ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტი

ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტი

ცხრილის ამოცანის ეფექტური ალგორითმი და პროგრამული რეალიზაცია თბილისის სახელმწიფო უნივერსიტეტის მაგალითზე

მაგისტრანტი: სოფიო ჩიქვინიძე

სამაგისტრო პროგრამა: კომპიუტერული მეცნიერება

ხელმძღვანელი: ალექსანდრე გამყრელიძე, ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტის სრული პროფესორი

ნაშრომი შესრულებულია კომპიუტერული მეცნიერების მაგისტრის აკადემიური ხარისხის მოსაპოვებლად

თბილისი, 2018 წელი

შინაარსი

ანოტაცია

სამაგისტრო ნაშრომის მიზანია საგამოცდო ცხრილის შესადგენი ეფექტური პროგრამული უზრუნველყოფის შექმნა.

ნაშრომში განხილულია საგამოცდო ცხრილის შედგენის ამოცანა და მისი შესაძლო სახესხვაობები და წარმოდგენილია საგამოცდო ცხრილის შედგენის ჰიბრიდული ალგორითმი. ნაშრომის ფარგლებში, ჰიბრიდული ალგორითმის გამოყენებით, შექმნილია საგამოცდო ცხრილის შესადგენი დესკტოპ აპლიკაცია, რომელიც მორგებულია თბილისის სახელმწიფო უნივერსიტეტის მაგალთს. პროგრამული უზრუნველყოფის მუშაობა გატესტილია თბილისის სახელმწიფო უნივერსიტეტის საგამოცდო ცენტრის მიერ მოწოდებული მონაცემებით.

Annotation

The purpose of the master's thesis is the development of effective software for solving examination timetabling problem.

Examination timetabling problem and its variations are discussed and a hybrid algorithm for solving the examination timetabling problem is presented. Desktop application for creating an examination timetable is created using the hybrid algorithm. Application is fitted to be used for Tbilisi State University. Software is tested using data obtained from the Examination Center of Tbilisi State University.

შესავალი

უმაღლესი საგანმანათლებლო დაწესებულებებისათვის საგამოცდო ცხრილის შედგენის სირთულე დღითი-დღე მატულობს. მთელს მსოფლიოში უნივერსიტეტები ყოველ წელს უფრო და უფრო მეტ სტუდენტს იღებენ და სთავაზობენ მათ ახალ საგანმანათლებლო პროგრამებს. დამატებით სირთულეებთან უწევთ გამკლავება ისეთ უმაღლეს საგანმანათლებლო დაწესებულებებს, რომლებიც სტუდენტებს ერთზე მეტი სპეციალობის დაუფლების საშუალებას აძლევენ კომბინირებული საგანმანათლებლო პროგრამების მეშვეობით. აქედან გამომდინარე, საგამოცდო ცხრილის შედგენა კომბინატორული ოპტიმიზაციის რთული ამოცანაა, რომლის გადაჭრაც ადამიანური რესურსებით შეუძლებელია. უნივერსიტეტებისათვის მისაღები ცხრილების შესადგენად შესაბამისი ალგორითმებია საჭირო.

საგამოცდო ცხრილის შედგენის პოლინომიალური ალგორითმი ცნობილი არ არის. უფრო მეტიც - საგამოცდო ცხრილის შედგენის ამოცანის უნივერსალური განსაზღვრებაც კი არ არსებობს, რადგან ყველა უმაღლეს საგანმანათლებლო დაწესებულებას განსხვავებული მოთხოვნები და საჭიროებები აქვს საგამოცდო ცხრილთან მიმართებაში. შესაბამისად, ყოველი ცაკლეული უნივერსიტეტისთვის საგამოცდო ცხრილის შემდგენი პროგრამული უზრუნველყოფის შექმნისას სხვადასხვა მიდგომების გამოყენება შეიძლება იყოს საჭირო.

ეს ნაშრომი ძირითადად დაფუძნებულია მელბურნის უნივერსიტეტის თანამშრომლების კვლევაზე[1], რომელთაც შეიმუშავეს ჰიბრუდული ალგორითმი მელბურნის უნივერსიტეტისთვის. მელბურნის უნივერსიტეტის მიერ საგამოცდო ცხრილის მიმართ დაწესებული შეზღუდვები და მოთხოვნები, ისევე როგორც უნივერსიტეტში სტუდენტებისა და გამოცდების რაოდენობა ახლოსაა თბილისის სახელმწიფო უნივერსიტეტში არსებულ ვითარებასთან, რითიც შეიძლება ვივარაუდოთ, რომ მსგავსი მიდგომა კარგად იმუშავებს თბილისის სახელმწიფო უნივერსიტეტის მაგალითზეც. წარმოდგენილ ნაშრომში სწორედ ეს ვარაუდია შემოწმებული.

საგამოცდო ცხრილის შედგენის ამოცანა და მისი სირთულე

საგამოცდო ცხრილის შედგენის ამოცანის ამოხსნა ძირითადად გულისხმობს ყოველი მოცემული გამოცდისათვის სესიისა და ოთახის შერჩევას ისე, რომ დაკმაყოფილებული იყოს გარკვეული შეზღუდვები. ამოცანის ამონახსნი უნდა იყოს კონკრეტული საგანმანათლებლო დაწესებულებისათვის მისაღები და ხარისხიანი საგამოცდო ცხრილი. ყოველ საგანმანათლებლო დაწესებულებას საგამოცდო ცხრილთან დაკავშირებული უნიკალური შეზღუდვები აქვს. გარდა ამისა, საგამოცდო ცხრილის ხარისხის ცნება სხვადასხვა მნიშვნელობას ატარებს სხვადასხვა დაწესებულებაში. ზოგიერთ შემთხვევაში ნებისმიერი შესრულებადი საგამოცდო ცხრილი მისაღებია, ზოგჯერ კი მოთხოვნილი ან სასრუველია, რომ ცხრილი გარკვეულ თვისებებს აკმაყოფილებდეს. ამ ყველაფრის გათვალისწინებით, საგამოცდო ცხრილის შედგენის ამოცანის უნივერსალური განსაზღვრება არ არსებობს, თუმცა ცნობილია საგანმანათლებლო დაწესებულებების მიერ მოთხოვნილ შეზღუდვათა ძირითადი ფორმები. ესენია:

1. დამთხვევა: ერთ სტუდენტს არ უნდა ჰქონდეს ორი ან მეტი გამოცდა ერთსა და იმავე სესიაში.
2. ტევადობა: მოცემულ ოთახში არც ერთ სესიაში არ უნდა განთავსდეს იმაზე მეტი სტუდენტი, ვიდრე ოთახი იტევს.
3. საერთო ტევადობა: არც ერთ მოცემულ სესიაში არ უნდა იყოს განთავსებული გარკვეულ რაოდენობაზე მეტი სტუდენტი.
4. გამოცდების ტევადობა: შეზღუდულია ერთ სესიაში გამოცდების მაქსიმალური დასაშვები რაოდენობა.
5. გამოცდის დანიშვნის შესაძლებლობა: ზოგი გამოცდა მხოლოდ სესიების შეზღუდულ სიმრავლეში შეიძლება ჩატარდეს.
6. ოთახის გამოყენების შესაძლებლობა: ზოგი ოთახი მხოლოდ სესიების შეზღუდულ სიმრავლეში შეიძლება იყოს გამოყენებული.
7. შეზღუდვა გამოცდების წყვილზე: გამოცდების ზოგიერთი წყვილი უნდა აკმაყოფილებდეს გარკვეულ შეზღუდვებს ერთმანეთის მიმართ (მაგალითად, ერთი აუცილებლად მეორეზე ადრე უნდა ჩატარდეს).

8. გამოცდებისა და ოთახების შესაბამისობა: ზოგი გამოცდა უნდა ჩატარდეს სპეციალურ ოთახში.
9. სტუდენტის შეზღუდვები: შეზღუდვები სტუდენტების ინდივიდუალურ საგამოცდო ცხრილზე (მაგალითად, სტუდენტს არ უნდა ჰქონდეს 2 ან მეტი გამოცდა 3 მიმდევრობით სესიაში).
10. შეზღუდვა დიდ გამოცდებზე: გამოცდები, რომლებზეც ბევრი სტუდენტია დარეგისტრირებული, საგამოცდო პერიოდის დასაწყისში უნდა ჩატარდეს (მაგალითად, გამოცდები რომლებზეც 500 ან მეტი სტუდენტია დარეგისტრირებული უნდა ჩატარდეს პირველ 10 სესიაში).

ყოველი საგანმანათლებლო დაწესებულება აწესებს ზემოთ მოყვანილი შეზღუდვებიდან რამდენიმეს ან ყველას. შეზღუდვების ზუსტი ფორმა დამოკიდებულია დაწესებულებებზე, რომლებმაც ზოგიერთი შეზღუდვა შეიძლება შემოიღონ, როგორც „მსუბუქი“ შეზღუდვა. მსუბუქი შეზღუდვების დაკმაყოფილება სასურველია, მაგრამ არა აუცილებელი. საგამოცდო ცხრილის ხარისხის მაჩვენებელი, როგორც წესი, სწორედ იმაზეა დამოკიდებული, თუ რამდენ შემთხვევაშია დაკმაყოფილებული მსუბუქი შეზღუდვები.

ზოგიერთი დაწესებულებისათვის გამოცდებისთვის ოთახების მინიჭება მეორეხარისხოვანი პრობლემაა: ოთახები შეიძლება დიდი ტევადობის იყოს ან მარტივად იყოს შესაძლებელი მოცემულ გამოცდაზე დარეგისტრირებული სტუდენტების რამდენიმე ოთახში გადანაწილება. ასეთ შემთხვევებში საგამოცდი ცხრილის შედგენისას შეგვიძლია გავითვალისწინოთ მხოლოდ საერთო ტევადობის შეზღუდვა, გამოცდებისათვის ოთახების შერჩევა კი მათი სესიებში გადანაწილების შემდეგ, ცალკე აქტივობად შეგვიძლია განვიხილოთ. ხაზს ვუსვამთ, რომ ეს არ ეხება ყველა საგანმანათლებლო დაწესებულებას.

საგამოცდო ცხრილის შედგენის ამოცანის ოპტიმალური ამოხსნა ჯერ-ჯერობით ცნობილი არ არის. მეტიც - ის NP-სრული ამოცანაა.

სამ ეტაპიანი მეთოდი

ჩვენ განვიხილავთ საგამოცდო ცხრილის შედგენის ამოცანის ისეთ ვარიანტს, რომელშიც ყოველ გამოცდას უნდა შევუსაბამოთ ერთი სესია. გამოცდების ოთახებში გადანაწილება ჩვენი მიზანი არ არის. წინა თავში განხილული შეზღუდვებიდან ვიყენებთ მხოლოდ სამს: დამთხვევის, გამოცდების დანიშვნის შესაძლებლობისა და საერთო ტევადობის შეზღუდვებს. ჰიბრიდული ალგორითმი, რომლითაც ვცდილობთ საგამოცდო ცხრილის შედგენის ამოცანის გადაჭრას, შედგება 3 ეტაპისაგან:

1. შეზღუდვების დაპროგრამება (constraint programming): ისეთი ცხრილის შესადგენად, რომელიც ყველა შეზღუდვას აკმაყოფილებს.
2. ე.წ. ფოლადის წრთობის სიმულაცია (simulated annealing): ცხრილის ხარისხის გასაუმჯობესებლად.
3. ლოკალური ძებნა (Hill climbing): მეორე ეტაპის შედეგად მიღებული ცხრილის მეტად დასახვეწად.

მოცემული ჰიბრიდული ალგორითმის პირველი ეტაპის მიზანია ისეთი საწყისი ცხრილის შედგენა, რომელიც ყველა შეზღუდვას დააკმაყოფილებს. როგორც ქვემოთ ვნახავთ, ჩვენ მიერ გამოყენებული მიდგომით, პირველი ეტაპი გამოყენებული სესიების რაოდენობის მინიმიზაციას ცდილობს. მართალია, შეზღუდვების დაპროგრამების შედეგად მიღებული ცხრილი ყველა მოთხოვნას აკმაყოფილებს, მაგრამ, გარკვეულ შემთხვევებში, მისი დასრულების შემდეგ ზოგი გამოცდა დაუგეგმავი რჩება, ანუ, ვერც ერთ სესიაში ვერ თავსდება.

მეორე და მესამე ეტაპების მიზანია ცხრილის ხარისხის გაუჯობესება და რაც შეიძლება მეტი გამოცდის დანიშვნა მოცემულ სესიებში. მეთოდები, რომლებიც ამ ორი ეტაპის დროს გამოიყენება ოპტიმიზაციის მეთოდებია, რომლებიც მოცემული სამიზნე ფუნქციის მინიმიზაციას ცდილობს. ამისათვის გამოიყენება სამიზნე ფუნქცია, რომელიც ზემოთ მოცემულ ორივე მიზანს ითვალისწინებს. საგამოცდო ცხრილის ხარისხის მაჩვენებელი, და, შესაბამისად, სამიზნე ფუნქცია სხვადასხვა მონაცემებისათვის სხვადასხვაა, მაგრამ, როგორც წესი, სამიზნე ფუნქცია წარმოადგენს სტუდენტების ინდივიდუალურ საგამოცდო ცხრილში

გამოცდების ერთმანეთთან სიახლოვისათვის დაწესებული ჯარიმების კომბინაციას. ჩვენ მიერ გამოყენებულ კონკრეტულ სამიზნე ფუნქციას შემდგომში განვიხილავთ.

შესაძლებელია, რომ ჰიბრიდული ალგორითმის სამივე ეტაპის დასრულების შემდეგ კვლავ დარჩეს ისეთი გამოცდები, რომლებსაც სესია ვერ მოეძებნა. ასეთ შემთხვევაში, უსესიოდ დარჩენილი გამოცდების დანიშვნა შესაძლოა მოხერხდეს მარტივი ხარბი ევრისტიკით, რომელსაც შემდგომში განვიხილავთ.

შეზღუდვების დაპროგრამება

ჰიბრიდული ალგორითმის პირველი ეტაპი - შეზღუდვების დაპროგრამება, გამოიყენება იმისათვის, რომ მივიღოთ საწყისი ცხრილი, რომელიც ყველა შეზღუდვას დააკმაყოფილებს. შეზღუდვების დაპროგრამების მოდელი განისაზღვრება შემდეგნაირად - ვიყენებთ ქვემოთ მოცემულ აღნიშვნებს:

- $E = \{1, \ldots, n\}$ - აღნიშნავს მოცემული n გამოცდის სიმრავლეს.
- S_i - i-ურ გამოცდაზე დარეგისტრირებული სტუდენტების რაოდენობა, ყოველი $i \in E$-სათვის.
- $T = \{1, \ldots v\}$ - აღნიშნავს სესიების v ელემენტიან სიმრავლეს.
- $R \subset E \times T$ - წარმოადგენს შემდეგი სახის შეზღუდვას: $(a, b) \in R$ მიუთითებს იმაზე, რომ გამოცდა a ვერ დაინიშნება სესია b-ში.
- C_t - t სესიის საერთო ტევადობა, ანუ სტუდენტების მაქსიმალური რაოდენობა, რომლებსაც ამ სესიაში შეუძლიათ გამოცდის დაწერა, ყოველი $t \in T$-სთვის.
- D_{ij} - იმ სტუდენტების რაოდენობა, რომლებიც დარეგისტრირებულები არიან როგორც i-ურ, ასევე j-ურ გამოცდაზე ყოველი $i, j \in E$-სთვის.
- ცვლადი $x_i \in T$ - აღნიშნავს i-ური გამოცდისათვის შერჩეულ სესიას ყოველი $i \in E$-სთვის.

აღსანიშნავია, რომ სესიები გადანომრილია დაწყების დროების ზრდადობის მიხედვით. ანუ, $t_1 < t_2$ მაშინ და მხოლოდ მაშინ, როდესაც სესია t_1 იწყება სესია t_2-ზე ადრე. აქვე განვმარტავთ ყოველი x_i-ის განსაზღვრის არეს, როგორც ყველა იმ სესიების სიმრავლეს,

რომლებშიც შეიძლება დაინიშნოს i -ური გამოცდა შეზღუდვების დარღვევის გარეშე. თავდაპირველად, x_i -ის განსაზღვრის არე იქნება ისეთი t სესიების სიმრავლე, რომლებისთვისაც სრულდება $(i,t) \notin R$. შეზღუდვების დაპროგრამების ეტაპის მსვლელობისას, შესაძლოა, ზოგიერთი ცვლადის განსაზღვრის არე შემცირდეს მისგან სესიების ამოკლებით იმისათვის, რომ ცხრილის თავსებადობა შეზღუდვებთან არ დაირღვეს.

გამოცდების დანიშვნის შესაძლებლობის შეზღუდვის დაკმაყოფილება უნდა უზრუნველვყოთ ცვლადების განსაზღვრის არეების ინიციალიზაციისას.

დამთხვევის შეზღუდვის მოდელირებას ვაკეთებთ შემდეგნარიად:

$$x_i \neq x_j \forall i, j \in E \text{ -სათვის, სადაც } i \neq j \text{ და } D_{ij} > 0 .$$

საერთო ტევადობის შეზღუდვა მოდელირებულია შემდეგნაირად:

$$\sum_{i \in E} s_i (x_i = t) \leq C_t, \forall t \in T .$$

ზემოთ მოცემულ გამოსახულებაში $(x_i = t)$ წარმოადგენს ორობით ფუნქციას, რომელიც 1-ის ტოლ მნიშვნელობას იღებს მაშინ და მხოლოდ მაშინ, როდესაც $x_i = t$. სხვა შემთხვევაში ის 0-ის ტოლ მნიშვნელობას იღებს.

შეზღუდვების დაპროგრამების ეტაპზე ვირჩევთ რომელიმე გამოცდას და მას ვუსადაგებთ რომელიმე სესიას თავისი განსაზღვრის არედან, რაც ნიშნავს, რომ ამ გამოცდის განსაზღვრის არედან ამოვარდება ყველა სესია, გარდა არჩეულისა. ასეთ გამოცდაზე ვიტყვით, რომ ის *დაგეგმილია*. სხვა შემთხვევაში გამოცდა *დაუგეგმავია*. ყოველი გამოცდის დაგეგმვის შემდეგ საჭიროა შემოწმდეს ყველა სხვა x_i ცვლადის განსაზღვრის არის თავსებადობა შეზღუდვებთან. ეს ძირითადად გულისხმობს იმის შემოწმებას, დაირღვევა თუ არა დამთხვევის ან საერთო ტევადობის შეზღუდვები i -ური გამოცდის ჩასმით ამა თუ იმ სესიაში. შეუთავსებლობის აღმოჩენის შემთხვევაში, სესია უნდა ამოვიღოთ x_i -ის განსაზღვრის არედან. შეზღუდვებთან თავსებადობის უზრუნველყოფის შემდეგ სხვა, დაუგეგმავ გამოცდას ვირჩევთ და ამ პროცესს ვაგრძელებთ მანამ, სანამ ყველა გამოცდის დაგეგმვას არ შევეცდებით - ზოგიერთი გამოცდის განსაზღვრის არე, შესაძლოა,

დაცარიელდეს პირველი ეტაპის მსვლელობისას, შესაბამისად, მას ვერც ერთ სესიას ვერ შევუსაბამებთ. ასეთ შემთხვევაში ვქმნით ახალ, *წარმოსახვით სესიას*, რომელიც ყოველი გამოცდის განსაზღვრის არეს ემატება და სწორედ ამ სესიაში ვსვამთ არჩეულ გამოცდას. ვთვლით, რომ წარმოსახვით სესიებს შეუზღუდავი ტევადობა აქვს და ისინი რეალური სასესიო პერიოდის შემდეგ იწყება. აღსანიშნავია, რომ წარმოსახვით სესიაში ჩასმული გამოცდები დაუგეგმავად ითვლება.

შეზღუდვების დაპროგრამების ეტაპის ცალკეულ ბიჯზე გამოცდისა და სესიის არჩევა მრავალნაირად შეიძლება. ჩვენ მიერ გამოყენებული სტრატეგია შემდეგნაირია: ყოველ ბიჯზე ვირჩევთ იმ გამოცდას, რომლის განსაზღვრის არეც ყველაზე პატარაა და მას ვსვამთ იმ სესიაში, რომელიც განსაზღვრის არეში არსებულ სესიებს შორის ყველაზე ადრე იწყება. ამ სტრატეგიის მიზანია, რომ შეზღუდვების დაპროგრამების ეტაპზე რაც შეიძლება მეტი გამოცდა დაიგეგმოს რაც შეიძლება ნაკლები სესიის გამოყენებით. თუ მინიმალური განსაზღვრის არის მქონე გამოცდა რამდენიმეა, მათ შორის ვირჩევთ ნებისმიერს. უკეთესი შედეგის მისაღებად ვცადეთ, რომ ნებისმიერის ნაცვლად აგვერჩია ისეთი გამოცდა, რომელზეც ყველაზე მეტი სტუდენტია დარეგისტრირებული და რომელზე დარეგისტრირებულ სტუდენტთა სიმრავლესაც ყველაზე დიდი თანაკვეთა აქვს სხვა გამოცდებზე დარეგისტრირებული სტუდენტების სიმრავლეებთან, თუმცა ამით შედეგი არ გაუმჯობესებულა.

შემდგომი გაუმჯობესების მიზნით შესაძლოა ექსპერიმენტების ჩატარება მოცემული გამოცდისათვის სესიის შერჩევის სტრატეგიაზე: მაგალითად, ყველაზე ადრეული სესიის ნაცვლად შეგვიძლია ვცადოთ იმ სესიის არჩევა, რომელშიც ყველაზე ნაკლები თავისუფალი ადგილია დარჩენილი ან პირიქით - ყველაზე მეტი თავისუფალი ადგილი.

ამგვარად, შეზღუდვების დაპროგრამების ეტაპის დასრულების შემდეგ ვიღებთ საგამოცდო ცხრილს, რომელიც ჩვენს შეზღუდვებს აკმაყოფილებს, მაგრამ ვთანხმდებით, რომ ამ ეტაპის შემდეგ ზოგიერთი გამოცდა შეიძლება დაუგეგმავი დარჩეს, ანუ ე.წ. წარმოსახვით სესიაში იყოს განთავსებული. ჰიბრუდული ალგორითმის მეორე ეტაპის მთავარი მიზანი წარმოსახვითი სესიების დაცარიელება, ანუ ყველა გამოცდის დაგეგმვაა. გარდა ამისა, მეორე

ეტაპი ცხრილის ხარისხის გაუმჯობესებას ცდილობს. ხარისხის ცნებას შემდგომში განვმარტავთ.

ე. წ. ფოლადის წრთობის სიმულაცია (simulated annealing)

შეზღუდვების დაპროგრამების შედეგად მიღებული ცხრილი ჰიბრიდული ალგორითმის მეორე ეტაპს გადაეცემა საწყის მონაცემად. მეორე ეტაპის მიზანია დაუგეგმავი გამოცდების დაგეგმვა და ცხრილის ხარისხის გაუმჯობესება.

ფოლადის წრთობის სიმულაცია მოცემული ფუნქციის გლობალური მინიმუმის ძებნის საკმაოდ გავრცელებული ალბათური ტექნიკაა. ეს მეთოდი წარმოადგენს მოდელირებას მეტალების გაცხელების და შემდგომ ნელი გაგრილების ფიზიკური პროცესისა, რომლის მიზანიც მეტალზე არსებული დეფექტების გასწორებაა. ფოლადის წრთობის სიმულაციის მეთოდს მიეწოდება რომელიმე წერტილი საძიებო არედან, როგორც საწყისი ამონახსნი. მეთოდი ყოველ ბიჯზე შემთხვევითად ირჩევს ახალ წეტილს მიმდინარე ამონახსნის სამეზობლოდან (სამეზობლოს განსაზღვრება დამოკიდებულია კონკრეტულ ამოცანაზე). თუ სამიზნე ფუნქციის მნიშვნელობა ახალ წერტილში ნაკლებია მიმდინარე ამონახსნზე, მას ახალი წერტილი ჩაანაცვლებს. წინააღმდეგ შემთხვევაში, მიმდინარე ამონახსნის ჩანაცვლება ახალი წერტილით ხდება *გარკვეული ალბათობით*. ამის მიზანია ის, რომ მეთოდი არ ჩარჩეს ლოკალურ მინიმუმში და უფრო მეტი შესაძლო ამონახსნი განიხილოს. ალბათობა, რომლითაც ხდება მიმდინარე ამონახსნის ჩანაცვლება უარესი ამონახსნით, დამოკიდებულია *სისტემის ტემპერატურაზე*. საწყის ტემპერატურად ირჩევენ რაიმე დიდ რიცხვს, რომელიც წინასწარ შერჩეული გეგმის მიხედვით მცირდება ალგორითმის მიმდინარეობისას. უარესი ამონახსნის არჩევის ალბათობა იკლებს ტემპერატურის კლებასთან ერთად.

შემდგომში *ამონახსნს* ვუწოდებთ კანდიდატ ცხრილს დაუგეგმავ, (ანუ, წარმოსახვით სესიებში განთავსებულ) გამოცდებთან ერთად. ნებისმიერ შემთხვევაში, ყველა დაგეგმილი გამოცდა ყველა ამონახსნში უნდა აკმაყოფილებდეს დამთხვევის, საერთო ტევადობისა და გამოცდის დანიშვნის შესაძლებლობის შეზღუდვებს, რაც ნიშნავს, რომ ერთსა და იმავე სესიაში განთავსებული გამოცდების არც ერთი წყვილი არ უნდა წარმოშობდეს კონფლიქტს არც ერთი სტუდენტის საგამოცდო ცხრილში, ერთ სესიაში განთავსებული სტუდენტების

საერთო რაოდენობა არ უნდა აჭარბებდეს სესიის ტევადობას, და არც ერთი გამოცდა არ უნდა იყოს დანიშნული მისთვის შეუსაბამო სესიაში.

ჰიბრიდული ალგორითმის მეორე ეტაპის მნიშვნელოვანი კომპონენტია ამონახსნის *სამეზობლოს* ცნება. აქ გამოყენებული სამეზობლოს ცნება წარმოადგენს ე.წ. *კემპეს ჯაჭვის* ვარიანტს. კემპეს ჯაჭვი განისაზღვრება i-ური გამოცდით, რომელიც ამჟამად განთავსებულია t-ურ სესიაში, და რომელიმე სხვა $t' \neq t$ სესიით. ვთქვათ, რომ G არის t სესიაში განთავსებული ყველა გამოცდის სიმრავლე, ხოლო G' არის t' სესიაში განთავსებული ყველა გამოცდის სიმრავლე. შევნიშნოთ, რომ ამონახსნის ჩვენ მიერ გამოყენებული განმარტება უზრუნველყოფს იმას, რომ G და G' გამოცდების უკონფლიქტო სიმრავლეებია. კემპეს ჯაჭვი შეგვიძლია წარმოვიდგინოთ, როგორც უნიკალური წყვილი გამოცდების მინიმალური სიმრავლეებისა $F \subseteq G$ და $F' \subseteq G'$ ისეთი, რომ $i \in F$ და ორივე: $(G \setminus F) \cup F'$ და $(G' \setminus F') \cup F$ გამოცდების უკონფლიქტო სიმრავლეებია. მოცემული i გამოცდისთვის t სესიიდან, და მეორე - t' სესიისათვის კემპეს ჯაჭვის (ანუ F-ისა და F'-ის) ასაგებად მარტივი, იტერაციული პროცესი გამოიყენება. საგამოცდო ცხრილს, რომელიც მიიღება F'-ის ყველა გამოცდის გადასმით t სესიაში და F-ის ყველა გამოცდის გადასმით t' სესიაში, ვუწოდებთ *მეზობელ ამონახსნს. ამონახსნის სამეზობლო* განისაზღვრება, როგორც ყველა შესაძლო მეზობლების სიმრავლე.

ჰიბრიდული ალგორითმის მეორე ეტაპის დროს მიმდინარე ამონახსნის მეზობელი ირჩევა შემთხვევითად. მეზობელი ამონახსნის ასარჩევად E-დან ვირჩევთ შემთხვევით i გამოცდას და იმ სესიების სიმრავლიდან, რომლებიც i-სათვის შეზღუდული არ არის, ანუ სრულდება $(i,t') \notin R$, ვირჩევთ ისეთ შემთხვევით t' სესიას, რომ $t' \neq t$, სადაც t არის მიმდინარე ამონახსნში i-სათვის გამოყოფილი სესია. i-ს, t-სა და t'-ის ერთობლიობა გვაძლევს კემპეს ჯაჭვს, რაც, თავის მხრივ გვაძლევს მიმდინარე ამონახსნის მეზობელ ამონახსნს. მეზობელი ამონახსნის გენერირება სხვაგვარადაც შეიძლება: მაგალითად, ორი შემთხვევითი სესიის არჩევით და პირველი სესიიდან გამოცდის შემთხვევითად ამორჩევით. ჩვენთვის უფრო მოსახერხებელი პირველი მიდგომა იყო.

მიმდინარე ამონახსნის დაგენერირების შემდეგ მოწმდება, აკმაყოფილებს თუ არა ის ყველა შეზღუდვას (ჩვენს შემთხვევაში ესენია საერთო ტევადობისა და გამოცდების დანიშვნის

შესაძლებლობის შეზღუდვები). თუ მიმდინარე ამონახსნი მისაღებია, მისთვის გამოითვლება სამიზნე ფუნქციის მნიშვნელობა, რომელიც ცხრილის ხარისხის მაჩვენებელია. თუ ახალი ამონახსნი ხარისხით აღემატება მიმდინარე ამონახსნს (სამიზნე ფუნქციის მნიშვნელობა ნაკლებია ახალი ამონახსნისათვის), მიმდინარე ამონახსნს ანაცვლებს ახალი ამონახსნი. სხვა შემთხვევაში მიმდინარე ამონახსნი შეიძლება ჩანაცვლდეს ან არ ჩანაცვლდეს ახლით. ეს დამოკიდებულია სამიზნე ფუნქციის მნიშვნელობებს შორის სხვაობასა და სისტემის მიმდინარე ტემპერატურაზე. $o(p)$ -თი აღვნიშნოთ სამიზნე ფუნქციის მნიშვნელობა p ამონახსნისათვის. დავუშვათ, რომ x არის მიმდინარე ამონახსნი, ხოლო q არის x -ის მეზობელი, რომელიც შემთხვევითად შევარჩიეთ მოცემულ იტერაციაზე. u იყოს მიმდინარე ტემპერატურა. მაშინ, თუ $o(q) > o(x)$, ალბათობას იმისა, რომ q ჩაანაცვლებს x -ს გამოითვლება შემდეგი ფორმულით: $\frac{1}{1+e^{\frac{o(q)-o(x)}{u}}}$

შევნიშნოთ, რომ ზემოთ მოცემული გამოსახულების მნიშვნელობა u -ს კლებასთან ერთად მცირდება.

უარესი ამონახსნის არჩევის ალბათობა სხვაგვარადაც შეიძლება გამოითვალოს, მაგალითად, შემდეგი ფორმულით: $e^{\frac{o(x)-o(q)}{u}}$. ჩვენმა ექსპერიმენტებმა აჩვენა, რომ პირველი ფორმულა უკეთეს შედეგებს იძლევა, თუმცა უნდა აღინიშნოს, რომ განსხვავება საკმაოდ მცირე იყო.

ასევე მრავალნაირადაა შესაძლებელი გაგრილების, ანუ ტემპერატურის კლების რეჟიმის შედგენა. აქ გამოყენებულია ე.წ. გეომეტრიული რეჟიმი, რაც გულისხმობს იმას, რომ ყოველი a იტერაციის შემდეგ ტემპერატურა მრავლდება α -ზე. a და α პარამეტრებია, რომლებიც ალგორითმს გადაეცემა. ჩვენ ვიყენებთ ზუსტად იმ პარამეტრებს, რომლებიც გამოყენებულია [1]-ში. ესენია:

- საწყისი ტემპერატურა = 30 000
- a = 10
- α = 0.999
- შეჩერების ტემპერატურა = 10^{-11}

ამ პარამეტრების გამოყენებით მეორე ეტაპი დაახლოებით 350 000 იტერაციას ასრულებს.

ფოლადის წრთობის სიმულაციის მეთოდით პირველ ეტაპზე მიღებული ცხრილის ხარისხის მნიშვნელოვანი გაუმჯობესებაა შესაძლებელი. თუმცა, უნდა აღინიშნოს, რომ ხარისხიანი ცხრილის შედგენაში მთავარი როლი მაინც ჰიბრიდული ალგორითმის პირველ ეტაპს უჭირავს. [1]-ში განიხილავენ მათ მიერ ჩატარებულ ექსპერიმენტს, რომელშიც ფოლადის წრთობის სიმულაციის მეთოდს საწყის მონაცემად შემთხვევითად დაგენერირებული ცხრილი გადასცეს და ძალიან ცუდი შედეგები მიიღეს. ჩვენი გამოცდილებაც ამაზე მიუთითებს: ჩვენ მიერ ჩატარებულ ყველა გამოთვლაში გამოცდების უმეტესობა სწორედ პირველ ეტაპზე იგეგმებოდა.

ფოლადის წრთობის სიმულაციის მეთოდის მიერ გამოყენებული სამიზნე ფუნქცია მრავალნაირად შეიძლება განისაზღვროს. ჩვენ საგამოცდო ცხრილის ხარისხის მაჩვენებლებად ვიყენებთ დაუგეგმავი გამოცდების რაოდენობას. კერძოდ, ყოველი სტუდენტისათვის ყოველი დაუგეგმავი გამოდა სამიზნე ფუნქციის მნიშვნელობას ზრდის 10,000-ით.

გარდა დაუგეგმავი გამოცდების რაოდენობსა თითოეული სტუდენტისთვის, სამიზნე ფუნქციის მნიშვნელობის გამოთვლისას შეიძლება გავითვალისწინოთ შემთხვევები, როცა სტუდენტს ერთ დღეში ორი გამოცდა აქვს დანიშნული, ან ორი მიმდევრობით სესიაში აქვს დანიშნული გამოცდები. ამ შემთხვევებისთვის „საჯარიმო ქულა“, რომელიც სამიზნე ფუნქციის მნიშვნელობას ემატება, ბევრად ნაკლები უნდა იყოს დაუგეგმავი გამოცდისათვის დაწესებულ საჯარიმო ქულაზე, რადგან მთავარი ამოცანა მაქსიმალურად მეტი გამოცდის დაგეგმვაა.

ცხრილის ხარისხის ზემოთ განხილული მაგალითების გარდა კიდევ არაერთი ხარისხის მაჩვენებლის გამოყენებაა შესალებელი, თუმცა ხარისხის ახალი მაჩვენებლის შემოღებისას სიფრთხილე გვამრთებს, რადგან ზოგჯერ სამიზნე ფუნქციისათვის ახალი კომპონენტის დამატებამ, შესაძლოა, შედეგის გაუარესება გამოიწვიოს. ეს ჩანს [1]-ში განხილული მელბურნის უნივერსიტეტის მაგალითზე, სადაც გამოიყენეს სამიზნე ფუნქცია შემდეგი კომპონენტებით:

- U - ჯარიმა ყოველ დაუგეგმავ გამოცდაზე დარეგისტრირებული თითოეული სტუდენტისათვის.

- w_{sd} - ჯარიმა ყოველი სტუდენტისათვის ერთ დღეში დანიშნულ ორ გამოცდაზე.
- w_{am} - ჯარიმა ყოველი სტუდენტისათვის დანიშნულ ყოველ ორ გამოცდაზე, რომელთაგან ერთი საღამოს სესიაშია დანიშნული, მეორე კი მომდევნო დღის დილის სესიაში.
- w_{g2} - ჯარიმა ყოველი სტუდენტისათვის დანიშნულ ყოველ ორ გამოცდაზე, რომელთაც მხოლოდ ერთი სესია ყოფთ.
- w_{ma} - ჯარიმა ყოველი სტუდენტისათვის დანიშნულ ყოველ ორ გამოცდაზე, რომელთაგან ერთი ტარდება დილის სესიაში, მეორე კი მომდევნო დღის შუადღის სესიაში.

უნდა აღვნიშნოთ, რომ მელბურნის უნივერსიტეტში ყოველ დღე ორი სესია ტარდება.

ექსპერიმენტების შედეგად მკვლევარებმა ზემოთ მოყვანილი პარამეტრების ის მნიშვნელობები იპოვნეს, რომლებიც საბოლოო ჯამში საუკეთესო საგამოცდო ცხრილს იძლეოდა. ესენია: $U = 10000,\ w_{sd} = 2,\ w_{am} = 1,\ w_{g2} = 0$ და $w_{ma} = 0$. ანუ აღმოჩნდა, რომ ბოლო ორი პარამეტრი არა თუ არ აუმჯობესებდა ალგორითმის ეფექტურობას, არამედ აუარესებდა კიდეც.

ლოკალური ძებნა (Hill Climbing)

იმ შემთხვევაში, თუ ჰიბრიდული ალგორითმის მეორე ეტაპის დასრულების შემდეგ მეტი ოპტიმიზაცია გახდა საჭირო, შეგვიძლია გამოვიყენოთ ლოკალური ოპტიმუმის ძებნის მეთოდი, რომელსაც ქვემოთ განვიხილავთ.

ზოგჯერ ფოლადის წრთობის სიმულაციის მეთოდის მსვლელობისას, შესაძლოა, ვერ მოხერხდეს მის მიერ ნაპოვნი საუკეთესო ამონახსნის სამეზობლოს საკმარისად კარგად გამოკვლევა. ამის მიზეზი შეიძლება ის იყოს, რომ ალგორითმმა საუკეთესო ამონახსნი მისი მუშაობის დასაწყისში იპოვნოს, როცა ტემპერატურა და, აქედან გამომდინარე, უარესი ამონახსნის არჩევის ალბათობა დიდია. ალგორითმი შესაძლოა გადავიდეს უარეს ამონახსნზე და ტემპერატურის შემცირების შედეგად, ლოკალურ მინიმუმში დარჩეს. სწორედ ასეთი შემთხვევების დროს შეიძლება მივიღოთ მნიშვნელოვანი გაუმჯობესება მესამე ეტაპით, თუმცა, ლიტერატურის მიხედვით, ასეთი შემთხვევები იშვიათია.

ლოკალური ძებნის ეტაპს საწყის მონაცემად გადაეცემა რაიმე x მონაცემი, რომელსაც მიმდინარე ამონახსნს ვუწოდებთ. ვთქვათ, რომ სამიზნე ფუნქციის მნიშვნელობა ამ ამონახსნისათვის არის $o(x)$. ალგორითმი სათითაოდ განიხილავს ყველა გამოცდას რაიმე წინასწარ განსაზღვრული მინდევრობით, მაგალითად, გამოცდის კოდის ზრდადობით. ყოველი e გამოცდისათვის ალგორითმი განიხილავს x-ის ყველა ისეთ მეზობელს (ზემოთ განხილული კემპეს ჯაჭვის განმარტების გამოყენებით), რომელშიც e განთავსებულია რომელიმე სხვა სესიაში (არა იმ სესიაში, რომელშიც ის x-შია განთავსებული). ყველა ასეთი მეზობლისათვის, რომელიც გამოცდის დანიშვნის შესაძლებლობისა და საერთო ტევადობის შეზღუდვებს აკმაყოფილებს, გამოითვლება სამიზნე ფუნქციის მნიშვნელობა და ირჩევა q მეზობელი, რომლისთვისაც სამიზნე ფუნქციის მნიშვნელობა ყველაზე ნაკლებია. თუ სამიზნე ფუნქციის მინიმალურ მნიშვნელობას რამდენიმე მეზობელი გვაძლევს, მაშინ შეიძლება ამოირჩეს ისეთი მეზობელი ამონახსნი, რომელიც e გამოცდას რაც შეიძლება ადრე დანიშნავს. თუ $o(q) \leq o(x)$, მაშინ მიმდინარე x ამონახსნს q ჩაანაცვლებს. სხვა შემთხვევაში ამონახსნი უცვლელი რჩება. ნებისმიერ შემთხვევაში, ამით სრულდება e გამოცდის განხილვა და ალგორითმი გადადის რიგით შემდეგი გამოცდის განხილვაზე.

არ არის აუცილებელი, რომ ლოკალური ძებნა დასრულდეს ყველა გამოცდის განხილვის შემდეგ: ყველა გამოცდის განხილვით ლოკალური ძებნის ერთი იტერაცია სრულდება. ჰიბრიდული ალგორითმის მესამე ეტაპმა მრავალი იტერაცია შეიძლება ჩაატაროს. ლოკალური ძებნა შეგვიძლია დავასრულოთ იმ შემთხვევაში, თუ მის მიერ დაგენერირებული არც ერთი მეზობელი არ გააუმჯობესებს სამიზნე ფუნქციის მნიშვნელობას ან წინასწარ განსაზღვრული რაოდენობის იტერაციების დასრულების შემდეგ. [1]-ში განხილული მელბურნის უნივერსიტეტის მაგალითზე ჩანს, რომ ლოკალური ძებნის 7 იტერაციის შემდეგ შედეგი არ უმჯობესდებოდა, ასე რომ მათ იტერაციების მაქსიმალურ რაოდენობად 10 დააწესეს.

ჩვენ სასურველი შედეგი მივიღეთ ჰიბრიდული ალგორითმის მხოლოდ პირველი ორი ეტაპის გამოყენებითაც, თანაც [1]-ში ხაზგასმულია, რომ ლოკალური ძებნით შედეგის მნიშვნელობვანი გაუმჯობესება მოსალოდნელი არ არის, ამიტომ მესამე ეტაპის იმპლემენტაცია საჭიროდ არ ჩავთვალეთ.

ხარბი ევრისტიკა დარჩენილი გამოცდების დასაგეგმად

როგორც ზემოთ განვიხილეთ, შეზღუდვების დაპროგრამების ეტაპზე ზოგ გამოცდას ეგრედ წოდებულ "წარმოსახვით" სესიებში ვათავსებთ. როგორც წესი, ფოლადის წრთობის სიმულაციის ეტაპზე ასეთი გამოცდები რეალურ სესიებში გადაინაცვლებს, მაგრამ შეიძლება ისეც მოხდეს, რომ ჰიბრიდული ალგორითმის სამივე ეტაპის დასრულების შემდეგ ზოგი გამოცდა კვლავ დარჩეს წარმოსახვით სესიაში.

ასეთ შემთხვევაში, შესაძლებელია, რომ გამოცდების რეალურ სესიებში გადატანა მოხერხდეს ხარბი მეთოდით.

მეთოდი მუშაობს შემდეგნაირად: ალგორითმი განიხილავს ყველა იმ გამოცდას, რომელიც წარმოსახვით სესიაშია განთავსებული, გარკვეული, წინასწარ განსაზღვრული მიმდევრობით. გავიხსენოთ, რომ ასეთი გამოცდები დაუგეგმავად ითვლება. ყოველი ასეთი e გამოცდისათვის ალგორითმი ეძებს ისეთ რეალურ $t \in \{1, \ldots, v\}$ სესიას, რომ მასში e გამოცდის ჩასმით არ დაირღვეს ძირითადი შეზღუდვები. ასეთი რამდენიმე სესიის არსებობის შემთხვევაში, უნდა ავირჩიოთ ისეთი სესია, რომელში e გამოცდის ჩასმითაც სამიზნე ფუნქციის მნიშვნელობის მინიმიზაცია მოხდება. თუ ასეთი სესია არ მოიძებნა, გამოცდა დაუგეგმავი რჩება.

თბილისის სახელმწიფო უნივერსიტეტის მაგალითი

ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტში შვიდი ფაკულტეტია. ესენია: ზუსტ და საბუნებისმეტყველო მეცნიერებათა, ჰუმანიტარულ მეცნიერებათა, სოციალურ და პოლიტიკურ მეცნიერებათა, ფსიქოლოგიისა და განათლების მეცნიერებათა, ეკონომიკისა და ბიზნესის, მედიცინისა და იურიდიული ფაკულტეტები.

ბაკალავრიატის სტუდენტთა გამოცდები ცენტრალიზირებულად ტარდება თბილისის სახელმწიფო უნივერსიტეტის საგამოცდო ცენტრის ორგანიზებით. საგამოცდო ცენტრი პასუხისმგებელია გამოცდების ჩატარებაზე ყველა ფაკულტეტზე გარდა მედიცინის ფაკულტეტისა. ყოველი აკადემიური წელი შედგება ორი სემესტრისაგან. ორივე სემესტრში თითოეულ ფაკულტეტზე ტარდება შუალედური და საბოლოო გამოცდები. გამოცდები ტარდება თბილისის სახელმწიფო უნივერსიტეტის ბიბლიოთეკაში, გაშლილ

აუდიტორიებში. აქედან გამომდინარე, გამოცდების ოთახებში გადანაწილება როგორც წესი, არ წარმოადგენს პრობლემას. მთავარი ამოცანა გამოცდების სესიებში გადანაწილებაა.

შუალედური გამოცდები სამ კვირაზეა გადანაწილებული. გამოცდები ტარდება ყოველ დღე კვირის გარდა, ანუ გამოცდები 54 სესიაზე უნდა გადანაწილდეს. დღეში ტარდება სამი სესია. თითოეულ სესიაზე სტუდენტების მაქსიმალური დასაშვები რაოდენობაა 1742. ამასთან, წინასწარ არის განსაზღვრული ის, თუ რომელი ფაკულტეტების გამოცდების დანიშვნაა ნებადართული ამა თუ იმ კვირაში. კერძოდ:

- ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტი - II კვირა
- ჰუმანიტარულ მეცნიერებათა ფაკულტეტი - I კვირა
- სოციალურ და პოლიტიკურ მეცნიერებათა ფაკულტეტი - I კვირა
- ფსიქოლოგიისა და განათლების მეცნიერებათა ფაკულტეტი - I კვირა
- ეკონომიკისა და ბიზნესის ფაკულტეტი - II კვირა
- იურიდიული ფაკულტეტი - II-III კვირა

ეს შეზღუდვა განპირობებულია იმით, რომ შუალედური გამოცდები საგამოცდო პროცესის პარალელურად მიმდინარეობს. რაც შეეხება დასკვნით გამოცდებს, ისინი სასწავლო პროცესის დასრულების შემდეგ ტარდება, ამიტომ მათი ჩატარების გრაფიკი უფრო მოქნილია და ესა თუ ის კვირა არ არის შეზღუდული კონკრეტული ფაკულტეტებისათვის. დასკვნითი გამოცდების პერიოდში დღეში 2 სესია ტარდება.

ჩვენ სატესტოდ გამოვიყენეთ საგამოცდო ცენტრის მიერ მოწოდებული საგამოცდო მონაცემები 2014 წლის შემოდგომის სემესტრიდან. ამ მონაცემების მიხედვით უნდა დაიგეგმოს 841 გამოცდის ჩატარება. ყველა გამოცდაზე დარეგისტრირებული სტუდენტების ჯამური რაოდენობაა 83,548. ქვემოთ მოყვანილ ცხრილში ნაჩვენებია გამოცდებისა და მათზე დარეგისტრირებული სტუდენტების რაოდენობა ფაკულტეტების მიხედვით:

ცხრილი 1 - თბილისის სახელმწიფო უნივერსიტეტში ჩასატარებელი გამოცდებისა და მათზე დარეგისტრირებული სტუდენტების ჯამური რაოდენობა ფაკულტეტების მიხედვით 2014 წლის შემოდგომის სემესტრიდან

ფაკულტეტი	საგნების რაოდენობა	სტუდენტების ჯამური რაოდენობა
ზუსტ-საბუნებისმეტყველო	138	15194
ჰუმანიტარული	427	21949
სოციალურ-პოლიტიკური	106	13410
ეკონომიკისა და ბიზნესის	63	15407
იურიდიული	107	17588
სულ	**841**	**83548**

ექსპერიმენტალური შედეგები

ჰიბრიდული ალგორითმის ჩვენი იმპლემენტაცია გავტესტეთ წინა პარაგრაფში წარმოდგენილი მონაცემებისათვის. პირველ რიგში, განვიხილოთ ალგორითმის მუშაობის შედეგი შუალედური გამოცდების ცხრილის შესადგენად, რადგან სწორედ შუალედური გამოცდების დროს აქვს უნივერსიტეტს დაწესებული ყველაზე მეტი შეზღუდვა.

ჰიბრიდული ალგორითმის პირველ ეტაპზე 841 გამოცდიდან დაიგეგმა 573, ხოლო 254 დაუგეგმავი დარჩა. დაუგეგმავი გამოცდების გადასანაწილებლად, 54 ნამდვილი სესიის გარდა, 17 წარმოსახვითი სესიის შექმნა გახდა საჭირო. პირველი ეტაპის შედეგები ფაკულტეტების მიხედვით წარმოდგენილია ქვემოთ მოცემულ ცხრილში

ცხრილი 2 - ჰიბრიდული ალგორითმის პირველი ეტაპის შედეგები შუალედური გამოცდებისათვის

ფაკულტეტი	პირველ ეტაპზე დაუგეგმავი გამოცდების რაოდენობა
ზუსტი-საბუნებისმეტყველო	33
ჰუმანიტარული	105
სოციალურ-პოლიტიკური	36
ეკონომიკა და ბიზნესი	24
იურიდიული	27

მეორე ეტაპის შედეგად ყველა გამოცდა დაიგეგმა. ამას ფოლადის წრთობის სიმულაციის მეთოდის დაახლოებით 72,000 ბიჯი დასჭირდა.

თუ მოვხსნით შეზღუდვებს გამოცდების ფაკულტეტების მიხედვით გადანაწილებაზე, მაშინ პირველი ეტაპის შემდეგ დაუგეგმავი მხოლოდ 37 გამოცდა რჩება, რომელთაგან მეორე ეტაპზე ყველა იგეგმება. ამას დაახლოებით 72,000 ბიჯი სჭირდება.

საგამოცდო ცხრილის შესადგენი პროგრამული უზრუნველყოფის აღწერა

სამაგისტრო ნაშრომის ფარგლებში შეიქმნა Windows Forms აპლიკაცია, რომელიც გვაძლევს საშუალებას ნაშრომში აღწერილი ჰიბრიდული ალგორითმის მუშაობა გავტესტოთ თბილისის სახელმწიფო უნივერსიტეტის მაგალითზე.

აღნიშნულ აპლიკაციას მონაცემად გადაეცემა Excel ტიპის ფაილი, რომელშიც საგამოცდო მონაცემები წინასწარ განსაზღვრული ფორმატითაა წარმოდგენილი. ჩასატარებელ გამოცდებსა და მათზე დარეგისტრირებულ სტუდენტებზე ინფორმაციას პროგრამა სწორედ ამ ფაილიდან კითხულობს. მას შემდეგ, რაც ექსელის ფაილიდან მონაცემების წაკითხვა დასრულდება, მომხმარებელს შეუძლია აირჩიოს საგამოცდო პერიოდის დაწყებისა და დასრულების თარიღები. პროგრამა დააგერენირებს იმდენ სესიას, რამდენი დღეც იქნება სასესიო პერიოდის დაწყებისა და დასრულების თარიღებს შორის გარდა კვირა დღეებისა. ანუ იგულისხმება, რომ გამოცდები ტარდება ყოველ დღე კვირის გარდა. თბილისის სახელმწიფო უნივერსიტეტში მართლაც ასეა.

სასესიო პერიოდის არჩევის შემდეგ მომხმარებელს შეუძლია აირჩიოს ერთ დღეში ჩასატარებელი სესიების რაოდენობა. ერთ დღეში შეიძლება დაინიშნოს ერთი, ორი ან სამი სესია. თბილოსის სახელმწიფო უნივერსიტეტში შუალედური გამოცდების დროს დღეში 3 სესია ტარდება, ხოლო საბოლოო გამოცდების დროს დღეში 2 სესია.

სესიების რაოდენობის არჩევის შემდეგ მომხმარებელს შეუძლია აირჩიოს თითოეული სესიის ტევადობა. თბილისის სახელმწიფო უნივერსიტეტში არსებული სიტუაციიდან გამომდინარე, პროგრამა მომხმარებელს ერთი სესიის ტევადობად სთავაზობს 1742, 3484 ან 5226 სტუდენტის არჩევას, თუმცა არც სხვა მნიშვნელობის მითითებას უზღუდავს.

რადგან თბილისის სახელმწიფო უნივერსიტეტში შუალედური გამოცდების ჩატარების ვადები განსხვავდება ფაკულტეტების მიხედვით, მომხმარებელს შეუძლია აირჩიოს ის ფაკულტეტები, რომელთა გამოცდების დანიშვნაც უნდა და სასესიო პერიოდი ცალკეული ფაკულტეტისათვის. ცალკეული ფაკულტეტის სასესიო პერიოდი არ უნდა სცდებოდეს საერთო საგამოცდო პერიოდს, რომელიც მომხმარებელს უკვე მითითებული უნდა ჰქონდეს.

ზემოთ მოყვანილი მონაცემების დაზუსტების შემდეგ, ეკრანზე გამოდის ინფორმაცია ყოველ სესიაზე. მომხმარებელს შეუძლია წინასწარ, გამოთვლების დაწყებამდე, მისთვის სასურველი გამოცდა ჩასვას მისთვის სასურველ სესიაში. ეს ხდება ამ სესიისათვის გამოყოფილ არეზე მაუსის მარჯვენა ღილაკით დაწკაპებით და სასურველი გამოცდის არჩევით. ამ ეტაპზე დანიშნული გამოცდები გამოთვლების დაწყების შემდეგ სხვა სესიაში არ გადადის.

ზოგიერთი გამოცდის წინასწარ დანიშვნის შემდეგ, მომხმარებელს შეუძლია დაიწყოს გამოთვლები. Calculate ღილაკზე დაჭერა ჰიბრიდული ალგორითმის პირველ ეტაპს გამოიძახებს, რომლის დასრულებასაც 1-2 წამი სჭირდება და რომლის შედეგებიც გამოვა ეკრანზე. კერძოდ, თითოეული სესიისათვის მომხმარებელს შეუძლია დაინახოს ამ სესიაში ჩასმული გამოცდები.

პირველი ეტაპის დასრულების შემდეგ მომხმარებელს შეუძლია Resume ღილაკით დაიწყოს ფოლადის წრთობის ეტაპის სიმულაცია. ამ ეტაპს შედარებით მეტი დრო სჭირდება, 1-დან 2 წუთამდე, და მომხმარებელს შეუძლია თვალი ადევნოს პროგრესს. კერძოდ, მომხმარებელს საშუალება აქვს თვალი ადევნოს სამიზნე ფუნქციის მნიშვნელობას ყოველი იტერაციისათვის. ასევე, ყოველ სესიაში დარჩენილი ადგილების რაოდენობას. მომხმარებელს შეუძლია გამოთვლები შეაჩეროს, შეამოწმოს სამიზნე ფუნქციის მნიშვნელობა მისთვის საინტერესო ყოველი იტერაციისათვის და სურვილის შემთხვევაში გამოთვლები უკან დააბრუნოს, მისთვის სასურველ იტერაციაზე. ამ დროს მას საშუალება აქვს მისთვის სასურველი გამოცდა ჩასვას რომელიმე სესიაში, თუ ეს შესაძლებელია. თუ ასეთი ცვლილების განხორციელება შეუძლებელი იქნება კონფლიქტის გამო, მაშინ მომხმარებელს შეეძლება ნახოს იმ სტუდენტთა სია, რომლებიც იწვევენ კონფლიქტს.

იმპლემენტაციის დეტალები

ნაშრომის ფარგლებში შექმნილი გამოცდების ცხრილის შემდგენი პროგრამული უზრუნველყოფა დაიწერა პროგრამირების ენა C#-ზე. ამ თავში აღვწერთ იმპლემენტაციის მნიშვნელოვან დეტალებს, რომლებიც მკითხველს, რომელსაც სურვილი აქვს მსგავსი პროგრამული უზრუნველყოფა შექმნას ან არსებული განავითაროს, გაუადვილებს კოდის გარჩევას.

პროგრამული უზრუნველყოფა წარმოადგენს Windows Forms აპლიკაციას. აპლიკაცია მონაცემებს კითხულობს .xlsx ფაილიდან (Microsoft Office Excel), რომელიც თბილისის სახელმწიფო უნივერსიტეტის საგამოცდო ცენტრმა მოგვაწოდა. ამ ეტაპზე, პროგრამის გამართული მუშაობა პირდაპირ არის დამოკიდებული ამ ფაილის ფორმატზე და მასში ცვლილებების შეტანის შემთხვევაში პროგრამული კოდის მოდიფიკაცია გახდება საჭირო. მოცემულ ექსელის ფაილში რამდენიმე ფურცელია (excel sheet). მათგან ერთ-ერთში, სახელად „განთავსება“, მოცემულია ინფორმაცია თითოეული სტუდენტის ყოველ გამოცდაზე. კერძოდ, ყოველ რიგში (row) მოცემულია შემდეგი ინფორმაცია: სტუდენტის სასწავლო ID, საგნის ID, ფაკულტეტის ID, საგნის დასახელება, სტუდენტის ID, სტუდენტის გვარი, სტუდენტის სახელი, სტუდენტის პირადი ნომერი, სტუდენტის მობილური ტელეფონის ნომერი, სესიის ID, გამოცდის ჩატარების ადგილის მისამართი, თარიღი, დრო, საგნის პედაგოგი, ჯგუფის პედაგოგი და ჯგუფის ნომერი. აპლიკაცია კითხულობს ინფორმაციას შემდეგი სვეტებიდან: სტუდენტის ID, სტუდენტის გვარი, სტუდენტის სახელი, სტუდენტის პირადი ნომერი, საგნის ID, ფაკულტეტის ID, საგნის დასახელება. ეს ხდება ExcelReader კლასში Microsoft.Office.Interop.Excel-ის ხელსაწყოების გამოყენებით. ExcelReader კლასის კონსტრუქტორი ახდენს `private object[,] values` ველის ინიციალიზაციას და მასში ინახავს ექსელის ფაილიდან წაკითხულ ყველა მწრკივს(row). ExcelReader კლასის API შედგება მეთოდებისაგან, რომლებიც მოცემული ექსელის ფაილის ყოველი მწკრივისათვის აბრუნებენ სასურველ ინფორმაციას. კერძოდ:

```
public int getRowCount();

public rowNotNull(int rowIndex);

public string getExamName(int rowIndex);
```

```
public string getStudentID(int rowIndex);

public string getExamID(int rowIndex);
```

და ასე შემდეგ, ექსელის ფაილიდან წაკითხული ყოველი სვეტისათვის.

ძირითადი მონაცემების წაკითხვის შემდეგ პროგრამა აკეთებს ზოგიერთი მნიშვნელობის პრეკალკულაციას, რაც პროგრამის წარმადობისთვისაა საჭირო. ეს ხდება ExaminationData კლასში. ExaminationData კლასის კონსტრუქტორს გადაეცემა ExcelReader კლასის ობიექტი, რომლიდანაც კითხულობს ექსელის ფაილის ყოველი მწკრივში მოცემულ სტუდენტის ID-ს, გამოცდის ID-სა და ფაკულტეტის ID-ს და ინახავს `List<Tuple<string, string, string>>` ტიპის `studentExamFacultyData` ველში. რადგანაც string ტიპის IDებთან მუშაობა არც ისე მოსახერხებელია, სიმარტივისათვის ExaminationData კლასს შემოაქვს ახალი ID-ები სტუდენტებისა და გამოცდებისთვის მათი გადანომვრის საშუალებით. ძველ და ახალ ID-ებს შორის კავშირი შენახულია შემდეგ ველებში:

```
private Dictionary<string, int> solutionExamIDs;
private List<string> examIDs;
private Dictionary<string, int> solutionStudentIDs;
private List<string> studentIDs;
```

ამ ველების ინიციალიზაციის შემდეგ, ExaminationData კლასი, ისევე, როგორც დანარჩენი პროგრამული კოდი, ახალ, int ტიპის ID-ებს იყენებს სტუდენტებისა და გამოცდების იდენტიფიკაციისათვის.

ახალი ID-ების შემოღების შემდეგ, ExaminationData კლასში ხდება ზოგიერთი მონაცემის პრეკალკულაცია პროგრამის მომავალი მუშაობის დასწრაფებისთვის. კერძოდ:

1. `Dictionary<int, List<int>> studentExams` - ყოველი სტუდენტისათვის გამოცდების სია, რომლებზეც ის დარეგისტრირებულია.
2. `Dictionary<int, List<int>> examStudents` - ყოველი გამოცდისათვის სტუდენტების სია, რომლებიც ამ გამოცდაზე არიან დარეგისტრირებული.
3. `List<int>[,] commonStudents` - გამოცდების ყოველი i,j წყვილისათვის იმ სტუდენტების სია, რომლების დარეგისტრირებულები არიან როგორც i, ასევე j გამოცდაზე. ეს საჭიროა დამთხვევის შეზღუდვის შესრულების უზრუნველსაყოფად.

4. `HashSet<int>[] commonStudentsUnion` - ყოველი გამოცდისათვის იმ სტუდენტების სია, რომლებიც დარეგისტრირებულები არიან ამ გამოცდისაგან განსხვავებულ ერთ გამოცდაზე მაინც. ამჟამად ამ მონაცემებს გამოთვლებში არ ვიყენებთ, თუმცა შეიძლება ჰიბრიდული ალგორითმის პირველ ეტაპზე შემდეგი დასანშნი გამოცდის არჩევისას ამ მონაცემის გათვალისწინება.

ExaminationData კლასს, მსგავსად ExcelReader კლასისა, აქვს public მეთოდები მის მიერ ჩატარებული პრეკალკულაციების შედეგებზე წვდომისათვის.

შემდეგი მნიშვნელოვანი კლასია Solution.cs. ამ კლასში თავმოყრილია მეთოდები, რომლებიც საგამოცდო ცხრილის შესადგენად გამოიყენება.

Solution კლასის კონსტრუქტორს ExaminationData კლასის ობიექტი გადაეცემა, რომლიდანაც ყველა საჭირო ინფორმაციას იღებს. Solution კლასში ასევე განსაზღვრულია `List<Session> sessions` ველი, რომელიც ყველა არსებულ სესიას ინახავს (ასევე წარმოსახვით სესიებსაც, რომლებიც გამოთვლების მიმდინარეობისას იქმნება და ემატება sessions სიას). ყოველი Session ობიექტი ინახავს ინფორმაციას სესიის თარიღსა და დროზე, სესიის ტევადობასა და იმაზე, ნამდვილია თუ წარმოსახვითი ეს სესია. ასევე, ყოველი Session ობიექტი ინახავს იმ გამოცდების სიას, რომლებიც ამ სესიაშია დანიშნული. Solution კლასში და დანარჩენ პროგრამულ კოდშიც, სესიის ID-ს როლში გამოიყენება სესიის ინდექსი Solution.sessions სიაში.

გარდა ზემოთ აღნიშნულისა, Solution კლასს აქვს შემდეგი ველები:

1. `int realSessionCount` - ნამდვილი სესიების რაოდენობა.
2. `int iteration` - ჰიბრიდული ალგორითმის მიერ ჩატარებული იტერაციების მიმდინარე რაოდენობა.
3. `bool[] restricted` - ყოველი გამოცდისთვის გამოხატავს დაშვებულია თუ არა გამოცდის გადატანა სხვა სესიაში (ზოგი გამოცდა წინასწარ ინიშნება მომხმარებლის მიერ და მისი გადანაცვლება არ შეიძლება).
4. `int[] examSession` - ყოველი გამოცდისათვის ის სესია, რომელშიც ეს გამოცდაა დანიშნული.

5. `List<byte[]> pastExamSession` - ინახავს განვლილ ამონახსნებს. კერძოდ, pastExamSession[iteration][examId]-ში ინახება ჰიბრიდული ალგორითმის examId გამოცდის სესია iteration იტერაციის დროს. მეხსიერების რესურსის დაზოგვის მიზნით byte ტიპია გამოყენებული, რადგან არ არის მოსალოდნელი იმაზე მეტი სესიის არსებობა, ვიდრე byte ტიპი იტევს. ჰიბრიდული ალგორითმის პირველ ეტაპს ერთ იტერაციად ვთვლით, ისევე, როგორც ფოლადის წრთობის სიმულაციის მეთოდის ყოველ იტერაციას.
6. `List<long> objectiveScore` - სამიზნე ფუნქციის მნიშვნელობა ყოველი განვლილი იტერაციისათვის.
7. `List<int>[] sessionDomain` - წარმოადგენს გამოცდის დანიშვნის შესაძლებლობის შეზღუდვას. თითოეული გამოცდისათვის მოცემულია იმ სესიების სია, რომლებშიც ამ გამოცდის დანიშვნა დაშვებულია.

მნიშვნელოვანია Solution კლასში განსაზღვრული შემდეგი მეთოდები:

`public void generateInitialSolution()` - ჰიბრუდული ალგორითმის პირველი ეტაპის იმპლემენტაცია, რომელიც რიგ-რიგობით განიხილავს ყველა გამოცდას და ცდილობს რომელიმე სესიაში ჩასვას. თუ გამოცდისათვის შესაფერებელი სესია არ მოიძებნა, ის ახალ წარმოსახვით სესიას ქმნის. წარმოსახვითი სესიის თარიღი ერთი დღით მეტია ბოლო რეალური სესიის თარიღზე, მაქსიმალური ტევადობა კი int ტიპის მაქსიმალური მნიშვნელობაა. აქედან გამომდინარე, წარმოსახვით სესიაში გამოცდის ჩასმა წარუმატებელი შეიძლება იყოს მხოლოდ იმ შემთხვევაში, თუ ეს გამოიწვევს დამთხვევის შეზღუდვის დარღვევას. გამოცდების რიგითობას განსაზღვრავს მისთვის ხელმისაწვდომი სესიების რაოდენობა - პირველ რიგში ყველაზე პატარა დომეინის მქონე გამოცდა ინიშნება. გამოცდის არჩევის შემდეგ, მისთვის ხელმისაწვდომი სესიებიდან მეთოდი ირჩევს იმას, რომელიც ყველაზე ადრე ტარდება.

`public void generateNextSolution(double temperature)` – ასრულებს ფოლადის წრთობის სიმულაციის მეთოდის ერთ იტერაციას. მეთოდი ირჩევს შემთხვევით გამოცდას და შემთხვევით სესიას მისი დომეინიდან და აგებს კემპეს ჯაჭვს, რომელიც მიიღება არჩეული გამოცდითა და სესიით და რომელიც განსაზღვრულია ზემოთ, ჰიბრიდული ალგორითმის

მეორე ეტაპის განხილვისას. თუ არჩეული გამოცდის გადატანა არჩეულ სესიაში შეუძლებელი აღმოჩნდა, მეთოდი ბრუნდება ამონახსნში ცვლილებების შეტანის გარეშე. სხვა შემთხვევაში მოქმედებს ისე, როგორც მოყვანილია ფოლადის წრთობის სიმულაციის აღწერაში. კერძოდ, თუ სამიზნე ფუნქციის მნიშვნელობა ახალი ამონახსნისათვის უკეთესია არსებული ამონახსნის სამიზნე ფუნქციის მნიშნელობაზე, მაშინ მიმდინარე ამონახსნი ჩანაცვლდება ახლით. სხვა შემთხვევაში მეთოდი გმოითვლის უარესი ამონახსნის არჩევის ალბათობას, რომელიც მეთოდისათვის გადაცემულ ტემპერატურის მნიშვნელობაზეა დამოკიდებული, და დააგენერირებს შემთხვევით რიცხვს C#-ის Random კლასის გამოყენებით. თუ შემთხვევით რიცხვის მნიშვნელობა ნაკლები აღმოჩნდება ამონახსნის არჩევის ალბათობაზე, მაშინ ახალი ამონახსნი ჩაანაცვლებს მიმდინარეს.

`private bool swapExams(int examId, Session session_1, Session session_2)` - ეს მეთოდი გამოიყენება generateNextSolution მეთოდის მიერ და აგებს კემპეს ჯაჭვს examId-სა და session_2-სათვის. იგულისხმება, რომ session_1 არის ის სესია, რომელშიც დანიშნულია examId. მეთოდი იტერაციულად აგებს გამოცდების ორ სიას - პირველი სესიიდან მეორეში გადასატანი გამოცდების სია, და მეორე სესიიდან პირველში გადასატანი გამოცდების სია. თუ კემპეს ჯაჭვი არსებობს, მაშინ მეთოდი მიმოცვლის გამოცდებს სესიებს შორის და დააბრუნებს true-ს. წინააღმდეგ შემთხვევაში დააბრუნებს fale-ს. აღსანიშნავია, რომ generateNextSolution მეთოდი swapExams მეთოდს არჩეული სესიების ასლებს გადასცემს რომ სამიზნე ფუნქციის მნიშვნელობის გამოთვლამდე ამონახსნში ცვლილებები არ აისახოს.

`private void saveCurrentSolution(long objectiveScore)` - გამოიძახება generateInitialSolution და generateNextSolution მეთოდების მიერ და მიმდინარე ამონახსნზე ინფორმაციას ამატებს pastExamSession ველში. ასევე objectiveScore-ის მნიშვნელობას, რომელსაც პარამეტრად იღებს, ამატებს Solution.objectiveScore სიაში და ზრდის Solution.iteration ცვლადის მნიშვნელობას ერთით.

`public void loadPastSolution(int iteration)` - ეს მეთოდი უკან აბრუნებს გამოთვლებს iteration იტერაციაზე მიღებულ ამონახსნზე. ამისათვის ის იყენებს pastExamSession ველში შენახულ მნიშვნელობებს. კერძოდ, ეს მეთოდი ცვლის Solution.examSessions ველში შენახულ მნიშვნელობებს და ყოველი სესიისათვის მინიღებული გამოცდების სიას. ეს მეთოდი

გამოიძახება მაშინ, როცა მომხმარებელი გრაფიკული ინტერფეისიდან მოითხოვს გამოთვლების უკან დაბრუნებას.

ExcelReader, ExaminationData და Solution კლასების ობიექტები იქმნება MainForm.cs კლასში და სწორედ აქედან ინიციალიზდება მნიშვნელოვანი გამოთვლები. MainForm.cs კლასი იძახებს Solution კლასის generateInitialSolution და generateNextSolution მეთოდებს. MainForm.cs კლასის მეთოდებიდან განსახილველია `btnResumeCalculation_click` მეთოდი, რომელიც გამოიძახება, როცა მომხმარებელი გრაფიკული ინტერფეისის Resume Calculation ღილაკზე დაკლიკავს. ეს მეთოდი წარმოადგენს ფოლადის წრთობის სიმულაციის მეთოდის იმპლემენტაციას. `btnResumeCalculation_click` მეთოდი იძახებს Solution კლასის generateNextSolution მეთოდს და მას გადასცემს მიმდინარე ტემპერატურას, რომელსაც MainForm.cs კლასის შესაბამისი ველი წარმოადგენს. generateNextSolution მეთოდის გამოძახებამდე მოწმდება, ხომ არ მოითხოვა მომხმარებელმა გამოთვლების შეჩერება, ან ხომ არ სრულდება ფოლადის წვრთნის სიმულაციის შეჩერების პირობა. generateNextSolution მეთოდის მუშაობის დასრულების შემდეგ იცვლება სისტემის ტემპერატურა.

პროგრამის მუშაობის სისწორის შემოწმებისათვის შემოღებულია SoltutionTest კლასი, რომელშიც განსაზღვრულია მეთოდები Solution კლასის გასატესტად. კერძოდ, ამ კლასში განსაზღვრული მეთოდებით მოწმდება, აქვს თუ არა ყველა გამოცდას მინიჭებული ზუსტად ერთი სესია და მოწმდება Solution კლასში განსაზღვრული ველების მნიშვნელობების ერთმანეთთან შესაბამისობა.

პროგრამის სამომავლო განვითარებისთვის შესაძლებელია ჰიბრიდული ალგორითმის მესამე და მეოთხე ეტაპების იმპლემენტაცია. ასევე, შესაძლებელია პროგრამული ინტერფეისის დახვეწა, მისი მოქნილობის გაზრდა და ახალი ფუნქციონალის დამატება. აპლიკაციას შეიძლება დაემატოს მონაცემთა ბაზა, რომელშიც ექსელიდან წაკითხული მონაცემებისა და პროგრამის მუშაობის შედეგების შენახვა იქნება შესაძლებელი და აღარ გახდება აუცილებელი ექსელის ფაილის ხელახალი ჩატვირთვა პროგრამის ყოველი გაშვებისას. გარდა ამისა, უფრო ხარისხიანი საგამოცდო ცხრილების მისაღებად შესაძლებელია შეიცვალოს სამიზნე ფუნქცია. როგორც ადრეც ვთქვით, სამიზნე ფუნქცია შეიძლება ბევრი კომპონენტისაგან შედგებოდეს, მაგალითად, სტუდენტის ინდივიდუალურ ცხრილში

გამოცდების ერთმანეთთან სიახლოვისათვის დაწესებული ჯარიმებისაგან. სასურველი იქნება, თუ მომხმარებელს ექნება საშუალება თავად შემოიღოს საგამოცდო ცხრილის ხარისხის მისთვის მნიშვნელოვანი მაჩვენებლები. თუ მომხმარებლისთვის სამიზნე ფუნქციის მთლიანად განსაზღვრის ფუნქციონალის დამატება დიდი სირთულე იქნება, შეიძლება საგამოცდო ცხრილის ხარისხის წინასწარ განსაზღვრული მაჩვენებლების შემოღება და მომხმარებლისთვის შესაძლებლობის მიცემა, რომ მათგან ერთი ან რამდენიმე აირჩიოს. ასევე, სასურველია რომ არა მხოლოდ ცალკეული ფაკულტეტისთვის იყოს შესაძლებელი ინდივიდუალური სასესიო პერიოდის მითითება, არამედ ცალკეული გამოცდისთვისაც. გარდა ამისა, კარგი იქნება თუ ცალკეული თარიღებისთვის სესიების რაოდენობის განსაზღვრა ინდივიდუალურად იქნება შესაძლებელი, ისევე როგორც ცალკეული სესიების მაქსიმალური ტევადობის განსაზღვრა.

როგორც ზემოთ აღვნიშნეთ, ამჟამად პროგრამას მონაცემები გადაეცემა ექსელის ფაილის სახით და მისი მუშაობის სისწორე აღნიშნული ფაილის ფორმატზეა დამოკიდებული. ეს საკმაოდ მოუხერხებელია. სასურველია, თუ პროგრამა მისთვის საჭირო საგამოცდო ინფორმაციას პირდაპირ sms.tsu.ge-სა lms.tsu.ge-ს ბაზიდან მიიღებს.

დასკვნა

ნაშრომში განხილულია საგამოცდო ცხრილის შედგენის ამოცანა და მისი სახესხვაობები, აღწერილია საგამოცდო ცხრილის შედგენის ამოცანის ამოხსნის ჰიბრიდული ალგორითმი, რომელიც გატესტილია თბილისის სახელმწიფო უნივერსიტეტის მაგალითზე. ნაშრომში აგრეთვე აღწერილია აღნიშნულ ალგორითმზე დაფუძნებით შექმნილი პროგრამული უზრუნველყოფა, მისი იმპლემენტაციის დეტალები და მოხმარების სპეციფიკა.

ნაშრომიდან შეიძლება დავასკვნათ, რომ მასში განხილული ჰიბრიდული ალგორითმი სრულიად შეეფერება თბილისის სახელმწიფო უნივერსტიტეტის საჭიროებებს. წარმოდგენილი პროგრამული უზრუნველყოფის შედეგები დამაკმაყოფილებელია. ჰიბრიდული ალგორითმის განსაკუთრებულ უპირატესობად შეიძლება ჩაითვალოს მისი მუშაობის პროცესში მომხმარებლის ჩართულობის სიმარტივე. ვგულისხმობთ, რომ მომხმარებელს შეუძლია ნებისმიერ დროს დროებით შეაჩეროს ალგორითის მუშაობა და თავისი შეხედულებისამერ შეიტანოს ცვლილებები ამონახსნში, რითაც შეუძლია გავლენა მოახდინოს ალგორითმის მომავალი მუშაობის მიმართულებაზე. ეს მნიშვნელოვანი ფუნქციონალია, რადგან ჰიბრიდული ალგორითმის მიერ მიღებული გადაწყვეტილებები მხოლოდ სამიზნე ფუნქციის მნიშვნელობასა და ალბათობაზეა დამოკიდებული, გამოცდილ ადამიანს კი ინტუიცია ეხმარება. სწორი ინტუიციით ჰიბრიდული ალგორითმის შედეგის მნიშვნელოვანი გაუმჯობესება უნდა იყოს შესაძლებელი ზოგიერთ შემთხვევაში.

გამოყენებული ლიტერატურა

1. ლ. მერლოტი, ნ. ბოლანდი, ბ. ჰიუსი, პ. სტაკი - ჰიბრიდული ალგორითმი საგამოცდო ცხრილის შედგენის ამოცანისათვის

   L. Merlot, N. Boland, B. Hughes, P. Stuckey – A Hybrid Algorithm for the Examination Timetabling Problem

   https://www.researchgate.net/publication/2835832_A_Hybrid_Algorithm_for_the_Examination_Timetabling_Problem
2. https://www.mathworks.com/help/gads/how-simulated-annealing-works.html