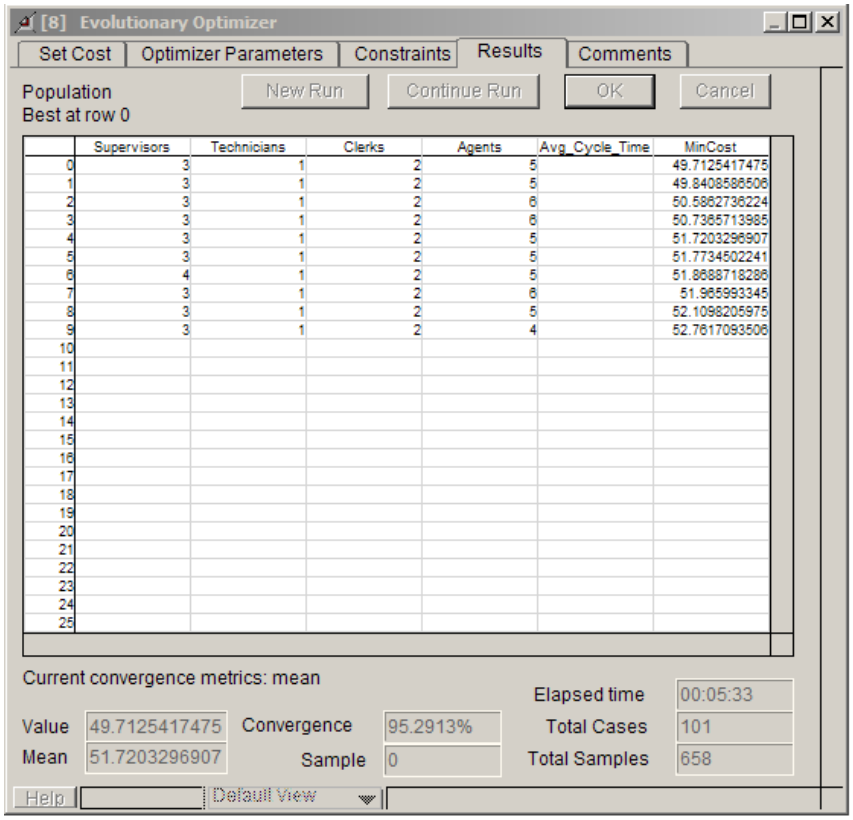
Quicker Defaults, Random Model and Better Defaults, Random Model:(სურ.61)



სურ.61

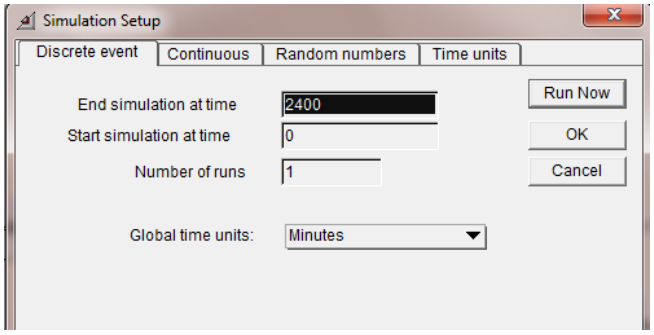
ძებნის რეჟიმი პარამეტრით Quicker Defaults გამოიყენება მაშინ, როცა გვჭირდება შედარებით სწრაფი ძებნა, ხოლო უფრო ზუსტი ძებნისთვის გამოიყენება ძებნის რეჟიმი პარამეტრით Better Defaults.

ამოირჩიეთ ძებნის ერთ-ერთი რეჟიმი და დააწკაპუნეთ ღილაკს New Run. დაიწყება ძებნა და გამოჩნდება შემდეგი (ან მსგავსი) ფანჯარა: სურ.62



სურ.62

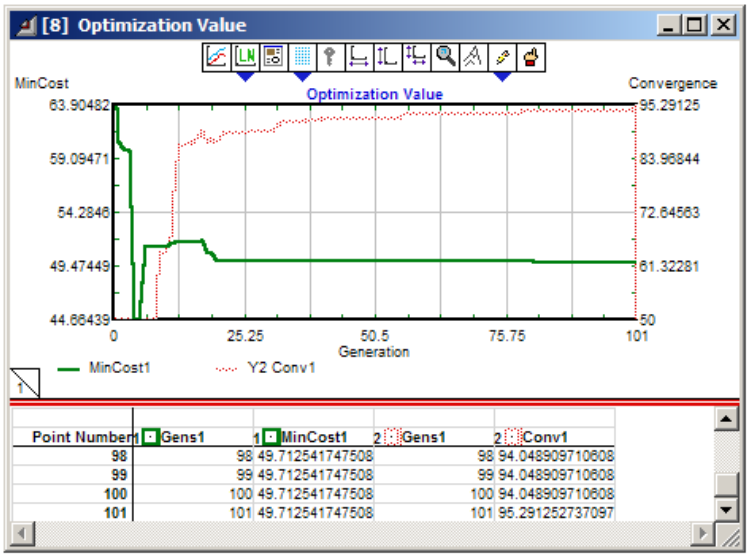
ზოგიერთი ამონახსნი ერთი და იგივეა, მაგალითად, 0, 1, 4, 5 და 8 რიგებში გადაწყვეტილებათა ცვლადების მნიშვნელობების კომბინაციებია: Supervisors=3, Technicians=1, Clerks=2, Agents=5. მაგრამ, რადგან ისინი მიღებული არიან სხვადასხვა სიმულაციის გაშვების დროს, საშუალო ციკლის დრო (Average Cycle Time) არის სხვადასხვა. იმისათვის, რომ დავაზუსტოთ მიღებული ოპტიმალური ამონახსნი, უნდა ამოვირჩიოთ ერთ-ერთი კომბინაცია (მაგალითად, ნულოვან სტრიქონიდან) და ჩავატაროთ დამატებითი ექსპერიმენტები. ამისათვის გადადით მენიუს ზოლზე, გახსენით პუნქტი Run -> Simulation Setup და ველში Numbers of runs შეიტანეთ რიცხვი 10 ან 25 ან 50 (რაც მეტია ექსპერიმენტების რაოდენობის რიცხვი, მით უფრო ზუსტ შედეგს მივიღებთ): სურ 63



სურ.63

დამატებითი ექსპერიმენტების ჩატარების შემდეგ შეგვიძლია ამოვირჩიოთ პროცესის ჩვენთვის საინტერესო მაჩვენებლები (მაგალითად, რესურსების გამოყენების დონეები, დახარჯული თანხები, რიგების სიგრძეები, რიგებში გატარებული დრო და ა.შ.), მივიღოთ ისინი როგორც პროცესის დიზაინის ყველაზე ზუსტი შედეგები, და ამ საფუძველზე გამოვიმუშავოთ კომპანიის ან ორგანიზაციის ხელმძღვანელობისთვის რეკომენდაციები, რათა მათ მიიღონ საბოლოო გადაწყვეტილება შემდგომი განვითარების შესახებ.

ეკრანზე აგრეთვე გამოჩნდება ოპტიმიზაციის მსვლელობის გრაფიკი:(სურ.64)

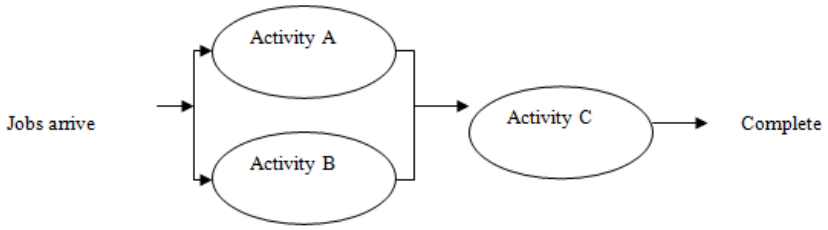


სურ.64

რომელიც აჩვენებს ძებნის **ტრაექტორიას,** ანუ მიზნობრივი ფუნქციის მნიშვნელობას, რომელიც ნაპოვნია მიმდინარე ოპტიმიზაციის გაშვებაში (run). გრაფიკში აგრეთვე ნაჩვენებია **კრებადობის (კონვერგენციის)** სიდიდე (convergence rate). ეს სიდიდე (დიაპაზონში 0÷100%) გვიჩვენებს, რამდენად განსხვავდებიან მიმდინარე

მომენტისთვის ნაპოვნი მიზნობრივი ფუნქციის საუკეთესო და უარესი მნიშვნელობები. ოპტიმიზაცია მთავრდება, როცა ეს მაჩვენებელი მიაღწევს წინასწარ დადგენილ მნიშვნელობას (სტანდარტულად ≈ 95%).

მაგალითი. ბიზნეს პროცესი შედგება 3 აქტივობისაგან (ოპერაციისგან): აქტივობა (ოპერაცია) A, აქტივობა(ოპერაცია) B, აქტივობა (ოპერაცია) C. კლიენტების (სამუშაოების) შემოდინების შორის დრო განაწილებულია ექსპონენციალურად საშუალოდ 3 წუთი. შემოდის ორი ტიპის სამუშაო: პირველი ტიპის სამუშაო შემოდის სისტემაში 60% შემთხვევაში და მეორე ტიპის სამუშაო - 40 % შემთხვევაში: სურ.65



სურ.65

როცა სამუშაო შემოდის სისტემაში, ის უნდა შევიდეს აქტივობის A ან აქტივობის B რიგში (balking). თუ აქტივობის A რიგის სიგრძე 5-ზე ნაკლებია, მაშინ სამუშაო შედის ამ რიგში. თუ აქტივობის A რიგის სიგრძე 5-ზე მეტია, მაშინ სამუშაო ამოწმებს აქტივობის B რიგის სიგრძეს და თუ იგი 5-ზე ნაკლებია, მაშინ სამუშაო შედის ამ რიგში. თუ ორივე სამუშაოს რიგის სიგრძე მეტია 5-ზე, მაშინ სამუშაო საერთოდ ტოვებს სისტემას დაუმუშავებლად.

აქტივობა (ოპერაცია) C ამჟამად ხორციელდება პარალელურად მომუშავე 2 ტრანზაქციის მეშვეობით, თითოეული აქტივობა (ოპერაცია) A და B - ერთი ტრანზაქციით. დამუშავების დროის განაწილებები და ხანგრძლივობა (წუთებში) მოცემულია ცხრილში:

	სამუშაოს ტიპი	
აქტივობა	ტიპი I (წუთებში)	ტიპი II (წუთებში))
აქტივობა A	Normal (5,2)	Normal (8,3)
აქტივობა B	Normal (3, 1)	Normal (5, 10)

აქტივობა C	Real Uniform (3, 10)	Real Uniform (8, 15)

აქტივობის C დასრულების შემდეგ პირველი ტიპის სამუშაოების 10% და მეორე ტიპის სამუშაოების 15 % თხოულობს განმეორებით დამუშავებას (აღმოჩენილი წუნის გამო).

აქტივობის (ოპერაციის) A და B ღირებულებაა 20 ლარი/საათში, ოპერაცია C – 30 ლარი/საათში. ყველა აქტივობის რიგის მაქსიმალური სიგრძეა 10 სამუშაო (კლიენტი). როცა თითოეული სამუშაო (კლიენტი) საბოლოოდ ტოვებს სისტემას, გენერირდება 100 ლარიანი შემოსავალი .

შექმენით მოცემული პროცესის ოპტიმიზაციის მოდელი შემდეგი პირობების გათვალისწინებით.

გადაწყვეტილებების ცვლადები:

1. აქტივობის (ოპერაციის) A პარალელურად მომუშავე ტრანზაქციების რაოდენობების დიაპაზონი : 1÷ 3

   აქტივობის (ოპერაციის) B პარალელურად მომუშავე ტრანზაქციების რაოდენობების

   დიაპაზონი : 1÷ 3

   აქტივობის (ოპერაციის) C პარალელურად მომუშავე ტრანზაქციების რაოდენობების

   დიაპაზონი : 2÷ 4
2. დაუმუშავებელი სამუშაოების რაოდენობა დიაპაზონში 0÷50
3. სრულად დამუშავებული სამუშაოების რაოდენობა დიაპაზონში 0÷1000 (მასალის შეზღუდვის გამო, დასრულებული სამუშაოების რაოდენობა არ უნდა აღემატებოდეს 1000-ს)

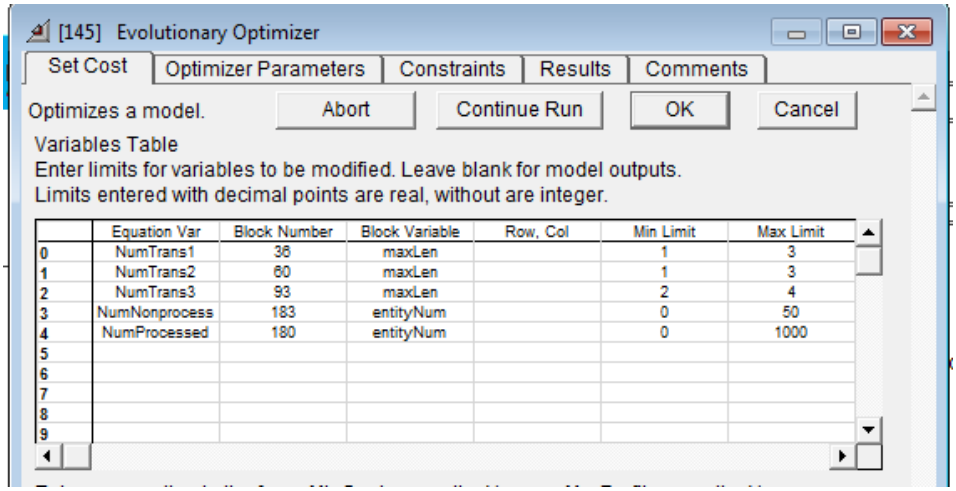
მიზნობრივი ფუნქცია: **Net Total Profit**-ის (სუფთა მოგების) მაქსიმიზაცია. სუფთა მოგება გამოიანგარიშება შემდეგნაირად: ყოველი დასრულებული სამუშაოს შემდეგ კომპანია იღებს შემოსავალს 100 ლარი ოდენობით. აქედან გამომდინარე, სუფთა მოგება გამოიანგარიშება როგორც: 100 * სრულად დამუშავებული სამუშაოების

რაოდენობა - (20 ლარი * 40საათი/კვირაში * აქტივობის A ტრანზაქციების რაოდენობა + 20 ლარი * 40საათი/კვირაში * აქტივობის B ტრანზაქციების რაოდენობა + 30 ლარი * 40საათი/კვირაში * აქტივობის C ტრანზაქციების რაოდენობა). აქ 20 და 30 ლარი არის ტრანზაქციების A, B და C აქტივობის ღირებულებები საათში, 40 საათი არის სამუშაო საათების რაოდნობა (5 დღე * 8 საათზე) .

გლობალური შეზღუდვაა : დაუმუშავებელი სამუშაოების რაოდენობა უნდა იყოს 10-ზე ნაკლები.

ამონახსნი.

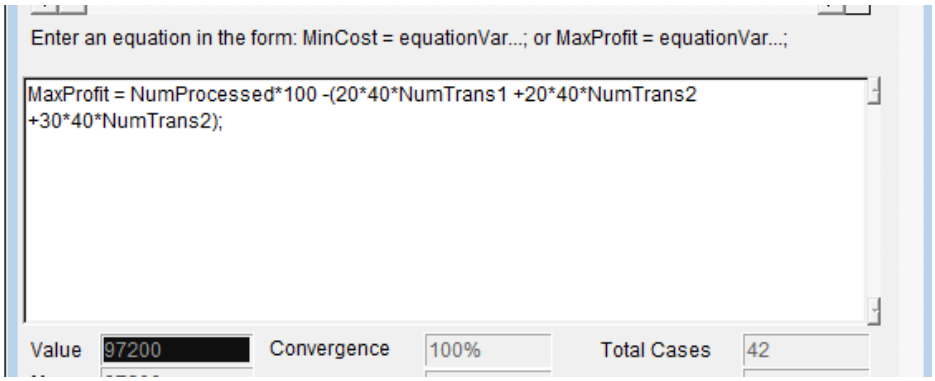
ზემოდ მოხსენებული მეთოდიკის თანახმად, ბლოკის Evolutionary Optimizer ფურცლის Set Cost გადაწყვეტილებათა ცვლადების დიალოგური ფანჯარა გამოიყურება შემდეგნაირად: (სურ.66)



სურ.66

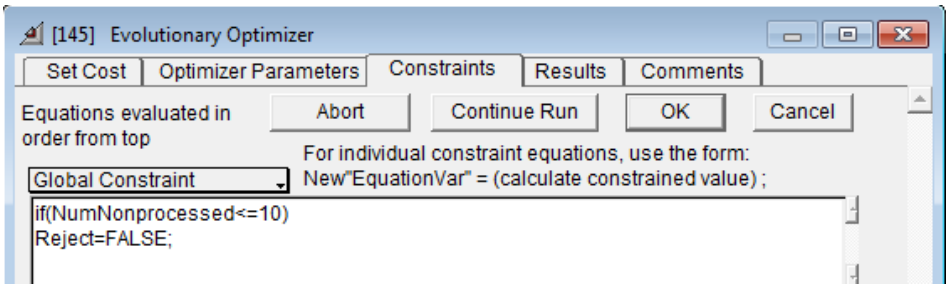
აქ ცვლადს „აქტივობის A ტრანზაქციების რაოდენობა“ მინიჭებული აქვს სახელი „NumTrans1”, ცვლადს „აქტივობის B ტრანზაქციების რაოდენობა“ - სახელი „NumTrans2”, ცვლადს „აქტივობის C ტრანზაქციების რაოდენობა“ - სახელი „NumTrans3”, ცვლადს “დაუმუშავებელი სამუშაოების რაოდენობა” - სახელი “NumNonprocessed”, ცვლადს სრულად დამუშავებული სამუშაოების რაოდენობა ” - სახელი “NumProcessed”. თქვენ შეგიძლიათ მიანიჭოთ ცვლადებს ნებისმიერი სხვა სახელი.

შეიტანეთ მიზნობრივი ფუნქცია:სურ.67



სურ.67

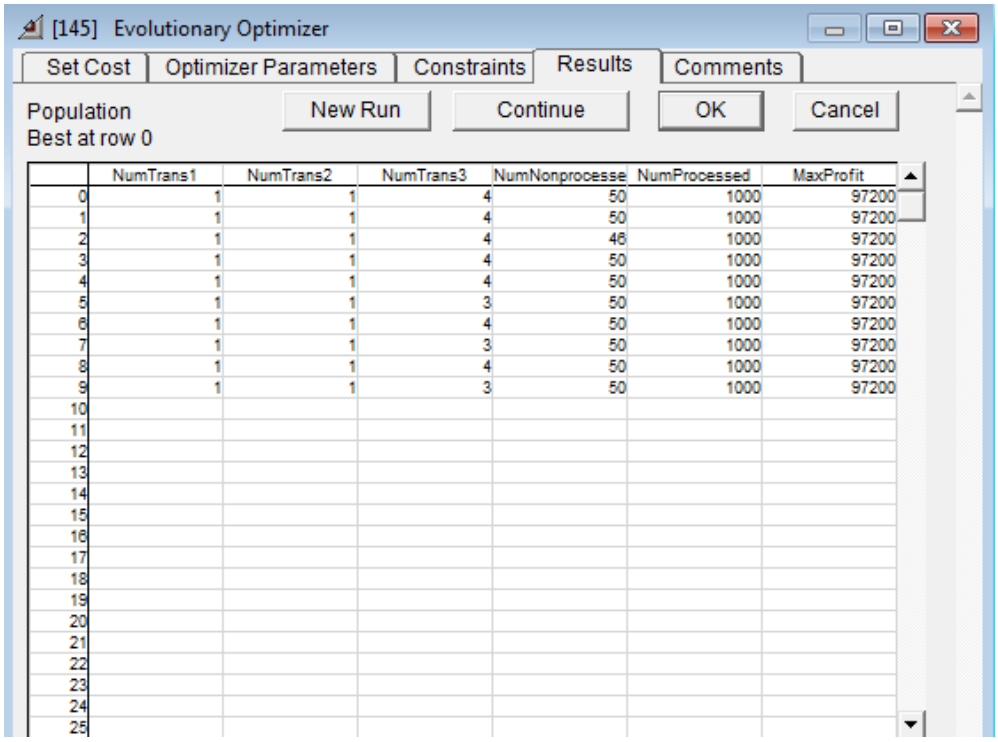
გადადით ფურცელზე Constraints და შეიტანეთ გლობალური შეზღუდვა: (სურ.67)



სურ.67

გადადით ფურცელზე Optimizer Parameters, ამოირჩიეთ რეჟიმი Quicker Defaults, Random Model და მერე დააჭირეთ New Run.

თქვენ მიიღებთ შემდეგ (ან მსგავს) შედეგს:სურ.68



	NumTrans1	NumTrans2	NumTrans3	NumNonprocesse	NumProcessed	MaxProfit
0	1	1	4	50	1000	97200
1	1	1	4	50	1000	97200
2	1	1	4	46	1000	97200
3	1	1	4	50	1000	97200
4	1	1	4	50	1000	97200
5	1	1	3	50	1000	97200
6	1	1	4	50	1000	97200
7	1	1	3	50	1000	97200
8	1	1	4	50	1000	97200
9	1	1	3	50	1000	97200
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						

სურ.68

აქედან ჩანს, რომ ოპტიმალური შედეგი შეესაბამება კომბინაციას:

აქტივობის A ტრანზაქციების რაოდენობა (ცვლადის სახელი „NumTrans1)=1;

აქტივობის B ტრანზაქციების რაოდენობა (ცვლადის სახელი „NumTrans2)=1;

აქტივობის C ტრანზაქციების რაოდენობა (ცვლადის სახელი „NumTrans3)=4;

სუფთა მოგება (MaxProfit) =97200 ლარი