ივანე ჯავახიშვილის სახელობის თბილისის
სახელმწიფო უნივერსიტეტი

ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტი

კომპიუტერულ მეცნიერებათა დეპარტამენტი

ლექციათა კურსი საგანში

ინტელექტუალური სისტემები

ასოცირებული პროფესორი **ზ. ქოჩლაძე**

2014

ლექცია 1. ხელოვნური ინტელექტი

1.1. ინტელექტი და ინტელექტუალური სისტემები. ინტელექტუალური სისტემები, ინტელექტუალური გამოთვლები, ინტელექტუალური ქცევა, ინტელექტუალური რობოტები. ეს ტერმინები დღეს ხშირად გამოიყენება სამეცნიერო ლიტერატურაში. რა იგულისხმება ამ ტერმინების ქვეშ, როდის შეიძლება ჩავთვალოთ, რომ რობოტის, ან კომპიუტერული სისტემის ქცევა ინტელექტუალურია? ამ კითხვაზე პასუხი შეიძლება იყოს ძალიან მარტივიც და ძალიან რთულიც. ძალიან მარტივი პასუხი ასეთია: სისტემა არის ინტელექტუალური, თუ მას შეუძლია შეასრულოს სამუშაო, რომლის შესრულებაც ადამიანის მიერ ჩაითვლებოდა ინტელექტუალურ ქცევად. განვიხილოთ ერთი მაგალითი. არსებობენ პროგრამული მართვის ჩარხები, რომლებსაც შეუძლიათ ადამიანის ზედამხედველობის გარეშე, მიკრონის სიზუსტით გამოჩარხონ ნაკეთობა. ამას უზრუნველყოფს პროგრამა, რომელიც მართავს ჩარხის მუშაობას. ერთი შეხედვით, საქმე გვაქვს მართლაც ინტელექტუალურ სისტემასთან, რადგანაც ამ სამუშაოს შესრულება ადამიანის მიერ მართლაც ინტელექტუალური ქცევაა. უფრო მეტიც, ადამიანი ალბათ ვერ შეძლებდა ასეთი სიზუსტით მუშაობას. მაგრამ აქ არის ერთი პატარა პრობლემა, საკმარისია დასამუშავებელი დეტალი დააყენოთ ჩარხში არაზუსტად, ან პროგრამამ დაუშვას სულ პატარა შეცდომა და გამოჩარხული დეტალი გადასაგდები გახდება. სისტემას არა აქვს კავშირი გარემოსთან და არ შეუძლია გარემოდან მიღებული ინფორმაციის შესაბამისად შეცვალოს თავისი ქცევა, ანუ ***სისტემას არ შეუძლია შეეგუოს გარემოს ცვლილებებს.***

განვიხილოთ კიდევ ერთი მაგალითი. დღეს არსებობს უამრავი, სხვადასხვა დონის ჭადრაკის მოთამაშე პროგრამა, არიან თუ არა ისინი ინტელექტუალური სისტემები? ჩავატაროთ ასეთი ცდა. ვეთამაშოთ ასეთ პროგრამას ერთი და იგივე პარტია ძალიან ბევრჯერ და დავუშვათ, ყველა პარტია მთავრდება ადამიანის სასარგებლოდ. თუ პროგრამა ასეთი ექსპერიმენტის ბოლოსაც კი არ ცვლის თავის თამაშს, ის არ შეიძლება იყოს ინტელექტუალური, რადგანაც მას არ გააჩნია ***დასწავლის უნარი,*** ეს კი ინტელექტუალური ქცევის ერთერთი მნიშვნელოვანი პარამეტრია. ადამიანი, რომელსაც ნამდვილად გააჩნია ინტელექტუალური ქცევა, ყოველთვის სწავლობს თავის (უფრო მაღალი ინტელექტის მქონე კი - სხვის) შეცდომებზე.

ეს მაგალითები გვიჩვენებენ, რომ ინტელექტუალურობის ისეთი განსაზღვრა, რომელიც საშუალება მოგვცემს შევაფასოთ შეესაბამება თუ არა მოცემული სისტემის ქცევა ინტელექტუალური, ძალიან რთულია. ყველა ამ პრობლემას, დაკავშირებულს ინტელექტთან და ინტელექტუალურ ქცევასთან შეისწავლის ***ხელოვნური ინტელექტის თეორია.***

1.2. ხელოვნური ინტელექტის თეორია. ადამიანები იმით გამოირჩევიან სხვა ცოცხალი არსებებისაგან, რომ მათ შეუძლიათ ***აზროვნება.*** უკვე მრავალი საუკუნეა ადამიანი ცდილობს ამოხსნას აზროვნების ფენომენი, გაიგოს თუ როგორ ახერხებს ის გარე სამყაროს შეცნობას გრძნობათა ორგანოების საშუალებით, უკვე ერთხელ შეხვედრილი სიტუაციების დამახსოვრებას, სწავლას მათ საფუძველზე, თავისი ცოდნის განზოგადებას, ლოგიკური მსჯელობის საფუძველზე სწორი გადაწყვეტილებების მიღებას და თავის მოთხოვნილებების დასაკმაყოფილებლად გარე სამყაროს მართვასა და შეცვლას. სწორედ ასეთ ქმედებათა ერთობლიობა ქმნის ადამიანის ისეთ ქცევას, რომელსაც ჩვენ ვუწოდებთ ინტელექტუალურ ქცევას.

როგორც ვიცით, ყველა ამ პრობლემას შეისწავლის ფსიქოლოგია, მაშ რა შუაშია აქ ხელოვნური ინტელექტის თეორია? განსხვავებით ფსიქოლოგიისგან ხელოვნური ინტელექტი არის მეცნიერების დარგი, რომელიც ცდილობს შეისწავლოს აზროვნების პროცესები **ინფორმაციის გადამუშავების** დონეზე შემდეგ შექმნას ინტელექტუალური ქცევის მქონე ხელოვნური სისტემები, ანუ როგორც მათ უწოდებენ ხელოვნური ინტელექტუალური სისტემები. ხელოვნური ინტელექტი მეცნიერების ერთერთი ყველაზე ახალგაზრდა დარგია. პირველი შრომები ამ მიმართულებით გაჩნდა მეორე მსოფლიო ომის შემდეგ, ხოლო ტერმინი - ხელოვნური ინტელექტი პირველად გამოიყენეს 1956 წელს. მისი წარმო-

შობაც და განვითარებაც მჭიდროდაა დაკავშირებული კომპიუტერული ტექნიკის წარმოშობასა და განვითარებასთან და ეს რა თქმა უნდა შემთხვევითი არაა. ყველა ის ზემოთ ჩამოთვლილი ქმედებები, რომლებიც განსაზღვრავენ ინტელექტს, როგორც ფენომენს, თავისი არსით წარმოადგენენ ინფორმაციულ პროცესებს, ანუ ყველა ამ ქმედების დროს ხდება ინფორმაციის მიღება, დამუშავება, შენახვა და გადაცემა. კომპიუტერული ტექნიკის წარმოშობამდე ასეთი უნარი გააჩნდათ მხოლოდ ცოცხალ ორგანიზმებს (სხვადასხვა დონეზე). კომპიუტერებმა საშუალება მოგვცეს მოგვეხდინა ამ პროცესების მექანიზაცია ტექნიკურ სისტემებში, რაც გახდა ხელოვნური ინტელექტუალური სისტემების განვითარების საფუძველი.

სწორედ ამიტომაც კომპიუტერების წარმოშობასთან ერთად გაჩნდა კითხვა – შეიძლება თუ არა კომპიუტერს გააჩნდეს ინტელექტუალური ქცევა? უფრო მძაფრად კი ეს კითხვა ჟღერდა ასე: შეუძლია თუ არა მანქანას იაზროვნოს? ამ ორ კითხვას შორის არსებითი განსხვავებაა. მოვიყვანოთ ასეთი მაგალითი, შეუძლია თუ არა მანქანას ფრენა? რა თქმა უნდა შეუძლია, იტყვის ნებისმიერი ადამიანი, თვითმფრინავი ხომ სწორედ ამისთვისაა შექმნილი. მაგრამ აქ არსებითია რა იგულისხმება ცნება “ფრენის” ქვეშ. თუ ჩვენ ფრენის ქვეშ ვიგულისხმებთ მხოლოდ ჩიტივით ფრენას, მაშინ მანქანას უკვე აღარ შეუძლია ფრენა, მას შეუძლია მხოლოდ გადაადგილება ჰაერში. მაგრამ თუ ფრენა ესაა გადაადგილება ჰაერში, მაშინ მანქანასაც შეუძლია ფრენა, ზუსტად ასევე, თუ აზროვნება ესაა ხოლოდ და მხოლოდ ცოცხალი სისტემების პრეროგატივა და ის დამახასიათებელია მხოლოდ თავის ტვინში მიმდინარე პროცესებისთვის, მაშინ ცხადია მანქანა ვერასოდეს შეძლებს იაზროვნოს და ის შეძლებს მხოლოდ შეასრულოს რაღაც გარკვეული ქმედებები, რომლებიც შესრულებული ადამიანის მიერ აღიქმება როგორც აზროვნების პროცესი. მაგრამ თუ აზროვნება სწორედ ასეთი ქმედებების შესრულების უნარია, მაშინ ცხადია მანქანაც შეძლებს ყველა იმ პარამეტრის ჩამოთვლა, რომლებსაც უნდა აკმაყოფილებდეს ხელოვნური ინტელექტუალური სისტემა, რათა მისი ქცევა შეფასდეს როგორც აზროვნება, ალბათ ძალიან ძნელია. 1948 წელს ინგლისელმა მეცნიერმა ა. ტიურინგმა, რომელიც თვითონ იდგა კომპიუტერების შექმნის სათავეებთან, სტატიაში “ინტელექტუალური მანქანები”, შემოგვთავაზა “მარტივი” ტესტი კომპიუტერის ინტელექტუალური ქცევის შესამოწმებლად. ტესტი მდგომარეობს შემდეგში: მოვათავსოთ ერთ ოთახში ადამიანი, მეორეში მანქანა და მესამეში გამომცდელი. მათ შორის ვიზუალური კავშირი არ არსებობს და გამომცდელმა არ იცის, რომელ ოთახში ზის ადამიანი და რომელ ოთახში დგას მანქანა. ისინი დაკავშირებულნი არიან ერთმანეთთან ისე, რომ გამომცდელს შეუძლია გაცვალოს ინფორმაცია ადამიანთან და მანქანასთან მხოლოდ წერილობით. თუ გამომცდელი თავის მიერ დასმულ შეკითხვებზე მიღებული პასუხების მიხედვით ვერ შეძლებს გაარჩიოს რომელ ოთახში ზის ადამიანი, ხოლო რომელში მანქანა, მაშინ მანქანა შეიძლება ჩაითვალოს ინტელექტუალურ სისტემად, წინააღმდეგ შემთხვევაში კი არა. ცხადია, რომ მანქანას, რომელიც მიიღებს მონაწილეობას ასეთ გამოცდაში უნდა გააჩნდეს მინიმუმ შემდეგი შესაძლებლობები:

- შეეძლოს ტექსტების დამუშავება ბუნებრივ ენაზე, რაც მისცემს მას საშუალებას ესაუბროს გამომცდელს რომელიმე ბუნებრივ ენაზე;
- გააჩნდეს ცოდნის წარმოდგენის საშუალებები, რათა შეძლოს გაიგოს რას ეკითხება გამომცდელი;
- ჰქონდეს ლოგიკური დასკვნების გამოტანის უნარი, რათა შეძლოს მის მეხსიერებაში შენახული ცოდნის და მიღებული ინფორმაციის საფუძველზე გამოიტანოს დასკვნები და უპასუხოს დასმულ შეკითხვას;
- ფლობდეს მანქანური დასწავლის საშუალებებს, რომ შეეძლოს აღმოაჩინოს უკვე ნაცნობი სიტუაციები და აგრეთვე გაერკვეს ახალ სიტუაციებში და მოახდინოს მათზე რეაგირება.

ესაა ტიურინგის არასრული ტესტი. სრული ტესტის შემთხვევაში გამომცდელს გააჩნია აგრეთვე ვიდეოკავშირი, რომლის საშუალებითაც მას შეუძლია შეამოწმოს ობიექტის მიერ (ისე, რომ ვერ ხედავდეს თვით ობიექტს) გარე სამყაროს აღქმისა და მასში მოქმედების უნარი.

ასეთ შემთხვევაში მანქანას დამატებით უნდა შეეძლოს ობიექტების აღქმა, ანუ უნდა ჰქონდეს მხედველობა და გააჩნდეს რობოტოტექნიკური საშუალებები, რათა შეძლოს სივრცეში გადაადგილება ობიექტებით მანიპულირება.

გასაგებია რომ ასეთი უნივერსალური პროგრამა დღეისათვის ჯერ არ არსებობს, მაგრამ ყოველწლიურად ტარდება იმ პროგრამების კონკურსი, რომლებიც ყველაზე უკეთესად დააკმაყოფილებენ ტიურინგის ტესტს და გამარჯვებულს ენიჭება ლებნერის პრიზი.

ზემოთჩამოთვლილი ექვსი შესაძლებლობა, რომელსაც უნდა ფლობდეს პროგრამა, რათა დააკმაყოფილოს ტიურინგის ტესტი, ფაქტობრივად წარმოადგენს ხელოვნური ინტელექტის იმ ექვს ძირითად მიმართულებას, რომლებიც განსაზღვრავენ დღეს ხელოვნური ინტელექტის განვითარების გზებს. და მაინც, დავუბრუნდეთ მთავარ კითხვას და განვმარტოთ რას ნიშნავს, რომ მანქანა აზროვნებს? განვიხილოთ მარტივი მაგალითი, კვადრატული განტოლების ამოხსნა. თითქმის ყველასათვის ცნობილია ამ განტოლების ამოხსნის ალგორითმი. დავსვათ ასეთი კითხვა – აზროვნებს თუ არა ადამიანი კვადრატული განტოლების ამოხსნის დროს? პასუხი ალბათ ძალიან მარტივია. თუ მან იცის ამოხსნის ალგორითმი, ის მექანიკურად ჩასვავს რიცხვებს ფორმულაში და გამოიანგარიშებს პასუხს, მაგრამ თუ მას დაავიწყდა ეს ფორმულა, ან არც არასდროს უსწავლია, ის გამოიყენებს რა მთელ თავის გამოცდილებას რაც მას გააჩნია საერთოდ, შეეცდება რაიმე საშუალებებით მიაგნოს ამ ამოხსნას. მოქმედებას პირველ შემთხვევაში ძნელად თუ ვუწოდებთ აზროვნების პროცესს, მაშინ, როდესაც მეორე შემთხვევა ნამდვილად წარმოადგენს აზროვნების პროცესს იმის და მიუხედავად მიაგნებს თუ არა ადამიანი ამოხსნას.

იმ მიზნით, რომ მოვახდინოთ აზროვნების პროცესის მოდელირება, პირველ რიგში ჩვენ უნდა ვიცოდეთ როგორ აზროვნებს ადამიანი. ამ საკითხში გასარკვევად არსებობს ორი გზა: პირველი ესაა ინტროსპექცია (მივყვეთ ჩვენი საკუთარი აზროვნების პროცესს და გადმოვცეთ იგი სიტყვების საშუალებით), რაც ხშირ შემთხვევაში შეუძლებელია, რადგანაც ადამიანი იშვიათად თუ შეძლებს გადმოსცეს ზუსტად თავისი აზროვნების პროცესი სიტყვებით. მეორე გზაა შევისწავლოთ აზროვნების პროცესი ფსიქოლოგიური ექსპერიმენტების საშუალებით და ჩამოვაყალიბოთ აზროვნების მეტნაკლებად ზუსტი თეორია, წარმოვადგინოთ ის პროგრამის სახით და თუ ასეთი პროგრამის შესასვლელი და გამოსასვლელი მონაცემები და პროგრამის მიერ შესრულებული ქმედებების დროში განაწილება ზუსტად შეესაბამება ადამიანის ქცევას, ეს იქნება იმის დასტური, რომ ამ პროგრამის ზოგიერთი მექანიზმი მართლაც აღწერის ადამიანის თავის ტვინში მიმდინარე ინფორმაციულ პროცესებს.

1.3. პირველი ევრისტიკული პროგრამა. ამ გზით წავიდნენ ამერიკელი მეცნიერები ა. ნიუელი და გ. საიმონი, რომლებმაც მიზნად დაისახეს შეექმნათ ისეთი პროგრამა, რომლის ქცევაც იქნებოდა ადამიანის აზროვნების პროცესის მოდელი. ამ მიზნით მათ შეარჩიეს სტუდენტები, რომლებსაც არ ჰქონდათ შესაბამისი მომზადება ისეთ დარგში, როგორსაც წარმოადგენს მათემატიკური ლოგიკა და დაუსვეს ასეთი ამოცანა: მოცემული გვაქვს ცვლადები **p,q,r,…,A,B,C,…,** კავშირები: ¬ (უარყოფა), ∨ (ან) და → (იმპლიკაცია, ჩვენი სპეციალობის სტუდენტებისთვის არ მიმაჩნია საჭიროდ ამ ოპერაციების განმარტება). გვაქვს ხუთი გამოსახულება, რომლებსაც ეწოდებათ აქსიომები, ანუ ჭეშმარიტი გამოსახულებები;

$$(p \vee p) \rightarrow p$$
$$p \rightarrow (q \vee p)$$
$$(p \vee q) \rightarrow (q \vee p)$$
$$[[p \vee (q \vee r)] \rightarrow [q \rightarrow (p \vee r)]$$
$$(p \rightarrow q) \rightarrow [(r \vee p) \rightarrow (r \vee q)].$$

გვაქვს გამოყვანის სამი წესი: ჩასმის, შეცვლის და გამოყოფის. ჩასმის წესის თანახმად, ცვლადის მაგივრად შეიძლება ნებისმიერი გამოსახულების ჩასმა, თუ ეს ჩასმა ხორციელდება მთელ გამოსახულებაში. შეცვლის წესი საშუალებას გვაძლევს შევცვალოთ კავშირი მის განმარტებით და პირიქით, ყველგან და ყოველთვის. გამოყოფის წესის თანახმად,

თუ A თეორემაა და A → B თეორემაა, მაშინ B-ც თეორემაა. თეორემად ითვლება გამოსახულება, რომელიც შეიძლება გამოვიყვანოთ რომელიმე ზემოთმოყვანილი ხუთი აქსიომიდან ან უკვე დამტკიცებული თეორემიდან, გამოყვანის ამ სამი წესის საშუალებით.

აშკარაა, რომ გამოსახულებები შეიძლება იყოს ძალიან ბევრი და ისინი შეიძლება ადგენდნენ მიმდევრობათა ჯაჭვს, როდესაც ერთი გამოსახულებიდან გამოდის მეორე და ა.შ.. თეორემის დამტკიცება მოცემული გამოსახულებისათვის დაიყვანება იმ მიმდევრობის პოვნაზე, რომელიც გამომდინარეობს აქსიომებიდან (ან უკვე დამტკიცებული თეორემებიდან) დამთავრდება დასამტკიცებელი გამოსახულებით. შესაბამისად, ალგორითმი, რომელიც სრული გარანტიით მოგვცემს საშუალებას (ასეთი ალგორითმი ცნობილია ბრიტანეთის მუზეუმის ალგორითმის სახელით) ვიპოვოთ თეორემის დამტკიცება (თუკი ეს დამტკიცება არსებობს), მუშაობს შემდეგი პრინციპით: საწყისი აქსიომებიდან და უკვე დამტკიცებული თეორემებიდან ააგებს ყველა შესაძლო მიმდევრობებს სამი წესის გამოყენებით. ყოველ ნაბიჯზე შეამოწმებს, ხომ არ მიიღო უკვე დასამტკიცებელი გამოსახულება. ასევე ყოველ ნაბიჯზე მოიშორებს დუბლიკატებს, ანუ ისეთ გამოსახულებებს, რომლებიც უკვე აიგო ამ ან რომელიმე სხვა მიმდევრობაში. ადრე თუ გვიან ალგორითმი აუცილებლად იპოვის სასურველი თეორემის დამტკიცებას, რა თქმა უნდა, თუკი ის არსებობს. ალგორითმი იძლევა დამტკიცების პოვნის სრულ გარანტიას, მაგრამ მისი ეფექტიანობა სერიოზულ ეჭვს იწვევს.

ა.ნიულემა და გ. საიმონმა ჩაატარეს შემდეგი სახის ექსპერიმენტები. მათ შეარჩიეს სტუდენტები, რომლებმაც არ იცოდნენ მათემატიკური ლოგიკა, ჩამოაყალიბეს ამოცანა და სთხოვეს სუდენტებს "ეფიქრათ ხმამაღლა", ანუ აეხსნათ თუ რატომ ღებულობდნენ ამა თუ იმ გადაწყვეტილებებს. ექსპერიმენტის მიზანი იყო აღმოეჩინათ ის მექანიზმები, რომლებიც განაპირობებენ ადამიანის ინტელექტუალურ ქცევას უცნობ გარემოში (ამიტომ იყო შერჩეული სტუდენტთა ჯგუფები, რომლებსაც არ ჰქონდათ შესწავლილი მათემატიკური ლოგიკა). ჩატარებული ექსპერიმენტების საფუძველზე ა. ნიუელმა და გ. საიმონმა დაადგინეს, რომ განსხვავებით ზემოთაღწერილი ალგორითმისგან ადამიანი ყოველთვის ცდილობს დაიწყოს მიმდევრობის აგება დასამტკიცებელი გამოსახულებიდან ცნობილი თეორემებისა და აქსიომებისაკენ და ამ მიმდევრობების აგებისას ძირითადად ხელმძღვანელობს შემდეგი ოთხი წესით: ჩასმის მეთოდი, გამოყოფის მეთოდი, პირდაპირი ჯაჭვწარმოქმნის მეთოდი და უკუჯაჭვწარმოქმნის მეთოდი. ჩასმის მეთოდი ეძებს თეორემის დამტკიცებას ცნობილ თეორემებსა და აქსიომებში ცვლადებისა და კავშირების შეცვლის გზით. გამოყოფის მეთოდი გამოიყენებს რა გამოყოფის წესს, ცვლის ამოცანას ქვეამოცანით. მაგალითად, თუ **B** გამოსახულებაა, რომელიც უნდა დავამტკიცოთ, მეთოდი ეძებს გამოსახულებას **A→B** ან გამოსახულებას **B→A** (რომელიც უკვე თეორემაა) და მაშინ ახალი ქვეამოცანა ხდება **A**-ს დამტკიცება. პირდაპირი და უკუჯაჭვწარმოქმნის მეთოდები კი **A→B** გამოსახულების დასამტკიცებლად ეძებს აქსიომას ან თეორემას **C→B** (**B→C**) და თუ იპოვნის, მაშინ ახალ ქვეამოცანად იქცევა **A→C** (**C→A**) გამოსახულების დამტკიცება. განსხვავებით ადამიანისაგან, **LG** ამ წესებს იყენებს მკაცრად განსაზღვრული მიმდევრობით, რაც ცხადია ამცირებს მის ეფექტიანობას.

განვიხილოთ კონკრეტული მაგალითი. დავუშვათ, მოცემულია გამოსახულება

$$\neg (p \to q) \to \neg p.$$

LG –თვის გარდა ხუთი აქსიომისა ცნობილია აგრეთვე ორი თეორემა:

1. $A \to (A \vee B)$ (1)
2. $(A \to B) \to (\neg B \neg A)$ (2)

LG ამ თეორემას ამტკიცებს შემდეგნაირად: პირველ თეორემაში **A**–ს მაგივრად ჩასვავს **p**-ს, ხოლო **B** –ს მაგივრად - **q**-ს. მიიღებს გამოსახულებას

$$p \to (p \vee q) \qquad (3)$$

შემდეგ (2) –ში ჩასვავს **A**–ს მაგივრად **p**-ს და **B**–სმაგივრად **(p∨q)**–ს. მიიღებს გამოსახულებას

$$[p \to (p \vee q)] \to [\neg(p \to q) \to \neg q]$$

$$p \to (q \to p)$$

შედარების პროცედურა ძალიან ეფექტურია და საშუალებას გვაძლევს შევამციროთ ქვეამოცანების ძებნის პროცესი. ამ პროცედურის ეფექტიანობა აიხსნება იმით, რომ შედარება ხდება კომპონენტების მიხედვით და უკუკავშირის საშუალებით შედეგები შეიძლება გამოყენებული იქნას შემდეგი ბიჯის სამართავად. შედარების პროცედურა გამოიყენება ყველა მეთოდში, მაგრამ მისი ეფექტიანობისათვის საჭიროა ჯერ ვიპოვოთ სწორი თეორემა ან აქსიომა, რომელსაც შევადარებთ გამოსახულებას. ამ მიზნისთვის LG–ს გააჩნია მსგავსებაზე გამოცდის პროცედურა, რომლის თანახმადაც, LG ორ ლოგიკურ გამოსახულებას თვლის მსგავსად, თუ მარცხენა და მარჯვენა ნაწილებში თანაბარია:

1. დონეთა მაქსიმალური რაოდენობა მთავარი კავშირიდან ნებისმიერ ცვლადამდე;
2. განსხვავებულ ცვლადთა რაოდენობა;
3. ცვლადების საერთო რაოდენობა გამოსახულებაში.

LG-ს ყველა ოთხივე მეთოდის სამართავად გააჩნია მმართველი პროგრამა, რომელიც თავისი არსით არის ალგორითმული, და მდგომარეობს შემდეგში. მეთოდების გამოყენება ხდება მკაცრად განსაზღვრული მიმდევრობით. ჯერ გამოიყენება ჩასმის მეთოდი, შემდეგ გამოყოფის მეთოდი, პირდაპირი ჯაჭვწარმოქმნის მეთოდი და ბოლოს უკუჯაჭვწარმოქმნის მეთოდი. ამ გზით LG ცდილობს გამოიმუშაოს ახალი ქვეამოცანა, რომელსაც დაამტკიცებს ჩასმის მეთოდით. თუ ეს შეუძლებელია, ქვეამოცანა შეიტანება ქვეამოცანათა სიაში. თუ დამტკიცდება თუნდაც ერთი ქვეამოცანა ჩასმის მეთოდით, ცხადია ამით დამტკიცებული იქნება საწყისი თეორემაც. თუ ვერ მოხერხდა საწყისი თეორემის დამტკიცება ვერცერთი მეთოდის გამოყენებით, მაშინ ამოირჩევა ახალი ქვეამოცანა ქვეამოცანათა სიიდან და მისთვის ყველაფერი მეორდება ახლიდან. პროცესი დამთავრდება როცა ნაპოვნი იქნება დამტკიცება, ან როდესაც კომპიუტერს აღარ ეყოფა მეხსიერება. პროცესი შეიძლება შეწყდეს მაშინაც, როდესაც ამოიწურება დრო, რომელიც ამ თეორემის დასამტკიცებლად იყო გამოყოფილი.

ამგვარად, თეორემის დამტკიცებისას LG წარმოქმნის ქვეამოცანათა გარკვეულ ხეს, რომლის საშუალებითაც ის ცდილობს დასამტკიცებელი გამოსახულებიდან მიაღწიოს აქსიომებს და უკვე დამტკიცებულ თეორემებს. მიუხედავად იმისა, რომ იმ ქვეამოცანათა სიმრავლე, რომელსაც გამოიმუშავებს LG გაცილებით მცირეა, ვიდრე ის სიმრავლე, რომელსაც ააგებდა ბრიტანეთის მუზეუმის ალგორითმი, არსებობს ამოცანები, რომელთა გადაწყვეტა არ შეუძლია არც LG. ეს იმიტომ ხდება, რომ მმართველი პროგრამა არის ალგორითმული სახის, რომელც იძლევა იმის გარანტიას, რომ თუკი რაიმე ქვეამოცანა შეიძლება მიღებული იქნას გამოყოფის და ჯაჭვწარმოქმნის მეთოდებით, ის აუცილებლად იქნება აგებული (თუნდაც არ არსებობდეს მისი საჭიროება). გარდა ამისა, ყველა ეს ქვეამოცანა განიხილება იმ თანმიმდევრობით, რომლითაც ისინი წარმოიშვნენ, რაც აგრეთვე დიდ გავლენას ახდენს დამტკიცების პროცესსზე. სეთი მმართველი პროგრამის გამო და იმის გამო რომ მეთოდები დიდი რაოდენობით გამოიმუშავებენ ქვეამოცანებს, ყოველ ბიჯზე ქვეამოცანათა ხე იზრდება ექსპონენციალურად. ეს კი იწვევს იმას, რომ თუ დამტკიცება შეიცავს რამდენიმე ბიჯს (სამი, ოთხი ან მეტი) LG ეფექტურობა ძალიან მცირეა. ასე მაგალითად, მან შესძლო დაემტკიცებინა მხოლოდ ერთი ოთხბიჯიანი თეორემა.

ეს პროგრამა გამოქვეყნდა 1956 წელს და გამოიწვია დიდი დაინტერესება იმის გამო რომ ეს იყო პირველი პროგრამა, რომელიც ცდილობდა მოეხდინა ადამიანის აზროვნების მოდელირება ინფრომაციული პროცესების დონეზე და ამავე დროს იყო პირველი პროგრამა, რომელიც ამუშავებდა არა რიცხვებს, არამედ სიმბოლოებს.

ძალიან მალე გაჩნდა კიდევ სხვა პროგრამები, რომლებიც ცდილობდნენ ამოეხსნათ სხვადასხვა ამოცანები ანალოგიური მიდგომით. კერძოდ, შეიქმნა პროგრამა, რომელიც ხსნიდა გეომეტრიულ ამოცანებს სასკოლო კურსიდან; პროგრამა, რომელიც სიმბოლურ დონეზე ხსნიდა ინტეგრალურ განტოლებებს; პროგრამა, რომელიც ახდენდა ამწყობი კონვეირის მუშაობის ბალანსირებას და სხვა. ყველა ამ პროგრამისათვის დამახასიათებელი იყო ისეთი პროცედურების გამოყენება, რომლებსაც უწოდებნენ ევრისტიკებს და რომლებიც გამომდინარეობენ არა ამოცანის პირობებიდან, არამედ ეფუძნებიან ბუნებრივი ინტე-

ლექტის ზოგად მოსაზრებებს ან /და გამოცდილებას ასეთი ამოცანების ამოხსნის დროს. ამიტომ ხელოვნური ინტელექტის ამ მიმართულებას ეწოდა ევრისტიკული პროგრამირება.

ევრისტიკული პროგრამირების ძირითადი პრინციპია პრობლემის (ან ამოცანის) წარმოდგენა ძებნის პროცესის სახით და შემდეგ ამ პროცესის გამარტივება ევრისტიკების საშუალებით. ევრისტიკული პროგრამების მრავალფეროვნებამ დაგვანახა, რომ ეს პროგრამები ხშირ შემთხვევაში არ წარმოადგენდნენ ადამიანის აზროვნების მოდელს. მართლაც, ყველა ეს პროგრამა შექმნილი იყო იგივე პრინციპით, რა პრინციპითაც **LG**, ანუ ემყარებოდნენ ექსპერიმენეტებს და ჰქონდათ პრეტენზია, რომ წარმოადგენდნენ ადამიანის აზროვნების ინფორმაციულ მოდელს. მაგრამ ეს პროგრამები აბსოლუტურად განსხვავდებოდნენ ერთმანეთისაგან და ერთ პროგრამას არ შეეძლო მეორე პროგრამის ამოცანების გადაჭრა. ამგვარად, გამოდიოდა, რომ პრაქტიკულად მათგან არც ერთი არ წარმოადგენდა აზროვნების მოდელს, რადგანაც ცხადია, რომ აზროვნების პროცესი არ შეიძლება იცვლებოდეს იმის და მიხედვით, თუ რა ამოცანას დავუსვავთ მას.

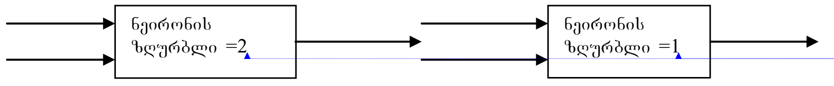
აქ მოხდა ევრისტიკული პროგრამირების გაყოფა ორ მიმართულებად. ამ წინააღმდეგობის გადასალახავად და იმისათვის, რომ ეჩვენებინათ, რომ მათ მიერ შედგენილი პროგრამა ნამდვილად არის აზროვნების პროცესის მოდელი ინფორმაციულ დონეზე, ა. ნიუელმა და გ. საიმონმა დაიწყეს მუშაობა ახალ პროგრამაზე, რომელსაც უწოდეს პრობლემათა ზოგადი ამომხსნელი (**GPS–General Problem Solvers**). ამ პროგრამას უკვე შეეძლო ამოეხსნა სხვადასხვა სახის ამოცანები, მაგრამ იყო ნაკლებად ეფექტური და უკვე საშუალო ზომის ამოცანების ამოხსნის დროს აღარ ყოფნიდა მეხსიერება და დრო. ავტორები ამას ხსნიდნენ იმდროინდელი კომპიუტერების სისუსტით და ფიქრობდნენ, რომ მომავალში, როდესაც გაიზრდებოდა კომპიუტერების სისწრაფე და მეხსიერება, შესაძლებელი იქნებოდა ნებისმიერი ზომის ამოცანების ამოხსნა. ევრისტიკული პროგრამირების მეორე მიმართულება ცდილობდა შეექმნა ამოცანათა ამოხსნის ეფექტური ალგორითმები, რომლებშიც გამოყენებული იქნებოდა ნებისმიერი ხერხი გადარჩევის შესამცირებლად ისე, რომ ყურადღება აღარ მიექციათ, არის თუ არა ასეთი მეთოდი დამახასიათებელი ადამიანის აზროვნებისთვის. ანუ ამ შემთხვევაში მთავარი იყო პროგრამას ემუშავა ეფექტურად. თუ შევადარებთ ჩვენს მაგალითს ჩიტისა და თვითმფრინავის შესახებ, შეიძლება ითქვას, რომ პირველი მიმართულება ცდილობდა ხელოვნური ჩიტის შექმნას, ხოლო მეორე კი სხვადასხვა სახის თვითმფრინავების აგებას კონკრეტული მიზნებისათვის.

ევრისტიკული პროგრამირების ასეთი აღმავლობა გრძელდებოდა გასული საუკუნის სამოცდაათიან წლებამდე. ამ დროისთვის თანდათანობით აშკარა გახდა, რომ ის დაპირებები, რომლებსაც უხვად იძლეოდნენ მეცნიერები, შეუსრულებელი იყო. პირველ რიგში ამ პროგრამათა უდიდეს ნაწილს არ ჰქონდათ (ან ჰქონდათ ძალიან მცირე) ცოდნა იმ საკითხებზე, რომელთა გადაჭრასაც ისინი ცდილობდნენ. ამ მხრივ ტიპური მაგალითია ის სამუშაოები, რომლებსაც უხვად აფინანსებდა აშშ ნაციონალური სამეცნიერო – კვლევითი საბჭო. საქმე ეხებოდა სამეცნიერო ტექსტების მანქანურ თარგმნას რუსული ენიდან ინგლისურზე.ითვლებოდა, რომ წინადადების ზუსტი აზრის შესანარჩუნებლად საკმარისი იყო ჩაეტარებინათ სინტაქსური გარდაქმნა, დამყარებული რუსული და ინგლისური ენების გრამატიკებზე და შეეცვალათ სიტყვები ელექტრონული ლექსიკონის გამოყენებით. მაგრამ საქმე იმაშია, რომ არაერთმნიშვნელოვნების მოსასპობად და წინადადების აზრის გასაგებად საჭიროა სისტემა ფლობდეს ცოდნას მეცნიერების იმ დარგში, რომელსაც ეხება სტატია. აი როგორი შედეგი მიიღეს ინგლისური ფრაზის **"The spirit is willing but the fiesh is weak"** (სული სავსეა სურვილებით, მაგრამ ხორცი სუსტია), ჯერ რუსულ და შემდეგ კვლავ ინგლისურ ენაზე თარგმნის შემდეგ: "**The vodka is good but the meat is rotten"** (არაყი კარგია, მაგრამ ხორცი გაფუჭებულია).

არსებითი გავლენა ხელოვნური ინტელექტის განვითარების შეფერხებაზე მოახდინა იმანაც, რომ ყველა ამ პროგრამებში პრობლემების გადაჭრას ცდილობდნენ გადარჩევის გზით, ანუ შესაძლო ნაბიჯების სხვადასხვა კომბინაციების შემოწმების გზით, რომელიც გრძელდებოდა სანამ არ იქნებოდა ნაპოვნი ამოხსნა. პატარა ამოცანების ამოხსნის

დროს ეს სტრატეგია ამართლებდა. სანამ არ იყო დამუშავებული გამოთვლითი სირთულის თეორია, მიაჩნდათ, რომ უფრო დიდი ამოცანების ამოსახსნელად საკმარისია უფრო სწრაფი კომპიუტერები მეხსიერების დიდი მოცულობით. მაგალითად, ის ოპტიმიზმი, რომელიც მოჰყვა რეზოლუციის პრინციპის აღმოჩენას, ძალიან მალე გაქრა, როდესაც ვერ შეძლეს დაემტკიცებინათ ამ პრინციპით თეორემები, რომლებიც შეიცავდნენ რამოდენიმე ათეულ ფაქტს. აღმოჩნდა, ის ფაქტი, რომ პროგრამას პრინციპულად შეუძლია იპოვოს ამოხსნა, სულაც არ ნიშნავს რომ ის მართლაც შეიცავს ყველა მექანიზმს, რომელიც საშუალებას მოგვცემს ვიპოვოთ ამოხსნა პრაქტიკულად.

1.3. ნეირონული ქსელების თეორია. ა. ნიუელისა და გ. საიმონის სტატიის გამოჩენამდე დიდი ხნით ადრე, ჯერ კიდევ ორმოციან წლებში, უ. მაკ –კალოკმა და უ. პიტსმა გამოაქვეყნეს სტატია, რომელშიც მათ შემოგვთავაზეს ადამიანის ნერვული სისტემის ძირითადი ელემენტის – ნერვული უჯრედის – ნეირონის მოდელი. იმ ინფორმაციის საფუძველზე, რაც მეცნიერებას გააჩნია ნეირონის აგებულების და ფუნქციონირების შესახებ, მათ დაასკვნეს, რომ ნეირონი ეს არის ელემენტი მრავალი შესასვლელითა და ერთი გამოსასვლელით. შესასვლელები შეიძლება იყვნენ ორ მდგომარეობაში, აღგზნებულ ან მოსვენებულ მდგომარეობაში. იმის და მიხედვით რამდენი შესასვლელია აგზნებული და როგორია ნეირონის აგზნების ზღურბლი, ნეირონის გამოსასვლელზე ან გვექნება სიგნალი (ნეირონი აღიგზნება) ან არა. უ. მაკ – კალოკმა და უ. პიტსმა შექმნეს ასეთი ელემენტის ტექნიკური მოდელი, რომელსაც უწოდეს ხელოვნური ნეირონი. მაგალითად, ორშესასვლელიანი ხელოვნური ნეირონის საშუალებით ძალიან მარტივად შესაძლებელია ∧ და ∨ კავშირების მოდელირება (იხ. ნახ. #1.1)



Formatted: Font: AcadNusx, 10 pt

Formatted: Font: AcadNusx, 10 pt

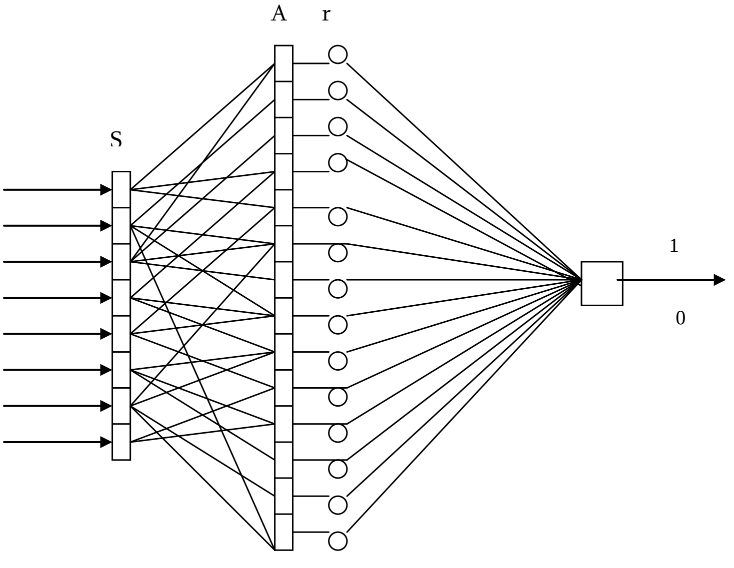
სურ. #1.1. ლოგიკური ფუნქციების განხორციელება ხელოვნური ნეირონების საშუალებით.

უ. მაკ-კალოკმა და უ. პიტსმა აჩვენეს, რომ ასეთი ხელოვნური ნეირონებისგან შეიძლება შეიქმნას ქსელი, რომელშიც ნეირონები იმყოფებიან “ჩართულ” ან “გამორთულ” მდგომარეობაში. გადასვლა “ჩართულ” მდგომარეობაში ხდება მეზობელი ნეირონების სტიმულაციის შედეგად და ასეთი ნეირონული ქსელების საშუალებით შეიძლება გამოვთვალოთ ნებისმიერი გამოთვლადი ფუნქცია და ამასთანავე მარტივი ლოგიკური ქსელით მოვახდინოთ ნებისმიერი ლოგიკური კავშირის (“არა”, “ან”, “და” და ასე შემდეგ) რეალიზაცია. გარდა ამისა, მათ გამოთქვეს მოსაზრება, რომ ასეთ ლოგიკურ ქსელებს აქვთ დასწავლის უნარი.

ეს მოსაზრება დაადასტურა 1957 წელს ფ. როზენბლატმა, რომელმაც გამოაქვეყნა ასეთი ნეირონული ქსელების საშუალებით შექმნილი პერცეპტრონის (აღმქმელი მოწყობილობა) აღწერა, რომელსაც შეუძლია შეასრულოს უფრო მეტი, ვიდრე ესაა ლოგიკური გამოთვლები, კერძოდ პერცეპტრონს შეუძლია გამოიცნოს ორგანზომილებიანი ფიგურები. ამასთან, რაც უფრო მთავარია, მას გააჩნდა დასწავლის უნარი. პროცესები, რომლებიც მიმდინარეობენ პერცეპტრონში ემყარება არა მარტო ლოგიკას, არამედ სტატისტიკასაც. ფ. როზენბლატის უმარტივესი სამშრიანი პერცეპტრონი შედგება გამოსახულების აღმქმელი ნეირონების პირველი (**S**) შრისაგან, რომელზეც ხდება გამოსახულების პროეცირება. თუ ნეირონზე მოხვდება გამოსახულება, ნეირონი აღიგზნება და გამოსასვლეზე გვექნება სიგნალი (1), თუ არა, ნეირონი მოსვენებულ მდგომარეობაშია და გამოსასვლელზე სიგნალი არ გვაქვს (0).

გამოსახულების აღმქმელი შრე დაკავშირებულია ასოციატური ნეირონების (**A**) შრე-სთან ნებისმიერად (მაგალითად, ყველა ყველასთან). ამ შრის ნეირონებს გააჩნიათ გარკვე-ული ზღურბლი და თუ მოსული სიგნალების ჯამი გადააჭარბებს ამ ზღურბლს, გამოსასვლელზე გვექნება სიგნალი, რომელიც (r) რეგულატორების საშუალებით შეიძლება გავაძლიეროთ, ან შევასუსტოთ. მესამე შრე შედგება ერთი ნეირონისაგან, რომელიც წარმოად-

გენს სუმატორს გარკვეული ზღურბლით და ერთი გამოსასვლელით, რომელზედაც შეიძლება გვქონდეს სიგნალი (1), ან არ გვქონდეს სიგნალი (0). ასეთ პერცეპტრონს შეუძლია განასხვაოს ერთმანეთისაგან ობიექტების ორი კლასი.



სურ. #1.2. როზენბლატის სამშრიანი პერცეპტრონი.

ჩვენ უნდა მივაღწიოთ იმას, რომ ერთი კლასის ობიექტების პროეცირებისას პერცეპტრონის გამოსასვლელზე ვღებულობდეთ ერთ სიგნალს (მაგალითად 1) და მეორე კლასის ობიექტებზე მეორე სიგნალს (0), ანუ უნდა დავასწავლოთ პერცეპტრონს ობიექტების ერთმანეთისაგან გარჩევა. ეს მიზანი მიიღწევა (r) რეგულატორების საშუალებით. ფაქტობრივად, პერცეპტრონის თავდაპირველი სტოქასტური სქემა დასწავლის პროცესში ღებულობს გარკვეულ სტრუქტურას, რომელიც შეესაბამება იმ ობიექტებს, რომელთა პროეცირებაც ხდება პერცეპტრონის შესასვლელზე. ფ. როზენბლატმა დაამტკიცა თეორემა ასეთი პერცეპტრონის კრებადობის შესახებ, რომელიც ადასტურებს, რომ მის მიერ შემოთავაზებულ ალგორითმს შეუძლია პერცეპტრონში გაერთიანებული ხელოვნური ნეირონებს შორის კავშირების რაოდენობების კორექტირება ნებისმიერი შესასვლელი მონაცემების შესაბამისად, იმ პირობით, რომ ასეთი შესაბამისობა არსებობს.

1969 წელს მ. მინსკმა და ს. პეიპერტმა გამოაქვეყნეს შრომა, რომელშიც აჩვენეს, რომ პერცეპტრონს, რომელიც წარმოადგენს ნეირონული ქსელის მარტივ ფორმას, მართლაც შეუძლია დაიწავლოს ყველაფერი, რაც შეიძლება წარმოდგენილი იქნას პერცეპტრონში, მაგრამ პრობლემა მდგომარეობს იმაში, რომ ძალიან ბევრი, საინტერესო სტრუქტურების წარმოდგენა პერცეპტრონში შეუძლებელია. მაგალითად, პერცეპტრონი მხოლოდ ორი შესასვლელი ნეირონით ვერასოდეს ვერ შეძლებს გამომრიცხავი ან ოპერაციის დასწავლას.

1.4. ცოდნის წარმოდგენა ინტელექტუალურ სისტემებში. აშკარა გახდა, რომ სპეციფიკური ცოდნის გარეშე, მხოლოდ ძებნის ზოგადი პრინციპების გამოყენებით, რთული ამოცანების ამოხსნა შეუძლებელია. აუცილებელია, ასეთ სისტემებს გააჩნდეთ გარკვეული ცოდნა, არა მარტო ძებნის ზოგადი პრინციპების შესახებ, არამედ იმ კონკრეტულ სფეროშიც, რომელშიც ისინი ხსნიან ამოცანებს. ეს იდეაც მოდის ბუნებრივი ინტელექტიდან. ადამიანები, როგორც წესი, იშვიათი გამონაკლისების გარდა, რომლებსაც წარმოადგენენ გენიოსები, სპეციალიზირდებიან გარკვეულ ვიწრო დარგში და გააჩნიათ ცოდნა და გამოცდილება ამ დარგში. მათ შეუძლიათ წარმატებით გადაჭრან ამოცანები მოცემულ დარგში, მაგრამ შე-

იძლება იყვნენ აბსოლუტური დილეტანტები სხვა საკითხებში. თუ ჩვენი მიზანია შევქმნათ ხელოვნური ინტელექტუალური სისტემა, რომელიც წარმატებით ხსნის ამოცანებს რომელიმე კონკტერულ დარგში და არა ხელოვნური გონება, რომელსაც შეუძლია განვითარება თავისი ინტერესების მიხედვით შემდგომში, მაშინ ჩვენ ასეთ სისტემებში აუცილებლად უნდა შევიტანოთ გარკვეული სახის ცოდნა, რომელიც საშუალებას მისცემს ხელოვნურ ინტელექტუალურ სისტემას წარმატებით გადაჭრას ამოცანები კონკრეტულ დარგში. ერთერთი პირველი პროგრამა, რომელშიც რეალიზებული იყო ასეთი მიდგომა, იყო პროგრამა დენდრალი. პროგრამა შექმნილი იყო ქიმიური ნივთიერებების სტრუქტურის დასადგენად. პროგრამის შესასვლელები იყო შენაერთის ქიმიური ფორმულა და მასათა სპექტრი, პროგრამას კი უნდა დაედგინა მოლეკულის სივრცითი სტრუქტურა. პროგრამის პირველი, პრიმიტიული ვერსია ცდილობდა გამოემუშავებინა მოცემული ფორმულის შესაბამისი ყველა შესაძლო სტრუქტურა და შემდეგ დაედგინა როგორი მასათა სპექტრი უნდა შეესაბამებოდეს მოცემულ სტრუქტურას. დიდი ზომის მოლეკულებისთვის ასეთი ამოცანის ამოხსნა შეუძლებელი აღმოჩნდა, ამიტომ, ქიმიკოს-ანალიტიკოსებთან კონსულტაციების შემდეგ, პროგრამის შემდგენელები მივიდნენ იმ დასკვნამდე, რომ საჭიროა პირიქით, მასათა სპექტრში არსებული პიკების საშუალებით მოეძებნათ ის ცნობილი ქვესტრუქტურები, რომლებიც იძლევიან ამ პიკებს. მაგალითად, კეტონური ქვეჯგუფების (C=O) ატომური მასით 28 დასადგენად, მათ გამოიყენეს შემდეგი წესი:
თუ არსებობს ორი წერტილი **X1** და **X2** , ისეთი ,რომ:

ა) **X1 + X2 = 28+ M** (სადაც M არის მთელი მოლეკულის მასა);
ბ) წერტილში X1 – 28 - მაღალია პიკია;
გ) წერტილში X2 – 28 - მაღალი პიკია;
დ) ერთერთ წერტილში მაინც **X1** ან **X2** - მაღალი პიკია;
მაშინ არსებობს კეტონური ჯგუფი.

პროგრამა დენდრალი აღმოჩნდა ძალიან ძლიერი და ეფექტური პროგრამა, რაც განპირობებული იყო იმით, რომ ყველა ის თეორიული ცოდნა, რომელიც საჭიროა ასეთი ამოცანის ამოსახსნელად ჩამოყალიბებული იქნა ზუსტი რეცეპტების სახით. ეს იყო პირველი ექსპერტული სისტემა, რომელიც ამოცანის ამოხსნის დროს იყენებდა არა მარტო ძებნის ზოგად მეთოდებს, არამედ ცოდნასაც. ასე გაჩნდა ხელოვნურ ინტელექტში ცოდნის წარმოდგენის პრობლემა.

1.5. ხელოვნური ინტელექტის ჩამოყალიბება მეცნიერებად. გასული საუკუნის ოთხმოციანი წლებიდან ხელოვნური ინტელექტი თანდათანობით ჩამოყალიბდა როგორც მეცნიერება, რომელსაც გააჩნია თავისი კვლევის საგანი, მეთოდოლოგია და საკმაოდ ფართო პრაქტიკული გამოყენება. პირველ რიგში ძირითადი აქცენტი გადატანილი იქნა იმაზე, რომ ხელოვნური ინტელექტის თეორიები უნდა იყოს დაფუძნებული მათემატიკურ პრინციპებზე. ამას ხელი შეუწყო იმანაც, რომ თვითონ მათემატიკამ განიცადა სერიოზული ცვლილებები ხელოვნური ინტელექტის გავლენით, მხედველობაში გვაქვს არამკაფიო სიმრავლეთა თეორიის განვითარება და მისი სულ უფრო ფართო გამოყენება გადაწყვეტილების მიღებისა და მართვის თეორიის პრობლემებში. აქედან გამომდინარე, შესაძლებელი გახდა მათემატიკურ მეცნიერებებში მიღებული შედეგების ფართოდ გამოყენება ხელოვნური ინტელექტის პრობლემების გადწყვეტის დროს (პირველ რიგში უნდა აღინიშნოს მიღწევები სახეთა გამოცნობის დარგში).

ერთიანმა თეორიულმა მიდგომამ შესაძლებელი გახადა კვლავ წინა პლანზე წამოწეულიყო “ერთიანი აგენტის” პრობლემის შექმნის საკითხიც, რაც საშუალებას მოგვცემს დავაკავშიროთ ერთმანეთთან ხელოვნური ინტელექტის სხვადასხვა მიმართულებებში მიღებული შედეგები.

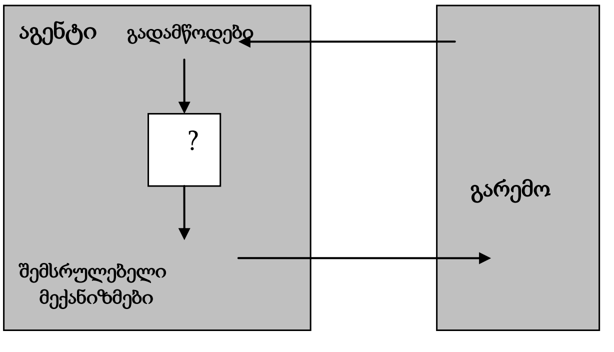
საკონტროლო კითხვები:

1. რას შეისწავლის ხელოვნური ინტელექტი?
2. რატომაა ხელოვნური ინტელექტის წარმოშობა დაკავშირებული კომპიუტერული ტექნიკის წარმოშობასთან?

3. რაში მდგომარეობს ა. ტიურინგის ტესტი?
4. რა ძირითად მიმართულებებს მოიცავს ხელოვნური ინტელექტი?
5. რომელ წელს შეიქმნა პირველი ევრისტიკული პროგრამა და რით განსხვავდებოდა ის მანამდე არსებული პროგრამებისაგან?
6. რას ეფუძნება ის ევრისტიკები, რომლებიც გამოიყენება LG-ში?
7. რატომ ვერ ხერხდება ევრისტიკული პროგრამირების მეთოდებით რთული ამოცანების ამოხსნა?
8. რას წარმოადგენს ხელოვნური ნეირონი და რა ფუნქციები შეიძლება აღიწეროს ნეირონული ქსელების საშუალებით?
9. რა პრინციპი უდევს საფუძვლად ფ. როზენბლატის მიერ შექმნილ პერცეპტრონს?
10. რა კომპონენტი არ არის გათვალისწინებული ევრისტიკულ პროგრამირებაში?
11. რა მიმართულებით განვითარდა ხელოვნური ინტელექტი გასული საუკუნის ოთხმოციანი წლებიდან?

ლექცია 2. ინტელექტუალურ აგენტთა თეორია.

2.1. ინტელექტუალური აგენტის ცნება. აგენტად შეიძლება ჩაითვალოს ყველაფერი, რაც მოქმედებს (სიტყვა აგენტი ლათინური სიტყვაა და ნიშნავს **agere** - იმოქმედე), მაგრამ იგულისხმება, რომ კომპიუტერული აგენტები განსხვავებით ჩვეულებრივი “პროგრამებისაგან” ფლობენ კიდევ სხვა თვისებებს. ასე მაგალითად, მათ შეუძლიათ ფუნქციონირება ავტონომიურად, აქვთ გარემოს აღქმის უნარი, არსებობენ ხანგრძლივი დროის განმავლობაში, ადაპტურები არიან ცვლილებების მიმართ და რაც ყველაზე უფრო მთავარია შეუძლიათ თავის თავზე აიღონ სხვების მიერ დასმული მიზნების მიღწევა.



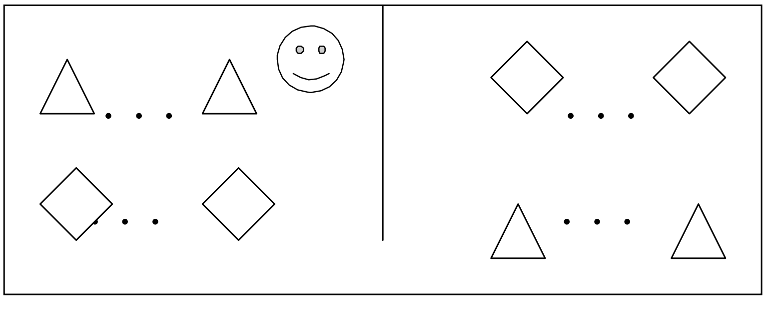
სურ. #2.1. აგენტის ურთიერთქმედება გაერმოსთან

რაციონალურია აგენტი, თუ მას შეუძლია მიაღწიოს საუკეთესო შედეგებს დეტერმინირებულ გარემოში, ან თუ გარემო არაა დეტერმინირებული – საუკეთესო მოსალოდნელ შედეგებს. რაციონალურობის კონცეფცია შეგვიძლია გამოვიყენოთ ნებისმიერი აგენტის მიმართ, რომელიც მოქმედებს ნებისმიერ გარემოში, რომლის წარმოდგენაც კი ჩვენ შეგვიძლია. თავის მხრივ აგენტი შეიძლება იყოს ნებისმიერი სისტემა, რომელიც აღიქვამს გარემოს გადამწოდების საშუალებით და ზემოქმედებს ამ გარემოზე შემსრულებელი მექანიზმების საშუალებით. თუ განვიხილავთ ადამიანს აგენტის როლში, მას გააჩნია გადამწოდები - თვალები, ყურები და გრძნობის სხვა ორგანოები, რომელთა საშუალებითაც ის აღიქვამს გარემოს და ზემოქმედებს ამ გარემოზე ხელებით, ფეხებით და სხეულის სხვა ნაწილებით, რომლებიც წარმოადგენენ მის შემსრულებელ მექანიზმებს. რობოტს, რომელიც ასრულებს აგენტის როლს, გადამწოდებად შეიძლება ჰქონდეს ვიდეოკამერა, ინფრა-

წითელი მანძილმზომი და სხვა. მის შემსრულებელ მექანიზმებს კი წარმოადგენენ მოტორები, მექანიკური ხელი და სხვა. პროგრამული უზრუნველყოფაც შეიძლება წარმოვიდგინოთ ინტელექტუალური აგენტის როლში. ამ შემთხვევაში გადამწოდის როლს ასრულებს კლავიატურა და ინფორმაციის შეყვანის სხვა საშუალებები, ხოლო გარემოზე ზემოქმედება გამოიხატება მონაცემების გამოყვანით ეკრანზე, ფაილზე ჩაწერით და ასე შემდეგ.

2.2. აღქმა, ფუნქცია, პროგრამა. დავიწყოთ აგენტის მიერ ინფორმაციის მიღების მექანიზმებით. ტერმინს აღქმა ვიყენებთ იმისთვის, რომ აღვნიშნოთ სენსორული ინფორმაციის მიღება დროის ნებისმიერ კონკრეტულ მომენტში. აღქმათა აქტების მიმდევრობა ეწოდება იმ სრულ ისტორიას, რაც საერთოდ როდესმე აღიქვა აგენტმა. ზოგადად აგენტის მიერ დროის ნებისმიერ მომენტში არჩეული ქმედება შეიძლება დამოკიდებული იყოს აღქმათა აქტების მთელ მიმდევრობაზე დასაწყებული აგენტის პირველი ქმედებიდან მოცემულ მომენტამდე. თუ შესაძლებელია დავადგინოთ რომელ ქმედებას აირჩევს აგენტი ნებისმიერ შესაძლო აღქმათა აქტების მიმდევრობაზე, მაშინ შესაძლებელია აგენტის მეტნაკლებად ზუსტი განსაზღვრა. მათემატიკურად ეს ნიშნავს, რომ რომელიღაც აგენტის ქცევა შეიძლება აღიწეროს აგენტის ფუნქციის საშუალებით, რომელიც აღქმის აქტების ნებისმიერ კონკრეტულ მიმდევრობას ასახავს რომელიმე ქმედებაზე. აგენტის ფუნქციის დასადგენად შესაძლებელია შევადგინოთ ცხრილი, რომელიც აღწერს ამ ფუნქციას, მაგრამ ეს ცხრილი იქნება პრაქტიკულად უსასრულო, თუ არ დავადგენთ აღქმათა აქტების მიმდევრობის მაქსიმალურ ზომას. თავისთავად ასეთი ცხრილის დადგენა არაა ცუდი, მაგრამ ეს იქნება აგენტის გარეგნული აღწერა. შინაგანი აღწერა მდგომარეობს იმის განსაზღვრაში, თუ აგენტის რა ფუნქცია ხორციელდება აგენტის პროგრამის საშუალებით. აგენტის ფუნქციისაგან განსხვავებით, რომელიც წარმოადგენს აგენტის აბსტრაქტულ, მათემატიკურ აღწერას, აგენტის პროგრამა არის კონკრეტული რეალიზაცია, რომელიც მოქმედებს აგენტის არქიტექტურულ ჩარჩოებში.

განვიხილოთ ძალიან მარტივი მაგალითი: დავუშვათ გარე სამყარო შედგება მხოლოდ ორი (**A** და **B**) ოთახისაგან, რომლებშიც დგას ოთხკუთხა და სამკუთხა პირამიდები. ამ გარემოში შეიძლება იმყოფებოდეს აგენტი, რომელსაც შეუძლია აღიქვას რომელ ოთახში იმყოფება თვითონ, დგას თუ არა ამ ოთახში პირამიდები და გაარჩიოს ერთმანეთისაგან ოთხკუთხა და სამკუთხა პირამიდები. შემსრულებელი მექანიზმების საშუალებით აგენტს შეუძლია იმოძრაოს გარემოში, აიღოს და გადაიტანოს ერთი ადგილიდან მეორეზე პირამიდები. აგენტის მიზანია **A** ოთახში შეაგროვოს მხოლოდ ოთხკუთხა და **B** ოთახში მხოლოდ სამკუთხა პირამიდები. ასეთი აგენტის ერთერთი უმარტივესი ფუნქცია შეიძლება მდგომარეობდეს შემდეგში: თუ არის **A** ოთახში სამკუთხა პირამიდა, გადაიტანოს ის **B** ოთახში, თუ არ არის **A** ოთახში სამკუთხა პირამიდა, გადავიდეს **B** ოთახში და ჩაატაროს იგივე პროცედურა ოთხკუთხა პირამიდებისათვის.



სურ. #2.2. მარტივი აგენტის მაგალითი

თუ შევადგენთ აღქმათა აქტების მიმდევრობის ცხრილს ასეთი აგენტისთვის გვექნება ცხრილი #2.1.

ცხრილი #2.1.

აღქმათა აქტების მიმდევრობა	ქმედება
[A, არ არის სამკუთხა პირამიდა]	გადადი B ოთახში
[A, არის სამკუთხა პირამიდა]	გადაიტანე ის B ოთახში
[B, არ არის ოთხკუთხა პირამიდა]	გადადი A ოთახში
[B,არის ოთხკუთხა პირმიდა]	გადაიტანე ის A ოთახში
[A, არ არის სამკუთხა პირამიდა],	გადადი B ოთახში
....	
[A, არ არის სამკუთხა პირამიდა],	გადადი B ოთახში
....	

როგორც ამ ცხრილიდან ჩანს, შესაძლებელია მისი სხვადასხვანაირად შევსება, რაც თავისთავად გამოიწვევს აგენტის სხადასხვა ქცევას. მაშინ ცხადია დგება კითხვა თუ როგორ უნდა შევავსოთ ცხრილი, რომ აგენტის ქცევა იყოს ინტელექტუალური?

2.3. რაციონალური აგენტი. ჩვენ ძალიან მარტივად განვსაზღვერთ ხელოვნური ინტელექტუალური სისტემის ცნება დავაკავშირეთ რა ის ბუნებრივ ინტელექტთან და ვთქვით, რომ ხელოვნურ სისტემას ვუწოდოთ ინტელექტუალური, თუ იგივე ქცევა, შესრულებული ბუნებრივი ინტელექტის მიერ ჩაითვლება ინტელექტუალურად. მაგრამ ახლა უკვე დაისმის კითხვა, თუ როგორი ქცევა შეიძლება ჩაითვალოს ინტელექტუალურ ცნებად? აქ შემოდის ინტელექტუალური ქცევისადმი ასეთი მიდგომის ყველაზე არსებითი ცნება – რაციონალურობის ცნება. რაციონალური აგენტი არის ისეთი აგენტი, რომელიც ასრულებს სწორ ქმედებებს, ანუ მის ფუნქციათა ცხრილში ყოველი ჩანაწერი არის სწორი. ცხადია რომ სწორი ქმედებების შესრულება უნდა სჯობდეს არასწორი ქმედებების შესრულებას, მაგრამ რომელია სწორი და რომელი არასწორი ქმედება? ამ პრობლემიდან თავის დასაღწევად შემოგვაქვს კიდევ ერთი არაფორმარული ცნება - აგენტის წარმატებული ფუნქციონირების ცნება. იმისათვის კი, რომ შევაფასოთ, წარმატებულია თუ არა აგენტის ფუნქციონირება, საჭიროა რაიმე ფორმალური კრიტერიუმები, თორემ ყველა ეს განმატებები იქნება ცარიელი წყლის ნაყვა.

მაშასადამე, შეიძლება ვთქათ, რომ გარემოს, გადამწოდების, შემსრულებელი მექანიზმებისა და წარმატების კრიტერიუმების აღწერა წარმოადგენს იმ ამოცანის სრულ სპეციფიკაციას, რომლის გადაჭრაც უხდება აგენტს. როდესაც ჩვენ გვექნება ყველა ეს კომპონენტი, შევძლებთ უფრო ზუსტად განვმარტოთ რა იგულისხმება რაციონალურობის ქვეშ.

2.4. მუშაობის მახსიათებლები. აგენტის წარმატებული ქცევის შეფასების კრიტერიუმებს წარმოდგენენ მისი მუშაობის მახასიათებლები. მას შემდეგ, რაც აგენტს მოვათავსებთ გარემოში, ის გამოიმუშავებს ქმედებათა მიმდევრობახ მიღებულ აღქმათა შესაბამისად. ქმედებათა ეს მიმდევრობა აიძულებს გარემოს გაიაროს რაღაც მდგომარეობები. თუ ეს მიმდევრობა შეესაბამება სასურველს, მაშინ ცხადია აგენტი მუშაობს კარგად. შეესაბამება თუ არა მიღებული შედეგი სასურველს? ამის შესახებ შეიძლებოდა გვეკითხა თვით აგენტისათვის, მაგრამ ზოგიერთი აგენტი საერთოდ ვერ შეძლებს ამ კითხვაზე პასუხის გაცემას, ხოლო ზოგიერთმა მათგანმა შეიძლება ან თავი მოიტყუოს ან ჩვენ მოგვატყუოს რომ სწორედ ამ მიზნის მიღწევა სურდა. (გავიხსენოთ მელიის და ყურძნის არაკი). აქედან გამომდინარე განსაკუთრებით მნიშვნელოვანია თუ რა კრიტერიუმებით შევაფასებთ აგენტის მიერ შესრულებულ სამუშაოს. დავუშვათ, ჩვენი მაგალითის შემთხვევაში კრიტერიუმად ავიღეთ გადატანილი პირამიდების რაოდენობა, რომელსაც აგენტი გადაიტანს ერთი ოთახიდან მეორეში რვა საათიანი სამუშაო დღის განმავლობაში. მაშინ ჩვენი აგენტი შეიძლება ასე მოიქცეს: გადაიტანოს სამკუთხა პირამიდები **A** ოთახიდან **B** ოთახში და

ოთხკუთხა პირამიდები **B**-დან **A**-ში, შემდეგ კვლავ გადმოიტანს **B**-დან **A**-ში სამკუთხა პირამიდებს და ასე შემდეგ. ჩვენი კრიტერიუმის მიხედვით ის იქნება წარმატებული აგენტი და უნდა მიიღოს შესაბამისი ანაზღაურება. ამ მაგალითიდან კარგად ჩანს, რომ ასეთ მარტივ შემთხვევაშიც კი აუცილებელია აგენტის ქმედების შეფასების კრიტერიუმების სწორად შერჩევა, თუ გვინდა რომ მივიღოთ მართლაც რაციონალური ქმედება დასახული მიზნის მისაღწევად. აგენტის ქმედების რაციონალურობას დროის ნებისმიერ მომენტში განსაზღვრავს შემდეგი ოთხი ფაქტორი:

- მუშაობის მახასიათებლები, რომლებიც განსაზღვრავენ წარმატების კრიტერიუმებს;
- აგენტის ცოდნა გარემოს შესახებ, რომელიც მან მიიღო ადრე;
- ქმედებები, რომელთა შესრულებაც შეუძლია აგენტს;
- აგენტის მიერ აღქმული აქტების მიმდევრობა დროის მოცემულ მომენტამდე.

ყველა ამ ფაქტორის გათვალისწინებით შეგვიძლია ჩამოვყალიბოთ რაციონალური აგენტის განმარტება. აღქმის აქტების ყველა შესაძლო მიმდევრობებისათვის აგენტმა უნდა ამოირჩიოს ის ქმედება, რომელიც, როგორც მოსალოდნელია, მოახდენს მისი მუშაობის მახასიათებლების მაქსიმიზაციას მოცემული აქტების მიმდევრობით მიღებული ფაქტებისა და აგენტში ჩამონტაჟებული მთელი ცოდნის გათვალისწინებით.

2.5. რაციონალურობის საზღვრები. ძალიან მნიშვნელოვანია განვასხვაოთ ერთმანეთისგან რაციონალურობა და ყველაფრის ცოდნა. ყველაფრის მცოდნე აგენტმა ფაქტობრივად იცის თავისი ქმედების შედეგი (რაც ზოგიერთ გარემოში შეუძლებელია) და შეუძლია იმოქმედოს შესაბამისად. განვიხილოთ მაგალითი. დავუშვათ, აგენტმა სეირნობის დროს გადაწყვიტა გადავიდეს ქუჩის მეორე მხარეს. ის ხედავს რომ ქუჩაში არ მოძრაობს არცერთი ავტომობილი და როგორც რაციონალურმა აგენტმა, დაიწყო ქუჩის გადაკვეთა. ამ მომენტამდე ცოტა ადრე კი 10 000 მეტრის სიმაღლეზე თვითმფრინავს მოსწყდა კარები, რომელიც დაეცა ჩვენს აგენტს. იყო თუ არა აგენტის ქმედება რაციონალური? ალბათ არცერთი ადამიანი ქუჩაზე გადასვლის დროს არ იხედება მაღლა და არ ეძებს საშიშროებას ზევიდან. ეს მაგალითი გვიჩვენებს თუ რა განსხვავებაა რაციონალურ ქმედებასა და ყველაფრის ცოდნას შორის.

მეორე შემთხვევაში, თუკი აგენტი ისე გადადის ქუჩაზე, რომ წინასწარ არ გაიხედავს მარცხნივ და მარჯვნივ, მის მიერ აქამდე აღქმული აქტების მიმდევრობა ვერ უკარნახებს მას, რომ გადასასვლელს დიდი სისწრაფით უახლოვდება სატვირთო მანქანა. რაციონალურ აგენტს უნდა შეეძლოს ისეთ ქმედებათა შესრულება, რომელიც საშუალებას მისცემს მას შესაძლებლობის ფარგლებში წინასწარ განსაზღვროს მოსალოდნელი შედეგი. ანუ მას უნდა შეეძლოს შეაგროვოს ინფორმაცია გარემოს შესახებ და აგრეთვე გამოიკვლიოს უცნობი სიტუაციები.

2.6. პრობლემური გარემო. რაციონალური აგენტის განმარტება გულისხმობს აგრეთვე, რომ აგენტი არა მარტო უნდა აგროვებდეს ინფორმაციას გარემოს შესახებ, არამედ სწავლობდეს კიდეც. სწავლის ქვეშ იგულისხმება რომ აგენტის საწყისი კონფიგურაცია შეიძლება შეიცავდეს რაიმე წინასწარ ცოდნას გარემოს შესახებ, მაგრამ თანდათანობით, აგენტის გამოცდილებასთან ერთად ეს ცოდნა იცვლება და განიცდის მოდიფიცირებას. არსებობს განსაკუთრებული შემთხვევები, როდესაც გარემო მთლიანად ცნობილია. ასეთ შემთხვევაში აგენტს აღარ სჭირდება არც ინფორმაციის შეგროვება და არც დასწავლა, მაგრამ ასეთი აგენტი იქნება ძალიან უსუსური. საკმარისია სულ პატარა ცვლილება გარემოში, რომ აგენტის ქმედებები უკვე აღარ იქნება რაციონალური. როდესაც ჩვენ ვიხილავდით აგენტს, რომელსაც პირამიდები გადაჰქონდა ერთი ოთახიდან მეორეში, საჭირო გახდა მისი მუშაობის მახასიათებლების, გარემოს, გადამწოდებისა და შემსრულებელი მექანიზმების განსაზღვრა. გავაერთიანოთ ამ ფაქტორების აღწერა პრობლემური გარემოს ცნებაში. განვიხილოთ ახალი მაგალითი – ტაქსის ავტომატიზებული მძღოლის შექმნის პრობლემა. მანქანის ტარების პრობლემის სრული გადაჭრა ძალიან რთული ამოცანაა, რადგანაც უსაზღვროა იმ ყველანაირ სიტუაციათა რაოდენობა, რომლებიც შეიძლება წარმოიშვას მანქანის ტარების დროს და ამიტომ ყველაფერს წინასწარ ვერ გაითვალისწინებ.

პირველ რიგში განვსაზღვროთ მუშაობის მახასიათებლები, რომლებიც განსაზღვრავენ ტაქსის ავტომატიზებული მძღოლის მოქმედებას მთლიანობაში: სასურველია წარმატებით მიაღწიოს დანიშნულების ადგილს; მოახდინოს საწვავის, ავტომანქანის დეტალების ცვეთის, მგზავრობის ხანგრძლივობის და/ან მგზავრობის ღირებულების მინიმიზაციას; იმოძრაოს ისე, რომ არ დაარღვიოს მოძრაობის წესები და არ შეუქმნას პრობლემები სხვა მძღოლებს; შეუქმნას მაქსიმალური უსაფრთხოება და კომფორტი მგზავრებს; მიაღწიოს მაქსიმალურ მოგებას. ცხადია, რომ ამ მიზნებიდან ზოგიერთი არაა მკაფიოდ ჩამოყალიბებული და აგრეთვე ზოგიერთი მათგანი მოდის წინააღმდეგობაში სხვებთან, ამიტომ საჭიროა რაღაც კომპრომისის გამონახვა.

განვიხილოთ, რისგან შედგება გარემო, რომელშიც მოძრაობს მანქანა. პირველ რიგში ესაა გზები დაწყებული ავტობანებიდან დამთავრებული სოფლის ორღობეებით. ამ გზებზე მძღოლს შეიძლება შეხვდნენ სხვა სატრანსპორტო საშუალებები, ფეხით მოსიარულენი, ცხოველები, მუშები, რომლებიც აწარმოებენ სარემონტო სამუშაოებს, საპატრულო პოლიცია. ტაქსის მძღოლს საქმე აქვს პოტენციურ ან რეალურ მგზავრებთან. მან უნდა გაითვალისწინოს აგრეთვე მეტეოროლოგიური პირობები. ავტომატიზებულ მძღოლს დასჭირდება შემსრულებელი მექანიზმები, რომელთა საშუალებითაც ის შესძლებს მართოს მანქანა და იქონიოს ურთიერთობა როგორც მგზავრებთან, ასევე მოძრაობის სხვა მონაწილეებთან. ამ მიზნების მისაღწევად მას დასჭირდება გადამწოდები. მათ შორის ერთი ან რამდენიმე ვიდეოკამერა, სპიდომეტრი, ოდომეტრი, აქსელომეტრი და ყველა ის სისტემები, რომლებიც შეატყობინებენ მანქანის ტექნიკურ გამართულობას. გარდა ამისა მას დასჭირდება აგრეთვე გარემოს ელექტრონული რუკა და ნავიგაციის გლობალური თანამგზავრული სისტემა თავისი ადგილმდებარეობის განსასაზღვრავად. აი ასეთი შეიძლება იყოს ავტომატიზებული მძღოლის პრობლემური გარემო. საერთოდ ცხადია, რომ რაც უფრო რთულია ამოცანა, რომელიც დგას აგენტის წინაშე, მით უფრო რთულია პრობლემური გარემოც. შევეცადოთ, დავახასიათოთ, როგორი შეიძლება იყოს პრობლემური გარემო.

1. **მთლიანად დაკვირვებადი ან ნაწილობრივ დაკვირვებადი.** თუკი გადამწოდები აგენტს აწვდიან სრულ ინფორმაციას გარემოს შესახებ დროის ნებისმიერ მომენტში, ასეთი გარემო იქნება მთლიანად დაკვირვებადი. სრული ინფორმაციის ქვეშ ვგულისხმობთ იმ ინფორმაციას, რომელიც აგენტს სჭირდება გადაწყვეტილების მისაღებად. ასეთი გარემო მოსახერხებელია იმით, რომ აგენტს არ სჭირდება შეინახოს თავის მეხსიერებაში გარემოს მდგომარეობა. გარემო შეიძლება იყოს ნაწილობრივ დაკვირვებადი იმ შემთხვევაში, როდესაც წარმოიშვება ხმაური ან როდესაც გადამწოდები ვერ აწვდიან ზუსტ ინფორმაციას აგენტს. ნაწილობრივ დაკვირვებადი იქნება გარემო მაშინაც, როდესაც აგენტს არ გააჩნია ინფორმაცია გარემოს რომელიმე მახასიათებელზე. მაგალითად, ჩვენს მიერ განხილულ აგენტს, რომელსაც გადააქვს პირამიდები ერთი ოთახიდან მეორეში, გააჩნია მხოლოდ ლოკალური გადამწოდი, და ამიტომ არ შეუძლია განსაზღვროს არის თუ არა სამკუთხა პირამიდა მეორე ოთახში, ხოლო ავტომატიზებულ მძღოლს არ გააჩნია ინფორმაცია რა მანევრის შესრულებას აპირებენ მოძრაობის სხვა მონაწილენი.

2. დეტერმინირებული ან სტოქასტური. თუ გარემოს შემდეგი მდგომარეობა მთლიანად განსაზღვრულია მიმდინარე მდგომარეობით და აგენტის ქმედებით, მაშინ გარემო დეტერმინირებულია, წინააღმდეგ შემთხვევაში კი სტოქასტური. მთლიანად დაკვირვებად გარემოში აგენტს არ უხდება იმოქმედოს განუზღვრელობის პირობებში, მაგრამ თუ გარემო ნაწილობრივ დაკვირვებადია, მაშინ იქმნება შთაბეჭდილება, რომ გარემო სტოქასტურია, რადგან აგენტს არ შეუძლია დააკვირდეს მის ყველა მახასიათებელს. ამიტომ ზოგჯერ უფრო მოსახერხებელია ვთქვათ, რომ გარემო დეტერმინირებულია ან სტოქასტურია აგენტის თვალსაზრისით. აშკარაა, რომ ასეთი განმარტების შემთხვევაში აგენტ – ტაქსისტის გარემო სტოქასტურია, რადგან მას არ შეუძლია ზუსტად განსაზღვროს სხვა მონაწილეთა ქცევა, ან შეიძლება მის ავტომობილს მოულოდნელად რაიმე გაუფუჭდეს.

3. ეპიზოდური ან მიმდევრობითი. ეპიზოდურ პრობლემურ გარემოში აგენტი კონკრეტულ ეპიზოდს აღიქვამს, ღებულობს შესაბამის გადაწყვეტილებას და მოქმედებს. არსები-

თია, რომ მიღებული გადაწყვეტილება არანაირ გავლენას არ ახდენს შემდეგ ეპიზოდზე. მაგალითად, დავუშვათ აგენტი ახდენს დეტალების გადაჩევას კონვეირზე. აგენტი აკვირდება დეტალს რომელიც მის წინაშეა და თუ მას გააჩნია დეფექტი, გადადებს მას კონვეირიდან. ეს გადაწყვეტილება ვერანაირად ვერ განსაზღვრავს იქნება თუ არა შემდეგი დეტალი დეფექტით. ასეთ გარემოში აგენტს უადვილდება მოქმედება, რადგან არაა საჭირო გაითვალისწინოს რა გავლენას მოახდენს მისი გადაწყვეტილება პრობლემურ გარემოზე. განსხვავებით ასეთი გარემოსგან, მიმდევრობით გარემოში ყოველი გადაწყვეტილება მოქმედებს პრობლემური გარემოს შესაძლო მდგომარეობაზე და აგენტს უხდება ამის გათვალისწინება. ასეთი გარემოს მაგალითია ჭადრაკის თამაში.

4. სტატიკური ან დინამიური. თუ პრობლემური გარემო შეიძლება შეიცვალოს იმ პერიოდში, როდესაც აგენტი ირჩევს თავის ქმედებას, ასეთ გარემოს ეწოდება ამ აგენტისათვის დინამიური გარემო, წინააღმდეგ შემთხვევაში პრობლემური გარემო სტატიკურია. სტატიკურ გარემოში მოქმედება უფრო მარტივია, რადგან აგენტს არ სჭირდება გადაწყვეტილების მიღების მომენტში აკვირდებოდეს გარემოს და აგრეთვე არაა შეზღუდული დროში. თუ დროთა განმავლობაში გარემო არ იცვლება, მაგრამ იცვლება აგენტის მუშაობის მახასიათებლები, მაშინ ასეთ გარემოს უწოდებენ ნახევრად დინამიურ გარემოს.

5. დისკრეტული ან უწყვეტი. განსხვავება დისკრეტულ და უწყვეტ გარემოს შორის შეიძლება ეხებოდეს გარემოს მდგომარეობებს, დროის აღრიცხვის მეთოდებს და აგრეთვე აგენტის მიერ გარემოს აღქმასა და ქმედებებს. მაგალითად, ჭადრაკის თამაშის გარემო დისკრეტულია როგორც გარემოს მდგომარეობებით, აგრეთვე აღქმისა და ქმედებების მიხედვით, განსხვავებით ტაქსის ტარების გარემოსგან, რომელშიც გარემოს მდგომარეობები, დრო, აღქმა და ქმედებებიც უწყვეტია.

6. ერთაგენტიანი და მულტიაგენტიანი. განსხვავება ერთ და მულტიაგენტიან გარემოებებს შორის ერთი შეხედვით ძალიან მარტივია. მაგალითად, კროსვორდის ამოხსნის გარემო ერთაგენტიანი გარემოა, ხოლო ჭადრაკის თამაშის გარემო კი ორაგენტიანი, თუმცა აქაც არსებობს გარკვეული ნიუანსები. ჩვენ აღვწერეთ რა ნიშნების საფუძველზე შეიძლება ესა თუ ის არსი შეიძლება ჩავთვალოთ აგენტად, მაგრამ არ მიგვითითებია რომელი არსი უნდა განვიხილოთ როგორც აგენტი. უნდა ჩათვალოს თუ არა ტაქსის მძღოლმა სხვა სატრანსპორტო საშუალებების მძღოლები აგენტებად, თუ მოექცეს მათ როგორც უბრალოდ სტოქასტურად მოძრავ ობიექტებს. არსებითი განსხვავება მდგომარეობს იმაში, უნდა ჩაითვალოს თუ არა ობიექტის ქცევა ისეთ ფაქტორად, რომელიც გავლენას ახდენს მისი მუშაობის მახასიათებლებზე. ჭადრაკის თამაშში მოწინააღმდეგე ცდილობს რა მოახდინოს თავისი მუშაობის მახასიათებლების მაქსიმიზაცია, ავტომატურად იწვევს აგენტის მუშაობის მახასიათებლების მინიმიზაციას. ამიტომ ჭადრაკის თამაში არის კონკურენტული მულტიაგენტური გარემო. ტაქსის ტარების გარემოში კი დაჯახებათა თავიდან აცილება იწვევს ყველა აგენტის მუშაობის მახასიათებლების მაქსიმიზირებას, ამიტომ ასეთი გარემო არის ნაწილობრივ კოოპერაციული გარემო (ნაწილობრივ იმიტომ რომ სხვა პარამეტრებით ის შეიძლება იყოს კონკურენტული).

2.7. აგენტთა ტიპები. აქამდე ჩვენ აგენტების თვისებებს განვიხილავდით მხოლოდ მათი ქცევის ანალიზის მიხედვით. გადავიდეთ მათი ფუნქციონირების აღწერაზე. ხელოვნური ინტელექტის ამოცანა მდგომარეობს აგენტის პროგრამის შექმნაში, რომელიც ახდენს აგენტის ფუნქციის რეალიზაციას. იგულისხმება, რომ ეს პროგრამა უნდა მუშაობდეს გარკვეულ გამომთვლელ მოწყობილობაში ფიზიკური გადამწოდებით და შემსრულებელი მექანიზმებით. ამ კომპონენტებს მთლიანობაში უწოდებენ არქიტექტურას. აგენტის სტრუქტურა შეიძლება წარმოვადგინოთ შემდეგი სახით:

აგენტის სტრუქტრურა = არქიტექტურა + პროგრამა

ცხადია, რომ არჩეული პროგრამა უნდა შეესაბამებოდეს აგენტის არქიტექტურას. ასე მაგალითად, თუ პროგრამა გამოიმუშავებს ისეთ რეკომენდაციებს, როგორიცაა ‘‘სიარული’’, არქიტექტურაში გათვალისწინებული უნდა იყოს შესაბამისი მექანიზმი. ლექციათა კურსში ჩვენ განვიხილავთ მხოლოდ ისეთ პროგრამებს, რომელთა შესასვლელზე გადამწოდებისგან შემოდის მხოლოდ მოცემულ მომენტში აღქმული ინფორმაცია და მისი და-

მუშავების შემდეგ შემსრულებელ მექანიზმებს მიეწოდება ამორჩეული ქმედების ვარიანტი. ამით აგენტის პროგრამა განსხვავდება აგენტის ფუნქციისაგან, რომელიც თავის შესასვლელზე ღებულობს აღქმათა აქტების მთელ მიმდევრობას. აგენტისთვის ეს შეუძლებელია, ამიტომ თუ მას ქმედების შესასრულებლად სჭირდება აღქმათა აქტების გარკვეული მიმდევრობა, მან თვითონ უნდა დაიმახსოვროს ეს მიმდევრობა.

2.8. მარტივი რეფლექსური აგენტი. ასეთი აგენტები ირჩევენ ქმედებას მხოლოდ მიმდინარე აღქმის აქტის საფუძველზე და არ ითვალისწინებენ წინაისტორიას. განხილული აგენტი, რომელსაც ერთი ოთახიდან მეორეში გადააქვს პირამიდები წარმოადგენს სწორედ ასეთ აგენტს. მის პროგრამას აქვს შემდეგი სახე:

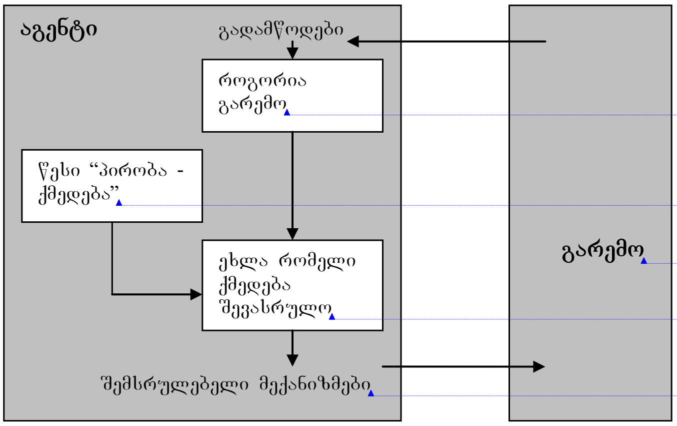
რეფლექსური აგენტის ფუნქცია [მდგომარეობა,სტატუსი] შესრულე ქმედება

ბიჯი 1: **თუ** მდგომარეობა = **A** ოთახი **და თუ** სტატუსი = არის სამკუთხა პირამიდა, **მაშინ შეასრულე:** გადაიტანე **B** -ში, **წინააღმდეგ შემთხვევაში** გადადი **B** ოთახში

ბიჯი 2: **თუ** მდგომარეობა = **B** ოთახი **და თუ** სტატუსი = არის ოთხკუთხა პირამიდა **მაშინ შეასრულე:** გადაიტანე A -ში, **წინააღმდეგ შემთხვევაში** გადადი ბიჯი 3

ბიჯი 3: გადადი **A** ოთახში

როგორც ვხედავთ, ეს პროგრამა გაცილებით პატარაა იმ ცხრილთან შედარებით, რომელიც აღწერდა აგენტის ფუნქციას. ეს შემცირება განპირობებულია იმით, რომ უგულვებელყოფილია აღქმათა აქტების ისტორია, რის გამოც ყველა შესაძლო ვარიანტების რიცხვი დავიდა ოთხამდე.



Formatted: Font: AcadNusx, 10 pt
Formatted: Font: AcadNusx, 10 pt
Formatted: Font: AcadNusx, Bold
Formatted: Font: AcadNusx, 10 pt
Formatted: Font: AcadNusx, 10 pt

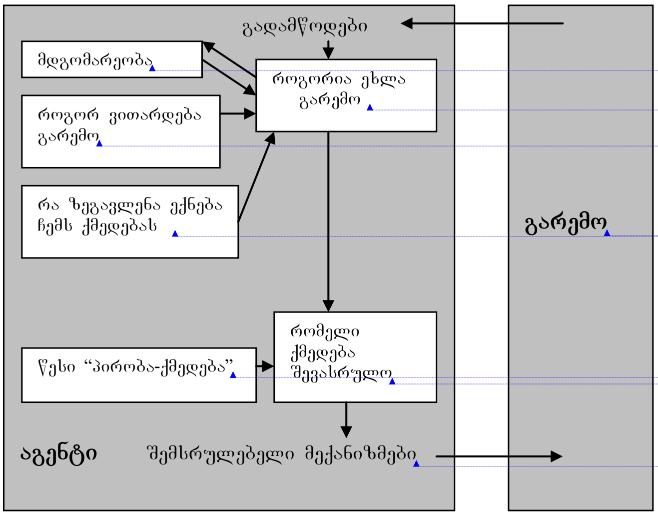
სურ. # 2.3. რეფლექსური აგენტის სტრუქტურა.

რეფლექსური აგენტის ქმედების საფუძველს წარმოადგენს პრინციპი: თუ სრულდება რაიმე პირობა, შეასრულე შესაბამისი ქმედება. შემოკლებით ეს შეგვიძლია აღვნიშნოთ როგორც წესი “პირობა – ქმედება”. ასეთი აგენტები მრავლად გვხვდება როგორც ტექნიკურ სისტემებში, ასევე ცოცხალ ორგანიზმებშიც. მათი ფუნქციონირება ძალიან მარტივია, მაგრამ მათ გააჩნიათ ძალიან შეზღუდული ინტელექტი (თუ გააჩნიათ საერთოდ). რეფლექსური აგენტის გადაწყვეტილებები მხოლოდ მაშინ იქნება სწორი, როდესაც მხოლოდ მიმდინარე აღქმის აქტი საკმარისია სწორი გადაწყვეტილების მისაღებად, ანუ თუ გარემო არის მთლიანად დაკვირვებადი. საკმარისია გარემო გახდეს ნაწილობრივ დაკვირვებადი და აგენტის ქმედებები აღარ იქნება სიტუაციის შესაბამისი. მაგალითად, განვიხილოთ ტაქსის ტარების პრობლემური გარემო. თუ წინ მიმავალი მანქანა იწყებს დამუხრუჭებას და ინთება მისი სამუხრუჭე ფარები, მძღოლმა ეს უნდა შენიშნოს და თვითონაც და-

იწყოს დამუხრუჭება. ანუ შეიძლება ჩამოვაყალიბოთ “პირობა – ქმედების” შემდეგი წესი: **თუ ავტო წინ – ამუხრუჭებს მაშინ დაიწყე დამუხრუჭება.**

დავუშვათ, ამ პირობის აღქმა ხდება მხოლოდ მიმდინარე აღქმის აქტით. რადგანაც ზოგიერთ ძველი მარკის ავტომობილს სამუხრუჭე ფარები შეთავსებული აქვს სხვა სასიგნალო ფარებთან, მიმდინარე აღქმის ფაქტით შეუძლებელია დავადგინოთ, ამუხრუჭებს თუ არა წინ მიმავალი მანქანა. მარტივი რეფლექსური აგენტი, რომელიც მისდევს ამ მანქანას, ან დაამუხრუჭებს მაშინ როდესაც ამის აუცილებლობა არ იქნება, ან რაც უფრო უარესია, საერთოდ არ დაამუხრუჭებს.

2.9. მოდელზე დაფუძნებული რეფლექსური აგენტები. ნაწილობრივ დაკვირვებადი გარემოს შემთხვევაში კარგი იქნება თუ აგენტი აკვირდება აგრეთვე მიმდინარე აღქმის აქტის ისტორიას. რაც ნიშნავს, რომ აგენტს უნდა გააჩნდეს გარკვეული შინაგანი მდგომარეობა, რომელიც დამოკიდებულია აღქმის აქტების ისტორიაზე და ამიტომ ასახავს მიმდინარე მომენტში ზოგიერთ დაუკვირვებად ასპექტს. ზემოთ განხილულ მაგალითში ამისათვის საკმარისია აგენტის შინაგანი მდგომარეობა ინახავდეს მხოლოდ წინა კადრს, რაც მისცემს აგენტს საშუალებას აღმოაჩინოს დამუხრუჭების დაწყების მომენტი. იმისათვის, რომ უზრუნველვყოთ ამ შინაგანი ინფორმაციის განახლება, საჭიროა აგენტის პროგრამაში შევიტანოთ ორი სახის ცოდნა. პირველი, აგენტმა უნდა იცოდეს თუ როგორ შეიცვლება გარემო მისი ჩარევის გარეშე და მეორე, რა ცვლილებებს გამოიწვევს მისი ქმედებები გარემოში. ასეთ ცოდნას უწოდებენ გარემოს მოდელს, ხოლო აგენტს, რომელშიც გამოიყენება ასეთი მოდელი, უწოდებენ მოდელზე დაფუძნებულ აგენტს.

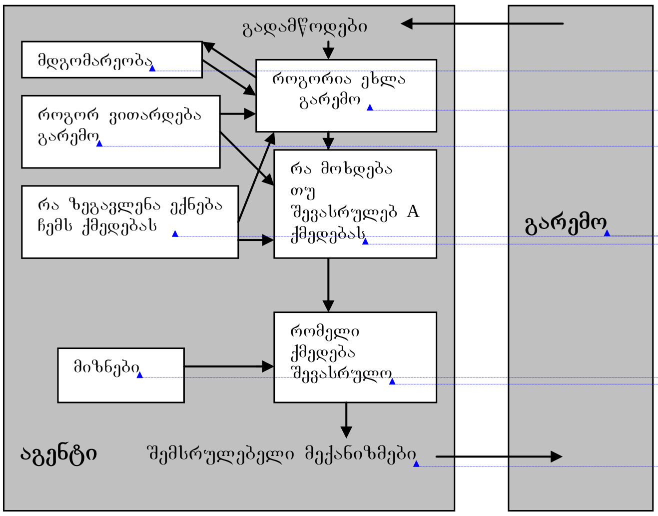


სურ. #2.4. მოდელზე დაფუძნებული რეფლექსური აგენტი

Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, Bold
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 10 pt

2.10. მიზანზე და მოდელზე დაფუძნებული აგენტები. გარემოს ცოდნა ხშირად არ არის საკმარისი იმისათვის რომ აგენტმა მიიღოს სწორი გადაწყვეტილება. თუ ტაქსი დგას გზაჯვარედინზე, სწორი გადაწყვეტილების მისაღებად აუცილებელია აგენტმა იცოდეს თუ სად უნდა მივიდეს ის. აგენტს სჭირდება არა მარტო გარემოს მიმდინარე მდგომარეობის ცოდნა, არამედ ინფორმაცია მიზნის შესახებ, რომელიც აღწერს პრობლემური გარემოს სასურველ სიტუაციებს. აგენტის პროგრამას შეუძლია ამ ინფორმაციისა და შესაძლო ქმედებათა შედეგების კომბინირება ისეთი ქმედების ამოსარჩევად, რომელიც შესაძლებლობას მისცემს მიაღწიოს მიზანს. ზოგჯერ მიზნის მიღწევა ძალიან ადვილია, რადგან მის მისაღწევად საკმარისია ერთი ქმედება. უფრო რთულ და რეალურ შემთხვევებში, ეს ასე არ არის. მაშინ აგენტის პროგრამაში გათვალისწინებული უნდა იყოს ისეთი პროცე-

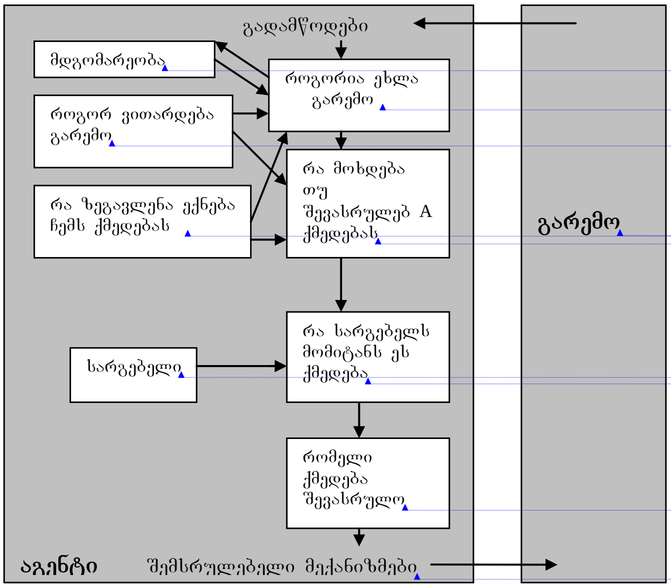
დურები, რომლებიც მოახდენენ მიზნის მებნას და ქმედებათა მიმდევრობის დაგეგმვას ამ მიზნის მისაღწევად.



სურ. #2.5. მიზანზე და მოდელზე დაფუძნებული აგენტის სტრუქტურა

Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, Bold
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 10 pt

ასეთ შემთხვევაში აგენტის გადაწყვეტილების მიღების პროცედურა არსებითად განსხვავდება რეფლექსური აგენტის გადაწყვეტილების მიღების პროცედურისაგან, რადგანაც აგენტს უხდება უპასუხოს ორ კითხვას: "რა მოხდება თუ მე მივიღებ ამ გადაწყვეტილებას?" და "მივაღწევ თუ არა ამ ქმედების შედეგად მიზანს?". რეფლექსურ აგენტებში ასეთი ინფორმაცია არაა წარმოდგენილი, რადგანაც მასში ჩამონტაჟებული წესები ამყარებენ პირდაპირ კავშირს გარემოს აღქმასა და ქმედებას შორის.



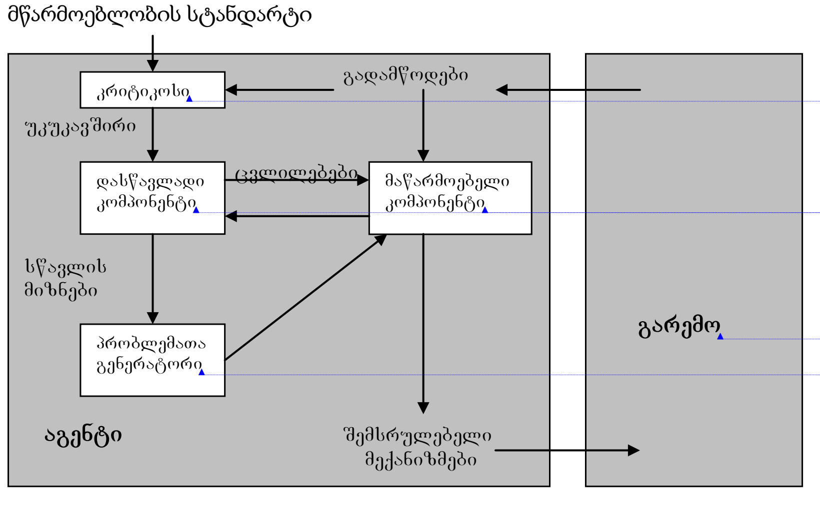
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, Bold
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 10 pt

სურ. #2.6. სარგებლიანობაზე დაფუძნებული აგენტის სტრუქტურა

2.11. სარგებლიანობაზე დაფუძნებული აგენტები. ხშირ შემთხვევაში კარგი გადაწყვეტილების მისაღებად საკმარისი არ არის მხოლოდ მიზნის მიღწევა. იგივე ტაქსის მართვის შემთხვევაში დანიშნულების ადგილს შეიძლება მივაღწიოთ სხვადასხვა გზით, რომელთაგან ზოგი გზა იქნება უფრო გრძელი და დაგვეხარჯება უფრო მეტი დრო და საწვავი, ზოგი კი უფრო მოკლე და შესაძლებელია დროისა და საწვავის დაზოგვა. ამიტომ საჭიროა აგენტმა ამოირჩიოს ისეთი ქმედებების მიმდევრობა, რომლებიც უკეთესად დააკმაყოფილებენ აგენტის მუშაობის მახასიათებლების კრიტერიუმებს. ის მდგომარეობა პრობლემური გარემოსი, რომელიც უფრო მიღწეული ზუსტად. ასეთ შემთხვევაში სარგებლიანობის ფუნქცია წარმოადგენს იმ მოსახერხებელ მეთოდს, რომელიც საშუალებას გვაძლევს მიზნების პრიორიტეტების გათვალისწინებით შევაფასით რომელიმე მიზნის მიღწევის ალბათობა.

2.12. დასწავლადი აგენტები. აქამდე ჩვენ ვიხილავდით აგენტების პროგრამებს, რომლებშიც გამოიყენება გადაწყვეტილების მიღების სხვადსახვა მეთოდები, მაგრამ არაფერი არ გვითქვამს, თუ როგორ უნდა შეიქმნას ასეთი პროგრამები. ჯერ კიდევ ა. ტიურინგმა გაანალიზა ასეთი პროგრამების შექმნის პრობლემა და მივიდა იმ დასკვნამდე, რომ საჭიროა რაიმე პროდუქციული მეთოდი, რომელიც გააადვილებს პროგრამების შექმნას. მანვე ასეთ მეთოდად მიიჩნია დასწავლადი მანქანის შექმნა და შემდეგ მათი სწავლება. დღეს სწორედ ასეთი მიდგომა დომინირებს ხელოვნურ ინტელექტში.

როგორც ნაჩვენებია სურ. #2.7. დასწავლადი აგენტის სტრუქტურა შედგება ოთხი კონცეპტუალური კომპონენტისაგან. ყველაზე მნიშვნელოვანია განსხვავება დასწავლად კომპონენტსა და მუშა კომპონენტს შორის. დასწავლადი კომპონენტი პასუხისმგებელია აგენტის მუშაობის გაუმჯობესებაზე, ხოლო მუშა კომპონენტი უზრუნველყოფს ქმედების შესრულებას. ფაქტობრივად, მუშა კომპონენტი არის ის, რასაც აქამდე განვიხილავდით როგორც აგენტს: ის ღებულობს აღქმით ინფორმაციას და იღებს გადაწყვეტილებას რა ქმედება უნდა განხორციელდეს. დასწავლადი კომპონენტი იყენებს რა ინფორმაციას, რომელსაც ის იღებს კრიტიკოსისაგან უკუკავშირის საშუალებით, რომელიც შეიცავს აგენტის მუშაობის შეფასებას, განსაზღვრავს თუ როგორ უნდა შეიცვალოს მუშა კომპონენტი რათა მისი მუშაობა გაუმჯობესდეს. აგენტის მეოთხე ძირითად კომპონენტს წარმოადგენს პრობლემათა გენერატორი, რომლის ამოცანას წარმოადგენს ისეთი ექსპერიმენტების შეთავაზება მუშა კომპონენტისათვის, რომელთა საფუძველზეც შესაძლებელი იქნება ახალი, უფრო უკეთესი ქმედებების აღმოჩენა.



სურ. N#2.7. სწავლებადი აგენტების ზოგადი სქემა.

Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, Bold
Formatted: Font: AcadNusx, 9 pt

საკონტროლო კითხვები:

1. რას ეწოდება ინტელექტუალური აგენტი?
2. როგორ აგენტს ეწოდება რაციონალური აგენტი?
3. აღწერეთ ინტელექტუალური აგენტის სტრუქტურა და მოიყვანეთ მაგალითები.
4. რა ფაქტორები განსაზღვრავენ აგენტის რაციონალურობას?
5. რა არის აგენტის ფუნქცია და აგენტის პროგრამა და რა განსხვავებაა მათ შორის?
6. გარდა ინფორმაციის შეგროვებისა, რა უნდა შეეძლოს რაციონალურ აგენტს?
7. რას წარმოადგენს პრობლემური გარემო?
8. რამდენ და რა ძირითად ტიპებად შეიძლება დაიყოს გარემო?
9. რისგან შედგება აგენტის სტრუქტურა?
10. აღწერეთ ყველა ტიპის აგენტების სტრუქტურები.

ლექცია 3. აგენტები, რომლებიც პრობლემებს წყვეტენ ძებნის საშუალებით

3.1. პრობლემების (ამოცანების) წარმოდგენა მდგომარეობათა სივრცეში. წინა ლექციაზე ჩვენ განვიხილეთ მიზანზე დამყარებული აგენტების კლასი, რომლებსაც შეუძლიათ თავისი მუშაობის მახასიათებლების მაქსიმიზაცია და აღვნიშნეთ, რომ თუ მიზანი არ არის ადვილად მიღწევადი, მაშინ აგენტის პროგრამაში გათვალისწინებული უნდა იყოს ისეთი პროცედურები, რომლებიც უზრუნველყოფენ მიზნის ძებნას. განვიხილოთ რა პროცედურების საშუალებით შეიძლება ამ ძებნის განხორციელება. ამ შემთხვევაში ჩვენ არ განვიხილავთ, თუ როგორ ხდება მიზნის ფორმირება. იგულისხმება, რომ აგენტი წინასწარ ახდენს მიზნის ფორმირებას, ან სხვა უსვამს მას მიზანს. უბრალოდ, მიზანი ჩვენთვის არის გარემოს მდგომარეობათა რაიმე სიმრავლე. აგენტის (ასეთი ტიპის აგენტებს უწოდებენ ამოცანათა ამომხსნელ აგენტებს) ამოცანაა განსაზღვროს ქმედებათა რა მიმდევრობა მიიყვანს მას მიზნის მდგომარეობამდე. მაგრამ სანამ ის დაიწყებს ამ ქმედებათა განხილვას, საჭიროა ჯერ დაზუსტდეს, თუ რა დონის ქმედებები იქნება მოცემული ამოცანისათვის მისაღები. ამ პრობლემის უკეთ გასაგებად განვიხილოთ ასეთი მაგალითი, დავუშვათ, აგენტი იმყოფება ბათუმში და მისი მიზანია მოხვდეს თბილისში. თუ ის დაიწყებს ქმედებების განხილვას იმ დონეზე, რომ ‘‘საჭიროა გადავდგა მარცხენა ფეხი 50 სანტიმეტრზე, შემდეგ მარჯვენა 60 სანტიმეტრზე და ასე შემდეგ’’, ან თუ აგენტი მოგზაურობს ავტომობილით - ‘‘მოვატრიალო საჭე ორი გრადუსით’’, მაშინ ცხადია ის ვერ შეძლებს ამოცანის გადაწყვეტას რეალურად.

3.2. ამოცანის ფორმულირება მდგომარეობს სწორედ იმ ქმედებებისა და მდგომარეობათა სიმრავლეების განსაზღვრაში, რომლებიც საშუალებას მოგვცემენ ყველაზე ოპტიმალურად მივაღწიოთ გარკვეულ მიზანს. ჩვენი მაგალითის შემთხვევაში, თუ დავუშვებთ, რომ აგენტი მოგზაურობს ავტომობილით, ალბათ ყველაზე სწორი იქნება თუ ის ქმედებებად განიხილავს ავტომობილით გადაადგილებას ერთი ქალაქიდან მეორეში. მაშინ აგენტისათვის მდგომარეობები იქნება მისი ყოფნა სხვადასხვა კონკრეტულ ქალაქებში, ქმედებები კი გადასვლა ერთი ქალაქიდან მეორეში.

დავუშვათ, აგენტი არ იცნობს საქართველოს გეოგრაფიას, ბათუმიდან კი გამოდის რამდენიმე გზა, რომლებსაც მივყავართ სხვადასხვა ქალაქებში. მაშინ, თუ აგენტს არ ექნება დამატებითი ინფორმაცია, ის ალალბედზე აირჩევს რომელიმე გზას და თუ ეს არჩევანი არ იქნება სწორი, ის თბილისის მაგივრად შეიძლება მოხვდეს ერზრუმში. დავუშვათ, ახლა მივაწოდეთ აგენტს დამატებითი ინფორმაცია საქართველოს რუკის სახით. ასეთ შემთხვევაში აგენტს შეუძლია განიხილოს ყველა გადასვლა ბათუმიდან სხვა ქალაქებში, შემ-

დეგ ამ ქალაქებიდან შემდეგ ქალაქებში და აღმოაჩინოს იმ ქმედებათა მიმდევრობა, რომელიც ჩაიყვანს მას ბათუმიდან თბილისში.

როგორ პრობლემურ გარემოში უხდება მუშაობა ამოცანათა ამომხსნელ აგენტს? ცხადია, რომ გარემო არის სტატიკური, რადგანაც ამოცანის ამოხსნის პროცესში არ არის გათვალისწინებული მისი ცვლილება (მაგალითად, რომ შეიძლება ხიდი ჩაინგრეს, ან გზა გადაიკეტოს, ან რომელიმე ქალაქი გაქრეს მთლიანად). ცნობილია აგრეთვე აგენტის საწყისი მდგომარეობაც (საიდან უნდა დავიწყოთ ქმედება), ესე იგი გარემო მთლიანად დაკვირვებადია. რადგანც გარემოს აღწერაში შედიან მხოლოდ ქალაქები და მათ შორის დამაკავშირებელი გზები და ჩვენ ვგულისხმობთ, რომ გადასვლა ერთი ქალაქიდან მეორეში ხდება მყისიერად, ცხადია, რომ გარემო არის დისკრეტული. ასეთ გარემოში ამოცანათა ამონახსნები წარმოადგენენ ქმედებათა ერთადერთ მიმდევრობებს და მათში არ შეიძლება გათვალისწინებული იყოს რაიმე მოულოდნელი შემთხვევები. უფრო მეტიც, აგენტი გადაწყვეტილების მიღების დროს არ ითვალისწინებს რაიმე აღქმის შედეგებს, რა იმას ნიშნავს, რომ გარემო არის დეტერმინირებული. ყოველივე ზემოთთქმული ნიშნავს რომ საქმე გვაქვს უმარტივეს გარემოსთან.

მდგომარეობათა გარემო ასეთ შემთხვევაში წარმოადგენს გრაფს, რომლის წვეროები აღნიშნავენ მდგომარეობებს, ხოლო რკალები შეესაბამებიან იმ ქმედებებს, რომელთა საშუალებითაც შესაძლებელია გადასვლა ერთი მდგომარებიდან მეორეში. ძებნის ამოცანა ასეთ შემთხვევაში შეიძლება განვსაზღვროთ შემდეგი ხუთი პარამეტრის საშუალებით: $(\{S_0\},\{S\},\{F\},\{T\},w(T))$, სადაც $\{S_0\}$ არის საწყის მდგომარეობათა სიმრავლე, ანუ გრაფის იმ წვეროების სიმრავლე, რომელშიც შეიძლება იმყოფებოდეს აგენტი ამოცანის ამოხსნის დაწყებისას ($S_0 \subset S$).

$\{F\}$ არის იმ ქმედებათა სიმრავლე, რომელიც ხელეწიფება აგენტს. სხვანაირად ეს არის შვილობილ წვეროთა განმსაზღვრელ ფუნქციათა სიმრავლე, რომელთა საშუალებითაც ერთი მდგომარეობიდან შეიძლება მივიღოთ მეორე მდგომარეობა $f_i(S_k) = S_l$. ყოველი ასეთი f_i ფუნქციის განსაზღვრის არეა $S_k \subset S$ სიმრავლე და მნიშვნელობათა სიმრავლეა $S_l \subset S$ სიმრავლე. ამ სიმრავლეს ხშირად უწოდებენ აგრეთვე ოპერატორთა სიმრავლეს. $\{S\}$ არის მდგომარეობების სივრცე, ანუ ყველა იმ მდგომარეობათა სიმრავლე, რომელიც შეიძლება შეგვხვდეს მოცემულ ამოცანაში. ეს სიმრავლე არაცხადად განისაზღვრება საწყისი მდგომარეობისა და ოპერატორთა სიმრავლის საშუალებით. გზა მდგო-მარეობათა სივრცეში ეწოდება იმ მდგომარეობათა მიმდევრობას, რომელიც იწყება სა-წყისი მდგომარეობით და ერთმანეთთან დაკავშირებულია დასაშვებ ოპერატორთა მიმდევრობით.

$\{T\} \subset \{S\}$ მიზნის მდგომარეობების სიმრავლე – წარმოადგენს იმ მდგომარეობებს, რომელთაგან ერთერთში მოხვედრა შეესაბამება პრობლემის გადაწყვეტას, ანუ ამოცანის ამოხსნას.

$w(T)$ არის ფუნქცია, რომელიც ყოველ გზას საწყისი მდგომარეობიდან მიზნის მდგომარეობამდე შეუსაბამებს რაიმე რიცხვით მნიშვნელობას. მას უწოდებენ გზის ღირებულების ფუნქციას. აგენტი ირჩევს ისეთ ფუნქციას, რომელიც შეესაბამება მისი მუშაობის მახასიათებლებს.

მნიშვნელოვანია, რომ ამოცანათა ამომხსნელ აგენტს უნდა გააჩნდეს მიზნის შემოწმების პროცედურა, რომელიც მას მისცემს საშუალებას განსაზღვროს არის თუ არა მოცემული კონკრეტული მდგომარეობა მიზნის მდგომარეობა. ზოგჯერ მიზნის მდგომარეობათა სიმრავლე მოიცემა ცხადი სახით და ეს პროცედურა დაიყვანება უბრალო შემოწმებაზე, არის თუ არა მოცემული მდგომარეობა სიმრავლის ელემენტი (როგორც ჩვენს მიერ ზემოთგანხილულ მაგალითში). შეიძლება მიზანი მოცემული იყოს რაიმე თვისებების აღწერით, როგორც გვაქვს ჭადრაკის თამაშის შემთხვევაში. ასეთ შემთხვევაში უნდა შემოწმდეს, აკმაყოფილებს თუ არა მოცემული მდგომარეობა ამ თვისებებს.

3.3 მაგალითები. ეს კომპონენტები განსაზღვრავენ ამოცანას და საჭიროა მათი გაერთიანება მონაცემთა სტუქტურაში, რომელიც მიეწოდება შესასვლელზე ამოცანათა ამომხსნელ

აგენტს. ასეთი ფორმულირების დროს ამოცანის ამონახსნი ეწოდება გზას საწყისი მდგომარეობიდან მიზნის მდგომარეობამდე. ამოხსნის ხარისხი იზომება გზის ღირებულების ფუნქციით, ხოლო ოპტიმალური ამონახსნი ეწოდება იმ გზას, რომელსაც აქვს მინიმალური ღირებულება.

განვიხილოთ ასეთი ამოცანების რამდენიმე მაგალითი. პირველ რიგში აღვნიშნოთ, რომ ეს ამოცანები შეიძლება იყოს სხვადასხვა თავსატეხები, ხელოვნურად მოფიქრებული მცირე ზომის ამოცანები, რომლებიც ძალიან კარგია ალგორითმების გამოსაკვლევად. ასეთ ამოცანებს უწოდებენ გამარტივებულ ამოცანებს. არსებობენ აგრეთვე რეალური ამოცანები, რომელთა ამოხსნა მართლაც სჭირდება ადამიანს. ასეთ ამოცანებს, როგორც წესი არ გააჩნიათ ისეთივე საყოველთაოდ მიღებული ერთიანი აღწერა, როგორც ეს გააჩნიათ გამარტივებულ ამოცანებს.

გამარტივებული ამოცანებიდან პირველ რიგში უნდა აღვნიშნოთ რვიანის თამაშის ამოცანა. ეს სათამაშო წარმოადგენს დაფას ცხრა უჯრით (3×3) და რვა ფიშკით, რომლებიც გადანომრილია ერთიდან რვამდე. ფიშკა, რომელიც მდებარეობს ცარიელი უჯრის გვერდით, შეიძლება გადაწეული იქნას ამ უჯრაზე. თამაშის მიზანია მივაღწიოთ რომელიმე კონკრეტულ მიზნის მდგომარეობას საწყისი მდგომარეობიდან (ისეთს, როგორიც მაგალითად მითითებულია სურ. #4.1.). ამ ამოცანის სტანდარტულ ფორმულირებას აქვს შემდეგი სახე:

2	8	3
1	6	4
7		5

ა)

1	2	3
8		4
7	6	5

ბ)

სურ. # 4.1. რვიანის თამაშის ა) საწყისი და ბ) მიზნის მდგომარეობები.

1. მდგომარეობები (მდგომარეობათა სიმრავლე). მდგომარეობათა აღწერა განსაზღვრავს თითოეული ფიშკის და ცარიელი ადგილის მდგომარეობას მოცემულ ცხრა კვადრატზე. ეს ძალიან ადვილია 3×3 მატრიცის საშუალებით, სადაც თითოეული ელემენტი ღებულობს მნიშვნელობებს ნოლიდან რვის ჩათვლით, იმ პირობით, რომ ცხრავე ელემენტი ღებულობს განსხვავებულ მნშვნელობებს,

$$A=\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

სადაც $0 \le a_{ij} \le 8, a_{ij} \in N$, $i, j \in N_3$ და $a_{ij} = a_{kl}$ მაშინ და მხოლოდ მაშინ, როდესაც $i = k, j = l$.

2. საწყისი მდგომარეობა. საწყის მდგომარეობად შეიძლება ავიღოთ ნებისმიერი მდგომარეობა. ამასთან, უნდა აღვნიშნოთ, რომ მოცემული მიზანი (მიზნის მდგომარეობა) მიიღწევა შესაძლო საწყისი მდგომარეობების ზუსტად ნახევრიდან.

3. შვილობილი წვეროს განმსაზღვრელი ფუნქცია (ოპერატორების სიმრავლე). ეს ფუნქცია ახდენს დასაშვები მდგომარეობების ფორმირებას, რომლებიც წარმოადგენენ მშობელ წვეროზე ოთხი ქმედების (ზევით, ქვევით, მარჯვნივ და მარცხნივ) შედეგს.

4. მიზნის შემოწმება. მიზნის შემოწმება საშუალებას გვაძლევს განვსაზღვროთ, შეესაბამება თუ არა მოცემული მდგომარეობა მიზნის მდგომარეობას.

5. გზის ფასი. თითოეული გადასვლა ერთი მდგომარეობიდან მეორეში მოითხოვს ერთი და იგივე სამუშაოს შესრულებას როგორც ფიზიკურად (ფიშკის გადაწევა ერთი უჯრიდან მეორეში), ასევე კომპიუტერული გამოთვლების თვალსაზრისით. ამიტომ გადასვლის ფასი a_{ij} უჯრიდან რომელიმე მეზობელ a_{ij+1}-ურ უჯრაში $w(a_{ij}, a_{ij+1})$ შეგვიძლია

ჩავთვალოთ ერთის ტოლად, მაშინ გზის ფასი $w(T)$ ტოლი იქნება გადასვლების რაოდენობისა საწყისი მდგომარეობიდან მიზნის მდგომარეობამდე.

განვიხილოთ კიდევ ერთი მაგალითი. რვა ლაზიერის ამოცანის მიზანია გან-ვალაგოთ ჭადრაკის დაფაზე რვა ლაზიერი ისე, რომ არც ერთი მათგანი არ ესხმოდეს თავს არცერთ სხვა ლაზიერს. მიუხედავად იმისა, რომ არსებობენ სპეციალიზირებული ალგო-რითმები, რომლებიც ზოგადად ხსნიან ამ ამოცანს n ცალი ლაზიერისათვის, ის მაინც სა-ინტერესოა ძებნის ალგორითმებისთვის.

შესაძლებელია ამ ამოცანის ორი ძირითადი ფორმულირება. ინკრემენტალური, რო-მლის დროსაც გამოიყენება ისეთი ოპერატორები, რომლებიც იწყებენ რა ცარიელი დაფი-დან, თანდათანობით ამატებენ ლაზიერებს დაფაზე, ამასთან, არ ითვალისწინებენ ემუქ-რება თუ არა რომელიმე ლაზიერი სხვა ლაზიერს. სრული მდგომარეობის ფორმულირება იწყება დაფაზე რვავე ლაზიერის დასმის შემდეგ და ითვალისწინებს მათ შემდგომ გადა-ადგილებებს. ასეთი ფორმულირების შემთხვევაში ამოცანის აღწერას აქვს შემდეგი სახე:

1.მდგომარეობები. მდგომარეობას წარმოადგენს დაფაზე ლაზიერების ნებისმიერი გან-ლაგება ნოლიდან რვამდე.

2. საწყისი მდგომარეობა. დაფაზე არ დგას არც ერთი ლაზიერი.

3. შვილობილის განმსაზღვრელი ფუნქცია (ოპერატორები). ლაზიერის დასმა დაფაზე.

4. მიზნის შემოწმება. დაფაზეა რვა ლაზიერი და არცერთი მათგანი არ ემუქრება სხვა რო-მელიმეს.

5. გზის ფასი. სვლების რაოდენობა საწყისი მდგომარეობიდან მინზნის მდგომარეობამდე.

ამ შემთხვევაში საჭიროა შემოწმდეს $64 \times 63 \times 62 \times ... \times 57 = 3 \times 10^{14}$ შესაძლო მიმდევრობა. თუ ჩვენ გვინდა გავაუმჯობესოთ ამოცანის ფორმულირება, რაც გულის-ხმობს მდგომარეობათა სიმრავლის შემცირებას, უნდა ავკრძალოთ ლაზიერის დასმა იმ უჯრებზე, რომლებიც იმყოფებიან შეტევის ქვეშ, მაშინ მივიღებთ ამოცანის ახალ ფორ-მულირებას.

1. მდგომარეობები. მდგომარეობას წარმოადგენს n ცალი ლაზიერის განლაგება $(0 \le n \le 8)$ დაფაზე, ამასთან ერთი ლაზიერი დგას მარცხნიდან დაწყებული თითო ვერ-ტიკალზე ისე, რომ არცერთი ლაზიერი არ ესხმის თავს სხვას.

2. შვილობილის განმსაზღვრელი ფუნქცია. ლაზიერის დასმა მარცხნიდან პირველ თავი-სუფალ ვერტიკალზე ისე, რომ არ მოხდეს მასზე თავდასხმა.

ასეთი ფორმულირების შემთხვევაში შესაძლებელი ხდება შევამციროთ მდგომარეობათა სიმრავლე 2057-მდე, და ამით ამოცანა სერიოზულად გაიოლდება. მაგრამ, თუ ჩვენ განვიხილავთ დავუშვათ 100 ლაზიერის ამოცანას, პირველი ფორმუ-ლირების შემთხვევაში მდგომარეობათა სიმრავლე იქნება 10^{400}, ხოლო მეორე შემთხვევაში – 10^{52}. ეს რა თქმა უნდა კოლოსალურად დიდი შემცირებაა, მაგრამ დარჩენილი სივრცეც ძალიან დიდია იმისათვის, რომ ჩვენმა ალგორითმებმა შეძლონ რეალურ დროში ასეთი ამოცანის ამოხსნა.

რეალური ამოცანებიდან აღვნიშნოთ მარშრუტის ძებნის ამოცანა, ცნობილი კო-მივოიაჟერის ამოცანა, ინტეგრალური სქემებში ელემენტების განლაგების ამოცანა, რობო-ტის გადაადგილების მართვის ამოცანა, კონვეირის მართვის ამოცანა და სხვა.

3.4. ძებნის ალგორითმების შეფასება. მას შემდეგ, რაც მოვახდენთ ამოცანის ფორმალიზე-ბას, საჭიროა ვიპოვოთ ამოხსნა. ჩვენ განვიხილავთ მეთოდებს, რომლებიც ამოცანას ხსნიან ძებნის საშუალებით მდგომარეობათა სივრცეში. აქ აუცილებელია გავარჩიოთ ორი შემთხ-ვევა. მდგომარებითა სივრცე შეიძლება იყოს გრაფი ან ხე (ხე არის გრაფის კერძო შემთხ-ვევა, როდესაც ყოველ წვეროს ჰყავს მხოლოდ ერთი მშობელი წვერო). მაგალითად, რვა ლაზიერის ამოცანაში (როდესაც ყოველი შემდეგი ლაზიერი დგება ყველაზე მარცხენა ცარიელ ვერტიკალზე), მდგომარეობათა სივრცე წარმოადგენს ხეს. ამიტომაც მდგომარე-ობათა სივრცე მიიღება შედარებით პატარა და ძებნა გაიოლებულია. თუ განვიხილავთ მეორე ფორმულირებას, სადაც ნებისმიერი ლაზიერი შეიძლება დავდგათ ნებისმიერ ვერტიკალზე, მდგომარეობათა სივრცე იქნება გრაფი, სადაც ერთი და იგივე მდგომარეობა

მიიღწევა განსხვავებული გზით, რაც ართულებს ძებნას. ზოგიერთ ამოცანებში მდგომარე-ობათა სივრცე აუცილებლად წარმოადგენს გრაფს (ასეთია ყველა ამოცანა, სადაც არ-სებობენ ოპერატორები, რომლებსაც მდგომარეობა გადაყავთ უკან, მშობელ მდგომა-რეობაში. რვიანის თამაში, მარშრუტის მოძებნის ამოცანა და სხვა). ასეთ ამოცანებში ძებნის ხე შეიძლება მივიღოთ უსასრულო მაშინაც კი, როდესაც მდგომარეობათა სივრეცე სას-რულოა. გარდა ამისა, გრაფზე ძებნის დროს ალგორითმი შეიძლება ჩაიციკლოს და ვეღარ იპოვნოს ამოხსნა. იმისათვის, რომ თავიდან ავიცილოთ ეს მომენტები, თითქმის ყველა ქვემოთ განხილულ ალგორითმებში გამოიყენება სპეციალური პროცედურა რომელიც თავიდან აგვაცილებს ძებნას გრაფზე და გრაფს გადააქცევს ხედ. ამისათვის ალგორითმი იმახსოვრებს ყველა წვეროს რომელიც წარმოიშვება ძებნის პროცესში. ამასთან წვეროებს, რომელთა შვილობილი წვეროები ჯერ არ აუგია, ანუ წვერო ჯერ არ გაუხსნია, იმახ-სოვრებს სიაში “ღიაა”, ხოლო იმ წვეროებს რომელთა შვილობილი წვეროებიც უკვე ააგო, სიაში “დახურულია”. მუშაობის პროცესში, ყოველი ახალი წვეროს აგების შემდეგ, ალ-გორითმი ადარებს ამ წვეროს იმ წვეროებს, რომლებიც მოთავსებულია ორივე სიაში. თუ ასეთი წვერო არ არის არც ერთ სიაში, ალგორითმი მას შეიტანს სიაში ღიაა. თუ ალგო-რითმი აღმოაჩენს ასეთ წვეროს რომელიმე სიაში, მისი შემდეგი ქმედება დამოკიდებულია ალგორითმის მოქმედების პრინციპზე და აუცილებლად წინასწარ უნდა იყოს განსაზღვ-რული ალგორითმში. ყველა შემთხვევაში შედეგი იქნება ის, რომ ჩვენ მივიღებთ ძებნის ხეს და არა გრაფს. ამოცანათა ამოხსნის ყველა ალგორითმს, რომელიც ეძებს ამონახსნს ძებნის საშუალებით, ჩვენ შევაფასებთ ოთხი კრიტერიუმის საშუალებით:

1. სისრულე. იძლევა თუ არა ალგორითმი ამოხსნის პოვნის გარანტიას, როდესაც ეს ამოხსნა არსებობს?
2. ოპტიმალურობა. უზრუნველყოფს თუ არა ალგორითმი ოპტიმალური ამონახსნის მოძებნას?
3. დროითი სირთულე. რა დროში შეუძლია ალგორითმს ამონახსნის მოძებნა?
4. სივრცითი სირთულე. რა მოცულობის მეხსიერება სჭირდება ალგორითმს ამონახსნის საპოვნელად?

დროითი და სივრცითი სირთულეები ყოველთვის იანგარიშება ამოცანის სირთულის გათვალისწინებით. განსაკუთრებით უნდა აღვნიშნოთ, რომ განსხვავებით კომპიუტერულ მეცნიერებათა სხვა დარგებისაგან, სადაც მდგომარეობათა გრაფი ყოველთვის მოიცემა ცხადი სახით და სირთულის ტიპურ კრიტერიუმს წარმოადგენს ამ გრაფის ზომა, ხელოვნურ ინტელექტში გრაფი მოიცემა არაცხადი სახით, საწყისი მდგომარეობისა და ოპერატორების საშუალებით და მისი აგების პროცესი გაერ-თიანებულია ამონახსნის ძებნის პროცესთან, ამიტომ ამ შემთხვევაში ის ხშირად უსას-რულოა, სირთულე აღიწერება სამი კოეფიციენტის საშუალებით: b – განშტოების კოე-ფიციენტი, ანუ ნებისმიერი წვეროს შვილობილ წვეროთა მაქსიმალური რაოდენობა; d – ყველაზე ზემოთ მყოფი მიზნის წვეროს სიღრმე და m – მდგომარეობათა სივრცეში ნების-მიერ გზათა შორის მაქსიმალური სიგრძე. დროითი სირთულე ხშირად იზომება დროის ერთეულში გამომუშავებულ წვეროთა რაოდენობით, ხოლო სივრცითი სირთულე – იმ წვეროთა მაქსიმალური რაოდენობით, რომლებიც უნდა დაიმახსოვროს ალგორითმმა.

საკონტროლო კითხვები:

1. რა მიზანს ისახავს ამოცანის ფორმალიზება?
2. როგორ პრობლემურ გარემოში უხდება მუშაობა ამოცანათა ამომხსნელ აგენტს?
3. რა კომპონენტებით აღიწერება ამოცანა?
4. მოიყვანეთ ერთი და იგივე ამოცანის სხვადასხვა აღწერის მაგალითები.
5. რა განსხვავებაა გრაფზე და ხეზე ძებნას შორის?
6. რა პარამეტრებით ფასდება ძებნის ალგორითმები?

ლექცია 4. ძებნის არაინფორმირებული ალგორითმები

4.1. სრული გადარჩევის ალგორითმი. ძებნის ალგორითმების განხილვას დავიწყებთ ისეთი ალგორითმებიდან, რომლებიც მიზნის მოსაძებნად იყენებენ მხოლოდ იმ ინფორმაციას რომელიც შედის ამოცანის ფორმალურ აღწერაში და არ გააჩნიათ არავითარი დამატებითი ინფორმაცია მდგომარეობების შესახებ. ასეთ ალგორითმებს უწოდებენ არაინფორმირებულ (ზოგჯერ ბრმად ძებნის) ალგორითმებს. ძებნის ამოცანების მნიშვნელოვანი პარამეტრებია საწყისი მდგომარეობიდან მიზნის მდგომარეობამდე გზის ფასი, რომელიც წარმოადგენს ამოცანის ამოხსნის გზაზე მდგომი ყველა მეზობელ წვეროებს შორის გადასვლათა ფასების ჯამს და ამონახსნის სიღრმე, გადასვლათა რაოდენობა საწყისი მდგომარეობიდან მიზნის მდგომარეობამდე. დავიწყოთ ყველაზე მარტივი, განივი ანუ სრული გადარჩევის ალგორითმით. ამოცანების (პრობლემების) ამომხსნელი აგენტი ამ ალგორითმს (ისევე როგორც ყველა დანარჩენ ალგორითმებს, რომლებიც აღწერილი იქნება ამ თავში, იყენებს მას შემდეგ, რაც მოახდენს ამოცანის ფორმალიზებას. ალგორითმის ბლოკ-სქემა მოცემულია სურ. #4.1. ის შედგება რვა ბიჯისაგან:

ბიჯი 1: S_0 საწყისი მდგომარეობა მოვათავსოთ სიაში "**ღიაა**";

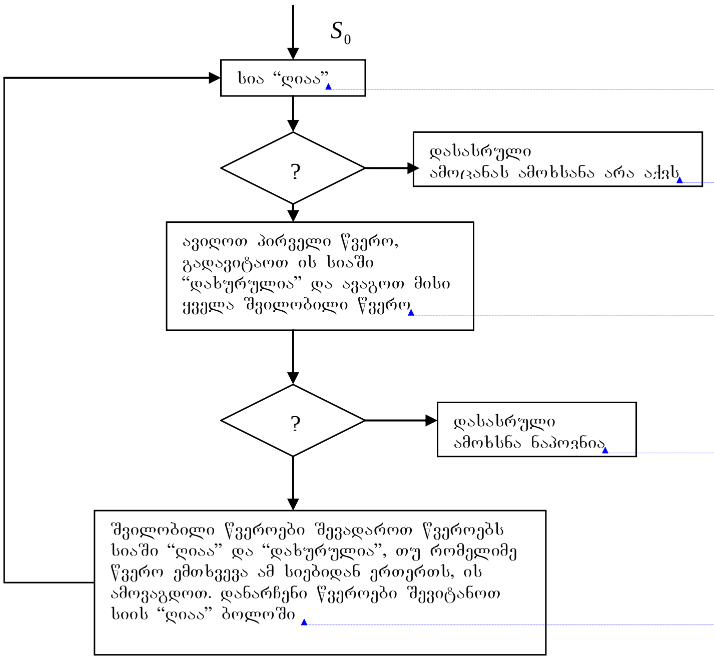
ბიჯი 2: **თუ** სია "**ღიაა**" ცარიელია გადადი მესამე ბიჯზე, **წინააღმდეგ შემთხვევაში** მეოთხე ბიჯზე;

ბიჯი 3: **გამოიტანე** „ამოცანას ამოხსნა არა აქვს". დაამთავრე მუშაობა.

ბიჯი 4: **აიღე** პირველი წვერო სიიდან "**ღიაა**", **გადაიტანე** სიაში "**დახურულია**", შვილობილის განმსაზღვრელი ფუნქციების (ოპერატორების) საშუალებით, **ააგე** ყველა შვილობილი წვეროები;

ბიჯი 5: **შეადარე** ახლად აგებული წვეროები მიზნის მდგომარეობას, **თუ** რომელიმე მათგანი ემთხვევა მიზნის მდგომარეობას **გადადი** მეექვსე ბიჯზე, **წინააღმდეგ შემთხვევაში** – მეშვიდე ბიჯზე;

ბიჯი 6: **გამოიტანე** ამოცანის ამოხსნა და დაამთავრე მუშაობა;



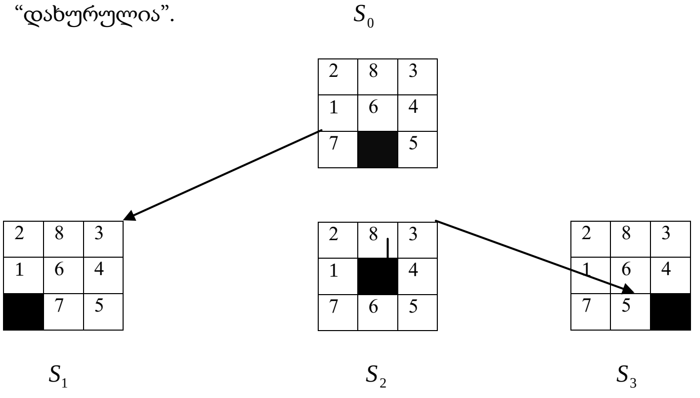
სურ. #4.1. სრული, ანუ განივი გადარჩევის ალგორითმის სქემა.

Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 9 pt
Formatted: Font: AcadNusx, 9 pt

ბიჯი 7: შეადარე ახლადაგებული წვეროები წვეროებს სიებში "**ღიაა**" და "**დახურულია**". **თუ** რომელიმე ახლადაგებული წვერო ემთხვევა რომელიმე წვეროს ერთერთ სიაში, **გადააგდე** ეს წვერო.

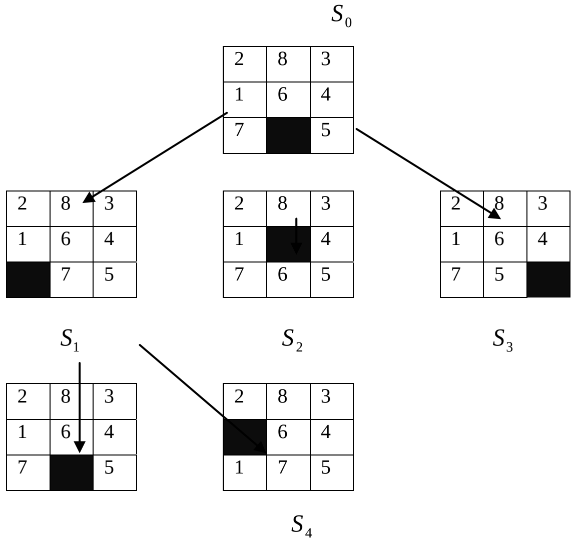
ბიჯი 8: **შეიტანე** დარჩენილი წვეროები სიის "**ღიაა**" ბოლოში.

ვნახოთ როგორ მუშაობს ეს ალგორითმი.განვიხილოთ მისი მუშაობა წინა ლექციაზე აღწერილი რვიანის თამაშის მაგალითზე. პირველ ნაბიჯზე, საწყისი მდგომარეობა შევა სიაში "ღიაა". რადგანაც სია "ღიაა" არ იქნება ცარიელი, დასაშვები ოპერატორების საშუალებით მიიღება სამი შვილობილი წვერო (იხ. სურ. #4.2.), ხოლო საწყისი წვერო გადაიტანება სიაში "დახურულია".



სურ. #4.2. ალგორითმის მუშაობის პირველი ეტაპი

ახლადაგებული წვეროებიდან არცერთი არ ემთხვევა მიზნის წვეროს და არც სიაში "დახურულია" მყოფ ერთადერთ წვეროს (სია "ღიაა" ცარიელია), ამიტომ სამივე ახლადაგებული წვეროები შევა სიაში "ღიაა" იმ მიმდევრობით, რა მიმდევრობითაც ისინი იყო აგებული (სურ. #4.2. მარცხნიდან მარჯვნივ). ვინაიდან ამ ალგორითმისთვის ყველა გადასვლას ერთი მეზობელი წვეროდან მეორეში აქვს ერთი და იგივე, ერთის ტოლი ფასი, თითოეული წვეროს ფასი $w(S_i) = 1$ ნებისმიერი i-თვის. ციკლის მეორედ გამეორებისას სიიდან "ღიაა" აიღება პირველი წვერო და აიგება მისი შვილობილი წვეროები (იხ. ნახ. #4.3).



სურ. #4.3. ალგორითმის მიერ ციკლის მეორედ გამეორებისას აგებული წვეროები

ალგორითმი კვლავ შეადარებს ახლადაგებულ წვეროებს მიზნის წვეროს და რადგანაც არცერთი მათგანი არ ემთხვევა მიზნის წვეროს, ალგორითმი გადავა რა ბოლო ბიჯზე, შეადარებს ამ წვეროებს წვეროებთან სიებიდან “ღიაა” და “დახურულია”. აღმოაჩენს, რომ ერთერთი მათგანი ემთხვევა სიაში “დახურულია” მყოფ S_0 წვეროს და გადააგდებს მას. მეორე (S_4) წვეროს კი შეიტანს სიის “ღიაა” ბოლოში, ამიტომ შემდეგ ეტაპზე ის გა-სახსნელად აიღებს S_2 წვეროს. ასე გააგრძელებს მუშაობას ალგორითმი მანამდის, სანამ ერთერთ ეტაპზე ან სია “ღიაა” არ იქნება ცარიელი, რაც იმას ნიშნავს, რომ ალგორითმმა გადასინჯა ყველა შესაძლო მდგომარეობები და ამოცანას ამოხსნა არა აქვს, ან რომელიმე ახლადაგებული წვერო არ დაემთხვევა მიზნის წვეროს, რაც იმას ნიშნავს, რომ ალგორით-მმა იპოვა ამოხსნა, ანუ გზა საწყისი წვეროდან მიზნის წვერომდე.

როგორც ვხედავთ, ეს ალგორითმი თანმიმდევრულად აგებს ყველა შესაძლო გზებს საწყისი წვეროდან ისე, რომ არ გამოტოვებს არც ერთ შესაძლო გზას, რადგანაც მისთვის ყველა წვერო ინფორმაციულად ერთნაირია. სწორედ ამიტომ უწოდებენ მას სრული, ანუ ბრმა გადარჩევის ალგორითმს. ჩვენი მაგალითისათვის ის ჩავა მეექვსე დონეზე და გახს-ნის 46 წვეროს სანამ იპოვნის მიზნის წვეროს.

შევაფასოთ ალგორითმი ზემოთ მოყვანილი კრიტერიუმებით. ცხადია, რომ ეს ალგორითმი არის სრული. მართლაც, თუკი არსებობს ამოცანის ამოხსნა, ის მას ადრე თუ გვიან აუცილებლად იპოვნის. თუ ყველაზე ზევით მდგომი მიზნის წვეროს სიღრმეა d, ალგორითმი მას იპოვნის $d-1$ დონის ყველა წვეროს გახსნის შემდეგ. რადგანაც ამ ალგორითმისთვის ყველა გადასვლას ერთი ფასი აქვს, ნაპოვნი ამოხსნა ავტომატურად იქნება ოპტიმალურიც. ამ ორი პარამეტრით ალგორითმი შესანიშნავია, მაგრამ კატასტრო-ფულად სუსტია დანარჩენი ორი პარამეტრის მიხედვით. დავუშვათ, თითოეული მდგო-მარეობიდან შესაძლებელია მაქსიმუმ b ცალი შვილობილი წვეროს აგება. მაშინ ალგო-რითმი პირველ დონეზე საწყისი მდგომარეობიდან ააგებს b ახალ მდგომარეობას, მეორე ეტაპზე b^2 მდგომარეობას, მესამეზე b^3 და ასე შემდეგ. დავუშვათ ახლა, რომ მიზნის მდგომარეობა მდებარეობს d დონეზე და ის აიგება ამ დონის წვეროებიდან ყველაზე ბოლოს, მაშინ მდგომარეობების რაოდენობა რომლებიც უნდა ააგოს ალგორითმა, რათა იპოვოს ამოხსნა ტოლია

$$b + b^2 + b^3 + ... + b^d$$

რაც გვაძლევს შეფასებას $O(b^d)$. ეს იმას ნიშნავს რომ ალგორითმი არის ექსპონენციალური და ამოცანის ზომებისა და ამონახსენის დონის ზრდასთან ერთად ძალიან სწრაფად იზრდება აგებულ წვეროთა რაოდენობა. ამონახსნის დონის, აგებულ წვეროთა რაოდენობის, დახარჯული დროისა და გამოყენებული მეხსიერების თანაფარდობა მოყვანილი ცხრილში N2. ამ ცხრილში $b = 10$-ს; წვეროების გახსნის სიჩქარეა - 10 000 წვ/წამში; მეხსიერების მოცულობა – 1000 ბაიტი/წვეროზე.

ცხრილი N 2

დონე	მდ.რაოდენობა	დრო	მეხსიერება
2	1100	0,11წმ	1 მეგაბაიტი
4	111 100	11 წმ	106 მეგაბაიტი
6	10	19 წთ	10 გიგაბაიტი
8	10	31 სთ	1 ტერაბაიტი
10	10	129 დღე	101 ტერაბაიტი
12	10	35 წელი	10 პეტაბაიტი
14	10	3523 წელი	1 ექსაბაიტი

როგორც ამ ცხრილიდან ჩანს, ამოცანის ზომების გაზრდასთან ერთად პირველ რიგ-ში კატასტროფულად იზრდება მოთხოვნა მეხსიერებაზე, ხოლო რეალური ამოცანების-თვის, როგორც მეხსიერება, ასევე ამონახსნის ძებნის დრო პრაქტიკულად შეუძლებელს ხდის ამოცანის ამოხსნას ამ ალგორითმის საშუალებით.

4.2. ძებნა გზის ფასის კრიტერიუმით. განივი გადარჩევის ალგორითმი იყო ოპტიმალური, თუ ყველა შვილობილი წვეროს აგების ფასი იყო ტოლი. მაგრამ ეს ყოველთვის ასე არ არის. დავუშვათ, ჩვენს ამოცანაში ფიშკის ზევით აწევაზე იხარჯება უფრო მეტი ენერგია, ვიდრე ქვევით ჩამოწევაზე, მაშინ გადასვლის ფასი ერთი წვეროდან მეორეში იქნება სხვადასხვა. მარტივი გარდაქმნით განივი გადარჩევის ალგორითმიდან შეიძლება მივიღოთ ახალი ალგორითმი, რომელიც ახორციელებს ძებნას გზის ფასის კრიტერიუმით ისე, რომ არ შევცვალოთ ალგორითმის სტრუქტურა. ამისთვის საჭიროა ორი ცვლილების შეტანა განივი ძებნის ალგორითმში. პირველი ეხება გაუხსნელი წვეროების განლაგებას სიაში “ღიაა”. შემოვიტანოთ ფუნქცია $g(n)$, რომელიც წაროადგენს გზის ფასს საწყისი მდგომარეობიდან n წვერომდე და სიაში “ღიაა” წვეროები განვალაგოთ ისე, რომ სიის თავში მოექცნენ ის წვეროები, რომელთა ფასიც არის მინიმალური. მეორე შესწორება ეხება გრაფის ხედ გადაქცევის პროცედურას. განივი ძებნის ალგორითმისგან განსხვავებით, თუ რომელიმე ახლად აგებული წვერო დაემთხვევა წვეროს სიაში “ღიაა” (რაც ნიშნავს რომ არსებობს ორი განსხვავებული გზა საწყისი წვეროდან ამ წვერომდე), ალგორითმი დატოვებს არა პირველად აგებულ გზას, როგორც ეს ხდებოდა სრული გადარჩევის ალგორითმში, არამედ გზას მინიმალური შეფასებით. თუ ახლად აგებული წვერო დაემთხვევა რომელიმე წვეროს სიაში “დახურულია”, ჩატარდება იგივე პროცედურა და თუ ახლად აგებული გზის ფასი ნაკლებია არსებულ გზის ფასზე, წვერო დაბრუნებული იქნება სიაში “ღიაა”. თუ გადასვლის ფასი ერთი წვეროდან მეზობელ წვეროში $w(n_i, n_{i+1}) \geq \varepsilon$, სადაც ε არის რაიმე ნოლისგან განსხვავებული დადებითი რიცხვი, მაშინ ალგორითმი იქნება სრული. ეს პირობა საკმარისია იმისთვისაც, რომ ალგორითმი იყოს ოპტიმალურიც.

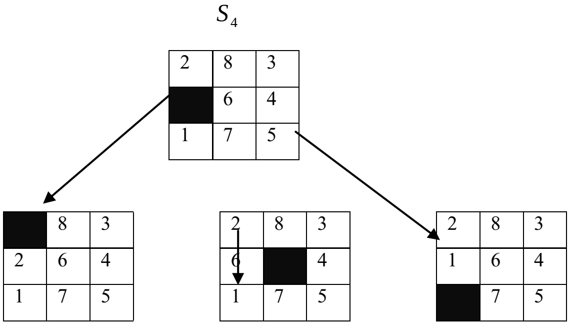
ამ ალგორითმში ძებნა ხდება არა დონეების მიხედვით, არამედ გადასვლების ფასის მიხედვით, ამიტომ მისი სირთულე ადვილად ვერ განისაზღვრება b და d ტერმინებში. შემოვიღოთ პარამეტრები C - ამოხსნის ოპტიმალური ფასი და ε - წვეროდან მეზობელ წვეროში გადასვლის მინიმალური ფასი, მაშინ ამ ალგორითმის დროითი და სივრცითი სირთულე ყველაზე ცუდ შემთხვევაში ტოლია $O(b^{1+C/\varepsilon})$, ანუ შეიძლება იყოს მეტი ვიდრე $O(b^d)$.

4.3. ძებნა სიღრმეში. ასევე სტრუქტურის შეუცვლელად შეგვიძლია განივი ძებნის ალგორითმი გარდავქმნათ სიღრმეში ძებნის ალგორითმად. ამისთვის საკმარისია ახლად აგებული წვეროები შევიტანოთ არა სიის “ღიაა” ბოლოში, არამედ თავში. ასეთი ძებნის დროს ალგორითმს სჭირდება მეხსიერების ძალინ მცირე მოცულობა, რადგანაც ის იმახსოვრებს მხოლოდ ერთ გზას საწყისი წვეროდან დამამთავრებელ წვერომდე (დამათავრებელია წვერო, თუ მას არ გააჩნია შვილობილი წვეროები) იმ მეზობელ წვეროებთან ერთად, რომლებიც ჯერ არ გაუხსნია ალგორითმს. მას შემდეგ, რაც გაიხსნება წვერო, ის შეიძლება წაშლილი იქნას მეხსიერებიდან. თუ ერთი წვეროდან შეიძლება მაქსიმუმ b წვეროს მიღება და ძებნის სიღრმეა d, ალგორითმი იმახსოვრებს მხოლოდ $b \cdot d + 1$ წვეროს, ამიტომაც თუკი ამონახსენი მდებარეობს სიღრმეზე, მას სჭირდება მხოლოდ 118 კილობაიტი მეხსიერება განსხვავებით განივი გადარჩევის ალგორითმისაგან, რომელსაც სჭირდება 10 პეტაბაიტი მეხსიერება. ამ ალგორითმის ერთერთ ვარიანტში, რომელსაც ეწოდება სიღრმეში ძებნის ალგორითმი უკან დაბრუნებით, საჭიროა კიდევ უფრო მცირე მეხსიერება. ეს ალგორითმი ყოველ ნაბიჯზე ხსნის მხოლოდ ერთ შვილობილ წვეროს და იმახსოვრებს რომელი შვილობილი უნდა გახსნას შედეგ, უკან დაბრუნების შემთხვევაში. მაშინ ალგორითმს სჭირდება $O(b)$ მეხსიერება ნაცვლად $O(bd)$.

განვიხილოთ ამ ალგორითმის მუშაობა ისევ ზემოთმოყვანილ მაგალითზე. სიღრმეში ძებნის ალგორითმისა და სრული გადარჩევის ალგორითმის მუშაობა პირველ და მეორე ეტაპზე ფაქტობრივად ერთნაირია მცირე გამონაკლისის გარდა, რომელიც მდგომარეობს შემდეგში: მეორე ეტაპზე მიღებული ერთადერთი წვერო (მეორე წვერო უკვე გვხვდება სიაში “დახურულია” და ამიტომ მას არც ერთი ალგორითმი აღარ განიხილავს), სრული გადარჩევის ალგორითმში შევა სიაში “ღიაა” სულ ბოლო, მეოთხე ადგილზე, მაშინ როდესაც სიღრმეში ძებნის ალგორითმში ეს წვერო სიაში “ღიაა” იქნება პირველ ადგილზე, ამი-

ტომ მესამე ეტაპიდან ამ ალგორითმების მუშაობა იქნება განსხვავებული. კერძოდ, სიღრმეში ძებნის ალგორითმისათვის მესამე ეტაპი გამოიყურება შემდეგნაირად (იხ. სურ. #4.4.).

სიღრმეში ძებნის ალგორითმს გააჩნია თავისი ნაკლოვანებები. პირველ რიგში ის უკვე აღარ არის სრული, რადგანაც ალგორითმი შეიძლება გაყვეს ქვევით ძალიან გრძელ გზას, ან საერთოდ უსასრულოს და ვერ იპოვოს ამონახსნი, მაშინ როდესაც მეორე გზით იპოვიდა ამონახსნს, რომელიც მდებარეობს საწყისი წვეროდან ძალიან ახლოს. ამავე მიზეზით ის არ იძლევა ოპტიმალური ამონახსნის პოვნის გარანტიას. უსასრულო მდგომარეობათა სივრცის შემთხვევაში კი ამ ალგორითმის გამოყენება პრაქტიკულად შეუძლებელია.



სურ. #4.4. სიღრმეში ძებნის ალგორითმის მესამე ეტაპი

ამ პრობლემის თავიდან აცილება შესაძლებელია სიღრმის წინასწარი შეზღუდვით. დავუშვათ, შემოვიღოთ პარამეტრი L და ალგორითმში მესამე და მეოთხე ბიჯებს შორის ჩავამატოთ ერთი ბიჯი: სანამ ალგორითმი გახსნის ახალ წვეროს, შეამოწმოს ხომ არ ემთხვევა ამ წვეროს სიღრმე L-ის მნიშვნელობას. თუ ემთხვევა, ასეთი წვერო აღარ გახსნას ალგორითმა და გადავიდეს მეორე ბიჯზე. მაშინ მდგომარეობათა სივრცე მიიღება სასრული და ალგორითმი შეძლებს იპოვოს ამონახსენი, თუ ეს ამონახსენი არ მდებარეობს L დონეზე ქვევით. ადვილი მისახვედრია, რომ ალგორითმი ისევაც არ იქნება სრული და ოპტიმალური. L პარამეტრის აღება შეიძლება ნებისმიერად, ან შეიძლება გამოვიყენოთ ექსპერტთა ცოდნა თუ რა სიღრმეზე შეიძლება მდებარეობდეს ამ ამოცანის ამოხსნა.

4.4. ორმხრივმიმართული ძებნა. საინტერესოა ორმხრივი ძებნის იდეა. წარმოვიდგინოთ ორი ერთნაირი ალგორითმი (დავუშვათ სიგანეში გადარჩევის), რომლებიც ხსნიან ერთსა და იმავე ამოცანას, ოღონდ ერთის საწყისი მდგომარეობა მეორესთვის წარმოადგენს მიზნის მდგომარეობას. ეს ალგორითმები დაკავშირებულნი არიან ერთმანეთთან და ყოველი წვეროს გახსნის წინ ამოწმებენ ხომ არ ემთხვევა ეს წვერო მეორე ალგორითმის რომელიმე წვეროს. დამთხვევის შემთხვევაში ამოცანის ამოხსნა უკვე ნაპოვნია. ასეთი ალგორითმის როგორც დროითი, ასევე სივრცითი სირთულე იქნება $O(b^{d/2})$, ამავე დროს ის იქნება სრული და ოპტიმალური. პრობლემა მდგომარეობს იმაში, რომ ყველა ამოცანისათვის ადვილი არ არის მოვახერხოთ ძებნა მიზნის მდგომარეობიდან საწყისი მდგომარეობისაკენ, რადგანაც რეალურ ამოცანებში ხშირად ჩვენთვის ცნობილია მიზნის მდგომარეობათა სიმრავლის აღწერა, მაგრამ უცნობია ამ სიმრავლის კონკრეტული ელემენტები (მაგალითად ჭადრაკის თამაშისათვის ეს შეუძლებელია).

საკონტროლო კითხვები:

1. რატომ ეწოდებათ ამ ალგორითმებს ძებნის არაინფორმირებული ალგორითმები?
2. აღწერეთ სრული გადარჩევის ალგორითმი, როგორია მისი შეფასება?
3. რა განსხვავებაა სრულ და გზის ფასის კრიტერიუმით ძებნის ალგორითმებს შორის?

4. რა უპირატესობა აქვს სიღრმეში გადარჩევის ალგორითმს სრული გადარჩევის ალგორითმთან შედარებით?
5. რა პრინციპით ხდება ორმხრივი ძებნა და რატომ არ შეიძლება მისი გამოყენება ნებისმიერი ამოცანისათვის?

ლექცია 5. ძებნის ინფორმირებული ალგორითმები.

5.1. ევრისტიკული ინფორმაცია. ძებნის ეს ყველა არაინფორმირებული ალგორითმები, რომლებიც ჩვენ განვიხილეთ, მოქმედებენ ერთი პრინციპით. ისინი რაღაც თანმიმდევრობით გამოიმუშავებენ მდგომარეობებს და შემდეგ ადარებენ მას მიზნის მდგომარეობას, ან მის აღწერას. მათ არ შეუძლიათ რაიმე ნიშნით განასხვაონ ეს მდგომარეობები ერთმანეთისაგან და ამოარჩიონ ის მდგომარეობები, რომლებიც უფრო სწრაფად მიგვიყვანენ მიზნის მდგომარეობასთან, რადგან არ გააჩნიათ ინფორმაცია, რომელიც საშუალებას მისცემთ, მოახდინონ მდგომარეობების დიფერენცირება. მაგალითად, ცნობილია, რომ ჭადრაკის თამაშის დროს პირველ სვლას არავინ აკეთებს $a2-a4$, თუმცა თამაშის წესებით ეს სვლა არაა აკრძალული და რადგანაც ძებნის არაინფორმირებული ალგორითმები იყენებენ მხოლოდ ინფორმაციას, რომელიც მოცემულია ამოცანის პირობებში, მათთვის ჭადრაკის ოცივე საწყისი სვლა აბსოლუტურად ერთნაირია, მაშინ, როდესაც იმისთვის, ვინც კარგად თამაშობს ჭადრაკს, ეს ასე არ არის.

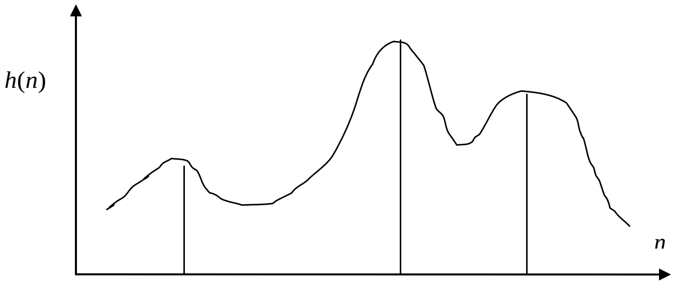
იმისათვის, რომ შევძლოთ შევაფასოთ მდგომარეობა, უნდა გაგვაჩნდეს დამატებითი ინფორმაცია იმ კლასის ამოცანების შესახებ, რომელსაც ვხსნით. ეს ინფორმაცია არ გამოდის ამოცანის პირობებიდან და ძირითადად ემყარება ან ლოგიკურ მსჯელობას, ან ჩვენს გამოცდილებას და ცოდნას, რომელიც დაგვიგროვდა ამ ტიპის ამოცანების ამოხსნის დროს. ასეთ ინფორმაციას უწოდებენ ევრისტიკულ ინფორმაციას, ხოლო ძებნის ალგორითმებს, რომლებიც იყენებენ ასეთ ინფორმაციას კი – ევრისტიკული ძებნის ალგორითმებს.

სიტყვა ევრისტიკა, ბერძნულად **eurisco**, ნიშნავს გამოკვლევას და მდგომარეობათა სივრცეში ძებნისას განისაზღვრება როგორც ისეთი წესების ერთობლიობა წვეროებისა და გზების ამოსარჩევად, რომლებიც ყველაზე დიდი ალბათობით მიგვიყვანენ ამოცანის ამონახსნთან. ასეთი ტიპის ძებნა გამოიყენება ორ ძირითად შემთხვევაში:

1. არ არსებობს დასმული ამოცანის ამოხსნის ალგორითმი იმის გამო, რომ შეუძლებელია ამოცანის მკაცრი ფორმალიზება, ან/და არაა საკმარისი საწყისი მონაცემები.
2. ამოცანას გააჩნია ამოხსნის ალგორითმი, მაგრამ მდგომარეობათა სივრცე იმდენად სწრაფად იზრდება, რომ ამ ალგორითმის გამოყენება პრაქტიკულად შეუძლებელია.

განვიხილოთ ასეთი ფუნქციის შედგენის პროცესი რვიანის თამაშის მაგალითზე. როგორც გვახსოვს, ამ თამაშში მდგომარეობა წარმოადგენს ცხრა კვადრატს მასზე განლაგებული რვა ფიშკით. ყოველ კონკრეტულ მდგომარეობაში რაღაც ნაწილი ფიშკებისა არის თავის ადგილზე (ანუ იმ პოზიციაში, რომელშიც უნდა იყოს მიზნის მდგომარეობაში) და ნაწილი კი – არა. მდგომარეობის შესაფასებლად დავითვალოთ რამდენი ფიშკა დგას თავის ადგილზე. ალბათ ბუნებრივი იქნება თუ ჩავთვლით, რომ რაც უფრო მეტი ფიშკა დგას თავის ადგილას, მით უფრო პოზიცია “ახლოს” არის მიზნის მდგომარეობასთან, თუმცა, ეს ალბათ ყოველთვის ასე არ არის. აღვნიშნოთ ეს ფუნქცია $h(n)$ -თ. ეს იქნება ევრისტიკული ფუნქცია. ასეთი შეფასება არ გამომდინარეობს თამაშის წესებიდან, ამიტომ ჩვენ ვერ შევძლებთ მკაცრად დავამტკიცოთ, რომ ასეთი შეფასება ყოველთვის სწორია, უფრო მეტიც, გამომდინარე იქიდან, რომ ამ თამაშში არსებობენ პოზიციები, რომლებიდანაც მიზნის მდგომარეობა მიუღწევადია, აშკარაა, რომ ასეთმა შემფასებელმა ფუნქციამ შეიძლება ვერც გვაპოვნინოს ამოხსნა, მაგრამ თუ მისი გამოყენება შეამცირებს გადარჩევას და გააიოლებს ამოცანის ამოხსნას, ჩვენი მიზანი მიღწეული იქნება.

5.2. "ხარბი" ალგორითმები. ალგორითმებს, რომლებიც ძებნისათვის გამოიყენებენ მხოლოდ ევრისტიკულ ინფორმაციას, უწოდებენ "ხარბ" ალგორითმებს. ერთერთ ასეთ ალგორითმს წარმოადგენს პიკისკენ აღმავალი ძებნის ალგორითმი. განვიხილოთ ეს ალგორითმი რვიანის თამაშის მაგალითზე. თუ ჩვენ ვიყენებთ ჩვენს მიერ ზემოთ აღწერილ ევრისტიკულ ფუნქციას, მაშინ ცხადია, რომ ამ ფუნქცის მნიშვნელობა მიზნის მდგომარეობაში იქნება მაქსიმალური. მაშინ ალგორითმი წარმოადგენს ჩვეულებრივ ციკლს, რომელშიც მუდმივად ხდება გადანაცვლება ფუნქციის ზრდისკენ (იხ.სურ.#5.1), ანუ ხდება ასვლა პიკზე მანამ, სანამ ალგორითმი არ მიაღწევს მდგომარეობას, რომელშიც მიზნის ფუნქციას აქვს ყველაზე დიდი მნიშვნელობა მეზობელ მდგომარეობებთან შედარებით. ეს ალგორითმი გამოიყენებს რა მხოლოდ ევრისტიკულ ფუნქციას $h(n)$ -ს,მოცემული მდგომარეობებიდან ამოარჩევს საუკეთესოს და შემდეგში განიხილავს მხოლოდ ამ წვეროს, დანარჩენებს კი არ იმახსოვრებს.



სურ. # 5.1. ლოკალური ოტპიმუმის ძებნის გრაფიკი

მაგალითად, თუ გამოვიყენებთ ამ ალგორითმს ზემოთგანხილული მაგალითისათვის, ძებნის ხეს ექნება შემდეგი სახე (იხ. სურ. #5.2.). საწყისი მდგომარეობიდან ალგორითმი ააგებს შვილობილ წვეროებს, გამოთვლის მათთვის ფუნქციის მნიშვნელობებს და ამის მიხედვით ამოარჩევს წვეროს, დანარჩენ წვეროებს კი გადააგდებს. მეორე ეტაპზე გახსნის წვეროს, გამოთვლის ამ წვეროს შვილობილებს ფუნქციის მნიშვნელობებს, ამოარჩევს მათ შორის საუკეთესოს და ასე გააგრძელებს, სანამ არ მიაღწევს პიკს.

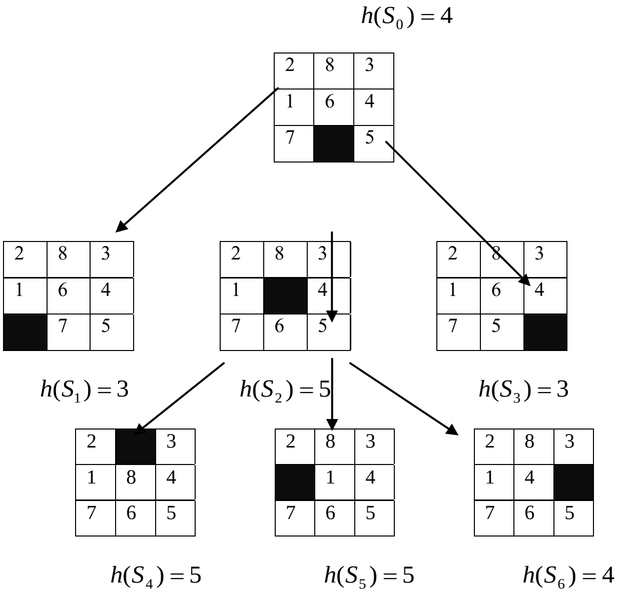
როგორც მოსალოდნელი იყო, ეს ალგორითმი ვერ აკმაყოფილებს შეფასების პირველ და მეორე კრიტერიუმებს, მაგრამ რადგანაც ეს ალგორითმი არ იმახსოვრებს არცერთ მდგომარეობას, გარდა მოცემულისა და მისი მეზობლებისა, მას არ სჭირდება დიდი მეხსიერება ამოცანის ამოხსნისათვის, რაც წარმოადგენს მის ძალიან დიდ უპირატესობას ადრე განხილულ ძებნის ალგორითმებთან შედარებით, თუმცა მას გააჩნია სერიოზული ნაკლიც. თუ დავუკვირდებით სურ. #5.2. მოცემულ სიტუაციას, ამ სიტუაციაში ალგორითმი წყვეტს მუშაობას იმის გამო, რომ არცერთი შემდეგი შვილობილი წვერო აღარ აუმჯობესებს ფუნქციას, ანუ ალგორითმმა იპოვა ლოკალური მაქსიმუმი და გაჩერდა. იმისათვის, რომ შევძლოთ ალგორითმის გამოყვანა ლოკალური მაქსიმუმის წერტილიდან, საჭიროა "ხარბ" ალგორითმში გავითვალისწინოთ შესაბამისი პროცედურა, რაც შეუძლებელი იქნება, თუ კვლავ არ გამოვიყენებთ სიებს "ღიაა' და "დახურულია", რაც საშუალებას აძლევს ალგორითმს გამოვიდეს ლოკალური მაქსიმუმის მდგომარეობიდან სხვა ტოტზე გადასვლის გზით. როგორც ადვილი მისახვედრია, ასეთი ალგორითმი კარგავს იმ უპირატესობას, რომელიც გააჩნია პიკისაკენ აღმავალ ალგორითმს და რომელიც მდგომარეობს იმ მეხსიერების მოცულობაში, რომელიც საჭიროა ამოცანის ამოსახსნელად, ამასთან, პირველი ორი კრიტერიუმით ასეთი ალგორითმი კვლავ რჩება სუსტ ალგორითმად, რადგანაც ის ვერ ხდება დასაშვები და ოპტომალური.

5.3. მოწესრიგებული ძებნის ალგორითმი. განვიხილოთ ეხლა ე.წ. A ტიპის ინფორმირებული ალგორითმი, რომელიც ამოხსნის ძებნის პროცესში იყენებს თითქმის იგივე ევრისტიკულ ფუნქციას, მაგრამ ძებნის სხვანაირი ორგანიზაციის ხარჯზე შესაძლებელი

ხდება დავაკმაყოფილოთ შეფასების პირველი და მეორე კრიტერიუმები. დავუშვათ, ჩვენ რომელიმე ალგორითმის (მაგალითად სრული გადარჩევის) საშუალებით უკვე ამოვხსენით ამოცანა და ვიპოვეთ ოპტიმალური ამონახსნი. მაშინ ამ ოპტიმალური ამონახსნის ფასი $f^*(n^/)$, რომელიც გადის რომელიმე კონკრეტულ $n^/$ წვეროზე, შეგვიძლია გავყოთ ორ ნაწილად:

$$f^*(n^/) = g^*(n^/) + h^*(n^/),$$

სადაც $g^*(n^/)$ არის ოპტიმალური გზის ფასი საწყისი წვეროდან $n^/$ წვერომდე, ხოლო $h^*(n^/)$ არის ასევე ოპტიმალური გზის ფასი $n^/$ წვეროდან მიზნის წვერომდე. ეს ფუნქცია განსაზღვრული იქნება მხოლოდ იმ წვეროებისთვის, რომლებიც მდებარეობენ ამოხსნის ოპტიმალურ გზაზე და ამიტომ შეუძლებელია გამოვიყენოთ ყველა იმ წვეროს შესაფასებლად, რომლებსაც ალგორითმი წარმოქმნის ამოცანის ამოხსნის პროცესში (მით უმეტეს, რომ სანამ ამოცანას არ ამოვხსნით $h^*(n^/)$ მნიშვნელობა ჩვენთვის უცნობია). ამიტომ A ტიპის ალგორითმებში გამოიყენება შემფასებელი ფუნქცია,



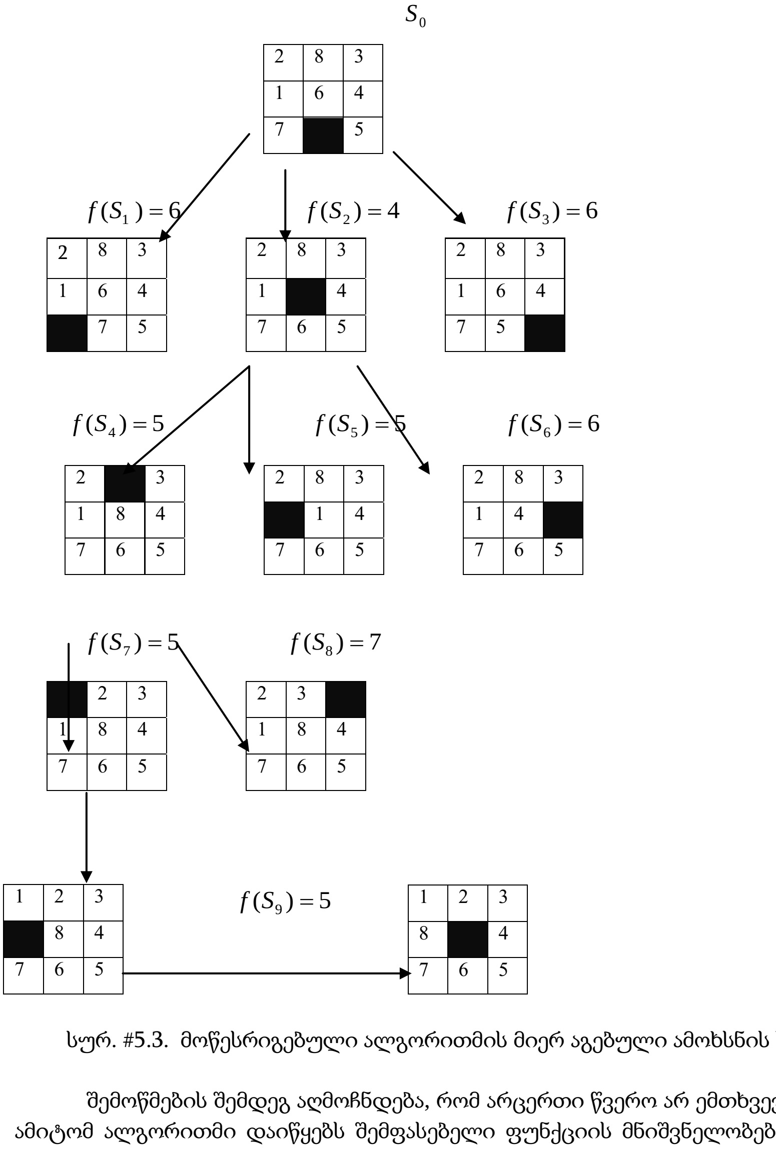
სურ. #5.2. პიკისაკენ აღმავალი ძებნის ალგორითმის მუშაობა

$$f(n) = g(n) + h(n)$$

რომელსაც გააჩნია ზუსტად იგივე სტრუქტურა, მაგრამ განსხვავებული შინაარსი. აქ $g(n)$ კვლავაც გზის ფასია (არა აუცილებლად ოპტიმალური) საწყისი წვეროდან წვერომდე, ხოლო $h(n)$ ევრისტიკული ინფორმაცია, რომელიც გამოიყენება გზის ფასის შესაფასებლად მოცემული წვეროდან მიზნის წვერომდე. იმისათვის, რომ განივი გადარჩევის ალგორითმიდან მივიღოთ A ტიპის (ხშირად მას უწოდებენ მოწესრიგებული ძებნის) ალგორითმი, საჭიროა მასში შევიტანოთ მცირეოდენი ცვლილებები. კერძოდ, პირველ რიგში, როგორ მოვძებნოთ $g^*(n^/)$ ოპტიმალური გზა რომელიმე კონკრეტული $n^/$ წვეროსათვის. თუ რომელიმე წვეროს გახსნის შემდეგ კვლავ შეგვხდება $n^/$ წვერო, რომელიც უკვე არის სიაში "ღიაა" ან "დახურულია", საჭიროა ხელახლა გამოვთვალოთ $g(n^/)$ ფუნქციის მნიშვნელობა და შევადაროთ ის უკვე არსებულს და თუ ახლადმიღებული ფუნქციის მნიშვნელობა უფრო პატარაა, შევცვალოთ არსებული შეფასება ახლით და თუ წვერო იყო სიაში "დახურულია", გადმოვიტანოთ ის სიაში "ღიაა". მეორე, სანამ ალგორითმი მერვე

ბიჯზე ახლადაგებულ წვეროებს შეიტანს სიაში “ღიაა”, ის გამოითვლის ამ წვეროების შეფასებას და ამის შემედეგ სიას “ღიაა” მოაწესრიგებს ისე, რომ პირველ ადგილზე იდგას წვეროები, რომელთა შეფასება ყველაზე პატარაა. იმისათვის რომ მივიღოთ ასეთი შემ-ფასებელი ფუნქცია ჩვენი მაგალითისათვის, $h(n)$-ით აღვნიშნოთ არა იმ ფიშკების რაო-დენობა, რომლებიც დგანან თავის ადგილზე, არამედ იმ ფიშკების რაოდენობა, რომლებიც არ დგანან თავის ადგილზე.

განვიხილოთ ეხლა ამ ალგორითმის მუშაობა კვლავ იგივე საწყისი და მიზნის მდგომარეობებისათვის. ალგორითმი კვლავ დაიწყებს მუშაობას იმით, რომ მოათავსებს საწყის წვეროს სიაში “ღიაა”. შემდეგ აიღებს პირველ წვეროს ამ სიიდან (ეს იქნება წვერო) და გახსნის რა მას, მიიღებს სამ ახალ წვეროს (იხ. სურ. #5.3.).



სურ. #5.3. მოწესრიგებული ალგორითმის მიერ აგებული ამოხსნის ხე.

შემოწმების შემდეგ აღმოჩნდება, რომ არცერთი წვერო არ ემთხვევა მიზნის წვეროს, ამიტომ ალგორითმი დაიწყებს შემფასებელი ფუნქციის მნიშვნელობების გამოთვლას ამ წვეროებისათვის. მიიღება: $f(S_1)=1+5=6$; $f(S_2)=1+3=4$; $f(S_3)=1+5=6$. ამიტომ სიაში “ღიაა”, ეს წვეროები განლაგდება შემდეგი თანმიმდევრობით: S_2, S_1 და S_3. აქედან

გამომდინარე მეორე ეტაპზე ალგორითმი აიღებს S_2 წვეროს და გახსნის მას. მიიღება სა-მი ახალი წვერო, რომელთაგან არცერთი არ ემთხვევა მიზნის წვეროს და რომელთა შეფა-სებებია შესაბამისად: $f(S_4)=5$, $f(S_5)=5$ და $f(S_6)=6$.

მეორე ეტაპის შემდეგ სიში "ღიაა" გაუხსნელი წვეროები განლაგდება შემდეგი თან-მიმდევრობით: S_5, S_6, S_1, S_3, S_4. შემდეგ ეტაპზე აიღება S_5 წვერო, რომლისაგანაც მიიღება ორი ახალი წვერო S_7 და S_8. როგორც ვხედავთ, მოწესრიგებულ ალგორითმს დასჭირდა მხოლოდ ათი წვეროს გახსნა განსხვავებით სრული გადარჩევის ალგორითმისაგან, რომე-ლიც იგივე ამოცანის ამოსახსნელად ხსნის ორმოცდაექვს წვეროს. ასევე განსხვავდება ეს ალგორითმი პიკისაკენ აღმავალი ალგორითმისაგან, რომელიც უბრალოდ ჩერდება და ველარ აგრძელებს ძებნას, მიუხედავად იმისა, რომ იყენებს იგივე ევრისტიკულ ფუნქციას.

5.4. A^* ალგორითმი და მისი შეფასებები. შევაფასოთ ალგორითმი ზემოთმოყვანილი ოთ-ხი კრიტერიუმის საშუალებით. დავიწყოთ სისრულით. ზოგადად ალგორითმის სისრულე დამოკიდებულია $h(n)$ ფუნქციის სახეზე. მაგალითად, თუ ჩვენ ავიღებთ ამ ფუნქციას ისეთს, რომელიც პირდაპირ კრძალავს ზოგიერთი წვეროების გახსნას, მაშინ თავისთავად ცხადია ალგორითმი ვერ იქნება სრული. A კლასის ალგორთმებიდან გამოყოფენ A^* კლა-სის ალგორითმებს, რომლებშიც ნებისმიერი n-თვის სრულდება პირობა, რომ $h(n)\le h^*(n)$. ასეთ შემთხვევაში შესაძლებელია ვაჩვენოთ, რომ ალგორითმი იქნება სრუ-ლი და ოპტიმალური. სანამ დავამტკიცებთ სისრულეს, ჯერ დავამტკიცოთ ასეთი ლემა.

ლემა: თუ სრულდება პირობა $h(n)\le h^*(n)$, მაშინ ნებისმიერ მომენტში, სანამ A^* ალგორითმი დაამთავრეს მუშაობას, ოპტიმალურ გზაზე S_0 წვეროდან მიზნის წვერომდე არსებობს ისეთი გაუხსნელი n' წვერო, რომ $f(n')\le f(s)$, სადაც $f(s)$ არის ოპტიმალური გზის ფასი S_0 წვეროდან მიზნის n_k წვერომდე.

დამტკიცება: დავუშვათ, $(n_0, n_1, ..., n_k)$ იყოს წვეროების მოწესრიგებული მიმდევრობა, რო-მელიც წარმოადგენს ოპტიმალურ გზას და n' არის პირველი გაუხსნელი წვერო ამ გზაზე. ასეთი წვერო უნდა არსებობდეს, რადგანაც, თუ A^* ალგორითმმა გახსნა n_k წვერო, მას და-უმთავრებია მუშაობა. მაშინ შემფასებელი ფუნქციის განმარტებიდან გამომდინარე, გვექ-ნება რომ $f(n')=g(n')+h(n')$. ჩვენ ვიცით, რომ ალგორითმმა უკვე იპოვა ოპტიმალური გზა $g(n^*)$, ამიტომ $g(n')=g^*(n')$. თუ გავითვალისწინებთ პირობას $h(n)\le h^*(n)$, მივი-ღებთ, რომ $f(n')=g^*(n')+h(n')\le g^*(n')+h^*(n')=f(s)$, ანუ $f(n')\le f(s)$, რის დამტკი-ცებაც გვინდოდა.

ამ ლემის გამოყენებით ეხლა ვაჩვენოთ, რომ თუ სრულდება პირობა, რომ ნების-მიერი n წვეროსთვის $h(n)\le h^*(n)$, მაშინ A^* ალგორითმი სრულია, ანუ ყოველთვის იპოვ-ნის ამოხსნას, თუ ეს ამოხსნა არსებობს. დავამტკიცოთ ეს დებულება საწინააღმდეგოს დაშ-ვების გზით, ანუ დავუშვათ, რომ ამოცანას ამოხსნა აქვს და ალგორითმი მას ვერ პოუ-ლობს. აქ შეიძლება გვქონდეს ორი შემთხვევა:

1) ალგორითმი ამთავრებს მუშაობას და ვერ პოულობს ამოხსნას და

2) ალგორითმი არ ამთავრებს მუშაობას. პირველი შემთხვევა შეუძლებალია. მართლაც, ალგორითმის სტრუქტურიდან გამომდინარე, ეს შემთხვევა შეიძლება დადგეს მხოლოდ მაშინ, როდესაც სია "ღიაა" იქნება ცარიელი, მაგარამ ჩვენს მიერ დამტკიცებული ლემის თანახმად, სანამ ალგორითმი დაამთავრებს მუშაობას, სიაში "ღიაა" არსებობს ისეთი გაუხსნელი n' წვერო, რომელიც მდებარეობს ამოხსნის ოპტიმალურ გზაზე და რომლის შეფასებაც ნაკლებია ოპტიმალური ამოხსნის ფასზე, ამიტომ, სანამ ალგორითმი დაამ-თავრებს მუშაობას ის აუცილებლად გახსნის ამ წვეროს და აქედან გამომდინარე, ავტო-მატურად იპოვის ამოხსნასაც. განვიხილოთ მეორე შემთხვევა, ანუ ამოხსნა არსებობს, ალ-გორითმი მას ვერ პოულობს და არც ამთავრებს მუშაობას.

დავუშვათ, t არის რომელიმე მიზნის წვერო, რომელიც საწყისი წვეროდან მი-იღწევა რაიმე სასრული k ბიჯის შემდეგ და მისი მინიმალური ფასია $f(s)$. ნებისმიერ

გადასვლას ერთი წვეროდან მეორეში აქვს რაიმე მინიმალური $\delta > 0$ ფასი მაინც, ამიტომ ნებისმიერი წვეროსთვის, რომელიც მდებარეობს $M = f(s)/\delta$ ბიჯზე უფრო შორს, გვექ-ნება

$$f(n) \geq g(n) \geq g^*(n) > M\delta = f(s).$$

აქედან გამოდის, რომ არასოდეს არ იქნება გახსნილი წვერო, რომელიც მდებარეობს M ბიჯზე შორს საწყისი წვეროდან (ზემოთ დამტკიცებული ლემის თანახმად) და თუ ალ-გორითმი არ ამთავრებს მუშაობას, ეს იმას ნიშნავს, რომ ის ხელახლა ხსნის ერთსა და იმავე წვეროებს, რომლებიც საწყისი წვეროდან დაშორებულები არიან არა უმეტეს M ბი-ჯით. ასეთი წვეროების სიმრავლე აღვნიშნოთ $\mu(M)$-ით. გზების რაოდენობა საწყისი წვე-როდან ნებისმიერ წვერომდე სასრულია. $\nu(n, M)$-თი აღვნიშნოთ გზების რაოდენობა საწ-ყისი წვეროდან n წვერომდე, რომელიც დაშორებულია საწყის წვეროს არაუმეტეს M ბი-ჯით. $\nu = \max_{n \in \mu(M)} \nu(n, M)$ იქნება ის მაქსიმალური რაოდენობა, რამდენჯერაც შეიძლება გაიხსნას ერთი წვერო, მაშინ $|\mu(M)|\nu$ ბიჯის შემდეგ ყველა წვერო, რომელიც მდებარეობს არაუმეტეს M ბიჯისა საწყისი წვეროდან, აღმოჩნდება სიაში "დახურულია" და სია "ღიაა" იქნება ცარიელი, რასაც უნდა მოჰყვეს ალგორითმის გაჩერება. ასეთ შემთხვევაში კი ძა-ლაში შევა პირველი შემთხვევა, ე. ი. A^* ალგორითმი ყოველთვის პოულობს ამოხსნას, თუკი ეს ამოხსნა არსებობს.

5.5. A^* ალგორითმი ოპტიმალურია. ვაჩვენოთ ალგორითმის ოპტიმალურობა იმ გაგებით, რომ ალგორითმის მიერ ნაპოვნი ამოხსნა იქნება მინიმალური გზა საწყისი წვეროდან მიზნის წვერომდე. დავუშვათ კვლავ საწინააღმდეგო, ანუ ალგორითმმა იპოვნა ამოხსნა, რომელიც მთავრდება რაიმე t მიზნის წვეროზე, მაგრამ ის არ არის მინიმალური გზა, ანუ $f(t) > f(s)$-ზე. მაგრამ ლემის თანახმად სიაში "ღიაა" არსებობს ისეთი გაუხსნელი n' წვერო, რომ $f(n') \leq f(s) < f(t)$, ამიტომ t წვეროს მაგივრად ალგორითმი გახსნიდა n' წვეროს და იპოვიდა ოპტიმალურ ამოხსნას. რაც შეეხება შეფასების დანარჩენ კრიტერიუმებს, დროით და სივრცით სირთულეებს, მიუხედავად იმისა, რომ სრული გადარჩევის ალგორითმთან შედარებით ისინი გაუმჯობესებულია, სამწუხაროდ ისინი კვლავაც ასიმპტოტურადაა დამოკიდებული მდგომარეობათა სივრცის ზომებზე, ამიტომ ალგორითმის გამოყენება ხდება არაეფექტური დიდი ზომის ამოცანების ამოხსნის დროს.

5.6. ალგორითმის ინფორმირებულობა. ალგორითმის ინფორმირებულობის ქვეშ ჩვენ გვესმის შემდეგი: დავუშვათ, გვაქვს ორი შემფასებელი ფუნქცია $h_1(n)$ და $h_2(n)$, ისეთი, რომ ნებისმიერი n წვეროსთვის სრულდება პირობა $h_1(n) \leq h_2(n) \leq h^*(n)$, მაშინ ალგო-რითმი, რომელიც იყენებს ევრისტიკულ ფუნქციას, ამოცანის ამოხსნის პროცესში არ გახ-სნის უფრო მეტ წვეროებს, ვიდრე ალგორითმი, რომელიც იყენებს ევრისტიკულ ფუნ-ქციას. სხვანაირად რომ ვთქვათ, რაც უფრო მივუახლოვდებით ფუნქციას ქვევიდან, მით უფრო უკეთესი იქნება ჩვენი ალგორითმი.

დავამტკიცოთ ეს დებულებაც საწინააღმდეგოს დაშვებით. დავუშვათ ალგორითმმა, რომელიც იყენებდა $h_2(n)$ ევრისტიკულ ფუნქციას, იპოვნა ამოხსნა, მაგრამ გახსნა რაიმე წვერო n'', რომელიც არ გაუხსნია ალგორითმის, რომელიც იყენებდა $h_1(n)$. ეს იმას ნიშნავს, რომ $h_2(n)$ ევრისტიკული ფუნქცის მქონე ალგორითმისათვის ამ წვეროს შეფასებაა $f(n'') = g^*(n'') + h_2(n'') \leq f(s)$, ხოლო $h_1(n)$ ევრისტიკული ფუნქციის მქონე ალგორითმისათვის კი $f(n'') = g^*(n'') + h_1(n'') \geq f(s)$. საიდანაც გამოვა, რომ $h_1(n) > h_2(n)$, რაც ეწინააღმდეგება ჩვენს დაშვებას.

საკონტროლო კითხვები:

1. რა სახის ინფორმაცია შეიძლება იყოს გამოყენებული ძებნის გასაუმჯობესებლად და საიდან უნდა მოვიპოვით ეს ინფორმაცია?

2. როგორი ამოცანების ამოსახსნელად გამოიყენება ლოკალური ძებნის ალგორითმები?
3. რა პრინციპით მუშაობს პიკისაკენ აღმავალი ძებნის ალგორითმი?
4. რა ნაკლი გააჩნია პიკისაკენ აღმავალი ძებნის ალგორითმს?
5. როგორ ალგორითმებს უწოდებენ "ხარბ" ალგორითმებს?
6. რა პრინციპიზეა დაფუძნებული მოწესრიგებული ძებნის ალგორითმი?
7. დაამტკიცეთ ალგორითმის სისრულე და ოპტიმალურობა.
8. რას ნიშნავს მოწესრიგებული ძებნის ალგორითმის ინფორმირებულობა?

ლექცია 6. ამოცანები შეზღუდვათა დაკმაყოფილებაზე.

6.1. ამოცანები შეზღუდვათა დაკმაყოფილებაზე. აქამდე ჩვენ ვიხილავდით ალგორითმებს, რომლებიც ამოცანის ამოხსნას ეძებენ მდგომარეობათა სივრცეში. ამ მიდგომის რეალიზაციისათვის საჭირო გახდა მდგომარეობათა შეფასება ევრისტიკული ფუნქციების საშუალებით, რომლებიც მჭიდროდ არიან დაკავშირებული იმ პრობლემურ გარემოსთან, რომელსაც მიეკუთვნება მოცემული ამოცანა. ამასთან, ყოველ ნაბიჯზე საჭიროა შევამოწმოთ, მივაღწიეთ თუ არა მიზნის მდგომარეობას. ამავე დროს ეს მდგომარეობები წარმოადგენენ რაღაც "შავ ყუთს", ერთნაირი შინაგანი სტრუქტურით. ეს სტრუქტურა მოცემულია მონაცემებით, რომლებთანაც წვდომა შეიძლება განხორციელდეს მხოლოდ იმ პროცედურების საშუალებით, რომლებიც მიეკუთვნება მოცემულ პრობლემურ გარემოს. ამის გარდა, ყველა ეს ამოცანა ისეთია, რომ ამოცანის ამოხსნა წარმოადგენს გზას საწყისი მდგომარეობიდან მიზნის მდგომარეობამდე.

ახლა უნდა განვიხილოთ ამოცანები ეგრეთ წოდებულ შეზღუდვათა დაკმაყოფილებაზე, რომლებშიც მდგომარეობები და მიზნის შემოწმება შეესაბამება სტანდარტულ, სტრუქტურირებულ და ძალიან მარტივ წარმოდგენას. ამასთან ამ ამოცანებში ამოხსნას წარმოადგენს მხოლოდ მიზნის მდგომარეობა და არა გზა საწყისი მდგომარეობიდან მიზნის მდგომარეობამდე. ეს საშუალებას მოგვცემს ჩამოვაყალიბოთ ძებნის ისეთი, უფრო მოხერხებული ალგორითმები, რომლებსაც შეეძლებათ ისარგებლონ მდგომარეობის ასეთი სტრუქტურირებული სახით წარმოდგენის უპირატესობით და ძებნისათვის გამოიყენონ ზოგადი დანიშნულების ევრისტიკული ფუნქციები. აგრეთვე არსებითია ისიც, რომ ის სტანდარტული წარმოდგენა მიზნის შემოწმებისა, რომელიც გამოიყენება ამ ალგორითმებში ხსნის თვით ამოცანის სტრუქტურას, და საშუალებას გვაძლევს შევქმნათ ამოცანის დეკომპოზიციის მეთოდები და დავადგინოთ შინაგანი კავშირი ამოცანის სტრუქტურასა და ამოხსნის სირთულეს შორის.

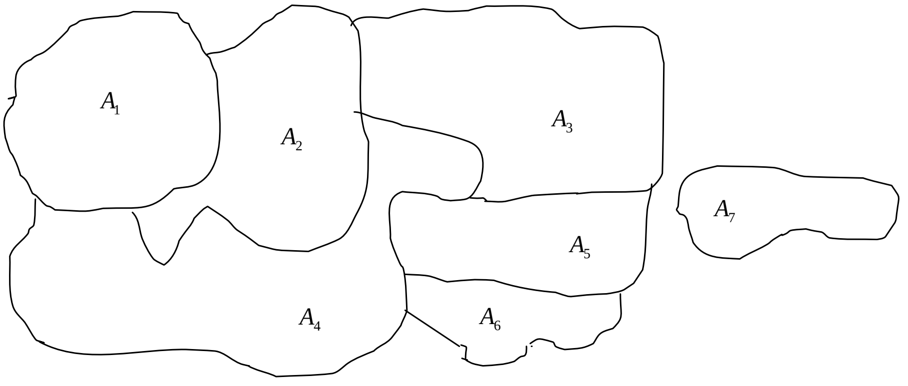
6.2. ცვლადები და შეზღუდვები. ფორმალურად, ნებისმიერი შეზღუდვათა დაკმაყოფილების ამოცანა (შდა) განსაზღვრულია $X_1, X_2, \ldots X_n$ ცვლადებითა და $C_1, C_2, \ldots, C_m$ შეზღუდვების სიმრავლეებით. თითოეულ X_i ცვლადს გააჩნია D_i შესაძლო მნიშვნელობათა არაცარიელი განსაზღვრის არე. თითოეული C_j შეზღუდვა შეიცავს ცვლადების რაიმე ქვესიმრავლეს და განსაზღვრავს ამ ქვესიმრავლისათვის დასაშვებ კომბინაციებს. ამოცანის მდგომარეობა განისაზღვრება ცვლადისათვის მნიშვნელობათა მინიჭების გზით. მინიჭებას, რომელიც არ არღვევს არცერთ შეზღუდვას, ეწოდება დასაშვები ან თავსებადი მინიჭება. სრული ეწოდება ისეთ მინიჭებას, რომელშიც მონაწილეობს ყველა ცვლადი, ხოლო ამოცანის ამონახსნი ეწოდება ისეთ სრულ მინიჭებას, რომელიც აკმაყოფილებს ყველა შეზღუდვას. გარდა ამისა, ზოგიერთ ამოცანებში საჭიროა ისეთი ამონახსნის პოვნა, რომელიც ახდენს მიზნის ფუნქციის ოპტიმიზაციას.

6.3. ამოცანის წარმოდგენა ცვლადებისა და შეზღუდვების საშუალებით. დავუშვათ, მოცემული გვაქვს რუკა, რომელიც შედგება შვიდი რეგიონისაგან და ჩვენი ამოცანაა გავაფერადოთ ეს რუკა სამი ფერის (წითელი, მწვანე, ცისფერი) საშუალებით ისე, რომ მეზობელ რეგიონებს არ ჰქონდეთ ერთი და იგივე ფერი (იხ. სურ. #6.1).

ჩამოვაყალიბოთ ეს ამოცანა შდა–ს ენაზე. აღვნიშნოთ რეგიონები ცვლადებით: $A_1, A_2, A_3, A_4, A_5, A_6, A_7$. ყველა ცვლადის დასაშვებ მნიშვნელობათა სიმრავლე შედგება სამი ფერისაგან {წითელი –1, მწვანე _2, ცისფერი _3}. შეზღუდვები მოითხოვენ, რომ ყველა წყვილი გაფერადებული იყოს სხვადასხვა ფერად. მაგალითად, A_1 და A_2 ცვლადებისათვის დასაშვებია ფერთა შემდეგი კომბინაციები:

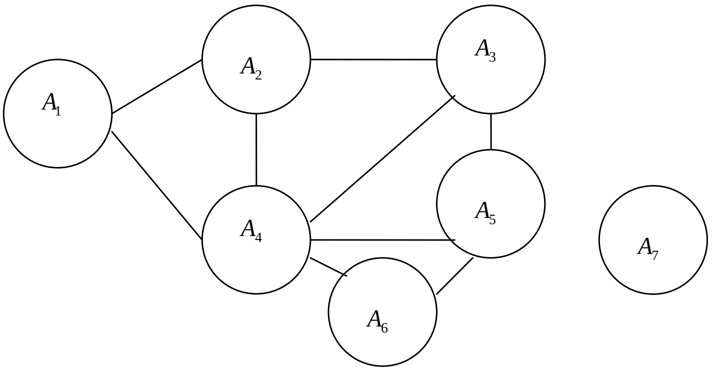
$$\{(1,2);(1,3);(2,1);(2,3);(3,1);(3,2)\}.$$

შესაძლებელია ამ შეზღუდვების უფრო მარტივი სახით ჩაწერაც, მაგალითად, , თუკი ალგორითმში არსებობს ისეთი პროცედურა, რომელიც შეძლებს მის წაკითხვას.



სურ. 6.1. ამოცანა რუკის გაფერადებაზე

ზოგადად ასეთი ამოცანები შეზღუდვებზე შესაძლებელია წარმოვადგინოთ გრაფის სახით, სადაც კვანძები შეესაბამებიან ცვლადებს, ხოლო რკალები შეზღუდვებს (იხ. სურ. #6.2). ამოცანის შდა-ს სახით წარმოდგენა საშუალებას გვაძლევს, მივაღწიოთ მნიშვნელოვან უპირატესობას ამოცანის მდგომარეობათა სიმრავლის სახით წარმოდგენასთან შედარებით:



სურ. #6.2. ამოცანის წარმოდგენა გრაფის საშუალებით.

1. მდგომარეობის წარმოდგენა ხდება სტანდარტული შაბლონით (ცვლადები, მათი მნიშვნელობები და შეზღუდვები), ამიტომ შვილობილის განმსაზღვრელი ფუნქცია

(ოპერატორები) და მიზნის შემოწმება შეიძლება ჩავწეროთ უნივერსალური ფორმით, ერთნაირად ყველა ამოცანისათვის.

2. შეიძლება დამუშავებული იქნას ეფექტური, სტანდარტული ევრისტიკული ფუნქციები, რომელთა შესაქმნელად საჭირო აღარ იქნება დამატებითი ცოდნა კონკრეტული პრობლემური გარემოს შესახებ.
3. ამოცანის ამოხსნის პროცესის გასამარტივებლად შეიძლება გამოვიყენოთ შეზღუდვათა გრაფის სტრუქტურა, რაც საშუალებას მოგვცემს ექსპონენციალურად შევამციროთ სირთულე.

ადვილი საჩვენებელია, რომ **შდა** კლასის ყველა ამოცანა შეიძლება აღწერილი იქნას ისევე, როგორც ამოცანები მიზნის ძებნაზე. ჩვენ აღვწერდით ამოცანას ხუთი კომპონენტის საშუალებით: მდგომარეობათა სიმრავლე; საწყისი მდგომარეობა; შვილობილი წვეროს განმსაზღვრელი ფუნქცია (ოპერატორების სიმრავლე); მიზნის მდგომარეობა და გზის ფასი. ამ შემთხვევაში საწყისი მდგომარეობა იქნება ცარიელი მინიჭება, რომლის დროსაც არცერთ ცვლადს არ მიენიჭება მნიშვნელობა. შვილობილი წვეროს განმსაზღვრელი ფუნქცია – რომელიმე ცვლადისათვის დასაშვები მნიშვნელობის მინიჭება. მიზნის შემოწმების პროცედურა ტოლფასია შევამოწმოთ, მიმდინარე მინიჭება სრულია თუ არა. გზის ფასი მუდმივი სიდიდე იქნება ყველა გადასვლისათვის (ყველა ცვლადისათვის რაიმე მნიშვნელობის მინიჭებისათვის).

ამ ამოცანებში ყველა ამოხსნა უნდა წარმოადგენდეს სრულ მინიჭებას და ამიტომ უნდა იმყოფებოდეს n დონეზე, სადაც n არის ცვლადების რაოდენობა. გარდა ამისა ხის სიღრმე აუცილებლად იქნება აგრეთვე n-ის ტოლი, ამიტომ **შდა** ამოცანების ამოხსნის დროს ფართოდ გამოიყენება სიღრმეში გადარჩევის ალგორითმები. ამ ამოცანებისათვის დამახასიათებელია აგრეთვე ის რომ, არავითარ ინტერესს არ წარმოადგენს ის გზა, რომლითაც მიიღწევა ამოხსნა, ამიტომაც შეიძლება მოვახდინოთ ამოცანის ფორმულირება სრულ მდგომარეობათა ენაზე, რომელშიც ყველა მდგომარეობა წარმოადგენს სრულ მინიჭებას, რომლებიც ან აკმაყოფილებენ შეზღუდვებს ან არა. ასეთ შემთხვევაში უფრო მოსახერხებელია ლოკალური ძებნის ალგორითმების გამოყენება.

უფრო მარტივი ამოცანებისათვის დამახასიათებელია დისკრეტული ცვლადები, რომელთაც აქვთ სასრული განსაზღვრის არეები. ასეთია ჩვენს მიერ განხილული მაგალითი. ასეთივეა რვა ლაზიერის ამოცანა, რომელშიც $Q_1, Q_2, ..., Q_8$ ცვლადებით შეგვიძლია აღვნიშნოთ თითოეული ლაზიერის პოზიცია 1, 2, . . ., 8 ვერტიკალებზე. თითოეული ცვლადის მნიშვნელობათა სიმრავლეა – {1,2,3,4,5,6,7,8}. თუ ნებისმიერი ცვლადის განსაზღვრის არის მაქსიმალური ზომაა d , მაშინ ყველა შესაძლო სრულ მინიჭებათა რაოდენობა იზომება სიდიდით O(d^n). მიუხედავად იმისა, რომ შეფასება აქაც ექპონენციალურია, ალგორითმები, რომლებიც გამოიყენება შდა ამოცანების ამოსახსნელად უმეტეს შემთხვევაში საშუალებას გვაძლევენ ამოვხსნათ რამდენიმე რიგით უფრო რთული ამოცანები.

დისკრეტულ ცვლადებსაც შეიძლება ჰქონდეთ უსასრულო განსაზღვრის არე. ასეთ შემთხვევაში უკვე შეუძლებელია შეზღუდვათა აღწერა ცვლადების დასაშვებ მნიშვნელობათა კომბინაციების ჩამოთვლით. ამ შემთხვევაში საჭიროა გამოვიყენოთ შეზღუდვათა აღწერის ენა. ყველაზე ხშირად ამ მიზნით გამოიყენება ალგებრული აღწერა. მაგალითად, თუ ორი ცვლადიდან x და y , ერთერთის მნიშვნელობები არ შეიძლება იყოს ნაკლები რაიმე მუდმივი სიდიდით მეორეზე, გვექნება შეზღუდვა $x + a < y$. ამ დროს უკვე აღარ შეიძლება ასეთი ამოცანების ამოხსნა უბრალოდ ყველა შესაძლო მინიჭებათა ჩამოთვლის გზით, რადგანაც ასეთ მინიჭებათა რაოდენობა იქნება უსასრულო სიმრავლე. **შდა** ამოცანებში ჩვენ არ განვიხილავთ ამოცანებს მთელმნიშვნელიანი ცვლადებითა და წრფივი შეზღუდვებით და ამოცანებს ცვლადთა უწყვეტი განსაზღვრის არეებით. პირველი ტიპის ამოცანები შეისწავლება წრფივ პროგრამირებაში, ხოლო მეორე სახის – ოპერაციათა გამოკვლევაში.

6.4. შეზღუდვათა ტიპები. გარდა ცვლადების ტიპების დახასიათებისა, სასარგებლოა განვიხილოთ შეზღუდვათა ტიპებიც. შეზღუდვები შეიძლება იყოს უნარული, ანუ ეხებოდეს

მხოლოდ ერთ ცვლადს. მაგალითად, დავუშვათ, რუკაზე A_1 რეგიონი არ შეიძლება შეიღებოს წითლად, რადგანაც მის მფლობელს არ უყვარს წითელი ფერი. უნარული შეზღუდვა გადაიჭრება მარტივად, განსაზღვრის არიდან შესაბამისი მნიშვნელობის ამოგდებით. უფრო რთულია ბინარული შეზღუდვები, რომლებიც მოქმედებენ ორ ცვლადზე. მაგალითად, $A_1 \neq A_2$.

უფრო მაღალი რიგის შეზღუდვებში მონაწილეობენ სამი და უფრო მეტი ცვლადები. ასეთი ამოცანების კარგი მაგალითია ე.წ. რიცხვითი რებუსები, რომლებშიც რიცხვები შეცვლილია ასოებით. მაგალითად, $TWO+TWO=FOUR$. აქ მოთხოვნაა, რომ ექვსივე ცვლადი (F,T,U,W,R,O) ამ გამოსახულებაში ღებულობს სხვადასხვა მნიშვნელობებს, შეიძლება ჩაწერილი იქნას, როგორც შეზღუდვა ექვსი ცვლადით - **ყველა განსხვ** (F,T,U,W,R,O) ან სხვანაირად ბინარული შეზღუდვების სიმრავლით $(F \neq T)$. შეზღუდვები შეკრებაზე სვეტებში ასევე შეიცავს დამატებით ცვლადებს:

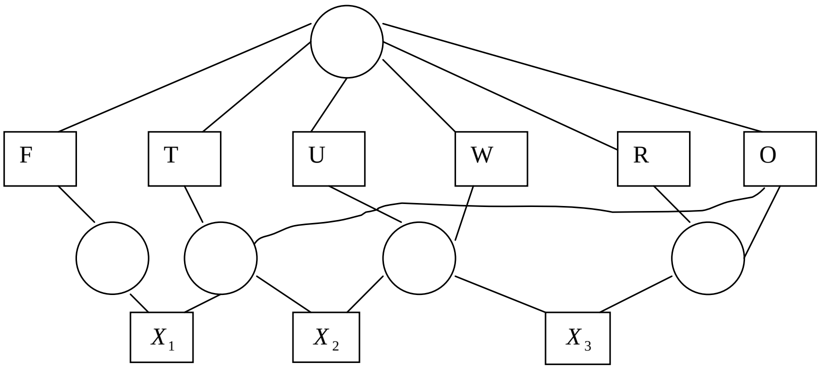
$O+O=R+10 \cdot X_1$

$X_1+W+W=U+10X_2$

$X_2+T+T=O+10X_3$

$X_3=F$

სადაც X_1, X_2, X_3 დამატებითი ცვლადებია, მნიშვნელობებით 0 ან 1, რომელიც გადაიტანება შემდეგ თანრიგში. მაღალი რიგის შეზღუდვები შეიძლება წარმოვადგინოთ შეზღუდვის ჰიპერგრაფის საშუალებით ისე, როგორც ეს მოყვანილია სურ. #6.3 როგორც ამ მაგალითიდანაც ჩანს, მაღალი რიგის შეზღუდვები შეიძლება დავშალოთ ბინარულ შეზღუდვებად დამატებითი ცვლადების შემოტანის გზით, ამიტომ შემდგომში განვიხილავთ მხოლოდ ბინარულ შეზღუდვებს.

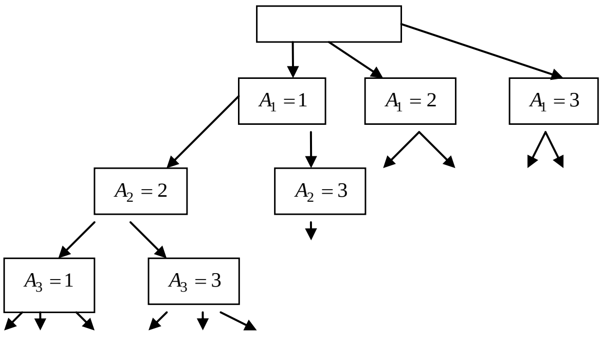


სურ. #6.3. შეზღუდვის ყველაგანსხვ ( ) ჰიპერგრაფი.
აქ რგოლებით აღნიშნულია შეზღუდვები, რომლებიც დაკავშირებულები არიან მათ მიერ შეზღუდულ ცვლადებთან

6.5. კომუტატურობა. ჩვენ ვნახეთ, რომ შდა ამოცანები შეიძლება წარმოვადგინოთ როგორც ძებნის ამოცანები და შესაბამისად შეიძლება მათ ამოსახსნელად გამოვიყენოთ ძებნის ნებისმიერი ალგორითმი. ვნახოთ რა მოხდება თუკი გამოვიყენებთ განივი გადარჩევის ალგორითმს, როდესაც ცვლადების რაოდენობაა n და თითოეულ მათგანს შეუძლია მიიღოს d მნიშვნელობა. პირველ ეტაპზე გვექნება $n \cdot d$ შესაძლო ვარიანტი, რადგანაც ნებისმიერი მნიშვნელობა შეგვიძლია მივანიჭოთ ნებისმიერ ცვლადს. მეორე ეტაპზე გვექნება უკვე $(n-1) \cdot d$ ვარიანტი და საბოლოოდ, მივიღებთ ხეს $n! \cdot d^n$ წვეროებით, მაშინ როდესაც სინამდილეში არსებობს მხოლოდ d^n შესაძლო სრული მინიჭება.ეს ხდება იმიტომ, რომ არ იქნა გათვალისწინებული ყველა **შდა** ამოცანებისათვის დამახასიათებელი

თვისება - **კომუტატურობა**. ამოცანას ეწოდება კომუტატურული, თუ ამოხსნის პროცესში გამოყენებული ნებისმიერი კონკრეტული ქმედებების რიგს არა აქვს მნიშვნელობა. სწორედ ეს თვისება ახასიათებს შდა ამოცანებს, რადგანაც რა რიგითაც არ უნდა მივაკუთვნოთ ცვლადებს მნიშვნელობები, ჩვენ ვღებულობთ ერთსა და იმავე ნაწილობრივ მინიჭებას. ამიტომ ყველა ალგორითმში, რომელიც გამოიყენება შდა ამოცანების ამოსახსნელად, შვილობილი წვეროები წარმოიქმნება იმის გათვალისწინებოთ, რომ ხის თითოეულ კვანძში ხდება მნიშვნელობის მინიჭება მხოლოდ ერთი ცვლადისათვის. მაგალითად, თუ დავუბრუნდებით რუკის გაფერადების ამოცანას, საწყის მდგომარეობაში უნდა ამოვირჩიოთ $A_1 = 1$, $A_1 = 2$ და $A_1 = 3$ ვარიანტებიდან ერთერთი და არა $A_1 = 1, A_2 = 2$ ვარიანტებიდან რომელიმე. მხოლოდ ამ პირობის დაცვით შეგვიძლია შევამციროთ გადარჩევა d^n-მდე.

6.6. ძებნა უკან დაბრუნებით. ძებნას სიღრმეში, რომლის დროსაც ყოველ ჯერზე ამოირჩევა მნიშვნელობები ერთი ცვლადისათვის და ხდება უკან დაბრუნება, როდესაც აღარ რჩება დასაშვები მნიშვნელობები, რომელთა მინიჭებაც შესაძლებელია ამ ცვლადისათვის, ეწოდება ძებნა უკან დაბრუნებით. ალგორითმი ძალიან მარტივია. საწყის მდგომარეობაში (არცერთ ცვლადს არა აქვს მინიჭებული რაიმე მნიშვნელობა) აიღებს პირველ ცვლადს და მიანიჭებს ყველა დასაშვებ მნიშვნელობას. მეორე ეტაპზე აიღებს პირველ წვეროს და მეორე ცვლადს მიანიჭებს ყველა დასაშვებ მნიშვნელობას. თუ მიაღწევს სასრულ მდგომარეობას (ყველა ცვლადს აქვს მინიჭებული დასაშვები მნიშვნელობები), ამოცანა იქნება ამოხსნილი, თუ არა – დაბრუნდება ერთი ნაბიჯით



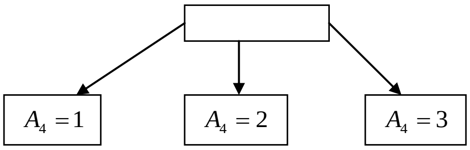
სურ. #6.4. ალგორითმის აგებული ხის ნაწილი რუკის გაფერადების ამოცანისათვის

უკან და განიხილავს სხვა წვეროს. ცხადია ეს ალგორითმი წარმოადგენს არაინფორმირებული ძებნის ალგორითმს და ამიტომ მისი გამოყენება დიდი ამოცანების ამოსახსნელად არაეფექტურია. ასეთ შემთხვევებში ალგორითმების ეფექტურობის გასაუმჯობესებლად ვიყენებდით ევრისტიკულ ინფორმაციას, რომელიც გამომდინარეობდა ჩვენი ცოდნიდან მოცემული კონკრეტული ამოცანის შესახებ.

6.7. ძებნა უკან დაბრუნებით დმმ-ის გამოყენებით. ეხლა ვნახოთ, როგორ შეიძლება შდა ამოცანებისათვის მოვახდინოთ გადარჩევის ეფექტური შემცირება ისე, რომ გამოვიყენოთ არა რაიმე ევრისტიკული ინფორმაცია კონკრეტული ამოცანის შესახებ, არამედ მხოლოდ ინფორმაცია ამოცანის ფორმალიზაციის პრინციპების შესახებ ამოცანებში შეზღუდვათა დაკმაყოფილებაზე. ამისათვის დავუკვირეთ როგორ მუშაობს ზემოთგანხილული ალგორითმი. ის ცვლადებს ირჩევს იმ მიმდევრობით, რა მიმდევრობითაც ისინი შეტანილია სიაში "ცვლადები" და ანიჭებს მათ დასაშვებ მნიშვნელობებს ასევე რიგის მიხედვით სიიდან "მნიშვნელობები". ეს მიდგომა ცხადია ყოველთვის ვერ იქნება ეფექტური. იგივე ჩვენი მაგალითის შემთხვევაში, მას შემდეგ, რაც ალგორითმი მიანიჭებს მნიშვნელობებს A_1 და A_2 ცვლადებს, მას A_4 ცვლადისათვის რჩება ერთადერთი დასაშვები მნიშვნელობა,

ამიტომ ლოგიკურად უფრო სწორი იქნება, თუ შემდეგ ეტაპზე ალგორითმი აიღებს A_4 ცვლადს და არა A_3 ცვლადს. მართლაც, თუ მესამე ეტაპზე ალგორითმი აიღებს A_4 ცვლადს, ყველა დარჩენილ ცვლადებს გარდა A_7 დარჩებათ ერთადერთი დასაშვები მნიშ-ვნელობები და ამოცანა სწრაფად გადაიჭრება. ეს დასკვნა, რომ ყოველ ეტაპზე უმჯობესია ამოვარჩიოთ ცვლადი დასაშვები მნიშვნელობების მინიმალური რაოდენობებით, შე-გვიძლია ჩამოვაყალიბოთ პირველი ევრისტიკის სახით- დარჩენილ მნიშვნელობათა მინი-მუმი (დმმ), ან სხვანაირად ცვლადი ყველაზე მეტი შეზღუდვებით. ასეთი ევრისტიკის გამოყენება იძლევა კიდევ იმ უპირატესობასაც, რომ თუ რომელიმე ცვლადის დასაშვებ მნიშვნელობათა სიმრავლე გახდება ნოლის ტოლი, ალგორითმი პირველ რიგში ამოირჩევს ამ ცვლადს, მივა წარუმატებლობამდე და მოკვეთავს ამ ტოტს. დავარქვათ ამ ალგორითმს ძებნა უკან დაბრუნებით დმმ–ის გამოყენებით. ამ ალგორითმის ეფექტურობა წინა ალგო-რითმთან შედარებით შეიძლება გაიზარდოს 3-დან 3000 ჯერ ამოცანის მიხედვით.
6.8. ხარისხოვანი ევრისტიკა. სამწუხაროდ, ეს ევრისტიკა არაფერს არ გვეუბნება პირველი ცვლადის ამორჩევაზე, რადგანაც ხშირ შემთხვევაში (ისევე როგორც ჩვენს მაგალითში), საწყის მდგომარეობაში ყველა ცვლადს გააჩნია ერთი და იგივე დასაშვებ მნიშვნელობათა სიმრავლე. ამ შემთხვევაში მოსახერხებელია ხარისხოვანი ევრისტიკის გამოყენება. ეს ევ-რისტიკა ცდილობს შეამციროს ხის დატოტვის კოეფიციენტი იმის ხარჯზე, რომ პირ-ველად ამოირჩევს ცვლადს, რომელიც მონაწილეობს ყველაზე მეტ შეზღუდვებში იმ ცვლადებთან, რომლებსაც ჯერ არა აქვთ მინიჭებული მნიშვნელობები. ჩვენს მაგალითში ასეთ ცვლადს წარმოადგენს A_4 ცვლადი. ეს ევრისტიკა დმმ ევრისტიკასთან შედარებით ნაკლებად ამცირებს გადარჩევას, მაგრამ ის ძალიან სასარგებლოა გაურკვეველ სიტუ-აციებში.

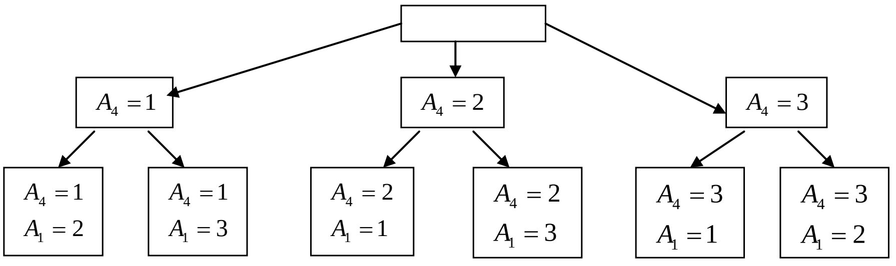
მას შემდეგ, რაც ამოვირჩევთ რომელიმე ცვლადს, საჭიროა დავადგინოთ მისი მნიშვნელობების მინიჭების რიგი. ამისათვის შეიძლება გამოვიყენოთ კიდევ ერთი ეფექტური ევრისტიკა - მინიმალურად შემზღუდავი მნიშვნელობის ამორჩევის ევრისტიკა. ეს ევრისტიკა უპირატესობას ანიჭებს ცვლადის იმ მნიშვნელობებს, რომლებიც მინი-მალურად ზღუდავენ დანარჩენი ცვლადებისთვის მნიშვნელობათა მინიჭების ვარიანტებს. იმავე მაგალითში თუ პირველ ცვლადს მინიჭებული აქვს მნიშვნელობა ერთი, მეორე ცვლადს კი მნიშვნელობა – ორი და ვიხილავთ მესამე ცვლადს, დასაშვები ორი ვარიანტიდან ეს ევრისტიკა ამოირჩევს მნიშვნელობას ერთი, რადგან სამის მინიჭება მეოთხე ცვლადს აღარ უტოვებს დასაშვებ მნიშვნელობას. თუ ჩვენ ვეძებთ ამოცანის ყველა ამოხსნას, ან ამოცანას არ გააჩნია ამონახსენი, მაშინ ამ ევრისტიკის გამოყენებას აზრი არა აქვს, ყველა დანარჩენ შემთხვევაში კი ეს ევრისტიკაც ამცირებს გადარჩევას.



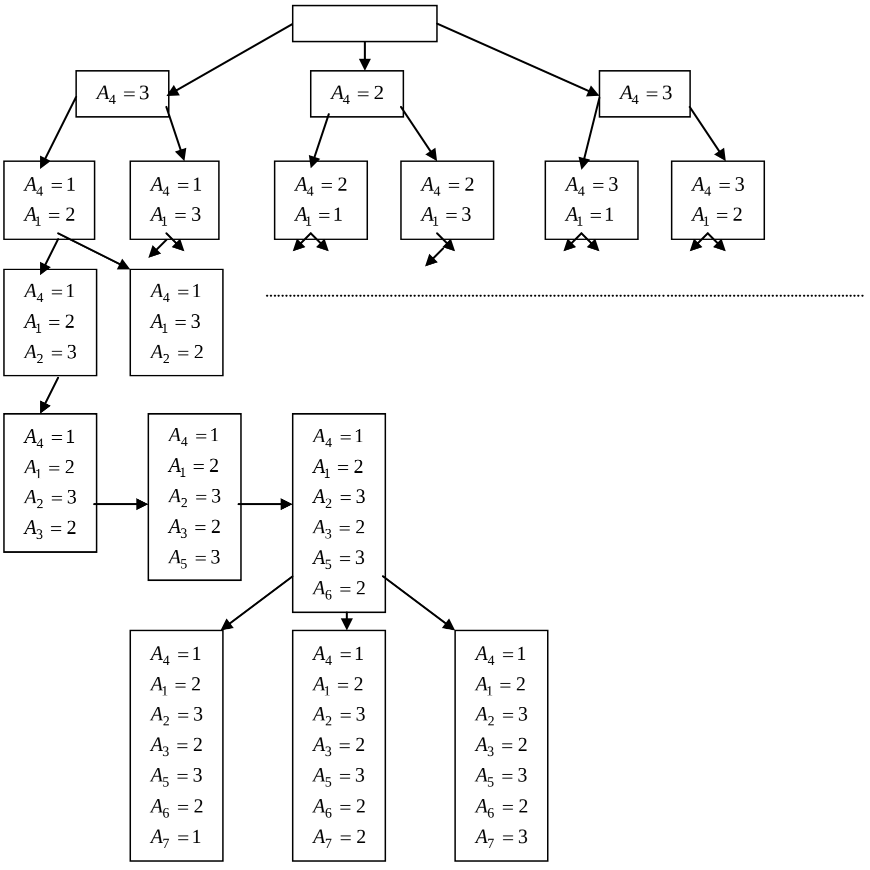
სურ. #6.5. ალგორითმის მუშაობის პირველი ეტაპი.

ვნახოთ, როგორ იმუშავებს ასეთი ევრისტიკებით აღჭურვილი ალგორითმი ჩვენი მაგალითის შემთხვევაში. ხარისხოვანი ევრისტიკის გამოყენებით ის პირველ ცვლადად ამოირჩევს A_4 ცვლადს, რადგანაც ის დაკავშირებულია ხუთ ცვლადთან. მინიმალურად შემზღუდავი მნიშვნელობის ამორჩევის ევრისტიკა აქ ვერ იმუშავებს იმიტომ, რომ ყველა დანარჩენი ცვლადების (გარდა A_7, რომლის მნიშვნელობათა სიმრავლე არ იზღუდება) დასაშვებ მნიშვნელობათა სიმრავლე იზღუდება ერთნაირად. აქედან გამომდინარე, პირველი ეტაპის შემდეგ ხეს ექნება შემდეგი სახე (იხ. სურ. #6.5.).

მეორე ეტაპზე ალგორითმი ეძებს ცვლადს, რომლის დასაშვებ მნიშვნელობათა სიმრავლე მინიმალურია. ასეთებია A_1, A_2, A_3, A_5, A_6, ამიტომ ის ამოირჩევს A_1-ს და მიიღება შემდეგი ხე (იხ სურ. #6.6.):



სურ. # 6.6. ალგორითმის მუშაობის მეორე ეტაპი.



სურ. #6.7 რუკის გაფერადების ამოცანის ამოხსნა

მესამე ეტაპზე ერთი (მინიმალური) დასაშვები მნიშვნელობა ექნება A_2 ცვლადს, შემდეგ ეტაპზე A_3 -ს და ასე გაგრძელდება სანამ ალგორითმი არ მიანიჭებს მნიშვნელო-

ბებს პირველ ექვს ცვლადს. შემდეგ დარჩება მხოლოდ მეშვიდე ცვლადი, რომელმაც შეიძლება მიიღოს სამი მნიშვნელობა დამოუკიდებლად სხვა ცვლადებისაგან, ამიტომ გვექნება ამოცანის თვრამეტი სხვადასხვა ამოხსნა (იხ.სურ. #6.7).

6.9. წინასწარი შემოწმება. აქამდე განხილულ ალგორითმში შეზღუდვა, რომელიც ვრცელდებოდა რომელიმე ცვლადზე, მხედველობაში მიიღებოდა მხოლოდ მაშინ, როდესაც ხდებოდა ამ ცვლადისათვის მნიშვნელობის მინიჭება. მაგრამ თუ ზოგიერთ შეზღუდვას განვიხილავთ უფრო ადრეულ ეტაპებზე, ან საერთოდ ძებნის დაწყებამდე, შესაძლებელია მკვეთრად შევამციროთ ძებნის სივრცე.

ერთერთ საშუალებას ძებნის დროს შეზღუდვების უკეთ გამოყენებისა წარმოადგენს წინასწარი შემოწმების მეთოდი, რომელიც მდგომარეობს შემდეგში: ცვლადისთვის რაიმე მნიშვნელობის მინიჭებისას ხდება ყველა იმ ცვლადის შემოწმება, რომლებსაც ჯერ არა აქვთ მინიჭებული მნიშვნელობა და რომლებიც რაიმე შეზღუდვით დაკავშირებულნი არიან იმ ცვლადთან, რომელსაც ვანიჭებთ რაიმე მნიშვნელობას და მათი განსაზღვრის არედან ამოიშლება ის მნიშვნელობები, რომლებიც შეუთავსებელნი არიან ცვლადისთვის მინიჭებულ მნიშვნელობასთან. სურ. #6.8. მოცემულია წინასწარი შემოწმების მეთოდის გამოყენება ჩვენს მიერ ზემოთ განხილული მაგალითისთვის. პირველ სტრიქონში მოცემულია საწყისი განსაზღვრის არეები ძებნის დაწყებამდე. მეორე სტრიქონში განსაზღვრის არეები მას შემდეგ, რაც პირველ ცვლადს მივანიჭეთ მნიშვნელობა 1. მესამე სტრიქონში – პირველ ცვლადს მიენიჭა მნიშვნელობა ერთი და მესამე ცვლადს – 2. მეოთხე სტრიქონში პირველს –1, მესამეს – 2, მეხუთს – 3. როგორც ამ მაგალითიდან ჩანს, მას შემდეგ, რაც პირველ ცვლადს მივანიჭეთ მნიშვნელობა 1 და მესამეს მნიშვნელობა –2, მეორე და მეოთხე ცვლადებს დარჩათ ერთადერთი დასაშვები მნიშვნელობები. ეს რა თქმა უნდა ამცირებს ძებნის ხის დატოტვას, რასაც აუცილებლად გამოიყენებს დმმ ევრისტიკა. მეორე, რაც ასევე ძალიან მნიშვნელოვანია, არის ის, რომ მეხუთე ცვლადისათვის მნიშვნელობის 3 მინიჭების შემდეგ მეოთხე ცვლადის დასაშვებ მნიშვნელობათა სიმრავლე აღმოჩნდა ცარიელი, რაც იმას ნიშნავს, რომ ეს კომბინაცია შეუთავსებელია შეზღუდვებთან, ამიტომ ალგორითმი აღარ გააგრძელებს მის განხილვას და დაბრუნდება უკან.

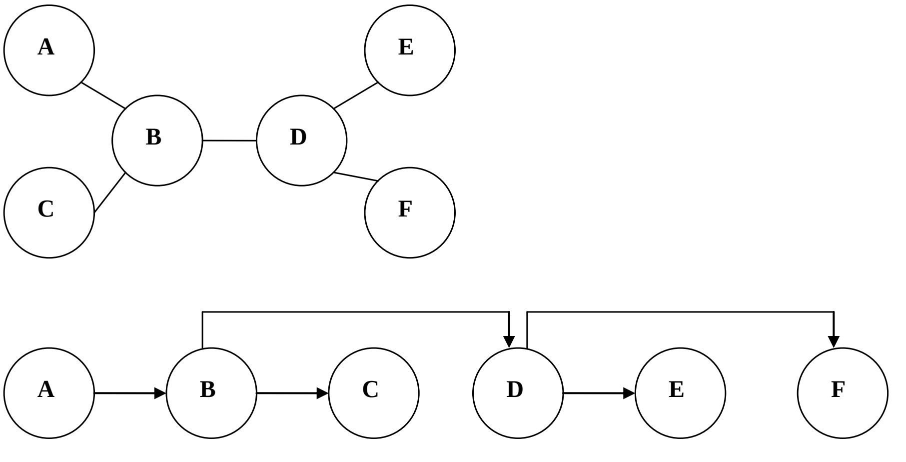
A_1	A_2	A_3	A_4	A_5	A_6	A_7
1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
1	2 3	1 2 3	2 3	1 2 3	1 2 3	1 2 3
1	3	2	3	1 3	1 2 3	1 2 3
1	3	2		3	1 2	1 2 3

სურ. #6.8. წინასწარი შემოწმების მეთოდის ცხრილი

მიუხედავად იმისა, რომ წინასწარი შემოწმების მეთოდი საშუალებას გვაძლევს აღმოვაჩინოთ ბევრი შეუსაბამოზა, მას მაინც არ შეუძლია ყველაფრის დანახვა. მაგალითად, მესამე სტრიქონში, პირველი ცვლადისათვის 1 მინიჭებისა და მესამე ცვლადისათვის 2 მინჭების შემდეგ მეორე და მეოთხე ცვლადებს რჩებათ მხოლოდ მნიშვნელობა 3. ეს რეგიონები კი ერთმანეთის მეზობლები არიან და არ შეუძლიათ მიიღონ ერთი და იგივე მნიშვნელობები. ასეთ შემთხვევებში შესაძლებელია გამოვიყენოთ სხვადასხვა მეთოდები, რომლებიც გაერთიანებულნი არიან ერთი სახელის ქვეშ შეზღუდვათა გავრცელება. ყველაზე სწრაფი ასეთ მეთოდებს შორის არის რკალების თავსებადობის შემოწმება. ამ შემთხვევაში რკალის ქვეშ გვესმის ორიენტირებული გადასვლა შეზღუდვათა გრაფზე, ისეთი, როგორიცაა მაგალითად რკალი A_6-დან A_4-ზე. განსაზღვრის თანახმად, რკალი ერთი წვეროდან (ერთი ცვლადიდან) მეორისკენ თავსებადია, თუ პირველი წვეროს (პირველი ცვლადის) ყოველი მნიშვნელობისათვის არსებობს მეორე ცვლადის რაიმე მნიშვნელობა, რომელიც დასაშვებია (თავსებადია) პირველი ცვლადის მნიშვნელობასთან. ჩვენი მაგალითისათვის

დავუშვათ, A_5 ცვლადის დასაშვები მნიშვნელობაა {1,3} და A_4 ცვლადის კი-3 (როგორც ესაა წინასწარი შემოწმების ცხრილის მესამე სტრიქონში). მაშინ ორიენტირებული რკალი A_4-დან A_5-ში თავსებადია, რადგანაც თუ A_4-ს მივანიჭებთ მნიშვნელობას 3, A_5-ს შეიძლება მივანიჭოთ 1, მაგრამ პირიქით ორიენტირებული რკალი უკვე აღარაა თავსება-დი, რადგანაც , თუ A_5-ს მივანიჭეთ მნიშვნელობა 3, მაშინ A_4-ს აღარ დარჩება დასაშვები მნიშვნელობა.

6.10. ამოცანის დეკომპოზიცია. არსებობენ ასეთი ტიპის ამოცანების ამოხსნის უფრო სწრაფი მეთოდები, რომლებიც იყენებენ ამოცანის სტრუქტურას და ახდენენ ამოცანის დეკომპოზიციას ქვეამოცანებად, რომლებიც შემდეგ ცალცალკე ამოიხსნება. ჩვენს მიერ განხილულ მაგალითში, პირველი ექვსი რეგიონის გაფერადება არანაირ გავლენას არ ახდენს მეშვიდე რეგიონზე, ანუ ეს ქვეამოცანები არიან დამოუკიდებელი ქვეამოცანები, ამას აჩვენებს ამოცანის გრაფიც, რომელზეც მეშვიდე რეგიონი არ არის დაკავშირებული არცერთ სხვა რეგიონთან. მაშინ ჩვენი ძირითადი ამოცანა შეიძლება გავყოთ ორ ქვე-ამოცანად: პირველი ექვსი რეგიონის გაფერადება და მეშვიდე რეგიონის გაფერადება. ამ შემთხვევაში ამოცანის ამოხნის სიღრმე შემცირდება შვიდიდან ექვსამდე.ეს რა თქმა უნდა არც თუ ისე ბევრია, მაგრამ წარმოვიდგინოთ, რომ გვაქვს ამოცანა, რომელიც შეიცავს n ცვლადს და ეს ამოცანა იყოფა ისეთ დამოუკიდებელ ქვეამოცანებად, სადაც თითოეული ქვეამოცანა შეიცავს c ცვლადს (რადგანაც ქვეამოცანები დამოუკიდებელია, ერთი ქვე-ამოცანის ცვლადები არ მონაწილეობენ მეორე ქვეამოცანაში), ამიტომ სულ გვექნება n/c ქვეამოცანა და თითოეული ქვეამოცანის ამოხსნა მოითხოვს მაქსიმუმ d^c ვარიანტს, ამიტომ მთელი ამოცანის სირთულე იქნება $O(d^c \cdot (n/c))$, რომელიც წრფივადაა დამო-კიდებული n-ზე. განსხვავებით $O(d^n)$, თუ არ დავშლით ქვეამოცანებად. ეს რა თქმა უნდა მნიშვნელოვნად ამცირებს ამოცანის ამოხსნის დროს.



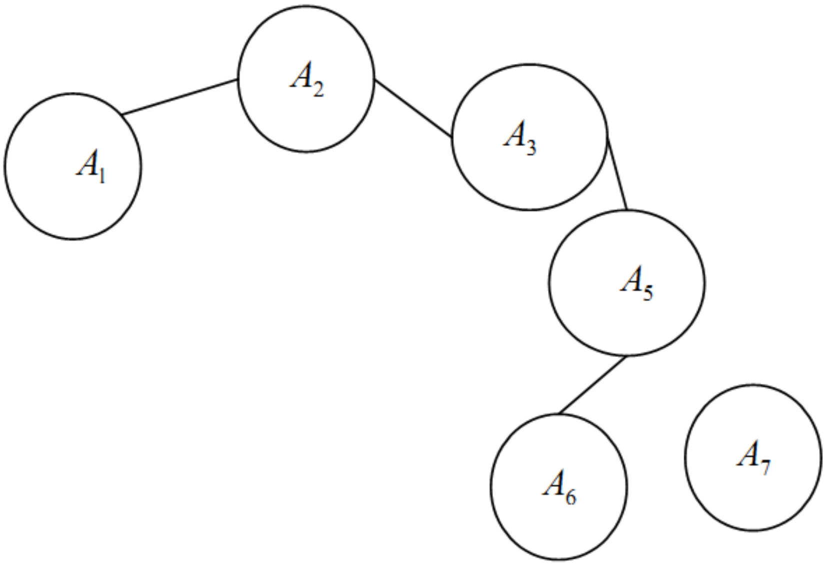
სურ. # 6.9. ა) ხისებური სტრუქტურა; ბ) ცვლადების წრფივი მოწესრიგება ფესვით A

ასეთ შემთხვევაში, არსებობს ალგორითმი, რომლის სირთულეც წრფივადაა დამოკი-დებული ცვლადების რაოდენობაზე. ეს ალგორითმი შედგება შემდეგი ეტაპებისაგან:

1. ამოირჩიე ხის ფესვად ნებისმიერი ცვლადი და მოაწესრიგე ცვლადები ფესვიდან ფოთლებამდე ისე, რომ თითოეული კვანძის მშობელი კვანძი ხეზე წინ უსწრებდეს ამ კვანძს (იხ. სურ. #6.9 ბ). აღნიშნე ცვლადები ამ მიმდევრობით, როგორც X_1 , X_2 ,...X_n, მაშინ ყველა ცვლადს გარდა ფესვისა, ეყოლება მხოლოდ ერთი მშობელი ცვლადი.

2. ციკლში j იცვლება n-დან 2-მდე გამოიყენე რკალების თავსებადობის შემოწმება (X_i, X_j) რკალებისათვის, სადაც X_i არის მშობელი კვანძი X_j კვანძისათვის და თუ საჭიროა ამოყარე მნიშვნელობები X_i-ს განსაზღვრის არიდან.
3. ციკლში j იცვლება 1-დან n-მდე მიანიჭე X_j ნებისმიერი მნიშვნელობა, რომელიც თავსებადია X_i-ს მნიშვნელობასთან, სადაც X_i არის მშობელი კვანძი.

ახლა ვნახოთ, შესაძლებელია თუ არა მივიყვანოთ რთული გრაფები ხისებურ სტრუქ-ტურამდე. არსებობს ორი გზა ამის მისაღწევად: პირველი მოითხოვს კვანძების მოცი-ლებას, მეორე – კი კვანძების გარეთიანებას. დავუბრუნდეთ ჩვენს მაგალითს. თუ ჩვენ შევძლებთ A_4 ცვლადის მოშორებას. მაშინ დანარჩენი ცვლადები მოგვცემენ გრაფს (ის. სურ. # 6.10.).



სურ. # 6.10. გრაფის ხედ გადაქცევის პირველი გზა კვანძის მოცილებით

ამ კვანძის მოშორება შესაძლებელია, თუ ჩვენ A_4 ცვლადს დავუფიქსირებთ რაიმე მნიშვნელობას და დანარჩენი ცვლადების (გარდა A_7) განსაზღვრის არეებიდან ამოვი-ღებთ ამ მნიშვნელობას. უფრო რთულ შემთხვევებში შესაძლებელია ერთი ცვლადის ამო-ღება არ იყოს საკმარისი ხის მისაღებად, ამიტომ ზოგადად ალგორითმს ექნება შემდეგი სახე:

1. ამოვირჩიოთ ისეთი ქვესიმრავლე S ცვლადების სიმრავლიდან, რომ შეზღუდვათა დარჩენილი გრაფი წარმოდგენდეს ხეს (S-ს ეწოდება ციკლის გაწყვეტის სიმრავლე).
2. S-ში ცვლადების ყველა შესაძლო მინიჭებებისათვის, რომლებიც აკმაყოფილებენ შეზღუდვებს S-ში, შეასრულე შემდეგი:
3. დარჩენილი ცვლადების მნიშვნელობათა სიმრავლიდან ამოიღე ყველა მნიშვნე-ლობები, რომლების არათავსებადია S-ში მინიჭებულ მნიშვნელობებთან;
4. თუ დარჩენილ ამოცანას აქვს ამოხსნა, გამოიტანე ეს ამოხსნა S-ში შესრულებულ მინიჭებებთან ერთად.

თუ ციკლის გაწყვეტის სიმძლავრეა $|c|$, მაშინ ალგორითმის სირთულე იქნება

$$O(d^{|c|}(n-|c|)d^2).$$

იმ შემთხვევაში, როდესაც გრაფი თავისი ფორმით "ძალიან ახლოსაა ხესთან, $|c|$ იქნება პატარა, ხოლო დროის ეკონომია ძებნის უკან დაბრუნების ალგორითმთან შედარებით, ძალიან დიდი. ყველაზე ცუდ შემთხვევაში $|c|$ შეიძლება იყოს $(n-2)$-ის ტოლი. ციკლის გაწყვეტის ამოცანა ზოგადად წარმოადგენს **NP** სირთულის ამოცანას, მაგრამ ცნობილია რამდენიმე ეფექტური ალგორითმი ამ ამოცანის ამოსახსნელად.

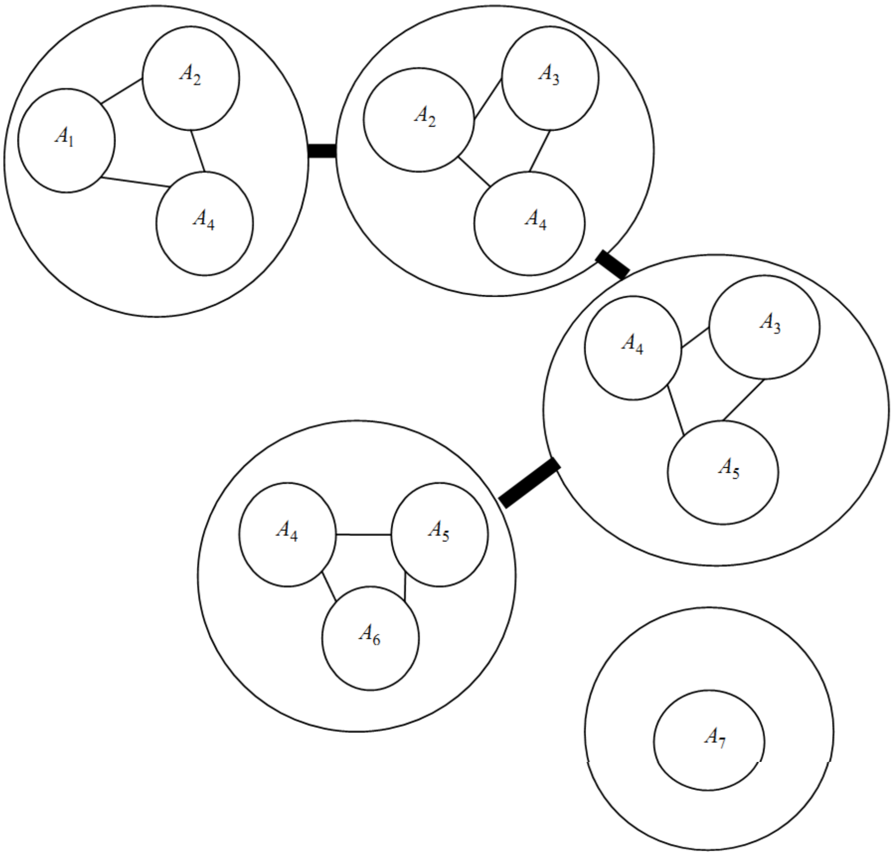
მეორე გზა, რომელიც შეიძლება გამოვიყენოთ ხისებური სტრუქტურის მისაღებად ეფუძნება შეზღუდვათა გრაფის ხისებური დეკომპოზიციისა და დაკავშირებულ ქვეამოცანათა სიმრავლის შექმნას. თითოეული ქვეამოცანა გადაიჭრება დამოუკიდებლად და შემდეგ ხდება მიღებული შედეგების კომბინაციების შემოწმება. თუ ქვეამოცანები არ იქნება ძალიან დიდი, მაშინ ალგორითმი მუშაობს საკმარისად წარმატებულად. განვიხილოთ ეს მიდგომა კვლავ ჩვენი მაგალითის შემთხვევაში (იხ. სურ. #6.11.). აქ მოხდა ხისებური სტრუქტურის მიღება სამ-სამი ცვლადის გაერთიანებით დაკავშირებულ ქვეამოცანათა სიმრავლეში. თუ დეკომპოზიციის შემდეგ რომელიმე ქვეამოცანას არა აქვს ამოხსნა, მაშინ ცხადია, არც მთლიან ამოცანას არ ექნება ამოხსნა. თუ შევძელით ქვეამოცანათა ამოხსნა, მაშინ შეგვიძლია ვცადოთ ძირითადი ამოცანის ამოხსნა შემდეგი გზით. თითოეული ქვეამოცანა განიხილება როგორც "მეგაცვლადი", რომლის განსაზღვრის არესაც წარმოადგენს ამ ქვეამოცანის ამონახსნები. მაგალითად, პირველი ქვეამოცანაა სამი რეგიონის გაფერადება სამი ფერით, ისე, რომ მეზობელი რეგიონები არ იყოს გაფერადებული ერთი და იგივე ფერით, ამიტომ ამ ქვეამოცანას აქვს ექვსი ამოხსნა. შესაბამისად, პირველი მეგაცვლადის მნიშვნელობების სიმრავლე შეიცავს ამ ექვს ამოხსნას. ამის შემდეგ ქვეამოცანების შეზღუდვების დასადგენად გამოიყენება ზემოთმოყვანილი ხეების ალგორითმი. შეზღუდვები, რომლებიც აკავშირებენ ქვეამოცანებს, მიუთითებენ, რომ ქვეამოცანათა ამოხსნები უნდა შეთანხმებული იყოს მათი საერთო ცვლადების მიმართ. მაგალითად, თუ ვიწყებთ პირველი ქვეამოცანის ამონახსნით $A_1 = 1, A_2 = 2, A_4 = 3$, ერთადერთი თავსებადი ამონახსნი მეორე ქვეამოცანისათვის შეიძლება იყოს $A_2 = 2, A_3 = 1, A_4 = 3$.

შეზღუდვათა ნებისმიერი გრაფისათვის შესაძლებელია დიდი რაოდენობის ხისებური დეკომპოზიცია. დეკომპოზიციის ამორჩევის დროს უნდა ვეცადოთ, რომ ქვეამოცანათა რაოდენობა იყოს ცოტა. ხისებური დეკომპოზიციის ხის სიგანე ერთით ნაკლები უნდა იყოს ყველაზე დიდი ქვეამოცანის ზომაზე. თვით გრაფის ხის სიგანე განისაზღვრება როგორც მინიმალური ხის სიგანე ყველა მის ხისებურ დეკომპოზიციებს შორის. თუ გრაფის ხის სიგანე ტოლია w და მოცემულია შესაბამისი ხისებური დეკომპოზიცია, მაშინ ამოცანის სირთულე იქნება $O(nd^{w+1})$, რაც იმას ნიშნავს, რომ ამოცანები შეზღუდვათა დაკმაყოფილებაზე, რომელთა შეზღუდვათა გრაფი ხასითდება ხის სასრული სიგანით, შეიძლება ამოიხსნას პოლინომიალურ დროში. სამწუხაროდ, მინიმალური ხის სიგანის მქონე დეკომპოზიციის მოძებნის ამოცანა წარმოადგენს NP სირთულის ამოცანას, მაგრამ არსებობენ ევრისტიკული მეთოდები, რომლებიც წარმატებით გამოიყენება პრაქტიკაში.

საკონტროლო კითხვები:

1. რა განსხვავებაა ამოცანის წარმოდგენაში მდგომარეობათა სივრცესა და შეზღუდვების დაკმაყოფილებაზე?
2. ჩამოთვალეთ რა უპირატესობები აქვს შდა წარმოდგენას?
3. როგორი შეიძლება იყოს ცვლადების მნიშვნელობები და შეზღუდვები?
4. როგორ ამოცანებს არ განვიხილავთ განვიხილავთ ამოცანებში შეზღუდვაზე?
5. მოიყვანეთ ერთი და იგივე ამოცანის წარმოდგენის მაგალითი მდგომარეობათა სივრცეში და შდა წარმოდგენაში.
6. აღწერეთ ალგორითმი ძებნა უკან დაბრუნებით
7. აღწერეთ ევრისტიკები, რომლებიც გამოიყენება შდა ალგორითმებში.
8. რით განსხვავდება ეს ევრისტიკები მდგომარეობათა სივრცის ევრისტიკებისაგან?
9. რას ნიშნავს ამოცანის დეკომპოზიცია?

10. როგორ ქვეამოცანებს ეწოდება დამოუკიდებელი ქვეამოცანები?
11. რას ნიშნავს გრაფის ხისებური დეკომპოზიცა/
12. როგორი ალგორითმი შეიძლება გამოვიყენოთ ამოცანის ამოსახნელად, თუ შეზღუდვების გრაფის ნაცვლად გვექნება ხე.
13. რას ნიშნავს ამოცანის დეკომპოზიცია?
14. როგორ ქვეამოცანებს ეწოდებათ დამოუკიდებელი ქვეამოცანები?
15. რა მეთოდები არსებობს რთული გრაფის ხისებურ სტრუქტურაზე მისაყვანად?



სურ. # 6.11. გრაფის ხისებური დეკომპოზიცია

ლექცია 7. ძებნა წინააღმდეგობის შემთხვევაში

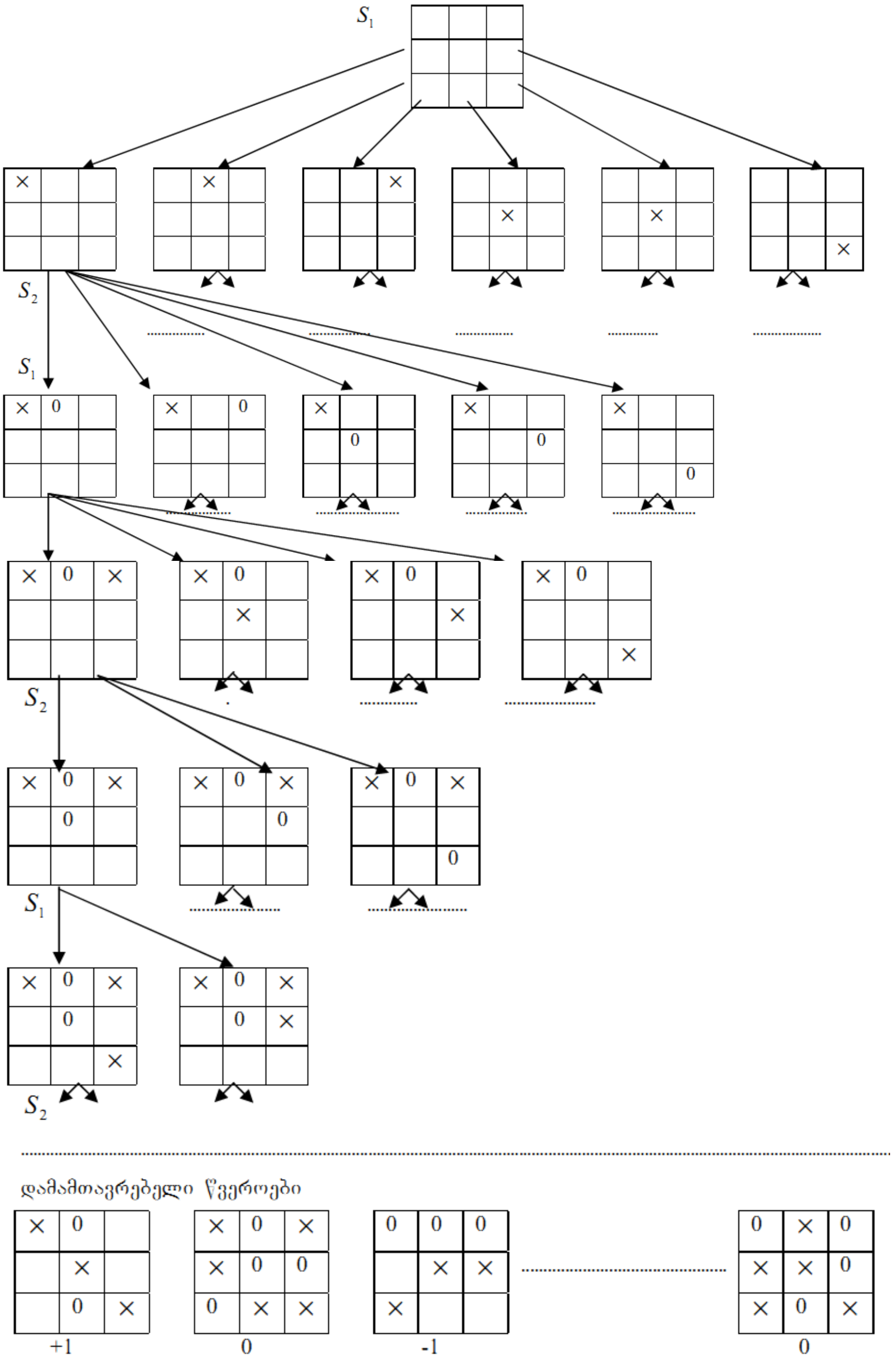
7.1 თამაშები, როგორც კონკურენტული პრობლემური გარემო. ჩვენ აგენტთა თეორიაში განვიხილეთ მულტიაგენტური გარემო, რომელშიც აგენტს უხდებოდა მხედველობაში მიეღო სხვა აგენტების ქმედებები და დაედგინა თუ რა გავლენა შეეძლო მოეხდინა ამ ქმედებებს მის საკუთარ გადაწყვეტილებებზე და ამის მიხედით მოეხდინა საკუთარი ქმედებების კორექცია. მულტიაგენტური გარემო თავისი შინაარსით შეიძლება იყოს როგორც კოოპერაციული, ასევე კონკურენტული. პირველ შემთხვევაში აგენტები ურთიერთთანამშრომლობენ დასახული მიზნის მისაღწევად, მეორე შემთხვევაში კი მათი მიზნები იმდენად განსხვავებულია, რომ ერთი აგენტის მიერ მიზნის მიღწევა ავტომატურად იწვევს იმ-

ას, რომ მეორე აგენტი ან ვერ მიაღწევს მიზნას, ან მიაღწევს მხოლოდ ნაწილობრივ. ხელოვნურ ინტელექტში ძებნის ამოცანებს ისეთ კონკურენტულ მულტიაგენტურ გარემოში, სადაც აგენტთა მიზნები ურთიერთსაწინააღმდეგოა, უწოდებენ ძებნას წინააღმდეგობის პირობებში და რადგანც ასეთი გარემოს ყველაზე ხშირი მაგალითებია ინტელექტუალური თამაშები – ზოგჯერ თამაშებს.

თამაშები თავისი სტუქტურით განსხვავდებიან ერთმანეთისაგან. მაგალითად ჭადრაკის, ან შაშის თამაში არსებითად განსხვავდება პრეფერანსის თამაშისაგან და აქ საქმე მარტო იმაში კი არ არის, რომ ერთ შემთხვევაში საქმე გვაქვს ჭადრაკის ფიგურებთან, ხოლო მეორე შემთხვევაში კარტებთან. განსხვავება უფრო ღრმაა. ჭადრაკის თამაშის დროს გვყავს ორი მოწინააღმდეგ, რომელთა მიზნები დიამეტრალურად განსხვავდება ერთმანეთისგან, ანუ არის ანტაგონისტური. ერთი მოთამაშის მოგება ავტომატურად იწვევს მეორის წაგებას, პრეფერანსის თამაშის დროს კი შესაძლებელია რამდენიმე მოთამაშე გაერთიანდეს დანარჩენების წინააღმდეგ, ანუ შექმნან კორპორაცია და გაუჩნდეთ საერთო მიზანი. გარდა ამისა, ჭადრაკის თამაში მიმდინარეობს დაფაზე, და დროის ნებისმიერ მომენტში მოთამაშეებს გააჩნიათ სრული ინფორმაცია თამაშის მიმდინარეობის შესახებ, მაშინ, როდესაც პრეფერანსის თამაშის დროს მოთამაშეს არ გააჩნია სრული ინფორმაცია თამაშის შესახებ, რადგანაც მან ზუსტად არ იცის რა კარტები უჭირავთ დანარჩენ მოთამაშეებს.

7.2. თამაშები სრული ინფორმაციით. თამაშთა მათემატიკურ თეორიაში პირველი ტიპის თამაშებს უწოდებენ ორი მოწინააღმდეგის თამაშს ნულოვანი ჯამით და სრული ინფორმაციით. პრობლემური გარემოს ენაზე ეს შეესაბამება დეტერმინირებულ, მთლიანად დაკვირვებად, მიმდევრობით გარემოს, რომელშიც მოქმედებს ურთიერთსაწინააღმდეგო მიზნების მქონე ორი აგენტი. ცხადია, ასეთ შემთხვევაში ძებნის **ზემოთ განხილული ალგორითმები ვერ მოგვცემენ სასურველ შედეგს, ვინაიდან ყველა ეს ალგორითმი ყოველ ნაბიჯზე აუმჯობესებს თავის პარამეტრებს და ისე ეძებს ამონახსნს,** რაც ამ შემთხვევაში შეუძლებელია. საერთოდ თამაშები იმითაც განსხვავდება სხვა ამოცანებისაგან, რომ ძალიან ძნელია მათი ამოხსნა. ასე მაგალითად ჭადრაკის თამაშში განტოტვის კოეფიციენტი საშუალოდ უდრის 35, ხოლო თამაში ხშირად გრძელდა 50 სვლამდე თითოეული მოთამაშის მხრიდან, ამიტომ ძებნის ხე შეიცავს დაახლოებით 10^{154} წვეროს (თუმცა თვით ძებნის გრაფი შეიცავს "მხოლოდ" 10^{40} განსხვავებულ წვეროს. შედარებისთვის ჩვენს გალაქტიკაში მოლეკულების რაოდენობა არ აღემატება 10^{85}-ს). თამაშთა თეორიაში ადვილად მტკიცდება, რომ ასეთი თამაშებისთვის არსებობს ერთადერთი სტრატეგია, რომელიც საუკეთესოა მოთამაშისათვის და როგორც არ უნდა ითამაშოს მოწინააღმდეგემ ეს სტრატეგია ყოველთვის მიგვიყვანს მოცემულ თამაშში ობიექტურად არსებულ საუკეთესო შედეგამდე. იმისათვის, რომ ვნახოთ, თუ როგორ უნდა მოვძებნოთ ასეთი ოპტიმალური სტრატეგია, განვიხილოთ ისეთი მარტივი თამაში,როგორიცაა ე.წ. "კრესტიკი – ნოლიკი". ამ თამაშში გვაქვს დაფა ცხრა უჯრით (სამი სამზე), რომელშიც მოწინააღმდეგეები რიგრიგობით სვავენ "ჯვარს" და "ნოლს". თამაში მთავრდება მაშინ, როდესაც ერთერთი მოწინააღმდეგე ჰორიზონტალურად, ვერტიკალურად ან დიაგონალზე დასვავს სამ "ჯვარს" ან "ნოლს", ან როდესაც შეივსება ყველა უჯრა. ყველა შესაძლო სტრატეგიების რაოდენობას ამ თამაშში ტოლია $9! = 381024$, მაგრამ თუ გავითვალისწინებთ თამაშის სიმეტრიულობას დიაგონალის მიმართ, მაშინ სტრატეგიების რაოდენობა შეგვიძლია შევამციროთ 720-მდე. სურ. #7.1. ნაჩვენებია ამ თამაშის ხე. S_1-ით აღნიშნულია ის პოზიციები, რომელშიც სვლას აკეთებს პირველი მოთამაშე, ხოლო S_2-ით კი პოზიციები, რომელშიც სვლას აკეთებს მეორე მოთამაშე. დამამთავრებელი პოზიციები შეფასებულია პირველი მოთამაშის სასარგებლოდ. თამაშთა თეორიაში ოპტიმალური ამონახსნის მოძებნა შეიძლება დავიწყოთ მხოლოდ მას შემდეგ, რაც აიგება ასეთი სრული ხე.

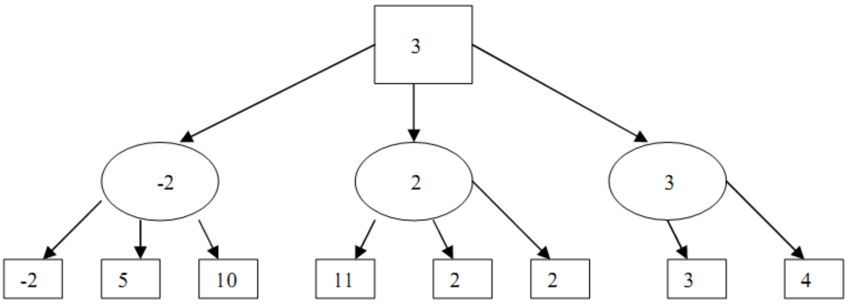
7.3. მინიმაქსისის პროცედურა. ახლა ვნახოთ, როგორ შეიძლება მოვძებნოთ ის ერთადერთი სტრატეგია, რომლის არსებობასაც ამტკიცებს თამაშთა თეორია? მას შემდეგ, რაც ავაგებთ თამაშის სრულ ხეს, ყველა დამამთავრებელი წვერო შეგვიძლია შევაფასოთ თამაშის წესებიდან გამომდინარე.



სურ. #7.1 თამაშის სრული ხე.

ამის შემდეგ ვიწყებთ საუკეთესო სვლების მიმდევრობის, ანუ თამაშის სტრატეგიის ამორჩევას. ამისათვის ვიქცევით შემდეგნაირად: დავუშვათ დამამთავრებელ წვეროებში გადა-

ვყევართ მოწინააღმდეგის სვლებს, ცხადია, რომ მოწინააღმდეგე ეცდება მიაღწიოს წარმატებას, რაც ნიშნავს ჩვენს დამარცხებას, ამიტომ ყველა შესაძლო პოზიციებიდან, რომლებშიც გადავყავართ მოწინააღმდეგეს, ჩვენ უნდა ამოვარჩიოთ პოზიციები მინიმალური შეფასებებით და ეს შეფასებები მივაწეროთ პოზიციებს, რომლებიც დგანან ერთი საფეხურით ზევით. ამ პოზიციებში უკვე გადავდივართ ჩვენი სვლების შედეგად, ამიტომ ყველა შესაძლო პოზიციებს შორის უნდა ამოვარჩიოთ პოზიციები მაქსიმალური შეფასებებით და ეს შეფასებები გადმოვიტანოთ ერთი საფეხურით ზევით. ასე უნდა გავაგრძელოთ, სანამ არ მივაღწევთ საწყის პოზიციას. ამ პროცედურას ეწოდება მინიმაქსის პროცედურა და საშუალებას გვაძლევს ვიპოვოთ ის ერთადერთი სტრატეგია, რომელიც მიუხედავათ იმისა, თუ რას ითამაშებს მოწინააღმდეგე, ყოველთვის მიგვიყვანს სასურველ შედეგამდე. განვიხილოთ მაგალითი. დავუშვათ, მოცემული გვაქვს თამაშის ხე (იხ. სურ. #7.2), რომლის დამათავრებელ პოზიციებში თამაშის წესების თანახმად გვაქვს თამაშის შედეგები, ანუ დამამთავრებელი პოზიციების შეფასებები. შეფასებები აღებულია S_1 მოთამაშისათვის და საუკეთესო სტრატეგიის ამორჩევაც სრულდება ამ მოთამაშისთვის.



სურ. #7.2. თამაშის ხე დამამთავრებელი წვეროებით, რომლისთვისაც ვიყენებთ მინიმაქსის პროცედურას

ამ სურათზე ოვალებით აღნიშნულია პოზიციები, რომლიდანაც სვლას აკეთებს მოწინააღმდეგე. ასეთი პოზიციები სამია და შესაძლო სვლების რაოდენობა კი - რვა. პირველი პოზიციიდან მოწინააღმდეგეს შეუძლია გააკეთოს სამი სვლა, რის შედეგადაც ის ან წააგებს ხუთ ქულას, ან წააგებს ათ ქულას და ან მოიგებს ორ ქულას. S_1 მოთამაშე ჩათვლის, რომ მოწინააღმდეგე აუცილებლად გააკეთებს სვლას, რომელიც მას მოაგებინებს ორ ქულას და ამ პოზიციას შეაფასებს მინუს ორი ქულით. პრაქტიკულად, პირველმა მოთამაშემ გამოთვალა იმ სამი დამამთავრებელი პოზიციების შეფასებათა მინიმუმი, რომლებშიც შეიძლება გადაიყვანოს მოწინააღმდეგემ პირველი ოვალური პოზიციიდან და ეს შეფასება მიაწერა ამ პოზიციას. ანალოგიურად მოიქცევა ის დანარჩენი პოზიციების მიმართაც და მიიღებს შეფასებებს (-2,2,3). ამ პოზიციებში გადასვლა ხდება უკვე პირველი მოთამაშის სვლის შედეგად. ამიტომ ის ცხადია, ამოირჩევს სვლას, რომელიც მოუტანს მაქსიმალურ მოგებას. მივიღეთ, რომ ასეთ თამაშში როგორც არ უნდა ითამაშოს მოწინააღმდეგემ, პირველ მოთამაშეს გარანტირებული აქვს მინიმუმ სამი ქულის მოგება (თუ მოწინააღმდეგე დაუშვებს შეცდომას, ეს მოგება შეიძლება გაიზარდოს, მაგრამ არავითარ შემთხვევაში არ შემცირდება). ოპტიმალური სტრატეგიის მოძებნის ამ ალგორითმს ეწოდება მინი-მაქსის პროცედურა.

7.4. შემოკლებული მინიმაქსის პროცედურა. აშკარაა, რომ თამაშის ოპტიმალური სტრატეგიის მოძებნის ასეთი ალგორითმი შეუძლებელია გამოვიყენოთ მეტ-ნაკლებად საინტერესო თამაშებისათვის, რადგანაც პრაქტიკულად შეუძლებელია ავაგოთ ასეთი თამაშების სრული ხე. თუ ჩვენ უარს ვიტყვით გარანტირებულად ოპტიმალური სტრატეგიის მოძებნაზე, იგივე პროცედურა შესაძლებელია გამოვიყენოთ მაშინაც, როდესაც ავაგებთ თამაშის

არა სრულ ხეს, არამედ მის ფრაგმენტს, დავუშვათ საწყისი პოზიციიდან რაიმე სიღრმემდე. ასე იქცევა ალბათ ადამიანიც, როდესაც ის თამაშობს მაგალითად ჭადრაკს, შაშს ან რაიმე სხვა თამაშს.

ასეთ შემთხვევაში წარმოიშვება ორი პრობლემა. პირველი, რა დონეზე უნდა შევწყვიტოთ ახალი პოზიციების აგება, ანუ რა დონეზე უნდა გადავჭრათ ხე. რა თქმა უნდა ამ შემთხვევაში უნდა გათვალისწინებული იქნას კომპიუტერის სიმძლავრეები და თამაშის ხის დატოტვის კოეფიციენტი, მაგრამ მარტო ეს არ არის საკმარისი. წარმოვიდგინოთ, რომ ჭადრაკის თამაშისათვის შემოვიღეთ რაიმე დონე, რომელზედაც ვჭრით ხეს. ამ დონეზე ერთ ერთ პოზიციაში იწყება ლაზიერების გაცვლის კომბინაცია, მოწინააღმდეგემ უკვე აიყვანა ჩვენი ლაზიერი და შემდეგ სვლაზე იგივეს გავაკეთებთ ჩვენც. მაგრამ ეს შემდეგი პოზიცია უკვე აღარ შედის მინიმაქსის ალგორითმის იმ პოზიციათა სიმრავლეში, რომელშიც ის ეძებს საუკეთესო სვლას. თავისთავად ცხადია, რომ სიტუაცია დაწუნებული იქნება, მაშინაც კი, როდესაც ამ გზას მივყავართ საბოლოო გამარჯვებისაკენ. ასეთი შემთხვევების თავიდან ასაცილებლად შემოიღებულია ე.წ. მკვდარი პოზიციის ცნება. პოზიციას ეწოდება მკვდარი პოზიცია, თუ მისი შეფასება რადიკალურად არ განსხვავდება მისი შვილობილი პოზიციების შეფასებებისაგან. მკვდარი პოზიციის ცნება საშუალებას გვაძლევს გადავჭრათ ხე სხვადასხვა დონეზე (ანუ ხის ტოტები იყოს სხვადასხვა სიღრმის) და თავიდან ავიცილოთ შეცდომები ოპტიმალური გზის ძებნის დროს.

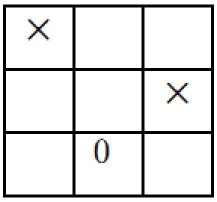
მეორე პრობლემა ეხება პოზიციების შეფასებას. დამამთავრებელ პოზიციებში შეფასება ადვილი გასაკეთებელია, რადგანაც ის გამომდინარეობს თამაშის წესებიდან მაგრამ დაუმთავრებელი თამაშის დროს შეფასებების გასაკეთებლად კვლავ უნდა გამოვიყენოთ ევრისტიკული შემფასებელი ფუნქცია, რომელიც აიგება ჩვენი ან სპეციალისტების მოსაზრებებიდან გამომდინარე და რომელიც ნებისმიერ პოზიციას შეუსაბამებს რაიმე რიცხვს. როგორც წესი ასეთ შემფასებელ ფუნქციას აქვს პოლინომის სახე:

$$f(n) = \sum_{i=1}^{m} a_i x$$

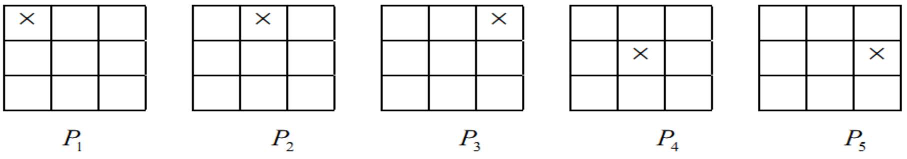
სადაც n არის პოზიცია, x_i -ური არის პარამეტრი, რომლითაც ხდება ამ პოზიციის შეფასება, ხოლო a_i - კი ამ პარამეტრის ხვედრითი წონა შემფასებელ ფუნქციაში. m არის პარამეტრების რაოდენობა, რომლებიც მონაწილეობენ პოზიციის შეფასებაში. უნდა აღინიშნოს, რომ ასეთი ფუნქციის შედგენა არც ისე მარტივია, რადგანაც პარამეტრები, რომლებითაც ფასდება მდგომარეობა, როგორც წესი არიან ხარისხოვანი სახის და მათთვის რიცხვითი მნიშვნელობების შეფარდება საკმაოდ რთული პროცესია. ასევე რთულია წონითი კოეფიციენტების შერჩევა, ხშირად ეს დაკავშირებულია ხანგრძლივ მანქანურ ექსპერიმენტებთან. მას შემდეგ, რაც ჩვენ გვექნება შემფასებელი ფუნქცია, უკვე შესაძლებელი იქნება მინიმაქსის პროცედურის გამოყენება გადაჭრილ, ანუ ქვეხეზე. განვიხილოთ ასეთი ფუნქციის შედგენისა და გამოყენების მაგალითი ზემოთმოყვანილი “კრესტიკი-ნოლიკის” თამაშისათვის. დავუშვათ, S_1 იყოს მოთამაშე, რომელიც სვავს ”×“-ებს, ხოლო S_2 მოთამაშე, რომელიც სვავს “0”-ებს. ჩვენ ფუნქციას ვადგენთ S_1 მოთამაშისათვის.შემოვიტანოთ $f(n)$ შემფასებელი ფუნქცია იმ პოზიციებისათვის, რომლებიც არ არიან დამამთავრებელი პოზიციები შემდეგი სახით:

$$f(n) = f_1(n) - f_2(n)$$

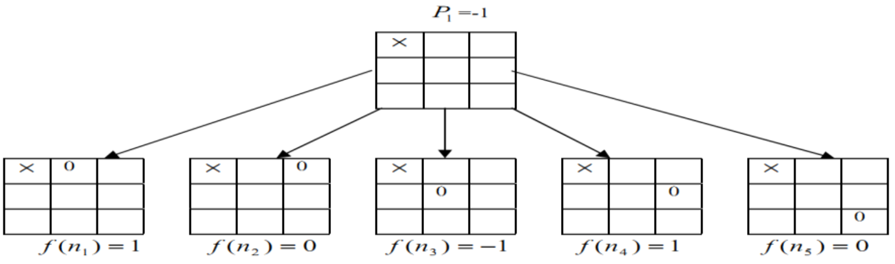
სადაც $f_1(n)$ არის იმ სრული სტრიქონების, სვეტების და დიაგონალების რაოდენობა, რომლებიც ღიაა S_1 მოთამაშისათვის, ხოლო $f_2(n)$ კი – იმ სრული სტრიქონების, სვეტებისა და დიაგონალების რაოდენობა, რომლებიც ღიაა S_2 მოთამაშისთვის. თუ n არის დამამთავრებელი პოზიცია, რომელშიც იგებს S_2 მოთამაშე, მაშინ $f(n) = -\infty$ და თუ n არის დამამთავრებელი პოზიცია, რომელშიც იგებს S_1, მაშინ $f(n) = \infty$. განვიხილოთ მაგალითი. დავუშვათ, მოცემული გვაქვს პოზიცია

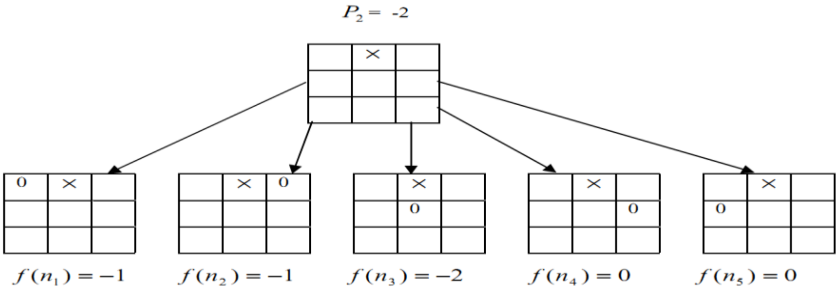


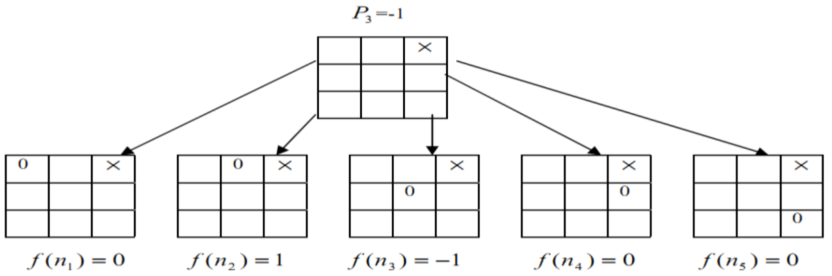
მაშინ S_1 მოთამაშისათვის ღიაა პირველი და მეორე სტრიქონები, პირველი და მესამე სვეტები და ორივე დიაგონალი, S_2 მოთამაშისათვის ღიაა მესამე სტრიქონი, მეორე სვეტი და ერთი დიაგონალი, ამიტომ $f(n)=6-3=3$. განვიხილოთ, როგორ შეიძლება შემოკლებული მინიმაქსის ალგორითმის გამოყენება ამ თამაშისათვის. დავუშვათ თამაშს იწყებს S_1 მოთამაშე და განიხილავს თამაშს სამის ტოლ სიღრმეზე. თუ გავითვალისწინებთ სიმეტრიას, რომელიც ახასიათებს ამ თამაშს, მაშინ არსებობს ხუთი განსხვავებული სვლა, რომელიც S_1 მოთამაშეს შეუძლია გააკეთოს საწყის პოზიციაში. შესაბამისად მიიღება ხუთი განსხვავებული პოზიცია P_1, P_2, P_3, P_4, P_5.

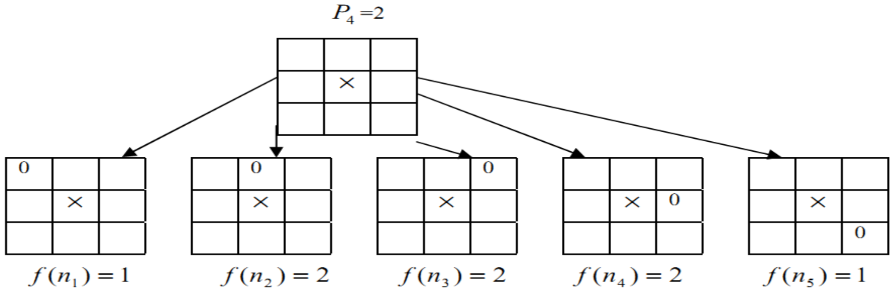


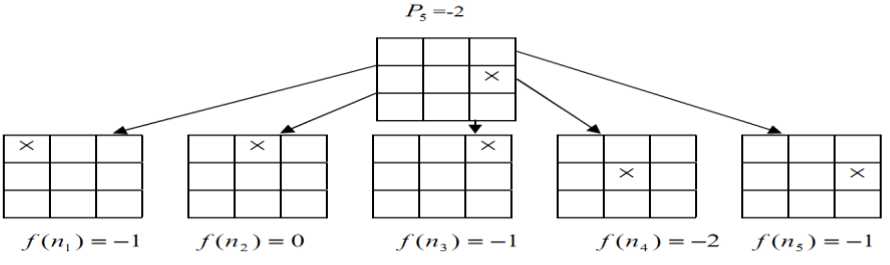
განვიხილოთ რა საპასუხო სვლების გაკეთება შეუძლია მოწინააღმდეგეს და შევაფასოთ მიღებული პოზიციები:



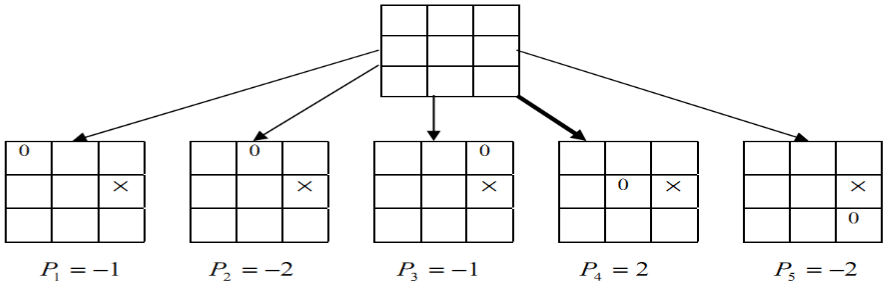








ეს გამოთვლები გვიჩვენებს, რომ თუ S_1 მოთამაშე გააკეთებს P_1 სვლას, მისი მოგება ტოლი იქნება -1-ის. ანალოგიურად უნდა გამოვთვალოთ ამ მოთამაშის მოგებები დანარჩენი ოთხი სვლის შემთხვევაშიც.

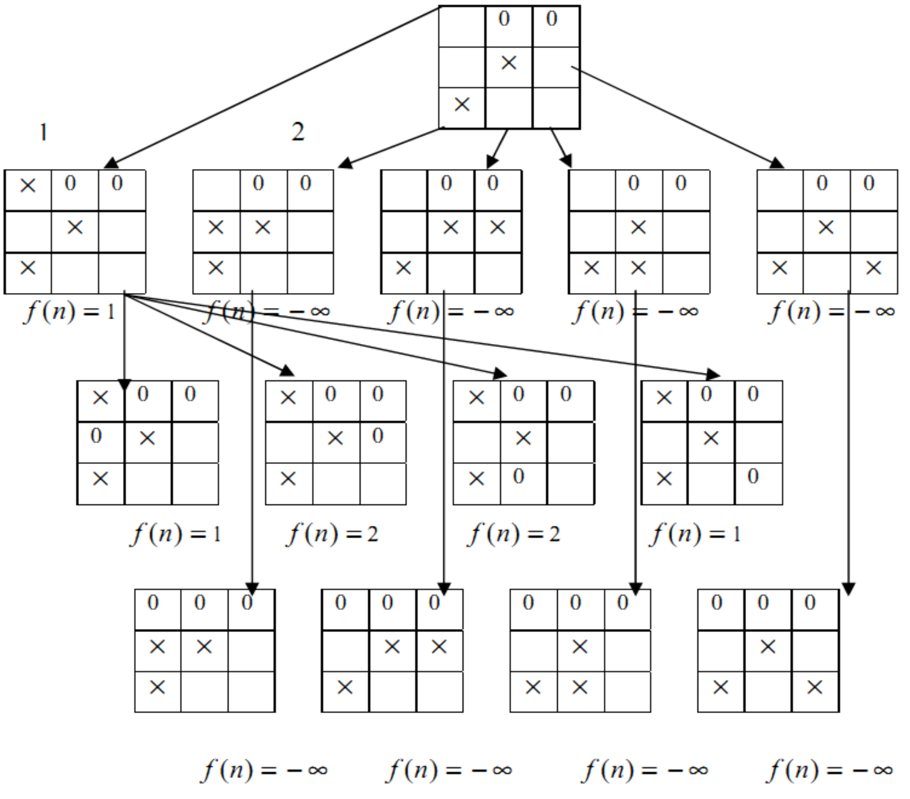


. #7.3. შემოკლებული მინიმაქსის პროცედურა “კრესტიკი – ნოლიკის” თამაშისათვის.

ამის შემდეგ შემოკლებული მინი-მაქსის პროცედურით შეიძლება მოიძებნოს მოცემული კონკრეტული შემფასებელი ფუნქციის შესაბამისი ოპტიმალური სტრატეგია (იხ. სურ.#7.3). შევნიშნოთ, რომ შემოკლებული მინიმაქსის ალგორითმი, განსხვავებით მინიმაქსის პროცედურისაგან, უკვე აღარ იქნება ოპტიმალური და შეიძლება ზოგიერთ შემთხვე-

ვაში სრულიც. დროითი და სივრცული პარამეტრები როგორც მინიმაქსის პროცედურის, ასევე შემოკლებული მინიმაქსის პროცედურისა, შეფასდება $O(b^m)$, სადაც b არის დატოტვის კოეფიციენტი, ხოლო m - კი ხის სიღრმე.

7.5. ალფა – ბეტა პროცედურა. აქამდე განხილულ ძებნის ყველა ალგორითმში, დაწყებული განივი გადარჩევის ალგორითმით და დამთავრებული ამოცანებით შეზღუდვათა დაკმაყოფილებაზე, ამოცანის ამოხსნის გრაფი (ან ხე) ყოველთვის მოიცემოდა არაცხადი სახით და ევრისტიკული პროგრამირების ძირითადი პრინციპის თანახმად, გრაფის (ხის) აგების და ძებნის პროცესები იყო გაერთიანებული, რაც წარმოადგენს ევრისტიკული ძებნის მნიშვნელოვან უპირატესობას გრაფზე ძებნის მეთოდებთან შედარებით. როგორც ვხედავთ, შემოკლებული მინიმაქსის პროცედურაში ეს პრინციპი დარღვეულია.

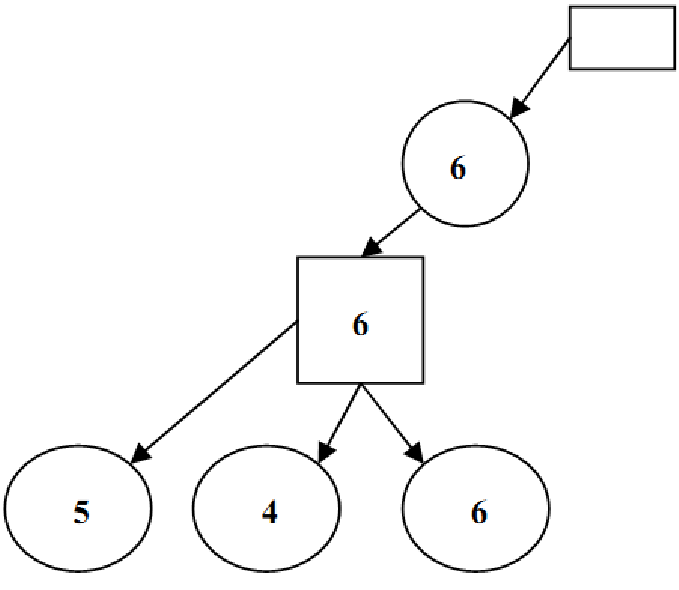


სურ. #7.4. ალფა-ბეტა პროცედურის გამოყენების მაგალითი

ჩვენ ეხლა განვიხილავთ პროცედურას, რომელშიც ეს პრინციპი კვლავ აღდგენილია და რომელიც საშუალებას გვაძლევს გავხსნათ უფრო ნაკლები წვეროები, ვიდრე ეს ხდება შემოკლებული მინი-მაქსის ალგორითმში და ერთი და იგივე შემფასებელი ფუნქციის შემთხვევაში, მოვძებნოთ ზუსტად იგივე ამოხსნა, რომელსაც გვაპოვნინებდა მინიმაქსის ალგორითი. ამ პროცედურას, რომელი ძალიან წააგავს სიღრმეში გადარჩევის ალგორითმს ეწოდება ალფა – ბეტა პროცედურა. ალგორითმის მუშაობის პრინციპის გასაგებად განვიხილოთ მაგალითი.

დავუშვათ, გვაქვს პოზიცია (იხ. სურ. #7.4.), რომელშიც სვლა უნდა გააკეთოს S_1 მოთამაშემ და ის იწყებს შესაბამისი ხის აგებას, რომლის სიღრმე ორის ტოლია, ანუ განიხილავს მხოლოდ თავის სვლას და მოწინააღმდეგის პასუხს. მოცემულ სიტუაციაში მას შეუძლია ხუთი სვლის გაკეთება და თითოეულზე მოწინააღმდეგეს შეუძლია უპასუ-

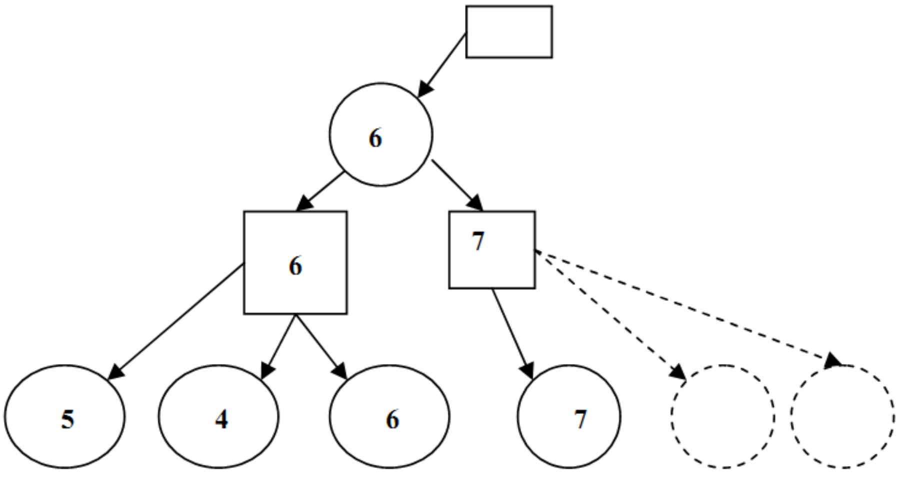
ხოს ოთხი სვლით, რაც ნიშნავს, რომ S_1 მოთამაშემ უნდა ააგოს ხე, რომელიც შეიცავს ოცდაერთ წვეროს, თუ უნდა, რომ გამოიყენოს შემოკლებული მინიმაქსის ალგორითმი, რაც აშკარაა, რომ სრულიად ზედმეტია ამ პოზიციაში, რადგანაც ერთი სვლის გარდა ყველა დანარჩენი ავტომატურად იწვევს S_1 მოთამაშის დამარცხებას. მაგრამ ამას ალგორითმი ვერ ხედავს. დავუშვათ, ეხლა გვაქვს ისეთი ალგორითმი, რომელიც ააგებს რა მოცემულ სიღრმეზე დამამთავრებელ წვეროს, მაშინვე აფასებს მას და ეს შეფასება გადმოქვს ზევით და ადარებს ამ დონეზე მიღებულ სხვა შეფასებებს. ასეთ შემთხვევაში ჩვენს მიერ განხილულ მაგალითში ალგორითმი ააგებს პირველ წვეროს ააგებს მის შვილობილ წვეროებს, შეაფასებს მათ და მინიმალურ შეფასებას გადმოიტანს წვეროში (1). ამის შემდეგ ააგებს მეორე (2) წვეროს, ააგებს მის შვილობილ წვეროს, მიიღებს შეფასებას -∞ , გადმოიტანს მას, როგორც (2) წვეროს შეფასებას, აღმოაჩენს, რომ ეს შეფასება უფრო ნაკლებია, ვიდრე (1) წვეროს შეფასება და აღარ განიხილავს (2) წვეროს დანარჩენ წვეროებს, რადგანაც ცხადია, რომ მოწინააღმდეგე არ გააკეთებს სხვა სვლას. ასევე მოიქცევა ეს ალგორითმი დანარჩენი წვეროების მიმართაც, რაც მას საშუალებას აძლევს შეამციროს გადარჩევის ხე ოცდაერთიდან თოთხმეტამდე, ეს მაშინ, როდესაც სიღრმე იყო მხოლოდ ორის ტოლი. თუ გადარჩევის სიღრმე იქნება უფრო მეტი, ალგორითმი შეძლებს უფრო ადრე აღმოაჩინოს, რომ არაა საჭირო ზოგიერთი წვეროს გახსნა და ამით საგრძნობლად შეამციროს გადარჩევის ხე.



სურ. #7.5. ალფა-ბეტა პროცედურის პირველი ნაწილი

არ უნდა ვიფიქროთ, რომ ეს ხდება მხოლოდ მაშინ, როდესაც ალგორითმი შეხვდება თამაშის დამამთავრებელ წვეროს. თუ ალგორითმი ჩავიდა მოცემულ სიღრმეზე და შეაფასა წვერო, ის ამ შეფასებას ამოიტანს ზედა დონეზე და კვლავ გვექნება იგივე პროცედურა, რომლის არსიც მდგომარეობს შემდეგში: თუ გვაქვს რაიმე წვეროს შეფასება და ალგორითმმა აღმოაჩინა, რომ მისი მეზობელი წვეროს რომელიმე შვილობილის შეფასება ამ წვეროს შეფასებაზე ნაკლებია, აზრი აღარ აქვს დანარჩენი შვილობილი წვეროების აგებას, რადგანც წვეროს შეფასება ვეღარ გახდება ამ შეფასებაზე მეტი.

განვიხილოთ მეორე მაგალითი (იხ.სურ. #7.5). ამ სურათზე კვადრატებით აღნიშნულია პოზიციები, რომლებშიც სვლას აკეთებს S_1 მოთამაშე. იმისათვის, რომ ალფა-ბეტა პროცედურამ მოგვცეს შედეგი, ხის სიღრმე არ უნდა იყოს სამზე ნაკლები. S_1 მოთამაშემ საწყისი პოზიციიდან ააგო პირველი სვლა, შემდეგ ააგო მოწინააღმდეგის პირველი და შემ-

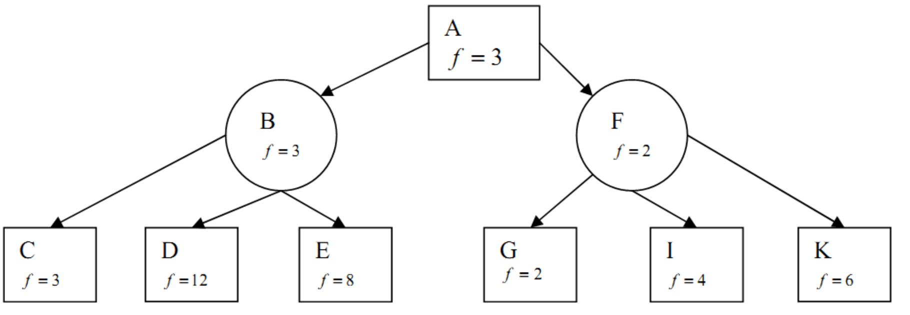


სურ. #7.6. ალფა-ბეტა პროცედურის მიერ მოკვეთილი წვეროები

დეგ ყველა შესაძლო საკუთარი პასუხი. აქ დამთავრდა ხის სიღრმე, ამიტომ S_1 მოთამაშემ შეაფასა მიღებული წვეროები და ეს შეფასება ამოიტანა ორი დონით ზევით (რადგანაც ბოლო პოზიციებში გადადის საკუთარი სვლით, ამოარჩია მაქსიმალური შეფასება). ეს მომენტი ანალოგიურია მინი-მაქსის პროცედურის. განსხვავება იწყება შემდეგი ბიჯიდან (იხ. სურ. #7.6). ამის შემდეგ S_1 მოთამაშე განიხილავს მეორე სვლას საწყისი პოზიციიდან. ჩაყვება მას ბოლომდე და განსხვავებით პირველი ნაწილისგან, როგორც კი გახსნის ბოლო წვეროს, მის შეფასებას მაშინვე ამოიტანს ორი პოზიციით ზევით და შეადარებს შეფასებას, რომელიც ჰქონდა პირველ ნაწილში. ჩვენს მაგალითში ეს შეფასება მეტია პირველი წვეროს შეფასებაზე, ამიტომ ის აღარ განიხილავს დანარჩენ ბოლო დონის წვეროებს (ნახაზზე აღნიშნულია პუნქტირით). ამგვარად, ალფა -ბეტა პროცედურამ შეამცირა გადარჩევა. ასეთ მოკვეთას ეწოდება ბეტა მოკვეთა. თუ იმ პოზიციებში, რომლებშიც განხორციელდა მოკვეთა გადასვლა მოხდებოდა არა მოწინააღმდეგის, არამედ S_1 მოთამაშის სვლით, მაშინ ცხადია, S_1 მოთამაშე ამოარჩევდა არა მინიმს, არამედ მაქსიმუმს. ასეთ მოკვეთას უწოდებენ ალფა მოკვეთას. პროცედურამაც აქედან მიიღო სახელი.

ალფა – ბეტა პროცედურაში არსებითი მნიშვნელობა აქვს წვეროების გახსნის მიმდევრობას. მართლაც, განვიხილოთ მაგალითი (იხ. სურ. #7.7.). თუ ვიყენებთ ალფა – ბეტა პროცედურას, მაშინ, ალგორითმი ააგებს **B** წვეროს, შემდეგ მის შვილობილ წვეროებს **C,D,E** და ამოარჩევს მათ შორის მინიმალურ შეფასებას, რომელსაც გადაიტანს როგორც **B,** ასევე **A** წვეროებში. ამის შემდეგ ის ააგებს **F** წვეროს, მის შვილობილ **G** წვეროს, მიიღებს მის შეფასებას, გადმოიტანს **F** წვეროში, შეადარებს მას **B** წვეროს შეფასებას და რადგანაც ამ დონეზე ხდება მაქსიმუმის მოძებნა, აღარ გახსნის **I** და **K** წვეროებს. მაგრამ თუ **F** წვეროს შვილობილი წვეროები განლაგდებოდნენ შებრუნებული მიმდევრობით, მაშინ ალგორითმს მოუწევდა ყველა შვილობილი წვეროს გახსნა და მისი ეფექტიანობა იქნებოდა ისეთივე, როგორიც შემოკლებული მინიმაქსის პროცედურის. აქედან გამომდინარე, შეგვიძლია დავასკვნათ, რომ უკეთესია ალგორითმმა პირველ რიგში გამოიკვლიოს ის წვეროები, რომლებიც ალბათ შეიძლება გახდნენ საუკეთესოები. ცხადია, რომ შეუძლებელია წვეროების აბსოლუტურად მოწესრიგება, მაგრამ რაღაც ალბათობით ეს მაინც შესაძლებელია. მაგალითად, ჭადრაკის შემთხვევაში შეიძლება გამოვიყენოთ მარტივი ფუნქცია, ჯერ განვიხილოთ სვლები, რომლებიც დაკავშირებულია ფიგურების აყვანასთან, შემდეგ მუქარებთან, სვლები წინ და ასე შემდეგ. ანუ განსახილველი წვეროების მოწესრიგება მთლიანად დამოკიდებულია შემფასებელ ფუნქციაზე. ასეთ შემთხვევაში ალგორითმის სივრცი-

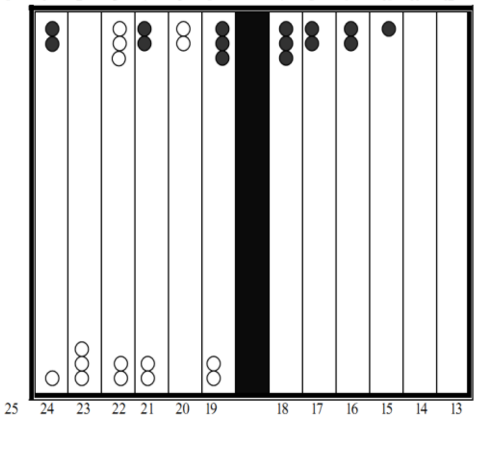
თი (და შესაბამისად დროითი) სირთულე გახდება $O(b^{m/2})$, რაც მნიშვნელოვნად ამცირებს გადარჩევას.



სურ. #7.7. ალფა – ბეტა პროცედურა. მართკუთხედებით აღნიშნულია წვეროები, რომლებშიც გადავდივართ მოწინააღმდეგის სვლით.

როგორც ვიცით, თუ ამოცანის ამოხსნის პროცესში ვღებულობთ გრაფს და არა ხეს, ეს გამოიწვევს სივრცითი და დროითი სირთულეების ექსპონენციალურ ზრდას. თამაშებში სვლების სხვადასხვა მიმდევრობებს ხშირად მივყავართ ერთსა და იმავე პოზიციაში. ამ მოვლენას უწოდებენ ტრანსპოზიციას. ვინაიდან შეუძლებელია თავიდან ავიცილოთ ეს მომენტი, კარგი იქნება, თუ სპეციალურ ცხრილში შევინახავთ პოზიციის შეფასებას. ასეთ ცხრილს უწოდებენ ტრანსპოზიციის ცხრილს. ეს ცხრილი ძალიან წააგავს სიას "დახურულია", რომელიც გამოიყენებოდა ძებნის ალგორითმებში და მას შეუძლია თითქმის ორჯერ გაზარდოს ალფა – ბეტა პროცედურის სიღრმე ისეთ თამაშში, როგორიცაა ჭადრაკი.

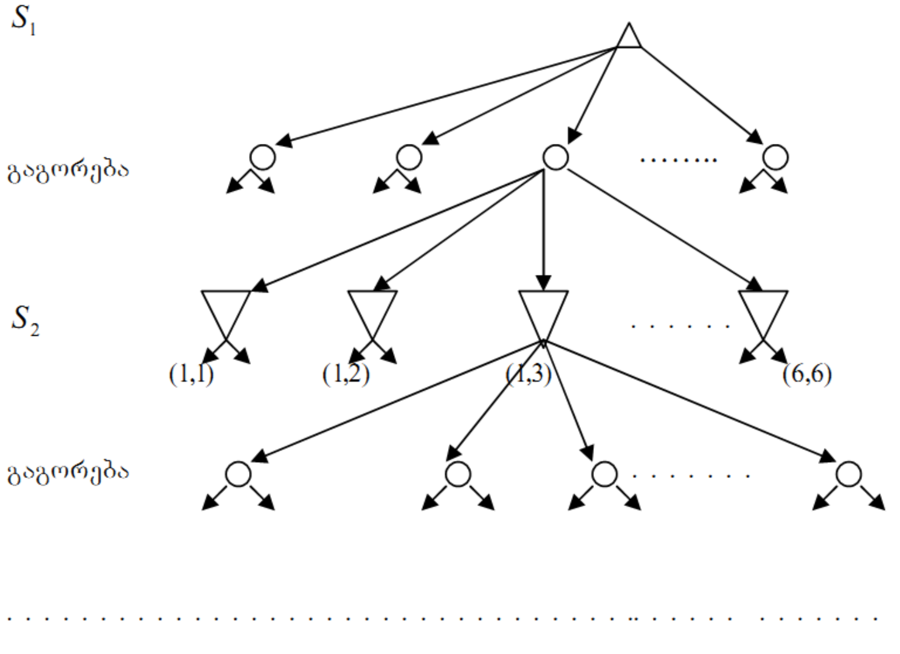
7.6. თამაშები არასრული ინფორმაციით. რეალურ ცხოვრებაში ხშირად ხდება არაპროგნოზირებადი მოვლენები, რის გამოც ადამიანები აღმოჩნდებიან ხოლმე სხვადასხვა მოულოდნელ სიტუაციებში. ეს არაპროგნოზირებადობა ბევრ თამაშში ასახულია მასში შემთხვევითი ელემენტის ჩართვის გზით. ამიტომ ასეთი თამაშების განხილვა სასარგებლოა იმის დასადგენად, თუ გადაწყვეტილების მიღების როგორი მექანიზმები მოქმედებენ ასეთ შემთხვევებში. ნარდის თამაში წარმოადგენს სწორედ ასეთ თამაშს. ყოველი სვლის წინ მოთამაშე აგორებს კამათელს იმის დასადგენად, თუ რა სვლების გაკეთება შეუძლია მას მოცემულ პოზიციაში.



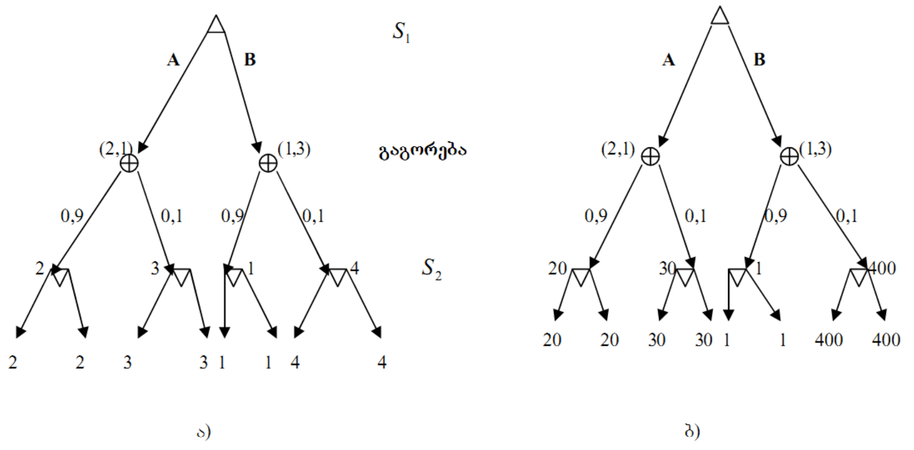
სურ. #7.8. ნარდის თამაშში შექმნილი ერთ ერთი პოზიცია

სურ. #7.8. გამოსახულია ერთერთი შესაძლო პოზიცია. თეთრი ქვები მოძრაობენ საათის ისრის მიმართულებით, ხოლო შავი ქვები საათის ისრის საწინააღმდეგო მიმართულებით. დავუშვათ, S_1 მოთამაშეს, რომელიც თამაშობს თეთრი ქვებით, მოუვიდა შაში ბეში (ექვსი და ხუთი), მაშინ მას შეუძლია გააკეთოს შემდეგი სვლები: (5-10;5-11), (5-10;10-16), (5-11;19-24), (5-11,11-16). მიუხედავად იმისა, რომ S_1 მოთამაშემ იცის თავისი სვლები, მას არ შეუძლია ააგოს თამაშის ხე, რადგანაც მისთვის უცნობია, რას გააგორებს მოწინააღმდეგე და როგორი იქნება მისი დასაშვებ სვლათა სიმრავლე.ამიტომ ნარდის თამაშის ხე გარდა S_1 და S_2 მოთამაშეების წვეროებისა, აგრეთვე უნდა შეიცავდეს კამათლების გაგორების წვეროებსაც. ყოველი გაგორების დროს მოსალოდნელია 36 კომბინაცია, სადაც წყვილის მოსვლის ალბათობა ტოლია $1/36$, ხოლო დანარჩენი კომბინაციების კი $1/18$. თამაშის ხეს ექნება შემდეგი სახე (იხ.სურ.#7.9.).

რა თქმა უნდა ამ შემთხვევაშიც მოთამაშე ცდილობს იპოვოს ისეთი სვლა, რომელიც მიიყვანს მას საუკეთესო შედეგამდე, მაგრამ აქ უკვე წვეროებს აღარ აქვთ განსაზღვრული მნიშვნელობები, აქ შეიძლება ლაპარაკი მხოლოდ შესაძლო, მოსალოდნელ მნიშვნელობებზე, რომელშიც მოსალოდნელი შედეგი გამოითვლება კამათლის გაგორების ყველა შესაძლო შედეგების გათვალისწინებით. ამიტომ, ამ შემთხვევაში ჩვენ უკვე გვაქვს თამაშის მოსალოდნელი მინიმაქსური მნიშვნელობა. ისევე, როგორც მინიმაქსის, ან ალფა–ბეტა პროცედურის გამოყენების დროს, ნარდის თამაშის შემთხვევაშიც ჩვენ უნდა შევწყვიტოთ ხის აგება რაღაც მომენტში და გამოვიყენოთ შემფასებელი ფუნქცია წვეროების შესაფასებლად. ერთი შეხედვით მეტი არაფერი არ იცვლება, გარდა იმისა, რომ აქ ვეძებთ მოსალოდნელ მინიმაქსურ მნიშვნელობას, მაგრამ სინამდვილეში წილისყრის (კამათლის გაგორების) კვანძების არსებობა მნიშვნელოვნად ცვლის საუკეთესო სვლის ამორჩევას. განვიხილოთ მაგალითი. დავუშვათ, მოცემული გვაქვს ერთი და იგივე ქვეხე, მაგრამ შემფასებელი ფუნქცია არის სხვადასხვა (იხ. სურ. #7.8).



სურ. #7.9. ნარდის თამაშის ხის სტრუქტურა



სურ. № 7.8 ა) ალბათ უმჯობესია სტრატეგია **A**

ბ) ალბათ უმჯობესია სტრატეგია **B**

თუ პროგრამაში წინასწარ იქნებოდა ცნობილი, თუ რა შედეგი გვექნება ყოველი წილისყრის შემდეგ, მაშინ ამონახსნის ძებნა იქნებოდა ანალოგიური წილისყრის გარეშე ძებნისა, რომელსაც მინიმაქსის ალგორითმი ასრულებს $O(b^m)$ დროში. მაგრამ, რადგან მოსალოდნელი მინიმაქსური მნიშვნელობის ალგორითმში განიხილება ყველა შესაძლო მიმდევრობები წილისყრის შედეგებისა, მის შესრულებას სჭირდება $O(b^m n^m)$ დრო, სადაც n არის წილისყრის სხვადასხვა ვარიანტების რაოდენობა. ეს გაცილებით ართულებს ძებნას და გვაიძულებს მაქსიმალურად შევზღუდოთ ხის სიღრმე, რაც უარყოფითად მოქმედებს მეტ-ნაკლებად კარგი სტრატეგიის ამორჩევაზე.

საკონტროლო კითხვები:

1. როგორ გარემოში უხდება მოქმედება აგენტს თამაშების დროს
2. როგორ თამაშებს ეწოდებათ თამაშები სრული ინფორმაციით?
3. რატომ ვერ გამოვიყენებთ ძებნის ინფორმირებულ ალგორითმებს თამაშებში?
4. აღწერეთ მინიმაქსის პროცედურა.
5. რატომ არ იძლევა შემოკლებული მინიმაქსის პროცედურა ოპტიმალური ამონახსნის პოვნის გარანტიას?
6. რა არსებითი განსხვავებაა მინიმაქსისა და ალფა-ბეტა პროცედურებს შორის?
7. ჩამოაყალიბეთ რომელ გარემოში უხდება მოქმედება ნარდის მოთამაშე აგენტს.
8. რომელი თამაში უფრო ძნელია ალფა-ბეტა პროცედურისათვის, ჭადრაკის თუ ნარდის? ადამიანისათვის?

ლექცია 8. ცოდნის წარმოდგენის პრობლემა კომპიუტერულ სისტემებში

8.1. ცოდნის წარმოდგენის აუცილებლობა. ჩვენ განვიხილეთ მიზანზე დაფუძნებული აგენტები, რომლებიც პრობლემათა გადასაწყვეტად იყენებდნენ ძებნის ალგორითმებს. როგორც ვნახეთ, ასეთი აგენტები არ იძლევიან საშუალებას გადავჭრათ ყველა პრობლემა, რადგანაც ეს ალგორითმები არის ექსპონენციალური და ამოცანის ზომების ზრდასთან

ერთად მათი ეფექტურობა მცირდებოდა. ჩვენ ეხლა დავიწყებთ ისეთი აგენტების შესწავლას, რომლებიც მოქმედებენ ცოდნის საფუძველზე (შემოკლებით ცოდნაზე დაფუძნებული აგენტი). ცოდნის წარმოდგენა და მსჯელობის პროცესები წარმოადგენენ ხელოვნური ინტელექტის ცენტრალურ საკითხებს. თავისთავად ცხადია, რომ ადამიანებმა ბევრი რამ იციან და გააჩნიათ განსჯის უნარი. ცოდნა და განსჯის უნარი ასევე მნიშვნელოვანია ხელოვნური აგენტებისათვის, რადგანაც უზრუნველყოფს წარმატებული ქცევის ფორმირებას, რომლის მიღწევაც სხვა გზით ალბათ შეუძლებელი იქნებოდა. ასე მაგალითად, თუ ამოცანათა ამომხსნელ აგენტს ეცოდინება თავისი ქმედებების შედეგები, მას შეეძლება წარმატებულად იმოქმედის რთული გარემოს პირობებშიც და არ ეძებოს გზა ბათუმიდან თბილისამდე ბრმად, როგორც ამას აკეთებდა მიზანზე დამყარებული აგენტი. მაგრამ ამოცანათა ამომხსნელი აგენტების ცოდნა არის ძალიან სპეციფიკური და მოუქნელი. ჭადრაკის მოთამაშე პროგრამას შეუძლია გათვალოს მეფის დასაშვები სვლები (იმ ფერის, რომელი ფერითაც ის თამაშობს), მაგრამ მას არ გააჩნია მრავალი სხვა სასარგებლო ცოდნა, მაგალითად, რომ არც ერთ ფიგურას არ შეუძლია ერთდროულად იდგეს ორ უჯრაზე. ცოდნაზე დამყარებულ აგენტებს შეუძლიათ ძალიან ზოგად ფორმაში გამოხატული ცოდნის გამოყენება, ამ ცოდნის კომბინირება და რეკომბინირება სხვადასხვა გარემო პირობების შესაბამისად.

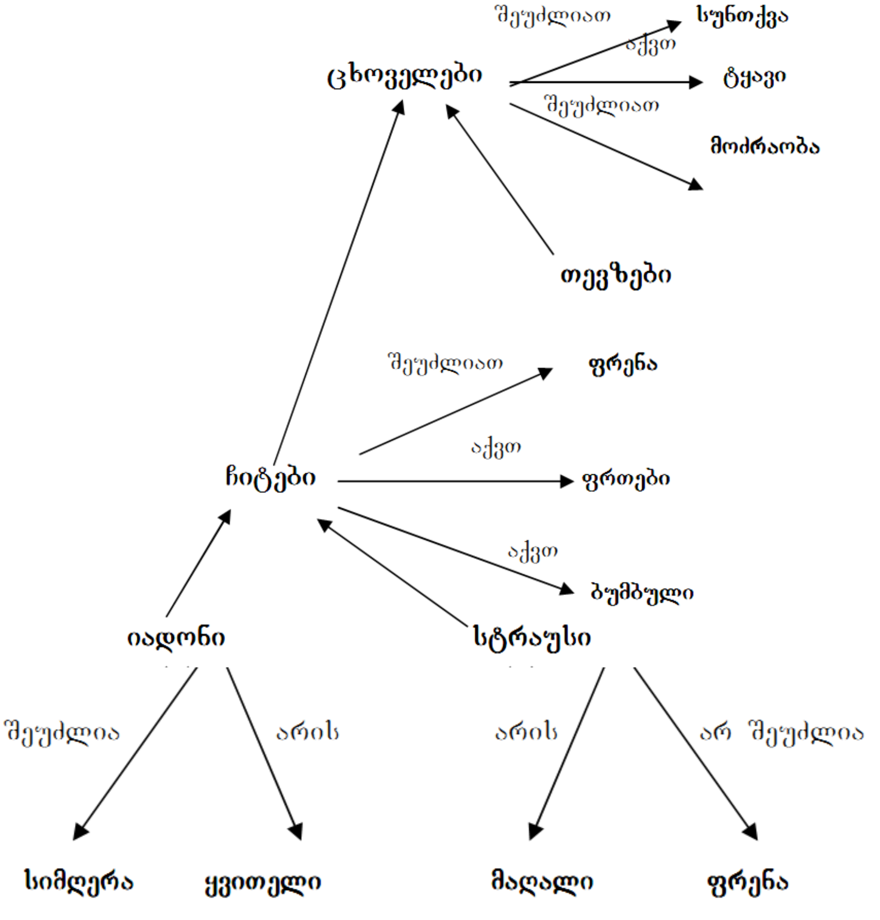
ცოდნა და განსჯა თამაშობენ გადამწყვეტ როლს მაშინაც, როდესაც აგენტს უხდება მოქმედება ნაწილობრივ დაკვირვებად პრობლემურ გარემოში. ცოდნაზე დამყარებულ აგენტს უნარი აქვს შეუთავსოს ერთმანეთს ცოდნა და მოცემულ მომენტში მიღებული აღქმა, რათა გამოავლინოს მიმდინარე მდგომარეობის დაფარული ასპექტები მანამადე, სანამ ის დაიწყებს მოქმედებას. ძალიან მნიშვნელოვანია ისიც, რომ ასეთი აგენტები არიან ძალიან მოქნილები. მათ შეუძლიათ მიიღონ და შეასრულონ ახალი ამოცანები, რომლებიც ჩამოყალიბებულია ცხადად დასმული მიზნის სახით, გარდა ამისა მათ შეუძლიათ შეეგუონ იმ გარემოს ცვლილებას, რომელშიც ისინი მოქმედებენ თავისი ცოდნის განახლების გზით.

ცოდნაზე დამყარებული ნებისმიერი აგენტის ცენტრალურ კომპონენტს ცხადია წარმოადგენს მისი ცოდნის ბაზა.სწორედ ასეთი ბაზის ორგანიზება კომპიუტერულ სისტემაში წარმოადგენს ხელოვნური ინტელექტის ერთერთ ძირითად პრობლემას. ისევე, როგორც პრობლემათა გადაწყვეტის დროს ჩვენ გვაინტერესებდა თუ როგორ წყვეტდა ასეთ პრობლემებს ადამიანი, აქაც რა თქმა უნდა საინტერესოა, თუ როგორაა ორგანიზებული ცოდნა ადამიანში.

8.2. სემანტიკური ქსელები. ფსიქოლიგიიდან ცნობილი ასოციონისტური თეორიის თანახმად, ობიექტის მნიშვნელობა განისაზღვრება სხვა ობიექტებთან ასოციაციის ტერმინებში. ამ თეორიის თანახმად, ობიექტების აღქმა ხდება ცნებების საშუალებით. ცნებები წარმოადგენენ გარემოს შესახებ ჩვენი ცოდნის მნიშვნელოვან ნაწილს და დაკავშირებულები არიან სხვა ცნებებთან ასოციაციების საშუალებით. ეს კავშირები წარმოადგენენ ობიექტის თვისებებს და ქცევას. მაგალითად, ცდების საფუძველზე ჩვენ ვახდენთ ცნება "თოვლის" ასოცირებას სხვა ცნებებთან, დავუშათ ისეთებთან, როგორებიცა: "სიცივე", „თეთრი", "თოვლის ადამიანი", "ყინული" და ასე შემდეგ. ჩვენი ცოდნა თოვლის შესახებ და გამონათქვამის "თოვლი თეთრია" ჭეშმარიტება იქმნება ამ ასოციაციის ქსელიდან. ასევე ფსიქოლოგიიდანაა ცნობილი, რომ ეს ასოციაცები ქმნიან იერარქიულ სტრუქტურებს.

1969 წელს კოლინსმა და კვილიანმა ჩაატარეს ცდები, რომლის არსი მდგომარეობდა იმაში, რომ ისინი ცდის პირებს აძლევდნენ ისეთ შეკითხვებს ფრინველების თვისებების შესახებ, როგორიცაა მაგალითად "იადონი ეს ჩიტია?" "შეუძლია იადონს ფრენა?" "სტრაუსი მაღალია?" და სხვა. ზომავდნენ რა დროს, რომელიც სჭირდებოდათ ცდის პირებს პასუხების მოსაფიქრებლად, მათ აღმოაჩინეს, რომ იმ კითხვებზე პასუხის გასაცემად, რომლებიც ეხება მოცემული ობიექტის კონკრეტულ თვისებებს, ცდის პირებს დასჭირდათ უფრო ცოტა დრო, ვიდრე იმ კითხვებზე, რომლებიც ეხება უფრო ზოგად თვისებებს, დამახასიათებელს არა მარტო ამ კონკრეტული ობიექტისათვის. მაგალითად, კითხვას "იადონი ყვითელია?" სჭირდება გაცილებით ცოტა დრო, ვიდრე კითხვას "შეუძლია იადონს ფრენა?", მაშინ, როდესაც ანალოგიური შეკითხვა "შეუძლია თუ არა ფრენა სტრა-

უსს?" მოითხოვს გაცილებთ ცოტა დროს. ცდის ავტორები ამას ხსნიან იმით, რომ ადამიანები ინფორმაციას იმახსოვრებენ მაქსიმალურად აბსტრაქტულ დონეზე. იმის მაგივრად, რომ დაიმახსოვრონ იადონს შეუძლია თუ არა ფრენა, ისინი იმახსოვრებენ, რომ იადონი არის ჩიტი, ხოლო ჩიტებს (ძირითადად) შეუძლიათ ფრენა. ამ ცდის საფუძველზე მათ შემოგვთავაზეს სემანტიკური ქსელი, რომელიც ორგანიზებულია იერარქიული პრინციპით, და აღწერს ადამიანის თავის ტვინში ცოდნის შენახვის სქემას. როგორც ამ ქსელიდან ჩანს, იმისათვის, რომ ცდის პირმა უპასუხოს შეკითხვას, შეუძლია თუ არა იადონს სუნთქვა, ადამიანმა თავის სემანტიკურ ქსელში უნდა გაიაროს გზა "იადონი" → "ჩიტი" → "ცხოველი" ← "სუნთქავს". სემანტიკური ქსელების ეს პრინციპი ფორმალიზებული იქნა მემკვიდრეობით სტრუქტურებში. ეს სტრუქტურები საშუალებას გვაძლევენ დავიმახსოვროთ ინფორმაცია აბსტრაქციის ყველაზე მაღალ დონეზე, რაც ამცირებს ცოდნის ბაზის ზომებს და საშუალებას გვაძლევს თავიდან ავიცილოთ წინააღმდეგობები. მაგალითად, თუ ჩვენ ვაგებთ ცოდნის ბაზას ჩიტების შესახებ, მაშინ შეგვიძლია განვსაზღვროთ ყველა ჩიტისთვის დამახასიათებელი ისეთი ზოგადი თვისებები, როგორიცა "შეუძლიათ ფრენა", "აქვთ ფრთები" და ა. შ. დავიმახსოვროთ ეს ინფორმაცია ჩიტების აბსტრაქციის დონეზე და მივცეთ შესაძლებლობა ჩიტების



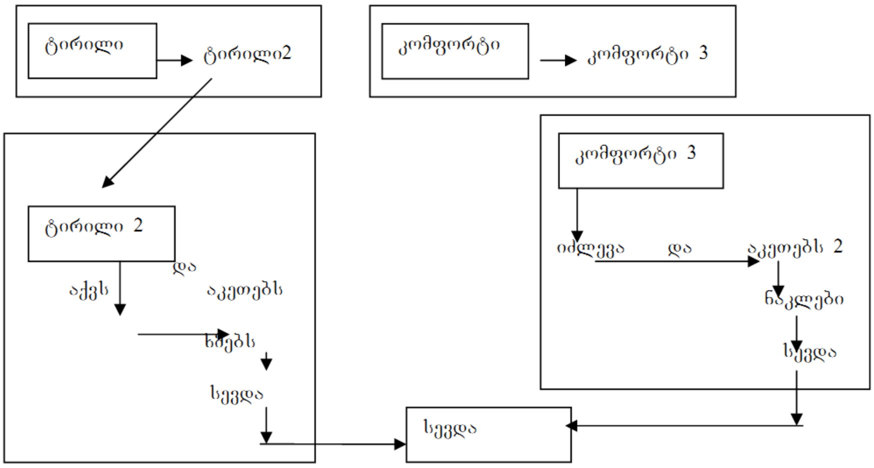
სურ. #8.1. კოლინსისა და კვილინის მიერ შედგენილი სემანტიკური ქსელი

კონკრეტულ წარმომადგენელს მემკვიდრეობით მიიღოს ეს თვისებები. ეს რა თქმა უნდა ამცირებს ცოდნის ბაზის ზომას, იმიტომ რომ ამ თვისებებს განვსაზღვრათ მხოლოდ

ერთხელ და არა ყოველი კონკრეტული წარმომადგენელისათვის ცალ - ცალკე. ეს მოსა-ხერხებელია აგრეთვე ცოდნის ბაზაში ცვლილებების შეტანის დროსაც. თუ ჩვენ ბაზას ვუმატებთ ჩიტების რომელიმე კონკრეტულ წარმომადგენელს ან მთლიანად ქვეკლასს, კლასისათვის აღწერილი თვისებები ავტომატურად გავრცელდება ამ ქვეკლასზეც.

სემანტიკური ქსელების აღწერის ყველაზე მოხერხებული ფორმაა გრაფები რადგა-ნაც შესაძლებელია ცნებებს შორის დამოკიდებულებების აღწერა კვანძებისა და რკალების საშუალებით. კვილიანის მიერ შედგენილ ცოდნის ბაზაში, რომელიც შეიქმნა სამოციანი წლების ბოლოს და რომელიც ორგანიზებული იყო ლექსიკონის პრინციპის შესაბამისად, გრაფის თითოეული კვანძი შეესაბამებოდა სიტყვებით აღწერილ ცნებას, ხოლო რკალების საშუალებით აღიწერებოდა ასოციაციური კავშირები სხვა ცნებებთან. ცოდნის ბაზა იყო ორგანიზებული სიბრტყეებით, სადაც თითოეული სიბრტყე წარმოადგენდა გრაფს, რომე-ლიც ახასიათებდა ერთ ცნებას. ამ ქსელიდან ერთერთი ცნების მაგალითი მოყვანილია სურ. #8.2.

პროგრამა იყენებდა ამ ბაზას მიმართების მოსაძებნად სიტყვათა წყვილს შორის. ამისათვის ის ახორციელებდა ძებნას სიგანეში, სანამ არ იპოვნიდა საერთო ცნებას, ანუ თანაკვეთის კვანძს. გზები საერთო კვანძამდე წარმოადგენდნენ მიმართებებს ცნებებს შო-რის. მაგალითად, სურ. #8.2. ნაჩვენებია თანაკვეთის კვანძი, ორ ცნებას, **ტირილი** და **მოსვე-ნება**-ს შორის. გამოიყენა რა ამ გზების თანაკვეთა, პროგრამამ დაასკვნა, რომ: "ტირილი სხვა დანარჩენთან ერთად დაკავშირებულია სამწუხარო ხმების წარმოქმნასთან. მოსვენე-ბას შეუძლია შეამციროს მწუხარება."



სს

სურ. 8.2. ცნებების **ტირილი** და **კომფორტი** თანაკვეთა.

ავტორი თვლიდა, რომ ასეთი მიდგომა სემანტიკისადმი უზრუნველყოფდა სისტე-მის მიერ ბუნებრივი ენის გაგების შემდეგ შესაძლებლობებს:

1. ინგლისური ტექსტის აზრის გაგება თანაკვეთის კვანძების აგების გზით.
2. სიტყვის მნიშვნელობის ამორჩევა, წინადადების სხვა სიტყვებთან გადაკვეთამდე უმოკლესი გზის ამორჩევის საფუძველზე. მაგალითად ინგლისურ წინადადებაში "**Tom wen**t **home to water his new plant**" ("ტომი მოვიდა სახლში თავისი ახალი ნარგავების მოსარწყავად"), სისტემას შეუძლია ამოირჩიოს სიტყვა "**plant**"-ის მნიშვ-ნელობა ცნებების "**water**" და "**plant**" თანაკვეთაზე დაყრდნობით.

შეკითხვაში მოცემული ცნებების ასოციაციებზე დაყრდნობით, სისტემა შეძლებს მოქნი-ლად უპასუხოს შეკითხვებს.

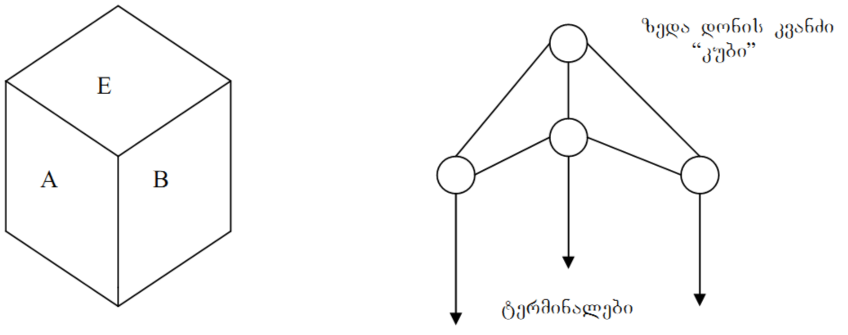
მიუხედავად იმისა, რომ ამ და სხვა მისმა მსგავსმა სისტემებმა ნათლად აჩვენეს გრაფების გამოყენების ეფექტურობა ასოციაციური აზრის მოდელირების დროს, ისინი

შეზღუდულნი იყვნენ მეტისმეტად ზოგადი ფორმალიზმის გამო. ცოდნის სტრუქტური-რება როგორც წესი ხდებოდა სპეციფიკური დამოკიდებულებების ტერმინებში, ისეთში, როგორიცაა კლასი – ქვეკლასი, აგენტი – ზმნა – ობიექტი.

8.3 ფრეიმები. განსხვავებული იდეა ცოდნის კომპიუტერულ სისტემაში წარმოდგენაზე წა-მოაყენა ცნობილმა ამერიკელმა მეცნიერმა მ. მინსკმა. მან შემოგვთავაზა ფრეიმების თეო-რია, რომლის ძირითადი მამოძრავებლი იდეა მდგომარეობს იმაში, რომ როდესაც ადამი-ანი ცდილობს შეიცნოს მისთვის ახალი სიტუაცია, ან ახლებურად შეხედოს მისთვის უკვე ცნობილ საგნებს, ის თავისი მეხსიერებიდან ამოირჩევს მონაცემთა რაღაც სტრუქტურას (სახეს, რომელსაც ავტორი უწოდებს ფრეიმს), იმ მიზნით, რომ მასში გარკვეული დეტა-ლების ცვლილების გზით გახადოს ეს სტრუქტურა გამოსადეგი უფრო ფართო კლასის პროცესებისთვის.

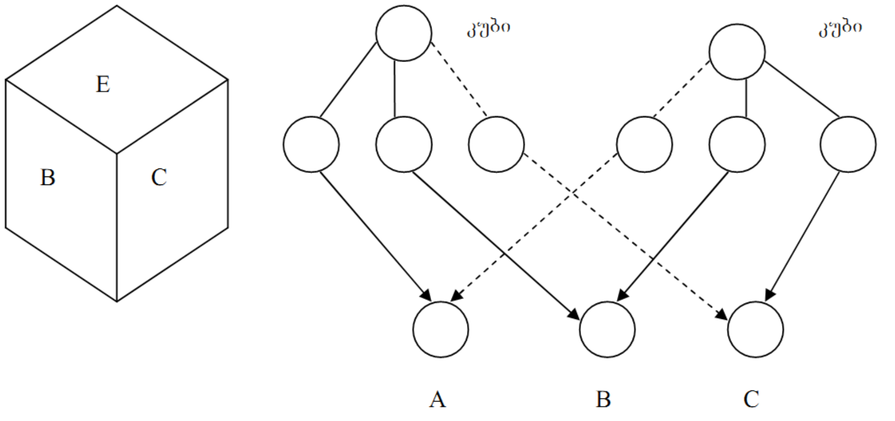
ფრეიმი არის სტერეოტიპული სიტუაციის წარმოსადგენი მონაცემთა სტრუქტურა. თითოეულ ფრეიმთან ასოცირებულია სხვადასხვა სახის ინფორმაცია. მისი ერთი ნაწილი მიუთითებს, თუ როგორ უნდა გამოვიყენოთ ფრეიმი, მეორე – თუ რა შეიძლება მოჰყვეს მის შესრულებას და მესამე – რა უნდა მოვიმოქმედოთ, თუ ეს ჩვენი მოლოდინი არ გამართლდა. ფრეიმი შეიძლება წარმოვიდგინოთ როგორც ქსელი, რომელიც შედგება კვანძებისა და კავშირებისაგან ამ კვანძებს შორის. ფრეიმის "ზედა დონეები" მკაცრად განსაზღვრულია, რადგანაც შეესაბამებიან ცნებებს, რომლებიც ყოველთვის სწორია მოცე-მული შესაძლო სიტუაციებისათვის. შედარებით უფრო დაბალ დონეებზე არის მრავალი განსაკუთრებული წვერო – ტერმინალები ანუ უჯრედები, რომლებიც უნდა შეივსოს დამა-ხასიათებელი მაგალითებით, ან მონაცემებით.

თითოეული ტერმინალის მიერ შეიძლება წამოყენებული იქნას პირობები, რომლე-ბსაც უნდა აკმაყოფილებდნენ მისი დავალებები. მარტივი პირობები, მაგალითად, მოთ-ხოვნა, რომ ტერმინალის დავალება იყოს რომელიმე სუბიექტი, ან გარკვეული ზომის სა-განი, განისაზღვრება მარკერების საშუალებით. უფრო რთული პირობებით მოიცემა მი-მართებები ცნებებს შორის, რომლებიც მოთავსებულნი არიან სხვადასხვა ტერმინალურ წვეროებში. სემანტკურად ახლო ფრეიმების ჯგუფი შეიძლება გავაერთიანოთ ფრეიმთა სისტემაში. განვიხილოთ უმარტივესი მაგალითი. დავუშვათ, ჩვენ გვინდა შევადგინოთ ფრეიმი, რომლის საშუალებითაც შევძლებთ ცოდნის ბაზაში აღვწეროთ კუბი ისე, რომ რა პოზიციიდანაც არ უნდა უყურებდეს ამ კუბს ხელოვნური ინტელექტუალური სისტემა, მან შეძლოს მისი ამოცნობა.



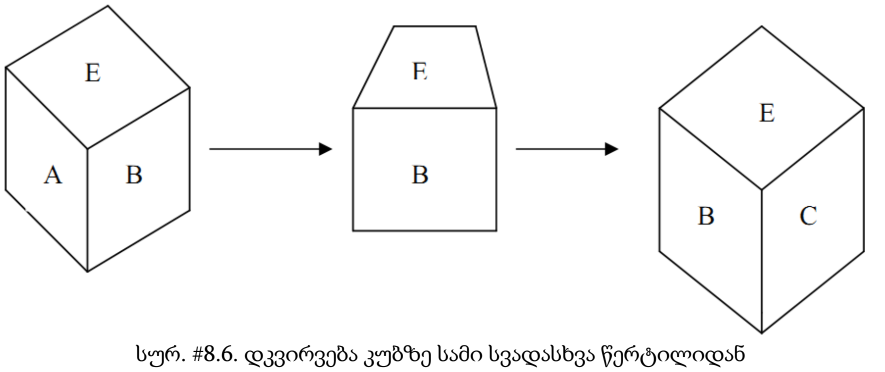
სურ. #8.3. კუბის გამოსახულება და მისის ფრეიმი

სურ. #8.3. მოცემულია კუბის გამოსახულება რაიმე კონკრეტული წერტილიდან და შესაბამისი ფრეიმის სტრუქტურა. დავუშვათ, ეხლა დამკვირვებელი იწყებს მოძრაობას მარჯვნივ, მაშინ **A** წახნაგი გაქრება და მის ადგილზე გამოჩნდება წახნაგი **B**, ხოლო **B** წახნაგის ადგილზე **C** წახნაგი.



სურ. #.8.4

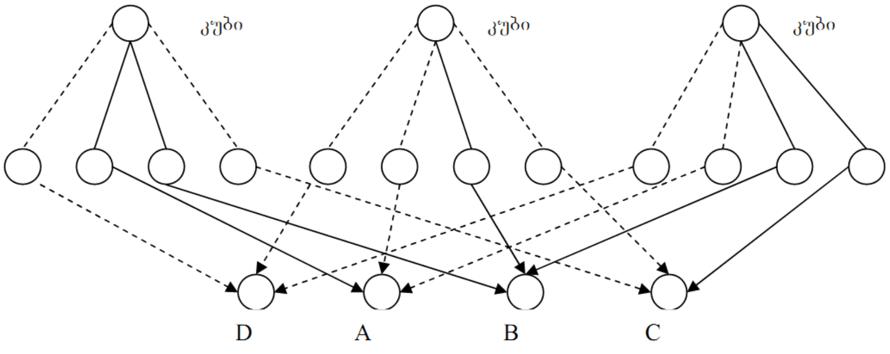
თუ ეხლა შევეცდებით ხელახლა აღვწეროთ კუბი, უნდა დავივიწყოთ ყველაფერი, რაც ვიცოდით და ხელახლა აღვწეროთ წახნაგები **B** და **E** და აღვწეროთ წახნაგი **C** . მაგრამ, თუ ჩვენ ვიცით, რომ საგანი არ შეცვლილა, მაშინ შეგვიძლია შევინახოთ ინფორმაცია **B** და **E** წახნაგების შესახებ აღვწეროთ მხოლოდ **C** წახნაგი. ამასთან, რომ არ დავკარგოთ ინფორმაცია **A** წახნაგის შესახებ, ჩვენ შეგვიძლია დავუკავშიროთ ის უხილავი წახნაგის ტერმინალს. მაშინ ჩვენს ფრეიმს ექნება შემდეგი სახე (იხ. სურ. #8.5.).



სურ. #8.6. დკვირვება კუბზე სამი სვადასხვა წერტილიდან

შესაძლებელია გავაგრძელოთ ასეთი რთული სტრუქტურების აგება, რომელსაც ეწოდება ფრეიმთა სისტემა, როდესაც დაკვირვების წერტილები გვაქვს უფრო მეტი (სურ. #8.6. მოცემულია დაკვირვების სამი წერტილი). შესაბამისი ფრეიმების სტრუქტურა კი ნაჩვენებია სურათზე #8.7.

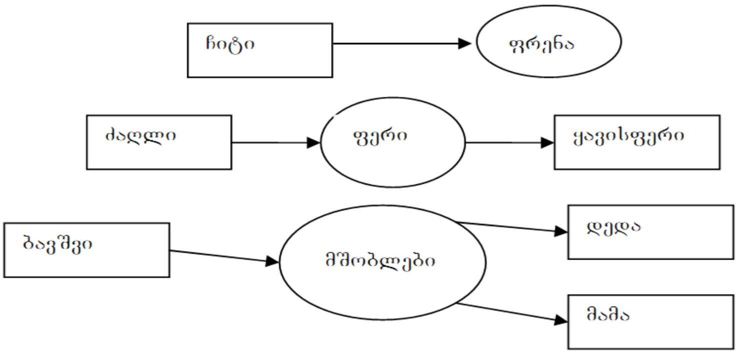
8.4. კონცეპტუალური გრაფები. ცოდნის ბაზების შექმნაში შემდეგი საფეხური იყო ქსელური ენების შექმნა ბუნებრივი ენის სემანტიკის და სხვა გარე სამყაროს მოდელირებისათვის ინტელექტუალურ სისტემებში. ჩვენ განვიხილავთ ქსელური ენების ერთ წარმომადგენელს – კონცეპტუალურ გრაფებს. კონცეპტუალური გრაფი წარმოადგენს სასრულ, შეკრულ, ორმხრივ მიმართულ გრაფს. გრაფის კვანძები წარმოადგენენ ცნებებს ან კონცეპტუალურ მიმართებებს. კონცეპტუალურ გრაფებში რკალების ჭდეები არ გამოიყენება. მიმართებები ცნებებს შორის წარმოიდგინება კონცეპტუალური მიმართებების კვანძებით. იმისათვის, რომ განვასხვაოთ ერთმანეთისგან ცნებები და მიმართებები, სქემაზე პირველს აღნიშნავენ მართკუთხედებით, მეორეს კი – ელიფსებით. სურ. #8.8. მოყვანილი კონცეპტუალური მიმართებები წარმოადგენენ სხვადასხვა n-არობის მიმართებებს.



სურ. #8.7. ფრეიმთა სისტემა დაკვირვების სამი წერტილისათვის. გამარტივების მიზნით სურათზე ნაჩვენები არაა E წახნაგი, რომელიც ჩანს ყველა წერტილიდან.

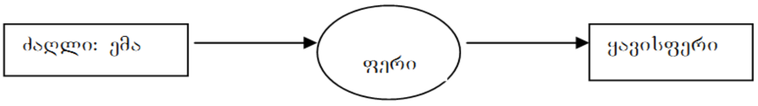
ჩიტი ფრინავს არის უნარული მიმართება, აქ ჩიტი არის ობიექტი, ხოლო ფრინავს – კონცეპტუალური მიმართება. ძაღლის ფერია ყავისფერი არის ბინარული მიმართება, ხოლო ბავშვის მშობლები არიან დედა და მამა კი – ტერანული მიმართება.

კონცეპტუალურ გრაფებში ცნებების კვანძები წარმოადგენენ ან გარე სამყაროში არსებული კონკრეტული ობიექტების ცნებებს ან მსჯელობის დროს გამოყენებულ ისეთ აბსტრაქტულ ცნებებს, რომლებიც გარე სამყაროში არ გვხვდება (მაგალითად პატრიოტიზმი, ღალატი, სიყვარული და ასე შემდეგ).



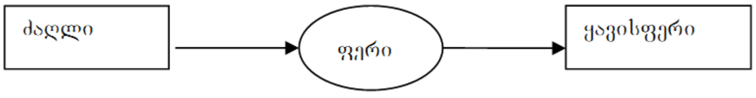
სურ .#8.8. სხვადასხვა *n* -არობის კონცეპტუალური მიმართებები.

სემანტიკური ქსელების შექმნისას, როგორც წესი ყურადღებას არ აქცევდნენ ისეთი მიმართებების განსაზღვრას, როგორებიცაა კლასი – ელემენტი, კლასი – ქვეკლასი, რის გამოც წარმოიშვა სემანტიკური არაერთგვაროვნება. მაგალითად, მიმართება ეგზემპლარსა და კლასს შორის განსხვავდება მიმართებისაგან კლასსა და მის სუპერკლასს შორის. აქედან გამომდინარე ზოგიერთი თვისება ეკუთვნის ეგზემპლარს, ხოლო ზოგიერთი კი კლასს. ცოდნის წარმოდგენას უნდა გააჩნდეს მექანიზმები ამ განსხვავების აღსაწერად. კონცეპტუალურ გრაფებში თითოეული ცნება წარმოადგენს კონკრეტული ტიპის უნიკალურ ეგზემპლარს. თითოეული ცნების მართკუთხედი აღინიშნება ტიპის ჭდით, რომელიც განსაზღვრავს ამ კვანძით წარმოდგენილ კლასს ან ეგზემპლარის ტიპს. ტიპის ჭდე და ეგზემპლარი გამოყოფილია ორი წერტილით.



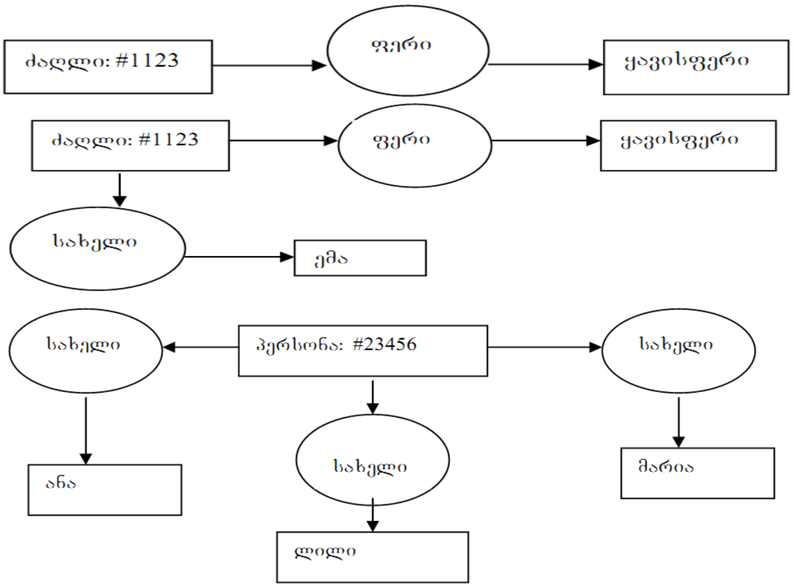
სურ. #8.9. კონცეპტუალური გრაფი, რომელიც გვიჩვენებს, რომ ძაღლი ემა ყავისფერია.

სურ. #8.9. გვიჩვენებს, რომ ძაღლი ემა ყავისფერია. გრაფი, მოცემული სურ. #8.10. გვიჩვენებს, რომ რომელიღაც არასპეციფირებული, ძაღლის ტიპის ობიექტი ყავისფერია. კონცეპტუალური გრაფები გვაძლევენ საშუალებას აღვწეროთ აგრეთვე კონკრეტული, მაგრამ არასპეციფირებული, უსახელო ეგზემპლარები.



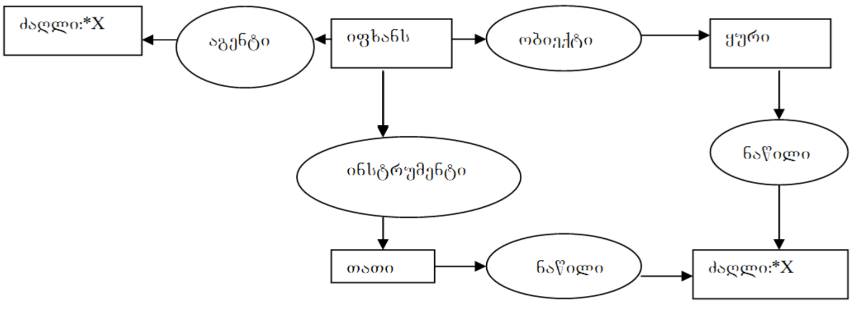
სურ. #8.10. კონცეპტუალური გრაფი, რომელიც გვიჩვენებს, რომ რომელიღაც არასპეციფირებული, ძაღლის ტიპის ობიექტი ყავისფერია.

ამისათვის, თითოეული ეგზემპლარის აღსაწერად მონაცემთა ბაზაში გამოიყენება უნიკალური დესკრიპტორი (აღმწერი), რომელსაც უწოდებენ მარკერს. ის წარმოადგენს რიცხვს, რომლის წინაც განთავსებულია სიმბოლო "#". მარკერი სახელისაგან განსხვავდება იმით, რომ ის არის უნიკალური, ეგზემპლარს შეიძლება საერთოდ არ ჰქონდეს სახელი, შეიძლება ჰქონდეს ერთი, ან რამდენიმე სახელი, მაგრამ მას ყოველთვის აქვს ზუსტად ერთი მარკერი. ეს გვაძლევს საშუალებას თავიდან ავიცილოთ სემანტიკური არაერთმნიშვნელობა, კერძოდ როდესაც გვაქვს ორი ობიექტი ერთი და იგივე სახელით. სურ. #8.11. მოყვანილია მაგალითები, როდესაც ეგზემპლიარს არ გააჩნია სახელი, ან აქვს ერთი, ან რამდენიმე სახელი.



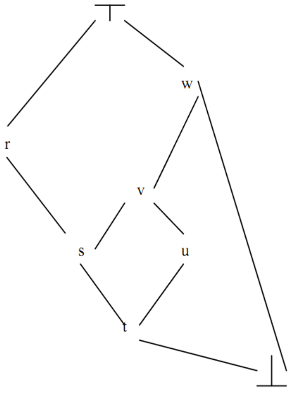
სურ. 8.11. კონცეპტუალური გრაფები, რომლებიც გვიჩვენებენ მარკერის და სახელის გამოყენების სხვადასხვა ვარიანტებს

მაგალითად, წინადადება "ძაღლი იფხანს ყურს თათით" კონცეპტუალური გრაფის საშუა-ლებით აღიწერება ასე (იხ. სურ. #8.12.).



სურ. 8.12. კონცეპტუალური გრაფი წინადადებისათვის "ძაღლი იფხანს ყურს თათით"

ტიპებს გააჩნიათ იერარქიული სტრუქტურა. ტიპი “ძაღლი” არის ტიპი “ხორცის-მჭამელების” ქვეტიპი, რომელიც თავის მხრივ არის “ძუძუმწოვრების” ქვეტიპი და ასე შემდეგ. ტიპების იერარქია - ესაა ნაწილობრივი მოწესრიგება ტიპების სიმრავლეზე, რომ-ელიც შეგვიძლია აღვნიშნოთ სიმბოლოთი ≤ . თუ **S** და **T** არიან ტიპები და **T**≤**S**, მაშინ ამბობენ, რომ **T** არის ქვეტიპი, ხოლო **S** - სუპერტიპი. რადგანაც მოწესრიგება არის ნაწილობრივი, ტიპს შეიძლება გააჩნდეს როგორც რამდენიმე სუპერტიპი, ასევე რამდენიმე ქვეტიპი. თუ **T**<**S** და **T**<**U**, მაშინ ამბობენ, რომ **T** არის **S**-ს და **U**-ს საერთო ქვეტიპი. ანა-ლოგიურად განიმარტება საერთო სუპერტიპიც. კონცეპტუალური გრაფების ტიპების იერ-არქია წარმოადგენს ბადეს, რომელის აღწერს მრავალჯერადი მემკვიდრეობის სისტემის ზოგად სახეს. ამ ბადეში ტიპებს შეიძლება გააჩნდეთ მრავალი მშობელი და ასევე მრავალი შვილობილი წვეროები, მაგრამ ყოველ წყვილს ტიპებისა აუცილებლად გააჩნია მინიმა-ლური საერთო სუპერტიპი და მინიმალური საერთო ქვეტიპი.



სურ. #8.13. ტიპების იერარქია

კონცეპტუალური გრაფების თეორია შეიცავს გარკვეულ ოპერაციებს არსებული გრაფებისაგან ახალი გრაფების მისაღებად, რითაც ის არსებითად განსხვავდება სემანტი-კური ქსელებისაგან. ეს ოპერაციები საშუალებას გვაძლევენ მივიღოთ ახალი გრაფები არ-სებული გრაფების სპეციალიზაციის, ან განზოგადების გზით და ძალიან მნიშვნელოვანია

ბუნებრივი ენის სემანტიკის წარმოსადგენად ცოდნის ბაზებში. ეს ოპერაციებია: კოპირე-ბა, შეზღუდვა, გაერთიანება და გამარტივება.

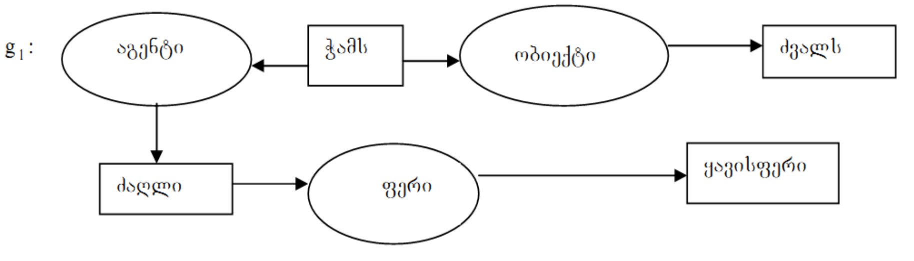
კოპირების ოპერაცია საშუალებას გვაძლევს **g** გრაფისაგან მივიღოთ ახალი $\mathbf{g}_1$ გრაფი, რომელიც წარმოადგენს **g** -ს ზუსტ ასლს.

შეზღუდვის ოპერაციით შეგვიძლია გრაფში ცნების კვანძი შევცვალოთ მისი სპეცი-ალიზაციით. აქ შესაძლებელია ორი შემთხვევა:

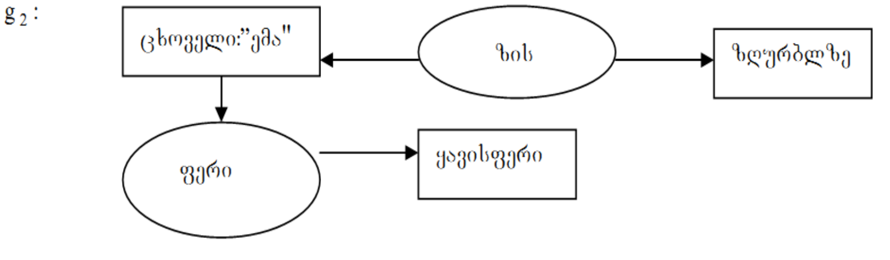
1. თუ ცნება მონიშნულია ზოგადი მარკერით, შესაძლებელია ის შევცვალოთ ინდივი-დუალური მარკერით.
2. ტიპის ჭდე შეიძლება შეიცვალოს მისი ქვეტიპის ჭდით, თუ ის შეესაბამება ობიექტს, რომელზედაც მიუთითებს ცნება.

გაერთიანების ოპერაციით შეგვიძლია მოვახდინოთ ორი გრაფის ინტეგრირება ერთში. თუ მათ გააჩნიათ ცნების იდენტური კვანძები, მაშინ ის ინტეგრირებულ გრაფში შევა ერთხელ. გაერთიანების ოპერაცია წარმოადგენს სპეციალიზაციის წესს, რადგანაც ინტეგრირებული გრაფი არის ნაკლებად ზოგადი, ვიდრე ნებისმიერი მისი კომპონენტი.

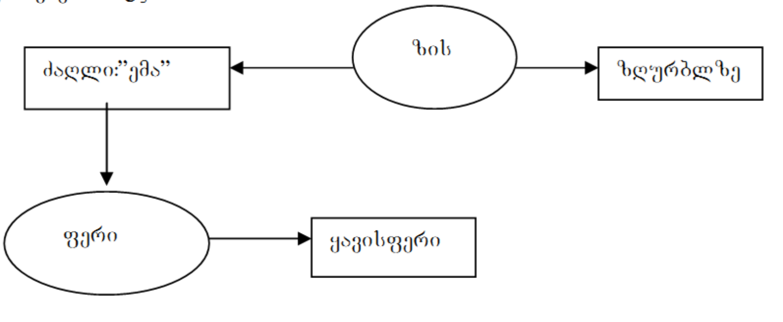
თუ გრაფი შეიცავს ორ ერთნაირ მიმართებას, მაშინ შესაძლებელია ამოვშალოთ ერთერთი მათგანი თავის რკალებთან ერთად. ესაა გამარტივების ოპერაცია. სურ. #8.14. მოცემულია ამ ოპერაციების გამოყენების მაგალითები. დავუშვათ, გვაქვს ორი გრაფი $\mathbf{g}_1$



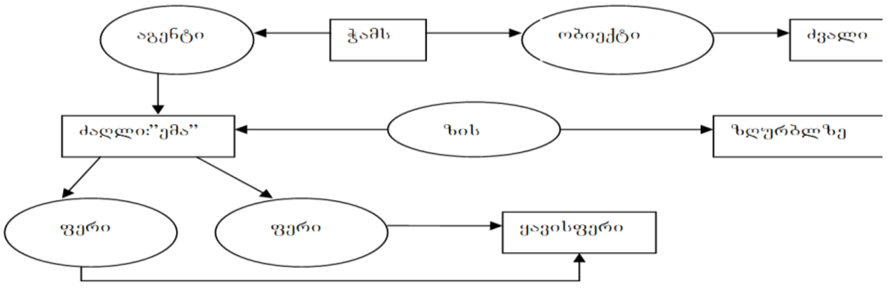
და $\mathbf{g}_2$,



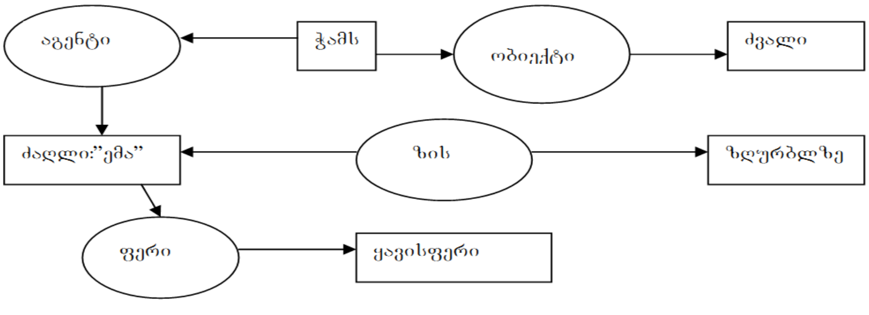
$\mathbf{g}_2$ -ს შეზღუდვა იქნება $\mathbf{g}_3$:



g_1 და g_3 გაერთიანება იქნება g_4:



g_4 გამარტივება იქნება g_5:



სურ. #8.14. ოპერაციები კონცეპტუალურ გრაფებზე.

8.6. ცოდნის წარმოდგენა კონცეპტების საშუალებით. როდესაც ადამიანები მსჯელობენ, საუბრობენ ერთმანეთში, ან ფიქრობენ რაიმე პრობლემაზე, ისინი იყენებენ ცნებებს, რომლებიც პრაქტიკულად ბუნებაში არ არსებობენ. მაგალითად, ჩვენ ვხმარობთ ცნებას „მაგიდა", მაგრამ ბუნებაში არ არსებობს რაღაც ზოგადი მაგიდა, ყოველთვის არსებობს რომელიღაც კონკრეტული მაგიდა, ასევე არ არსებობს ზოგადად ხე, ან შენობა. როგორ ხდება კონკრეტულ ობიექტებზე დაკვირვებით ამ ზოგადი ცნებების ფორმირება ბუნებრივ ინტელექტში? ჩვენ რა თქმა უნდა ზუსტად არ შეგვიძლია აღვწეროთ ის პროცესები, რომლებიც მიმდინარეობს ადამიანის თავის ტვინში, როდესაც ხდება რაიმე ცნების ფორმირება, მაგრამ ზოგადი სქემა გამოიყურება ასე: ადამიანი გარემოსთან ურთიერთობის პროცესში აკვირდება რა ერთი და იგივე ტიპის ობიექტებს, ახდენს მათ რაღაც პარამეტრების საშუალებით აღწერას, მრავალ ობიექტებზე დაკვირვების შედეგად ამ პარამეტრებიდან გამოყოფს ისეთებს, რომლებიც არსებითია მოცემული კლასის ობიექტებისათვის და ასეთნაირად დაფიქსირებულ ინფორმაციას უწოდებს ცნებას. ამ პროცესში მნიშვნელოვანია, რომ ცნების ჩამოყალიბების პროცესი არ გრძელდება უსასრულოდ და ადამიანს შეუძლია მოცემული კლასის სასრული ელენეტების საფუძველზე ჩამოაყალიბოს თავისი ცნება. მეორე მნიშვნელოვანი მომენტია, რომ ამ ცნების განმსაზღვრელია არა ადამიანის შეხედულება ამა თუ იმ ობიექტზე, არამედ სწორედ ამ ობიექტების თვისებები. ეს რომ ასე არ იყოს, ადამიანებს შორის შეუძლებელი იქნებოდა ურთიერთობა, რადგან ერთსა და იმავე ცნების ქვეშ ისინი იგულისხმებდნენ სულ სხვადასხვა ობიექტებს. ცხადია, რომ განსხვავება სხვადასხვა ადამიანებების ერთსა და იმავე ცნებებს შორის არსებობს, მაგრამ თუ ეს ცნებები ასახავენ ობიექტებს, რომლებიც ბუნებაში არსებობენ, მაშინ ეს განსხვავება უმნიშვნელოა.

როგორ შეიძლება გავიმეოროთ ცნებათა ფორმირების პრინციპი კომპიუტერულ სისტემაში, რათა ცოდნა ჩამოვაყალიბოთ კონცეპტის (ცნების შესაბამისი, კომპიუტერით გამოთვლილი არტეფაქტი) სახით, რომელიც საშუალებას მოგვცემს არა მარტო კომპაქტურად შევინახოთ ცოდნა კომპიუტერში, არამედ გამოვიყენოთ ეს ცოდნა გადაწყვეტილებების მისაღებად თუ რაიმე პროცესების სამართავად. პირველ რიგში საჭიროა გვქონდეს შესაბამისი მათემატიკური აპარატი, რომელიც საშუალებას მოგვცემს აღვწეროთ ისეთი არამკაფიო, არამკაცრად ჩამოყალიბებული ობიექტები, როგორიცაა ცნებები. შემოვიტანოთ ალ-სიმრავლის (**M**) ცნება, რომელშიც ელემენტის მიკუთვნების ოპერაცია $\in$ თანაბრად შეიძლება ეკუთვნოდეს როგორც a, ასევე $\bar{a}$ ელემენტს, ანუ ამ სიმრავლეს ეკუთვნის $\check{a} \in \underline{M}$, სადაც

$$\check{a} = \left\{ \begin{matrix} a \\ \bar{a} \end{matrix} \right. .$$

განუზღვრელობის ეს ფორმა, რომელიც ჩვენ შემოგვაქვს, საშუალებას მოგვცემს აღვწეროთ ის განუზღვრელობა, რომელიც ახასიათებს ცნებების ფორმირების პროცესს. კონცეპტის გამოთვლის პროცედურა შედგება რამდენიმე ეტაპისაგან. პირველ რიგში უნდა გამოიყოს ნიშნების (თვისებების) ის სიმრავლე, რომლის საშუალებითაც მოხდება კონკრეტული ობიექტების აღწერა. მთავარი პრინციპი ასეთი ნიშნების გამოყოფის დროს მდგომარეობს იმაში, რომ შესაძლებელი უნდა იყოს ამ ნიშნების მნიშვნელობათა ობიექტურად დაკვირვება, ან შეფასება ექსპერტთა საშუალებით. ამასთან მნიშვნელობა არა აქვს ეს ნიშნები რაოდენობრივია, თუ ხარისხობრივი (ანუ შესაძლებელია მათი მნიშვნელობების რაოდენობრივად გაზომვა, თუ მხოლოდ ხარისხობრივი შეფასება). ნიშნების სიმრავლის გამოყოფასთან ერთად ხდება მათი მნიშვნელობების სიმრავლეების დადგენაც. ამასთან ეს სიმრავლეები უნდა იყოს დისკრეტული და სასრული.

ამ ეტაპის შემდეგ ჩვენ გვექნება მატრიცა $m \times n$, სადაც m არის ნიშნების რაოდენობა, ხოლო $n = \max_i (n_i)$ მნიშვნელობათა მაქსიმალური რაოდენობა. პირველი ეტაპი მთავრდება თითოეული ნიშნის მნიშვნელობათა ბინარიზაციით, ანუ მათი გაყოფით ორ ქვესიმრავლედ.

მეორე ეტაპზე უნდა მოვახდინოთ ამ ნიშნების საშუალებით იმ კონკრეტული ობიექტების აღწერა, რომელთა საშუალებითაც უნდა გამოვთვალოთ კონცეპტი. ამ დროს ყველაზე მნიშვნელოვანია კონკრეტული ობიექტების შეფასება იყოს ობიექტური, ანუ ყოველი კონკრეტული ობიექტისათვის გვქონდეს ცალსახა შეფასება ეკუთვნის, თუ არ ეკუთვნის მოცემული ობიექტი იმ ცნებას, რომლის კონცეპტის გამოთვლასაც ჩვენ ვაპირებთ. თუ ჩვენ არ ვაპირებთ გამოვთვალოთ რომელიმე კონკრეტული ადამიანის ცნების შესაბამისი კონცეპტი, მაშინ ეს შეფასებები უნდა იყოს მიღებული ექსპერიმენტატორისაგან დამოუკიდებელად.

მესამე ეტაპზე ხდება უკვე კონცეპტის გამოთვლა. ამისათვის შეიძლება გამოვიყენოთ კონცეპტის გამოთვლის ლოგიკური, ან სტატისტიკური მეთოდი. განვიხილოთ ეს უკანასკნელი. დავუშვათ, p_{ij} იყოს A_i ნიშნის a_{ij} მნიშვნელობის შეხვედრის სტატისტიკა იმ N ცალ კონკრეტულ ობიექტში, რომელთა საშუალებითაც ვითვლით კონცეპტს

$$p_{ij} = \frac{N(a_{ij})}{N}, i = \overline{1,m}, j = \overline{1,n_i} .$$

შემდეგ თითოეული A_i ნიშნისთვის შევკრიბოთ იმ მნიშვნელობების სიხშირეები, რომლებიც გაერთიანებულნი არიან ერთ ქვესიმრავლეში. ერთი სიმრავლე პირობითად აღვნიშნოთ როგორც a_i და მეორე კი $\bar{a}_i$. გვექნება $P_i = \sum_k p_{ik}$, სადაც აჯამვა ხდება იმ მნიშვნელობების მიხედვით, რომლებიც გაერთიანებულნი არიან a_i ქვესიმრავლეში და $\overline{P}_i = \sum_l p_{il}$, სადაც აჯამვა ხდება იმ მნიშვნელობების მიხედვით, რომლებიც გაერთია-

ნებულნი არიან $\overline{a_i}$ ქვესიმრავლეში. რა თქმა უნდა, რომ ადგილი აქვს ტოლობას $P_i + \overline{P_i} = 1$. ამ პროცედურის დამთავრების შემდეგ ჩვენ მივიღებთ

$$\varphi(P) = \begin{pmatrix} P_1, P_2, \dots, P_m \\ \overline{P_1}, \overline{P_2}, \dots, \overline{P_m} \end{pmatrix}$$

გამოსახულებას. ამის შემდეგ, შემოგვაქვს ფილტრაციის დონე, რათა გადავიდეთ კონცეპ-ტის აღწერის ლოგიკურ დონეზე. ფილტრაციის დონე, ანუ სიზუსტის დონე დამოკიდე-ბულია იმაზე, თუ რა სიზუსტით გვინდა გამოვთვალოთ კონცეპტი. შემოგვაქვს მუდმივი სიდიდეები ψ და ψ, ისე, რომ $\psi + \overline{\psi} = 1$ და ვახდენთ ფილტრაციას შემდეგი წესით: თუ $P_i \geq \psi$ (შესაბამისად $\overline{P_i} \leq \overline{\psi}$) მაშინ ვთვლით, რომ ეს ნიშანი წარმოადგენს კონცეპტის დამახასიათებლ, მნიშვნელოვან ნიშანს და კონცეპტის აღწერაში ის უნდა შედიოდეს A_i (შესაბამისად $\overline{A_i}$) ფორმაში. თუ $P_i < \psi$ (შესაბამისად $\overline{P_i} > \overline{\psi}$), მაშინ ეს ნიშანი არაარ-სებითია და ის კონცეპტის აღწერაში უნდა შედიოდეს $\check{A}_i$ ფორმაში. ასეთი ოპერაციის შემ-დეგ ჩვენ მივიღებთ კონცეპტის საბოლოო სახეს:

$$\varphi(S) = (A_1, A_2, \overline{A_3}, \check{A}_4, \dots \check{A}_m)$$

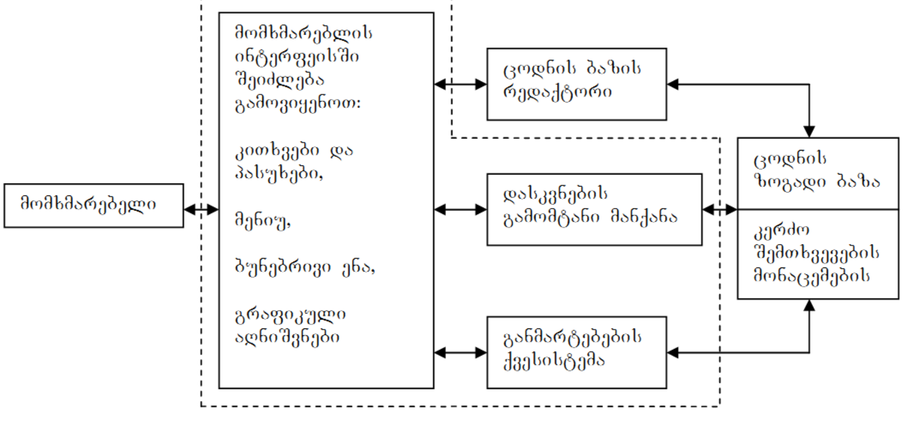
სადაც ნიშნებს შორის იგულისხმება ლოგიკური ოპერაცია $\wedge$. ამგვარად, ჩვენ შევძლებთ რაიმე **S** ობიექტის ზოგადი აღწერიდან გამოვყოთ ის დამახასიათებელი ნიშნები, რომ-ლებიც განსაზღვრავენ ამ ობიექტის არსს. გამოთვლილი კონცეპტი იქნება ყველაზე უხეში მიახლოება (ე.წ. პირველი დონის კონცეპტი). თუ ეს დონე ჩვენ არ დაგვაკმაყოფილებს, შესაძლებელია გამოვთვალოთ უფრო მაღალი დონის კონცეპტი. ამისათვის საჭიროა უფ-რო დავაზუსტოთ ნიშნები და მათი მნიშვნელობები და ფილტრაციის დონე.

საკონტროლო კითხვები:

1. რა უპირატესობა აქვთ ცოდნაზე დაფუძნებულ აგენტებს სხვა აგენტებთან შედა-რებით?
2. რა განსხვავებაა კომპიუტერში მონაცემთა წარმოდგენასა და ცოდნის წარმოდგენას შორის?
3. რა პრინციპებს ეფუძნება სემანტიკური ქსელები?
4. როგორ ხდება ცნებების დაკავშირება სემანტიკურ ქსელში?
5. რა არის ფრეიმი?
6. როგორ ხდება ფრეიმის ჩამოყალიბება?
7. როგორ იქმნება ფრეიმული სისტემა?
8. რა უპირატესობა აქვს ფრეიმულ სისტემას სემანტიკურ ქსელთან შედარებით?
9. რას წარმოადგენს კონცეპტუალური გრაფი?
10. რით განსხვავდება ის სემანტიკურიქსელისა და ფრეიმებისაგან?
11. ჩამოთვალეთ კონცეპტუალურ გრაფებზე განმარტებული ოპერაციები.
12. რას ეწოდება კონცეპტი?
13. რა ეტაპებისაგან შედგება კონცეპტის გამოთვლის პროცედურა?
14. მოიფიქრეთ, როგორ შეიძლება გამოიყენოს ინტელექტუალურმა სისტემამ კონცეპტის სახით ჩამოყალიბებული ცოდნა.

ლექცია 9. ამოცანათა ამოხსნის ძლიერი მეთოდები

9.1. ცოდნის ინჟინერია. როგორც უკვე ვიცით წინა ლექციებიდან, ალგორითმები, რომლებიც ეძებენ ამოცანის ამოხსნას სპეციფიკური ცოდნის გარეშე, წარმოადგენენ სუსტ ალგორითმებს. ამ ლექციაში ჩვენ განვიხილავთ როგორ შეიძლება ცოდნის გამოყენებით უფრო ეფექტურად ამოვხსნათ ინტელექტუალური ამოცანები. ცოდნის ინჟინერიის ძირითადი პრინციპის თანახმად, ამოცანათა ამომხსნელი აგენტის შესაძლებლობებს პირველ რიგში განასხღვრავს ცოდნა, მისი ინფორმაციული ბაზა და მხოლოდ შემდეგ დასკვნების გამოტანის, ანუ ძებნის მეთოდები. ასეა ბუნებრივი ინტელექტის შემთხვევაშიც, ადამიანი – ექსპერტი კარგად ასრულებს თავის სამუშაოს სწორედ იმიტომ, რომ მან ბევრი იცის პრობლემის შესახებ და გააჩნია ამ პრობლების გადაჭრის პრაქტიკული ჩვევები. ასეთი უნდა იყოს ხელოვნური აგენტიც, იცოდეს ბევრი რამ კონკრეტული პრობლემის შესახებ და გააჩნდეს

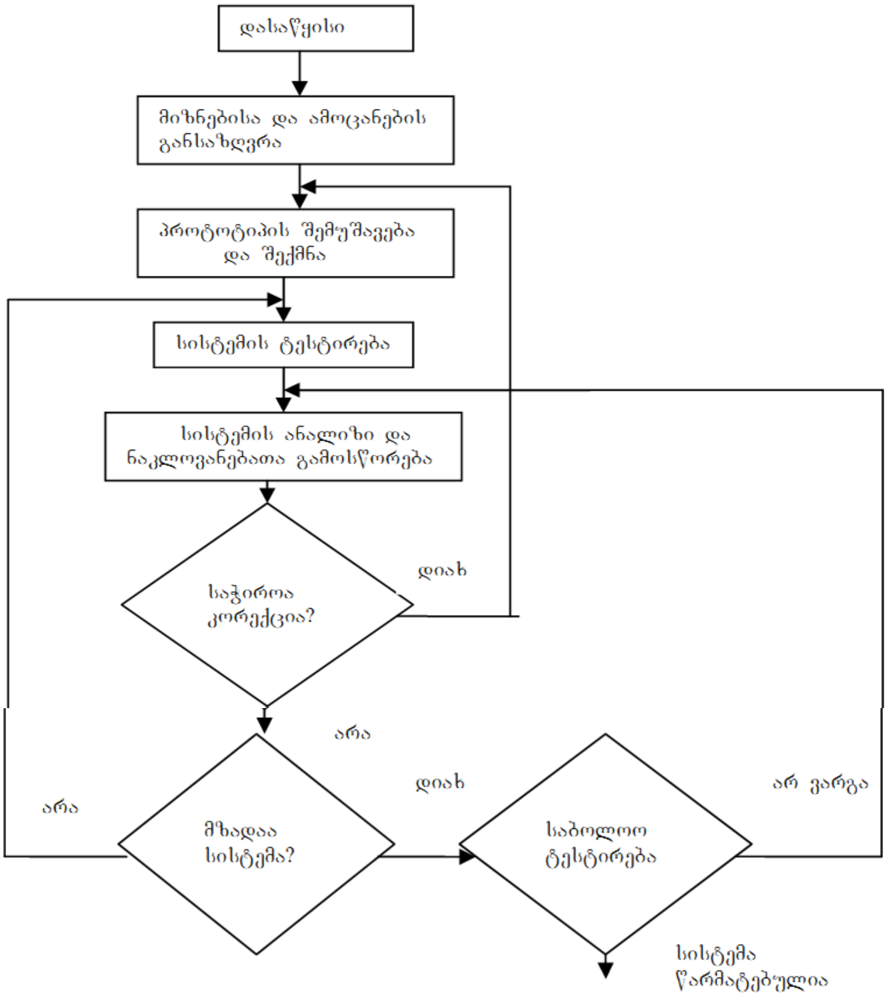


სურ. #9.1. ტიპური ექსპერტული სისტემის არქიტექტურა

ამ სფეროში პრობლემათა გადაჭრის პრაქტიკული ჩვევები. ასეთ აგენტებს სხვანაირად უწოდებენ ექსპერტულ სისტემებს. ცხადია, რომ როგორც ცოდნა, ასევე პრაქტიკული ჩვევები ექსპერტულ სისტემაში უნდა ჩაიდოს ადამიანის მიერ. ექსპერტული სისტემები გამოიყენება სხვადასხვა პრობლემების გადასაჭრელად პრაქტიკულად ადამიანების მოღვაწეობის ყველა სფეროში. მათი ძირითადი ამოცანებია: დიაგნოსტიკა, პროგნოზირება, პროექტირება, დაგეგმვა, მონიტორინგი და მართვა.

9.2. ექსპერტული სისტემების ტექნოლოგია. სურ. #9.1. ნაჩვენებია ტიპური ექსპერტული სისტემის არქიტექტურა. მომხმარებელი სისტემას უკავშირდება მომხმარებლის ინტერფეისის საშუალებით, რომელიც მომხმარებლის ინტერესებიდან, მონაცემთა ბაზისა და დასკვნების გამომტანი სისტემის მოთხოვნებიდან გამომდინარე, შეიძლება იყოს სხვადასხვა სახის. ექსპერტული სისტემა შეიცავს ცოდნის ბაზას, რომელიც წარმოადგენს სისტემის ძირითად ბირთვს და ინახავს ცოდნას რაიმე კონკრეტული პრობლემის გარშემო, განმარტებების ქვესისტემას და დასკვნების გამომტან მანქანას. ექსპერტულ სისტემებში, რომლებიც ეფუძნებიან წესებს, ცოდნა წარმოდგენილია შემდეგი სახით: **თუ…. . . მაშინ,** ანუ, თუ სრულდება რაიმე პირობა, მაშინ დასკვნა იქნება შემდეგი. მონაცემთა ბაზა შეიცავს როგორც ზოგად ცოდნას, ასევე ცოდნას კერძო შემთხვევების შესახებ. დასკვნების გამომტანი მანქანა იყენებს ბაზაში არსებულ ცოდნას და პროდუქციული სისტემის საშუალებით მათ საფუძველზე ღებულობს დასკვნებს. პროდუქციული სისტემა წარმოადგენს სისტემას, რომელიც უზრუნველყოფს ამოცანის ამოხსნის პროცესის მართვას რაიმე მოდელის მი-

ხედვით და შედგება პროდუქციული წესების, სამუშაო მეხსიერებისა და მართვის ციკლისაგან - “გამოცნობა – ქმედება“. პროდუქციული სისტემის მაგალითი ჩვენ გვქონდა პირველ ლექციაში, როდესაც ვიხილავდით პროგრამა **LG**-ს. პროცედურები, რომლებსაც ასრულებს მართვის ციკლი, განცალკევებულია პროდუქციული



სურ. #9.2. ექსპერტული სისტემის დამუშავების ციკლები

წესებისაგან, რაც იძლევა საშუალებას ავაგოთ ცოდნის ბაზა მმართველი პროგრამის გათვალისწინების გარეშე: შევიტანოთ ცვლილებები ბაზის რაიმე ნაწილში ისე, რომ არ წარმოიშვას პრობლემები ბაზის დანარჩენ ნაწილში და გამოვიყენოთ სხვადასხვა ექსპერტულ სისტემებში ერთი და იგივე მართვის პროგრამა და ინტერფეისი, ანუ ცლკე შევქმნათ ექსპერტული სისტემის გარსი და შემდეგ მასში სხვადასხვა პრობლემის შესახებ ცოდნის ბაზის ჩადებით მივიღოთ სხვადასხვა ექსპერტული სისტემა.

განმარტებების ქვესისტემა საშუალებას აძლევს მომხმარებელს მიიღოს სისტემისაგან განმარტებები, თუ რის საფუძველზე გამოაქვს სისტემას ესა თუ ის დასკვნა და დარწმუნდეს სისტემის სწორ ფუნქციონირებაში. ექსპერტული სისტემის შესაქმნელად საჭიროა ცოდნის ინჟინერიის სპეციალისტების, მოცემული პრობლემის ქსპერტებისა და მომხმარებელთა ერთობლივი მუშაობა. სურ. #9.2. ნაჩვენებია ექსპერტული სისტემის დამუშავების ციკლი. საიდანაც კარგად ჩანს, რომ ეს პროცესი საკმაოდ რთული, შრომატევადი და

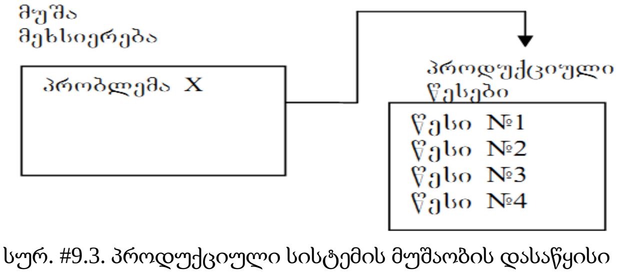
ძვირადღირებული პროცესია, მაგრამ კარგი ექსპერტული სისტემა ყოველთვის ანაზღაურებს მასზე დახარჯულ სახსრებს.

მუშაობა იწყება ცოდნის ინჟინერისა და მოცემული პრობლემის ექსპერტის მიერ იმ ცოდნის განსაზღვრით, რომელიც იქნება აუცილებელი ექსპერტული სისტემის ფუნქციონირებისთვის. მას შემდეგ, რაც ცოდნის ინჟინერი მიიღებს წარმოდგენას, თუ როგორ ხდება მოცემული პრობლემის გადაწყვეტა, ის უკვე იწყებს სისტემის დამუშავებას, ფიქრობს იმაზე, თუ რა სახით წარმოადგინოს ცოდნა სისტემაში (მაგალითად, წესებით თუ კონცეპტუალური გრაფებით), განსაზღვრავს ძებნის როგორ სტრატეგიას გამოიყენებს სისტემა (პირდაპირი თუ უკუძებნა), დაამუშავებს ინტერფეისს. ამის საფუძველზე აიგება პროტოტიპი. ამის შემდეგ იწყება ამ პროტოტიპის დახვეწა, სანამ მომხმარებელი არ დარჩება კმაყოფილი მიღებული შედეგით.

9.3. ექსპერტული სისტემის ფუნქციონირება. განვიხილოთ წესებზე დაფუძნებული ექსპერტული სისტემის მაგალითი. წესებზე დაფუძნებული ექსპერტული სისტემის არქიტექტურა შეიცავს ამოცანათა ამოხსნის პროდუქციულ სისტემას. ასეთ შემთხვევაში შეიძლება ჩავთვალოთ, რომ ცოდნის ბაზა შედგება პროდუქციული წესებისაგან. განვიხილოთ ავტომობილის დიაგნოსტიკის ექსპერტული სისტემის მარტივი მაგალითი. დავუშვათ, ცოდნის ბაზა შედგება მხოლოდ ოთხი პროდუქციული წესისგან:

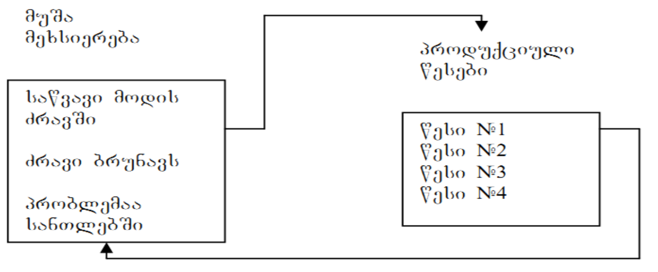
თუ საწვავი მოდის ძრავში და ძრავი ბრუნავს, **მაშინ** პრობლემას წარმოადგენენ სანთლები;
თუ ძრავი არ ბრუნვს და ფარებიც არ ინთება, **მაშინ** პრობლემას წარმოადგენს აკუმულატორი, ან გამტარები;
თუ ძრავი არ ბრუნავს და ფარები ინთება, **მაშინ** პრობლემას წარმოადგენს სტარტიორი;
თუ ავზში საწვავი არის და საწვავი მოდის კარბურატორში, **მაშინ** საწვავი მოდის ძრავშიც.

ამ წესების საფუძველზე მუშა მეხსიერებაში მოთავსდება მაღალი დონის მიზანი - პრობლემა X, რომელიც უნდა გამოიკვლიოს ექსპერტულმა სისტემამ. X შეიძლება იყოს ნებისმიერი ფრაზა, მაგალითად, თავიდან ეს არის ფრაზა “ძრავი არ იქოქება.” (იხ. სურ. #9.3).



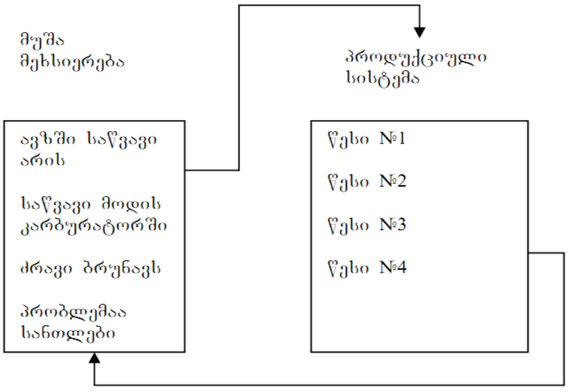
სურ. #9.3. პროდუქციული სისტემის მუშაობის დასაწყისი

მუშა მეხსიერებაში მოთავსებულ ფრაზას სისტემა შეუსაბამებს სამ წესს და თუ ამ წესების განხილვა იწყება წესით, რომლის ნომერიც ყველაზე პატარაა, მაშინ სისტემა გააქტიურებს #1 წესს, ამასთან X-ს შეუსაბამებს ფრაზას “სანთლები” (იხ. სურ. #9.4.),



სურ. #9.4. პროდუქციული სისტემა პირველი წესის გააქტიურების შემდეგ.

ხოლო მისი წინაპირობები კი მოთავსდება მუშა მეხსიერებაში. პირველი წესის სამართლიანობის დასადგენად დაკმაყოფილებული უნდა იყოს ორი პირობა: საწვავი მოდის ძრავში და ძრავი ბრუნავს. იმისათვის, რომ დაადგინოს საწვავი მოდის თუ არა ძრავში, სისტემა გააქტიურებს წესს #4 (იხ. სურ. #9.5).



სურ. #9.5. პროდუქციული სისტემა წესი #4 გააქტიურების შემდეგ

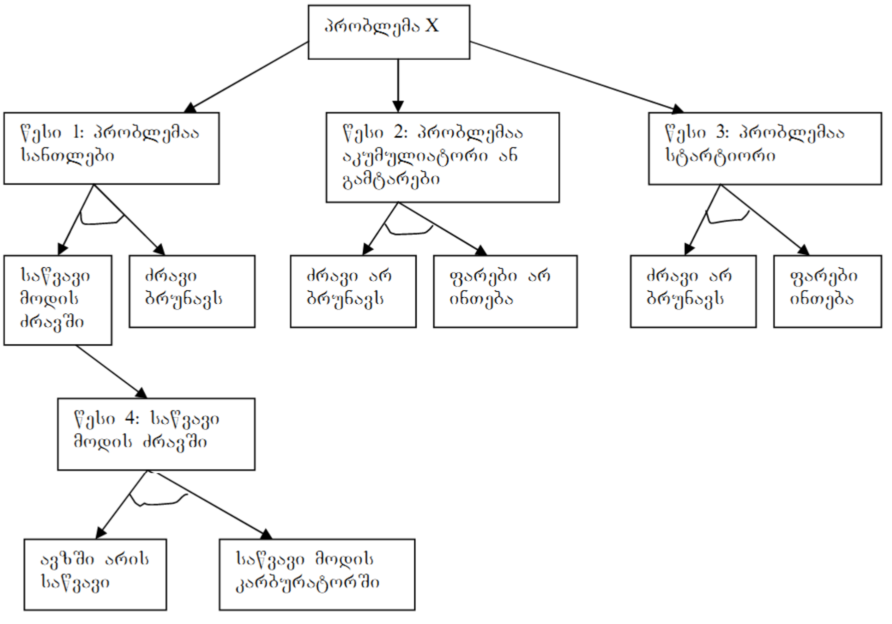
გადმოიტანს რა მეოთხე წესის წინაპირობებს მუშა მეხსიერებაში, სისტემა აღმოაჩენს, რომ აქ არის სამი ელემენტი, რომლებიც არ შეესაბამებიან ცოდნის ბაზაში მოთავსებულ არცერთ წესს, ამიტომ სისტემა შეეკითხება მომხმარებელს, სრულდება თუ არა ეს პირობები, თუ მომხმარებლისაგან ის მიიღებს სამივე ელემენტის დადასტურებას, მაშინ გამოიტანს დასკვნას, რომ პრობლემას წარმოადგენენ სანთლები, თუ არ დადასტურდება თუნდაც ერთი ელემენტი მაინც, სისტემა ჩათვლის რომ, პირველი წესი არ შეესაბამება მოცემულ სისტემას და გადავა მეორე წესის განხილვაზე.

სურ. #9.6. მოცემულია გრაფი, რომელიც აიგება სისტემის მუშაობის დროს. პატარა რკალი გრაფზე მიუთითებს, რომ პრობლემის გადასაწყვეტად საჭიროა ორივე პირობის შესრულება (ანუ კვანძი გრაფზე არის ''და'' ტიპის). როგორც ამ გრაფიდან კარგად ჩანს, ალგორითმი წარმოადგენს სიღრმეში გადარჩევის ალგორითმს. მიიღებს რა ინფორმაციას მომხმარებლისაგან იმის შესახებ, არის თუ არა საწვავი ავზში და მოდის თუ არა საწვავი კარბურატორში (უნდა სრულდებოდეს ორივე პირობა ერთდროულად) ალგორითმი ამოდის ზედა დონეზე და კვლავ ითხოვს ინფორმაციას "ბრუნავს თუ არა ძრავი?" დადებითი პასუხის შემთხვევაში სისტემა დაასკვნის, რომ პრობლემა მდგომარეობს სანთლებში, თუ არა – გადადის მეორე წესზე. მიუხედავად იმისა, რომ ეს სისტემა ძალიან მარტივია, ის კარგად ასახავს თუ როგორ შეიძლება საწყისი პრობლემის დეკომპოზიცია უფრო მარტივ ამოცანებად პროდუქციული სისტემის და და/ან ტიპის გრაფების გამოყენებით.

აღწერილი სისტემა წარმოადგენს პროდუქციული სისტემის გამოყენების მაგალითს მიზნის საფუძველზე, რომელშიც გამოიყენება ძებნა სიღრმეში, რადგანაც სისტემა არ გადავა მეორე ქვემიზნის გამოკვლევაზე მანამდის, სანამ ის არ მოახდენს მოცემული ქვემიზნის სრულ გამოკვლევას.

9.4. განმარტებების ქვესისტემა. რა როლს ასრულებს ექსპერტულ სისტემაში განმარტებების ქვესისტემა? პროდუქციული სისტემა ახდენს ძებნას გრაფზე, განმარტებების ქვესისტემა კი თვალყურს ადევნებს ამ პროცესს და იყენებს მიღებულ ინფორმაციას, რათა უპასუხოს მომხმარებლის შეკითხვებს. როგორც წესი, ექსპერტული სისტემები პასუხობენ ორი ტიპის შეკითხვებს – "რატომ?" და "როგორ?". პირველი ტიპის შეკითხვები წარმოიშვება, როდესაც სისტემა ითხოვს დამატებით ინფორმაციას მომხმარებლისაგან. მაგა-

ლითად, როდესაც სისტემამ მოითხოვა “არის თუ არა ავზში საწვავი?” მიიღო პასუხი “დიახ”. შემდეგ მოითხოვა ინფორმაცია მოდის თუ არა საწვავი კარბურატოში?” და მიიღო პასუხი “დიახ”. ამის შემდეგ ის კვლავ ეკითხება მომხმარებელს “ზრუნავს თუ არა ძრავი?”



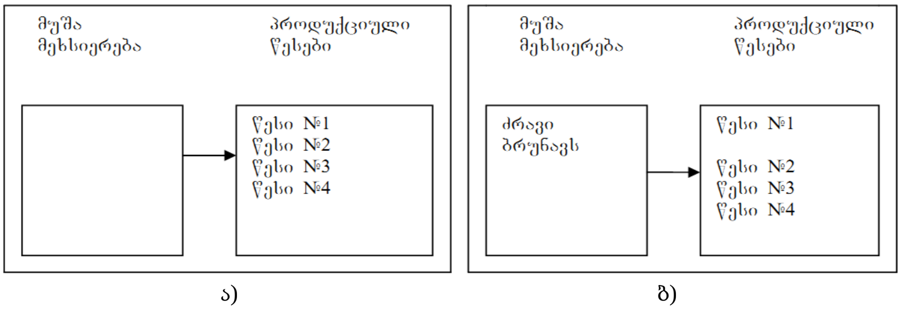
სურ. #9.6. და/ან ტიპის გრაფი, რომელიც აიგება ექსპერტულ სისტემაში. პრობლემა X აქ არის “ძრავი არ იქოქება”

ნებისმიერ ეტაპზე მომხმარებელი შეიძლება დაინტერესდეს, თუ რისთვის სჭირდება სისტემას ეს ინფორმაცია და ის ეკითხება სისტემას “რატომ?”, ანუ რისთვის სჭირდება სისტემას ეს ინფორმაცია. განმარტებების ქვესისტემა, რომელიც თვალყურს ადევნებდა ამ პროცესს, იძლევა განმარტებას ”თანახმად პირველი წესისა, თუ საწვავი მოდის ძრავში და ძრავი ბრუნავს, მაშინ პრობლემას წარმოადგენენ სანთლები”. თუ მომხმარებელი დაინტერესდება, თუ როგორ დაადგინა სისტემამ, რომ საწვავი მოდის ძრავში, მაშინ განმარტებების ქვესისტემა უპასუხებს, რომ “მეოთხე წესის თანახმად, თუ საწვავი არის ავზში და საწვავი მოდის კარბურატორში, მაშინ საწვავი მოდის ძრავში”. ამით განმარტებების ქვესისტემა უზრუნველყოფს იმას, რომ მომხმარებელი ყოველთვის საქმის კურსშია, თუ რას აკეთებს სისტემა და შეუძლია მიუთითოს, ან სულაც შეწყვიტოს მუშაობა, თუკი სისტემა უშვებს შეცდომას, ან მოქმედებს არასწორი მიმართულებით.

9.5. მონაცემებზე დაფუძნებული პროდუქციული სისტემა. განხილული სისტემა წარმოადგენდა პროდუქციულ სისტემას, რომელიც დაფუძნებული იყო მიზანზე. ეხლა ვნახოთ, როგორ შეიძლება იგივე სისტემა მუშაობდეს მონაცემების საფუძველზე. განსხვავებით მიზანზე დაფუძნებული სისტემისაგან, სადაც ძებნა მიმდინარეობს სიღრმეში რადგანაც სანამ სისტემა არ ჩაატარებს ერთი ქვემიზნის ამომწურავ ანალიზს, არ გადავა მეორე ქვემიზანზე, მონაცემთა საფუძველზე მუშაობისას სისტემა უფრო ხშირად აწარმოებს განივ ძებნას. ალგორითმი ძალიან მარტივია.

მუშა მეხსიერებაში მოთავსებული მონაცემები დარდება ცოდნის ბაზაში არსებული წესების პირობებს. თუ მონაცემებს მივყავართ რომელიმე წესის აქტივიზაციასთან, შედეგი მოთავსდება მუშა მეხსიერებაში და მართვა გადაეცემა შემდეგ წესს. ჩვენი მაგალითის შემთხვევაში საწყის ეტაპზე მუშა მეხსიერებაში არ არის არავითარი ინფორმაცია და სისტემა

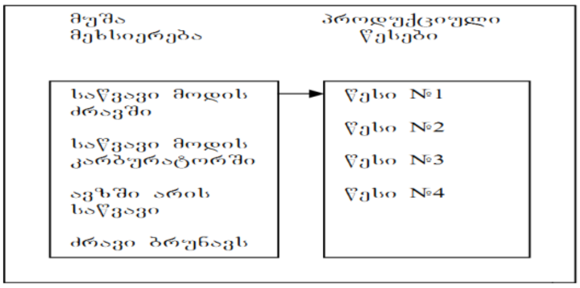
იწყებს წესების წინაპირობების შემოწმებას, რათა მოძებნოს ინფორმაცია, რომელსაც ის მოითხოვს მომხმარებლისგან (სურ. #9.7. ა)).



სურ. #9.7. მონაცემთა საფუძველზე მომუშავე სისტემის პირველი და მეორე ეტაპები:
ა) საწყის ეტაპი; ბ) სისტემა მეორე წესის პირველი წინაპირობის შეფასების შემდეგ.

პირველი წესის წინაპირობა "საწვავი მოდის ძრავში არ მიეკუთვნება ასეთ ინფორმაციას, ამიტომ სისტემა გადადის მეორე წესსზე და განიხილავს წინაპირობას "ძრავი არ ტრიალებს", ეს იქნება სწორედ ის ინფორმაცია, რომელიც სისტემამ უნდა მოითხოვოს მომხმარებლისაგან. დავუშვათ, აღმოჩნდა, რომ ეს წინაპირობა არასწორია, ამიტომ როგორც მეორე, ასევე მესამე წესის გამოყენება არ შეიძლება სისტემა გადავა მეოთხე წესის განხილვაზე და ნახავს, რომ ამ წესის ორივე წინაპირობა არის ის ინფორმაცია, რომელიც მან უნდა მოითხოვოს მომხმარებლისაგან. ამის შემდეგ მუშა მეხსიერებაში მას ექნება შემდეგი სახის მონაცემები: "საწვავი მოდის ძრავში", "საწვავი მოდის კარბურატორში", "საწვავი არის ავზში" და აქვე მოთავსდება დასკვნა, რომელსაც სისტემა მიიღებს მეოთხე წესის საფუძველზე - "ძრავი ტრიალებს". ამის შემდეგ სისტემა კვლავ დაიწყებს პროდუქციული წესების ხელახლა განხილვას. სისტემა აღმოაჩენს, რომ პირველი წესის თანახმად და იმ მონაცემების საფუძველზე, რაც მას გააჩნია, შესაძლებელია მხოლოდ ერთადერთი დასკვნის გაკეთება, რომ "პრობლემაა სანთლები" და დაამთავრებს მუშაობას.

განსხვავებით მიზანზე დაფუძნებული სისტემისაგან, მონაცემებზე დაფუძნებულ სისტემაში ძებნას აქვს უფრო არასისტემური ხასიათი. ამის მიზეზი ძალიან მარტივია, მიზანზე დაფუძნებული სისტემა ძირითად ამოცანას ყოფს უფრო პატარა ქვეამოცანებად და ეს ქვეამოცანების შეიძლება დაიყოს კიდევ უფრო პატარა ქვეამოცანებად. ამის გამო ძებნა ყოველთვის მიმართულია იერარქიულად მიზნიდან მის შემადგენელებისკენ. მონაცემებზე დაფუძნებულ სისტემაში ასეთი მიზანი არ არსებობს და ძებნა ხორციელდება ხეზე, რომელიც დამოკიდებულია მხოლოდ პროდუქციული წესების მიმდევრობასა და ახალი ინფორმაციის გამოჩენაზე. ამიტომ ძებნის პროცესი უფრო არამიმართულია და ნაკლებად ცხადია.



სურ. #9.8. მონაცემებზე დაფუძნებული სისტემა მეოთხე წესის განხილვის შემდეგ იწყებს წესების ხელმეორედ განხილვას.

აქედან გამომდინარე განმარტებების ქვესისტემის მუშაობაც უფრო შეზღუდულია, კერძოდ ამ სისტემაში განმარტებების ქვესისტემას არ შეუძლია უპასუხოს კითხვას "რატომ?" და თუ პროდუქციული წესების მიმდევრობა ზუსტად არ აღირიცხება, ქვესისტემას ასევე უჭირს გასცეს პასუხი კითხვას "როგორ?".

9.6. ევრისტიკები ცოდნის ბაზის აგების დროს. ცოდნის ბაზისა და დასკვნების გამოტანის მექანიზმის ასეთი გაყოფის გამო, ცხადია, რომ ძებნის ეფექტურობა თითქმის მთლიანად დამოკიდებულია ცოდნის ბაზაში პროდუქციული წესების სტრუქტურასა და მათ იერარქიულად მოწესრიგებაზე. განსაკუთრებით მნიშვნელოვანის, რომ წესის ფორმა, **თუ p,q** და **r**, **მაშინ s.** მიუხედავად იმისა რომ ერთი შეხედვით გვაგონებს ლოგიკურ გამოსახულებას, შესაძლებელია აღქმული იქნას როგორც ამოცანის ამოხსნის პროცედურები ან ბიჯები: "რომ მივიღოთ **s**, საჭიროა შევასრულოთ ჯერ **p**, შემდეგ **q** და შემდეგ **r** ქმედებები". ცოდნის ბაზის ასეთი მოწესრიგების პრობლემა ჩანს ჩვენ მაგალითშიც. პირველი პროდუქციული წესი მოითხოვს, რომ ჯერ დავადგინოთ, მოდის თუ არა საწვავი ძრავში და შემდეგ – ბრუნავს თუ არა ძრავი. მაგრამ ასეთი რიგითობა არაეფექტურია, რადგანც პირველი პირობის დადგენას სჭირდება უფრო მეტი გამოკვლევა, ვიდრე მეორე პირობის. აქედან გამომდინარე სისტემის ეფექტიანობა მთლიანად დამოკიდებულია ცოდნის ბაზის ისეთნაირ მოწესრიგებაზე, რომ გარკვეული იყოს წესების რიგი, წინაპირობების ორგანიზება წესებში და სხვადასხვა შემოწმებების ფასი. ყველა ეს ქმედება თავისი შინაარსით არის ევრისტიკული და ასახავენ ექსპერტის მიდგომებს და მის ცოდნას მოცემული პრობლემის შესახებ. თავისთავად ცხადია, რომ ზოგიერთ შემთხვევაში ისინი შეიძლება იყოს მცდარიც. მაგალითად, ჩვენს შემთხვევაში, თუ საწვავი მოდის ძრავში და ძრავი ბრუნავს, პრობლემა შეიძლება იყოს ელქტრული ნაპერწკალის გამანაწილებელ სისტემაში და არა სანთლებში. ამიტომ საჭიროა, რომ ექსპერტული სისტემის ცოდნის ბაზის შექმნის დროს გათვალისწინებული იყოს სხვადასხვა ექსპერტების რეკომენდაციები და ექსპერიმენტების საფუძველზე სრულად დაიხვეწოს ცოდნის ბაზა.

ჩვენი მაგალითი იყო მარტივი და ის ეხებოდა დიაგნოსტიკის მხოლოდ ერთ პრობლემას "ძრავი არ იქოქება". მაგრამ თუ ჩვენ მოვინდომებთ მანქანის სრული დიაგნოსტიკის სისტემის შექმნას, პროდუქციული წესების რაოდენობა გაიზრდება ძალიან სწრაფად და ასეთი პრინციპით მუშაობა, რომ შემოწმდეს ყველა წესი, უკვე აღარ იქნება ოპტიმალური, მით უმეტეს, რომ ცხადია, რომ მაგალითად, თუ მანქანის დამუხრუჭების მანძილი არ გვაკმაყოფილებს, აქ არავითავი მნიშვნელობა არა აქვს იქოქება თუ არა მანქანა. ამიტომ საჭირო ხდება პროდუქციული წესების ქსელურ სტრუქტურებში გაერთიანება. არსებობს შესაბამისი ალგორითმები, რომლებიც ასრულებენ ამ პროცესს და საშუალებას გვაძლევენ კიდევ უფრო გავაუმჯობესოთ ექსპერტული სისტემის მუშაობა.

ექსპერტული სისტემა აუცილებელი არ არის დაფუძნებული იყოს მიზანზე ან მონაცემებზე. არსებეობენ ექსპერტული სისტემები, რომლებიც ეფუძნებიან გამოცდილებას ან მოდელს. მოდელზე დაფუძნებული ექსპერტული სისტემა უფრო ხშირად გამოიყენება ისეთი პრობლემების გადასაჭრელად, რომელთა მოდელები მაქსიმალურად ობიექტურად აღიწერება და გვაქვს ხელსაწყოები, რომლებიც საშუალებას გვაძლევენ ზუსტად გავზომოთ საჭირო ინფორმაცია. ასეთი ექსპერტული სისტემების კარგ მაგალითს წარმოადგენს **NASA**-ს მიერ დამუშავებული ექსპერტული სისტემა, რომელიც გამოიყენება კოსმოსური ობიექტების ძრავის სამართავად ავტონომიურ რეჟიმში, მათი უწესრიგობებების გამოვლენა და შესაბამისად, რადგანც ამ უწესრიგობების აღმოფხვრა ავტონომიურ რეჟიმში შეუძლებელია ფუნქციონირების ახალი რეჟიმების შერჩევა.

სისტემები, რომლებიც დაფუძნებულნი არიან გამოცდილებაზე (ეს გამოცდილება რა თქმა უნდა წარმოადგენს ექსპერტის გამოცდილებას), პრობლემის გადაჭრის დროს თავის ცოდნის ბაზაში ეძებენ ანალოგიურ შემთხვევას და მოდელისა და ამ შემთხვევის საფუძველზე ღებულობენ გადაწყვეტილებებს. შესაძლებელია ჰიბრიდული ექსპერტული სისტემების არსებობაც, რომლებშიც გვხვდება ზემოთჩამოთვლილი მიდგომების სხვადასხვა კომბინაციები, რაც კიდევ უფრო ეფექტურს ხდის სისტემის მუშაობას. მიღწეული წარმატებები ექსპერტული სისტემების შექმნის დარგში ადასტურებს, რომ ეს წარმატე-

ბები მიიღწევა არა იმ მექანიზმების საფუძველზე, რომელთა საშუალებითაც სისტემა გამოაქვს დასკვნები, არამედ იმ ცოდნის ბაზების საშუალებით, რომლის საუძველზეც, ხდება დასკვნების გამოტანა. ეს კიდევ ერთხელ უსვავს ხაზს ცოდნის წარმოდგენის პრობლემას კომპიუტერში.

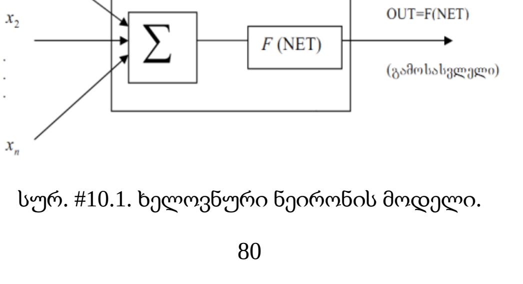
საკონტროლო კითხვები:

1. რატომაა გაყოფილი ცოდნის წარმოდგენისა და დასკვნების გამოტანის ქვესისტემები?
2. რა როლს ასრულებს განმარტებების ქვესისტემა?
3. რითი განსხვავდება ერთმანეთისაგან მიზანზე დაფუძნებული და მონაცემებზე დაფუძნებული ექსპერტული სისტემები?
4. კიდევ როგორ ექსპერტული სისტემები არსებობენ?
5. რა უფრო მნიშვნელოვანია ექსპერტულ სისტემაში, დასკვნების გამოტანის ქვესისტემა, თუ ცოდნის ბაზა?
6. როგორ უნდა იყოს ორგანიზებული ცოდნის ბაზა?

ლექცია 10. ნეირონული ქსელები

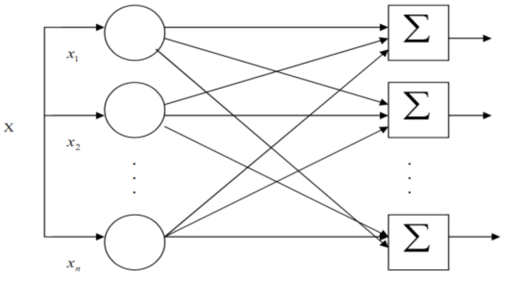
10.1. ნეირონული ქსელების არსი. როგორც უკვე აღნიშნული იყო პირველ ლექციაში, 1943 წელს უ. მაკ-კალოკმა და უ. პიტსმა გამოაქვეყნეს შრომა, რომელშიც აჩვენეს, რომ შესაძლებელია ადამიანის ნეირონული უჯრედის მოდელირება ტექნიკური სისტემის, ან კომპიუტერის საშუალებით, ხელოვნური ნეირონული ქსელების შექმნა და მათი საშუალებით ლოგიკური გამოთვლების შესრულება. შემოვიტანოთ ის ძირთადი ტერმინები, რომლებიც გამოიყენება ნეირონულ ქსელებში. ნეირონული ქსელების ძირითადი ელემენტია ხელოვნური ნეირონი, რომლის სქემაც მოცემულია სურ. #10.1. მას გააჩნია შემდეგი სტრუქტურა:

- **შესასვლელი სიგნალები** x_i - ესაა მონაცემები, რომლებიც მიეწოდება ნეირონს შესასვლელზე გარე სამყაროდან ან სხვა ნეირონებიდან. შესასვლელი ნეირონების დიაპაზონი სხვადასხვა მოდელებისათვის შეიძლება იყოს განსხვავებული. როგორც წესი ეს სიგნალები არიან დისკრეტული და ღებულობენ მნიშვნელობებს {0,1}, {-1,1} სიმრავლეებიდან ან **R**-დან.
- **წონითი კოეფიციენტები** w_i. წონითი კოეფიციენტები აიღება ნამდვილ რიცხვთა სიმ რავლიდან და განსაზღვრავენ კავშირის ძალას ნეირონებს შორის.
- ნეირონის აქტივაციის დონე. ესაა სიდიდე, რომელიც განსაზღვრება ნეირონში შემავალი $\sum w_i x_i$ სიგნალების შეწონილი ჯამით (ვექტორულად **WX**).
- ზღურბლოვანი ფუნქცია f. ამ ფუნქციის დანიშნულებაა აქტივაციის დონის რაიმე ზღურბლთან შედარების გზით გამოთვალოს გამოსასვლელი სიგნალის მნიშვნელობა. ზღურბლოვანი ფუნქცია განსაზღვრავს აქტიურია თუ არა ნეირონი.



სურ. #10.1. ხელოვნური ნეირონის მოდელი.

10.2. სხვადასხვა ტიპის ნეირონული ქსელები. იმისათვის, რომ ნეირონების საშუალებით შევასრულოთ სერიოზული გამოთვლები, საჭიროა მათი გაერთიანება ქსელებში. ყველაზე მარტივ ქსელს წარმოადგენს ნეირონების ჯგუფი, რომელიც გარეთიანებულია შრეში (იხ. სურ. #10.2).



სურ. # 10.2. ერთშრიანი ნეირონული ქსელი

ამ სურათზე ნეირონებს წარმოადგენენ კვადრატები, მათში მითითებული აჯამვის ნიშნით, ხოლო წრეები წარმოადგენენ მხოლოდ სიგნალების გამანაწილებლებს. ისინი არ ასრულებენ გამოთვლის არანაირ ფუნქციას და ამიტომ მათ არ უწოდებენ შრეს. თითოეული შესასვლელი სიგნალი თავისი წონით უკავშირდება ყველა ნეირონს. ეს წონები წარმოადგენენ წონების **W** მატრიცის ელემენტებს, რომელსაც აქვს *m* სტრიქონი და *n* სვეტი, სადაც *m* არის შესასვლელების რაოდენობა, ხოლო *n* – კი ნეირონების რიცხვი. ამგვარად, **N** გამოსასვლელი, რომლის კომპონენტებიცაა ნეირონების **OUT** გამოსასვლელები, გამოითვლება როგორც **XW=N** მატრიცული ნამრავლი.

შესაძლებელია უფრო რთული, მრავალშრიანი ნეირონული ქსელების შედგენაც, სადაც გვაქვს ორი, სამი და უფრო მეტი შრეები. აღმოჩნდა, რომ მრავალშრიან ქსელებს გააჩნიათ უფრო მეტი საშუალებები, ვიდრე ერთშრიან ქსელებს. აქ ყველაზე მთავარია, რომ ნებისმიერი კონფიგურაციის ქსელის შექმნისას დაცული უნდა იყოს შრისებური ორგანიზაცია, რაც ახდენს ადამიანის თავის ტვინის სხვადასხვა ნაწილების შრისებურ სტრუქტურების მოდელირებას. მრავალშრიან ქსელებში ერთი შრის გამოსასვლელები წარმოადგენენ მეორე შრის შესასვლელებს და თუ გვინდა, რომ ასეთ ქსელს გააჩნდეს უფრო მეტი გამოთვლების უნარი, საჭიროა რომ გამააქტივირებელი ფუნქცია იყოს არაწრფივი, რადგანაც, თუ ეს ფუნქცია იქნება წრფივი, მაშინ მეორე შრის გამოსასვლელზე ჩვენ მივიღებთ სიგნალს

$$OUT = (XW_1)W_2 = X(W_1W_2).$$

ცხადია, რომ ასეთივე სიგნალის მიღება შეგვეძლო ერთშრიანი ქსელით, რომლის კოეფიციენტების მატრიცაც იქნებოდა ორი მატრიცის ნამრავლი.

10.3. ხელოვნური ნეირონული ქსელების სწავლება. ყველაზე საინტერესო თვისება, რაც აღმოაჩნდათ ხელოვნურ ნეირონულ ქსელებს, არის მათი დასწავლის უნარი. მათი დასწავლის პროცესი ისე ძალიან წააგავს ადამიანის ინტელექტუალური განვითარების პროცესს, რომ ერთი შეხედვით შეიძლება მოგვეჩვენოს, რომ ჩვენ მართლაც ღრმად გვესმის ეს პროცესები, მაგრამ უნდა აღვნიშნოთ, რომ ხელოვნური ნეირონული ქსელების დასწავლის უნარი მაინც არის შეზღუდული და ჯერჯერობით უცნობია, ანალოგიურია თუ არა ეს პროცესები ადამიანის თავის ტვინში მიმდინარე პროცესებისა.

დასწავლის პროცესის ქვეშ ჩვენ გვესმის შემდეგი: მივაღწიოთ იმას, რომ შესასვლელი სიგნალების რაიმე სიმრავლისათვის ქსელი გამოსასვლელზე იძლეოდეს სასურველ (ან თუნდაც ამ სიმრავლისათვის შესაბამის) სიგნალებს. ყოველი შესასვლელი და გამოსასვლელი სიმრავლე განიხილება როგორც ვექტორი. დასწავლის პროცესი ხორციელდება შესასვლელი ვექტორების მიმდევრობითი შეყვანით ქსელში. ამასთან ერთდროულად

ხდება გარკვეული პროცედურის თანახმად წონების შერჩევა ისე, რომ გამოსასვლელზე მივიღოთ სასურველი ვექტორი. დასწავლის პროცესში წონები თანდათანობით ღებულობენ ისეთ მნიშვნელობებს, რომ ყოველი შესასვლელი ვექტორი გამოიმუშავებს გამოსასვლელ ვექტორს. დასწავლის პროცესი შეიძლება ხორციელდებოდეს მასწავლებლის საშუალებით ან მასწავლებლის გარეშე.

დასწავლა მასწავლებლის საშუალებით გულისხმობს, რომ თითოეული შესასვლელი ვექტორისათვის არსებობს შესაბამისი მიზნის ვექტორი, რომელიც გვინდა რომ მივიღოთ გამოსასვლელზე. ასეთ წყვილს ეწოდება დასწავლადი წყვილი. როგორც წესი, ქსელი სწავლობს რამდენიმე ასეთი წყვილის საშუალებით. ქსელს შესასვლელზე მიეწოდება შესასვლელი ვექტორი, გამოითვლება გამოსასვლელი ვექტორი და ხდება მისი შედარება მიზნის ვექტორთან. განსხვავება (რაც წარმოადგენს შეცდომას), უკუკავშირის საშუალებით მიეწოდება ქსელს და ხდება გარკვეული ალგორითმის საშუალებით წონების შეცვლა ისე, რომ მოხდეს შეცდომების მინიმიზაცია. ეს პროცედურა ხორციელდება თითოეული დასწავლადი წყვილისათვის მანამდის, სანამ არ მიიღწევა დასაშვები დონე შეცდომებისა მთელი დასასწავლი სიმრავლისათვის.

მიუხედავად იმისა, რომ დასწავლის ასეთი პრინციპი საშუალებას იძლევა გამოვიყენოთ ნეირონული ქსელები მრავალი პრაქტიკული ამოცანების გადასაჭრელად, მაინც ითვლება, რომ ასეთი მიდგომა არასწორია, რადგანაც შეუძლებელია წარმოვიდგინოთ, რომ დასწავლის ასეთი მექანიზმი არსებობდეს ადამიანის თავის ტვინში. დასწავლა მასწავლებლის გარეშე ალბათ იქნებოდა უფრო კარგი მოდელი თავის ტვინში მიმდინარე პროცესების აღსაწერად. ასეთი დასწავლის დროს საჭირო აღარ არის მიზნის ვექტორი და დასწავლა ხდება მხოლოდ შესასვლელი ვექტორების საშუალებით. ამასთან დასწავლის მიზანია გადაეწყოს წონები ისე, რომ ორი, საკმარისად ახლოს მდგომი შესასვლელი ვექტორისათვის გამოსავლელი ვექტორი იყოს ერთი და იგივე. დასწავლის პროცესი ამ შემთხვევაში გამოყოფს დასასწავლი სიმრავლის სტატისტიკურ მახასიათებლებს და აჯგუფებს მსგავს ვექტორებს ერთ კლასში. მოცემული კლასის ვექტორის მიწოდება შესასვლელზე მოგვცემს გარკვეულ ვექტორს გამოსასვლელზე, მაგრამ სანამ არ დამთავრდება დასწავლა, შეუძლებელია ვიცოდეთ, რა სახის იქნება ეს ვექტორი.

10.4. ერთშრიანი პერცეპტრონი. პირველი ნეირონული ქსელი, რომელსაც შეეძლო აღქმა (პერცეპცია) და გარკვეული რეაქციის ფორმირება ამ აღქმაზე იყო ფ. როზენბლატის პერცეპტრონი. ავტორი თვლიდა რომ ეს იყო თავის ტვინის მუშაობის მოდელი. ფ. როზენბლატი ამ მოდელს, რომელიც შეიქმნა კორნელის საავიაციო ლაბორატორიაში **MARK-1** სახელით, უწოდებდა სამშრიან პერცეპტრონს, მაგრამ დღევანდელი ტერმინოლოგიით ის წარმოადგენს ერთშრიან ნეირონულ ქსელს. პერცეპტრონის შემავალი ცვლადები (x_i) ღებულობენ მხოლოდ ორ მნიშვნელობას 0 ან 1, ხოლო წონები – მნიშვნელობებს ნამდვილ რიცხვთა სიმრავლიდან. აქტივაციის დონე გამოითვლება როგორც შემავალი მონაცემების შეწონილი ჯამი $\sum w_i x_i$. პერცეპტრონის გამომავალი მნიშვნელობა გამოითვლება მკაცრი ზღურბლური ფუნქციის საშუალებით: თუ აქტივაციის დონე მეტია ან ტოლი ზღურბლურ მნიშვნელობაზე, პერცეპტრონის გამოსასვლელი ტოლია 1-ს, წინააღმდეგ შემთხვევაში -1-ს. ამგვარად პერცეპტრონის გამომავალი მნიშვნელობები გამოითვლება შემდეგნაირად: **XW=1**, თუ $\sum w_i x_i \geq t$ და **XW=-1**, თუ $\sum w_i x_i < t$. პერცეპტრონის წონების რეგულირებისათვის გამოიყენება სწავლების მარტივი ალგორითმი მასწავლებლით, ანუ პერცეპტრონს შესასვლელზე მიეწოდება დასწავლადი წყვილის პირველი ელემენტი, პერცეპტრონი გამოთვლის გამოსასვლელ ვექტორს და ამის შემდეგ მას უჩვენებენ თუ რა შედეგი უნდა მიეღო. პერცეპტრონის წონებს ცვლიან ისეთნაირად, რომ შემცირდეს შეცდომები მის გამოსასვლელზე. ამ დროს გამოიყენება შემდეგი წესი: ვთქვათ c არის სწავლების სიჩქარის მუდმივი კოეფიციენტი, ხოლო d – გამოსასვლელის მოსალოდნელი მნიშვნელობა. შემავალი ვექტორის i-ური კომპონენტის წონითი კოეფიციენტის აგება სრულდება შემდეგი ფორმულით:

$$\Delta w_i = c(d - sign(\sum x_i w_i))x_i$$

სადაც $sign(\sum x_i w_i)$ არის პერცეპტრონის გამომავალი მნიშვნელობა, რომელიც ტოლია +1, ან -1. სხვაობა სასურველ და რეალურ გამომავალ მნიშვნელობებს შორის შეიძლება იყოს 0, 2 ან -2. შესაბამისად შემავალი ვექტორის ყოველი კომპონენტებისათვის სრულდება ასეთი მოქმედებები:

- თუ სასურველი და რეალური გამოსასვლელი ვექტორები ერთმანეთს ემთხვევა, წონები არ იცვლება;
- თუ რეალური გამოსასვლელი ვექტორის მნიშვნელობაა -1, ხოლო სასურველის კი 1, მაშინ i - ური შესასვლელის წონითი კოეფიციენტი იზრდება $2cx_i$ -ით;
- თუ რეალური გამოსასვლელი ვექტორის მნიშვნელობაა 1, ხოლო სასურველის კი -1, მაშინ i -ური შესასვლელის წონითი კოეფიციენტი მცირდება $2cx_i$ -ით;

დასწავლის ამ მეთოდს უწოდებენ დელტა წესს.

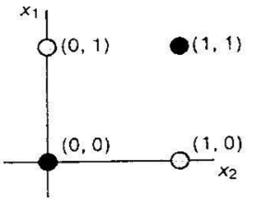
როგორც უკვე ვნახეთ, ქსელის შემავალი მონაცემები წარმოადგენენ n განზომილებიან ვექტორს და განსაზღვრავენ n განზომილებიან სივრცეს, ამიტომ კლასიფიკაციის სწავლების ამოცანა დაიყვანება ამ სივრცეში სხვადასხვა კლასის წერტილების ჰიპერსიბრტყეების საშუალებით რაიმე ჯგუფებად განცალკავების ამოცანაზე.

ფ. როზენბლატმა დაამტკიცა თეორემა პერცეპტრონის დასწავლადობის შესახებ, რომლის თანახმადაც პერცეპტრონს შეუძლია ისწავლოს ყველაფერი, რისი წარმოდგენაც მას შეუძლია. ამ დამტკიცების ქვეშ იგულისხმება, რომ დასწავლის პროცესი არის სასრული. აქ მთავარია გავარჩიოთ თუ რა განსხვავებაა წარმოდგენასა და დასწავლას შორის. წარმოდგენის ცნების ქვეშ იგულისხმება პერცეპტონის უნარი მოახდინოს გარკვეული ფუნქციის მოდელირება, დასწავლა კი ნიშნავს ქსელის წონების რეგულირების ისეთი სისტემატური პროცედურის არსებობას, რომელიც უზრუნველყოფს ამ ფუნქციის რეალიზაციას.

10.5. რა არ შეუძლია ერთშრიან პერცეპტრონს. ფ. როზენბლატმა დაამტკიცა, რომ პერცეპტრონს შეუძლია დაისწავლოს ყველაფერი, რისი წარმოდგენაც შესაძლებელია პერცეპტრონში. სამწუხაროდ, მოგვიანებით მ. მისკმა და ს. პეიპერტმა აჩვენეს, რომ არსებობს ძალიან ბევრი მარტივი ფუნქცია, რომლის წარმოდგენაც პერცეპტრონში შეუძლებელია, მაგალითად, ისეთი მარტივი ფუნქცია, როგორიცაა „გამომრიცხავი ან", ანუ **XOR** ფუნქცია. ამ ფუნქციას გააჩნია ორი შესასვლელი და ერთი გამოსასვლელი. ამასთან ეს ფუნქცია ღებულობს მნიშვნელობას ერთს, მხოლოდ მაშინ, როდესაც შესასვლელი მნიშვნელობები ღებულობენ სხვადასხვა მნიშვნელობებს.

განვიხილოთ პერცეპტრონი ორი, x_1 და x_2 შესასვლელით, ორი წონითი კოეფიციენტით w_1 და w_2 და ზღურბლით t. პერცეპტრონის მუშაობის პრინციპის თანახმად, ჩვენ მივიღებთ შემდეგ უტოლობებს:

$w_1*1+ w_2*1 < t$ (ჭეშმარიტობის ცხრილის პირველ სტრიქონიდან)
$w_1*1+ 0 > t$ (ჭეშმარიტობის ცხრილის მეორე სტრიქონიდან)
$0+ w_2*1 > t$ (ჭეშმარიტობის ცხრილის მესამე სტრიქონიდან)
$0+0<t$ (ჭეშმარიტების ცხრილის ბოლო სტრიქონიდან).



სურ. №10.3. "გამომრიცხავი ან" – ის პრობლემა, სიბრტყეზე შეუძლებლია ისეთი წრფის გატარება, რომელიც გაყოფს ერთმანეთისაგან თეთრ და შავ წერტილებს

ბოლო უტოლობიდან გამოდის, რომ ამ პერცეპტრონის ზღურბლი დადებითია, მაშინ უტოლობათა ამ სისტემას არ ექნება ამონახსენი, რადგანაც პირველი, მეორე და მესამე უტოლობები მოდიან წინააღმდეგობაში. ეს აიხსნება იმით, რომ პერცეპტრონმა წერტილების სიმრავლე $\{(0,0),(0,1),(1,0),(11)\}$ უნდა გაყოს ორ კლასად, $\{(0,0),(1,1)\}$ და $\{(0,1),(1,0\}$. როგორც სურ. #10.3-დან ჩანს, არ არსებობს წრფივი ფუნქცია, რომელიც ამას მოახერხებს.

საკონტროლო კითხვები:

1. რას წარმოადგენს ხელოვნური ნეირონი და რა ფუნქციების განხორციელება შეუძლია მას?
2. განმარტეთ ტერმინები: შესასვლელი სიგნალები, წონითი კოეფიციენტი, ნეირონის აქტივაციის დონე, ზღურბლოვანი ფუნქცია.
3. როგორ სისტემას ეწოდება ერთშრიანი ნეირონული ქსელი?
4. რა განსაკუთრებული თვისება აღმოაჩნდათ ნეირონულ ქსელებს?
5. რას ნიშნავს დასწავლა მასწავლებლით? მასწავლებლის გარეშე?
6. რა არის პრეცეპტონი?
7. ჩამოაყალიბეთ თქვენი სიტყვებით სახეთა კლასიფიკაციის ამოცანა.

ლექცია 11. ნეირონული ქსელები (გაგრძელება)

11.1. პერცეპტრონის გამოყენება სახეთა კლასიფიკაციის ამოცანისათვის. სახეთა კლასიფიკაციის ამოცანა უმარტივეს შემთხვევაში შეიძლება აღვწეროთ შემდეგნაირად: დავუშვათ, მოცემული გვაქვს ობიექტების სიმრავლე, რომელიც გვინდა გავყოთ ორ კლასად. მაგალითად, გვინდა განვასხვაოთ ერთმანეთისაგან სამკუთხედები და ოთხკუთხედები. იმისათვის, რომ შესაძლებელი იყოს ნეირონულ ქსელს (პერცეპტრონს) ვასწავლოთ ამ ორი კლასის ერთმანეთისაგან გარჩევა, პირველ რიგში საჭიროა გვქონდეს შემფასებელი ფუნქცია, რომელიც მოგვცემს საშუალებას მივაკუთვნოთ ეს ობიექტები ამა თუ იმ კლასს. კლასიფიკატორის მიერ გამოვლენილი მონაცემთა ყოველი ჯგუფი შეიძლება წარმოდგენილი იყოს რაიმე არეს საშუალებით მრავალგანზომილებიან სივრცეში. ყოველ R_i კლასს შეესაბამება ამ არესთან კუთვნილების ხარისხის განმსაზღვრელი g_i დისკრიმინანტული ფუნქცია. ყველა ასეთ g_i დისკრიმინანტულ ფუნქციებს შორის მაქსიმალური მნიშვნელობა R_i არესათვის აქვს ისეთ g_i ფუნქციას, რომლისთვისაც სრულდება პირობა:

$$g_i(x) > g_j(x)$$

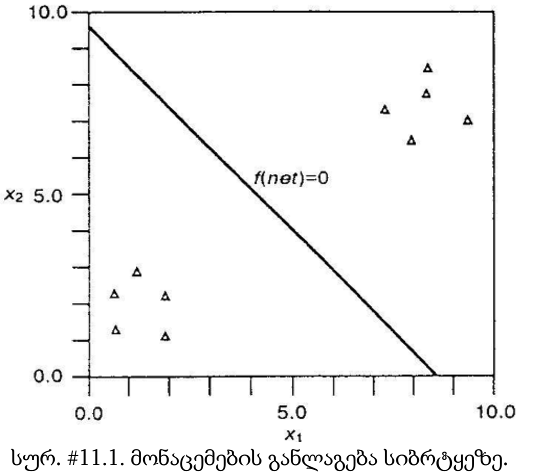
ყოველი $1 < j < n$. ყველაზე მარტივ შემთხვევაში ამ ფუნქციის მნიშვნელობები შეიძლება იყოს {-1,1} სიმრავლე, ანუ ობიექტი ეკუთვნის მოცემულ სიმრავლეს (1) და ობიექტი არ ეკუთვის მოცემულ სიმრავლეს (-1). უფრო მნიშვნელოვანია შემთხვევა, როდესაც დისკრიმინანტული ფუნქცია წარმოადგენს მიკუთვნების შეფასებას არეს რაიმე ცენტრალური წერტილიდან მანძილის საფუძველზე. ასეთ შემთხვევაში კლასიფიკაციას უწოდებენ კლასიფიკაციას მინიმალური მანძილით. თუ ორი g_i და g_j კლასები წარმოადგენენ მოსაზღვრე კლასებს, მაშინ არსებობს საზღვარი, რომლისთვისაც

$$g_i(x) - g_j(x) = 0.$$

განვიხილოთ მაგალითი, დავუშვათ, სიმრავლე, რომლის საფუძველზეც უნდა ვასწავლოთ პერცეპტრონს სახეთა გამოცნობა წარმოადგენს სიმრავლეს, რომელიც მოცემულია შემდეგი ცხრილით:

x_1	x_2	$f(NET)$
1,0	1,0	1
9,4	6,4	-1
2,5	2,1	1
8,0	7,7	-1
0,5	2,2	1
7,9	8,4	-1
7,0	7,0	-1
2,8	0,8	1
1,2	3,0	1
7,8	6,1	-1

ცხრილის პირველი ორი სვეტი შეიცავს მონაცემთა ვექტორებს, რომლებიც გამოიყენებიან ქსელის სწავლებისათვის. მესამე სვეტში წარმოდგენილია კლასიფიკაციის მოსალოდნელი შედეგები +1 ან -1, რომლებიც გამოიყენებიან ქსელის სწავლებისათვის. ამ მონაცემებისთვის სურ. #11.1. მოყვანილია მათი განლაგება სიბრტყეზე და მათ შორის წრფე, რომელიც წარმოადგენს საზღვარს ამ ორ სიმრავლეს შორის.



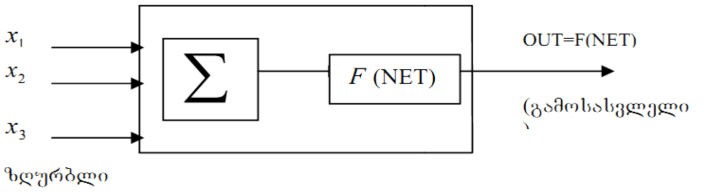
სურ. #11.1. მონაცემების განლაგება სიბრტყეზე.

თუ შესაძლებელია კლასების წრფივად გაყოფა, როგორც ეს ხდება სურ. #11.1, მაშინ ორივე კლასის დისკრიმინანტული ფუნქციები $g_i(x)$ და $g_j(x)$ წარმოადგენენ წრფეს. რადგანაც წრფე არის ისეთი წერტილების ერთობლიობა, რომლებიც თანაბარი მანძილებით არიან დაშორებული ორი ფიქსირებული წერტილიდან, გამოდის, რომ g_i და g_j ფუნქციები წარმოადგენენ თითოეული არეს დეკარტული ცენტრიდან მინიმალური მანძილის ფუნქციებს.

წრფივი გამყოფი ფუნქცია შეიძლება გამოვთვალოთ პერცეპტრონის საშუალებით რომელიც მოცემულია სურ. #11.2. იგი შეიცავს ორ ჩვეულებრივ შესავალს და ერთ ზღურბლურ შესავალს მუდმივი მნიშვნელობით 1. პერცეპტრონი ასრულებს გამოთვლებს შემდეგი ფორმულით:

$$f(NET) = f(w_1 x_1 + w_2 x_2 + w_3 1)$$

სადაც $f(x) = sign(x)$.თუ $f(x) = 1$ მაშინ x ობიექტი ეკუთვნის ერთ კლასს და თუ $f(x) = -1$, მაშინ x ობიექტი ეკუთვნის სხვა კლასს. ასეთ გარდაქმნას ეწოდება წყვეტილ-წრფივი ბიპოლარული ზღურბლური ფუნქცია.



სურ. #11.2. პერცეპტრონი, რომელიც ხსნის სახეთა კლასიფიკაციის მაგალითს.

ზღურბლური შესასვლელი ემსახურება ზღურბლური ფუნქციის გადაადგილებას ვერტიკალური ღერძის გასწვრივ. ამ გადაადგილების სიდიდე განისაზღვრება სწავლების პროცესში w_3 წონითი კოეფიციენტის დახმარებით. დავიწყოთ პერცეპტრონის სწავლება. ამისათვის შევარჩიოთ წონების საწყისი მნიშვნელობები შემთხვევითად, მაგალითად [0,75; 0,5; -0,6]. იმისათვის, რომ გავარჩიოთ, რომელი მონაცემებისთვის ვითვლით ფუნქციის მნიშვნელობას, დავუმატოთ ფუნქციას ზედა ინდექსი $f(NET)^n$. დავიწყოთ ცხრილის პირველი სტრიქონიდან.

$$f(NET)^1 = f(0{,}75 \cdot 1 + 0{,}5 \cdot 1 - 0{,}6 \cdot 1) = f(0{,}65) = 1.$$

რადგან ფუნქციის მნიშვნელობა მივიღეთ ერთის ტოლი და ის ემთხვევა ცხრილის მნიშვნელობას, ამიტომ წონითი კოეფიციენტების აგება არ არის საჭირო. შესაბამისად მეორე წყვილისთვისაც გამოვიყენებთ იგივე წონით კოეფიციენტებს.

$$f(NET)^2 = f(0{,}75 \cdot 9{,}4 + 0{,}5 \cdot 6{,}4 - 0{,}6 \cdot 1) = f(9{,}65) = 1$$

მოცემული წყვილისათვის გამოსასვლეზე უნდა მიგვეღო -1,ამიტომ საჭიროა წონითი კოეფიციენტების შეცვლა ზემოთმოყვანილი წესის თანახმად

$$W^t = W^{t-1} + c(d^{t-1} - \mathrm{sgn}((W^{t-1})^T X^{t-1}))X^{t-1}$$

სადაც c არის დასწავლის მუდმივი კოეფიციენტი, X არის შესასვლელის და W - და წონითი კოეფიციენტების ვექტორები t – სწავლების იტერაციის ნომერი, T - ტრანსპონირების სიმბოლო, d^{t-1} მოსალოდნელი შედეგი დროის $(t-1)$ მომენტში (ამ შემთხვევაში $t=2$). $t=2$ მნიშვნელობის დროს ქსელის გამოსასვლელი ტოლია 1-ის, ნაცვლად -1-ისა, ამიტომ შემდეგი წყვილისათვის წონითი კოეფიციენტები შეიცვლება და მივიღებთ

$$W^3 = W^2 + 0{,}2(-1-1)X^2 = \begin{bmatrix} 0{,}75 \\ 0{,}5 \\ -0{,}6 \end{bmatrix} - 0{,}4 \begin{bmatrix} 9{,}4 \\ 6{,}4 \\ 1 \end{bmatrix} = \begin{bmatrix} -3{,}01 \\ -2{,}06 \\ -1 \end{bmatrix}$$

ახლა საჭიროა გამოვთვალოთ ქსელის გამომავალი მნიშვნელობები სწავლების მესამე მაგალითისათვის აგებული წონების გათვალისწინებით. მივიღებთ, რომ $f(NET)^3 = 1$.

ეს მნიშვნელობა ისევ არ ემთხვევა მოსალოდნელ გამოსასვლელს. შემდეგი იტერაციის დროს კვლავ მოხდება წონების განახლება იგივე ფორმულით. პერცეპტრონის ქსელს სწავლების ათი იტერაციის შემდეგ ფორმირდება წრფივი გაყოფადობა, რომელიც მოცემულია სურ. #11.1. დაახლოებით სწავლების 500 ბიჯის შემდეგ წონითი კოეფიციენტების ვექტორი მიიღებს მნიშვნელობას: [-1,3; -1,1; 10,9]. ჩვენ გვაინტერესებს ორი კლასის წრფივი გაყოფა. $g_i(x)$ და $g_j(x)$ დისკრიმინანტული ფუნქციებს ტერმინებში გამყოფი ზედაპირი წარმოადგენს წერტილთა სიმრავლეს, რომელთათვისაც $|g_i(x) - g_j(x)| = 0$, ე.ი. აქტივაციის დონე ტოლია 0-ის. ქსელის ფუნქციონირების განტოლება შეიძლება ჩავწეროთ წონების ტერმინებით, შემდეგი სახით: $NET = w_1 \cdot x_1 + w_2 \cdot x_2 + w_3$. მაშინ, ჩვენს მაგალითში ორ კლასს შორის წრფივი გამყოფი ზედაპირი განისაზღვრება წრფივი განტოლებით:.

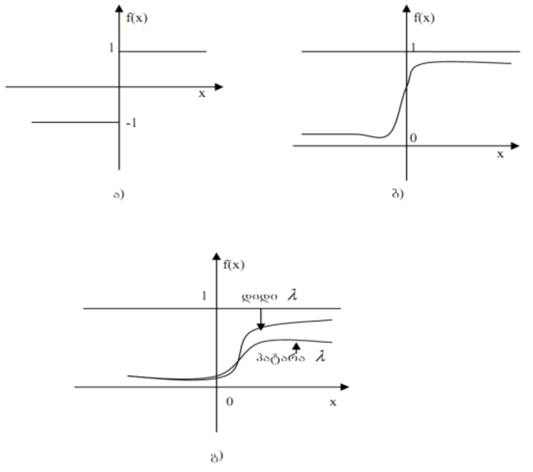
$$-1{,}3 \cdot x_1 + (-1{,}1) \cdot x_2 + 10{,}9 = 0 \quad 1{,}1)$$

11.2. დელტა –წესის განზოგადება. იმისათვის, რომ განვაზოგადოთ პერცეპტრონის იდეა და ავაგოთ დასწავლის უფრო დახვეწილი ალგორითმები, საჭიროა შევცვალოთ მისი ზღურბლოვანი ფუნქცია და გავხადოთ ის უფრო მოქნილი და უწყვეტი ფუნქცია, რომელიც საშუალებას მოგვცემს უფრო ზუსტად გამოვთვალოთ წონები. ერთერთ ასეთ ფუნქციას წარმოადგენს სიგმოიდური ფუნქცია, რომელმაც თავისი სახელი მიიღო ლათინურ **S**-თან მსგავსების გამო.

ტიპური სიგმოიდური აქტივაციური ფუნქცია, ანუ ლოგისტიკული ფუნქცია მოიცემა განტოლებით:

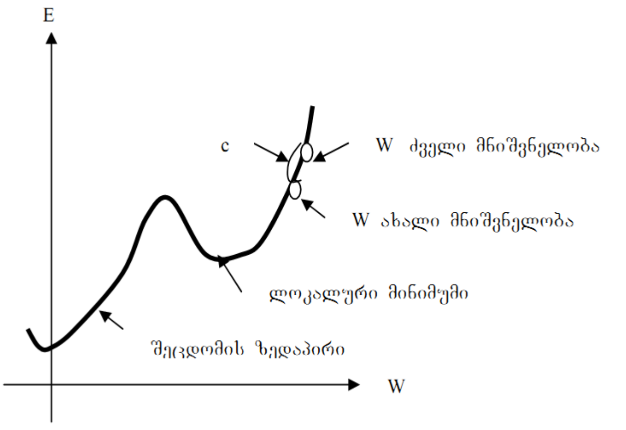
$$f(NET) = 1/(1+e^{-\lambda \cdot net})$$

სადაც $NET = \sum w_i x_i$. აქ λ არის სიმრუდის (ამოზნექილობის პარამეტრი), რომელიც გამოიყენება სიგმოიდური წირის ფორმის აგებისათვის. λ-ს დიდი მნიშვნელობებისთვის სიგმოიდას ფორმა უახლოვდება ჩვეულებრივ წყვეტილ-წრფივ ზღურბლურ ფუნქციას, რომლის მნიშვნელობები მოთავსებულია {0,1} დიაპაზონში, ხოლო სიმრუდის პარამეტრის იმ მნიშვნელობებისათვის, რომლებიც ახლოს არიან 1-თან, სიგმოიდური წირი ჰგავს სწორ ხაზს. ზღურბლური ფუნქციის არგუმენტს წარმოადგენს ნეირონის აქტივაციის დონე, ხოლო მისი მნიშვნელობაა ნეირონის გამოსასვლელი. სიგმოიდური აქტივაციური ფუნქცია არის უწყვეტი რაც, იძლევა ქსელის გამოსასვლელზე შეცდომების უფრო ზუსტად შეფასების საშუალებას. ჩვეულებრივი ზღურბლური ფუნქციის მსგავსად, სიგმოიდური აქტივაციური ფუნქცია ასახავს განსაზღვრის არის წერტილებს (0,1) ინტერვალში მოთავსებულ მნიშვნელობებში. თუმცა უნდა ითქვას ისიც, რომ ჩვეულებრივი ზღურბლური ფუნქციისგან განსხვავებით სიგმოიდას შეუძლია მიიღოს მნიშვნელობები მთელი ინტერვალიდან. ე.ი. ის უზრუნველყოფს კლასიკური ზღურბლური ფუნქციის უწყვეტ აპროქსიმაციას. λ პარამეტრი განსაზღვრავს გადასვლის სიმრუდეს.



სურ. #11.3. აქტივაციური ფუნქცები:ა) წყვეტილ- წრფივი ბიპოლარული ფუნქცია; ბ) სიგმოიდური უნიპოლარული ზღურბლოვანი ფუნქცია; გ) წაძრული სიგმოიდური ფუნქცია სხვადასხვა სიმრუდით. რაც მეტია λ პარამეტრი, მით უფრო უახლოვდება ფუნქცია წყვეტილ ფუნქციას.

ზღურბლის შეწონილი მნიშვნელობა გადაადგილებს ფუნქციის გრაფიკს აბსცისათა ღერძის გასწვრივ.



სურ. #11.4. შეცდომის ორგანზომილებიანი სიბრტყე. c კონსტანტა განსაზღვრავს დასწავლის ბიჯის ზომას.

უწყვეტი აქტივაციური ფუნქციების მქონე ქსელების გამოჩენამ ისტორიულად განაპირობა შეცდომების კორექციაზე დაფუძნებული სწავლების ახალი მიდგომები. მაგალითად, ვიდროუ-ჰოფის დასწავლის წესი გვთავაზობს საშუალო კვადრატული შეცდომების მინიმიზაციას გამომავალ მნიშვნელობასა და ქსელის აქტივაციის დონეს შორის იმის მიუხედავად, თუ როგორი სახე აქვს აქტივაციურ ფუნქციას, მაგრამ უწყვეტი აქტივაციური ფუნქციების მქონე ქსელებისათვის სწავლების ყველაზე მნიშვნელოვან წესს წარმოადგენს დელტა-წესი. ინტიუციურად დელტა-წესი ეფუძნება ისეთ ცნებას, როგორიცაა შეცდომის ზედაპირი (სურ. #11.4.), რომელიც მონაცემთა მთელი ნაკადისათვის ჯამურ შეცდომას განსაზღვრავს როგორც ფუნქციას დამოკიდებულს ქსელის წონებზე. წონების ყოველი შესაძლო კონფიგურაცია განსაზღვრავს შეცდომის ზედაპირის წერტილს. გვაქვს რა წონების გარკვეული კონფიგურაცია, სწავლების ალგორითმის საშუალებით შეიძლება ვიპოვოთ ამ ზედაპირზე მიმართულება, რომლის გასწვრივაც ყველაზე სწრაფად ხდება შეცდომის ფუნქციის შემცირება. ამ მიდგომას ეწოდება სწავლება გრადიენტული დაშვების მეთოდით, რადგანაც გრადიენტი განსაზღვრავს ზედაპირის დახრილობას მის ყოველ წერტილში.

დელტა-წესი გვთავაზობს უწყვეტი და დიფერენცირებადი აქტივაციური ფუნქციის გამოყენებას. ეს თვისებები გააჩნია ზემოდ განხილულ ლოგისტიკურ ფუნქციას. დელტა-წესს ქსელის i-ური კვანძის j-ური წონითი კოეფიციენტის რეგულირებისათვის აქვს შემდეგი სახე:

$$c(d_i - O_i)f'(net_i)x_j$$

სადაც c არის სწავლების სიჩქარის მუდმივი კოეფიციენტი, d_i-ური და O_i-ური i-ური ნეირონის მოსალოდნელი და რეალური გამოსასვლელები, f' i-ური კვანძის აქტივაციური ფუნქციის წარმოებული, ხოლო x_j i-ური კვანძის j-ური შესასვლელის მნიშვნელობა. ვაჩვენოთ თუ საიდან გამოდის ეს ფორმულა. ქსელის გამოსასვლელზე საშუალოკვადრატული შეცდომა პროპორციულია ყოველი კვანძისათვის შეცდომების კვადრატების ჯამისა და გამოითვლება ფორმულით

$$E = 1/2\sum_i (d_i - O_i)^2$$

სადაც d_i არის ნეირონის გამოსასვლელს მოსალოდნელი მნიშვნელობა, ხოლო O_i – მისი რეალური მნიშვნელობა. აჯამვის დროს ხდება შეცდომების კვადრატში აყვანა, იმისათვის, რომ დადებითმა და უარყოფითმა შეცდომებმა არ მოახდინონ ერთმანეთის კომპენსირება.

განვიხილოთ შემთხვევა, როცა ქსელი შედგება ნეირონის ერთადერთი (გამოსასვლელი) შრისაგან. თავდაპირველად აუცილებელია განვსაზღვროთ ქსელის ყოველ გამოსასვლელ ნეირონზე შეცდომის ცვლილების სიჩქარე. ამისათვის უნდა ავიღოთ კერძო წარმოებული, რომელიც განსაზღვრავს ვექტორული ცვლილების სიჩქარეს, მისი ერთ-ერთი ცვლადის მიხედვით. ზოგადი შეცდომის კერძო წარმოებული გამომავალი შრის ყოველი i -ური ნეირონისათვის იქნება

$$\frac{\partial E}{\partial O_i} = \frac{\partial 1/2\sum_i (d_i - O_i)^2}{\partial O_i} = \frac{\partial 1/2(d_i - O_i)}{\partial O_i} = -(d_i - O_i).$$

ასეთი გამარტივება შესაძლებელია იმიტომ, რომ განსახილველი ქსელი შეიცავს ნეირონების მხოლოდ ერთ გამომავალ შრეს და აქედან გამომდინარე შეცდომა ერთერთი კვანძის გამოსასვლელზე ვერ მოახდენს გავლენას სხვა ნეირონებზე. განვსაზღვროთ შეცდომის ცვლილების სიჩქარე i -ური კვანძის ყოველი w_k წონითი კოეფიციენტისათვის. ამისათვის გამოვთვალოთ შეცდომის ფუნქციის კერძო წარმოებული ყოველი კვანძისათვის, w_k წონითი კოეფიციენტების მიხედვით, რომელსაც მივყავართ ამ კვანძამდე.

$$\frac{\partial E}{\partial w_k} = \frac{\partial E}{\partial O_i} \cdot \frac{\partial O_i}{\partial w_k} \quad \text{ანუ} \quad \frac{\partial E}{\partial w_k} = -(d_i - O_i)\frac{\partial O_i}{\partial w_k}.$$

განვიხილოთ ამ გამოსახულების მარჯვენა მხარე, სადაც გამოითვლება i -ური ნეირონის რეალური გამომავალის კერძო წარმოებული კავშირების იმ წონითი კოეფიციენტების მიხედვით, რომლებსაც მივყავართ ამ ნეირონთან. ნეირონის გამოსასვლელი მნიშვნელობა წონითი კოეფიციენტების საშუალებით გამოისახება შემდეგი სახით:

$$O_i = f(W_i X_i), \text{ სადაც } W_i X_i = net_i$$

რადგან f ამ შემთხვევაში უწყვეტი ფუნქციაა, შეიძლება გამოვთვალოთ მისი წარმოებული.

$$\frac{\partial O_i}{\partial w_k} = x_k \cdot f'(net_i) \quad \text{მაშინ} \quad \frac{\partial E}{\partial w_k} = -(d_i - O_i) f'(net_i) x_k$$

შეცდომის მინიმიზაციისათვის აუცილებელია შევცვალოთ წონის მნიშვნელობა ანტი-გრადიენტის მიმართულებით, საიდანაც მივიღებთ, რომ

$$\Delta w_k = -c\frac{\partial E}{\partial w_k} = -c[-(d_i - O_i) f'(net_i) x_k] = c(d_i - O_i) f'(net_i) x_k$$

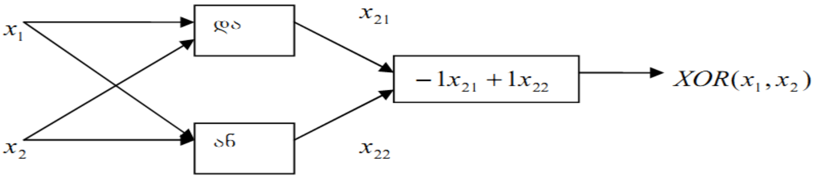
უნდა აღინიშნოს, რომ ისევე როგორც ლოკალური პიკის ძებნის ალგორითმი, დელტა წესსაც არ შეუძლია გაარჩიოს ლოკალური მინიმუმი გლობალურისაგან, რაც იწვევს ამ წესის არამდგრადობას. დელტა წესის ეფექტურობაზე არსებით გავლენას ახდენს აგრეთვე სწავლების სიჩქარის c კოეფიციენტი. (იხ. სურ. #10.7.). ამ კოეფიციენტის მნიშვნელობა განსაზღვრავს სწავლების ყოველ ბიჯზე წონების ცვლილების სიდიდეს. რაც უფრო დიდია c, მით უფრო ჩქარა მიდის წონა ოპტიმალურ მნიშვნელობამდე. თუმცა თუ c კოეფიციენტი ძალიან დიდია, ალგორითმი შეიძლება გასცდეს ექტრემუმის წერტილს ან დაიწყოს რხევა მის ირგვლივ. სწავლების სიჩქარის კოეფიციენტის მცირე მნიშვნელობები საშუალებას იძლევიან აღმოვფხრათ ეს პრობლემა, მაგრამ ანელებენ სწავლების პროცესს.

საკონტროლო კითხვები

1. როგორ შეიძლება გამოვიყენოთ პერცეპტრონი სახეთა კლასიფიკაციის ამოცანებისთვის?
2. როგორი ტიპის შეიძლება იყოს ზღურბლოვანი ფუნქცია?
3. რატომ დაგვჭირდა უწყვეტი ზღურბლოვანი ფუნქცია?
4. როგორ განზოგადდება დელტა წესი უწყვეტი აქტივაციის ფინქციის დროს?
5. რა მნიშვნელობა ენიჭება დასწავლის კოეფიციენტს ამ შემთხვევაში?
6. რატომაა ადვილი დელტა წესის გამოყენება ერთშრიანი ნეირონული ქსელისთვის?
7. რა არ შეუძლია ერთშრიან პერცეპტრონს?

ლექცია 12. მრავალშრიანი ნეირონული ქსელები

12.1. მრავალშრიანი ნეირონული ქსელი. წრფივი გაყოფადობის პრობლემის გამოვლენის შემდეგ აშკარა გახდა, რომ ერთშრიანი პერცეპტრონები ამ პრობლემის გადასაწყვეტად ვერ გამოდგებოდა. ამიტომ გაჩნდა იდეა მრავალშრიანი ქსელების. რა შესაძლებლობები ემატება პერცეპტრონს, თუ ის შეიცავს ერთ შრეზე მეტ შრეს? განვიხილოთ ორშრიანი ქსელი, რომელიც შედგება სამი ნეირონისგან (იხ. სურ. #12.1).



სურ. #12.1. უმარტივესი ორშრიანი ნეირონული ქსელი, რომელსაც შეუძლია "გამომრიცხავი ან" ოპერაციის განხორციელება.

აქ პირველ შრეში განლაგებულია ორი ნეირონი, რომლებიც ახდენენ "და" და "ან" ლოგიკური ოპერაციების განხორციელებას, ხოლო მეორე შრის ერთადერთი ელემენტი ასრულებს "გამომრიცხავი ან" ოპერაციას. ეს მაგალითი გვიჩვენებს, რომ მრავალშრიანი ქსელების საშუალებით შესაძლებელია უფრო რთული პროცესების მოდელირება და სახეთა კლასიფიკაციის უფრო საინტერესო შემთხვევების გადაჭრა.

12.2. პირდაპირი გავრცელების ნეირონული ქსელები. ნეირონულ ქსელს, რომელშიც არსებობს რამდენიმე შრე და d შრის ყველა ელემენტი იღებს შესასვლელ სიგნალებს $(d-1)$ შრის ყველა ნეირონისაგან და გადასცემს გამოსასვლელ სიგნალებს $(d+1)$ შრის ყველა ელემენტს, უწოდებენ პირდაპირი გავრცელების ნეირონულ ქსელებს. ჩვენი მიზანია ვაჩვენოთ, რომ შესაძლებელია ისეთი მრავალშრიანი ნეირონული ქსელების აგება, რომელთა გამოთვლითი შესძლებლობები ტოლია ტიურინგის მანქანის და გაფართოებული დელტა წესი წარმოადგენს დასწავლის ალგორითმს ასეთი ქსელებისათვის.

სიგნალის მრავალშრიანი დამუშავების დროს, შეცდომა ქსელის გამოსასვლელზე შეიძლება გამოწვეული იყოს მის შიგნით მიმდინარე რთული პროცესებით. ამიტომ ქსელის გამოსასვლელ შრეში შეცდომის წყარო უნდა გაანალიზდეს კომპლექსურად. შეცდომის უკუგავრცელების მეთოდი, რომელსაც ეხლა ჩვენ განვიხილავთ, უზრუნველყოფს წონების აგების საშუალებას მრავალშრიანი ნეირონული ქსელის სტრუქტურის გათვალისწინებით.

ამ მიდგომის თანახმად, შეცდომა ქსელის გამოსასვლელზე ვრცელდება უკუმიმართულებით გამოსასვლელი შრიდან დაფარული შრეებისკენ. დაფარული ეწოდება ყველა შრეს, რომლებიც მოთავსებულნი არიან შესასვლელსა და გამოსასვლელ შრეს შორის. სწავლების დელტა-წესით ანალიზისას არ არის რთული იმის დანახვა, რომ მთელი ინფორმაცია, რომელიც აუცილებელია ნეირონის წონების მოდიფიკაციისათვის, გარდა თვით შეცდომის სიდიდისა, ეკუთვნის მხოლოდ ამ ნეირონს. გამოსასვლელი შრის ნეირონებისთვის შეცდომის სიდიდე გამოითვლება მარტივად, როგორც სხვაობა მოსალოდნელ და რთულ გამოსასვლელ მნიშვნელობებს შორის. დაფარული შრეების კვანძებისათვის კი შეცდომის განსაზღვრა მნიშვნელოვნად უფრო რთულია. უკუგავრცელების მეთოდის გამოყენების დროს აქტივაციურ ფუნქციად როგორც წესი ირჩევენ ლოგისტიკურ ფუნქციას.

ამ ფუნქციას აქვს შემდეგი სახე:

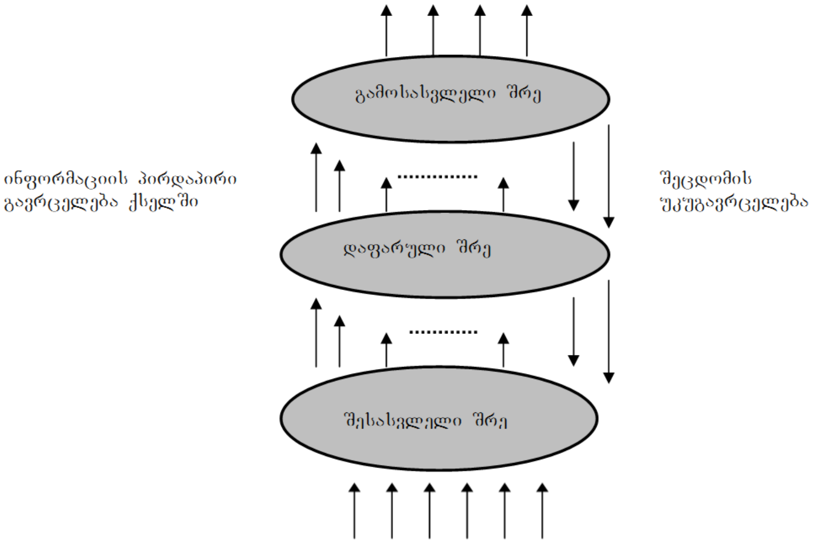
$$f(NET) = 1/(1+e^{-\lambda \cdot net})$$

სადაც $NET = \sum x_i w_i$.

ამ ფუნქციის გამოყენება განპირობებულია სამი მიზეზით:

1. იგი ეკუთვნის სიგმოიდური ფუნქციების კლასს.
2. რადგან ფუნქცია უწყვეტია, ამიტომ მისი წარმოებული არსებობს ყოველ წერტილში.
3. ვინაიდან წარმოებულის მაქსიმალური მნიშვნელობა შეესაბამება ფუნქციის ყველაზე უფრო სწრაფი ცვლილების სეგმენტს, ამიტომ უდიდესი შეცდომა შეესაბამება იმ კვანძებს, რომელთათვისაც აქტივაციის დონე ყველაზე ნაკლებად არის განსაზღვრული და ბოლოს, ამ ფუნქციის წარმოებული ადვილად გამოითვლება

$$f'(NET) = (1/(1+e^{-\lambda net}))' = \lambda(f(NET)\cdot(1-f(NET))).$$



სურ. #12.2. დაფარულშრიანი ნეირონულ ქსელში შეცდომის უკუგავრცელება.

12.3. შეცდომის უკუგავრცელების მეთოდი. უკუგავრცელების მეთოდი წარმოადგენს დელტა-წესის განზოგადოებას. აქაც გამოიყენება წინა ლექციებში აღწერილი გრადიენტული დაშვების მეთოდი. დაფარული შრის კვანძებისათვის შეცდომა გამოითვლება გამომავალი შრის შეცდომის საფუძველზე. შეცდომების უკუგავრცელების მეთოდის გამოყენების შემთხვევაში k და i ნეირონებს შორის კავშირის w_{ki} წონითი კოეფიციენტის ცვლილება შესასვლელი შრის კვანძებისათვის გამოითვლება ფორმულით:

$$\Delta w_{ki} = -\lambda c(d_i - O_i)O_i(1-O_i)x_k$$

და დაფარული შრეების კვანძებისათვის კი -

$$\Delta w_{ki} = -\lambda c O_i(1-O_i)\sum_j(-delta_j w_{ij})x_k$$

მეორე ფორმულაში j არის შემდეგი შრის კვანძის ინდექსი, სადამდისაც ვრცელდება სიგნალი i - ური ნეირონიდან და

$$delta_j = -\frac{\partial E}{\partial net_j} = (d_i - O_i)O_i(1-O_i).$$

გამოვთვალოთ ამ ფუნქციის წარმოებული. ჯერ ავიღოთ პირველი ფორმულის წარმოებული. ის ჩვენ უკვე გვქონდა, როდესაც ვითვლიდით დელტა წესს.

$$\frac{\partial E}{\partial w_k} = -(d_i - O_i)f'(net_i)x_k$$

როგორც უკვე ვნახეთ,

$$f'(net) = \lambda f(net)(1 - f(net)) .$$

გავიხსენოთ, რომ $f(net_i) = O_i$,

მივიღებთ

$$\frac{\partial E}{\partial w_k} = -\lambda(d_i - O_i)O_i(d_i - O_i)x_k$$

რადგან შეცდომის მინიმიზაციისათვის წონები უნდა იცვლებოდეს გრადიენტის საწინააღმდეგო მიმართულებით, მაშინ i-ური ნეირონისათვის წონითი კოეფიციენტის ცვლილების გამოთვლისას ეს წარმოებული უნდა გავამრავლოთ $-c$-ზე:

$$\Delta w_k = -c(d_i - O_i)O_i(1 - O_i)x_k$$

გამოვიყვანოთ დაფარული შრეებისათვის წონების აგების ფორმულა. სიმარტივისათვის დასაწყისში დავუშვათ, რომ ქსელი შეიცავს მხოლოდ ერთ დაფარულ შრეს. ავიღოთ ერთი i-ური ნეირონი დაფარული შრიდან და გავაანალიზოთ მისი წვლილი საერთო შეცდომაში ქსელის გამოსასვლელზე. ამისათვის განვიხილოთ i-ური ნეირონის წვლილი გამომავალი შრის j-ური ნეირონის გამოსასვლელზე არსებულ შეცდომაში, ხოლო შემდეგ შევკრიბოთ ეს წვლილები გამომავალი შრის ყველა ელემენტისათვის და ბოლოს გავითვალისწინოთ i-ურ ნეირონთან კავშირის k-ური წონითი კოეფიციენტის მთლიანი შეცდომა. ეს სიტუაცია აღწერილია სურ. #12.3.

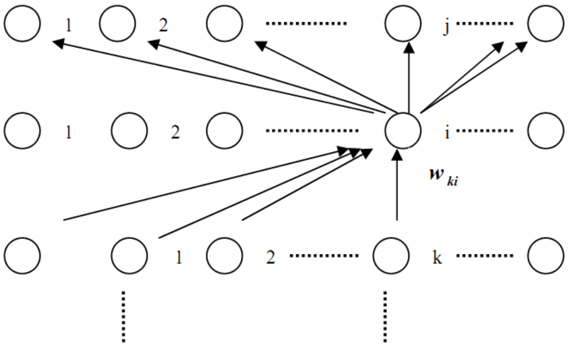
ჯერ განვიხილოთ ქსელის შეცდომის კერძო წარმოებული დაფარული შრის შეცდომაში i-ური ნეირონის გამოსასვლელის მიხედვით, ამისათვის შეგვიძლია ვისარგებლოთ იგივეობით:

$$\frac{\partial E}{\partial O_i} = \frac{\partial E}{\partial net_j} \cdot \frac{\partial net_j}{\partial O_i}$$

ამ დამოკიდებულების მარჯვენა მხარის პირველი მამრავლი, აღებული საწინააღმდეგო ნიშნით, აღვნიშნოთ $delta_j$. მაშინ გვექნება, რომ

$$\frac{\partial E}{\partial O_i} = -delta_j \cdot \frac{\partial net_j}{\partial O_i} .$$

გავიხსენოთ, რომ net_j, გამოსასვლელი შრის j-ური ნეირონის აქტივაციის დონე გამოითვლება როგორც დაფარული შრის წონითი კოეფიციენტებისა და გამოსასვლელების ნამრავლის ჯამი $net_j = \sum_i w_{ij} O$.



სურ. №12.3. i-ური ნეირონის $-\sum deltaa_j \cdot w_{ij}$ მთლიანი წვლილიშეცდომაში ქსელის გამოსასვლელზე, რომელიც მიიღება მხედველობაში w_{ki} წონითი კოეფიციენტის აგებისას.

რადგან კერძო წარმოებული გამოითვლება ჯამის მხოლოდ ერთი კომპონენტის – i და j ნეირონებს შორის კავშირით

$$\frac{\partial net_j}{\partial O_i} = w_{ij},$$

სადაც w_{ij} არის დაფარული შრის i-ურ ნეირონსა და გამომავალი შრის j-ურ ნეირონს შორის კავშირის წონა. თუ ჩავსვათ ამ შედეგს შეცდომის წარმოებულს ფორმულაში მივიღებთ:

$$\frac{\partial E}{\partial O_i} = -delta_j w_{ij}.$$

ავჯამოთ ეს გამოსახულება i-ური ნეირონის ყველა კავშირებით გამოსასვლელ შრესთან:

$$\frac{\partial E}{\partial O_i} = \sum_j -delta_j w_{ij}.$$

ეს ტოლობა განსაზღვრავს ქსელის გამოსასვლელზე შეცდომის დამოკიდებულებას დაფარული შრის i-ური ნეირონის გამოსასვლელზე. განვსაზღვროთ ეხლა $delta_i$-ს მნიშვნელობა – ქსელის შეცდომის დამოკიდებულება დაფარული შრის i-ური ნეირონის აქტივაციის დონეზე. ეს მოგვცემს საშუალებას, მივიღოთ ქსელის შეცდომის დამოკიდებულება i-ური კვანძის შემავალი კავშირების წონით კოეფიციენტებზე. კვლავ ვისარგებლოთ იგივეობით

$$-delta_i = \frac{\partial E}{\partial net_i} = \frac{\partial E}{\partial O_i} \cdot \frac{\partial O_i}{\partial net_i}$$

რადგან გამოიყენება ლოგისტიკური ფუნქცია, მივიღებთ

$$-delta_i = \lambda O_i(1-O_i)\sum_j -delta_j w_{ij}$$

შევაფასოთ ეხლა ქსელის გამოსასვლელზე შეცდომის დამოკიდებულება დაფარული შრის i-ური ნეირონის შესასვლელი კავშირების წონებზე. განვიხილოთ კავშირის k-ური წონითი კოეფიციენტი w_{ki} i-ური კვანძით. გვექნება

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial net_i} \cdot \frac{\partial net_i}{\partial w_{ki}} = -delta_i \cdot \frac{\partial net_i}{\partial w_{ki}} = -delta_i x_k$$

სადაც x_k არის i-ური ნეირონის k-ური შესასვლელი. თუ ჩავსვავთ ამ დამოკიდებულებას $delta_i$-ს გამოსახულებაში, მივიღებთ:

$$\frac{\partial E}{\partial w_{ki}} = O_i(1-O_i)\sum_j(-delta_j)x_k.$$

რადგან შეცდომის მინიმიზაციისათვის აუცილებელია წონის შეცვლა ანტიგრადიენტის მიმართულებით, მაშინ წონის აგება სრულდება შემდეგი ფორმულით:

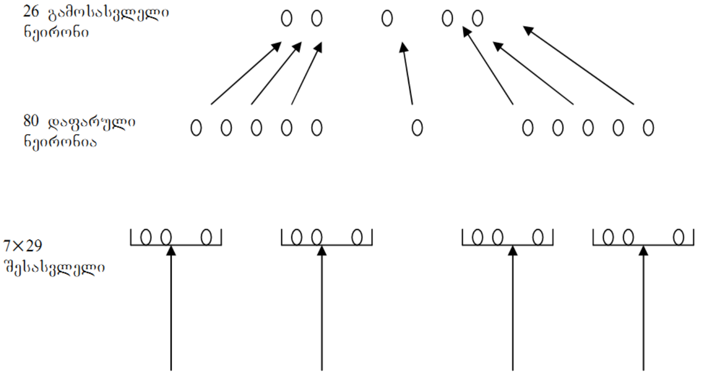
$$\Delta w_{ki} = -\lambda c\frac{\partial E}{\partial w_{ki}} = \lambda c O_i(1-O_i)\sum_j(delta_j w_{ij})x_k$$

ქსელებში, რომლებიც შეიცავენ რამდენიმე დაფარულ შრეს, შეცდომის გავრცელების ეს პროცედურა გამოიყენება რეკურსიულად n-ური დაფარულ შრიდან $(n-1)$ დაფარულ შრისაკენ. მიუხედავად იმისა, რომ შეცდომის უკუგავრცელების მეთოდი იძლევა მრავალშრიანი ქსელების სწავლების პრობლემის გადაწყვეტის საშუალებას, მას გააჩნია იგივე ნაკლი, რაც დელტა წესს. რადგან იგი დაფუძნებულია ექსტრემუმის ძებნის მეთოდზე, ამიტომ ის შეიძლება იკრიბებოდეს ლოკალური მინიმუმისკენ და ამასთან გამოსათვლელად არც ისე მარტივია, განსაკუთრებით ქსელებისათვის შეცდომის რთული ზედაპირით.

12.4. შეცდომის უკუგავრცელების მეთოდის გამოყენების მაგალითი: სისტემა NETtalk. ეს სისტემა წარმოადგენს სწავლების რთული პრობლემის ნეიროქსელური ამოხსნის საინტერესო მაგალითს. სისტემა სწავლობს ინგლისური ტექსტის წარმოთქმას. სისტემა **NETtalk** სწავლობს ტექსტის სტრიქონის კითხვას და აბრუნებს ფონემას შესაბამისი მახვილით სტრიქონის ყოველი ასოსათვის. ფონემა არის ენაში ბგერის ბაზური ერთეული, ხოლო მახვილი – ამ ბგერის ფარდობითი სიმძლავრე. რადგან ყოველი ასოს წარმოთქმა დამოკიდებულია კონტექსტზე და მის გარემომცველ ასოებზე, ამიტომ **NETtalk** მუშაობს ფანჯა-

რასთან, რომელიც შეიცავს შვიდ სიმბოლოს. ამ ფანჯარაში ტექსტის გადაადგილების დროს, **NETtalk** ყოველი ასოსათვის აბრუნებს წყვილს “ფონემა-მახვილი”.

ამ სისტემის არქიტექტურა მოცემულია სურ. #12.4. ქსელი შედგება ნეირონების სა-მი შრისაგან. შემავალი კვანძები შეესაბამებიან ტექსტური ფანჯრის შვიდ სიმბოლოს. ფან-ჯარაში ყოველი პოზიცია წარმოდგენილია 29 შემავალი ნეირონით, მათ შორის თითო ალ-ფაბეტის ყოველი ასოსათვის და სამი პუნქტუაციის ნიშნებისა და პრობელებისათვის. ასო ყოველ პოზიციაში ააქტიურებს შესაბამის ნეირონს. გამომავალი შრის კვანძებიდან ოცდა-ერთის საშუალებით წარმოიდგინება ფონემები ადამიანისათვის დამახასიათებელი არტი-კულაციის თვისებებით. დარჩენილი 5 ნეირონის საშუალებით ხდება მახვილისა და მარცვლების საზღვრების კოდირება. ქსელი შეიცავს 80 დაფარულ ნეირონს, 26 გამომავლ კვანძს და 18629 კავშირს.



სურ. № 12.4. NETtalk სისტემის ქსელის ტოპოლოგია.

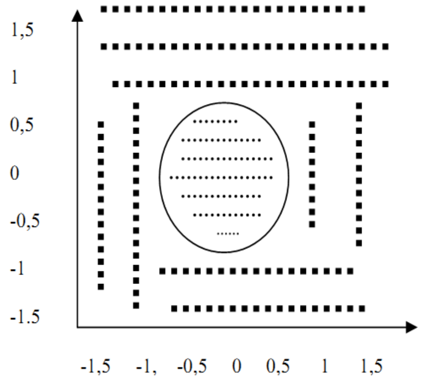
სწავლების დროს სისტემას წარედგინება 7 სიმბოლოიანი ფანჯარა და ის ცდილობს წარმოთქვას შუა ასო. წარმოთქმულის კორექტულთან შედარების შემდეგ, სისტემის წონე-ბი აიგება შეცდომის უკუგავრცელების მეთოდით.

ეს მაგალითი ახდენს ნეირონული ქსელების მრავალი საინტერესო თვისების ილ-უსტრაციას, რომელთა ნაწილიც ასახავს ადამიანური სწავლების ბუნებას. მაგალითად, დასაწყისში როდესაც კორექტული პასუხების პროცენტული შეფარდება მცირეა, სწავლე-ბის პროცედურა მიდის საკმარისად სწრაფად და მცირდება სწორი პასუხებს წილის გაზ-რდის შედეგად. ისე, როგორც ადამიანისათვის, ქსელი რაც უფრო მეტი სიტყვის წარმოთ-ქმას ისწავლის, მით უფრო ადვილი იქნება უცნობი სიტყვების გამოთქმა. ექსპერიმენტებ-მა, რომლებშიც წონითი კოეფიციენტები იცვლებოდნენ შემთხვევით, აჩვენა რომ ქსელი მდგრადია სტრუქტურის დაზიანებისადმი. გამოკვლევებმა აჩვენეს დაზიანებული ქსელის ხელახალი სწავლების მაღალი ეფექტურობა.

მრავალშრიანი ქსელების კიდევ ერთი საინტერესო ასპექტია დაფარული შრეების როლი. სწავლების ნებისმიერმა ალგორითმმა უნდა შეასრულოს განზოგადოება საგნობ-რივი არის წერტილების უცნობ ქსელებზე. დაფარული შრეები თამაშობენ მნიშვნელოვან როლს ქსელისათვის ამ განზოგადოების რეალიზაციაში. ბევრი ქსელის მსგავსად, რომლე-ბიც ახორციელებენ სწავლებას უკუგავრცელების მეთოდით **NETtalk** სისტემის დაფარუ-ლი შრე შეიცავს უფრო მცირე რაოდენობის ნეირონს, ვიდრე შემავალი შრე. ეს ნიშნავს, რომ სწავლების სახეებში შემავალი ინფორმაციის კოდირებისათვის დაფარულ შრეებში გამოიყენება მცირე რაოდენობის ნეირონები, ე.ი. ეს შრე რეალიზაციას უკეთებს აბსტრა-ქციის რომელიღაც ფორმას. ასეთ შეკუმშულ კოდირებას მივყავართ იქამდე, რომ სხვადა-სხვა შემავალ სახეებს დაფარულ შრეში აქვთ იდენტური წარმოდგენები. ზუსტად ამაში მდგომარეობს განზოგადოება. სისტემის სწავლება მიმდინარეობს საკმარისად ეფექტუ-

რად, თუმცა ამისთვის მოითხოვება დიდი რაოდენობის სასწავლო მაგალითები, აგრეთვე სწავლების იტერაციის მრავალჯერადი გამეორება ერთი და იგივე სასწავლო მონაცემებზე.

12.5. ბირთვული მანქანები. ჩვენ განვიხილეთ ერთშრიანი და მრავალშრიანი ნეირონული ქსელები და ვნახეთ, რომ ერთშრიანი ქსელები საშუალებას გვაძლევენ გამოვიყენოთ მარტივი და ეფექტური დასწავლის ალგორითმი, მაგრამ გააჩნიათ შეზღუდული შესაძლებლობები, რადგანაც სწავლის პროცესში შეუძლიათ დაადგინონ მხოლოდ წრფივი საზღვრები. მრავალშრიან ქსელებს გააჩნიათ უფრო მეტი შესაძლებლობები, რადგანაც მათ შეუძლიათ წარმოადგინონ არაწრფივი ფუნქციები, მაგრამ მათი დასწავლის ალგორითმი ხდება ძალიან რთული იმის გამო, რომ ჩნდება დიდი რაოდენობის ლოკალური მინიმუმები და ძალიან იზრდება წონების კოეფიცენტების სივრცის განზომილება. განვიხილოთ შედარებით ახალი მიმართულება დასწავლის ალგორითმებისა, რომლებიც ეფუძნებიან მხარდამჭერი ვექტორების მანქანების, ანუ სხვანაირად ბირთვულ მანქანებს. ეს მანქანები საშუალებას გვაძლევენ გამოვიყენოთ ერთშრიანი და მრავალშრიანი ქსელების საუკეთესო თვისებები. განვიხილოთ ასეთი მანქანების ძირითადი იდეა მაგალითზე.

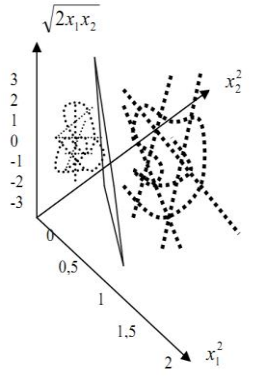


სურ. № 12.5. შესასვლელების ორგანზომილებიანი სივრცე

დავუშვათ, მოცემული გვაქვს შესასვლელების ორგანზომილებიანი სივრცე, რომელიც განისაზღვრება $X=(x_1,x_2)$ ვექტორით და რომელშიც წერტილები, რომლებიც ეკუთვნიან იმ კლასს, რომლის გამოყოფაც ჩვენ გვინდა, მოთავსებულია წრის შიგნით $y=+1$, ხოლო უარყოფითი წერტილები კი – წრის გარეთ $y=-1$. ცხადია, რომ ამ შემთხვევაში ფუნქცია, რომელიც მოახდენს ამ წერტილების გაყოფას, ვერ იქნება წრფივი და საჭიროა გამოვიყენოთ მრავალშრიანი ქსელი, მაგრამ ჩვენ მოვიქცეთ სხვანაირად. გამოვსახოთ შესასვლელი მონაცემები რაიმე გამოთვლითი მახასიათებლების საშუალებით, ანუ ყოველი შესასვლელი $X=(x_1,x_2)$ ვექტორი ასახულია ახალ ვექტორზე რაიმე $F(X)$ ფუნქციის საშუალებით. კერძოდ, ჩვენი მაგალითისათვის გამოვიყენოთ ასეთი ვექტორები:

$$f_1=x_1^2, f_2=x_2^2, f_3=\sqrt{x_1x_2}\ .$$

ასეთი გარდაქმნის შედეგად მივიღებთ წერტილების ისეთ განლაგებას სამგანზომილებიან სივრცეში, რომელიც საშუალებას მოგვცემს გავმიჯნოთ ეს წერტილები წრფივად (იხ. სურ. #12.6.).



სურ. № 12.6. იგივე წერტილები ახალ კოორდინატებში.

ასეთი მიდგომა მართლაც საკმაოდ ზოგადია, რაც უფრო მეტგანზომილებიან სივრცეში ავსახავთ წერტილებს, მით უფრო მეტი ადვილი იქნება მათი წრფივად გამიჯვნა. ამ შემთხვევაში გამოყენებული იყო სამგანზომილებიანი სივრცე, ზოგადად კი, თუ გვაქვს N წერტილი, ყოველთვის შეიძლება მათი წრფივად გაყოფა N-1 განზომილებიან სივრცეში, მაგრამ პრობლემას წარმოადგენს ასეთი გამყოფის აგება.

საკონტროლო კითხვები:

1. როგორ მუშაობს ორშრიანი ქსელი, რომელიც ახდენს “გამომრიცხავი ან” ოპერაციის მოდელირებას?
2. როგორია მრავალშრიანი ქსელების სტრუქტურა?
3. რა განსხვავებაა დელტა წესსა და შეცდომათა უკუგავრცელების მეთოდს შორის?
4. რა არის შეცდომათა უკუგავრცელების მეთოდის ნაკლი?
5. რატომ გამოიყენება ამ მეთოდში ლოგისტიკური ფუნქცია?
6. რას წარმოადგენს ბირთვული მანქანა?
7. რითი სჯობია ბირთვული მანქანა მრავალშრინა ნეირონულ ქსელს?
8. რა სირთულეები წარმოიშვება ბირთვული მანქანის პრაქტიკული გამოყენების დროს?

.ლექცია 13. გენეტიკური ალგორითმები

13.1. ევოლუცია. წინა ლექციებში განვიხილეთ ხელოვნური ნეირონული ქსელების გამოყენება მანქანური დასწავლის პრობლემის გადასჭრელად, მაგრამ ნეირონული ქსელები არ წარმოადგენენ ერთადერთ ბიოლოგიურ პროტოტიპს, რომლებიც გამოიყენება მანქანური დასწავლის ალგორითმების შემუშავებისათვის. ამ ლექციაში ჩვენ განვიხილავთ დასწავლის ალგორითმებს, რომლებიც დაფუძნებული არიან ბიოლოგიური ევოლუციის პრინციპებზე: პოპულაციის ძლიერი პერსონის გადარჩენა მისი განვითარების პროცესში. პოპულაციის უძლიერესი წარმომადგენლის არჩევის როლი ნათლად ვლინდება როგორც სახეთა ევოლუციის, ასევე სოციალურ პროცესებში. ამ პროცესების ფორმალიზებას ცდილობენ უჯრედოვანი ავტომატების თეორიის, გენეტიკური ალგორითმების, გენეტიკური პროგრამირების, ემერჯენტული გამოთვლების და სხვა ფორმების დახმარებით.

დასწავლის ემერჯენტული, ანუ გამოვლინებადი მოდელები (Emergent model) ახდენენ ადაპტაციის ყველაზე უფრო ელეგანტური და ძლიერი ფორმის – მცენარეული და

ცხოველური სამყაროს ცოცხალი ფორმების ევოლუციის იმიტაციას. თავისი ბუნებით ევოლუცია ძალიან მარტივი პროცესია. გარემოს ზემოქმედების შესაბამისად, თანდათანობით, პოპულიაციის ყველაზე უფრო ძლიერ წარმომადგენლებში ზოგიერთი თვისებების უბრალო ცვლილებებისა და შედარებით წარუმატებელი ეგზემპლარების გამოხშირვის შედეგად იზრდება პოპულიაციის ადაპტაციის უნარი და გამომჟღავნდება განსხვავება პოპულაციის სხვადასხვა წარმომადგენელებს შორის. ევოლუციის საფუძვლებში მიმდინარე პროცესების სიმარტივე უზრუნველყოფს მათ საკმარისად საყოველთაო დამკვიდრებას. ბიოლოგიური ევოლუციის პროცესში ახალი სახეობები იქმნებიან გენომის ყველაზე უფრო ხელსაყრელი ცვლილებების შერჩევის ხარჯზე. ანალოგიურად კულტურული ევოლუციის პროცესში, სოციალურად დამუშავებული და მოდიფიცირებული ინფორმაციის ერთეულის გადაცემისას ფორმირდება ცოდნა. გენეტიკური ალგორითმები და სხვა ფორმალური ევოლუციური ანალოგები განაპირობებენ ამოცანის უფრო ზუსტ ამოხსნას ამოცანის ამომხსნელ კანდიდატ პოპულაციებზე გარკვეული ოპერაციების საშუალებით.

ამოცანის ამოხსნის გენეტიკური ალგორითმი შედგება სამი სტადიისაგან, რომელთაგან პირველია ცალკეული პოტენციური ამოხსნების წარმოდგენა სპეციალური ფორმით, რომელიც მოსახერხებელია ცვლილებების და შერჩევის ევოლუციური ოპერაციების შესასრულებლად. უფრო ხშირად ესაა ჩვეულებრივ ბიტური (ორობითი) სტრიქონები. მეორე სტადიაზე ხდება შეჯვარების და მუტაციის რეალიზაცია, რომელიც დამახსიათებელია ცოცხალი ბიოლოგიური ფორმებისათვის. ამ ოპერაციების გამოყენების შემდეგ მიიღება ახალი თაობა თავიანთი მშობლების რეკომბინირებული თვისებებით. ბოლო, მესამე სტადიაზე, რომელიმე შერჩევის კრიტრიუმის საფუძველზე მოხდება საუკეთესოების, ანუ მოცემული პრობლემის ამოხსნის ყველაზე უფრო ზუსტად შესატყვისი ამოხსნების ამორჩევა. ეს ეგზემპლარები ამოირჩევა გასამრავლებლად, ანუ პოტენციური ამოხსნების ახალი თაობის ფორმირებისათვის. საბოლოოდ რომელიღაც თაობა გახდება საწყისი ამოცანის ამონახსნი. გენეტიკურ ალგორითმებში შეიძლება გამოყენებული იყოს უფრო რთული წარმოდგენებიც.

13.2. გენეტიკური ალგორითმების ზოგადი სახე. როგორც უკვე ითქვა, ნეირონული ქსელების მსგავსად, გენეტიკური ალგორითმები ეფუძნებიან ბიოლოგიური პროტოტიპების თვისებას. ამ შემთხვევაში დასწავლა განიხილება, როგორც კონკურენცია ამოცანის ამოხსნის როლის ევოლუციონერ კანდიდატების პოპულაციის შიგნით. ყოველი ამოხსნის შეფასება ხდება შერჩევის კრიტერიუმის თანახმად. შეფასების შედეგების მიხედვით განისაზღვრება მიიღებს თუ არა მონაწილეობას მოცემული ეგზემპლარი ამოხსნების შემდეგ თაობაში. ზოგადი სახით გენეტიკური ალგორითმი შეიძლება წარმოვადგინოთ შემდეგნაიად:

Procedure genetic algorithm:

begin

მივანიჭოთ t:=0;

მოახდინე $P(t)$ პოპულაციის ინიცირება;

while არ არის მიღწეული დამთავრების პირობა;

begin

გამოვთალე ხარისხის კრიტერიუმის მნიშვნელობა $P(t)$ პოპულაციის ყოველი წევრისათვის;

ხარისხის კრიტერიუმის მნიშვნელობის საფუძველზე ავირჩიოთ $P(t)$ – დან საჭირო რაოდენობის წევრები;

შევქმნათ შემდეგი თაობა გენეტიკური ოპერაციების დახმარებით;

ხარისხის კრიტერიუმის მნიშვნელობის გათვალისწინებით შევცვალოთ $P(t)$ პოპულაციის ეგზემპლარები მათი შთამომავლებით;

მიანიჭე $t = t + 1$;

end;

end;

სადაც $P(t)$ არის x_i^t ამონახსნ – კანდიდატების თაობა დროის რაღაც t მომენტში $P(t)=\{x_1^t, x_2^t, \dots, x_n^t\}$,.

ეს ალგორითმი ასახავს გენეტიკური სწავლების ძირითად პრინციპებს. მისი კონკრეტული რეალიზაციები შეიძლება განსხვავდებოდნენ ამოცანების მიხედვით. ~~პერსონების რა პროცენტული თანაფარდობა გადარჩება შემდეგ თაობაში? რამდენი ეგზემპლარი ღებულობს მონაწილეობას შეჯვარებაში? რა სიხშირით და ვის მიმართ გამოიყენებიან გენეტიკური ოპერატორები?~~

$P(t)$ პოპულაციის ეგზემპლარების შეცვლის პროცედურა შეიძლება შესრულდეს უმარტივესი გზით, თუ შევცვლით სუსტი კანდიდატების ფიქსირებულ პროცენტულ თანაფარდობას. უფრო რთული მიდგომა მდგომარეობს პოპულაციის დალაგებაში ხარისხის კრიტერიუმის მიხედვით და ეგზემპლარების ამოგდებაში ალბათობების გათვალისწინებით, რომელიც ხარისხის კრიტერიუმის მნიშვნელობის უკუპროპორციულია. მიუხედავად იმისა, რომ ამ შემთხვევაში პოპულაციის საუკეთესო ეგზემპლარის ამოგდების ალბათობა ძალიან მცირეა, ეს მაინც შესაძლებელია. ასეთი მიდგომის უპირატესობა მდგომარეობს ზოგიერთი "სუსტი" ეგზემპლარის შენარჩუნების შესაძლებლობაში, რომლებმაც შემდეგში შეიძლება შეიტანონ თავიაანთი წვლილი უფრო ზუსტი ამოხსნის მიღებაში. შეცვლის ასეთი ალგორითმი ცნობილია სხვადასხვა სახელით, მათ შორის როგორც მონტე-კარლოს მეთოდი, ან ხარისხის კრიტერიუმის პროპორციულად არჩევის ალგორითმი.

დავუშვათ, საჭიროა გენეტიკური ალგორითმის დამხარებით ვისწავლოთ სტრიქონების კლასიფიცირება, რომლებიც შედგებიან ერთიანებისა და ნულებისაგან. ასეთ ამოცანაში ბიტური სტრიქონების პოპულაციის აღწერა შეიძლება შაბლონის დახმრებით, რომელიც შედგება 1-ს 0-ს და # სიმბოლოებისაგან, სადაც # - სიმბოლო შეიძლება შეესაბამებოდეს როგორც ერთიანს ასევე ნულს. შესაბამისად შაბლონი 1##00##1 წარმოადგენს ყველა რვაბიტიან სტრიქონს, რომელიც იწყება და მთავრდება ერთიანით და შუაში შეიცავს ორ 0-ს მიყოლებით.

გენეტიკური ალგორითმის მუშაობის დროს ხდება P(0) კანდიდატების პოპულაციის ინიცირიება რაიმე სახით. როგორც წესი, ეს ხდება შემთხვევითი რიცხვების გენერატორის დახმარებით. ამონახსნ-კანდიდატების შეფასებისათვის შემოდის ხარისხის კრიტერიუმი $f(x_i^t)$, რომელიც განსაზღვრავს დროის t მომენტში ყოველი კანდიდატის შესაბამისობის ზომას. კლასიფიკაციის ამოცანების დროს კანდიდატების შესაბამისობის ზომას წარმოადგენს დასასწავლი მაგალითების სიმრავლეში სწორი პასუხების პროცენტული თანაფარდობა. ხარისხის ასეთი კრიტერიუმის დროს ყოველ ამონახსნ – კანდიდატს შეესაბამება მნიშვნელობა

$$f(x_i^t / m(P,t)$$

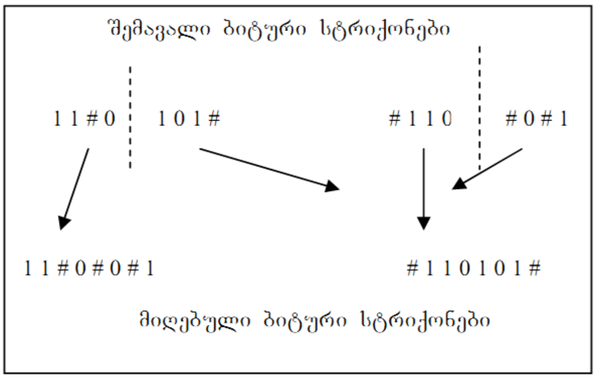
სადაც $m(P,t)$ არის ხარისხის კრიტერიუმის საშუალო მნიშვნელობა პოპულაციის ყველა წევრისათვის. ჩვეულებრივ შესაბამისობის ზომა დამოკიდებულია დროზე, ამიტომ ხარისხის კრიტერიუმი შეიძლება იყოს ან $f(x_i^t)$, ან ფუნქცია, დამოკიდებული ამოცანის ამოხსნის სტადიაზე. ყოველი კანდიდატის ანალიზის შემდეგ აირჩევა წყვილები რეკომბინაციისათვის. ამ დროს გამოიყენება გენეტიკური ოპერატორები, რომელთა შესრულების შედეგად მიიღება ახალი ამონახსნები მშობლების თვისებების კომბინირების გზით. ისევე, როგორც ეს ხდება ევოლუციის ბუნებრივ პროცესში, რეპროდუქტიულ პროცესში მონაწილეობის ხარისხი ყოველი კანდიდატისათვის განისაზღვრება ხარისხის კრიტერიუმის მნიშვნელობით: ხარისხის კრიტერიუმის უფრო მაღალი მნიშვნელობის მქონდე კანდიდატები მონაწილეობენ კვლავწარმოქმნის პროცესში უფრო დიდი ალბათობით. როგორც უკვე იყო აღნიშნული, ამორჩევა ხორციელდება ალბათური კანონების საფუძველზე, როდესაც პოპულაციის სუსტ წევრებს აქვთ კვლავწარმოქმნის მცირე ალბათობა, თუმცა ეს შესაძლებლობა გამორიცხული არ არის. ზოგიერთი უსუსტესი ეგზემპლარის გადარჩენას დიდი

Formatted: Tab stops: Not at 0.64" + 1.27" + 1.91" + 2.54" + 3.18" + 3.82" + 4.45" + 5.09" + 5.73" + 6.36" + 7" + 7.63" + 8.27" + 8.91" + 9.54" + 10.18"

Formatted: Tab stops: Not at 0.64" + 1.27" + 1.91" + 2.54" + 3.18" + 3.82" + 4.45" + 5.09" + 5.73" + 6.36" + 7" + 7.63" + 8.27" + 8.91" + 9.54" + 10.18"

მნიშვნელობა აქვს პოპულაციის განვითარებისათვის, იმიტომ რომ ისინი შეიძლება შეიცავდნენ ამოხსნის ზოგიერთ მნიშვნელოვან კომპონენტს, მაგალითად ბიტური სტრიქონის გარკვეულ ფრაგმენტს, რომელიც არ გააჩნიათ სხვა კანდიდატებს და რომელიც შეიძლება გამოყენებული იქნას კვლავწარმოქმნის დროს.

არსებობს შთამომავლების მიღების მრავალი გენეტიკური ოპერატორი, რომლებიც ინარჩუნებენ მშობლების თვისებებს. მათ შორის ყველაზე უფრო ტიპურია შეჯვარება. შეჯვარების დროს ორი ამონახსნ-კანდიდატი იყოფა რამდენიმე ნაწილად, ეს ნაწილები იცვლება ერთმანეთში და ამ ოპერაციის შედეგად მიიღება ორი ახალი კანდიდატი (იხ. სურ. # 13.1). როგორც ამ სურათიდან ჩანს, სტრიქონები იყოფა ორ ტოლ ნაწილად, რის შემდეგაც ფორმირდება ორი შთამომავალი, რომლებიც შეიცავენ ორივე მშობელის თითო ფრაგმენტს. ამონახსნ-კანდიდატები შეიძლება გაიყონ ნებისმიერ წერტილში და ნებისმიერი რაოდენობის ნაწილებად. გაყოფის წერტლების ადგილი და მათი რაოდენობა შეიძლება იცვლებოდეს ნებისმიერად არა მარტო სხვადასხვა ამოცანებისათვის, არამედ თვით ამოცანის ამოხსნის პროცესშიც.



სურ. # 13.1. შეჯვარების ოპერაციის გამოყენება ორი რვაბიტიანი სტრიქონისათვის
სიმბოლოო # აღნიშნავს " ნებისმიერ მნიშვნელობას".

Formatted: Tab stops: Not at 0.64" + 1.27" + 1.91" + 2.54" + 3.18" + 3.82" + 4.45" + 5.09" + 5.73" + 6.36" + 7" + 7.63" + 8.27" + 8.91" + 9.54" + 10.18"

დავუშვათ, რომ მიზნობრივ კლასს წარმოადგენს ყველა იმ სტრიქონების ნაკრები, რომლებიც იწყებიან და მთავრდებიან ერთიანებით. სურ. #13.1. მოცემული ორივე მშობელი სტრიქონი საკმარისად კარგად ერგებიან ამ ამოცანის ამოხსნას, მაგრამ მათი პირველი შთამომავალი უფრო ზუსტად შეესაბამება ხარისხის კრიტერიუმს ვიდრე რომელიმე მშობელი. მაგრამ მისი "ძმა" გაცილებით ცუდია ნებისმიერ თავის მშობლეზე, ამიტომ ეს ეგზემპლარი ალბათ გამოირიცხება ერთ-ერთ უახლოეს თაობაში.

Formatted: Tab stops: Not at 0.64" + 1.27" + 1.91" + 2.54" + 3.18" + 3.82" + 4.45" + 5.09" + 5.73" + 6.36" + 7" + 7.63" + 8.27" + 8.91" + 9.54" + 10.18"

მეორე მნიშვნელოვანი გენეტიკური ოპერატორია მუტაცია. ეს ოპერაცია შემთხვევითად ამოირჩევს კანდიდატს და ასევე შემთხვევითად შეუცვლის მას ზოგიერთ თვისებას. მაგალითად, მუტაცია იქნება შაბლონში ბიტის შემთხვევითად ამორჩევა და მისი მნიშვნელობის 1-იანის 0-ით ან #-ით შეცვლა. მუტაციის ოპერაციის არსი მდგომარეობს ამოხსნის ზოგიერთი ისეთი მნიშვნელოვანი კომპონენტის წარმოქმნის შესაძლებლობაში ,რომელიც არ იყო საწყის პოპულაციაში. თუ ზემოთ განხილულ მაგალითში საწყის პოპულაციის არცერთი წევრი არ შეიცავს 1-ს პირველ პოზიციაში, მაშინ შეჯვარების პროცესში არ შეიძლება ამ თვისების მქონე შთამომავლის მიღება. პირველი ბიტის მნიშვნელობის შეცვლა შეიძლება მხოლოდ მუტაციის შედეგად.

გენეტიკური ალგორითმის მუშაობა გრძელდება მანამ, სანამ არ იქნება მიღწეული მისი დამთავრების პირობა, მაგალითად, ერთი ან რამდენიმე კანდიდატისათვის ხარისხის კრიტერიუმის მნიშვნელობა არ გადააჭარბებს რაიმე ზღურბლს. განვიხილოთ

ამოცანის წარმოდგენისა და ხარისხის კრიტერიუმების შერჩევის პრობლემები კონკრეტულ მაგალითებზე.

13.3. გენეტიკური ალგორითმის გამოყენება ნორმალურ კონუნქციურ ფორმაში აღწერილი პრობლემისათვის. ნორმალურ კონუნქციურ ფორმაში (ნკფ) აღწერა ნიშნავს გამოსახულებას, რომელიც შედგება ერთმანეთთან "და" ოპერაციით დაკავშირებული ოპერატორების მიმდევრობისაგან. ამასთან, თითოეული ოპერატორი უნდა შედგებოდეს ან ერთი ასოსაგან, ან რამდენიმე ასოსაგან, რომლებიც ერთმანეთთან დაკავშირებულია "ან" ოპერაციით. მაგალითად, გამოსახულება

$$(\neg a\vee c)\wedge(\neg a\vee c\vee\neg e)\wedge(\neg b\vee c\vee d\vee\neg e)\wedge(a\vee\neg b\vee c)\wedge(\neg e\vee f)$$

წარმოადგენს ნკფ-ს, რომელიც შედგება ხუთი ოპერატორისაგან, სადაც თითოეული ოპერატორი წარმოადგენს რამდენიმე ლიტერალის დიზუნქციას. წარმოდგენა ნკფ-ში ნიშნავს, რომ ყველა ლიტერალისათვის (ჩვენს შემთხვევაში – ექვსი) არსებობს ლოგიკური მნიშვნელობები 0 (მცდარი) ან 1 (ჭეშმარიტი) ისეთი, რომ ნკფ ღებულობს მნიშვნელობას 1. მაგალითად, თუ ლიტერალები c და f ღებულობენ მნიშვნელობებს 1, მაშინ ნკფ-ც ღებულობს მნიშვნელობას 1. აქედან მტკიცდება, რომ ეს ნკფ წარმოდგენადია.

იმისათვის, რომ აღვწეროთ მონაცემები ამამოცანისათვის, დაგვჭირდება ექვსბიტიანი სტრიქონები, სადაც თითოეული ბიტი შეესაბამება a,b,c,d,e,f ლიტერალების მნიშვნელობებს. ასე მაგალითად 1 0 1 0 1 0 სტრიქონი ნიშნავს, რომ ლიტერალები a,c,e ღებულობენ მნიშვნელობას 1 და ლიტერალები b,d,f - კი მნიშვნელობას 0.

გენეტიკური ოპერაციების შესრულების შედეგად უნდა მივიღოთ შთამომავლობა, რომლისთვისაც ნკფ ღებულობს მნიშვნელობას 1. ეს იმას ნიშნავს, რომ გენეტიკური ოპერაციების შედეგად უნდა მივიღოთ ახალი სტრიქონი, რომელიც შეიძლება განხილული იყოს როგორც ამოცანის ამოხსნის კანდიდატი. ამ შემთხვევაში შესაძლებელია გამოყენებული იყოს როგორც შეჯვარების, ასევე მუტაციის ოპერაციები, რადგანაც მათი გამოყენების შედეგად კვლავ მიიღება ექვსბიტიანი სტრიქონები. ასევე შეიძლება გამოვიყენოთ ინვერსირების და გაცვლის ოპერაციები. პირველის არსი მდგომარეობს სტრიქონში ბიტების მიმდევრობის შეცვლა შებრუნებულზე, ხოლო მეორესი კი – ნებისმიერი ორი ბიტისათვის ადგილების გაცვლაში. ამ თვალსაზრისით ამოცანის სტრიქონების სახით წარმოდგენა ყველაზე მოსახერხებელია.

შედარებით რთულია ხარისხის კრიტერიუმების ამორჩევა, რადგანაც ამოხსნა წარმოადგენს ისეთ სტრიქონს, რომლისთვისაც ნკფ ღებულობს მნიშვნელობას ერთიანს და ერთი შეხედვით რთულია ვიწინასწარმეტყველოთ რამდენიმე სტრიქონიდან რომელი უფრო ახლოს დგას ამოხსნასთან. მაგრამ, თუ გავითვალისწინებთ, რომ ნკფ ღებულობს მნიშვნელობას ერთს მხოლოდ მაშინ, როდესაც ხუთივე გამოსახულება ღებულობს მნიშვნელობას ერთს, ჩვენ შეგვიძლია მოვახდინოთ ამოცანის დეკომპოზიცია და ხარისხის კრიტერიუმებად ავიღოთ ისეთი ბიტური სტრიქონები, რომლებიც უზრუნველყოფენ რაც შეიძლება მეტი გამოსახულებებისათვის ერთიანის მინიჭებას. კერძოდ, შეგვიძლია შევადგინოთ ასეთი ცხრილი:

1 1 0 0 1 0 – 1

0 1 0 0 1 0 – 2

0 1 0 0 1 1 – 3

1 0 1 0 1 1 – 5

ანუ პირველი სტრიქონი უზრუნველყოფს ერთი გამოსახულების ერთთან ტოლობას და მისი რეიტინგია ერთი, მეორე სტრიქონი უზრუნველყოფს ორი გამოსახულების ერთთან ტოლობას და მისი რეიტინგია ორი, მესამე სტრიქონი უზრუნველყოფს სამი გამოსა-

Formatted: Tab stops: Not at 0.64" + 1.27" + 1.91" + 2.54" + 3.18" + 3.82" + 4.45" + 5.09" + 5.73" + 6.36" + 7" + 7.63" + 8.27" + 8.91" + 9.54" + 10.18"

ხულების ერთთან ტოლობას და მისი რეიტინგია სამი. ბოლო სტრიქონი წარმოადგენს ფაქტობრივად ამოხსნას, რადგანც ის უზრუნველყოფს ხუთივე გამოსახულების ერთთან ტოლობას.

ამ აღწერის შემდეგ შესაძლებელია გამოვიყენოთ გენეტიკური ალგორითმი ამ ამოცანის ამოსახსნელად. ყველაზე მნიშვნელოვანი ასეთ მიდგომაში არის ის, რომ ალგო-რითმი ერთდროულად მუშაობს ამოხსნათა მთელ პოპულიაციასთან, ანუ ფაქტობრივად ძებნა ხდება პარალელურ რეჟიმში. ამას ემატება ისიც, რომ შესაძლებელია ამოცანის დე-კომპოზიცია და ალგორითმს შეუძლია იმუშაოს თითოეული გამოსახულების ამოხსნაზე ცალ-ცალკე.

13.4. გენეტიკური ალგორითმის გამოყენება კომივოიაჟერის ამოცანისათვის. კომივოიაჟე-რის ამოცანა – ეს არის ტესტური ამოცანის კლასიკური მაგალითი ხელოვნური ინტელექ-ტის მეთოდებისათვის და ზოგადად კომპიუტერული მეცნიერებისათვის. ჩვენ ეს ამოცანა ვახსენეთ გადრჩევის ალგორითმებთან დაკავშირებით . ჩამოვაყალიბოთ ამოცანა ზოგადი სახით:

კომივოიაჟერს მოეთხოვება მოინახულოს N ქალაქი. სვლის მარშრუტზე ქალაქების ყოველი წყვილისათვის დადგენილია ფასი (მაგალითად მანძილი). მოითხოვება მოიძებ-ნოს მინიმალური ფასის გზა, რომელიც იწყება რომელიმე ქალაქიდან და რომელიც უზრ-უნველყოფს დანარჩენი ქალაქების ნახვას ზუსტად ერთხელ და გამგზავრების ცენტრში დაბრუნებას.

Formatted: Font: Italic

Formatted: Font: Italic

Formatted: Font: Italic

კომივოიაჟერის ამოცანა ფართოდ გამოიყენება პრაქტიკაში, მათ შორის იგი უზრუნველ-ყოფს ისეთი ამოცანების ამოხსნას, როგორებიცაა ელექტრონული სქემების გახსნის პრობ-ლემა, რენტგენული კრისტალოგრაფიის და დიდი ინტეგრალური სქემების წარმოებისას მარშრუტიზაციის ამოცანების გადაჭრა. ზოგიერთ ამ ამოცანებში დამატებით მოითხოვება, რომ მოიძებნოს მინიმალური ფასის მქონე გზა. დიდ ინტერესს წარმოადგენს კომივოი-აჟერის კლასის ამოცანების ანალიზი მათი რეალიზაციის ეფექტიანობის თვალსაზრისით. რა უფრო იაფი დაგვიჯდება, დავხარჯოთ რამდენიმე საათი მძლავრი კომპიუტერის გამო-ყენებით სუბოპტიმალური ამოხსნის მისაღებად, თუ გამოვიყენოთ ნაკლებად ძლიერი კო-მპიუტერი, რომელიც რამდენიმე წუთში მოგვცემს არაოპტიმალურ, მაგრამ სავსებით მი-საღებ ამოხსნას? კომივოიაჟერის ამოცანა – ეს არის საინტერესო და რთული (NP სირთ-ულის) პრობლემა, რომელიც ეხება ძებნის სტრატეგიის რეალიზაციის მრავალ ასპექტებს.

იმისათვის, რომ გამოვიყენოთ გენეტიკური ალგორითმი ამ ამოცანის ამოსახსნე-ლად, პირველ რიგში უნდა ავირჩიოთ წარმოდგენა ქალაქების მონახულების მარშრუ-ტისათვის, აგრეთვე შევარჩიოთ გამოსაყენებელი გენეტიკური ოპერატორები. რაც შეეხება ხარისხის კრიტერიუმის არჩევას, ის ძალიან მარტივია ამ შემთხვევაში. საჭიროა მხოლოდ გზის სიგრძის შეფასება. ამის შემდეგ მარშრუტები შეიძლება დავალაგოთ სიგრძის მიხე-დვით – რაც მოკლეა, მით უკეთესი.

Formatted: Font: Sylfaen, Georgian (Georgia)

ვცადოთ მარშრუტის წარმოდგენა ბიტური სტრიქო-ნების საშუალებით. დავუშვათ აუცილებელია 9 ქალაქის 1,2,...,9 მონახულება. გზა შეგვი-ძლია წარმოვადგინოთ 9 მთელი რიცხვისაგან შედგენილი დალაგებული სიით. ყოველი ქალაქის რიგითი ნომერი წარმოვადგინოთ ოთხბიტიანი სტრიქონით: 0001, 0010,...,1001. ამ დროს გამოსახულება

0001 0010 0011 0100 0101 0110 0111 1000 1001

წარმოადგენს ქალაქების მონახულების მიმდევრობას მათი რიგითი ნომრის ზრდის მიხე-დვით. ამ გამოსახულებაში ჰარი (პრობელი) გამოყენებულია მხოლოდ აღქმის სიმარტივი-სათვის. რომელი გენეტიკური ოპერატორები შეიძლება გამოვიყენოთ ასე წარმოდგენილი ამოცანისათვის? შეჯვარების ოპერაცია ცალსახად არ გამოდგება, რადგან მის შედეგად მიღებული სტრიქონი არ წარმოადგენს ისეთ გზას, რომლის დროსაც ყოველი ქალაქის მო-ნახულება შესაძლებელი იქნება მხოლოდ ერთჯერ. მართლაც, შეჯვარების დროს ზოგი-ერთი ქალაქი ამოვარდება მიმდევრობიდან, ხოლო სხვები შეგხვდება მასში ორჯერ. რა შეიძლება ითქვას მუტაციაზე? დავუშვათ, რომ მეექვსე ქალაქის აღვნიშვნაში 0110 მარც-ხენა განაპირა წევრი იცვლება 1-ით, მაშინ მიღებული

რიცხვი 1110 შესაბამისი იქნება 14 რიგითი ნომრის, რომელიც არ შედის ქალაქების დასაშვებს ჩამონათვალში. გზის გამო-სახულებაში ქალაქების ინვერტირება მოცემულ შემთხვევაში წარმოადგენს ერთადერთ დასაშვებ ოპერაციას, მაგრამ საკმარისი იქნება თუ არა ეს ოპერაცია, უცნობია.

კომივოიაჟერის ამოცანა შეიძლება ამოიხსნას სხვანაირადაც: უარი ვთქვათ ბიტურ წარმოდგენაზე და მივანიჭოთ ქალაქებს ჩვეულებრივი რიგითი ნომრები: 1,2,..,9. ამ ქალაქებს შორის გზა წარმოადგენს ცხრა ციფრისგან შემდგარ რომელიღაც მიმდევრობას, ხოლო შესაბამისი გენეტიკური –ოპერატორები –მოგვცემენ –ახალი –გზის –ფორმირების საშუ-ალებას. ამ შემთხვევაში მუტაცია, როგორც მარშრუტში ორი ქალაქის შემთხვევითი გაცვ-ლა, იქნება დასაშვები ოპერაცია, მაგრამ შეჯვარება ისევ უსარგებლოა. ოპერატორი, რო-მელიც ცვლის მარშრუტის ფრაგმენტებს იმავე გზის სხვა ფრაგმენტებით ან ნებისმიერი ოპერატორი, რომელიც უცვლის ადგილებს მარშრუტში ქალაქების ნომრებს (ამოგდების, დამატების -ან დუბლირების -გარეშე), -აღმოჩნდება საკმარისად ეფექტური, მაგრამ ასეთი მიდგომების –დროს –შეუძლებელია –მშობლების საუკეთესო თვისებების შეთავსება შთამო-მავალში, რადგან ამისათვის მოითხოვება მისი ფორმირება ორი მშობლის საფუძველზე.

დამუშავებული იქნა სხვადასხვა სახის –შეჯვარების –ოპერატორები, –რომლებიც იცი-ლებენ ამ პრობელმებს და იძლევიან მოსანახულებელი ქალაქების დალაგებულ სიასთან მუშაობის საშუალებას. ერთერთი ასეთი ოპერატორია დალაგებული შეჯვარების ოპერა-ტორი.

დალაგებული შეჯვარების პროცესში შთამომავლის აგება ხდება ერთ-ერთი მშობლის გზაში ქალაქების ქვემიმდევრობის არჩევის საშუალებით. მასში აგრეთვე შენარჩუნებულია მეორე მშობლის ქალაქების ფარდობითი რიგი. დასაწყისში ამოირჩევა აღნიშნული კვეთის ორი წერტილი (აღინიშნება “I” სიმბოლოთი), რომლებიც შემთხვევითად დგებიან თითოეული მშობლის ერთსა და იგივე პოზიციებში. კვეთის წერტილების ადგილმდებარეობა აირჩევა შემთხვევითად, თუმცა მშობლებიდან თითოეულისათვის ეს წერტილები ემთხვევა –ერთმანეთს. –მაგალითად, –თუ –ორი მშობლისთვის **p1** –და **p2** კვეთის წერტილები განლაგებული არიან მესამე და მეშვიდე ქალაქების შემდეგ

p1=(192 I 4657 I 83)
p2=(459 I 1876 I 23)

მაშინ ორი შთამომავალი **c1** და **c2** მიიღება შემდეგნაირად. პირველ რიგში ყოველი შთამომავლისათვის კოპირდება წყვეტის წერტილებს შორის მოთავსებული მშობლების სტრიქონის ფრაგმნეტი:

c1=(xxx I 4657 I xx)
c2=(xxx I 1876 I xx)

(დააკვირდით მშობლების და შთამომამვლების ინდექსებს, **c1**-ით აღნიშნული შთამომავალი, რომელშიც გადადის **p1**-ის კვეთებს შორის მოთავსებული ფრაგმენტი). ამის შემდეგ **p2** მშობლის სტრიქონი გარდაიქმნება შემდეგნაირად: სტრიქონი იწყება მეორე კვეთის შედეგ დარჩენილი ფრაგმენტით და შემდეგ გაგრძელდება ჩვეულებრივად, მიიღება სტრიქონი:

234591876

რადგანაც ქალაქები 4,6,5 და 7 უკვე შედიან პირველი შთამომავლის შემადგენლობაში, ამიტომ ამოვყრით მათ სტრიქონიდან და მიიღება შემოკლებული სტრიქონი **2,3,9,1,8,** რომელიც ჩაისმება **c1**-ში, **p2**-ში არსებული რიგის შენარჩუნებით და გვექნება:

c1 = (239 I 4657 I 18).

ანალოგიურად მიიღება მეორე შთამომავალიც

c2 = (392I1876I45).

ამგვარად დალაგებული შეჯვარების დროს გზის ფრაგმენტები ერთი მშობლიდან (**p1**) გადაეცემა შთამომავალს (**c1**), ამ დროს ქალაქების მონახულების რიგი მას გადაეცემა მეორე (**p2**) მშობლისაგან. ასეთი მიდგომა ეყრდნობა ინტუიციურ ვარაუდს იმის შესახებ, რომ ქალაქების მონახულების რიგი მნიშვნელოვან როლს თამაშობს უმოკლესი

გზის მოძებნაში. ამიტომ ხდება შთამომავლებისათვის ქალაქების მონახულების რიგის შესახებ ინფრომაციის შენახვა.

დალაგებული შეჯვარების ალგორითმი იძლევა ყველა ქალაქის ერთხელ მონახულების გარანტიას. ამ ოპერაციის შედეგზე მუტაციის გამოყენება უნდა მოხდეს ძალიან ფრთხილად. როგორც ზევით უკვე იყო აღნიშნული, მუტაციის ოპერაცია დაიყვანება ერთი მარშრუტის ჩარჩოებში ორი ქალაქის ადგილის გაცვლაზე, ანუ ინვერტირებაზე. მოცემულ შემთხვევაში უბრალო ინვერტირება (ქალაქების მონახულების რიგის მარტივი შეცვლა) მიუღებელია, რადგან ამ დროს არ ხდება ახალი გზის ფორმირება. მაგრამ, თუ ერთი მარშრუტის ჩარჩოებში ავირჩევთ რაიმე ფრაგმენტს და მოვახდენთ მის ინვერტირებას მაშინ ამან შეიძლება მოგვცეს კარგი შედეგი. მაგალითად ავიღოთ გზა:

c1 = (239 I 4657 I 18)

და მოვახდინოთ მისი შუანაწილის ინვერტირება. შთამომავალი მიიღებს სახეს:

c1 = (239 I 7564 I 18).

შეიძლება შემოვიტანოთ მუტაციის კიდევ ერთი ოპერატორი, რომელიც მდგომარეობს შემდეგში: შემთხვევითად ავირჩიოთ რომელიმე ქალაქი და გადავადგილოთ იგი მარშრუტის შემთხვევითად არჩეულ ადგილზე. ასეთი ოპერატორი შეიძლება გამოვიყენოთ გზის ფრაგმენტისთვისაც, მაგალითად, ავირჩიოთ სამი ქალაქისგან შემდგარი ფრაგმენტი და მოვათავსოთ იგი ახალ, შემთხვევითად არჩეულ ადგილზე. განსხვავებით ნკფ-ის ამოცანისაგან ამ მაგალითში უფრო რთული იყო როგორც წარმოდგენის, ასევე ოპერატორების განსაზღვრა, რაც გვიჩვენებს, რომ გენეტიკური ალგორითმისთვის დამახასიათებელ ცოდნის წარმოდგენის, ოპერატორების არჩევის და ხარისხის კრიტერიუმის განსაზღვრის პრობლემები წარმოადგენს ერთერთ ყველაზე მთავარ პრობლემას გენეტიკური ალგორითმებისთვის. არჩეული წარმოდგენა უნდა იყოს ისეთი, რომ შესაძლებელი იყოს გენეტიკური ოპერატორების გამოყენება. ზოგჯერ ბუნებრივია ბიტური წარმოდგენა (როგორც ეს იყო პირველ მაგალითში). ამ შემთხვევაში პოტენციალური ამონახსნების მისაღებად საჭიროა პირდაპირ გამოვიყენოთ ისეთი ტრადიციული გენეტიკური ოპერატორები, როგორიცაა შეჯვარება და მუტაცია. მაგრამ კომივოიაჟერის ამოცანაში სრულიად სხვა სიტუაციაა. პირველ რიგში, მისთვის არ გამოდგება ბიტური წარმოდგენა. მეორეც, მუტაციის და შეჯვარების ოპერაციების განსაზღვრისას ყოველი შთამომავლისთვის აუცილებელია თვალი მივადევნოთ საჭირო თვისებების შესრულებას (მაშრუტში ყველა ქალაქის არსებობა და თითოეულის მხოლოდ ერთხელ ნახვა). ბოლოს, გენეტიკური ოპერაციების რეალიზაციისას არსებითი ინფორმაცია უნდა გადაეცეს შემდეგ თაობას. კომივოიაჟერის ამოცანაში გადამწყვეტია მარშრუტში ქალაქების თანმიმდევრობა, ამიტომ შთამომავლებს უნდა გადაეცეთ ამ ინფორმაციის ფარგმენტები. იმისათვის, რომ უზრუნველვყოთ თვისებების ასეთი მემკვიდრეობა, აუცილებელია, თითოეული ამოცანისათვის ცალცალკე შესაბამისი სახით ავირჩიოთ მონაცემების წარმოდგენის ხერხი და გენეტიკური ოპერატორები.

ცხრილი 13.1.

რიცხვი ორობით სისტემაში	გრეის კოდი
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111

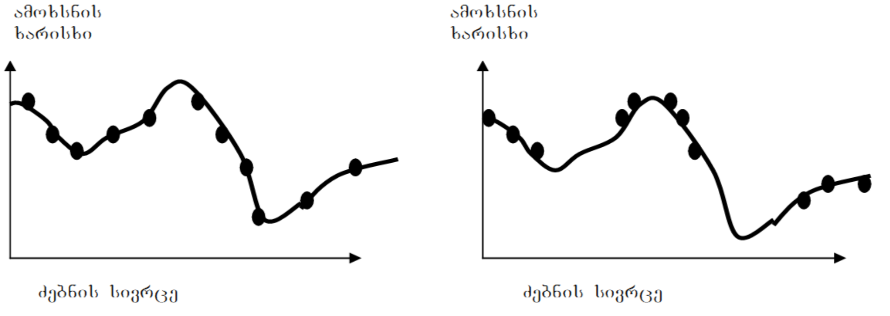
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

განვიხილოთ მონაცემების წარმოდგენის პრობლემის ერთი ასპექტი, რომელიც ცნობილია, როგორც წარმოდგენის “ბუნებრივობის” პრობლემა. ამისათვის განვიხილოთ ძალიან მარტივი, მაგალითი. განვიხილოთ 6,7,8 და 9 რიცხვების გარჩევადობის ამოცანა. ბუნებრივი, ანუ ათობითი წარმოდგენა, რომელიც უზრუნველყოფს მონაცემების დახარისხებას, არის ჩვეულებრივი აღწერა მთელი რიცხვებით, რადგანაც ათ ციფრს შორის ყოველი შემდეგი 1-ით მეტია წინაზე. ორობით აღწერაზე გადასვლის შემდეგ ეს ბუნებრივობა ქრება. განვიხილოთ 6,7,8 და 9 რიცხვების ორობითი წარმოდგენა:

6=0110, 7= 0111, 8=1000, 9=1001

შევნიშნოთ, რომ რიცხვები 6 და 7, აგრეთვე 8 და 9 განსხვავდებიან მართლაც ერთმანეთისგან ერთი ბიტით. მაგრამ 7 და 8 რიცხვებს საერთო არაფერი აქვთ. წარმოდგენის ამ თვისებამ შეიძლება გამოიწვიოს დიდი პრობლემები იმ ამოცანების ამოხსნის დროს, რომლებშიც მოითხოვება ამ სახეების სისტემატიზაცია. არაერთგვაროვანი წარმოდგენის პრობლემის მოსახსნელად, გენეტიკური ალგორითმების თეორიაში ფართოდ გამოიყენება გრეის კოდი. გრეის კოდი, რომელსაც ოფიციალურად ეწოდება რეფლექსური ორობითი კოდი, წარმოადგენს ნუმერაციის ორობით სისტემას, რომელშიც ორი მეზობელი მნიშვნელობა განსხვავდება მხოლოდ ერთ ორობით თანრიგში. მაგალითად, ორობით სისტემაში წარმოდგენილი პირველი თექვსმეტი რიცხვისათვის გრეის კოდს აქვს სახე (იხ. ცხრილი 14.1). შევნიშნოთ, რომ აქ ყოველი რიცხვი განსხვავდება თავისი მეზობლისგან ზუსტად ერთი ბიტით. ჩვეულებრივი ორობითი წარმოდგენის მაგივრად გრეის კოდის გამოყენების დროს მდგომარეობებს შორის გადასვლები, გენეტიკური ოპერატორების შესრულებისას, წარმოადგენენ უფრო ბუნებრივს.

გენეტიკური ალგორითმების მნიშვნელოვან უპირატესობას წარმოადგენს ძებნის პარალელური ბუნება. ეს ალგორითმები ახდენენ ექსტრემუმის ძებნის ერთ-ერთი მძლავრი ფორმის რეალიზაციას, რომელიც მდგომარეობს ერთდროულად რამდენიმე ამონახსკანდიდატების მხარდაჭერაში.



ა) ბ)

სურ. № 13.2. გენეტიკური ალგორითმი, როგორც ექსტრემუმის პარალელური ძებნა
ა) ძებნის საწყისი სივრცე ბ) ძებნის სივრცე *n* თაობის შემდეგ

Formatted: Tab stops: Not at 0.64" + 1.27" + 1.91" + 2.54" + 3.18" + 3.82" + 4.45" + 5.09" + 5.73" + 6.36" + 7" + 7.63" + 8.27" + 8.91" + 9.54" + 10.18"

სურ. #13.2. ნაჩვენებია, როგორ იკრიბება ამონახსნების სიმრავლე ექსტრემუმის წერტილი- საკენ ძებნის სივრცეში. ამ სურათზე ჰორიზონტალური ღერძზე აღებულია შესაძლო წერ-ტილები ამონახსნთა სიმრავლეში, ხოლო ვერტილაკურზე ამ ამოხსნათა ხარისხი. წერტი-ლები წირებზე არიან მიმდინარე პოპულაციის ამონახსნ-კანდიდატების წევრები, რომლე-ბიც მიღებული არიან გენეტიკური ალგორითმის მუშაობის შედეგად. დასაწყისში ამონახ-სნები თანაბრად ნაწილდებოდნენ ძებნის სივრცეში. რამდენიმე იტერაციის შემდეგ ისინი ჯგუფდებიან არეებში, რომლებიც შეესაბამებიან ამოხსნების უფრო მაღლა ხარისხს.

გენეტიკური ალგორითმის, როგორც ექსტრემუმის ძებნის ალგორითმის აღწერის დროს, არაცხადად იგულისხმება გადადგილება ხარისხის კრიტერიუმით განსაზღვრული ზედაპირის გასწვრივ. ამ ზედაპირზე არსებობენ თავისი მაღლობები და დაბლობები, აგ-რეთვე ლოკალური მაქსიმუმები და მინიმუმები. ამ სივრცის უწყვეტობის დარღვევა არის კონკრეტული ამოცანისთვის წარმოდგენის და გენეტიკური ოპერატორების არჩევის შედ-ეგი. მაგალითად, უწყვეტობის ასეთი გარღვევა შეიძლება გამოწვეული იყოს არასწორი კოდირებით, მათ შორის გრეის კოდის არ გამოყენებით. ავღნიშნოთ აგრეთვე, რომ გენეტი-კური ალგორითმები, ექსტრემუმის ძებნის "ხარბი" ალგორითმებისაგან განსხვავებით (ლექცია#5), მაშინვე არ აგდებენ არაპერსპექტიულ ამოხსნებს. გენეტიკური ოპერატო-რე-ბის რეალიზაციის დროს ცუდი ამონახსნებიც კი შეიძლება დარჩნენ პოპულაციაში და შეი-ტანონ თავიანთი წვლილი ამოხსნების შემდეგი თაობების ფორმირებაში.

მდგომარეობათა სივრცეში ევრისტიკული ძებნა განსხვავდება გენეტიკური ალგო- რითმებისაგან კიდევ ერთი მომენტით. ევრისტიკული ძებნის დროს ყოველ ბიჯზე ხდება მიმდინარე და მიზნობრივ მდგომარეობებს შორის განსხვავებების ანალიზი. ასეთი ინფო- რმაციის გათვალისწინება ხდება მაგალითად A* ალგორითმში (ლექცია #5), რაც მოით- ხოვს იმ ქმედებათა შეფასებას, რომელიც საჭიროა მიმდინარე მდგომარეობიდან მიზნობ- რივში გადასვლისათვის. გენეტიკური ალგორითმის მუშაობისათვის ასეთი ძალისხმევა საჭირო არ არის. უბრალოდ პოტენციალური ამოხსნების ყოველი თაობა ფასდება რაღაც ხარისხის კრიტერიუმის დახმარებით. აგრეთვე არ მოითხოვება შემდგომი მდგომარეო- ბების მკაცრი სისტემატიზაცია, რომელიც საჭიროა მდგომარეობების სივრცეში ძებნის დროს. უბრალოდ ფორმირდება ამონახსნ-კნადიდატების თაობა, რომელთაგან თითოე- ულს შეუძლია შეიტანოს თავისი წვლილი ახალ, შესაძლო ამონახსნის მიღებაში პარა- ლელური ძებნის პროცესის დროს.

გენეტიკური ალგორითმების ეფექტურობის მნიშვნელოვან წყაროს წარმოადგენს ევოლუციური ოპერატორების არაცხადი პარალელიზმი. მდგომარეობების სივრცეში ძებნ- ისგან განსხვავებით, მოცემულ შემთხვევაში ოპერაციები სრულდებიან პარალელურად პოტენციალური ამონახსნების მთელი ოჯახისათვის. სუსტი ამოხსნების რეპროდუქ- ტიული თვისებების შეზღუდვით, გენეტიკურ ალგორითმებს მივყავართ არამარტო ამ ამოხსნის, არამედ მთელი მისი შთამომავლების ამოგდებამდე. მაგალითად, თუ ალგო- რითმის მუშაობის პროცესში მოხდა

101#0##1

სტრიქონის გადაგდება, მაშინ მას აღარ შეუძლია დაბადოს

101#_ _ _ _

სახის სტრიქონი. თუ მშობელი ამონახსნი არ შეესაბამება ხარისხის კრიტერიუმს, მაშინ ალგორითმის მუშაობის პროცესში არ მოხდება მისი არცერთი პოტენციალური შთამომავ- ლის განხილვა.

საკონტროლო კითხვები:

1. რა ბიოლოგიური პროტოტიპი უდევს საფუძვლად გენეტიკურ ალგორითმებს?
2. რას წარმოადგენს ევოლუციის პროცესი?
3. რის საშუალებას იძლევიან გენეტიკური ალგორითმები?

Formatted: Tab stops: Not at 0.64" + 1.27" + 1.91" + 2.54" + 3.18" + 3.82" + 4.45" + 5.09" + 5.73" + 6.36" + 7" + 7.63" + 8.27" + 8.91" + 9.54" + 10.18"

4. რა სტადიებისაგან შედგება გენეტიკური ალგორითმი?
5. ჩამოაყალიბეთ ზოგადი გენეტიკური ალგორითმი.
6. ჩამოთვალეთ გენეტიკური ოპერაციები.
7. რა განსხვავებაა გენეტიკური ოპერაციების თვალსაზრისით ნკფ და კომივოიაჟერის ამოცანებს შორის?
8. რა არის გრეის კოდი და რისთვის იყენებენ მას?
9. რაში მდგომარეობს გენეტიკური ალგორითმების მნიშვნელოვანი უპირატესობა?
10. რა განსხვავებაა ევრისტიკული ძებნის ალგორითმებსა და გენეტიკურ ალგორითმებს შორის?
11. რას ნიშნავს არაცხადი პარალელიზმი?

ლექცია #14.
გენეტიკური ალგორითმების გამოყენება კლასიფიკაციის ამოცანებისათვის

14.1. წარმოდგენის ტიპები გენეტიკურ ალგორითმებში. გენეტიკური ალგორითმები სულ უფრო ხშირად გამოიყენება მათემატიკური მოდელირებისა და გამოყენებითი ამოცანების ამოსახსნელად, ამიტომ თავისთავად ცხადია, რომ გაიზარდა ინტერესი მათი თეორიული საფუძვლებისადმი. პირველ რიგში მნიშვნელოვანი გახდა იმ ამოცანების ტიპების დახასიათება, რომლებისთვისაც გენეტიკური ალგორითმები მუშაობენ უფრო ეფექტურად. ასევე არსებითია იმის დადგენა, თუ რა პირობებში მოქმედებენ გენეტიკური ალგორითმები უფრო ეფექტურად, ვიდრე ძებნის სხვა ინტელექტუალური მეთოდები.

ჩვენ ნკფ-ის მაგალითზე ვნახეთ, რომ ზოგიერთი ამოცანები კარგად აღიწერება ბიტური სტრიქონების დონეზე. ბიტური სტრიქონებით აღწერის შაბლონს უწოდებენ სქემას. მაგალითად, 10##01 სქემა წარმოადგენს ექვსბიტიანი სტრიქონების ოჯახს, რომლებიც იწყებიან 10 ფრაგმენტით და მთავრდებიან 01 სიმბოლოებით. რადგან ცენტრალური ფრაგმენტი ## –აღწერს ოთხ შესაძლო კომბინაციას 00, 01, 10, 11, ამიტომ მთელი სქემა წარმოადგენს ოთხ ბიტურ სტრიქონს, რომლებიც შედგებიან 1 და 0 –სიმბოლოებისგან. ტრადიციულად ითვლება, რომ ყოველი სქემა აღწერს ჰიპერსიბრტყეს. გენეტიკური ალგორითმების ტრადიციული თეორიის ცენტრალურ მომენტს წარმოადგენს მტკიცებულება იმის შესახებ, რომ მსგავსი სქემები წარმოადგენენ ამოხსნების ოჯახების სამშენებლო ბლოკებს. შეჯვარების და მუტაციის გენეტიკური ოპერატორები ოპერირებენ ასეთ სქემებთან პოტენციური ამონახსნების ძებნის პროცესში. ამ ოპერაციების აღწერა ხდება სქემების შესახებ თეორემის საშუალებით, რომლის თანახმადაც, წარმოებადობის გაზრდისათვის რაიმე გარემოში, ადაპტურმა სისტემამ უნდა მოახდინოს სქემების საშუალებით ფორმალიზებული ზოგიერთი სტრუქტურული თვისების იდენტიფიცირება, შემოწმება და რეალიზაცია.

ამ სქემების ანალიზი გულისხმობს, რომ ხარისხის კრიტერიუმის არჩევის ალგორითმი დაიყვანება ძებნის სივრცეში იმ ქვესიმრავლეების ძებნაზე, რომლებიც ყველაზე უკეთ აკმაყოფილებენ მოცემულ კრიტერიუმს. შესაბამისად, ეს ქვესიმრავლეები აღიწერებიან სქემებით, რომლებისთვისაც ხარისხის კრიტერიუმის მნიშვნელობა საშუალოზე მაღალია. შეჯვარების გენეტიკური ოპერატორის შესრულების დროს, ხარისხის მაღალი მაჩვენებლის მქონე სამშენებლო ბლოკები გროვდებიან ერთად და ახდენენ ‘‘გაუმჯობესებული’’ სტრიქონების ფორმირებას. მუტაციები იძლევიან იმის გარანტიას, რომ გენეტიკური თავისებურებები არ დაიკარგება ძებნის პროცესში, ანუ ეს ოპერატორები უზრუნველყოფენ ძებნის ზედაპირის ახალ მიდამოებში გადასვლას. ამგვარად,

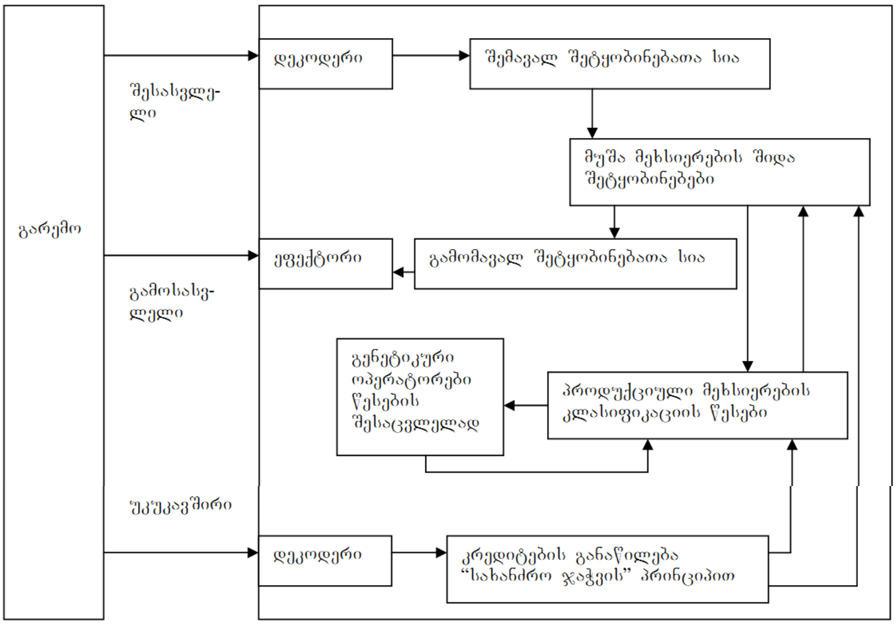
Formatted: Tab stops: Not at 0.64" + 1.27" + 1.91" + 2.54" + 3.18" + 3.82" + 4.45" + 5.09" + 5.73" + 6.36" + 7" + 7.63" + 8.27" + 8.91" + 9.54" + 10.18"
Formatted: Font: Not Bold
Formatted: Font: Sylfaen, Not Bold, Georgian (Georgia)
Formatted: Font: Not Bold

გენეტიკური ალგორითმები შეიძლება განვიხილოთ, როგორც ძებნის პროცესის რაიმე განზოგადება მნიშვნელოვანი გენეტიკური თვისებების შენარჩუნებით. დღეისათვის უკვე ცნობილია, რომ ამ ანალიზის შედეგები შეიძლება გავავრცელოთ წარმოდგენის უფრო მაღალი დონის, რთულ, არაბიტურ სქემებზეც.

14.2.კლასიფიკაციის სისტემები. პირველი გენეტიკური ალგორითმები თითქმის მთლიანად ბაზირდებოდნენ დაბალდონიან წარმოდგენებზე, კერძოდ, ბიტურ სტრიქონებზე (0,1,#). იმის გარდა, რომ ბიტური სტრიქონები წარმოადგენენ ბუნებრივ წარმოდგენებს გენეტიკური ოპერატორების რეალიზაციისთვის, ისინი უზრუნველყოფენ გენეტიკური ალგორითმებისათვის ისეთივე მაღალწარმოებადობას, როგორც სხვა არასიმბოლური მიდგომები, მათ შორის ნეიროქსელური მიდგომაც. მაგრამ არსებობენ ამოცანები, მაგალითად, კომივოიაჟერის პრობლემა, რომლებისთვისაც ბუნებრივია კოდირება წარმოდგენების უფრო მაღალ დონეზე. ჩვენ ეხლა განვიხილავთ გენეტიკური ალგორითმების გამოყენების მაგალითს კლასიფიკაციის ამოცანების გადასაჭრელად. კლასიფიკაციის სისტემებში გამოიყენება გენეტიკური სწავლება ლოგიკური დასკვნების გამოტანის წესებისადმი. კლასიფიკაციის სისტემა შეიცავს დასკვნების გამოტანის სისტემის სტანდარტულ ელემენტებს (სურ.#14.1.): დასკვნების გამოტანის წესები (კლასიფიკატორები), სამუშაო მეხსიერება, შემავალი გადამწოდები (ან დეკოდერები) და გამომავალი ელემენტები (ეფექტორები).

კლასიფიკაციის სისტემის განმასხვავებელ თვისებურებას წარმოდგენს კონფლიქტის მოგვარებისადმი კონკურენტული მიდგომა, სწავლებაში გენეტიკური ალგორითმის გამოყენება და ალგორითმის “სახანძრო ჯაჭვი” გამოყენება სწავლების პროცესში წახალისებების და სასჯელების გასანაწილებლად. გარე გარემოსთან უკუკავშირი იძლევა საშუალებას შეფასდეს კლასიფიკატორ-კანდიდატების ხარისხი, რაც აუცილებელია გენეტიკური სწავლებისათვის. ამოცანის ამოხსნის დროს კლასფიკატორი მუშაობს როგორც ლოგიკური დასკვნების გამოტანის ტრადიციული სისტემა. კლასიფიკაციის სისტემის დეტექტორს გარედან მიეწოდება შეტყობინება, მაგალითად ინფორმაცია მოთამაშის მიერ სვლის გაკეთების შესახებ. ეს ხდომილება კოდირდება და მოთავსდება დასკვნების გამოტანის სისტემის სამუშაო მეხსიერების შიდა შეტყობინებების სიაში, როგორც სახე. დასკვნების გამოტანის სისტემის ჩვეულებრივი მუშაობის დროს, ეს შეტყობინებები მონაცემების საფუძველზე შეესაბამებიან კლასფიკაციის წესების პირობებს. “უფრო მეტად აქტიური კლასიფიკატორის” არჩევა ხორციელდება აუქციონის სქემით, რომელშიც შემოთავაზებული ფასი – ეს არის ფუნქცია, რომელიც დამოკიდებულია ამ კლასიფიკატორის ხარისხის კრიტერიუმის აკუმულირებულ მნიშვნელობასა და შემავალ სტიმულსა და კლასიფიკატორის პირობას შორის შესაბამისობის დონეზე. ეს შეტყობინებები დაემატებიან შესაბამისობის ყველაზე უფრო მაღალი დონის მქონე კლასიფიკატორების სამუშაო მეხსიერებას. დასკვნების გამოტანის სისტემის მუშაობის პროცესში, შეტყობინებები განახლებული სიიდან შეიძლება ეფექტორების საშუალებით გადაეცეს გარემოს ან მათ შეიძლება გააქტიურონ კლასფიკაციის ახალი წესები.

Formatted: Indent: First line: 0.5"



სურ. #14.1 კლასიფიკაციის სიტემის ურთიერთქმედება გარემოსთან

~~კლასიფიკაციის სისტემის განმასხვავებელ თვისებურებას წარმოდგენს კონფლიქტის მოგვარებისადმი კონკურენტული მიდგომა, სწავლებაში გენეტიკური ალგორითმის გამოყენება და ალგორითმის “სახანძრო ჯაჭვი” გამოყენება სწავლების პროცესში წახალისებების და სასჯელების გასანაწილებლად. გარე გარემოსთან უკუკავშირი იძლევა საშუალებას შეფასდეს კლასიფიკატორ კანდიდატების ხარისხი, რაც აუცილებელია გენეტიკური სწავლებისათვის. ამოცანის ამოხსნის დროს კლასფიკატორი მუშაობს როგორც ლოგიკური დასკვნების გამოტანის ტრადიციული სისტემა. კლასიფიკაციის სისტემის დეტექტორს გარედან მიეწოდება შეტყობინება, მაგალითად ინფორმაცია მოთამაშის მიერ სვლის გაკეთების შესახებ. ეს ხდომილება კოდირდება და მოთავსდება დასკვნების გამოტანის სისტემის სამუშაო მეხსიერების შიდა შეტყობინებების სიაში, როგორც სახე. დასკვნების გამოტანის სისტემის ჩვეულებრივი მუშაობის დროს, ეს შეტყობინებები მონაცემების საფუძველზე შეესაბამებიან კლასფიკაციის წესების პირობებს. “უფრო მეტად აქტიური კლასიფიკატორის” არჩევა ხორციელდება აუქციონის სქემით, რომელშიც შემოთავაზებული ფასი — ეს არის ფუნქცია, რომელიც დამოკიდებულია ამ კლასიფიკატორის ხარისხის კრიტერიუმის აკუმულირებულ მნიშვნელობასა და შემავალ სტიმულსა და კლასფიკატორის პირობას შორის შესაბამისობის დონეზე. ეს შეტყობინებები დაემატებიან შესაბამისობის ყველაზე უფრო მაღალი დონის მქონე კლასიფიკატორების სამუშაო მეხსიერებას. დასკვნების გამოტანის სისტემის მუშაობის პროცესში, შეტყობინებები განახლებული სიიდან შეიძლება ეფექტორების საშუალებით გადაეცეს გარემოს ან მათ შეიძლება გააქტიურონ კლასფიკაციის ახალი წესები.~~

კლასიფიკაციის სისტემები ახდენენ წახალისებით სწავლების ერთ-ერთი ფორმის რეალიზაციას, ანუ იმის და მიხედვით, სწორი იყო თუ არა მათი გადაწყვეტილება, ხდება სისტემის წახალისება ან დასჯა ხარისხის კრიტერიუმის მნიშვნელობით განსაზღვრული,

Formatted: Tab stops: Not at 0.64" + 1.27" + 1.91" + 2.54" + 3.18" + 3.82" + 4.45" + 5.09" + 5.73" + 6.36" + 7" + 7.63" + 8.27" + 8.91" + 9.54" + 10.18"

Formatted: Tab stops: Not at 0.64" + 1.27" + 1.91" + 2.54" + 3.18" + 3.82" + 4.45" + 5.09" + 5.73" + 6.36" + 7" + 7.63" + 8.27" + 8.91" + 9.54" + 10.18"

ან მასწავლებლისგან მოსული ინფორმაციის საფუძველზე, დასწავლადი სისტემა ითვლის წესები-კანდიდატების პოპულაციისათვის ხარისხის კრიტერიუმის მნიშვნელობას და აგებს ახალ პოპულაციას გენეტიკური სწავლების ვარიანტებიდან ერთ-ერთის დახმარებით. კლასფიკაციის სისტემების სწავლება ხდება ორი ხერხით. პირველი ხერხი მდგომარეობს წახალისების სისტემის გამოყენებაში, რომელიც კლასიფიკაციის წესების ხარისხის ზომას აგებს სასარგებლო წესების გაძლიერების და არასწორი ქმედებების შესუსტების ხარჯზე. კრედიტების განაწილების ალგორითმი გადასცემს ჯილდების და ჯარიმების ნაწილს იმ კლასიფიკაციის ყოველ წევრს, რომლებმაც მიიღეს მონაწილეობა საბოლოო წესის ფორმირებაში. სხვადასხვა ჯილდოების ასეთი განაწილება ურთიერთმოქმედ კლასიფიკატორებს შორის, ხშირად რეალიზდება ‘‘სახანძრო ჯაჭვის’’ ალგორითმის დახმარებით. ეს ალგორითმი მოხსნის კრედიტების და ჯარიმების განაწილების პრობლემას იმ სისტემებისათვის, რომელთა გამოსასვლელი წარმოადგენს წესების ნაკადის მიმდევრობითი გამოყენების შედეგს. როგორ გავანაწილოთ ჯარიმები ამ ჯაჭვის წესებს შორის გამოსასვლელზე შეცდომის შემთხვევაში? ჯაჭვის რომელი წესი წარმოადგენს შეცდომის წყაროს: ბოლო თუ ერთ-ერთი წინამდებარეებიდან? ‘‘სახანძრო ჯაჭვის’’ ალგორითმი საშუალებას გვაძლევს გავანაწილოთ კრედიტები და ჯარიმები ამ მიმდევრობის წესებს შორის იმის მიხედვით, თუ რა წვლილი მიუძღვის თითოეულ წესს საბოლოო ამოხსნაში. ეს ალგორითმი ანალოგიურია წინა ლექციებში განხილული შეცდომათა უკუგავრცელების ალგორითმისა.

სწავლების მეორე ფორმა დაკავშირებულია თვით წესების მოდიფიკაციასთან, შეჯვარების და მუტაციის ტიპის გენეტიკური ოპერატორების საფუძველზე. ასეთი მიდგომების დროს გადარჩებიან საუკეთესო წესები, ხოლო მათი კომბინაციების შედეგად ფორმირდება ახალი კლასიფიკატორები.

კლასიფიკაციის ყოველი წესი შედგება სამი კომპონენტისაგან და მუშაობს როგორც დასკვნების გამოტანის ჩვეულებრივი სისტემა: რაღაც პირობის საფუძველზე მოწმდება მონაცემების შესაბამისობა სამუშაო მეხსიერების შიგთავსთან. სწავლების პროცესში გენეტიკურ ოპერატორებს შეუძლიათ მოახდინონ როგორც პირობების, ისე გამოტანის წესების მოქმედების მოდიფიკაცია. მეორე კომპონენტს, წესი – მოქმედება, შეუძლია შეცვალოს შეტყობინებათა შიდა სია პროდუქციული მეხსიერებაში. და ბოლოს ყოველ წესს შეესაბამება ხარისხის ზომა. როგორც უკვე იყო აღნიშნული, ეს პარამეტრი იცვლება წესის, როგორც წარმატებული ისე წარუმატებელი გამოყენების დროს. თავიდან ეს ზომა მიენიჭება ყველა წესს გენეტიკური ოპერატორის საშუალებით მათი შექმნის დროს. მაგალითად, ხარისხის ზომად შეიძლება იყოს ორი მშობლის ხარისხის საშუალო მნიშვნელობა.

ვაჩვენოთ მარტივ მაგალითზე კლასიფიკაციის სისტემის ამ კომპონენტების ურთიერთქმედება. დავუშვათ, ობიექტების ნაკრები, რომელთა კლასიფიკაციასაც ვახდენთ, აღიწერება ექვსი პარამეტრით (პირობებით $c_1, c_2, c_3, c_4, c_5, c_6$). დავუშვათ, აგრეთვე, რომ ამ პარამეტრებიდან თითოეული იღებს 5 სხვადასხვა მნიშვნელობას. ზოგადობის დაურღვევლად შეიძლება ყველა ატრიბუტის დასაშვები მნიშვნელობის აღწერა მთელი რიცხვებით {1,2,3,4,5}. ვთქვათ დასკვნების გამოტანის წესების თანახმად ეს ობიექტები იყოფიან 4 კლასად: A_1, A_2, A_3, A_4 ამგვარად, ყოველი კლასფიკატორი შეიძლება ავღწეროდ შემდეგი თანაფარდობით:

$$(c_1, c_2, c_3, c_4, c_5, c_6) \rightarrow A_i, \text{ სადაც } i = \overline{1,4} \quad (1)$$

სადაც თითოეული c_i პირობა იღებს მნიშვნელობას შესაბამისი პარამეტრის {1,2,..,5}

Formatted: Font: Sylfaen

მნიშვნელობათა დიაპაზონიდან. როგორც წესი, პირობას აგრეთვე შეიძლება შეესაბამებოდეს ყოველი პარამეტრის ნებისმიერი # მნიშვნელობა (ანალოგიურია კონცეპტების გამოთვლის მეთოდისა, სადაც პარამეტრმა შეიძლება მიიღოს ნებისმიერი მნიშვნელობა). 14.1 ცხრილ-ში მოყვანილია კლასიფიკატორების ნაკრები. ყურადღება

მიაქციეთ იმას, რომ განსხვავებული პირობების რამდენიმე შაბლონი შეიძლება შეესაბამებოდეს ერთ კლას, როგორც 1 და 2 წესში, ან ორი ერთნაირი შაბლონი შეიძლება შეესაბამებოდეს სხვადასხვა კლასს.

ცხრილი 11.2. პირობების ნაკრები სხვადასხვა კლასებისათვის.

პირობა(პარამეტრი)	მოქმედება(კლასი)	წესის ნომერი
(1###1#)	$\rightarrow A1$	1
(2##3##)	$\rightarrow A1$	2
(##43##)	$\rightarrow A2$	3
(1#####)	$\rightarrow A2$	4
(##43##)	$\rightarrow A3$	5

როგორც უკვე აღვნიშნეთ, კლასიფიკაციის სისტემა დაფუძნებულია ლოგიკური დასკვნების გამოტანის წესებზე. მოყვანილ მაგალითში გამოყენებულიკლასიფიკაციის წესების ერთადერთი განმასხვავებელი თვისება არის პირობების წარმოსადგენად ციფრებით და # სიმბოლოთი შედგენილი სტრიქონების გამოყენება. პირობების ასეთი წარმოდგენა უზრუნველყოფს გენეტიკური ალგორითმების გამოყენებას ლოგიკური გამოტანის წესებში. ვნახოთ, როგორ ხდება გენეტიკური დასწავლის გამოყენება კლასიფიკაციის სისტემებში.

Formatted: Tab stops: Not at 0.64" + 1.27" + 1.91" + 2.54" + 3.18" + 3.82" + 4.45" + 5.09" + 5.73" + 6.36" + 7" + 7.63" + 8.27" + 8.91" + 9.54" + 10.18"

Formatted: Tab stops: Not at 0.64" + 1.27" + 1.91" + 2.54" + 3.18" + 3.82" + 4.45" + 5.09" + 5.73" + 6.36" + 7" + 7.63" + 8.27" + 8.91" + 9.54" + 10.18"

სიმარტივისათვის განვიხილოთ სისტემის სწავლება მხოლოდ A_1 კლასისთვის. მივანიჭოთ შაბლონებს (სტრიქონებს) 1 ან 0 იმის და მიხედვით, შეესაბამებიან ისინი A_1 კლასს, თუ არა. შევნიშნოთ, რომ ასეთი გამარტივება არ ზღუდავს მსჯელობის ზოგადობას, რადგან ეს გამოთვლები შეიძლება გავავრცელოთ რამდენიმე კლასის სწავლების შემთხვევაზე. ამისათვის საკმარისია შემოვიტანოთ ვექტორი, რომელიც შეესაბამება კონკრეტულ პირობით შაბლონს. მაგალითად, კლასიფიკატორი 14.1. ცხრილიდან შეიძლება ავღწეროთ ასეთი სახით:

(1###1#) → (1000),

(2##3##) → (1000),

(1#####) → (0100),

(##43##) → (0110).

ამ მაგალითში ბოლო სტრიქონი შეესაბამება კლასიფიკაციის წესებს A_2 და A_3 კლასებისათვის და არა A_1 ან A_4. თუ შევცვლით 0 ან 1 კლასების ასეთი ვექტორული წარმოდგენებით, მაშინ შეიძლება შევაფასოთ კლასფიკაციისათვის წესების ეფექტურობა რამდენიმე კლასზე.

კლასიფიკაციის კორექტულობის განსაზღვრისათვის გამოვიყენოთ 14.1 ცხრილში მოცემული წესები. კერძოდ, განვიხილოთ ისინი, როგორც მასწავლებელი კლასიფიკაციის სისტემაში წესების ხარისხის შეფასებისათვის. როგორც უმრავლეს სწავლების გენეტიკურ სისტემებში, შემთხვევითად ავირჩიოთ წესების საწყისი პოპულაცია. პირობის ყოველ შაბლონს შევუსაბამოთ ხარისხის ან ძალის პარამეტრი (ნამდვილი რიცხვი [0,0; 1,0] დიაპაზონიდან). ძალის ამ s პარამეტრს გამოვთვლით ყოველი მშობლის წესის ხარისხის საფუძველზე წინა ისტორიის გათვალისწინებით. წესების დახმარებით სწავლების ყოველ ციკლში შევეცადოთ შესასვლელების კლასიფიცირებას და კლასიფიკაციის ხარისხის

შემოწ-მებას ხარისხის ზომის, ანუ მასწავლებლის დახმარებით. ვთქვათ, რომელიმე ბიჯზე მიღე-ბულია კლასიფიკაციის წესების როლზე კანდიდატების შემდეგი პოპულაცია, რომლის ყოველი –ელემენტისათვის 1 აღნიშნავს კლასიფიკაციის კორექტულ შედეგს, ხოლო -0 უთი-თებს არასწორ შედეგზე.

(###21#) → 1 s =0.6,

(##3##5) → 0 s =0.5,

(21####) → 1 s =0.4,

(#4###2) → 0 s =0.23.

დავუშვათ, გარემოდან შემოვიდა ახალი შეტყობინება (1 4 3 2 1 5) და მასწავლებელმა (14.1 ცხრილიდან პირველი წესის საფუძველზე) მოახდინა ამ ვექტორის კლასიფიცირება, როგორც დადებითი -მაგალითი A_1 კლასისათვის.– ვნახოთ

რა მოხდება ამ სახის გადაცემისას მუშა მეხსიერებაში, მისი კლასიფიკაციისას 4 წესს-კანდიდატის დახმა-რებით. ეს სახე შეესაბამება 1 და 2 წესებს. სათანადო წესებს—— შორის

კონფლიქტის მოგვარება ხდება კონკურენციის საფუძველზე. ჩვენს მაგალითში ყოველი ———წესის შესაბამისობის ხარი-სხი გამოითვლება როგორც ყოველი პარამეტრის შესაბამისობის ხარისხების და მოცე-მული წესის ხარისხის ზომის ნამრავლების ჯამი. განუსაზღვრელ პარამეტრს # შეესაბამება 0.5, ხოლო პარამეტრის ზუსტი შესაბამისობის შემთხვევაში მას მიენიჭება ხარისხი –1,0. ნორმირებისათვის მიღებული მნიშვნელობა იყოფა შემავალი ვექტორის სიგრძეზე. რად-გან პირველი კლასიფიკატორი მოცემული შემავალი ვექტორისათვის იძლევა ორ ზუსტ შესატყვის -და 4 განუსაზღვრელ პარამეტრს, ამიტომ შემავალ ვექტორთან მისი შესაბამი-სობის- საერთო —ხარისხი ———იქნება $(4\cdot 0{,}5+2\cdot 1)\cdot 0{,}6/6=0{,}4$. –მეორე –კლასიფიკატორისთვის ასევე არის ორი ზუსტი შესატყვისი და 4 განუსაზღვრელი პარამეტრი, ამიტომ მისი შესა-ბამისობის ხარისხი შეადგენს $0{,}33$. ჩვენს მაგალითში კონკურენციის პრინციპით იმარ-ჯვებს შესაბამისობის მაქსიმალური ხარისხის მქონე კლასიფიკატორი, მაგრამ უფრო რთულ ამოცანებში ალბათ სასურველია რაიმე ზღურბლის გათვალისწინება. ამგვარად, გა-იმარჯვა პირველმა წესმა, რომლის შესაბამისადაც წარმოდგენილი სახე მიეკუთვნება A_1 კლასს. რადგან ეს მოქმედება არის კორექტული, ამიტომ პირველი წესის ხარისხის ზომა იზრდება და იღებს 1-თან მიახლოებულ ახალ მნიშვნელობას. თუ ამ წესის შესრულების შედეგი იქნებოდა არაკორექტული, მაშინ ხარისხის ზომა შემცირდებოდა. თუ შედეგის მი-ღებისათვის სისტემაში მრავალჯერ სრულდებოდა წესების რაიმე ნაკრები, მაშინ წახა-ლისების განსაზღვრული ნაწილი უნდა მიიღოს ამ შედეგის მიღებაში მონაწილე ყველა წესმა. ხარისხის ზომის გადათვლის ზუსტი —პროცედურის განსაზღვრა დამოკიდებულია სისტემაზე და შეიძლება აღმოჩნდეს ძალიან რთული. ის შეიძლება აიგოს "სახანძრო ჯაჭვის" ალგორითმის ან კრედიტების განაწილების სხვა მეთოდის საფუძველზე.

წეს-კანდიდატების ხარისხის ზომის გამოთვლის შემდეგ სწავლების ალგორითმში გამოიყენება გენეტიკური ოპერატორები წესების შემდეგი თაობის შექმნისათვის. თავდა-პირველად შერჩევის პრინციპით ამოირჩევა ხარისხის –კრიტერიუმის –უფრო –მაღალი მნიშ-ვნელობის —მქონე წესების სიმრავლე. ეს ამორჩევა ეფუძნება ხარისხის ზომის მნიშვნელო-ბაზე, მაგრამ მან შეიძლება გაითვალისწინოს დამატებითი შემთხვევითი სიდიდეებიც. შემთხვევითობის ელემენტი უზრუნველყოფს ხარისხის ცუდი მაჩვენებლის მქონე წესების ამორჩევის შესაძლებლობას, რომლებსაც, ზოგადი შეუსაბამობის მიუხედავად, შეუძლიათ შემოიტანონ სასარგებლო ელემენტები ამოცანის ამოხსნაში. დავუშვათ, განხილულ მაგა-ლითში შემდგომი მუშაობისათვის არჩეულია კლასფიკაციის პირველი ორი წესი.

მეოთხე და მეხუთე ელემენტებს შორის შეჯვარების წერტილის შემთხვევით არჩე-ვის შემდეგ

(###2 I 1#) → 1 $s = 0.6$,

(##3# I #5) → 0 $s = 0.5$,

მივიღებთ შთამომავლებს

(##3# I 1#) → 0 $s = 0.53$,

(###2 I #5) → 1 $s = 0.57$.

თითოეული შთამომავლის ხარისხის საზომი არის მათი მშობლების ხარისხის მაჩვენებლების შეწონილი ფუნქცია. წონითი კოეფიციენტები განისაზღვრებიან შჯვარების წერტილის ადგილმდგომარეობით. პირველმა შთამომავალმა მიიღო 0,6 ხარისხის ზომის მქონე კლასიფიკატორისგან $1/3$ ინფორმაცია და 0,5 ხარისხის ზომის მქონე კლასიფიკატორისგან $2/3$ ინფორმაცია. ამიტომ მისი ხარისხის ზომა შეადგენს

$$(1/3 \cdot 0.6) + (2/3 \cdot 0,5) = 0,53 .$$

ანალოგიური მსჯელობით მივიღებთ, რომ მეორე შთამომავლის ხარისხის ზომა ტოლია 0,57. წესის შესრულების შედეგი (0 ან 1 მნიშვნელობა) განისაზღვრება იმის და მიხედვით, თუ რამდენი პარამეტრი შეესაბამება მოცემულ წესს. კლასიფიკაციის ტიპურ სისტემებში ეს ორი ახალი წესი, მათ მშობლებთან ერთად, შედის კლასიფიკატორთა იმ ნაკრებში, რომლებთანაც მუშაობს სისტემა შემდეგ ეტაპზე.

შეიძლება განვსაზღვროთ მუტაციის ოპერატორიც. მარტივი მუტაცია მდგომარეობს პარამეტრის ყოველი მნიშვნელობის დასაშვები დიაპაზონიდან ნებისმიერი მნიშვნელობით შეცვლაში. მაგალითად, 5 შეიძლება შევცვალოთ 1,2,3,4 ან # მნიშვნელობით. როგორც მითითებული იყო გენეტიკური ალგორითმის აღწერის დროს, მუტაციის ოპერატორები მოწოდებულნი არიან შეიტანონ შემთხვევითობის ელემენტი კლასიფიკატორის ძებნაში, მაშინ, როდესაც შეჯვარება საშუალებას იძლევა შევინახოთ შთამომავლებში მათი მშობლების შაბლონებიდან მონაცემების წარმატებული ფრაგმენტები.

განხილული მაგალითი საკმარისად მარტივია და მოწოდებულია მხოლოდ კლასიფიკაციის სისტემების ძირითადი კომპონენტების მუშაობის ილუსტრაციისათვის. რეალურ სისტემებში შეიძლება შეწყვილებულად იმუშაოს რამდენიმე წესმა და ყველა მათგანს შეუძლია გადასცეს თავისი შედეგები მუშა მეხსიერებას. ხშირად კლასიფიკატორებიდან ერთ-ერთის აშკარა დომინირების თავიდან აცილების მიზნით ამოხსნის პროცესში გამოიყენება ე.წ. დაბეგვრის სქემა, რომლის დროსაც ასეთი კლასიფიკატორის ხარისხის ზომა ყოველ წარმატებულ ბიჯზე რამდენადმე მცირდება. აქ არ არის აღწერილი აგრეთვე "სახანძრო ჯაჭვის" ალგორითმი, რომელიც ახდენს ამოცანის წარმატებით ამოხსნაში და გარემოსთვის შეტყობინებების გადაცემაში მონაწილე წესების სხვადასხვა ხარისხით წახალისებას. ამის გარდა, როგორც წესი, გენეტიკური ოპერატორები კლასიფიკატორების ტრანსფორმაციისათვის არ გამოიყენება ალგორითმის მუშაობის ყოველ ბიჯზე. ჩვეულებრივ არსებობს რაღაც ზოგადი პარამეტრი, რომელიც განისაზღვრება ყოველი ამოცანის სპეციფიკაციის გათვალისწინებით, კერძოდ, გარემოსთან უკუკავშირის ანალიზის საფუძველზე და რომლის საფუძველზეც მიიღება გადაწყვეტილება გენეტიკური ოპერატორების გამოყენების აუცილებლობის შესახებ.

Formatted: Tab stops: Not at 0.64" + 1.27" + 1.91" + 2.54" + 3.18" + 3.82" + 4.45" + 5.09" + 5.73" + 6.36" + 7" + 7.63" + 8.27" + 8.91" + 9.54" + 10.18"

14.3. პროგრამირება გენეტიკური ოპერატორების გამოყენებით. განვიხილოთ გენეტიკური ოპერაციების გამოყენება უფრო რთული შემთხვევებისათვის, კერძოდ, მათი გამოყენება კომპიუტერული პროგრამებისადმი. ამ დროს გენეტიკური პროგრამირების პრინციპები გამოიყენება კომპიუტერული პროგრამის იერარქიულად ორგანიზებული ფრაგმენტებზე. პროგრამის ხარისხი განისაზღვრება მისი უნარით ამოხსნას გარკვეული კლასის ამოცანა, ხოლო პროგრამის მოდიფიკაცია ხდება პროგრამის ქვეხეებისადმი შეჯვარების და მუტაციის გამოყენებით. გენეტიკური პროგრამირების დროს ძებნა მიმდინარეობს სხვადასხვა სირთულის და ზომის კომპიუტერული პროგრამების სივრცეში. ფაქტობრივად, ძებნის სივრცეს წარმოადგენს ამოცანის საგნობრივი არის შესაბამისი სიმბოლოების მიმდევრობებისგან და ფუნქციათა გამოძახებებისგან შედგენილი კომპიუტერული პროგრამების

ყველა შესაძლო სიმრავლე. ისევე, როგორც გენეტიკური დასწავლება, ასეთი ძებნა შეიცავს შემთხვევითობის ელემენტს, ის ბევრჯერ სრულდება "ბრმად", მაგრამ, რამდენადაც გასაკვირვი არ უნდა იყოს, არის საკმარისად ეფექტური. გენეტიკური პროგრამირების დროს თავდაპირველად ინიციალიზირდება შემთხვევითად გენერირებული პროგრამების პოპულაცია, რომლებიც შედგენილი არიან შესაბამისი პროგრამული ფრაგმენტებისაგან. ამოცანის საგნობრივ არეზე დამოკიდებულების მიხედვით ეს ფრაგმენტები შეიძლება შეიცავდნენ სტანდარტულ არითმეტიკულ ოპერაციებს, მათემატიკურ და ლოგიკურ ფუნქციებს და აგრეთვე სპეციფიკურ ფუნქციებს მოცემული საგნობრივი არისათვის. პროგრამული კომპონენტების რიცხვში შედიან სტანდარტული ტიპის მონაცემები: ლოგიკური, მთელი რიცხვითი, მცოცავმძიმიანი, ვექტორული, სიმბოლური ან მრავალმნიშვნელიანი.

ინიციალიზაციის შემდეგ გენერირდება პოპულაცია ათასამდე კომპიუტერული პროგრამიდან. ყოველი ახალი პროგრამა იქმნება გენეტიკური ოპერატორების გამოყენებით. კომპიუტერული პროგრამების შესქმნელად აუცილებლად უნდა მოხდეს შეჯვარების, მუტაციის და სხვა კვლავწარმოების უზრუნველმყოფი ოპერატორების ადაპტაცია. ყოველი ახალი პროგრამის ხარისხი განისაზღვრება მისი უნარით ამოხსნას ამოცანა კონკრეტული საგნობრივი არიდან. თვით ხარისხის კრიტერიუმიც ფორმირდება საგნობრივი არის გათვლისწინებით. ხარისხის კრიტერიუმის მაღალი მნიშვნელობის მქონე პროგრამები გადარჩებიან და მიიღებენ მონაწილეობას პროგრამების შემდეგი თაობის ფორმირებაში.

Formatted: Tab stops: Not at 0.64" + 1.27" + 1.91" + 2.54" + 3.18" + 3.82" + 4.45" + 5.09" + 5.73" + 6.36" + 7" + 7.63" + 8.27" + 8.91" + 9.54" + 10.18"

ამგვარად, გენეტიკური პროგრამირება შეიცავს შემდეგ ექვს კომპონენტს, რომელთაგან მრავალი გვაგონებს გენეტიკური ალგორითმების შემადგენელ ნაწილებს:

1. სტრუქტურების ნაკრები, რომლებიც ექვემდებარებიან ტრანსფორმაციას გენეტიკური ოპერატორების დახმარებით.
2. საწყისი სტრუქტურების ნაკრები, რომლებიც შეესაბამებიან საგნობრივ არეს.
3. ხარისხის ზომა ამ სტრუქტურების შესაფასებლად, რომელიც აირჩევა საგნობრივი არის გათვალისწინებით.
4. გენეტიკური ოპერატორების ნაკრები სტრუქტურების ტრანსფორმაციისათვის.
5. ყოველი თაობის ელემენტების პარამეტრების და მდგომარეობების აღწერა.
6. გაჩერების პირობების ნაკრები.

გენეტიკურ პროგრამირებაში ოპერაციები სრულდება იერარქიულად ორგანიზებულ პროგრამულ მოდულებზე. პროგრამული ენების კომპონენტების წარმოდგენის ძირითად საშუალებას წარმოადგენს ენა **LISP**-ი. პროგრამული ფრაგმენტები წარმოდგინება როგორც სიმბოლური გამოსახულებები **LISP** -ში, ან S-ს გამოსახულებები. გენეტიკური ოპერატორები ოპერირებენ S-ს გამოსახულებებთან. კერძოდ, ისინი ასახავენ S-ს გამოსახულებების ხისმაგვარ სტრუქტურებს (პროგრამის ფრაგმენტები **LISP** ენაზე) ახალ ხეებში (სხვა პროგრამები **LISP** ენაზე)..

Formatted: Tab stops: Not at 0.64" + 1.27" + 1.91" + 2.54" + 3.18" + 3.82" + 4.45" + 5.09" + 5.73" + 6.36" + 7" + 7.63" + 8.27" + 8.91" + 9.54" + 10.18"

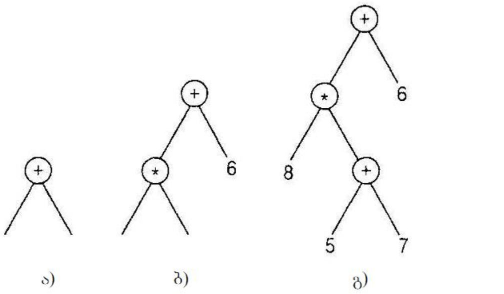
გენეტიკური პროგრამირება იძლევა სასარგებლო პროგრამების აგების საშუალებას იმ საწყისი მონაცემების და პრედიკატების საფუძველზე, რომლებიც აღწერენ ამოცანის საგნობრივ არეს. ამოცანების რომელიმე სიმრავლის ამოხსნისათვის

დანიშნული პროგრამების თაობისათვის განსაზღვრის არის მოცემისას, აუცილებელია თავდაპირველად გავაანალიზოთ თვით ამოხსნა და შემადგენელი ელემენტების შესაძლო საბოლოო მნიშვნელობები, აგრეთვე ფუნქციები, რომლებიც აუცილებელია ამ საბოლოო მნიშვნელობების მისაღებად. იმისათვის, რომ მოვახდინოთ გენეტიკური ოპერატორების დახმარებით მოდიფიცირებული სტრუქტურების ინიციალიზაცია, აუცილებელია შევქმნათ ორი სიმრავლე: ფუნქციების **F** სიმრავლე და საბოლოო მნიშვნელობების **T** სიმრავლე, რომლებიც აუცილებელია მოცემული საგნობრივი არისათვის. **F** სიმრავლე შეიძლება შედგებოდეს მარტივი ოპერაციებისაგან{+, -, *,/} ან შეიცავდეს უფრო რთულ ფუნქციებს, როგორიცაა $\sin(x)$, $\cos(x)$ და მატრიცული ოპერაციები. **T** სიმრავლე შეიძლება შეიცავდეს მთელ ან ნამდვილ რიცხვებს, მატრიცებს,

ან უფრო რთულ გამოსახულებებს. სიმბოლოების სიმ-რავლე **T** უნდა იყოს ჩაკეტილი **F** სიმრავლის ფუნქციების მიმართ. ამ სიმრავლეების შექმნის შემდეგ ხდება საწყისი “პროგრამების” გენერირება **F** და **T** სიმრავლეებიდან ელე-მენტების შემთხვევითად ამორჩევის გზით. **T** სიმრავლიდან ელემენტის ამორჩევის შე-დეგად, ჩვენ მივიღებთ გადაგვარებულ ხეს, რომელიც შედგება ერთ ~~ე~~ელემენტისგან. ამი-ტომ, უფრო საინტერესოა სიტუაცია, როდესაც თავდაპირველად ამოირჩევა ელემენტი **F** -დან, ვთქვათ, ოპერაცია “+”. ამ შემთხვევაში მიიღება ხის საწყისი კვანძი ორი პოტენ-ციალური შთამომავლით. დავუშვათ, ამის შემდეგ პირველი შთამომავლის როლში ინი-ციალიზატორი ამოირჩევს **F** სიმრავლიდან გამრავლების “*” ოპერაციას, ხოლო მეორე შთამომავლის როლში კი **T** სიმრავლიდან საბოლოო მნიშვნელობას 6-ს. შემდეგი შემთხვევითად ამორჩეული ელემენტები შეიძლება გახდნენ საბოლოო მნიშვნელობა 8 და “+” ფუნქცია **F**-დან. დავუშვათ, პროცედურა მთავრდება **T** ––სიმრავლიდან 5 და 7 მნიშვნელობების ამორჩევით.

Formatted: Font: Sylfaen, Georgian (Georgia)

ასეთი სახით შემთხვევითად გენერირებული პროგრამა წარმოდგენილია სურ. #14.2.



სურ. №14.2 საწყისი პროგრამის შემთხვევითი გენერაცია.
წრეებით აღნიშნული კვანძები შეესაბამებიან ფუნქციებს

სურ. #14.2. ა) ნაჩვენებია ხე, “+” ოპერაციის პირველი არჩევის შემდეგ, სურ. #14.2. ბ) მოცემულია ხე 6-სის ტოლი საბოლოო მნიშვნელობის არჩევის შემდეგ, ხოლო სურ. #14.2 გ) კი – ნაჩვენებია საბოლოო პროგრამა.

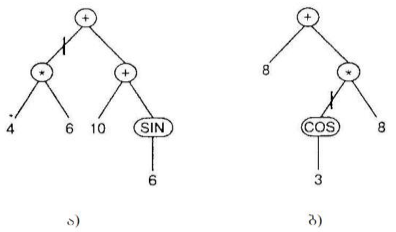
Formatted: Tab stops: Not at 0.64" + 1.27" + 1.91" + 2.54" + 3.18" + 3.82" + 4.45" + 5.09" + 5.73" + 6.36" + 7" + 7.63" + 8.27" + 8.91" + 9.54" + 10.18"

გენეტიკური პროგრამირების პროცესის ინიციალიზაციისათვის ფორმირდება მსგა-ვსი პროგრამების მთელი თაობა. ამ პოპულაციის განზომილების კონტროლისათვის შეიძ-ლება გამოვიყენოთ შეზღუდვები, მაგალითად პროგრამების მაქსიმალური სიღრმის შეზ-ღუდვა. არსებობს ასეთი –შეზღუდვების და აგრეთვე –საწყისი –პოპულაციების გენერირების სხვადასხვა მეთოდები, რომლებიც შეიძლება გამოვიყენოთ გენეტიკური პროგრამირების პროცესის ინიციალიზაციისათვის.

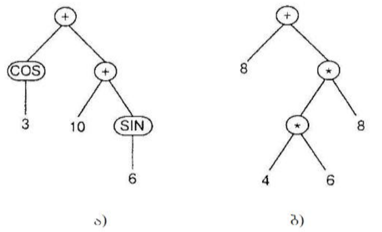
აქამდე განვიხილავდით წარმოდგენის საკითხებს (*S*-გამოსახულებები და ხისე-ბური სტრუქტურები), რომლებიც აუცილებელია პროგრამების ევოლუციის პროცესების გენერირებისათვის. იმისათვის, რომ ვმართოთ პროგრამათა პოპულაციები, საჭიროა განვ-საზღვროთ ხარისხის კრიტერიუმები. ხარისხის კრიტერიუმის არჩევა დამოკიდებულია კონკრეტულ ამოცანაზე და ჩვეულებრივ დაიყვანება იმ ამოცანების ნაკრების განსაზ-ღვრზე, რომლებიც უნდა ამოხსნას ევოლუციონირებადმა პროგრამამ. თვითონ ხარისხის კრიტერიუმი – ეს არის ფუნქცია, რომელიც ითვლის მოცემული პროგრამით ამოცანის ამოხსნის ეფექტურობას. ხარისხის წრფივი კრიტერიუმი ითვალისწინებს პროგრამის შესრულების შედეგსა და რეალურ ამოხსნას შორის განსხვავებას. მაშასადამე, ხარისხის წრფივი კრიტერიუმი შეიძლება განვიხილოთ როგორც შეცდომების ჯამი ამოცანების

ნაკრების დროს. შესაძლოა აგრეთვე ხარისხის სხვა კრიტერიუმები. ხარისხის ნორმალიზებული კრიტერიუმი გულისხმობს ხარისხის საერთო კრიტერიუმის გაყოფას ყველა შესაძლო შეცდომის რაოდენობაზე. მაშასადამე, ხარისხის ნორმალიზებული კრიტერიუმი შეიძლება იცვლებოდეს 0-დან 1-მდე დიაპაზონში. ნორმალიზაციის უპირატესობა ვლინდება პროგრამების დიდი პოპულაციის მუშაობის დროს. ხარისხის ზომა თვალისწინებს პროგრამის ზომას, მაგალითად, შეიძლება წახალისდეს მცირე ზომის კომპაქტური პროგრამების შექმნა.

გენეტიკური ოპერატორები პროგრამების სივრცეში შეიცავენ როგორც თვით ხის გარდაქმნას, ისე სხვადასხვა ხეებს შორის ფრაგმენტების გაცვლას. განვიხილოთ ორი ძირითადი გენეტიკური ოპერაცია, რომლებმაც მიიღეს კვლაწარმოების და შეჯვარების სახელწოდება. კვლავწარმოება არის მიმდინარე თაობის პროგრამების ამორჩევა და მათი კოპირება (ცვლილებების გარეშე) შემდეგი თაობის პოპულაციაში. შეჯვარება გულისხმობს ორი არსებული პროგრამის ქვეხეების გაცვლას. დავუშვათ, არსებობს სურ. #14.3 წარმოდგენილი ორი მშობელი პროგრამა, ა) და ბ) რომლებშიც “I” სიმბოლოთი ნაჩვენებია შეჯვარების წერტილები. შვილობილი ხეები, რომლებიც მიიღებიან შეჯვარების შედეგად, გამოსახული არიან სურ. #14.4. შეჯვარება შეიძლება გამოვიყენოთ აგრეთვე ერთი მშობელი ხის გარდაქმნისათვის მისი ორი ქვეხის ადგილების გაცვლის გზით. შეჯვარების წერტილის შემთხვევითად არჩევის დროს ორმა იდენტურმა მშობელმა ხემ შეიძლება მოგვცეს სხვადასხვა შთამომავალი. შეჯვარების წერტილი აგრეთვე შეიძლება იყოს პროგრამის ძირეული კვანძი.



სურ. №14.3. ორი პროგრამა, რომლებიც ექვემდებარებიან შეჯვარებას შემთხვევითად შერჩეულ წერტილში.



ნახ. 11.6. შვილობილი პროგრამები, რომლებიც მიიღებიან შეჯვარების შედეგად.

ცნობილია პროგრამების ხეების მრავალი ნაკლებად გავრცელებული და ნაკლებად სასარგებლო გენეტიკური გარდაქმნები. მათ მიეკუთვნება მუტაცია, რომლის დროსაც პროგრამის სტრუქტურაში უბრალოდ შეაქვთ შემთხვევითი ცვლილებები (მაგალითად, საბოლოო მნიშვნელობის შეცვლა სხვა მნიშვნელობით ან ქვეხით). გადანაცვლება, რომელიც სტრიქონების ინვესტირების ანალოგიურია. ეს ოპერაციაც სრულდება ცალკეული პროგრამებისთვის და მდგომარეობს საბოლოო სიმბოლოების ან ქვეხეების ადგილების გაცვლაში.

პროგრამების მიმდინარე შევსება ასახავს ამოხსნის მდგომარეობას. არ არსებობენ სპეციალური მეთოდები, რომლებიც მოგვცემდნენ საშუალებას გავითვალისწინოთ ხარისხის კრიტერიუმით განსაზღვრული ზედაპირის რელიეფი ევლოუციური პროცესის დროს. ამ თვლსაზრისით გენეტიკური პროგრამირება გვაგონებს ექსტრემუმის ძებნის “ხარბ” ალგორითმს, რომელიც განვიხილეთ ძებნის ინფორმირებული ალგორითმების განხილვის დროს. გენეტიკური პროგრამირება არის ევოლუციის ბუნებრივი პროცესის მსგავსი, რადგან პროგრამების ტრანსფორმაცია შეიძლება შესრულდეს უწყვეტად. მიუხედავად ამისა, დროის და გამოთვლითი რესურსების შეზღუდულობა თხოულობს გაჩერების პირობების დაფიქსურებას. ასეთი პირობები ჩვეულებრივ დამოკიდებულია ხარისხის კრიტერიუმის მნიშვნელობაზე და მოხმარებულ გამოთვლით რესურსებზე. რადგან გენეტიკური პროგრამირება არის კომპიუტერული პროგრამების გენერირების მეთოდი,

Formatted: Tab stops: Not at 0.64" + 1.27" + 1.91" + 2.54" + 3.18" + 3.82" + 4.45" + 5.09" + 5.73" + 6.36" + 7" + 7.63" + 8.27" + 8.91" + 9.54" + 10.18"

ამიტომ ის შეიძლება მივაკუთნოთ ავტომატური პროგრამების სფეროს. ფრაგმენტული ინფორმაციის საფუძველზე კომპიუტერული პროგრამების ავტომატური შექმნის პრობლემაზე მკვლევარები მუშაობენ ხელოვნური ინტელექტის განვითარების დაწყებიდან. გენეტიკური პროგრამირება შეიძლება განვიხილოთ, როგორც ერთ-ერთი საშუალება გამოკვლევების ამ მნიშვნელოვან სფეროში.

საკონტროლო კითხვები:

Formatted: Tab stops: Not at 0.64" + 1.27" + 1.91" + 2.54" + 3.18" + 3.82" + 4.45" + 5.09" + 5.73" + 6.36" + 7" + 7.63" + 8.27" + 8.91" + 9.54" + 10.18"

1. რას ეწოდება სქემა?
2. რაში მდგომარეობს სქემების ანალიზი?
3. რა ელემენტებს შეიცავს კლასიფიკაციის სისტემა?
4. რა მიზნით გამოიყენება კლასიფიკაციის სისტემაში გენეტიკური ოპერაციები?
5. რას ნიშნავს სწავლა "წახალისებით"?
6. რისი ანალოგია "სახანძრო ჯაჭვის" ალგორითმი?
7. როგორ გამოითვლება ხარისხის ზომა, რომელიც შეესაბამება მაკლასიფიცირებელ წესს?
8. როგორ გამოითვლება მაკლასიფიცირებელი წესის შესაბამისობის ხარისხი?
9. რას ნიშნავს გენეტიკური პროგრამირებია?
10. როგორ განისაზღვრება პროგრამის ხარისხი?
11. რა კომპონენტებს შეიცავს გენეტიკური პროგრამირება?
12. როგორ ხდება საწყისი "პროგრამების გენერირება?
13. რაში მდგომარეობს ხარისხის წრფივი კრიტერიუმი?
14. რა ძირითადი გენეტიკური ოპერაციები გამოიყენება გენეტიკურ პროგრამირებაში?
15. რატომაა ნაკლებად სასარგებლო მუტაციის ოპერაციის გამოყენება გენეტიკურ პროგრამირებაში?

~~ლაბორატორიული სამუშაოები~~

~~ლაბორატორიული სამუშაო 1.~~
~~ლოგიკური ამოცანები. თამაშები. ინტელექტუალური თამაშები. არაკორექტული ამოცანები.~~

~~ლაბორატორიული სამუშაო 2.~~
~~ამოცანების წარმოდგენა მდგომარეობათა სივრცეში.~~

ლაბორატორიული სამუშაო 3.

რვიანის ამოცანა (თამაში)--ამოცანის ამოხსნა სხვადასხვა არაინფორმირებული ალგორითმებით.

ლაბორატორიული სამუშაო 4.

ევრისტიკები. სხვადასხვა შემფასებელი ფუნქციების შედგენა რვიანისა და ლაზიერის ამოცანებისათვის.

ლაბორატორიული სამუშაო 5.

ამოცანები შეზღუდვათა დაკმაყოფილებაზე. რუქის გაფერადების ამოცანა.

ლაბორატორიული სამუშაო 6.

ძებნა უკანდაბრუნებით.ხარისხოვანი ევრისტიკა.

ლაბორატორიული სამუშაო 7.

ძებნა წინააღმდეგობის შემთხვევაში. მინი მაქსისა და ალფა-ბეტა პროცედურების გამოკვლევა და ერთმანეთთან შედარება.

ლაბორატორიული სამუშაო 8.

ცოდნის წარმოდგენა სხვადასხვა მეთოდებით და მათი შედარებითი ანალიზი.

ლაბორატორიული სამუშაო 9.

ამოცანების ამოხსნის ძლიერი მეთოდები. ექსპერტული სისტემის გარსის შექმნა.

ლაბორატორიული სამუშაო 10.

პროდუქციული სისტემების გამოყენების კონკრეტული მაგალითები.

ლაბორატორიული სამუშაო 11.

ნეირონული ქსელების გამოყენება კომპიუტერულ თამაშებში.

ლაბორატორიული სამუშაო 12.

დიოფანტეს განტოლების ამოხსნა გენეტიკური ალგორითმებით.