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

მიხეილ ჩუბინიძე

ექსტრემუმის ძებნის ავტომატური მოდელი

სამაგისტრო პროგრამა: ინფორმაციული სისტემები

სამაგისტრო ნაშრომი შესრულებულია ინფორმაციულ სისტემებში მეცნიერების მაგისტრის აკადემიური ხარისხის მოსაპოვებლად

ხელმძღვანელი: ტარიელ ხვედელიძე
ფიზ.-მათ. მეცნიერებათა კანდიდატი,
ასოცირებული პროფესორი

თბილისი
2015

ანოტაცია

სამაგისტრო ნაშრომში განხილულია ხმაურის პირობებში ექსტრემუმის ძებნის ბიჯური ალგორითმი. განცალკავებული სასინჯი და მუშა ბიჯების ალგორითმისაგან განსხვავებით, რომლის დროსაც ძებნის სწრაფქმედება ეცემა, განიხილება ძებნის ბიჯური ალგორითმი შეთავსებული მუშა და სასინჯი ბიჯებით. ექსტრემუმის ძებნის ბიჯური ალგორითმის უპირატესობა იმაში მდგომარეობს, რომ ბიჯური ალგორითმი მარტივია აპარატული რეალიზაციისთვის და ხმაურის პირობებში ძებნისთვის მისი მოდიფიცირება.

ხმაურის პირობებში ძებნის ეფექტურობის, სწრაფქმედების და საიმედოობის გასაუმჯობესებლად განიხილება ბიჯური ალგორითმის მოდიფიცირებული მეთოდი - ზღურბლური ფილტრაცია. ამ მეთოდის იდეა მდგომარეობს იმაში, რომ ხმაური ადვილად ახშობს მცირე სიგნალებს, ხოლო დიდი ზომის სიგნალის დამახინჯება ხმაურით რთულია. ხელშეშლის გავლენის შემცირება კი შესაძლებელია მოდულით დიდი სიგნალების გამოყენებით.

შემთხვევითი ძებნის ერთ-ერთი საინტერესო მოდელს წარმოადგენს ე.წ. ავტომატური ძებნა. ამ მოდელში ავტომატი გამოიყენება ალგორითმის სამართავად. ზღურბლური ფილტრაციის ალგორითმში მმართველი ორგანოს როლი დაეკისრება ისეთ სასრულ ავტომატებს, რომლებიც სამი კლასის რეაქციების მქონე ტერნარულ სტაციონარულ შემთხვევით გარემოში ხასიათდებიან ქცევის მიზანშეწონილობით. ავტომატური ოპტიმიზაციის იდეა იმაში მდგომარეობს, რომ ოპტიმიზაციის ყოველი პარამეტრი იმართება ურთიერთდამოკიდებული სტოქასტური ავტომატით. ამ მეთოდით ხდება თვითორგანიზებადი შემთხვევითი ძებნის ალგორითმის რეალიზაცია.

Annotation

In this document is described extremum searching step-based algorithm in conditions of noise. In contrast of isolated trial and working steps algorithm, witch case decrease speed, we consider searching step-based algorithm combined with working and trial steps. Advantage of extremum searching step-based algorithm is that, step-based algorithm implementation is simple on hardware and so it's simple to modify in conditions of noise.

To increase searching efficiency, speed and reliability in conditions of noise we consider step-based algorithm modified method – threshold filtration. Idea of this method is that, noise easily suppresses on small size signals, but distortion of big size signals using noise is difficult. Distortion impact reduction is possible using big module signals.

One of the interesting model of random search is "Automated Search". In this model automate is used to manage algorithm. In this model as managing authority will be finite automates, witch has behavior advisability in three reaction class ternal stationary environment. Automate optimization idea is that, every optimization parameter calculate using interdependent stochastic automate. Using this method happens implementation of self managed searching algorithm.

სარჩევი

შესავალი

ძებნის ექსტრემალური მართვის მოდელები, რომლებიც არ ითვალისწინებენ შემთხვევით ფაქტორებს, წარომადგენენ მხოლოდ აბსტრაქტულ შემთხვევებს. ეს შემთხვევითი ფაქტორები გარდაუვალია ყველა რეალურ სისტემაში. ძებნის სისტემებში ხელშეშლა შეიძლება ბევრი სხვადასხვა მიზეზით იყოს გამოწვეული:

- გაზომვის შეცდომები.
  მართვის ყველა სისტემა, მათ შორის ექსტრემალურიც, ითვალისწინებს სხვადასხვა სახის გაზომვებს, ხოლო გაზომვები ყოველთვის მიმდინარეობს გარკვეული ცდომილებებით.

- დამრგვალების შეცდომები.
  ნებისმიერ მოწყობილობას გააჩნია თავისი სიზუსტე. ეს სიზუსტე განსაზღვრავს დასამახსოვრებელი სიდიდის დამრგვალების შეცდომების ხარისხს, რომლებიც ექსტრემალური რეგულატორის მუშაობაზე ახდენს გავლენას.

- კავშირის არხზე ხმაური.
  თუ ობიექტის ხარისხის მაჩვენებელი საკმარისად მოშორებულია მარეგულირებელ მოწყობილობას, მაშინ მათ შორის კავშირის არხში შესაძლოა ხელშეშლის არსებობა.

- ობიექტის საკუთარი ხმაური.
  ზოგიერთ რეალურ ობიექტს გააჩნია საკუთარი ხმაურის მნიშვნელოვანი ფონი. შინაგანი ხმაური წარმოიქმნება დიდი რაოდენობით გაუთვალისწინებელი ცვალებადი ფაქტორის ხარჯზე.

- ობიექტის თვისებების შეცვლა.
  ყველა რეალური ობიექტისთვის მუშაობის პროცესში გარდაუვალია მისი თვისებების შეცვლა. ამ თვისებებს გააჩნიათ შემთხვევითი ხასიათი და ისინი ქმნიან გარკვეული სახის განუსაზღვრელობას ოპტიმიზაციის ობიექტის ყოფაქცევაში.

  შესაძლოა არსებობდეს აგრეთვე შემთხვევითი ხელშეშლების სხვა წყაროებიც.

ექსტრემუმის ძებნის ბიჯური ალგორითმის უპირატესობა იმაში მდგომარეობს, რომ ბიჯური ალგორითმი მარტივია აპარატული რეალიზაციისთვის და ხმაურის პირობებში ძებნისთვის მისი მოდიფიცირება. ბიჯური ალგორითმის გაუმჯობესებულ მეთოდს ხელშეშლის პირობებში წარმოადგენს ზღურბლური ფილტრაცია.

ვთქვათ ოპტიმიზაციის ობიექტს ხელშეშლის გარეშე აქვს შემდეგი სახის ექსტრემალური მახასიათებელი:

$$Q = Q(x)$$

მაშინ ε_1 და ε_2 ხელშეშლის პირობებში, იგივე მოდელი შეიძლება ასე წარმოვადგინოთ:

$$Q' = Q(x + \varepsilon_1) + \varepsilon_2$$

აქ ε_1 და ε_2 ხმაურის დამახასიათებელი შემთხვევითი სიდიდეები ან ფუნქციებია, რომლებითაც ხდება Q-ს გაზომვის ან დამრგვალების ხარისხის მოდელირება. ეს სიდიდეები შეიძლება იყოს როგორც დისკრეტული, ასევე უწყვეტი ხასიათის. პირველ შემთხვევაში, ისინი შემთხვევითი რიცხვები არიან, მეორე შემთხვევაში - უწყვეტი ფუნქციები.

შემთხვევითი ძებნის ერთ-ერთი საინტერესო მოდელია ე.წ. ავტომატური ძებნა. ამ მოდელში ავტომატი გამოიყენება ალგორითმის სამართავად. ზღურბლური ფილტრაციის ალგორითმში მმართველი ორგანოს როლი დაეკისრება ისეთ სასრულ ავტომატებს, რომლებიც სამი კლასის რეაქციების მქონე ტერნარულ სტაციონარულ შემთხვევით გარემოში ხასიათდებიან ქცევის მიზანშეწონილობით. ავტომატური ოპტიმიზაციის იდეა იმაში მდგომარეობს, რომ ოპტიმიზაციის ყოველი პარამეტრი იმართება ურთიერთდამოკიდებული სტოქასტური ავტომატით. ამ მეთოდით ხდება თვითორგანიზებადი შემთხვევითი ძებნის ალგორითმის რეალიზაცია.

ძებნის ბიჯური ალგორითმი

ძებნის ბიჯურ სისტემებში ჩვეულებრივ სასინჯ ბიჯებს მოჰყვება ე.წ. „მუშა“ გადაადგილებები, რომლებიც თავის მხრივ წარმოადგენენ წინასწარ მოცემული სიდიდის ბიჯებს. მაგ. g სიდიდის სასინჯი ბიჯის შემდეგ ორ მეზობელ $x \pm g$ წერტილში ამა თუ იმ მიმართულებით შეიძლება შესრულდეს მუშა ბიჯი (გადაადგილება), რომელიც თავის მხრივ დამოკიდებულია წინა სასინჯ ბიჯებზე მიღებული შედეგის ანალიზზე. ასეთ სისტემებს ეწოდება სისტემები განცალკავებული სასინჯი და მუშა ბიჯებით. მაგრამ ამ დროს ძებნის სწრაფქმედება ეცემა. ამ პრობლემის თავიდან ასაცილებლად გამოიყენება ძებნის ბიჯური ალგორითმი შეთავსებული მუშა და სასინჯი ბიჯებით.

ამ შემთხვევაში ზემოთ მოყვანილ ალგორითმში ვასრულებთ ერთ-ერთ სასინჯ ბიჯს და ვანაცვლებთ x-ის თავდაპირველ მნიშვნელობას ძებნის ყოველ ბიჯზე. მაშასადამე შევასრულებთ მხოლოდ მუშა ბიჯებს და ძებნის ყოველ ბიჯზე ვიმახსოვრებთ ხარისხის მაჩვენებლის მიღებულ მნიშვნელობას, რომელსაც გამოვიყენებთ შემდგომი მუშა ბიჯის მიმართულების დასადგენად.

მოცემული ალგორითმის არსი მდგომარეობს იმაში, რომ წარმატების დროს ვმოქმედებთ იგივენაირად, წარუმატებლობისას კი პირიქით. მოქმედების ასეთი სტრატეგია ე.წ. წრფივი ძებნის i-ურ ბიჯზე მუშა გადაადგილება შეიძლება განვსაზღვროთ შემდეგი ფორმულის საშუალებით:

$$\Delta x_i = \begin{cases} \Delta x_{i-1} & \Delta Q_{i-1} < -\delta \\ -\Delta x_{i-1} & \Delta Q_{i-1} > \delta \\ 0 & \Delta Q_{i-1} < |\delta| \end{cases}$$

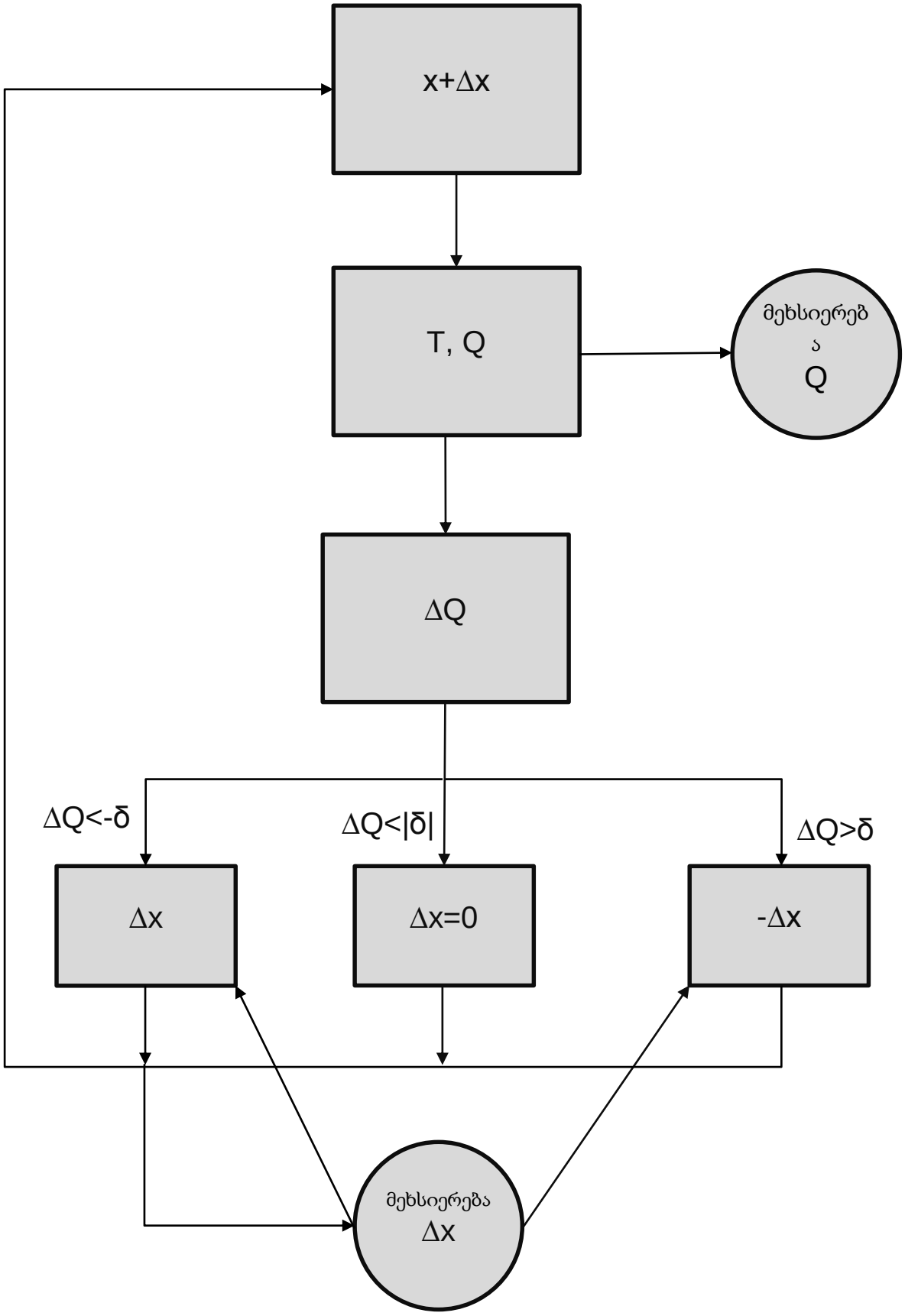
სადაც იგულისხმება, რომ

$$\Delta Q_{i-1} = Q(x_{i-1}) - Q(x_{i-2}), \qquad |\Delta x_i| = a,$$

ხოლო δ არის წინასწარ მოცემული რაიმე დადებითი რიცხვი (ზღურბლი).

განხილულ ალგორითმში ყოველი მომდევნო ბიჯის მიმართულების არჩევა ხდება იმ ინფორმაციის საფუძველზე, რომელიც მიიღება ერთმანეთისგან a მანძილით დაშორებულ წერტილებში.

ძებნის ბიჯური ალგორითმის სარეალიზაციო ბლოკ-სქემას აქვს შემდეგი სახე:


x+Δx
T, Q
მეხსიერება
Q
ΔQ
ΔQ<-δ
ΔQ<|δ|
ΔQ>δ
Δx
Δx=0
-Δx
მეხსიერება
Δx

მოცემულ სქემაზე Δx და $-\Delta x$ ოპერატორები უჩვენებენ შესაბამისად მუშა ბიჯს და მის საწინააღმდეგო მნიშვნელობას. ΔQ არის ხარისხის მაჩვენებლის ნაზრდი. T ოპერატორი არის სისტემის დაყოვნება T დროის განმავლობაში.

Q მეხსიერება აუცილებელია ალგორითმის ფუნქციონირებისათვის (საშუალედო შედეგების დამახსოვრება და სხვა). განხილული ალგორითმის გამოყენების დროს ადგილი აქვს დროის ეკონომიას, რაც დაკავშირებულია სასინჯი და მუშა ბიჯების შეთავსებასთან.

შემდგომ ჩვენ საქმე გვექნება სწორედ ასეთი ტიპის ბიჯური სისტემების სპეციალურ მოდიფიკაციასთან, რომელშიც განსაკუთრებულ როლს თამაშობს სასრული ავტომატი, როგორც მმართველი ორგანო.

§ 2. ზოგადი ცნებები სტაციონარულ შემთხვევით გარემოში სასრული და უსასრულო ავტომატების ფუნქციონირების შესახებ

ავტომატის ქვეშ იგულისხმება რაიმე მოწყობილობა, რომელიც ფუნქციონირებს დისკრეტული დროის $t = 1,2, \dots$ მომენტებში და გააჩნიათ შემდეგი თვისებები:

ა) ამ მოწყობილობას გააჩნია შიგა მდგომარეობათა სასრული ან თვლადი სიმრავლე. დროის ყოველ t მომენტში ავტომატი იმყოფება ამ მდგომარეობებიდან ერთ-ერთში.

ბ) ავტომატს შეუძლია შეასრულოს სასრული რაოდენობის რაიმე მოქმედება, რომლის არჩევაც დამოკიდებულია ავტომატის შიგა მდგომარეობაზე.

გ) ავტომატს შესასვლელზე შეუძლია მიიღოს სასრული რაოდენობის შემავალი სიგნალები და მიღებულ სიგნალებზე დამოკიდებულებით ცვალოს თავისი შიგა მდგომარეობები.

მაშასადამე, სტოქასტური (ალბათური) ავტომატი ფორმალურად განიმარტება როგორც

$$A_k = < S, F_k, \pi_0, A(s), \mu(f/x) >$$

სისტემა, სადაც $S = \{s_1, s_2, \dots, s_g\}$ შემავალი სიგნალების სასრული სიმრავლეა; $F_k = \{f_1, f_2, \dots, f_k\}$ - გამომავალი სიგნალების (მოქმედებების) სასრული სიმრავლე; $L = \{x_1, x_2, \dots, x_n, \dots\}$ - შიგა მდგომარეობათა სასრული ან თვლადი სიმრავლე; π_0 - მდგომარეობათა ალბათობების საწყისი (სასტარტო) განაწილება; $A(s)$- გადასვლის ფუნქცია, რომელიც მოიცემა სასრულგანზომილებიანი ან თვლადგანზომილებიანი გადასვლის ალბათობების $(a_{ij}(s))$ სტოქასტური მატრიცებით, სადაც

$$a_{ij}(s) = P(x(t+1) = x_j \,/\, x(t) = x_i);$$

$\mu(f/x)$- გამოსასვლელის ფუნქცია (პირობითი ალბათური განაწილება F_k- ზე), რომელიც იძლევა $L \to F_k$ ალბათურ ასახვას.

შევუსაბამოთ A_k სტოქასტურ ავტომატს

$$A_{k,\alpha} = \langle\, S,\ F_k,\ L_\alpha,\ A^\alpha(s) \,\rangle,\ \alpha = 1,2,\dots,k$$

ქვეავტომატები ერთადერთი f_α გამოსასვლელით და L_α ქვესიმრავლეზე ინდუცირებული $A^\alpha(s)$ გადასვლის ფუნქციებით.

ჩამოვაყალიბოთ ზოგიერთი დაშვება A_k ავტომატის სტრუქტურის შესახებ:

1) ყველა $A_{k,\alpha}$, $\alpha = 1,2,\dots,k$ ავტომატი იზომორფულია, ე.ი. განსხვავდებიან მხოლოდ მდგომარეობათა აღნიშვნით. აქედან გამომდინარეობს, რომ სასრულგანზომილებიან შემთხვევაში მდგომარეობათა L_α , $\alpha = 1,2,\dots,k$ ქვესიმრავლეებს გააჩნიათ ელემენტების ერთი და იგივე რაოდენობა და A_k ავტომატის მდგომარეობათა L სიმრავლე წარმოადგენს $A_{k,\alpha}$ ქვეავტომატების მდგომარეობათა სიმრავლეების არაგადამკვეთი ქვესიმრავლეების გაერთიანებას:

$$L = \cup_{\alpha=1}^{k} L_\alpha\ .$$

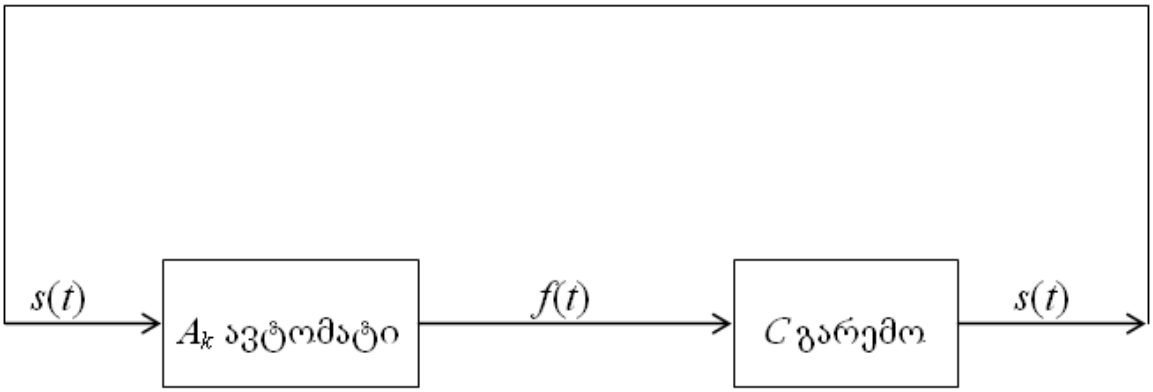
2) $L_\alpha \epsilon\, L$ მდგომარეობათა ნებისმიერი ქვესიმრავლიდან შესაძლებელია მდგომარეობათა ნებისმიერ სხვა $L_\beta \epsilon\, L$, $\alpha \neq \beta$ ქვესიმრავლეში გადასვლა.

შემდგომში მდგომარეობათა L_α ქვესიმრავლიდან მდგომარეობათა $L_{\alpha+1}$ ქვესიმრავლეში გადასვლისას გავითვალისწინებთ მხოლოდ ციკლურ გადასვლებს (სიმბოლურად $L_\alpha \to L_{\alpha+1},\ \alpha = 1,2,\dots,k-1,\quad L_k \to L_1$), ხოლო $A_k^{(n)}$ და A_k სიმბოლოებით ავღნიშნავთ შესაბამისად სასრული და უსასრულო (მდგომარეობათა თვლადი რიცხვით) ავტომატებს, $L^{(n)}$ და L_α სიმბოლოებით კი მათ მდგომარეობათა სიმრავლეს. ცხადია, რომ

$$L^{(n)} = \bigcup_{\alpha=1}^{k} L_\alpha^{(n)}$$

განვიხილოთ ავტომატის ქცევა შემთხვევით C გარემოში. ეს ნიშნავს, რომ ავტომატის გამომავალი სიგნალები (მოქმედებები) წარმოადგენენ შემავალ სიგნალებს რაიმე C

მოწყობილობისათვის. C გარემო ავტომატის ქცევაზე რეაგირებს საპასუხო რეაქციებით, რომლებიც ავტომატისათვის წარმოადგენს შემავალ სიგნალებს, ხოლო ავტომატი მათ იყენებს შემდგომი გადაწყვეტილებების მისაღებად (ნახ.1).



ნახ.1

ვიგულისხმოთ, რომ C გარემოს ყველა შესაძლებელი $S \in \{s_1, s_2, \dots, s_g\}$ რეაქცია ავტომატის მიერ აღიქმება, როგორც ერთ-ერთი სახის რეაქცია შემდეგი სამი კლასიდან - სასურველი რეაქციების კლასი (მოგება, დაჯილდოება, $s = +1$), არასასურველი რეაქციების კლასი (წაგება, დაჯარიმება, $s = -1$) და ნეიტრალური რეაქციების კლასი (ინდიფერენტულობა, $s = 0$).

ვიტყვით, რომ ავტომატი მოწყობილია მიზანშეწონილად, თუ იგი ხშირად იგებს და იშვიათად აგებს. ცხადია, რომ ავტომატის მიზანშეწონილობა იზრდება სასურველი რეაქციების (მოგება) რიცხვის გაზრდითა და არასასურველი რეაქციების (წაგება) რიცხვის შემცირებით.

განმარტება 1. ვიტყვით, რომ A_k ავტომატი ფუნქციონირებს სტაციონარულ $C = C(a_1, r_1; a_2, r_2; \dots; a_k, r_k)$ შემთხვევით გარემოში, თუ ავტომატის ქცევა და შემავალი სიგნალის მნიშვნელობები ერთმანეთთან დაკავშირებულია შემდეგნაირად: ავტომატის მიერ დროის t მომენტში შესრულებული f_α მოქმედება ავტომატის შესასვლელზე იწვევს დროის შემდგომ $t+1$ მომენტში $s = +1$ სიგნალის (მოგება) მოსვლას $q_\alpha = \frac{1-r_\alpha+a_\alpha}{2}$ ალბათობით, $s = -1$ სიგნალის (წაგება) მოსვლას $p_\alpha = \frac{1-r_\alpha-a_\alpha}{2}$ ალბათობით და $s = 0$ სიგნალის (ინდიფერენტულობა) მოსვლას $r_\alpha = 1 - q_\alpha - p_\alpha$ $(\alpha = 1,2, \dots, k)$ ალბათობით.

აქ $a_\alpha = q_\alpha - p_\alpha$ $(|a_\alpha| < 1 - r_\alpha,\ \alpha = 1,2,\dots,k)$ სიდიდეს აქვს f_α მოქმედების შესრულებისას მოგების მათემატიკური მოლოდინის აზრი.

განსაკუთრებულ ინტერესს იწვევს ისეთი ავტომატები, რომელთა ქცევა მიზანშეწონილია და რომელთა სტრუქტურაშიც არავითარი ინფორმაცია არაა იმის შესახებ, თუ როგორ სტაციონარულ გარემოში უხდებათ მათ ფუნქციონირება. ასეთი

ავტომატების სტრუქტურა უნდა უზრუნველყოფდეს სიმეტრიულობის თვისებას: ავტომატის მიერ სხვადასხვა მოქმედების შესრულებისას ავტომატის შესასვლელზე

შემავალი სიგნალების ერთი და იგივე მიმდევრობის შემთხვევაში ავტომატის ქცევა უნდა იყოს ერთნაირი.

ვთქვათ $x_i^\alpha \in L_\alpha$, $\alpha = 1,2,\dots,k,\quad i = 1,2,\dots,n$ $A_{k,\alpha}$ ქვეავტომატების ისეთი მდგომარეობებია, რომ $F(x_i^\alpha) = f_\alpha$.

x_i^α მდგომარეობის სიღრმე ვუწოდოთ შემავალი სიგნალების უმცირეს სიგრძეს, რომელსაც x_i^α მდგომარეობა გამოჰყავს L_α ქვესიმრავლიდან. A_k ავტომატის $d(A_k)$ სიღრმე ვუწოდოთ მდგომარეობათა სიღრმეებს შორის უდიდესს. ცხადია, რომ $d(A_k) \le n$, სადაც n არის $A_{k,\alpha}$ ქვეავტომატის მდგომარეობათა რაოდენობა.

შემდეგში ჩვენ განვიხილავთ ისეთი ავტომატების ფუნქციონირებას სტაციონარულ შემთხვევით გარემოში, რომელთა $d(A_k)$ სიღრმე ემთხვევა მდგომარეობათა n რაოდენობას: $d(A_k) = n$.

A_k ავტომატის ფუნქციონირება $C(a_1, r_1; a_2, r_2; \dots; a_k, r_k)$ სტაციონარულ შემთხვევით გარემოში აღიწერება მარკოვის ჯაჭვებით. ჩვენთვის საინტერესო შემთხვევებში ეს ჯაჭვები ერგოდულია. მაშასადამე, აღნიშნულ გარემოში არსებობს ავტომატის მდგომარეობათა ფინალური ალბათობები, რომლებიც არაა დამოკიდებული საწყის მდგომარეობაზე. შესაბამისად, $C(a_1, r_1; a_2, r_2; \dots; a_k, r_k)$ სტაციონარულ შემთხვევით გარემოში A_k ავტომატის მოგების მათემატიკური $M(A_k; C)$ მოლოდინი გამოისახება ფორმულით :

$$M(A_k; C) = \sum_{\alpha=1}^{k} \sigma_\alpha a_\alpha$$

სადაც σ_α, $\sum_{\alpha=1}^{k} \sigma_\alpha = 1$ სიდიდეებს აქვს A_k ავტომატის მიერ სტაციონარულ შემთხვევით გარემოში f_α მოქმედების შესრულების ალბათობის აზრი.

განმარტება 2. A_k ავტომატის ქცევა სტაციონარულ შემთხვევით $C(a_1, r_1; a_2, r_2; \dots; a_k, r_k)$ გარემოში მიზანშეწონილია, თუ $M(A_k; C) > M_0$; არაა მიზანშეწონილი, თუ $M(A_k; C) < M_0$; ინდიფერენტულია, თუ $M(A_k; C) = M_0$.

აქ

$$M_0 = \frac{1}{k}\sum_{\alpha=1}^{k} a_\alpha$$

ისეთი ავტომატის მოგების მათემატიკური მოლოდინია, რომელიც თავის მოქმედებას ირჩევს გარემოს რეაქციებისაგან დამოუკიდებლად და თანაბარალბათურად.

ცხადია, რომ

$$\min_\alpha a_\alpha < M(A_k; C) < \max_\alpha a_\alpha \ ,$$

მაგრამ შეიძლება ავაგოთ სასრული ავტომატების ისეთი მიმდევრობები, რომ ავტომატის მოგების მათემატიკურმა მოლოდინმა მიაღწიოს მაქსიმალურ მნიშვნელობას. ასეთ მიმდევრობებს ეწოდებათ ასიმპტოტურად ოპტიმალური მიმდევრობები.

ხმაურის პირობებში ძებნის გაუმჯობესებული მეთოდი (ზღურბლური ფილტრაცია)

ხმაურის პირობებში ძებნის ეფექტურობა არსებითად მცირდება. რაც უფრო მეტია ხმაური, მით უფრო მეტად მცირდება ძებნის სწრაფქმედება და საიმედოობა. ამიტომ, ბუნებრივია ახალი მეთოდების შექმნა ხმაურის პირობებში ძებნის გაუმჯობესებითვის.

განვიხილოთ ექსტრემუმის ძებნის გაუმჯობესების ერთ-ერთი მეთოდი - ზღურბლური ფილტრაცია. ამ მეთოდის იდეა მდგომარეობს იმაში, რომ ხმაური ადვილად ახშობს მცირე სიგნალებს, ხოლო დიდი ზომის სიგნალის დამახინჯება ხმაურით რთულია. ხელშეშლის გავლენის შემცირება შესაძლებელია მოდულით დიდი სიგნალების გამოყენებით.

ზღურბლური ფილტრაციის ალგორითმი ჩაიწერება შემდეგი სახით:

$$\Delta x_i = -a * \alpha_i$$

სადაც a - ბიჯის სიდიდე, ხოლო α_i იცვლება i-ურ ბიჯზე მიღებული შედეგების მიხედვით:

$$\alpha_i = \begin{cases} sign\ \Delta Q_i' & |\Delta Q_i'| \geq \delta \\ 0 & |\Delta Q_i'| < \delta \end{cases}$$

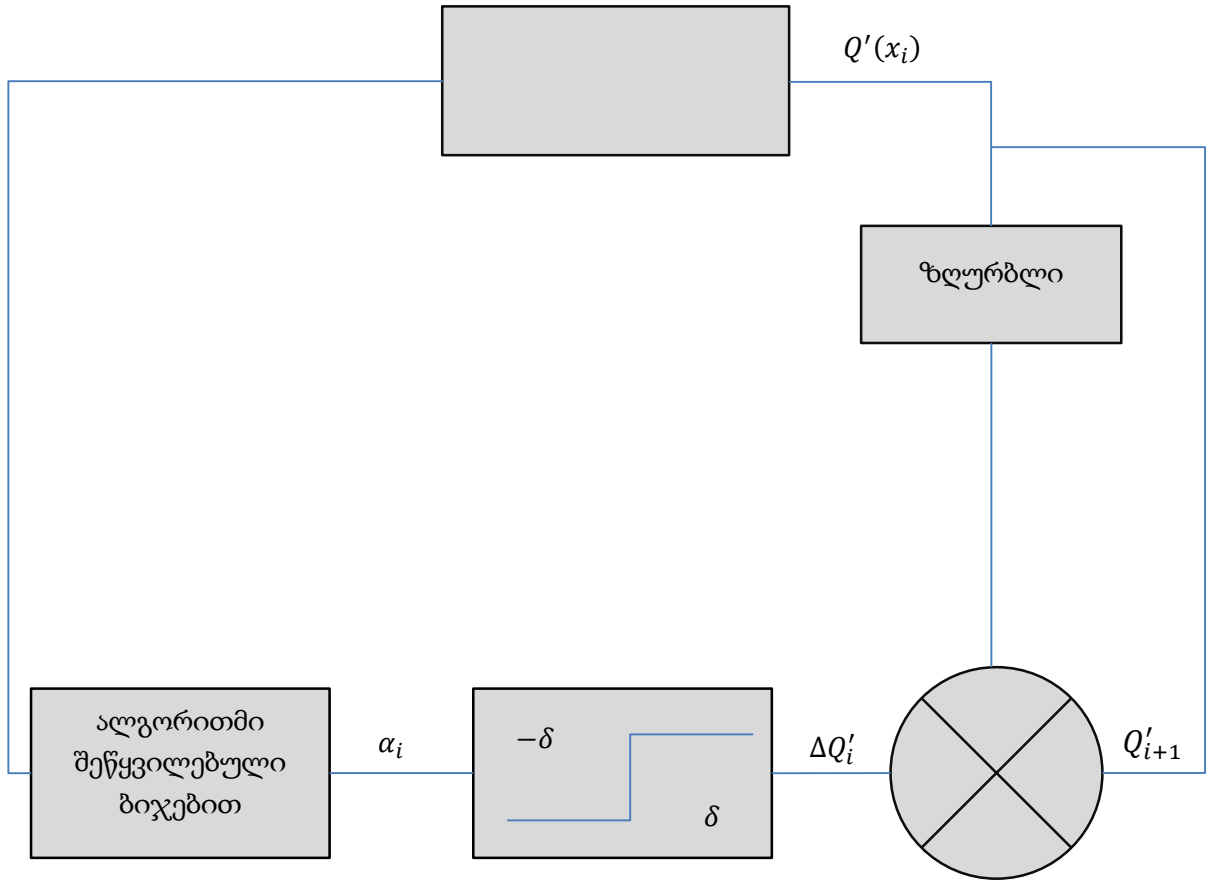
აქ δ არის წინასწარ მოცემული რაიმე დადებითი რიცხვი (ზღურბლი), ხოლო

$$\Delta Q_i' = Q'(x_i) - Q'(x_{i-1})$$

აქ Q-ზე ზემოქმედებას ახდენს ε ხმაური, რომელიც წარმოადგენს შემთხვევით სიგნალებს მუდმივი მათემატიკური მოლოდინით და σ^2 დისპერსიით.

როგორც ვხედავთ, ალგორითმი მდგომარეობს იმაში, რომ მუშა ბიჯი სრულდება მხოლოდ მაშინ, თუ ხარისხის მაჩვენებლის გაზომილი მნიშვნელობის ნაზრდის მოდული აღემატება მოცემულ დადებით δ ზღურბლს.

ზღურბლური ფილტრაციის გამოყენებით ოპტიმიზატორის ბლოკ-სქემას აქვს სახე



ნახ.2

ამ სქემაზე შემოტანილია ზღურბლური ელემენტი, რომლის გამოსავალზეც გვაქვს α სიდიდე.

განვსაზღვროთ შეცდომის ალბათობა

$$P_0 = P(sign\Delta Q \neq sign\Delta Q'), \quad \text{როცა } |\Delta Q'| \geq \delta$$

და სწორი გადაწყვეტილების მიღების ალბათობა

$$P_1 = P(sign\Delta Q = sign\Delta Q'), \text{ როცა } |\Delta Q'| \geq \delta$$

ხოლო გაჩერების ალბათობა ე.ი. $\Delta x = 0 (\propto = 0)$, ტოლია

$$P_{stop} = P(|\Delta Q| < \delta) = 1 - P_0 - P_1$$

განვიხილოთ შემთხვევები

$$k_i = \frac{dQ(x_i)}{dx} > 0$$

მაშინ

$$P_0 = P\left[ak_i + \varepsilon_1^i - \varepsilon_2^i < -\delta\right] = P[\xi_i < -\delta - ak_i]$$

სადაც $\xi_i = \varepsilon_1^i - \varepsilon_2^i$ დამოუკიდებელი ნორმალური შემთხვევითი სიდიდეა ნულოვანი მათემატიკური მოლოდინით და $2\sigma^2$ დისპერსიით.

ამ შემთხვევაში ვიღებთ

$$P_0 = \frac{1}{2}\left[1 - \Phi\left(\frac{\sigma + 2k_i\alpha}{2\sigma}\right)\right]$$

სადაც Φ არის ლაპლასის ფუნქცია.

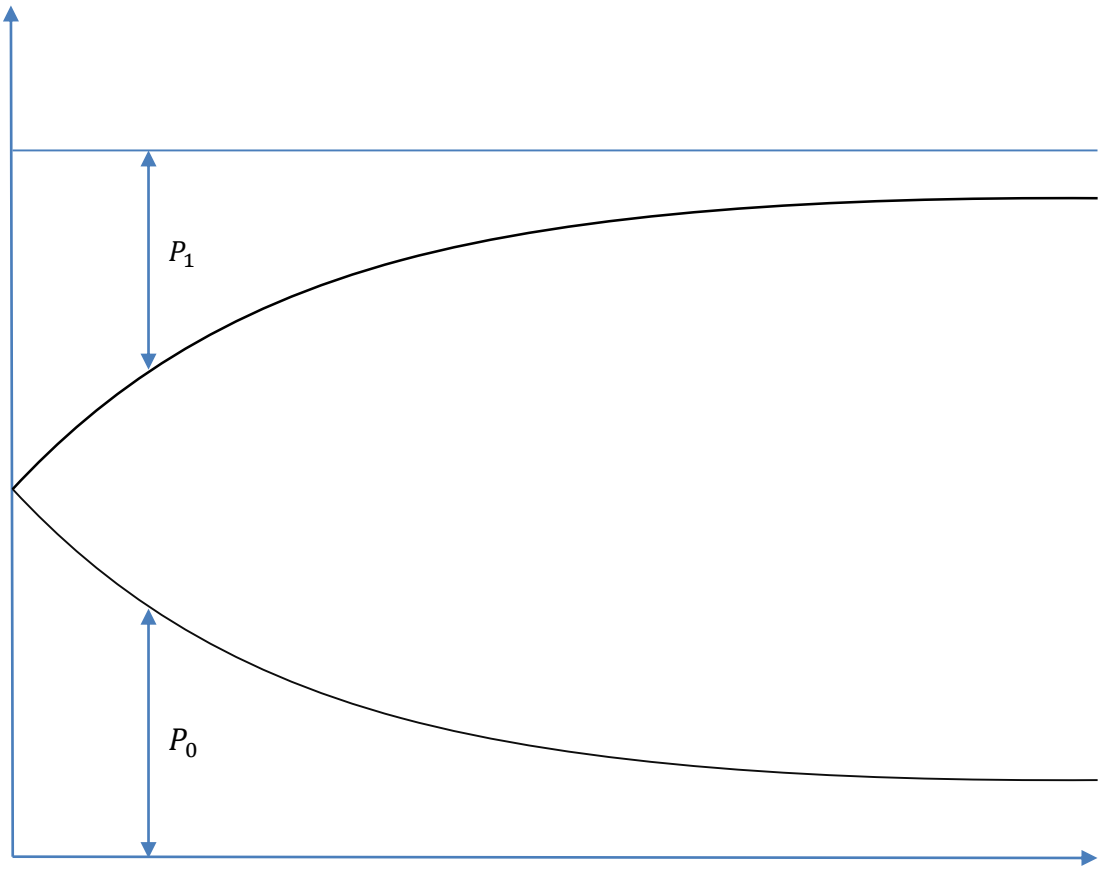
ანალოგიურად, სწორი გადაწყვეტილების მიღების ალბათობა

$$P_1 = P\left[2k_i\alpha + \varepsilon_1{}^{(i)} - \varepsilon_2{}^{(i)} > \delta\right] = P[\xi_i > \delta - 2k_i\alpha] = \frac{1}{2}\left[1 - \Phi\left(\frac{\sigma + 2k_i\alpha}{2\sigma}\right)\right]$$

გაჩერების ალბათობა კი იქნება

$$P_{stop} = \Phi\left(\frac{\delta + 2k_i\alpha}{2\sigma}\right) + \Phi\left(\frac{\delta - 2k_i\alpha}{2\sigma}\right)$$

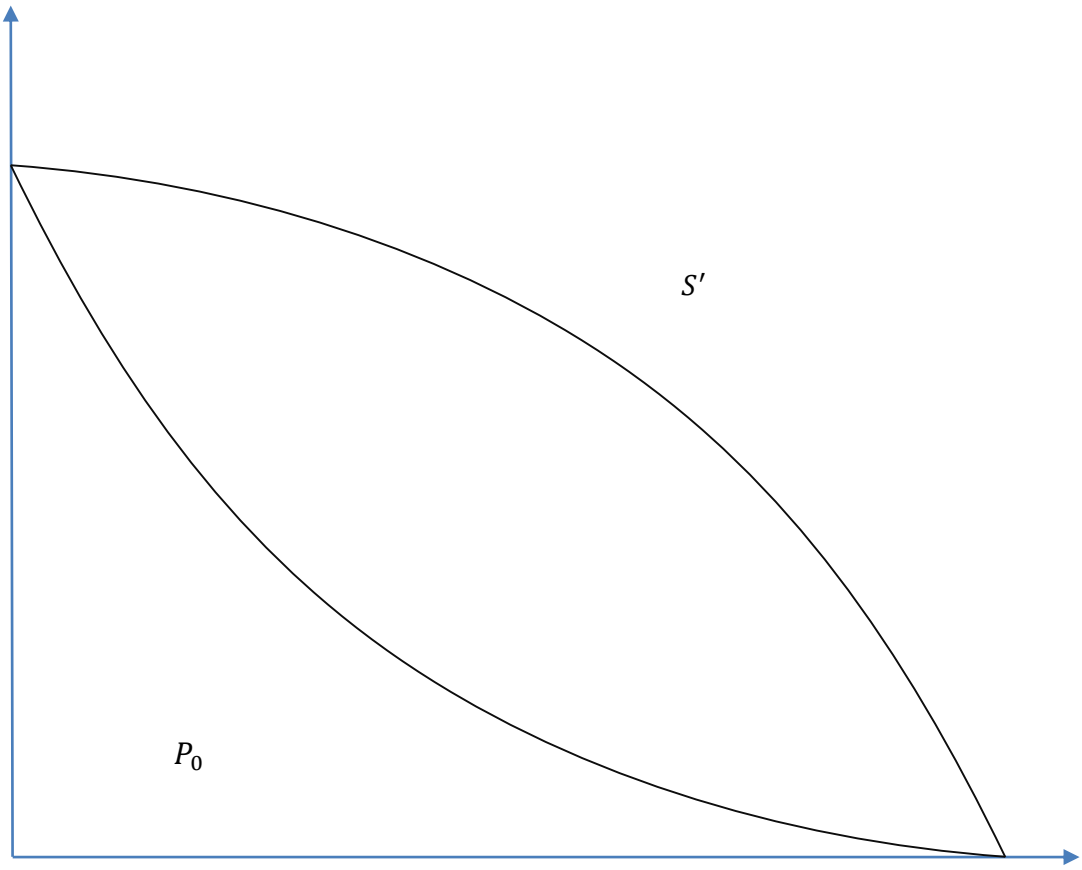
აღნიშნული სიდიდეების ყოფაქცევა ნაჩვენებია შემდეგ ნახაზზე



ნახ.3

როგორც ვხედავთ ზღურბლის ზრდასთან ერთად მცირდება შეცდომის ალბათობა. მაგრამ ამ დროს მცირდება აგრეთვე სწორი გადაწყვეტილების მიღების ალბათობაც, რაც ხდება გაჩერების ალბათობის გაზრდით . ე.ი. სისტემა უფრო დიდ ხანს არის გაჩერებული, ე.ი. $\Delta x = 0$. ამიტომ ზღურბლური ფილტრაციის შემოღების ეფექტურობა უნდა შევაფასოთ მხოლოდ მუშა ბიჯების ალბათობის მიხედვით, ე.ი. $\Delta x \neq 0$, მაშინ შეცდომის ფარდობითი ალბათობა იქნება

$$P_0{}^f = \frac{P_0}{P_0 + P_1}$$



ნახ.4

როგორც ვხედავთ სისტემაში შეცდომები მცირდება არა მარტო აბსოლუტურად, არამედ შეფარდებითადაც. სწორედ ეს გარემოება ამართლებს ზღურბლური ფილტრაციის გამოყენებას.

ახლა განვიხილოთ ძებნის სწრაფქმედება ზღურბლური ფილტრაციის დროს. სწრაფქმედებას შევაფასებთ ძებნის ერთი ეტაპის განმავლობაში მიზნისკენ საშუალო გადახრის სიდიდის მიხედვით.

$x < x^*$ მდგომარეობიდან მიზნის x^* მდგომარეობისკენ გადახრის ფორმულა შეიძლება ჩავწეროთ შემდეგი სახით:

$$S_i = \begin{cases} b & P_1 \\ 0 & P_{stop} \\ -b & P_0 \end{cases}$$

საშუალო გადახრა იქნება

$$\overline{S}_i = \frac{b}{2}\left[\Phi\left(\frac{\delta + 2k_i\alpha}{2\sigma}\right) + \Phi\left(\frac{\delta - 2k_i\alpha}{2\sigma}\right)\right]$$

ნახ. 3-ზე ნაჩვენებია, რომ ზღურბლის ზრდასთან ერთად საშუალო გადახრა მონოტონურად კლებულობს, რაც უნდა მომხდარიყო.

განვიხილოთ საშუალო გადახრის დამოკიდებულება $\frac{\sigma}{ka}$ ხმაურზე. ეს დამოკიდებულება ნაჩვენებია ნახაზებზე ზღურბლის სხვადასხვა მნიშვნელობებისთვის, როცა $\delta < 2ka$, დამოკიდებულებას $\overline{S}(\frac{\sigma}{ka})$ აქვს მაქსიმუმი. ეს ნიშნავს, რომ ხმაურის დაბალი დონის დროს სწრაფქმედების გაზრდისთვის საჭიროა გავზარდოთ ხმაური სისტემაში ან შევამციროთ ka სიდიდე. ეს ერთი შეხედვით პარადოქსული შედეგი აიხსნება საკმარისად მარტივად. მართლაც, აღნიშნულ მოვლენას ადგილი აქვს მხოლოდ დიდი $\delta < 2ka$ ზღურბლისთვის, რომელიც აღემატება სასარგებლო სიგნალს. ასეთი ზღურბლი ხმაურის გამორიცხვის შემთხვევაში საერთოდ არ გზავნის სიგნალს და ძებნა წყდება.

ამ შემთხვევაში ხმაური, რომელიც ედება სასარგებლო $2ka$ სიგნალს. ქმნის შესაძლებლობას გაატაროს სიგნალი მაღალი ზღურბლის ქვეშ, რაც განაახლებს ძებნას.

ამრიგად, აღნიშნულიდან სრულებითაც არ გამომდინარეობს ის, რომ გაიზარდოს ხმაურის დონე. ეს მხოლოდ იმას ნიშნავს, რომ δ ზღურბლი არჩეულია არასწორად - ის ძალიან დიდია. ამიტომ საკმარისია შევამციროთ ზღურბლი, რომ გაიზრდება სწრაფქმედება.

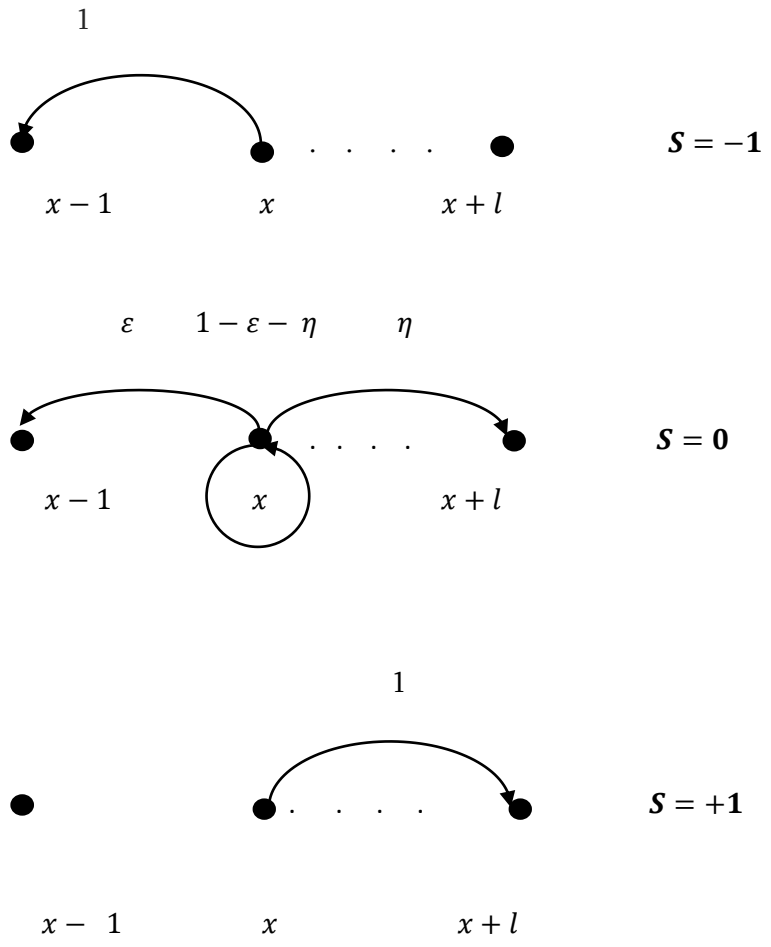
შემთხვევითი ხელშეშლის (ხმაურის) პირობებში ძებნის დრო ბუნებრივია იზრდება, რადგან სწორი მიმართულებით შესრულებული მუშა ბიჯების გარდა, სისტემა ასრულებს მუშა ბიჯებს არასასურველი მიმართულებით (მცდარი ბიჯები).

ოპტიმიზაციის პროცესის ავტომატური მოდელი

ვთქვათ $T_{2n,2}^{(x)}(\varepsilon,\eta)$ $(0 \le \varepsilon,\eta \le 1,\ 0 \le \varepsilon + \eta \le 1)$ სასრული სტოქასტური ავტომატი $L^{(n)} = L_1^{(n)} \cup L_2^{(n)} = \{\pm 1, \pm 2, \dots, \pm n\}$ შიგა მდგომარეობებითა და ორი $F_2 = \{f_1, f_2\}$ მოქმედებით ფუნქციონირებს სტაციონარულ შემთხვევით $C(a_1, r_1; a_2, r_2)$ გარემოში. ავტომატი იმ მდგომარეობებში ყოფნისას, რომელთა ნომერია $x = \{-\mathrm{n}, \dots -2, -1\}$ ასრულებს პირველ მოქმედებას, ხოლო იმ მდგომარეობებში ყოფნისას, რომელთა ნომერია $x = \{1,2, \dots, n\}$ -

მეორეს. გარკვეულობისათვის ვიგულისხმოთ, რომ $a_1 > a_2$, ე.ი. ავტომატის პირველი მოქმედება უკეთესია, ვიდრე მეორე.

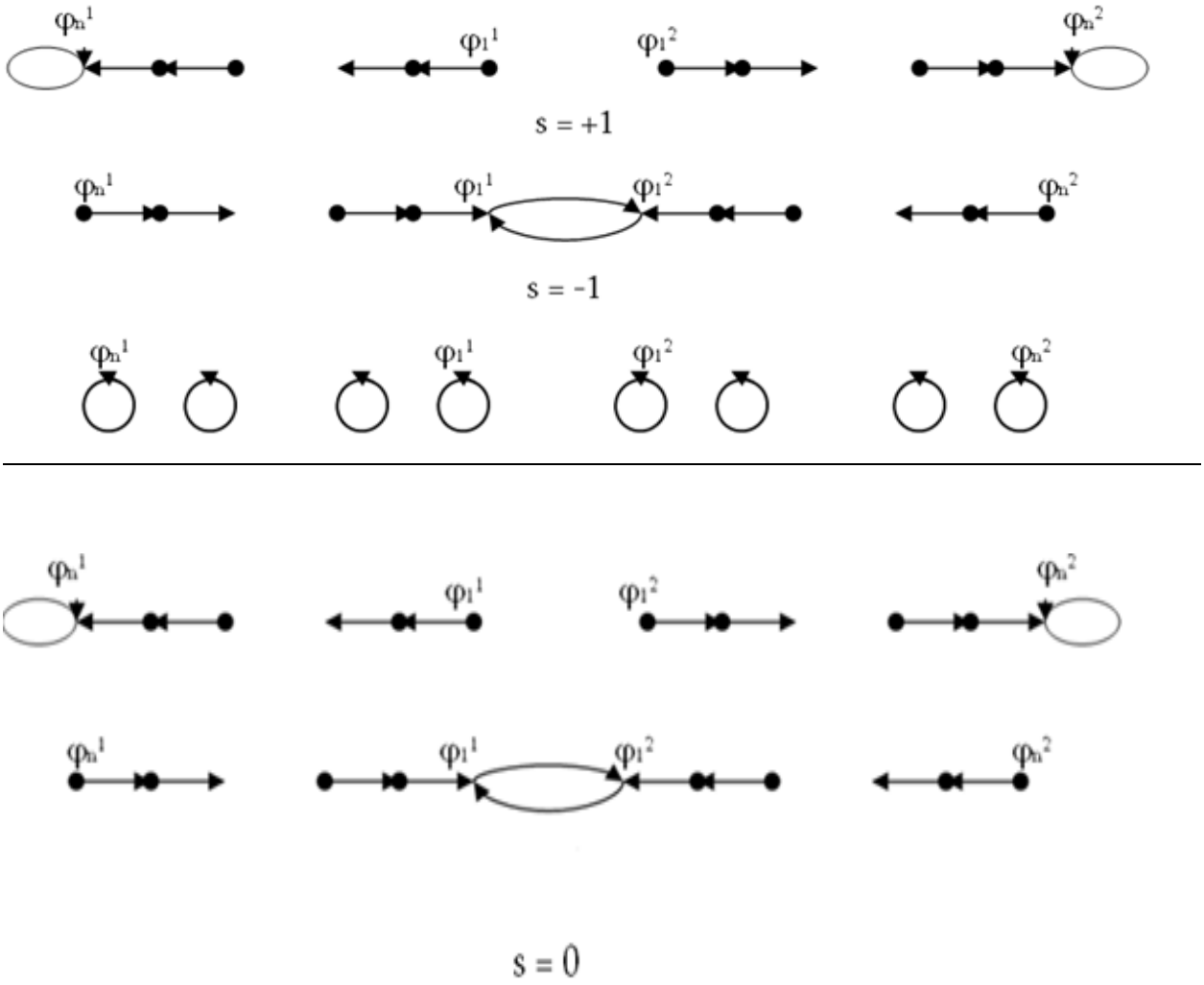
სასრული სტოქასტური $T_{2n,2}^{(x)}(\varepsilon,\eta)$ ავტომატის მდგომარეობებს შორის გადასვლა განვსაზღვროთ შემდეგნაირად: $s=+1$ სიგნალის შემთხვევაში $x=i$ და $x=-i$ $(i=1,2,\dots,n-1)$ მდგომარეობები შესაბამისად გადადიან $x=i+l$ და $x=-i-l$ მდგომარეობებში, ხოლო $|x|=n$ მდგომარეობები თავის თავში; $s=-1$ სიგნალის შემთხვევაში $x=i$ და $x=-i$ $(i=2,\dots,n)$ მდგომარეობები შესაბამისად გადადიან $x=i-1$ და $x=-i+1$ მდგომარეობებში; $x=1$ მდგომარეობა გადადის $x=-i$ მდგომარეობაში, ხოლო $x=-1$ მდგომარეობა - $x=i$ მდგომარეობაში ; $s=0$ სიგნალის შემთხვევაში ყველა $x=i$ $(i=\pm1,\pm2,\dots,\pm n)$ მდგომარეობა $1-\varepsilon-\eta$ ალბათობით რჩება თავის თავში, ხოლო ε და η ალბათობებით მდგომარეობებს შორის გადასვლა შესაბამისად განისაზღვრება ისევე, როგორც $s=-1$ და $s=+1$ სიგნალის შემთხვევაში (ნახ.5).



ნახ.5. სასრული სტოქასტური $T_{2n,2}^{(x)}(\varepsilon,\eta)$ ავტომატის მდგომარეობებს შორის გადასვლის გრაფის ფრაგმენტი

მაშასადამე, სასრული სტოქასტური $T_{2n,2}^{(x)}(\varepsilon,\eta)$ ავტომატი ერთშესასვლელიანი და ერთგასასვლელიანია: თითოეულ მოქმედებაში შემავალ მდგომარეობას წარმოადგენს ნებისმიერი ერთი x მდგომარეობა, ხოლო გამოსასვლელია $|x|=1$ მდგომარეობა.

სასრული სტოქასტური $T_{2n,2}^{(x)}(\varepsilon,\eta)$ ავტომატის კერძო სახეა წრფივი ტაქტიკის სასრული $T_{2n,2}^{(1)}(0,0)$, $T_{2n,2}^{(1)}(0,1)$ და $T_{2n,2}^{(1)}(1,0)$ დეტერმინირებული ავტომატები, რომელთათვისაც მდგომარეობებს შორის გადასვლის გრაფები მოცემულია ნახ. 6-ზე.



ნახ.6. წრფივი ტაქტიკის სასრული $T_{2n,2}^{(1)}(0,0)$, $T_{2n,2}^{(1)}(0,1)$ და $T_{2n,2}^{(1)}(1,0)$ დეტერმინირებული ავტომატების მდგომარეობებს შორის გადასვლის გრაფები

როგორც აღვნიშნეთ, ჩვენს მიერ განხილულ ექსტრემუმის ძებნის ბიჯურ ალგორითმში მმართველ რგოლს წარმოადგენს სასრული ავტომატი $T_{2n,2}^{(x)}(\varepsilon,\eta)$.

შესასვლელზე ავტომატს მიეწოდება $\Delta Q_i = Q_i - Q_{i-1}$ - ის ნიშანი, ხოლო გამოსასვლელზე იქნება i-ური პარამეტრის ცვლილების ნიშანი: $\Delta x_i = \alpha_i a_i$, სადაც a_i = const > 0 i - ური ბიჯის მოდულია, ხოლო $\alpha_i = \pm 1$ - ამ ბიჯის ნიშანი.

ავტომატის მდგომარეობათა რაოდენობაა $2n$. პირველ n მდგომარეობაში ბიჯი კეთდება დადებითი მიმართულებით: $\alpha_i = +1$, ხოლო დანარჩენ n მდგომარეობაში კი უარყოფითი მიმართულებით: $\alpha_i = -1$.

ექსტრემუმის ძებნის ბიჯურ ალგორითმში ავტომატთან ერთად ზღურბლის შემოტანა ავტომატს გახდის სამსიგნალიანს, ე.ი. გაჩნდება ე.წ. ნეიტრალური რეაქციების კლასი, რომლის დროსაც $S(t) = 0$. ასეთი სიგნალის მიღების დროს ავტომატი დარჩება იგივე მდგომარეობაში, რომელშიც იმყოფებოდა.

სასრული ავტომატის მდგომარეობათა ცვლილების ალგორითმი მდგომარეობს შემდეგში:

როდესაც $S < 0$ $(\Delta Q \le \delta)$, მაშინ გვაქვს

$$\varphi_j^{(t+1)} = \begin{cases} \varphi_j^{(t)},\ j^{(t)} = 1 \\ \varphi_j^{(t)} - 1, j^{(t)} \le n \\ \varphi_j^{(t)} + 1, j^{(t)} > n \\ \varphi_j^{(t)}, j^{(t)} = 2n \end{cases}$$

აქ $f^{(t)}$ ნომერია მდგომარეობის t-ურ ტაქტზე.

როცა $S > 0$ $(\Delta Q \ge \delta)$, ალგორითმს აქვს სახე

$$\varphi_j^{(t+1)} = \begin{cases} \varphi_j^{(t)} + 1, & j^{(t)} \le n \\ \varphi_j^{(t)} - 1, & j^{(t)} > n \end{cases}$$

ხოლო, როცა $S = 0$ $(-\delta < \Delta Q < \delta)$, გვაქვს

$$\varphi^{(t+1)} = \varphi^{(t)}$$

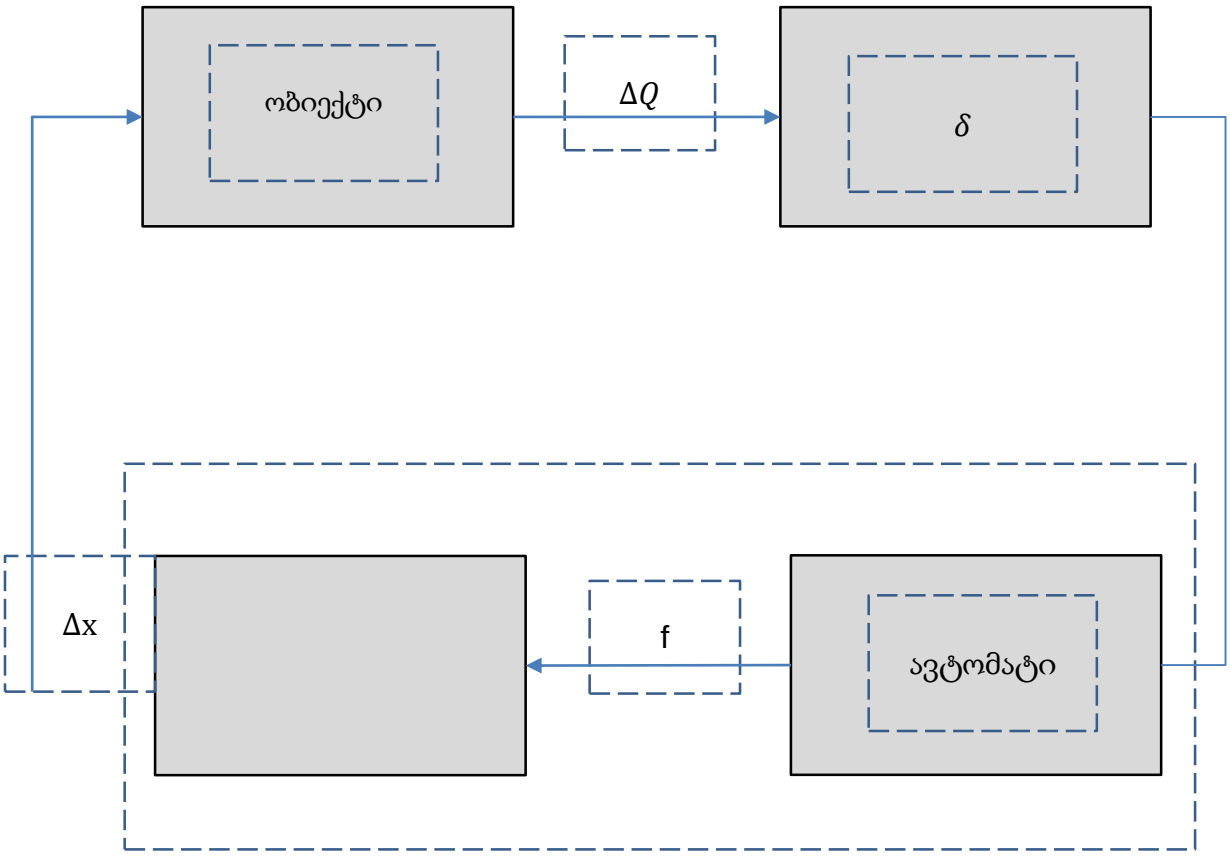
ნებისმიერი j^t-თვის.

ავტომატის გამომავალი სიგნალი მნიშვნელობებს ღებულობს ორელემენტიანი სიმრავლიდან $f \in \{+1, -1\}$

ამასთან:

$$\phi_{i=} \begin{cases} +1 & \varphi = \varphi_j\ (j = 1, \dots, n) \\ -1 & \varphi = \varphi_j\ (j = n+1,\ \dots,\ 2n) \end{cases}$$

ავტომატისა და ზღურბლური ფილტრაციის გამოყენებით ოპტიმიზატორის ბლოკ-სქემა შეიძლება წარმოვადგინოთ შემდეგი სახით:



სადაც $\Delta x_i = f^t \cdot a$, t მიუთითებს t-ურ ტაქტზე, a არის ბიჯის სიდიდე. ადვილი დასანახია, რომ როდესაც $n = 1$, ექსტრემუმის ძებნის ბიჯური ალგორითმი წარმოადგენს ჩვეულებრივ რეგულარულ ბიჯურ ალგორითმს.

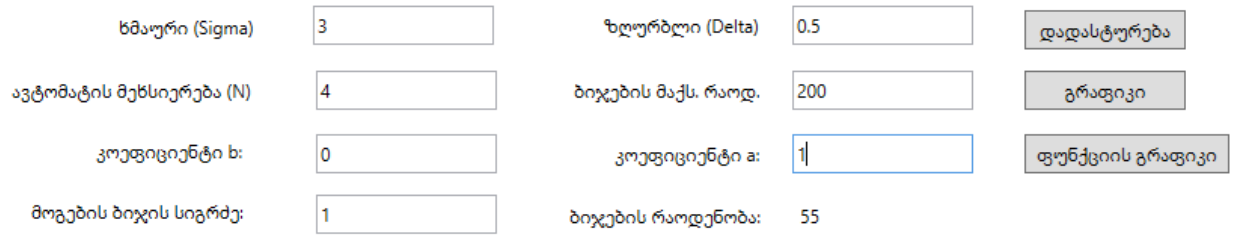
ამოცანის პროგრამული რეალიზაცია

ქვემოთ აღწერილია ძებნის ავტომატური მოდელის ალგორითმის პროგრამული რეალიზაცია, რომელიც ითვლის და გვაჩვენებს ამ ალგორითმის ბიჯებს მუშაობის პროცესში. პროგრამაში გამოყენებულია ფუნქცია f(x) = -a|x+b|.

პროგრამას აქვს ძებნისთვის აუცილებელი პარამეტრების შეყვანის საშუალება, რომელთა გამოყენებითაც ცდილობს მივიდეს ექსტრემუმის წერტილამდე. პროგრამამ შეიძლება

იპოვოს ამონახსნი ან შეიძლება ვერც იპოვოს, ეს დამოკიდებულია შემავალ პარამტრებზე და ხმაურის დონეზე.

პროგრამას აქვს შემავალი პარამეტრები - ხმაური(Sigma), ზღურბლი(Delta), ავტომატის მეხსიერება(N), ბიჯების მაქსიმალური რაოდენობა, კოეფიციენტი b და კოეფიციენტი a.

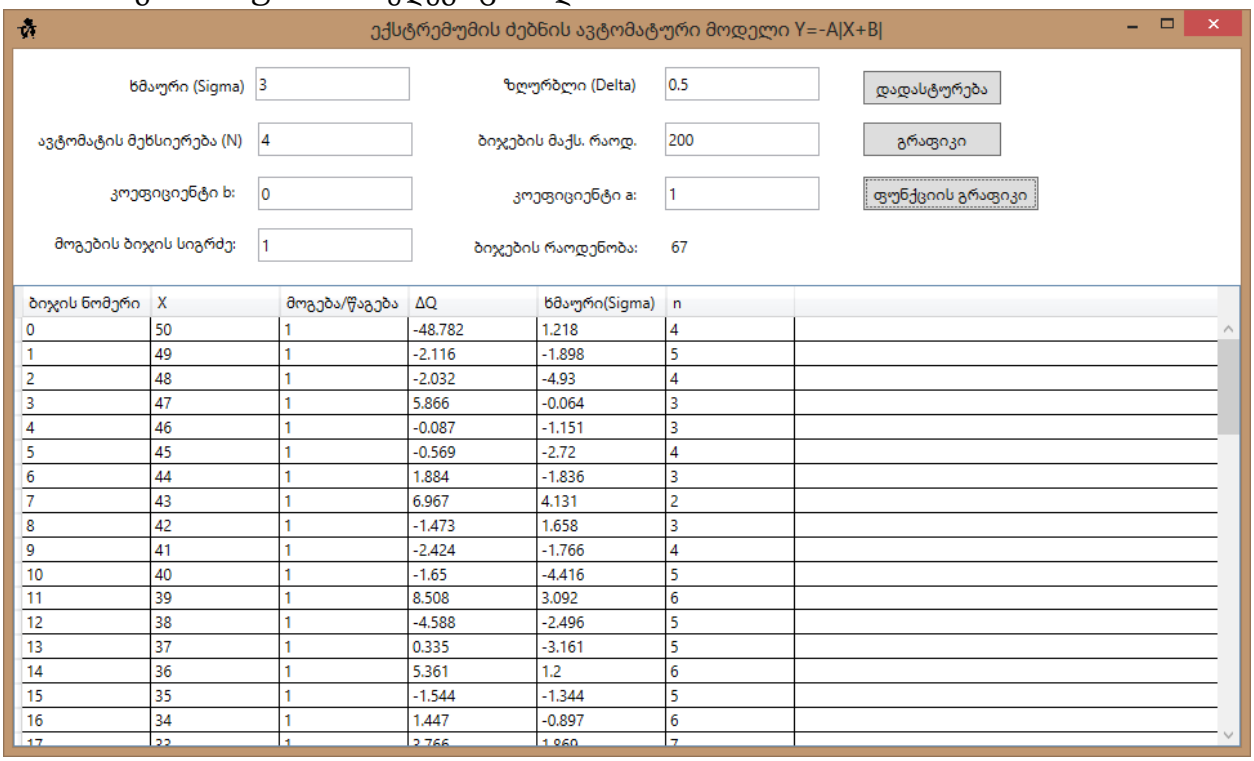


პროგრამა ინახავს ყოველ ბიჯზე ავტომატის მიმდინარე მდგომარეობის ინფორმაციას და შემდეგ აჩვნებს ცხრილის სახით:

- ბიჯის ნომერი
- X ცვლადის მნიშვნელობა
- მოგება/წაგება(ავტომატის მუშაობის შედეგი)
- ΔQ - ხარისხის მაჩვენებლის ნაზრდი
- ხმაური კავშირის არხში
- N

ყოველ ბიჯზე ავტომატი აგენერირებს შემთხვევით რიცხვს [0, 1] შულედში რომელსაც შემდეგ ვიყენებ შემთხვევითი ხმაურის მისაღებად.

პროგრამის მუშაობის შედეგი ცხრილის სახით:



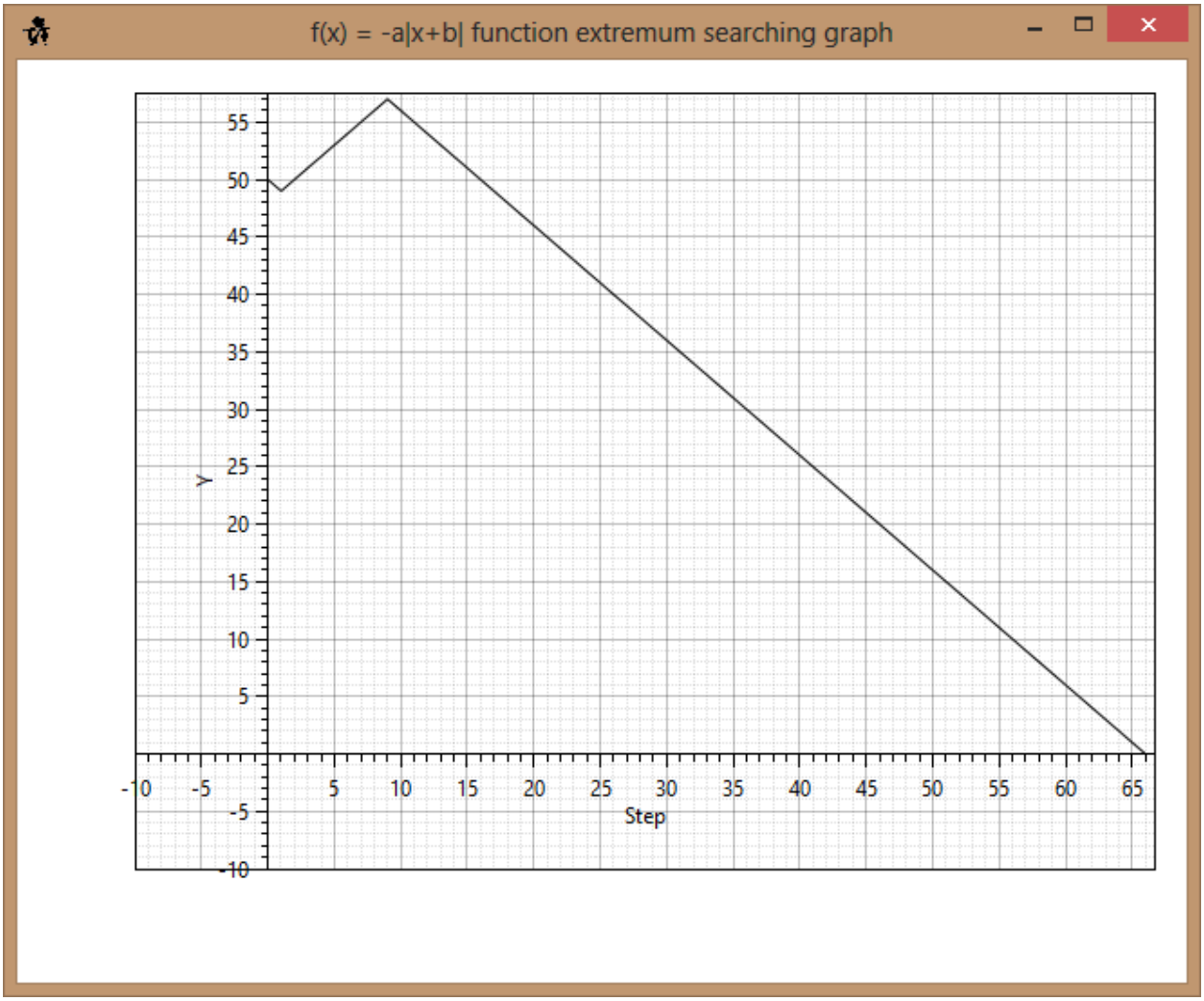
ბიჯის ნომერი	X	მოგება/წაგება	ΔQ	ხმაური(Sigma)	n
0	50	1	-48.782	1.218	4
1	49	1	-2.116	-1.898	5
2	48	1	-2.032	-4.93	4
3	47	1	5.866	-0.064	3
4	46	1	-0.087	-1.151	3
5	45	1	-0.569	-2.72	4
6	44	1	1.884	-1.836	3
7	43	1	6.967	4.131	2
8	42	1	-1.473	1.658	3
9	41	1	-2.424	-1.766	4
10	40	1	-1.65	-4.416	5
11	39	1	8.508	3.092	6
12	38	1	-4.588	-2.496	5
13	37	1	0.335	-3.161	5
14	36	1	5.361	1.2	6
15	35	1	-1.544	-1.344	5
16	34	1	1.447	-0.897	6
17	33	1	3.766	1.869	7

		Delta=0.0	Delta=0.5	Delta=1.0	Delta=1.5	Delta=2.0
Sigma	N					
1	1	116	92	75	67	65
1	2	60	55	54	55	58
1	3	54	54	55	55	57
1	4	54	54	54	54	58
2	1	194	188	173	147	132
2	2	93	75	66	62	59
2	3	65	58	57	56	59
2	4	63	57	56	57	57
3	1	199	199	197	196	188
3	2	123	113	95	86	79
3	3	82	74	70	65	61
3	4	69	68	62	63	61
4	1	200	200	200	199	197
4	2	142	134	125	115	101
4	3	101	85	81	79	71
4	4	86	77	75	68	66

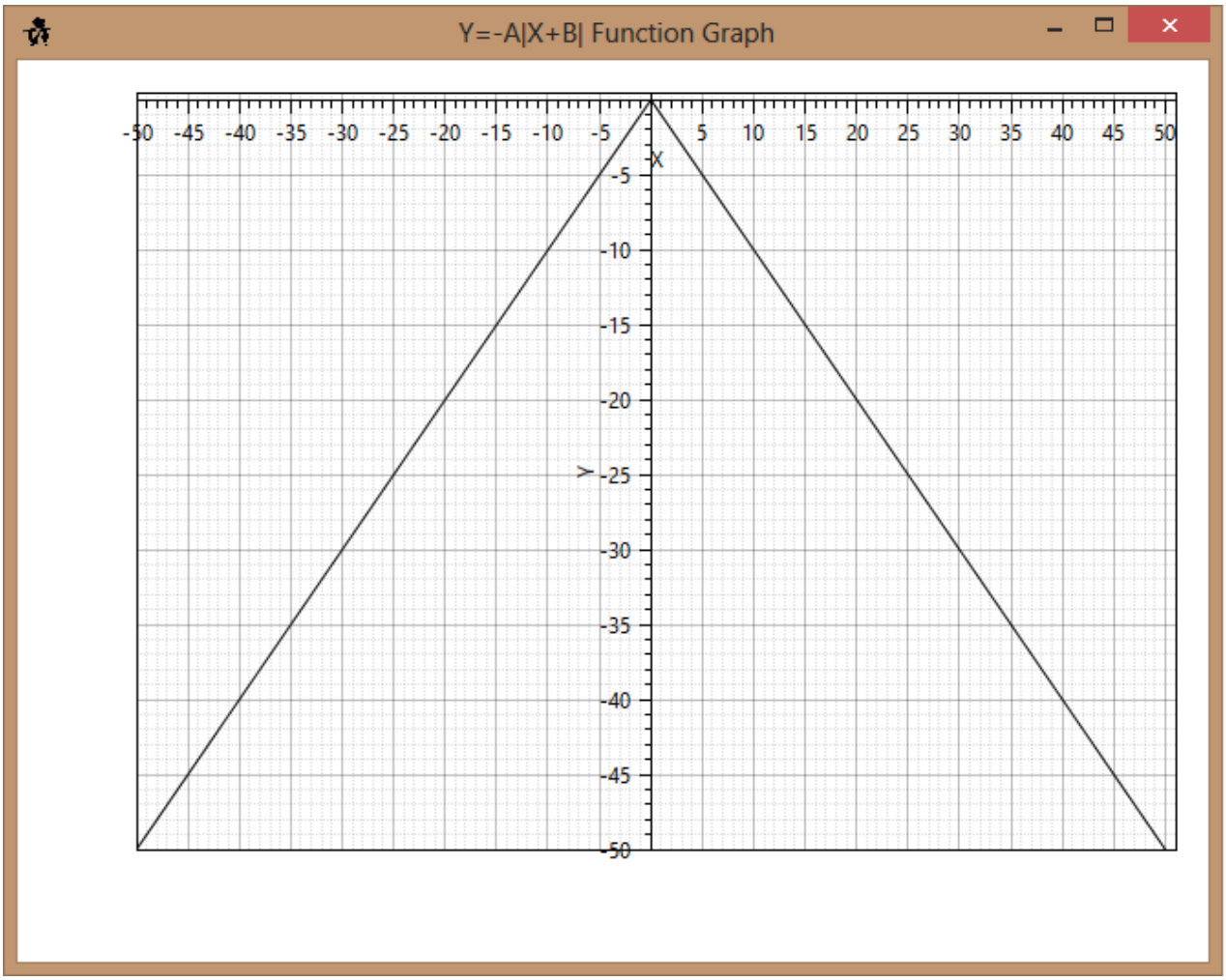
		Delta=2.5	Delta=3.0	Delta=3.5	Delta=4.0	Delta=4.5
Sigma	N					
1	1	72	73	95	144	171
1	2	62	76	100	142	164
1	3	62	73	102	141	176
1	4	63	74	95	132	160
2	1	117	110	101	100	102
2	2	61	61	65	66	75
2	3	59	59	62	64	75
2	4	58	60	61	66	69

3	1	176	167	161	150	146
3	2	73	71	69	71	70
3	3	60	60	61	63	67
3	4	60	61	60	64	70
4	1	199	190	191	189	179
4	2	97	87	83	79	85
4	3	68	66	62	65	69
4	4	70	64	64	72	63

პროგრამას ასევე აქვს ცხრილის მიხედვით გრაფიკის აგების ფუნქციონალი:



„ფუნქციის გრაფიკი“ ღილაკზე დაჭერით ასევე იხაზება გრაფიკი, რომელიც აიგება a და b პარამეტრების გათვალისწინებით.



დასკვნა

როგორც ავღნიშნეთ, ჩვენს მიერ განხილული ექსტრემუმის ძებნის ბიჯური ალგორითმში მმართველ რგოლს წარმოადგენს სასრული ავტომატი, რომელიც სამი კლასის რეაქციის მქონე სტაციონალურ შემთხვევით გარემოში ფლობს მიზანშეწონილ ქცევას. მანქანური ექსპერიმენტების შედეგად შესაძლებელია შემდეგი დასკვნების გამოტანა:

ა) ექსტრემუმის ძებნის ჩვეულებრივი ბიჯური ალგორითმისაგან განსხვავებით ძებნის ავტომატური მოდელის საშუალებით უფრო სწრაფად ხდება ექსტრემუმში მიღწევა;

ბ) ვინაიდან ხმაურის პირობებში ზღურბლური ფილტრაცია წარმოადგენს ძებნის ბიჯური ალგორითმის გაუმჯობესებულ მეთოდს, შესაბამისად ექსტრემუმის ძებნის ავტომატურ მოდელში მმართველი ორგანოს როლში ისეთი სასრული ავტომატების გამოყენება, რომლებიც სამი კლასის რეაქციების მქონე ტერნარულ სტაციონარულ შემთხვევით გარემოში ხასიათდებიან ქცევის მიზანშეწონილობით, უფრო ეფექტურია, ვიდრე ბინარულ გარემოში იმავე კლასის მიზანშეწონილი ქცევის ავტომატებისა.

გ) ზღურბლური ფილტრაციის განხილულ ალგორითმში მნშვნელოვან როლს ასრულებს როგორც ხმაურის დონე, ასევე ავტომატის მეხსიერება და ზღურბლის სიდიდე: ხმაურის ყოველი დონის შემთხვევაში არსებობს ავტომატის მეხსიერებისა და ზღურბლის სიდიდის ოპტიმალური მნიშვნელობები.

გამოყენებული ლიტერატურა

1. Л. А. Растригин. Системы экстремального управления. Рига, Зинатне, 1974.

2. Цетлин М. Л. Исследование по теории автоматов и моделированию биологических систем. М., Наука, 1969.

3. Королюк В. С., Плетнев А. И., Эйдельман С. Д. Автоматы. Блуждания. Игры. Успехи математических наук. Т. 43, вып. 1(259), 1988.

4. Хведелидзе Т. Д. Об одной конструкции конечного автомата в стационарной случайной среде с тремя классами реакций. ქ.ე.ს.ჟ. კომპიუტერული მეცნიერებები და ტელეკომუნიკაციები http//gesj. Internet-academy. Org.ge N2(34), 2012.

დანართი

ალგორითმის რეალიზაციის პროგრამული კოდის ნაწილი

```
using System;
using System.Collections.Generic;
using System.Windows.Input;
using WPF.Extremum.Extentions;

namespace WPF.Extremum
{
    /// <summary>
    /// Initial paremeters model for Y = -a|x+b| function
```

```
/// </summary>
public class Automate
{
    /// <summary>
    /// Automate's memory
    /// </summary>
    public int N { get; set; }
    /// <summary>
    /// Steps maximum number in algorithm
    /// </summary>
    public int MaxStepNum { get; set; }
    /// <summary>
    /// Algorithm initial direction
    /// </summary>
    public int Direction { get; set; }
    /// <summary>
    /// Coefficient of function y=-a|x+b|
    /// </summary>
    public int FuncCoefficientA { get; set; }

    public int FuncCoefficientB { get; set; }

    /// <summary>
    /// Automate's noise
    /// </summary>
    public double Sigma { get; set; }
```

```
        public bool HasRandomSigma { get; set; }

        /// <summary>
        /// Searching algorithm initial value
        /// </summary>
        public int X { get; set; }
        /// <summary>
        /// Limit of quality sign
        /// </summary>
        public double Delta { get; set; }

        public int WinStepLength { get; set; }

        public Automate()
        {
            X = 50;
            MaxStepNum = 500;
            Direction = 1;
            HasRandomSigma = true;
        }

        public void FindExtremum()
        {
            StaticStorage.OutputData = new
List<OutputData>();

            var randomGenerator = new Random();
```

```
            var localSigma =
CalculateSigma(randomGenerator);
            var q = CalculateDeltaQ(localSigma);
            var deltaQ = q;

            var y = this.X;
            var n = N;//This must be 1

            StaticStorage.OutputData.Add(new OutputData
            {
                Step = 0,
                Y = y,
                X = X,
                Direction = Direction,
                InternalState = n,
                DeltaQ = deltaQ,
                N = n,
                Sigma = localSigma
            });

            for (int step = 1; step < MaxStepNum; step++)
            {
                X--;

                //ΔQ=Q_i-Q_(i-1) ავტომატის შემავალი სიგნალი
- S
                double qTemp = q;
```

```
                localSigma =
CalculateSigma(randomGenerator);
                q = CalculateDeltaQ(localSigma);
                deltaQ = q - qTemp;
                deltaQ = Math.Round(deltaQ, 3);

                //Win ____ ΔQ > δ
                if (deltaQ >= Delta)
                {
                    if (1 < n && n <= N)
                        n--;
                    else if (N + 1 <= n && n < 2 * N)
                        n++;
                }
                //Loose ____ ΔQ < -δ
                else if (deltaQ <= -Delta)
                {
                    if (1 <= n && n <= N)
                        n++;
                    else if (N + 1 <= n && n <= 2 * N)
                        n--;
                }
                //Neutral ____ -δ < ΔQ < δ
                else
                {
                    n = n;
                }
```

```
                if (n <= N) y++;
                else y -= WinStepLength;
                //y = (y < 0) ? 0 : y;

                StaticStorage.OutputData.Add(new OutputData
                {
                    Step = step,
                    Y = y,
                    X = X,
                    Direction = Direction,
                    InternalState = n,
                    DeltaQ = deltaQ,
                    N = n,
                    Sigma = localSigma
                });

                if (y <= 0)
                    break;
            }
        }

        private double CalculateSigma(Random
randomGenerator)
        {
            double localSigma = Sigma;
            if (HasRandomSigma)
```

```
                localSigma = Sigma *
randomGenerator.NextGaussian();

            localSigma = Math.Round(localSigma, 3);
            return localSigma;
        }

        public double MathFunction()
        {
            //f(x) = -a|x+b|
            return -this.FuncCoefficientA * Math.Abs(X +
FuncCoefficientB);
        }

        public double CalculateDeltaQ(double sigma)
        {
            var deltaQ = MathFunction() + sigma;
            return Math.Round(deltaQ, 3);
        }
    }
}
```