ՀԱՅԱՍՏԱՆԻ ԱԶԳԱՅԻՆ ՊՈԼԻՏԵԽՆԻԿԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

Սաֆարյան Ներսես Աշոտի

ԳԵՆԵՏԻԿ ԾՐԱԳՐԱՎՈՐՄԱՆ ՕԳՏԱԳՈՐԾՄԱՄԲ ԹՎԱՅԻՆ ՊԱՏԿԵՐՆԵՐՈՒՄ ՕԲՅԵԿՏՆԵՐԻ ՀԱՅՏՆԱԲԵՐՄԱՆ ՀԱՄԱԿԱՐԳԵՐԻ ՄՇԱԿՈՒՄ

Ե.13.04 – «Հաշվողական մեքենաների, համալիրների, համակարգերի և ցանցերի մաթեմատիկական և ծրագրային ապահովում» մասնագիտությամբ

Տեխնիկական գիտությունների թեկնածուի գիտական աստիճանի հայցման

Ատենախոսություն

Գիտական ղեկավար՝

ֆիզ. մաթ. գիտ. դոկտոր, պրոֆեսոր

Հ. Գ. Սարուխանյան

ԵՐԵՎԱՆ 2016

Կառուցվածքային ցանկ

ՆԵՐԱԾՈՒԹՅՈՒՆ

Թվային պատկերներում օբյեկտների ճանաչումը լայն կիրառություն ունի բազմաթիվ բնագավառներում (թիրախների ճանաչման, տեսաշարերի մշակման և այլն): Այն լայնորեն կիրառվում է նաև ռազմական նպատակներով: Կիրառման օրինակներ են՝ դաշտում գտնվող տանկի տարբերակումը շրջակայքից, ծովում գտնվող նավի ճանաչումը, ավիավթարից տուժած ինքնաթիռի բեկորների հայտնաբերումն ու ճանաչումը և այլն: Պատկերներում օբյեկտների ճանաչման թվարկած վերջին տիպի կիրառություններում նպատակահարմար է օգտագործել արհեստական ռադիոտեղորոշման բացվածքի (ԱՌԲ) (Synthetic Aperture Radar (SAR)) [1, 2] (ռադիոտեղորոշային բացվածքը դա անտենայի արդյունավետ մակերեսն է [3]) նկարահանված պատկերները, քանի որ նման պատկերները կախված չեն ոչ՛ օրվա ժամից, ոչ՛ եղանակային պայմաններից: ԱՌԲ ռադիոտեղորոշիչ սարքի տիպե, որը օգտագործվում է երկրի մակերևույթի՝ լանդշաֆտի և երկրի մակերևույթի վրա գտնվող օբյեկտների նկարահանման համար: Այս ռադիոտեղորոշիչ սարքերը տեղադրվում են թռչող սարքերի վրա, որպեսզի բարձրությունից շարժման ժամանակ նկարահանվի հնարավորինս մեծ տարածք: Ռադիոլոկացիոն պատկերներ ստանալու համար, ԱՌԲ-ն իմպուլսներով ցրում է ռադիոալիքները և յուրաքանչյուր իմպուլսի անդրադարձը ձայնագրում: Ռադիոլոկացիոն պատկերները *մոխրագույն* (grey scale) պատկերներ են: Յուրաքանչյուր փիքսելի ինտենսիվությունը համապատասխանեցվում է տվյալ տարածքից անդրադարձած ռադիոալիքների համամասնությանը (proportion) [4]: Ստացված պատկերը կախված է բազում գործոններից, ինչպիսիքն են՝ օբյեկտի չափերը, ձևը, դիրքը, բաղադրությունը և այլն:

Թվային պատկերներում օբյեկտների ճանաչման խնդրի լուծման համար մշակվել են բազմաթիվ մեթոդներ: Այդ մեթոդներից է գենետիկ ծրագրավորմամբ օբյեկտների հայտնաբերման և ճանաչման մեթոդը: Գենետիկ ծրագրավորումը իրենից ներկայացնում է էվոլուցիոն հաշվարկային մեթոդների համախումբ: Այն լայն

կիրառություն ունի զանազան բնագավառներում (ինժեներական, դասակարգող համակարգերի, ազդանշանների և պատկերների մշակման և այլ բնագավառներում):

Համպոն (R. J. Hampo) և Մարկոն (K. A. Marko) [5] առաջիններն էին, ովքեր գենետիկ ծրագրավորման մեթոդը կիրառեցին ազդանշանների մշակման խնդրում: Նրանք առաջարկել են մի ալգորիթմ, որի օգնությամբ մշակվում են էլեկտրական շարժիչից ստացված ազդանշանները, ինչը թույլ է տալիս հետևել և ղեկավարել շարժիչի աշխատանքը:

Թակեթը (Tackett) մշակել է ինֆրակարմիր պատկերներում թրթուրավոր և անվավոր տրանսպորտային միջոցները աղմուկից կամ այլ օբյեկտներից տարբերակելու ալգորիթմը [6, 7]: Հովիարդը (Daniel Howard), Ռոբերտսը (Simon C. Roberts) և Բրանիկին (Richard Brankin) [8, 9] մշակել են արհեստական արբանյակներից ստացվող պատկերներից ափին գտնվող նավերի տեղի հայտնաբերման ալգորիթմ: Ինչպես նաև մշակել են ծրագիր, որը թույլ է տալիս հետևել երթևեկությանը, չափել մեքենաների արագությունները և այլն:

Դեյդը (J. M. Daida), Համասը (J. D. Hommes), Ռոսը (S. J. Ross) և Վեսեկին (J. F. Vesecky) գենետիկ ծրագրավորումը օգտագործել են արբանյակից ստացված բևեռային սառույցների պատկերները հետազոտելու նպատակով [10, 11]:

Յուը (Jiangang Yu) և Բանյուը (Bhanu) գենետիկ ծրագրավորումը օգտագործել են պատկերներում մարդկանց դեմքերի հայտնաբերման և ճանաչման խնդիրներում [12, 13]:

Թվային պատկերներում օբյեկտների ճանաչումն իրականացնելու համար անհրաժշտ է լուծել մի շարք խնդիրներ: Նշենք դրանցից մի քանիսը՝

1. Պատկերները զտիչների միջոցով մաքրել աղմուկից;
2. Պատկերը զտելուց հետո նրանում հայտնաբերել հետաքրքրության հատվածները;

3. Դասակարգել հետաքրքրության հատվածներում գտնվող օբյեկտները;
4. Ուսուցանել ծրագրային համակարգը, դասակարգված օբեկտները ճանաչելու համար;
5. Իրականացնել ճանաչում՝ օգտագործելով մշակված ծրագրային համակարգի "կուտակած փորձը":

Ատենախոսության շրջանակներում նախագծված և իրականացված է մի ծրագրային համակարգ, որում լուծված են վերոնշված խնդիրները: Պատկերները բնական աղմուկից մաքրելու համար գենետիկ ծրագրավորման միջոցով գեներացվում են բաղադրյալ օպերատորներ: Մշակված ալգորիթմով պատկերում հայնաբերվում են հետաքրքրության հատվածները և նրանցում գտնվող օբյեկտները դասակարգվում են ըստ դասերի, եթե համակարգի կողմից ճանաչելի են այդ օբյեկտները: Եթե համակարգը չի ճանաչում որևէ օբյեկտ, ապա հնարավորություն կա համակարգին ուսուցանելու տվյալ օբյեկտի տիպի օբյեկտների դասը ճանաչելու համար:

Ատենախոսությունը բաղկացած է չորս գլուխներից: Առաջին գլխում դիտարկված է գենետիկ ծրագրավորումը, նրա զարգացման փուլերը, գենետիկ ծրագրավորման մեջ կիրառվող գործողություններն ու ֆիտնես ֆունկցիաները: Երկրորդ գլխում դիտարկված են վերհիշյալ թվարկած խնդիրները մասամբ կամ ամբողջությամբ լուծող հայտնի մեթոդներն ու ալգորիթմները: Երրորդ գլխում նկարագրված են գենետիկ ծրագրավորման կիրառմամբ թվային պատկերներում օբյեկտների հայտնաբերման և ճանաչման ծրագրային համակարգի մշակման փուլերը: Չորրորդ գլխում դիտարկված է գենետիկ ծրագրավորման կիրառությունը պատկերներում օբյեկտների հայտնաբերման և ճանաչման խնդրում: Բերված է թվային պատկերներում օբյեկտների հայտնաբերման և ճանաչման մշակված ծրագրային համակարգի հակիրճ նկարագրությունը:

Ատենախտության նպատակն է գենետիկ ծրագրավորման կիրառմամբ ռադիոլոկացիոն պատկերներում օբյեկտների հայտնաբերման և ճանաչման արդյունավետ մեթոդների ու ալգորիթմների մշակումը և

դրանց հիման վրա ծրագրային համակարգերի ստեղծումը: Այդ նպատակով լուծվել են հետևյալ խնդիրները՝ մշակվել է գենետիկ ծրագրավորման օգտագործմամբ պատկերների ֆիլտրման մեթոդներ և կատարվել են նրանց ծրագրային իրականացումները, ինչպես նաև օբյեկտների հայտնաբերման և ճանաչման ալգորիթմների մշակումը և իրականացումը:

Հետազոտության մեթոդները: Ուսումնասիրությունները հիմնվել են օբյեկտների հայտնաբերման և ճանաչման խնդրին առնչվող թվային պատկերների մշակման մեթոդների վրա(գենետիկ ալգորիթմ, գենետիկ ծրագրավորում, նեյրոնային ցանցեր, տարատեսակ զտիչներ,դասակարգողներ,սեգմենտացիայի մեթոդներ և այլն):

Գիտական նորությունը

- Մշակված է գենետիկ ծրագրավորման կիրառմամբ, ռադիոլոկացիոն պատկերներում օբյեկտների հայտնաբերման և ճանաչման մեթոդ, որն անկախ է ռադիոլոկացիոն սարքի պարամետրերից:
- Մշակված է օբյեկտների ճանաչման մեթոդ, որի դեպքում ճանաչվող օբյեկտների դասերի քանակի աճը չի բերում այդ դասերին պատկանող օբյեկտների ճանաչման արդյունավետության նվազմանը:
- Մշակված է ռադիոլոկացիոն պատկերներում օբյեկտների հայտնաբերման և ճանաչման նոր և արդյունավետ ծրագրային համակարգ:

Արդյունքների կիրառական նշանակությունը: Մշակված ծրագրային համակարգը կարելի է օգտագործել տարբեր ոլորտներում՝ թվային պատկերներում օբյեկտները ճանաչելու խնդրի լուծման նպատակով: Համակարգը կարող է կիրառվել նաև գենետիկ ծրագրավորման և օբյեկտների ճանաչման բնագավառներում տարաբնույթ փորձեր իրականացնելու նպատակով: Մշակված ծրագրային համակարգը կարելի է օգտագործել այնպիսի կիրառական ասպարեզներում, ինչպիսին տարածքների հետազոտման և ռազմական ոլորտներն են:

Այս ոլորտներում դիտարկվող տեղանքից ստացված պատկերների հետազոտման անհրաժեշտություն կա, ընդ որում տեղանքի պատկերները հիմնականում ռադիոլոկացիոն պատկերներ են: Ծրագրային համակարգը հնարավորություն է ընձեռում ճանաչել ռադիոլոկացիոն պատկերները:

Պաշտպանության ներկայացվող դրույթները

- Ռադիոլոկացիոն սարքի բնութագրերից անկախ, գենետիկ ծրագրավորման կիրառմամբ, պատկերներում օբյեկտների հայտնաբերման և ճանաչման մեթոդը:
- Թվային պատկերներում օբյեկտների ճանաչման ալգորիթմը, որի դեպքում ճանաչվող օբյեկտների դասերի քանակի աճը չի բերում այդ դասերին պատկանող օբյեկտների ճանաչման արդյունավետության նվազմանը:
- Ռադիոլոկացիոն պատկերներում օբյեկտների հայտնաբերման և ճանաչման ծրագրային համակարգը:

Աշխատանքի արդյունքների հավաստիությունը հիմնավորվում է մշակված ծրագրային համակարգի կիրառմամբ ստացված մի շարք փորձնական արդյունքներով:

Աշխատանքի արդյունքների ապրոբացիան: Աշխատանքի հիմնական արդյունքները զեկուցվել են՝ ՀԱՊՀ տարեկան գիտաժողովներում (2008-2009թթ.), ՀԱՊՀ-ի հիմնադրման 75-ամյակին նվիրված ՀԱՊՀ ուսանողական հոբելյանական գիտաժողովում, 8-րդ տեղեկատվական տեխնոլոգիական միջազգային համաժողովին, CSIT-11, Երևան:

Հրատարակումներ: Աշխատանքի հիմնական արդյունքները ներկայացված են 7 գիտական հոդվածներում, որոնց ցանկը բերված է սեղմագրի վերջում:

Աշխատանքի կառուցվածքը և ծավալը: Աշխատանքը բաղկացած է ներածությունից, 4 գլխից, եզրահանգումից, օգտագործված գրականության ցանկից: Աշխատանքի ծավալը կազմում է 136 էջ:

Գլուխ1

ԳԵՆԵՏԻԿ ԾՐԱԳՐԱՎՈՐՈՒՄ

Այս գլխում բերված են գենետիկ ալգորիթմի և գենետիկ ծրագրավորման նկարագրություններն ու համեմատությունները: Ինչպես նաև նկարագրված են գենետիկ ծրագրավորման գործողություններն ու նրանց տիպերը:

1.1. Ներածություն

1954 թ. Նիլս Էոլ Բերրսելին (Nils Aall Barricelli) առաջին անգամ կիրառեց Էվոլուցիոն ալգորիթմները մոդելավորման խնդիրներում [14]: 1960-ականներին Էվոլուցիոն ալգորիթմները լայնորեն ճանաչվում էին որպես օպտիմալացման մեթոդ: Ինգո Ռիչենբերգը (Ingo Rechenberg) և իր խումբը ունակ էին լուծել բարդ ինժեներական խնդիրներ էվոլուցիոն ալգորիթմների միջոցով: 1964 թ. Լորենս Ջ. Ֆոգելը (Lawrence J. Fogel) էվոլուցիոն ալգորիթմները կիրառում էր վերջավոր ավտոմատներ գտնելու համար: Ավելին, կիրառվեց դասակարգող համակարգեր սովորեցնելու համար [14, 15] : Գենետիկ ծրագրավորման (ԳԾ) ժամանակակից "tree-based" ձևակերպումը (ընթացակարգային լեզուներով մշակված ծրագրեր, որոնք ունեն ծառատիպ կառուցվածք և հարմարեցված են ԳԾ-ի գործողություններին) տրվել է 1985թ.-ին Նիքել Լ. Քարմեր-ը (Nichael Lynn Cramer) [16]: Հետագայում այս աշխատանքը լայնորեն ընդլայնվեց Ջհոն Ռ. Կոզայի (John R. Koza) կողմից: Ջհոն Ռ. Կոզան հանդիսանում է

գենետիկ ծրագրավորման հիմնական մշակողներից մեկը: Նա առաջինն է գենետիկ ծրագրավորումը կիրառել բազմաթիվ օպտիմալացման և փնտրման բարդ խնդիրներում: 1990-ականներին գենետիկ ծրագրավորումը առավելապես կիրառվում է պարզ խնդիրներ լուծելու համար քանի որ այն պահանջում էր հաշվողական մեծածավալ ռեսուրսներ և տեսության մշակումը բավականաչափ բարդ էր: Վերջին տարիներին գենետիկ ծրագրավորումը ունի բազում յուրօրինակ կիրառություններ զանազան բնագավառներում՝ քվանտային հաշվարկներում, դասակարգման, փնտրման և այլ խնդիրներում: 2000թ. սկզբներին այդ բնագավառում որոշ քանակությամբ նվաճումներից հետո գենետիկ ծրագրավորման տեսությունը լայնամաշտաբ և արագ զարգացում ունեցավ:

1.2 Գենետիկ ալգորիթմը և գենետիկ ծրագրավորումն ու նրանց համեմատական վերլուծությունը

Հարմարվողականությունը (ադապտացիան) կենդանի համակարգերի հիմնական հատկություններից մեկն է, այն կենդանի օրգանիզմների արձագանքն է շրջակա միջավայրին: Պարզ կենդանի օրգանիզմները ադապտացվում են միջավայրին ավելի արագ քան առավել բարդ օրգանիզմները: Բարդ օրգանիզմներում ադապտացիան տեղի է ունենում որոշակի ժամանակահատվածում, որի ընթացքում կենդանի օրգանիզմը սովորում է արձագանքել միջավայրին՝ հիմնված կյանքի փորձի վրա: Շատ երկար ժամանակահատվածում էվոլուցիոն ադապտացիայի ազդեցությամբ կենդանի օրգանիզմները փոփոխվել են և հարմարվել շրջակամիջավայրին:

Բնական ադապտացիայի գաղափարը կիրառվել է այնպիսի արհեստական համակարգերում, ինչպիսիք են՝ կառույցների միկրոկլիման կարգավորող համակարգեր, հաշվողական համակարգերում (արհեստական բանականություն, պատկերների ճանաչում ծրագրային համակարգեր) և այլն:

Քոմփյութերային ծրագրերի ստեղծումը մարդկային գործունեության կարևոր ճյուղերից է, որում հիմնական ռազմավարությունը հաջողված լուծումների վերակիրառումն է, ինչպես նաև փորձի ու սխալների ընդգրկումը: Քոմփյութերային ծրագրերի ադապտիվ կառուցման մեխանիզմ է առաջարկել Կոզան [17], որն էլ անվանվել է գենետիկ ծրագրավորում, որն ենթադրում է հաջողված լուծումների վերակիրառում և փորձի ու սխալների ընդգրկում: Գենետիկ ծրագրավորումը կիրառում է էվոլուցիոն ադապտացիայի սկզբունքը, որը հիմնվում է Հոլանդի գենետիկ ալգորիթմի վրա [18], որպեսզի ստեղծի ընթացակարգային (պրոցեդուրային) ծրագրեր: Գենետիկ ծրագրավորումը հանդիսանում է արհեստական բանականության ճյուղ, որի նպատակն է գեներացնել կամ տվյալների բազայից գտնել օգտագործողի կողմից նախապես սահմանված առաջադրանքը իրականացնող քոմփյութերային ծրագիր: Այն հիմնված է կենսաբանական էվոլուցիայի գաղափարի վրա և հանդիսանում է մեքենաներին սովորեցնելու հմտություն: Այն մասնագիտացված գենետիկ ալգորիթմ է (որի յուրաքանչյուր անհատ (սերնդի ներկայացուցիչ) հանդիսանում է քոմփյութերային ծրագիր), որը կիրառվում է քոմփյութերային ծրագրերը սերնդեսերունդ օպտիմալացնելու համար՝ համաձայն ֆիտնեսային լանդշաֆտի: Էվոլուցիոն կենսաբանության մեջ ֆիտնեսային լանդշաֆտները օգտագործվում են, որպեսզի ցույց տրվեն գենոտիպերի միջև կապը և վերարտադրողականության հնարավորությունը: Այն գենոտիպերը, որոնք շատ նման են, կոչվում են միմյանց “մոտիկ” և որոնք շատ տարբեր են՝ “հեռու” միմյանցից: Բոլոր հնարավոր գենոտիպերի բազմությունը, նրանց նմանությունների աստիճանը և նրանց համեմատության ֆիտնեսային արժեքը միասին անվանվում են ֆիտնեսային լանդշաֆտ [19]: Էվոլուցիոն օպտիմալացման խնդիրներում ֆիտնեսային լանդշաֆտը ֆիտնես ֆունկցիայի գնահատականն է: Ֆիտնես ֆունկցիաները օպտիմալացնող ֆունկցիաների մասնավոր տիպեր են, որոնց միջոցով հայտնաբերվում է օպտիմալ լուծում (այսինքն, գենետիկ ալգորիթմի որևէ անհատկարող է գերադասվել բոլոր մյուսներից):

Գենետիկ ալգորիթմի համառոտ ներկայացումը

Գենետիկ ալգորիթմը (ԳԱ) մշակվել է Ջոն Հենրի Հոլլանդը (Միչեգանի համալսարան) 1960-ականների սկզբին: 1970-ականների սկզբին Հոլլանդը ձևակերպեց կառուցվածքային թեորեմը (Schema Theorem) [20], որը ցույց է տալիս գենետիկ վերախմբավորման կարևորությունն էվոլուցիայի և ադապտացիայի խնդիրներում:

ԳԱ-ի հիմնական տարրերն են սերնդի (պոպուլացիայի) ընտրված անհատները: Պոպուլացիան (անհատների բազմությունը) հիմնականում ֆիքսված N չափի է, որը ժամանակի ընթացքում չի փոփոխվում: Ինչպես կենդանի օրգանիզմների պոպուլացիայում, այնպես էլ ԳԱ-ի պոպուլացիայում բացառապես նույն անհատների առկայությունը քիչ հավանական է: Անհատների բազմազանությունը շատ կարևոր է, այն թույլ է տալիս հաջորդ քայլերում ավելի լավ անհատներ ունենալ: Դասական ԳԱ-ում անհատը ֆիքսված երկարության երկուական (բինար) տող է (տես՝ նկար 2.1):

0 0 0 1 0 1 1 1 1 0 1 1 1 1 0 0

1 1 0 0 0 1 0 1 0 1 0 0 0 1 0 0

1 0 0 1 1 0 0 0 0 1 1 1 0 1 0 1

0 0 1 0 1 1 1 0 1 0 1 1 0 0 1 1

Նկար 2.1. Ութ երկարությամբ բինար 8 անհատներից (բինար տողերից) կազմված պոպուլացիա[20]:

Գենետիկայի տերմինալոգիայով անհատները (բինար տողերը) հանդիսանում են *գենոտիպեր*: ԳԱ-ի հիմնական նպատակն է տվյալ խնդրի պայմաններին բավարարող գենոտիպի սինթեզումը:

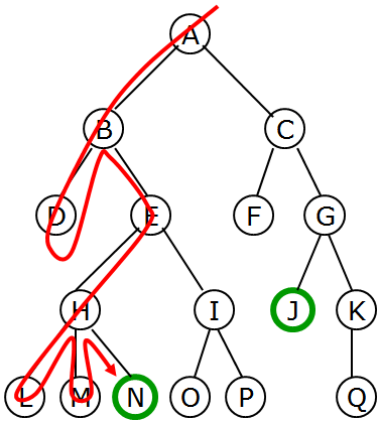
Գենետիկ ծրագրավորման համառոտ ներկայացումը

Գենետիկ ծրագրավորումը հանդիսանում է արհեստական բանականության (ԱԲ) ճյուղերից: Հետևաբար գենետիկ ծրագրավորման մեջ սահմանվում և կիրառվում են ԱԲ-ին բնորոշ գործիքներ և տերմինալոգիա: Նմանատիպ գործիքի օրինակ է հանդիսանում փնտրումը: Փնտրումը արհեստական բանականության մեջ հիմնական գործիքներից է: Ստորև բերվում են ԱԲ-ում և ԳԾ-ում կիրառվող փնտրման որոշ մեթոդներ:

Ուղղորդված գրաֆի գագաթների շրջանցում. փնտրման դասական մեթոդ է համարվում ուղղորդված գրաֆի գագաթների շրջանցումը [21]: Սկզբում ունենք ուղղորդված *սկզբնական վիճակը*, որը հանդիսանում է գրաֆի արմատը՝ "մակարդակ 0": Փնտրվող հաջորդ վիճակը (ծառի մակարդակը, հանգուցը) ստացվում է որոշ օպերատոր կիրառելով նախորդ վիճակի վրա, և այն հանդիսանում է նախորդ մակարդակի՝ հանգույցի *երեխան*: Ծնող հանգույցի երեխա հանգույց գեներացնող գործընթացը կոչվում է *ընդլայնում*: Այսինքն օպերատորները կիրառվում են 0 մակարդակի հանգույցների վրա, որպեսզի ստացվեն 1 մակարդակի հանգույցները: Գործընթացը շարունակվում է մինչ նպատակային վիճակին (հանգույցին) հասնելը, որը պետք է բավարարի նպատակային պայմաններին: Բազմաթիվ կիրառություններում փնտրումը նույնացվում է այն ճանապարհի (սկզբնական և նպատակային վիճակների հանգույցների միջև ընկած հանգույցների բազմությունը) հետ, որով անցել ենք ընդլայնման ընթացքում մինչ նպատակային հանգույցին հասնելը: Այսինքն, եթե նպատակը հայտնի է՝ փնտրվում է հենց ճանապարհը: Բացառությամբ նպատակից այլ ինֆորմացիաների բացակայության դեպքում տարբեր տիպի փնտրումներ են կատարվում: Նմանատիպ փնտրումների օրինակներ են՝ "լայնակի փնտրում"-ը և "խորությամբ փնտրում"-ը [20, 22-24]:

Լայնակի փնտրման դեպքում մինչև հաջորդ վիճակին անցնելը (մակարդակին անցնելը), պետք է անցնել տվյալ մակարդակի բոլոր հանգուցներով:

Խորությամբ փնտրման դեպքում տվյալ մակարդակի հանգույցները ընտրվում են որոշակի կանոնակարգով և նրանցով շարունակվող ճյուղով հասնում ենք մինչ տերևներ-y: Եթե տերևներից որևէ մեկը նպատակային հանգույց է, ապա փնտրումը ավարտվում է, եթե ոչ՝ ապա վերադառնում ենք նույն մակարդակին և մյուս ճյուղով ենք շարունակում փնտրումը և այդպես շարունակ (տես՝ Նկար 1.2.):



Նկար.1.2. Խորությամբ փնտրում գրաֆում [20]:

Էվրիստիկ փնտրումը հնարավոր է առավել արդյունավետ լինի քան նախորդ քննարկվածները, եթե որևէ ֆունկցիայի միջոցով գտնվում է տվյալ վիճակի հեռավորությունը նպատակից: Այդ ֆունկցիաները կարող են կիրառվել ցանկացած ընթացիկ վիճակի վրա և նրանց անվանում են *Էվրիստիկ ֆունկցիաներ* [25]: Էվիստիկ փնտրման դեպքում հնարավոր է դառնում հասնել նախապես անհայտ նպատակակետին, քանի որ կարող ենք ասել *հասել ենք նպատակին*, եթե էվրիստիկ ֆունկցիան կհասնի ցանկալի արդյունքի (նպատակից հեռավորությունը սահմանված չափի է): Գենետիկ ծրագրավորման մեջ ֆիտնեսը հանդիսանում է էվրիստիկ ֆունկցիա: Ծրագրի ֆիտնես արժեքի և նպատակային ֆիտնես արժեքի տարբերությունն անվանում ենք *հեռավորություն նպատակից*: Ծատ էվրիստիկ ֆունկցիաներ հիշում են նպատակին հասնող ճանապարհը, բայց գենետիկ ծրագրավորման մեջ կարևորը նպատակն է, այսինքն այն ծրագիրը, որը բավարարում է նպատակային ֆիտնեսին: Այս տիպի էվրիստիկ ֆունկցիաները օգտագործում են քիչ հիշողություն, քանի որ ճանապարհը չի հիշվում: Նմանատիպ փնտրում է Hill-climbing մեթոդը,

որն իրականացնում է *կլանող* ալգորիթմը [20, 26]: Այս ալգորիթմի դեպքում տվյալ վիճակի երեխաները գեներացվում են և գնահատվում ըստ էվրիստիկ ֆունկցիայի: Եթե տվյալ վիճակի երեխաների էվրիստիկ արժեքները նպատակին առավել մոտ են քան տվյալ վիճակը, ապա նրանցից ստանում ենք նոր *երեխա-վիճակներ* և գործընթացը շարունակվում է: Եթե ստացված *երեխա-վիճակների* էվրիստիկ արժեքները ավելի վատն են, ապա գործընթացը կասեցվում է: Hill-climbing անունը մեթոդը ստացել է այն բանի համար, որ ֆիտնեսի լանդշաֆտում գտնում է մոտակա բլրի գագաթը: Բայց պարտադիր չի, որ գտնված լոկալ մաքսիմումը համընկնի գլոբալ մաքսիմումին, հետևաբար այս ալգորիթմի դեպքում հնարավոր է գործընթացը ավարտվի առանց նպատակային ֆիտնեսին հասնելը:

Այլընտրանքային փնտրում է համարվում լավագույն-առաջինի (best-first) փնտրման մեթոդը [20]: Այս մեթոդի դեպքում հիշվում են էվրիստիկ եղանակով ստացված բոլոր վիճակները և դասավորվում են մեծից փոքր ըստ էվրիստիկ արժեքի: Ընդլայնման համար ընտրվում է լավագույնը (մեծ էվրիստիկ արժեք ունեցող վիճակը): Չնայած այս մեթոդը գլոբալ օպտիմալ վիճակի գտնելը երաշխավորում է, սակայն, խորությունից կախված պահանջվող հիշողության չափերը կարող են էքսպոնենցիալ կերպվաճել:

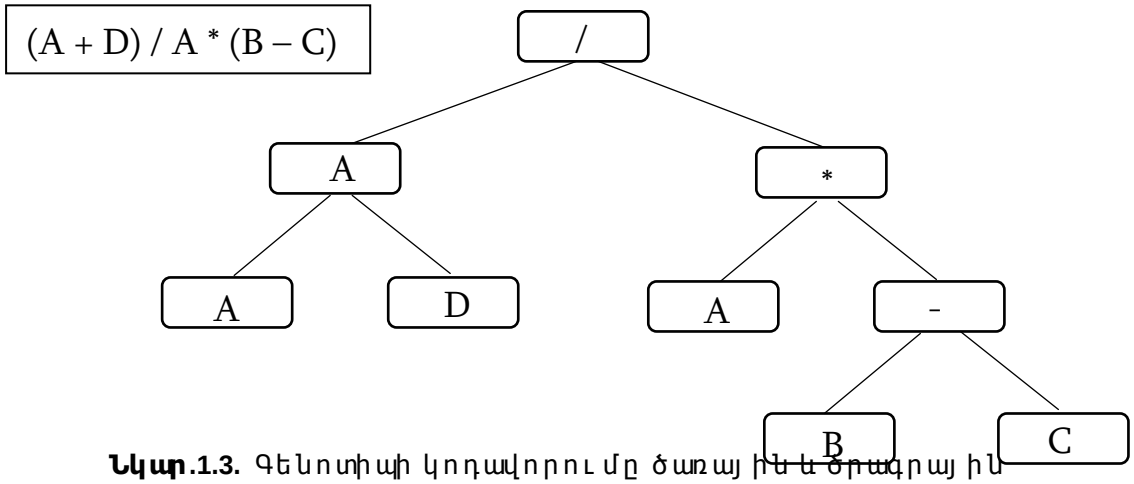
Բիմ փնտրումը [20, 27] շատ նման է լավագույնի փնտրմանը: Տարբերությունը միայն այն է, որ այս դեպքում պահանջվող հիշողության չափը նախապես տրվում է: Այսինքն, հիշվող վիճակների քանակը սահմանափակվում է: Նման սահմանափակման դեպքում հնարավոր է տվյալ վիճակը հասցնել որևէ օպտիմալ վիճակի: Շատ մեծ փոխկապակցվածություն կա գենետիկ ծրագրավորման և Բիմ փնտրման միջև:

Գենետիկ ալգորիթմի և գենետիկ ծրագրավորման համեմատական վերլուծությունը

Գենետիկ ալգորիթմի և գենետիկ ծրագրավորման հիմնական տարբերությունը անհատների կառուցվածքի և նրանց ղեկավարման

մեջ է: Գենետիկ ալգորիթմում լուծումները գտնվում են բիթերի տողերի տեսքով, իսկ գենետիկ ծրագրավորման մեջ՝ ծառերի կամ ծառատիպ սխեմաների տեսքով: ԳԱ-ում բիթերի տողերը հավասար չափի են, իսկ ԳԾ-ում ծառերի կառուցվածքները փոփոխվում են, ԳԱ-ում բիթերի տողը ստանալու համար օգտագործվում է բինար այբուբեն, իսկ ԳԾ-ում կախված խնդրից օգտագործվում են տարբեր չափերի և տիպերի այբուբեններ: ԳԾ-ի ծառերը բաղկացած են հանգույցներից և տերևներից, որոնք կախված խնդրից ընտրվում են տարրական էլեմենտների բազմությունից: Բիթերի տողերի հետ համեմատած ծառերը շատ ավելի ճկուն են: ԳԱ-ում պոպուլացիայի անհատները ֆիքսված քանակի և երկարության են և նրանց հատուկ է (ոչ պարտադիր) թվային բինար կառուցվածքը: ԳԾ-ում անհատները ներկայացվում են ծառերի տեսքով, որոնք իրենց հերթին կոդավորվում են որպես քոմփյութերային ծրագրեր:

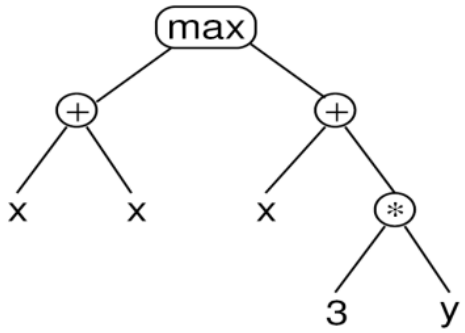
Գենոտիպը կոդավորվում է որպեսզի ստացվի տվյալ անհատի ֆենոտիպը, որի միջոցով հնարավոր կլինի գնահատել անհատի վերատադրողական որակները: Ֆենոտիպերի սահմանումը գենետիկ ծրագրավորման մեջ ավելի հետաքրքիր է դառնում, քանի որ այն իրենից ներկայացնում է քոմփյութերային ծրագրի վարքագիծ կամ սեմանտիկա: Այդ քոմփյութերային ծրագրերը ծրագրավորման լեզուներում հանդես են գալիս ֆունկցիաների հերթական և հաջորդական կանչերի (արտահայտությունների) տեսքով (տես՝ Նկար. 1.3):



Նկար.1.3. Գենոտիպի կոդավորումը ծառային և ծրագրային տեսքով:

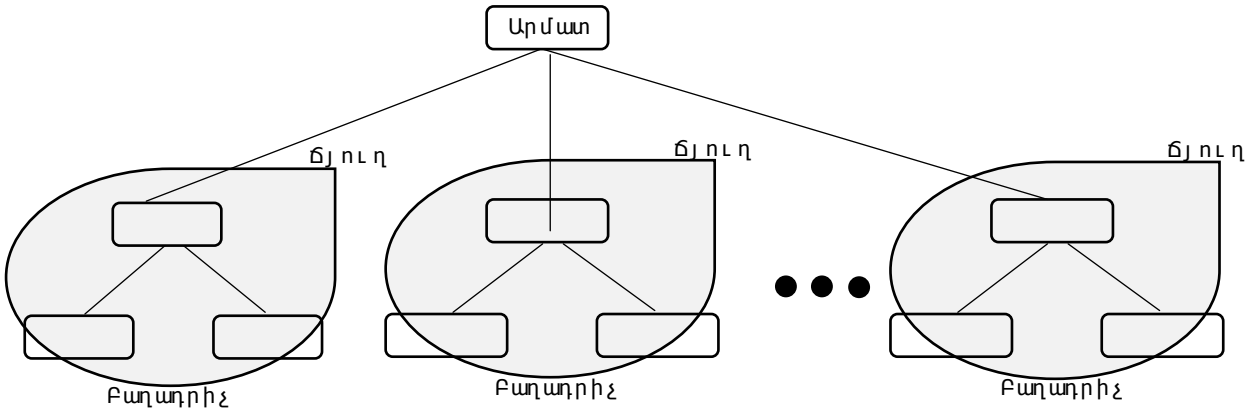
1.3 Գենետիկ ծրագրավորում

Գենետիկ ծրագրավորման մեջ անհատները (քոմփյութերային ծրագրերը) հաճախ ներկայացվում են ծառերի տեսքով [28]: Օրինակ՝ $\max(x + x,\ x + 3 * y)$ ծրագիրը ծառի տեսքով ներկայացված է Նկար. 1.4-ում



Նկար. 1.4. $\max(x + x,\ x + 3 * y)$ ծրագրի ծառի տեսքով

Այս ծրագրում փոփոխականները և հաստատունները (x, y և 3) հանդիսանում են ծառի տերևներ: Գենետիկ ծրագրավորման մեջ դրանք կոչվում են տերմինալներ: Մաթեմատիկական գործողությունները ներքին հանգույցներն են, որոնք կոչվում են ֆունկցիաներ (օպերատորներ): Թույլատրված ֆունկցիաների և տերմինալների բազմությունները միասին կոչվում են *գենետիկ ծրագրավորման տարրական բազմություն* [28]: Գենետիկ ծրագրավորման առավել բարդ ալգորիթմներում անհատները կարող են բաղկացած լինել մի քանի բաղադրիչներից: Այս դեպքում գենետիկ ծրագրավորման անհատները ներկայացնելու համար օգտագործվում են ծառերի բազմություններ (յուրաքանչյուր բաղադրիչի համար մեկ ծառ), որոնք համախմբվում են մեկ ընդհանուր *արմատ* հանգույցի տակ (տես՝ Նկար. 1.5): Այդ ենթածառերը անվանվում են ճյուղեր [28]:



Նկար 1.5. Բազմաբաղադրիչ ծրագրի ներկայացումը:

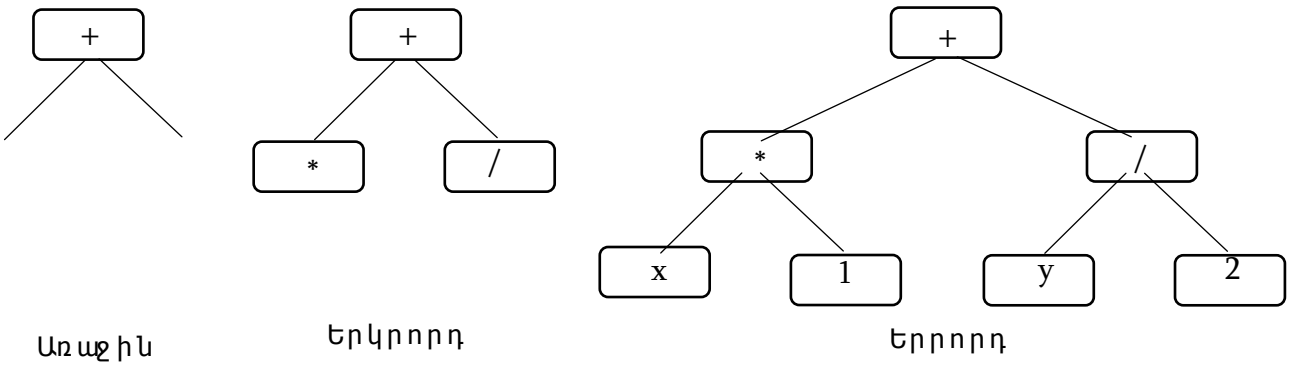
Սովորաբար գենետիկ ծրագրավորման արտահայտությունները ներկայացվում են այնպես, ինչպես Լիսփ ծրագրավորման լեզվում: Օրինակ՝ $\max(x + x,\ x + 3 * y)$ արտահայտությունը ունի հետևյալ տեսքը՝ (max (+ x x) (+ x (* 3 y))) [28]: Այս աշխատանքում գենետիկ ծրագրավորման անհատները հիմնականում ներկայացվում են ծառատիպ կառուցվածքներով: Գենետիկ ծրագրավորման ծառերի ծրագրային իրականացումը մեծապես կախված է օգտագործվող լեզվից և գրադարաններից: Այն լեզուներում, որոնցում գոյություն ունի garbage collection (ավտոմատ աղբ հավաքող) և դինամիկ ցուցակ (list) որպես տվյալների տիպ, կարող են ավելի հեշտ իրականացվել գենետիկ ծրագրավորման ծառերի ներկայացումը և գենետիկ ծրագրավորման գործողությունները [28, 20]:

1.3.1 Սկզբնական սերնդի գեներացում

Այլ էվոլուցիոն ալգորիթմների նման (Գենետիկ ալգորիթմ, էվոլուցիոն ռազմավարություն, էվոլուցիոն ծրագրավորում և այլն), գենետիկ ծրագրավորման մեջ նույնպես, սկզբնական սերնդի անհատները գեներացվում են պատահականորեն: Գոյություն ունեն սկզբնական սերնդի գեներացման տարբեր մոտեցումներ: Ստորև բերվում է երեք պարզ մեթոդներ՝ ամբողջական, աճման և կես - կես (խառը) [28, 29]:

Աճման և ամբողջական ալգորիթմների դեպքում սկզբնական անհատների խորությունը չի գերազանցում օգտագործողի կողմից նախապես նշված չափը: Հանգույցի խորություն է հանդիսանում այն կողմերի քանակությունը, որոնցով պետք է անցնել (սկսած արմատից, որի խորությունը համարվում է 0) այդ հանգույցին հասնելու համար: Ծառի խորությունը ամենախորը տերևի խորությունն է (օրինակ՝ Նկար 1.6–ում պատկերված ծառի խորությունը հավասար է 3):

Ամբողջական ալգորիթմի դեպքում (այդպես է կոչվում քանի որ այդ ալգորիթմով գեներացված ծառի բոլոր տերևները գտնվում են նույն խորության վրա) յուրաքանչյուր պատահական ընտրվող հանգույց ընտրվում է ֆունկցիաների բազմությունից մինչ ծառի նախապես սահմանված խորության հասնելը, ծառի մաքսիմալ խորության բոլոր պատահական հանգույցները ընտրվում են տերմինալների բազմությունից: Նկար 1.6-ում պատկերված են ամբողջական ալգորիթմով գեներացվող 2 խորությամբ ծառի ստացման քայլերը, որպես ֆունկցիաների բազմություն ընտրված է [+, -, *, /] բազմությունը, իսկ որպես տերմինալների բազմություն՝ [x,y,1,2]: Քանի որ ծառի նախապես սահմանված խորությունը 2 է, հետևաբար, առաջին և երկրորդ քայլերում ընտրումը պետք է կատարվի ֆունկցիաների բազմությունից, իսկ երրորդ քայլում տերմինալների բազմությունից:

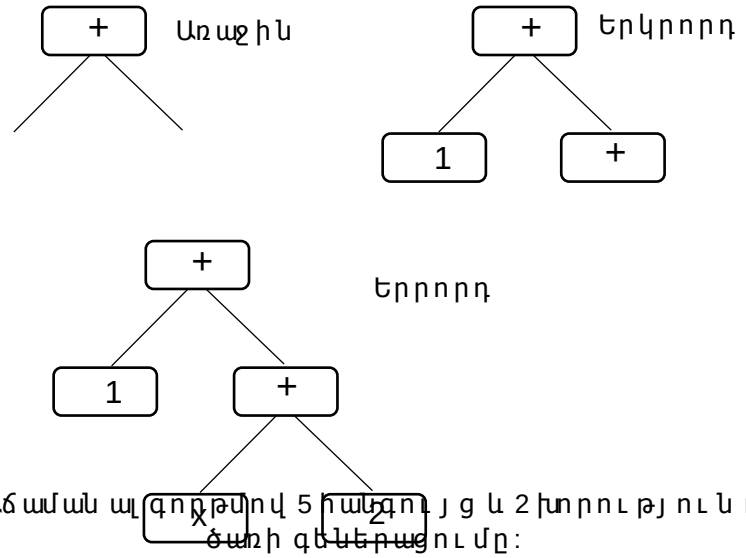


Նկար 1.6. Ամբողջական մեթոդով 2 խորությամբ ամբողջական ծառի

Չնայած այս դեպքում ծառի բոլոր տերևները նույն խորության են, դա չի նշանակում որ սկզբնական բոլոր ծառերը ունեն հավասար քանակության հանգույցներ (այդպես կլիներ, եթե բոլոր ֆունկցիաները ունենային հավասար քանակության արգումենտներ): Այնուամենայնիվ, եթե օգտագործում ենք տարբեր քանակի արգումենտներ ունեցող ֆունկցիաներ, միևնույն է ամբողջական ալգորիթմով գեներացված ծառերի չափերի և ձևերի բազմազանությունը սահամանափակ կլինի:

Աճման ալգորիթմը հնարավորություն է ընձեռում գեներացնել բազմազան ձևեր և չափեր ունեցող ծառեր: Այս դեպքում

հանգույցները ընտրվում են տարրական բազմությունից (տերմինալների և ֆունկցիաների բազմություններից) մինչև ծառի առավելագույն խորության հասնել: Առավելագույն խորության հասնելու դեպքում հանգույցները ընտրվում են միայն տերմինալների բազմությունից: Նկար 1.7-ում ներկայացված է աճման մեթոդով ծառի գեներացման գործընթացը: Այստեղ (+) արմատի առաջին արգումենտը պատահականորեն ընտրվել է տերմինալ (1, ինչը փակում է այդ ճյուղը), իսկ մյուսը՝ (+) ֆունկցիան, որի արգումենտները պետք է տերմինալներ լինեն, քանի որ ծառի առավելագույն խորությունը նախապես 2 էր ընտրվել և այս արգումենտները կունենան 2 խորություն:



Նկար 1.7. Աճման ալգորիթմով 5 հանգույց և 2 խորություն ունեցող ծառի գեներացումը:

Քանի որ աճման և ամբողջական ալգորիթմերն ինքնուրույն չեն կարող ծառերի ձևերի և չափերի լայն բազմազանություն ունենալ, հետևաբար Կոզան [28, 30] առաջարկեց կես-կես (խառը) ալգորիթմը: Այս դեպքում սկզբնական սերնդի անհատների կեսը ստացվում են ամբողջական, մյուս կեսը՝ աճման մեթոդներով, որն էլ ապահովում է սկզբնական սերնդի անդամների ձևերի և չափերի բազմազանությունը: Չնայած այս մեթոդները հեշտ իրականացնել, բայց դժվար է հսկել ծառերի ձևերի և չափերի վիճակագրական բաշխվածությունը: Օրինակ՝ եթե օգտագործվող տերմինալների բազմությունը շատ մեծ է ֆունկցիաների բազմությունից, ապա աճման մեթոդի դեպքում գեներացվող ծառերի խորությունները հաճախ շատ սահմանափակ կլինեն և հակառակը՝ եթե ֆունկցիաների

բազմություն է մեծ, ապա ճման ալգորիթմը կնմանվի ամբողջական ալգորիթմին:

1.3.2 Գենետիկ ծրագրավորման գործողությունները

Ինչպես բազմաթիվ էվոլուցիոն ալգորիթմներում՝ գենետիկ ծրագրավորման մեջ նույնպես *գենետիկ գործողությունները* կիրառվում են սերնդի ներկայացուցիչների վրա, որոնք պատահականորեն ընտրվում են կախված իրենց ֆիտնեսային արժեքից: Գենետիկ ծրագրավորման մեջ օգտագործվում են երեք տիպի գենետիկ գործողություններ՝ ընտրում, խաչասերում և մուտացիա:

Ընտրում. Ընտրում գործողությունը կիրառվում է սերնդից անհատ ընտրելու նպատակով: Գենետիկ ծրագրավորման մեջ սերնդից բաղադրյալ օպերատոր ընտրելու համար հիմնականում կիրառվում է մրցակցային ընտրման ալգորիթմը [28, 30], բայց կարող են օգտագործվել նաև ֆիտնեսային չափաբաժնային ընտրումը [31], վիճակագրական համընդհանուր ընտրումը [32] կամ այլ ընտրման ալգորիթմներ, որոնք օգտագործվում են էվոլուցիոն ալգորիթմներում: (Մեր աշխատանքում կիրառվել է մրցակցային ընտրման ալգորիթմը):

Մրցակցային ընտրման ալգորիթմ [28]. Մրցակցային ալգորիթմի դեպքում սերնդից պատահականորեն ընտրվում են որոշ քանակությամբ անհատներ: Ընտրված անհատները համեմատվում են և նրանցից լավագույնն ընտրվում է որպես ծնող: Մրցակցային ալգորիթմը դիտարկում է թե անհատներից ըստ ֆիտնեսի արժեքի որն է ավելի լավը, այլ ոչ թե որքանով է այն լավ: Այս ալգորիթմի դեպքում բացարձակ վատանհատը չի կարող անցնել հաջորդ սերունդ: Եթե այնուամենայնիվ դա տեղի ունենա, ապա գենետիկ ծրագրավորման հետագա աշխատանքի ընթացքում կկորցնենք անհատների բազմազանությունը: Մյուս կողմից մրցակցային ընտրման դեպքում անհատների ֆիտնեսային փոքր տարբերությունը բավարար է որպեսզի գերադասվի ավելի լավ անհատը: Քանի որ

մրցակցության համար անհատները ընտրվում են պատահականորեն, հետևաբար միջին ֆիտնեսային արժեք ունեցող անհատները նույնպես հնարավորություն ունեն ժառանգ ունենալու: Մրցակցային ընտրման ալգորիթմը գենետիկ ծրագրավորման մեջ առավել հաճախկիրառվող ընտրման ալգորիթմն է:

Ֆիտնեսային չափաբաժնային ընտրում **[31, 32].** Այս ալգորիթմի դեպքում ֆիտնեսային արժեքը օգտագործվում է անհատի և նրա ընտրման հավանականության մեջ կապ հաստատելու համար: Եթե fi -ը սերնդում i –րդ անդամի ֆիտնեսն է, ապա նրա ընտրման հավանականությունը կլինի (1.1) արտահայտությունը, որտեղ N-ը սերնդում ներկայացուցիչների քանակն է [31]:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \qquad (2.1)$$

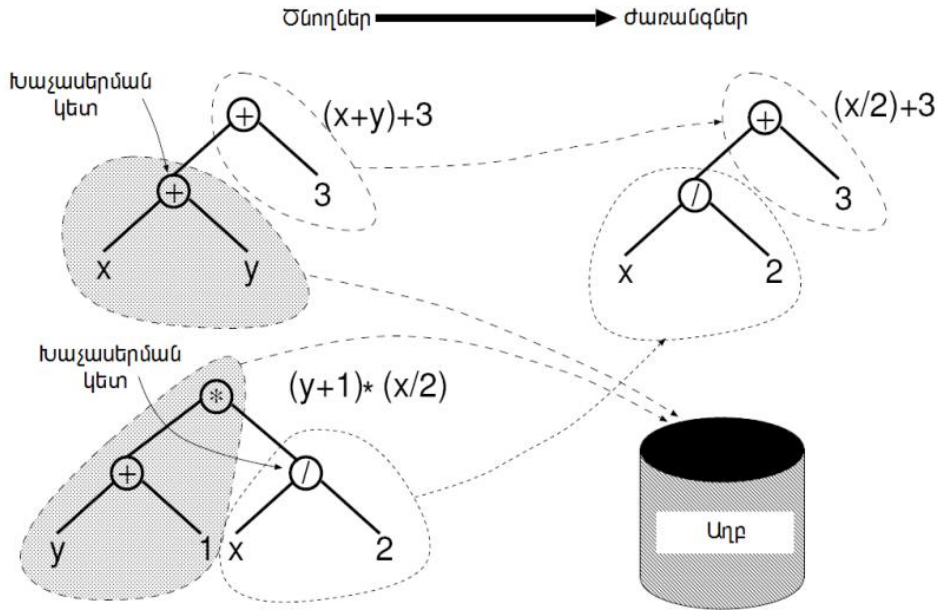
Այս ալգորիթմը կոչվում է նաև *պտուտահարկ* (Roulette Wheel Selection) ընտրում [32]: Ի տարբերություն իրական խաղատների ռուլետ անիվների, որոնք բաժանված են հավասար չափաբաժիններով, այս դեպքում որքան անհատի ֆիտնեսը մեծ է այնքան մեծ չափաբաժին է ունենում անիվի վրա: Անիվը պտտվում է այնքան անգամ, ինչքան պետք է հաջորդ սերունդ գեներացնելու համար անհրաժեշտ ծնողների ամբողջական բազմությունը ստանալու համար: Այս ալգորիթմի դեպքում հնարավոր է, որ մեկ կամ մի քանի անհատներ ընտրվեն մի քանի անգամ: Դա նորմալ է, քանի որ մի անհատը կարող է մի քանի ժառանգներ ունենալ:

Վիճակագրական համընդհանուր ընտրումը **[32]** հանդիսանում է ֆիտնեսային չափաբաժնային ընտրման մեթոդի զարգացումը: Վիճակագրական համընդհանուր ընտրումը երաշխավորում է տվյալ անհատի ընտրման հաճախականությունը նախապես սպասված քանակությամբ: Այսինքն, օրինակ, եթե տվյալ անհատի ֆիտնեսը հավասար է 4.6% և մենք պետք է ընտրենք 100 անհատ, ապա տվյալ անհատը պետք է ընտրվի 4 կամ 5 անգամ, ոչ ավել և ոչ պակաս: Ֆիտնեսային չափաբաժնային ընտրման մեթոդը դա չի երաշխավորում:

1.3.3 Խաչաերում և մուտացիա

Նախդիտարկենք խաչաերման գործողությունների որոշ տիպեր.

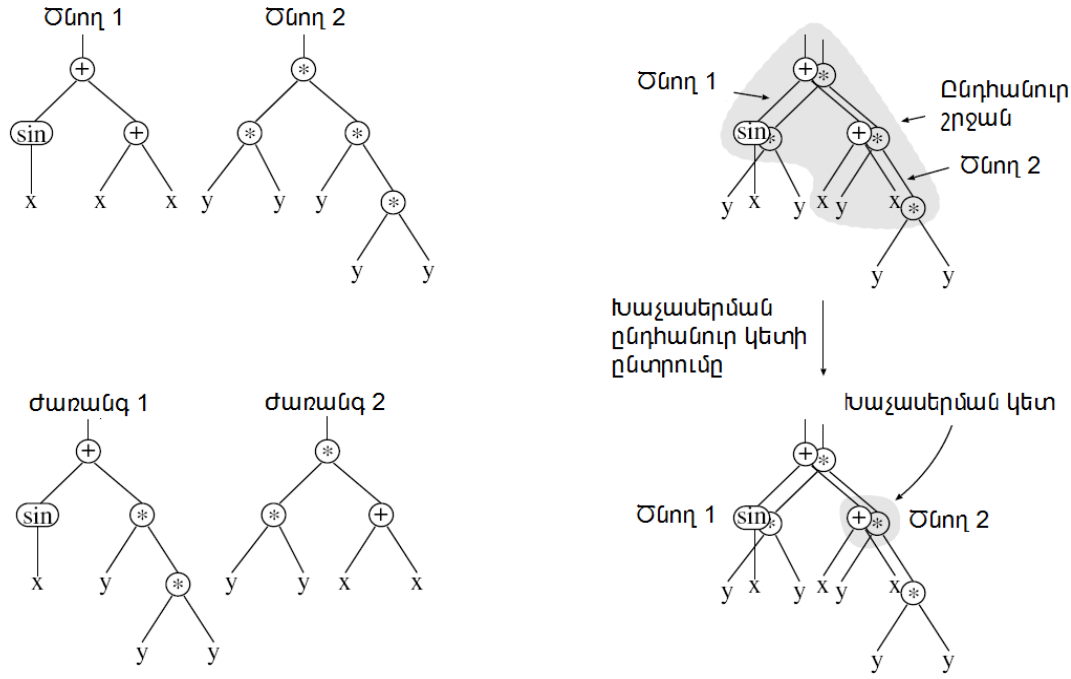
Ենթածառային խաչաերումը ամենատարածված եղանակն է [28, 30]: Ընտրվում են երկու անհատներ (ծնողներ, որոնք ներկայացվում են ծառի տեսքով): Ենթածառային խաչաերումը պատահականորեն յուրաքանչյուր ծնողում ընտրում է խաչաերման կետը (ծառի հանգույցը): Այնուհետև այն ստեղծում է ժառանգ՝ առաջին ծնողի կլոնում (պատճենում) փոխելով խաչաերման կետ հանդիսացող հանգույցից սկսվող ենթածառը երկրորդ ծնողի խաչաերման համար ընտրված հանգույցից սկսվող ենթածառով (տես՝ Նկար. 1.8): Պատճենները օգտագործվում են որպեսզի ծնողները չփչանան, քանի որ նրանք հետագայում կարող են այլ ժառանգներ ունենալ:



Նկար 1.8. Ենթածառային խաչաերման օրինակ:

Միակետխաչաերումը համարվում է ամենահին խաչաերման տեսակը [63]: Այն աշխատում է երկու ծնողներում ընդհանուր խաչաերման կետ ընտրելով և այդ կետերից սկսվող ենթածառերը տեղերով փոխելով, որպեսզի ստացվեն տարբեր ձևեր ունեցող ժառանգներ: Միակետ խաչաերումը աշխատանքը սկսում է դիտարկելով խաչաերման համար ընտրված երկու ծնողներին սկսած արմատից: Երկու ծնողներում որպես խաչաերման կետեր (հանգույցներ) ընտրվում են միայն այն հանգույցները, որոնցից սկսվող

ենթածառերը կառուցվածքով նույնն են (արգումենտների քանակը նույնն է)(տես՝ Նկար 1.9):

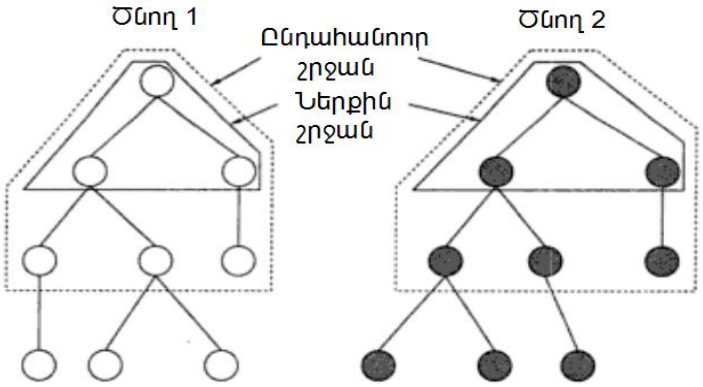


Նկար 1.9. Տարբեր ձև և չափ ունեցող ծնողների միջև միակետ

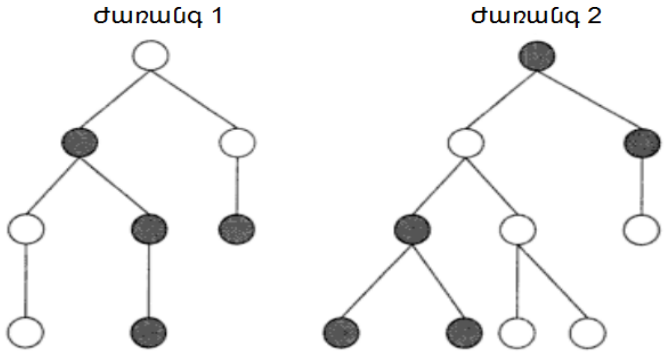
Միանման խաչասերումը [64] կարելի օգտագործել միայն այն դեպքում, երբ սերնդի բոլոր ներկայացուցիչները միանման են և ունեն նույն չափերը: Միանման խաչասերման դեպքում օգտագործվում է ծնողներում խաչասերման համար ընտրված հանգույցների 50 տոկոսի տեղափոխումը: Ալգորիթմը աշխատանքը սկսում է ստուգելով արմատ հանգույցը և շարունակում դեպի ներքև: Քանի որ խաչասերման համար ընտրված ծնողները միանման են, հետևաբար ալգորիթմը միանման ճանապարհով կանցնի մինչև տարբեր քանակությամբ արգումենտներ ունեցող հանգույցներին հասնելը:

Մենք կարող ենք այդ կետից վերև ընկած հանգույցները փոփոխել առանց ծնողները աղավաղելու: Աշխատանքը սկսելով արմատից և շարունակելով ներքև՝ կարող ենք նշել երկու շրջաններ: Յուրաքանչյուր հանգույց, եթե ունի համապատասխան հանգույց մյուս ծնողի նույն տեղում, ապա ընկած է ընդհանուր շրջանի մեջ: Ընդհանուր շրջանում ընկած այն հանգույցները, որոնք ունեն նույն քանակի արգումենտներ ընկած են ներքին շրջանի մեջ (տես՝

Նկար 1.10): Երբ ներքին հանգույցները հայտնի են դառնում գեներացվում են ծնողների պատճենները: Ծնողների պատճենների մեջ ներքին շրջանից որոշակի հավանականությամբ (0.5) ընտրվում են խաչասերման համար նախատեսված հանգույցներ: Ծնողների մեջ ընտրված հանգույցները տեղափոխվում են (տես՝ Նկար. 1.11):



Նկար 1.10 Միանման խաչասերման դեպքում ընդհանուր և ներքին շրջանների նշումը:



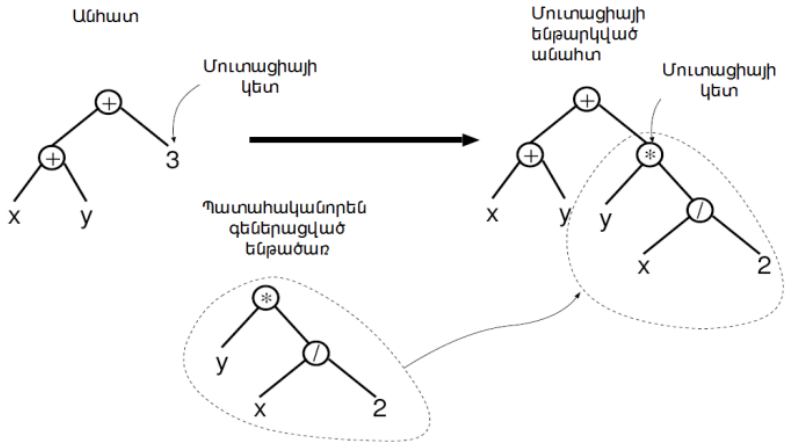
Նկար 1.11. Խաչասերումից ստացված ժառանգները:

Մուտացիան կիրառվում է սերնդի անհատների վրա փոխելով նրանց կառուցվածքը: Ստորև բերված են գենետիկ ծրագրավորման մեջ կիրառվող մուտացիաների որոշ օրինակներ.

Մուտացիայի ամենատարածված ձևը **ենթածառային մուտացիան** է [28, 30]: Այս դեպքում անհատում պատահականորեն ընտրվում է որևէ հանգույց (մուտացիայի կետ), և այդ հանգույցով սկսվող ենթածառը փոխարինվում է մեկ այլ պատահականորեն գեներացված ենթածառով (տես՝ Նկար 1.12): Այս ալգորիթմը ասոցացվում է սերնդի անհատի և պատահականորեն գեներացված անհատի խաչասերման հետ:

Կետային մուտացիան [28,30] անհատում պատահականորեն ընտրում է մուտացիայի կետ և այդ կետում գտնվող տարրական տերմինալը կամ

ֆունկցիան փոխարինվում է համարժեք տարրական ֆունկցիայով կամ տերմինալով: Եթե ընտրված հանգույցը ֆունկցիա է, ապա փոխարինման համար ընտրված տարրական ֆունկցիան պետք է ունենա նույն քանակի արգումենտներ որքան ուներ մուտացիայի կետ հանդիսացող հանգույցը:



Նկար 1.12 Ենթածառային մուտացիայի օրինակ:

Հավասարչափ (Size-fair) ենթածառային մուտացիա [28, 33]: Այս տեսակի մուտացիան նման է ենթածառային մուտացիային: Տարբերությունն այն է, որ փոխարինման համար պատահականորեն ընտրված ենթածառի և պատահական գեներացված ենթածառերի չափերը նույնը լինեն կամ ենթածառի չափերը պետք է լինեն հետևյալ շարքից՝ [L/2, 3L/2] (որտեղ L-ը անհատում պատահականորեն ընտրված ենթածառի խորությունն է):

Բարձրացվող մուտացիա [28, 34]: Անհատում պատահականորեն ընտրվում է ենթածառ և անհատը փոխարինվում է ընտրված ենթածառով: Այսինքն անհատի խորությունը փոքրանում է և փոխվում է նաև արմատհանգույցը:

Սեղմվող մուտացիա [28, 35]: Անհատում պատահականորեն ընտրված ենթածառը փոխարինվում է տերմինալներ բազմությունից պատահականորեն ընտրված տերմինալով: Սովորաբար բարձրացվող և սեղմվող մուտացիաները կիրառվում են այն դեպքերում, երբ պետք է անհատի չափերը փոքրացվեն:

1.3.4 Ֆիտնես ֆունկցիա

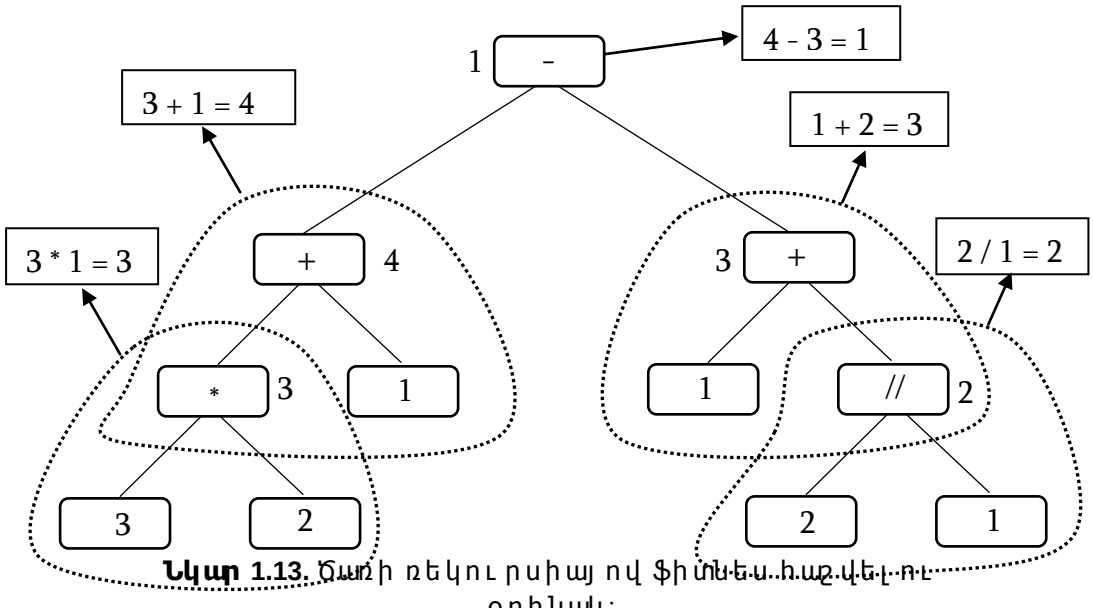
Ֆիտնես ֆունկցիաները օգտագործվում են գենետիկ ծրագրավորման անհատները գնահատելու համար: Այսինքն, այն պարզում է թե արդյոք տվյալ անհատը տվյալ խնդիրը: Ֆիտնեսը հաշվարկվում է

տարբեր եղանակներով: Օրինակ, արտադրության և ցանկալի արտադրության միջև սխալների քանակությունը, ժամանակի տևողությունը, որն անհրաժեշտ է համակարգը փնտրված վիճակին հասցնելու համար, կերպարները ճանաչող և դասակարգող ծրագրերի ճշտությունը, տվյալ կառուցվածքի համապատասխանությունը օգտագործողի կողմից նախապես սահմանված կառուցվածքին և այլն:

Ֆիտնես ֆունկցիաները տարբերվում են այլ էվոլուցիոն ալգորիթմների ֆիտնես ֆունկցիաներից, քանի որ գենետիկ ծրագրավորման անհատները ծրագրեր են, հետևաբար նրանց ֆիտնեսը հաշվելու համար պետք է աշխատացվեն ծրագիր անհատները: Գենետիկ ծրագրավորման անհատները աշխատացնելու համար հիմնականում օգտագործվում են ինտերպրետատորներ: Գենետիկ ծրագրավորման ծառ - ծրագրերը ինտերպրետատորով աշխատացնելը նշանակում է հերթով աշխատացնել ծառի հանգույցները հաշվի առնելով, որ յուրաքանչյուր հանգույց պետք է աշխատացվի իր երեխաների աշխատացվելուց հետո: Դա սովորաբար իրականացվում է ռեկուրսիվ եղանակով: Ռեկուրսիան սկսվում է արմատից և յուրաքանչյուր հանգույցի գնահատումը կատարվում է, երբ նրա երեխաների արժեքները արդեն հայտնի են դառնում: Նկար 1.13-ում դիտարկվում է 3 * 1 + 1 – (1+ 2 / 1) արտահայտության ծառատիպ կառուցվածքով ֆիտնեսի (տվյալ դեպքում արտահայտության արժեքի գտնելը) հաշվումը:

Որոշ խնդիրներում նպատակային է ծրագրի ելքային արժեքը, այսինքն ծառը արմատից սկսած հաշվարկելուց ստացված արժեքը (տես՝ Նկար 1.13, որտեղ 1-ը ծրագրի ելքին ստացված արժեքն է): Ուրիշ դեպքում նպատակային է համարվում ծրագրի գործողությունները, որոնք իրականացվում են որոշակի սխալանքով: Առաջին դեպքում ծրագրի ֆիտնեսը կախված է ծառի հաշվարկից ստացված արժեքից, իսկ երկրորդ դեպքում ծրագիրը տարբեր պայմաններում աշխատացնելուց: Որպես առաջին դեպքի օրինակ ծրագիրը կարող է փորձարկել մուտքին տալով x1, x2,..., xN պարամետրերի բոլոր հնարավոր համակցությունները կամ կարող է փորձարկվել ռոբոտի ղեկավարող ծրագիրը, փորձարկելով ռոբոտը տարբեր դիրքերից

շարժումը սկսելուց: Այս դեպքերը կոչվում են ֆիտնեսային դեպքեր [28]:



Նկար 1.13. Ծառի ռեկուրսիայով ֆիտնես հաշվելու օրինակ:

1.3.5 Գենետիկ ծրագրավորման պարամետրեր

Գենետիկ ծրագրավորման պարամետրերի համար հնարավոր չի գտնել օպտիմալ արժեքներ, քանի որ դրանք մեծապես կախված են գենետիկ ծրագրավորման տվյալ կիրառությունից: Այն հուսալի է և աշխատում է պարամետրերի տարբեր արժեքների համար: Օգտագործողը շատ ժամանակ չի կորցնի գենետիկ ծրագրավորումը փոփոխելու վրա, որպեսզի այն համապատասխանի իր խնդրին [28]:

Գենետիկ ծրագրավորման հիմնական պարամետրը հետևյալն են՝ ***սերնդի չափը***, ***սերունդների քանակը***, ***սկզբնական սերնդի անհատների խորությունը***, ***խաչաձևման գործակիցը***, ***մուտացիայի գործակիցը***, ***սերունդների քանակը և ֆիտնեսի շեմային արժեքը***:

Գենետիկ ծրագրավորման կարևոր պարամետրերից է **սերնդի չափը** (մեկ սերնդում անհատների քանակը): Սովորաբար սերնդի անդամների քանակը վերցվում է 500 (այսինքն՝ 500 անհատ մեկ սերնդում): Մեր խնդրում սերնդում անհատների քանակը 20-30 է:

Սերունդների քանակը ցույց է տալիս գենետիկ ծրագրավորման աշխատանքի ընթացքում ստացվող սերունդների հնարավոր մաքսիմալ քանակը: Սերունդների քանակը ընտրվում է 10-15, քանի որ

եթե առաջին սերունդներում չի գտնվել ցանկալի լուծումը, հետևաբար դժվար թե հետագա սերունդներում խելամիտ ժամկետներում ցանկալի լուծումը գտնվի:

Սկզբնական սերնդի անհատների խորությունը. սովորաբար սկզբնական սերունդ գեներացնելու համար օգտագործում են սկզբնական սերունդ գեներացնող կես-կես (խառը) ալգորիթմը: Սկզբնական սերնդի անհատների խորությունը ընտրվում է 2-7 միջակայքից:

Կա կարծիք, ըստ որի, պետք է սերնդի անդամների քանակը ինչքան հնարավոր է մեծացնել, սակայն, կան մարդիկ, որոնք առաջարկում են սերնդի անդամների քանակը փոքր վերցնել, բայց սերունդների քանակը մեծացնել [28]: Որոշ կիրառություններում չի պահանջվում ծառերի չափերի սահմանափակում, սակայն, քանի որ նրանց չափերը անկառավարելի են, սահմանափակում են կամ ծառերի չափը կամ ծառերի խորությունը և կամ երկուսը միասին:

Խաչասերման գործակիցը սահմանում է յուրաքանչյուր սերնդում անհատների քանակը, որոնք պետք ենթարկվեն խաչասերման: Այս գործակիցը հիմնականում ընտրվում է 90%, որոշ դեպքերում նաև 50%: (Մեր խնդրում խաչասերման գործակիցը 90% է):

Մուտացիայի գործակիցը սահմանում է յուրաքանչյուր սերնդում անհատների քանակը, որոնք պետք ենթարկվեն մուտացիայի: Այս գործակիցը խաչասերման գործակցի նման հիմնականում ընտրվում է 90%, որոշ դեպքերում նաև 50%: (Մեր խնդրում մուտացիայի գործակիցը 90% է):

Ֆիտնեսի շեմային արժեքը ֆիտնեսի նպատակային արժեքն է, այսինքն, եթե որևէ անհատի ֆիտնեսային արժեքը ստացվում է ֆիտնեսի շեմային արժեքից մեծ կամ հավասար, ապա տվյալ անհատը կհամարվի տվյալ խնդիրը լուծող անհատ:

Գենետիկ ծրագրավորման ալգորիթմի աշխատանքի ավարտման համար օգտագործվում են երկու պարամետր. *սերունդների քանակը* և *ֆիտնեսի շեմային արժեքը*: Գենետիկ ծրագրավորման աշխատանքը

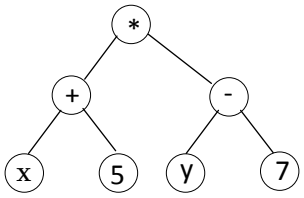
շարունակում է մինչ կգեներացվեն սերունդների քանակին հավասար սերունդներ կամ աշխատանքը կավարտվի ավելի շուտ, եթե մինչ այդ որևէ սերնդում գտնվում է անհատ, որի ֆիտնեսային արժեքը մեծ կամ հավասար է ֆիտնեսի շեմային արժեքից:

ԳԼՈՒԽ 2
ԳԵՆԵՏԻԿ ԾՐԱԳՐԱՎՈՐՄԱՄԲ ՊԱՏԿԵՐՆԵՐՈՒՄ ՕԲՅԵԿՏՆԵՐԻ ՃԱՆԱՉՄԱՆ ՄԵԹՈԴՆԵՐԻ ՎԵՐԼՈՒԾՈՒԹՅՈՒՆ

Այս գլխում քննարկվում են գենետիկ ծրագրավորման կիրառությունները ազդանշանների և պատկերների մշակման խնդիրներում: Ներկայացված են պատկերներում օբյեկտների հայտնաբերման և ճանաչման տարբեր մոտեցումներ: Ինչպես նաև կատարվել են քննարկված մեթոդների համեմատական վերլուծություններ:

2.1 Ինֆրակարմիր պատկերներում տրանսպորտային միջոցների տարբերակումը աղմուկից կամայլ օբյեկտներից

Tackett-ի կողմից գենետիկ ծրագրավորումը կիրառվել է հաշվարկային ծառեր մշակելու նպատակով, որոնք դասակարգում են պատկերից առանձնացված առանձնահատկությունները [6, 7]: Հաշվարկային ծառը մուտքին տրված չափումները համակարգում է գծային և ոչ գծային օպերատորների միջոցով՝ վերադաձվող արդյունքը ստանալու համար: Հաշվարկային ծառի օրինակ բերված է Նկար 1.1–ում, որտեղ պատկերված է $(x+5)*(y-7)$ արտահայտության ծառատիպներկայացումը:



Նկար 2.1. Արտահայտության ներկայացումը ծառի տեսքով:

Գեներացված ծառերը օգտագործվել են ինֆրակարմիր պատկերներում ընդգրկված օբյեկտները *թիրախներ* և *ոչ թիրախներ*

դասակարգելու համար [20]: Նշենք, որ Tackett-ի դիտարկված աշխատությունում թիրախ են համարվում տրանսպորտային միջոցները: Քանի որ ճանաչման գործընթացը կախված է առանձնահատկություններից և պատկերի սեգմենտացիայից, հետևաբար հնարավոր է դասերի ճանաչման շփոթմունք լինի, այսինքն, բացառվում է 100% ճիշգրիտ ճանաչումը [7]: Tackett-ը փորձերն իրականացրել է 7000 պատկեր պարունակող բազայով [20, 6]:

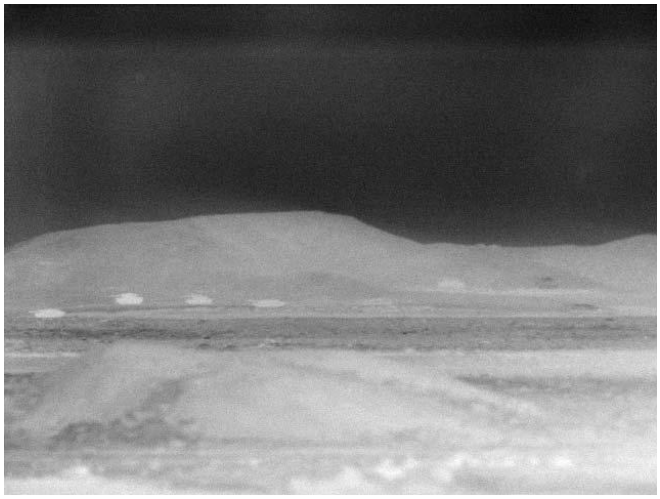
Երկու փորձ է իրականացվել՝ ***առաջին փորձում*** որպես տերմինալներ կիրառվել են դասական վիճակագրական առանձնահատկությունները (պատկերի հատվածի contrast-ը, ընդհանուր պատկերի ինտենսիվության միջին արժեքը, պատկերի ինտենսիվության ստանդարտ շեղումը և այլ բնութագրիչներ), թիրախ և ոչ թիրախ (օբյեկտ) դասակարգման սխալը նվազագույնին հասցնելու համար: ***Երկրորդում՝*** գենետիկ ծրագրավորումը հնարավորություն է տալիս գեներացնել պատկերի բաղադրյալ առանձնահատկություններ՝ օգտագործելով տարրական առանձնահատկությունները (տերմինալներ, ինտենսիվության որոշակի չափումներ, որոնք մանրամասն կդիտարկենք ավելի ուշ): Tackett-ը գենետիկ ծրագրավորումը օգտագործել այնպիսի խնդրում, ինչպիսին է՝ աղմուկ պարունակող պատկերը մշակելով և սեգմենտացնելով պատկերի առանձնահատկություններ ստանալը: Այնուհետև այդ առանձնահատկությունները օգտագործել է թիրախ և ոչ թիրախ օբյեկտները դասակարգելու համար: Պատկերների նույն բազան օգտագործվել է գենետիկ ծրագրավորումը համեմատելու երկու այլ դասակարգող համակարգերի հետ՝ բինար որոշման ծառերի [6, 36, 37] և նեյրոնային ցանցերի հետ (Multilayer Perceptron /Backpropagation) [6, 38, 39]:

2.1.1 Թիրախի ճանաչման համակարգի աշխատանքի նկարագրությունը

Վերը նշված 2 տիպի փորձերի համար օգտագործվել են ԱՄՆ-ի բանակի բազայից (NVEOD Terrain Board imagery) վերցված պատկերներ [20, 6]: Դրանք թրթուրավոր և անվավոր տրանսպորտային միջոցների, թռչող սարքերի, հակաօդային համակարգերի և աղմուկ պարունակող տեղանքի ինֆրակարմիր ճառագայթներով ստացված 512x600 չափանի

պատկերներն են: Ընդհանուր առմամբ օգտագործվել է 1300 պատկերներ, որոնք պարունակում են 13000 դասակարգման ենթակա օբյեկտ: Նրանց մեջ անհատական օբյեկտը զբաղեցնում է 100-300 փիքսել, տես՝ Նկար 2.2 [20, 6]: Փորձերը իրականացնելու համար որպես դասակարգող համակարգ կիրառվել է թիրախի ավտոմատ ճանաչման (ԹԱՃ) համակարգը (Multifunction Target Acquisition Processor - MTAP) [20, 6], որում իրականացված է գենետիկ ծրագրավորումը: MTAP համակարգը պատկերները մշակում է ստորև բերված փուլերի հաջորդականությամբ:

1. **Հետաքրքրության հատվածի առանձնացում**. Հետաքրքրության հատվածի առանձնացումը (ՀՀԱ), թիրախի ավտոմատ ճանաչման համակարգերի (կիրառվում է բոլոր այն համակարգերը, որոնցում ընդհանուր մեծ պատկերից ընտրվում են հետաքրքրության ենթահատվածները) համար սկզբնական կարևոր քայլն է: Պատկերների մշակման շատ արդյունավետ միջոց է որևէ պարզ հայտնաբերման ալգորիթմի կիրառումը, որը պատկերից կառանձնացնի հետաքրքրության հատվածները՝ այն հատվածները, որոնք հնարավոր է, որ պարունակում են օբյեկտ: Հետաքրքրության հատվածի [20, 30] առանձնացումը՝ Tackett-ի մոտնշվում է որպես թիրախոչ թիրախտարբերակում [20, 6]: Կոպիտ տարբերակմամբ այս հատվածները դասակարգվում են թիրախ և ոչ թիրախ դասերի (իհարկե, ավելի քիչ հուսալիությամբ, քան ճանաչող համակարգը), և միայն թիրախ համարվող հատվածներն են հայտնաբերող համակարգից տրվում ճանաչվող համակարգին՝ բարձրացնելով ճանաչվող համակարգի թողունակությունը:



Նկար 2.2. Տվյալների (NVEOD Terrain Board imagery) բազայից պատկերի օրինակ (դրական) [6]:

2. **AntiMean հայտնաբերման զտիչ (filter).** Համակարգը օգտագործում է AntiMean զտիչը պատկերից սպասվող օբյեկտների չափերին համապատասխան հատվածների (հետաքրքրության հատվածների) առանձնացնելու համար: Այդ հատվածները առանձնացնելու համար պատկերը բաժանվում է 62×62 չափանի հատվածների՝ մեծ պատուհաններ [20]: Այս հատվածները նույնպես բաժանվում են 5×5 չափանի հատվածների՝ փոքր պատուհաններ [20]: Երբ համապատասխան չափի հատված է հայտնաբերվում, նրանից առանձնացվում են 7 տարրական առանձնահատկություններ (տես՝ աղյուսակ 2.1-ը): Այնուհետև գենետիկ ծրագրավորումը կիրառվում է վերը նշված առանձնահատկությունները դասակարգելու համար: MTAP-ի ալգորիթմի համաձայն այս առանձնահատկությունները փոխանցվում են հաջորդ հաշվողական գործընթացին:

Աղյուսակ 2.1: Յոթ տարրական առանձնահատկություններ [6]:

F00	Շրջանակի contrast
F01	Ընդհանուր պատկերի ինտենսիվության միջինը
F02	Ընդհանուր պատկերի ինտենսիվության ստանդարտ
F03	Մեծ պատուհանի ինտենսիվության միջին
F04	Մեծ պատուհանի ինտենսիվության ստանդարտշեղում
F05	Փոքր պատուհանի ինտենսիվության միջին
F06	Փոքր պատուհանի ինտենսիվության ստանդարտ

3. **Սեգմենտացիա և առանձնահատկությունների առանձնացում.** MTAP համակարգում AntiMean filter-ից հետո ստացված յոթ առանձհատկությունները անցկացվում են պարզ գծային դասակարգողի միջով պարզելու համար, թե արդյոք իմաստունի սեգմենտացիան և առանձնահատկությունների առանձնացումը կիրառել տվյալ շրջանակի վրա [20]: Դրական արդյունքի դեպքում մեծ պատուհանը ենթարկվում է 3:1 դեսիմացիոն (decimation, ասինքն՝ յուրաքանչյուր 3 փիքսելից պատահականորեն ընտրվում է 1-ը) ֆիլտրացիայի, որպեսզի մեծացվի հաշվարկման թողունակությունը: Տեղային ինտենսիվության և contrast-ի վիճակագրական չափումները օգտագործվում են բինար սեգմենտացիայի համար, որի արդյունքում ստացվում է փակ ուրվագիծ [20, 6]: 20 առանձնահատկություններ են առանձնացվում այս սեգմնտացված հատվածից, որոնք ներկայացված են աղյուսակ 2.2-ում: Քանի որ սեգմենտացիան կախված է ընտրված շեմային արժեքից, հետևաբար ուրվագծի տեսքը խիստ փոփոխական է [30, 6]: Քանի որ առանձնահատկությունները մանրամասը չեն ներկայացնում օբյեկտի հատկությունները, հետևաբար, հնարավոր է, որ ոչ թիրախ օբյեկտները, ինչպիսին է օրինակ առանձին ժայռը, համարվեն թիրախ: Այսինքն՝ այս առանձնահատկությունները կարող են համընկումներ ունենալ և միաժամանակ պարունակեն շփոթություններ թիրախ դասակարգելուց:

Աղյուսակ 2.2. Քսան տարրական առանձնահատկություններ [20]:

F00	Ներգծված շրջանագծի շառավիղ
F01	Ուղղանկյուն
F02	Բարձրություն^2/երկարություն^2
F03	Երկարություն^2/Բարձրություն^2
F04	Նորմալացված contrast
F05	Սիմետրիկություն
F06	Թիրախից հեռավորություն

F07	Անկման անկյուն
F08	Պարագիծ^2/մակերեսի
F09	Մոխրագույն մակարդակի նորմալացված միջին
F10	Մակերես
F11	Բարձրություն^2/մակերես
F12	Բարձրություն$^{2/}$օբյեկտի հեռավորություն^2
F13-17	Բարձր կարգի մոմենտ
F18	Մակերես/օբյեկտի հեռավորություն^2
F19	Contrast-ի բևեռները

2.1.2 MTAP համակարգում կիրառված տերմինալներն ու ֆունկցիաները

Ֆունկցիաների բազմություն. Երկու փորձերում էլ օգտագործվում են ֆունկցիաների նույն բազմությունը՝ F = {+, -, *, %, IFLTE}: “+”-ը, “-“-ը և “*”- սովորական թվաբանական գործողություններ են, “%”-ը բաժանում գործողություն է, 0-ի վրաբաժանման դեպքում վերցվում է հենց իր արժեքը: Այս գործողությունները ներկայացվում են երկու արգումենտանոց ֆունկցիաներով: IFLTE(a,b,c,d)-ը չորս արգումենտանոց է: Եթե արգումենտներից առաջինը մեծ է երկրորդից վերցվում է երրորդը, հակառակ դեպքում՝ չորրորդը [20, 6]:

Առաջին փորձում օգտագործված տերմինալները և ֆունկցիաները. Առաջին փորձում օգտագործվել են Աղյուսակ 2.2-ում բերված տարրական առանձնահատկությունները՝ (T = {F00, F001, ..., F19, RANFLOAT}), ինչպես նաև պատահական RANFLOAT կոչվող արժեքը, որն ամեն անգամ հաստատուն արժեք է գեներացնում, երբ ընտրվում է որպես հանգույց: Արդյունքում առանձնահատկությունները (կոտորակային արժեքները և հաստատուն արժեքները) համախմբվում են գծային և ոչ գծային ֆունկցիաների ծառի տեսքով: Ծառի արմատում որպես վերջնական պատասխան ստացվում է թվային արժեք: Եթե այդ արժեքը մեծ կամ հավասար է 0-ից, ապա օբյեկտը կհամարվի թիրախ, հակառակ դեպքում՝ ոչ թիրախ:

Ծառերի թեսթավորումը 2000 պատկերների վրա իրականացվում է ի նկատի առնելով երկու չափանիշ: Առաջինը՝ դա ***սխալ դասակարգման հավանականությունն*** է կամ սխալ դասակարգված օբյեկտների չափաբաժինը ընդհանուրի մեջ: Եթե ուսուցման տվյալների (պատկերների) բազայում թիրախների քանակը մեծ է ոչ թիրախների քանակից, ապա ճանաչումը համարվում է արդյունավետ: Նկատենք, որ սակարող է հանգեցնել նրան, որ դասկարգողը ամեն ինչ (*թիրախև ոչ թիրախ* օբյեկտները) դասակարգում է որպես թիրախ: Դրանից խուսափելու համար դիտարկվում է մի այլ չափողականություն՝ դասակարգչի վերադարձրած արդյունքի (այսինքն՝ ***օբյեկտի որևէ դասի պատկանելության հավանականությունը***) բաշխման հաջորդականության էնտրոպիան - $H(\text{class}|\text{output})$ [20, 6, 40, 41]:

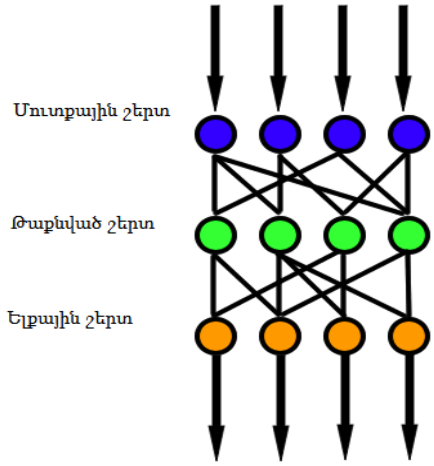
Այս բազմանպատակ խնդրի օպտիմալացումը կատարվում է այսպես կոչված Pareto օպտիմալացման օգնությամբ, որի դեպքում որևէ մասնակցի լավացումը չի բերում մյուսի վատացմանը, նրանցից ոչ մեկին չտալով որևէ առավելություն [42, 43]: Այս երկու չափողականությունները ձևավորում են անհատների ֆիտնեսները:

Երկրորդ փորձում օգտագործված տերմինալները և ֆունկցիաները. Երկրորդ փորձում օգտագործվում է միայն աղյուսակ 2.1–ում բերված յոթ առանձնահատկությունները (T = {F00, F01, ..., F06, RANINT}): RANINT-ը գեներացնում է բնական հաստատուն թիվ, եթե ընտրվում է որպես հանգույց: Այս փորձում ֆիտնես ֆունկցիան վերափոխվել է և հիմնականում այդ փոփոխությունները ներառվել են MTAP համակարգում: Ըստ այդմ հայտնաբերող ֆիլտրը պետք է կարողանա պատկերում հայտնաբերել հինգ թիրախ 80% հավանականությամբ: Սա նշանակում է, որ թիրախի հայտնաբերման հավանականությունը՝ $P(D)$-ն պետք է լինի $0.8^{0.2}$ կամ 96% [20, 6]: Միաժամանակ փորձ է արվում մինիմումին հասցնել կեղծ ազդանշանի հավանականությունը $P(FA)$, այսինքն, հավանականությունը այն բանի, որ ոչ թիրախը կդասակարգվի որպես թիրախ: Դա արվում է հետևյալ ձևով՝ հաշվարկային ծառը (անհատը) փորձարկվում է բազայում եղած օրինակների վրա և ֆիտնես արժեքները պահվում են առանձին զանգվածներում (մեկը թիրախի համար մյուսը աղմուկի): Այս

զանգվածները դասավորվում են ըստ աճման և որոշվում է շեմային արժեքը, որի դեպքում հենց 96% թիրախների ֆիտնեսները կլինեն ավելի մեծ: Այնուհետև համեմատում ենք այս արժեքը աղմուկի զանգվածի արժեքի հետ, որպեսզի տեսնենք սխալանքի տոկոսը տվյալ շեմային արժեքի համար, այսինքն, թե քանի տոկոսն է, որ սխալմամբ աղմուկ է գնահատվել: Այս տոկոսն էլ հենց կեղծ ազդանշանի գործակիցն է (False Alarm Rate) [20, 6], որն էլ փորձ է արվում մինիմալացնել:

1.1.3 Փորձերի իրականացման հակիրճ նկարագրությունը

Գենետիկ ծրագրավորման արդյունավետությունը գնահատելու համար նույն փորձերը միևնույն տվյալների բազայով իրականացվել են երկու այլ մեթոդներով: **Առաջինը բազմաշերտ ոչ գծային պերսեպտրոնային նեյրոնային ցանցն է** (multilayer nonlinear Perceptron) (ԲՊՆՑ), որն ուսուցանվում է Backpropagation ալգորիթմով [6, 38, 39] և ունի ելքի երկու հանգույցներ, մեկական յուրաքանչյուր դասի համար՝ *թիրախի* և *ոչ թիրախի*: Feedforward նեյրոնային ցանցերը՝ առաջին պարզ նեյրոնային ցանցերն են: Այս ցանցերում ինֆորմացիան տեղափոխվում է մեկ ուղությամբ՝ մուտքային հանգույցներից թաքնված հանգույցներով դեպի արտածման հանգույցներ: (ԲՊՆՑ) Feedforward նեյրոնային ցանցի տեսակներից է: (ԲՊՆՑ)-ն ներկայացվում է ուղորդված գրաֆի տեսքով և ունի հանգույցների երեք կամ ավելի շերտեր (մուտքի , ելքի և թաքնված մեկ կամ ավելի շերտեր): Գրաֆում յուրաքանչյուր շերտ ամբողջությամբ միանում է հաջորդին (տես՝ Նկար 2.3): Շերտերի քանակը հանդիսանում է ցանցի խորությունը [44]: Բացառությամբ մուտքային հանգույցների մյուս բոլոր հանգույցները հանդիսանում են ոչ գծային հաշվարկային էլեմենտներ: Որևէ շերտի յուրաքանչյուր հանգույց որոշակի կշռով միանում է հաջորդ շերտի բոլոր հանգույցներին: (ԲՊՆՑ)-ի ուսուցման մեթոդը կոչվում է backpropagation "backward propagation of errors" [45, 44]: Այն փնտրում է քաշերի տարածության սխալանաք ֆունկցիաների մինիմումը օգտագործելով գրադիենտի նվազեցման մեթոդը:



Նկար 2.3. Բազմաշերտ նեյրոնային ցանց:

Երկու փորձերի դեպքում էլ նեյրոնային ցանցերի համար ցանկալի արդյունք են համարվում {1.0, 0.0}, երբ դասակարգվում է թիրախ և {0.0, 1.0} երբ դասակարգվում է ոչ թիրախ (աղմուկ): Ուսուցողական մուտքերը նորմալիզացվում են այնպես, որ արդյունքը չգերազանցի 1.0-ին: Նեյրոնային ցանցի ելքերի շեմային արժեքները օգտագործվում են թրիափը և ոչ թիրափը տարբերակելու համար: Այդ շեմային արժեքը 0.0 է առաջին փորձի համար և փոփոխական՝ երկրորդի [20, 6]:

Երկրորդ փորձարկված մեթոդը ***որոշումների բինար ծառն է*** [6, 36, 37]: Այս բինար ծառը մշակվել և օպտիմալացվել է առաջին փորձում օգտագործելու համար, իսկ երկրորդում այն չի օգտագործվում:

Այս մեթոդների արդյունավետությունները համեմատելու համար հարց է առաջանում, թե արդյոք ինչ բնութագրով պետք է համեմատել: Եթե համեմատենք կենտրոնական հաշվողական սարքի (CPU)-ի աշխատանքի ժամանակը, արդար չի լինի, քանի որ գենետիկ ծրագրավորումը իրականացված է LISP ծրագրավորման միջավայրում, իսկ ԲՊՆՑ-ն՝ C-ով, որն ավելի էֆեկտիվ է, քան LISP-ը: Գենետիկ ծրագրավորումը կիրառվել է օգտագործելով ստանդարտ ֆունկցիաներ, որոնք ընդգրկված են ծրագրային ապահովման մեջ և նախկինում էլ կիրառվել են դասակարգման համար: Որոշ փորձեր են իրականացվել պարզելու համար պոպուլացիայի անդամների մինիմալ քանակը, որի դեպքում ժառանգների

վերարտադրողականությունը գնահատվում է բավարար: Նույն փորձը կատարվել է նեյրոնային ցանցերի համար, որպեսզի գտնվի թաքնված շերտերի և հանգույցների պահանջվող քանակը:

Արդյունքում համեմատության համար համեմատման հատկություն համարվել է մեթոդների ճշգրիտ աշխատանքը:

Թիրախ և ոչ թիրախ որոշելու նպատակով կատարված բոլոր փորձերում արդյունավետությունը հանդես է գալիս կեղծ ազդանշան հայտնաբերելու հավանականության $p(FA)$ և հայտնաբերման հավանականության $P(D)$ֆունկցիաների տեսքով:

Կեղծ ազդանշան հայտնաբերելու հավանականությունը ֆունկցիաէ, որը ստացվում է ոչ թիրախները որպես թիրախ ճանաչվածների քանակը բաժանած ընդհանուր ոչ թիրախների քանակին: Հայտնաբերման հավանականությունը ճիշտ դասակարգված թիրախների քանակը բաժանած թիրախների ընդհանուր քանակին: Իդեալական համակարգերում $p(FA)$ պետք է լինի 0,0 իսկ $P(D)$-ն՝ 1,0:

1.1.4 MTAP ծրագրային համակարգով իրականացված առաջին փորձի նկարագրությունը

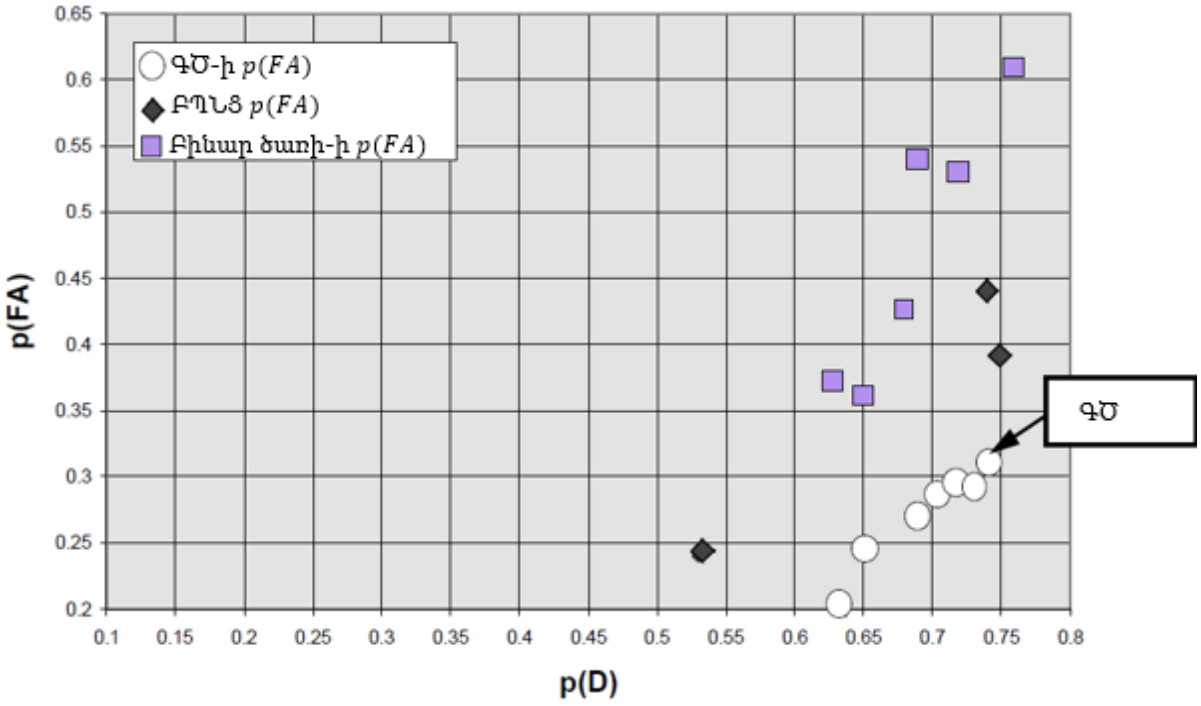
Փորձի ընթացքում 12 անգամ կիրառվել է գենետիկ ծրագրավորումը պատահական անհատներով: Երեք տարբեր աղմուկ/թիրախ հարաբերություններ (C/T գործակից) են օգտագործվել ուսուցման տվյալների բազայում՝ 0.5:1, 0.71:1 և 1:1 (0.5:1 լավագույն արդյունքով): Օրինակ՝ 0.5:1 հարաբերությունը նշանակում է, որ ընտրված բլոկը պետք է համարել օբյեկտի մաս կազմող բլոկ, եթե նրանում օբյեկտին պատկանող փիքսելների քանակը երկու անգամ մեծ է աղմուկին պատկանող փիքսելների քանակից [20, 6]:

Յուրաքանչյուր կիրառության ժամանակ հեղինակները [20, 6] օգտագործել են գենետիկ ծրագրավորման հետևյալ պարամետրերը. պոպուլացիայի չափը 500, սերունդների քանակը 60, յուրաքանչյուր աղմուկ/թիրախ հարաբերության (C/T գործակից) համար դիտարկվել են 120000 անհատներ:

ԲՊՆՑ-ն նույնպես փորձարկվել է օգտագործելով նույն 3 հարաբերությունները կրկին 0.5:1 լավագույն արդյունքով: Նեյրոնային ցանցը օգտագործել է 20 մուտքային հանգույցներ (առանձնահատկություններ) և 10 թաքնված (անտեսանելի) հանգույցներ: Յուրաքանչյուր հարաբերության համար օգտագործվել է պատահական սկզբնական կշիռների 4 բազմություն, ընդհանուր առմամբ 12 [20, 6]: Ուսուցման յուրաքանչյուր փուլից հետո ցանցը ստուգվում է առանձին թեստերով, որից հետո լավագույն արդյունքները գրանցվում են:

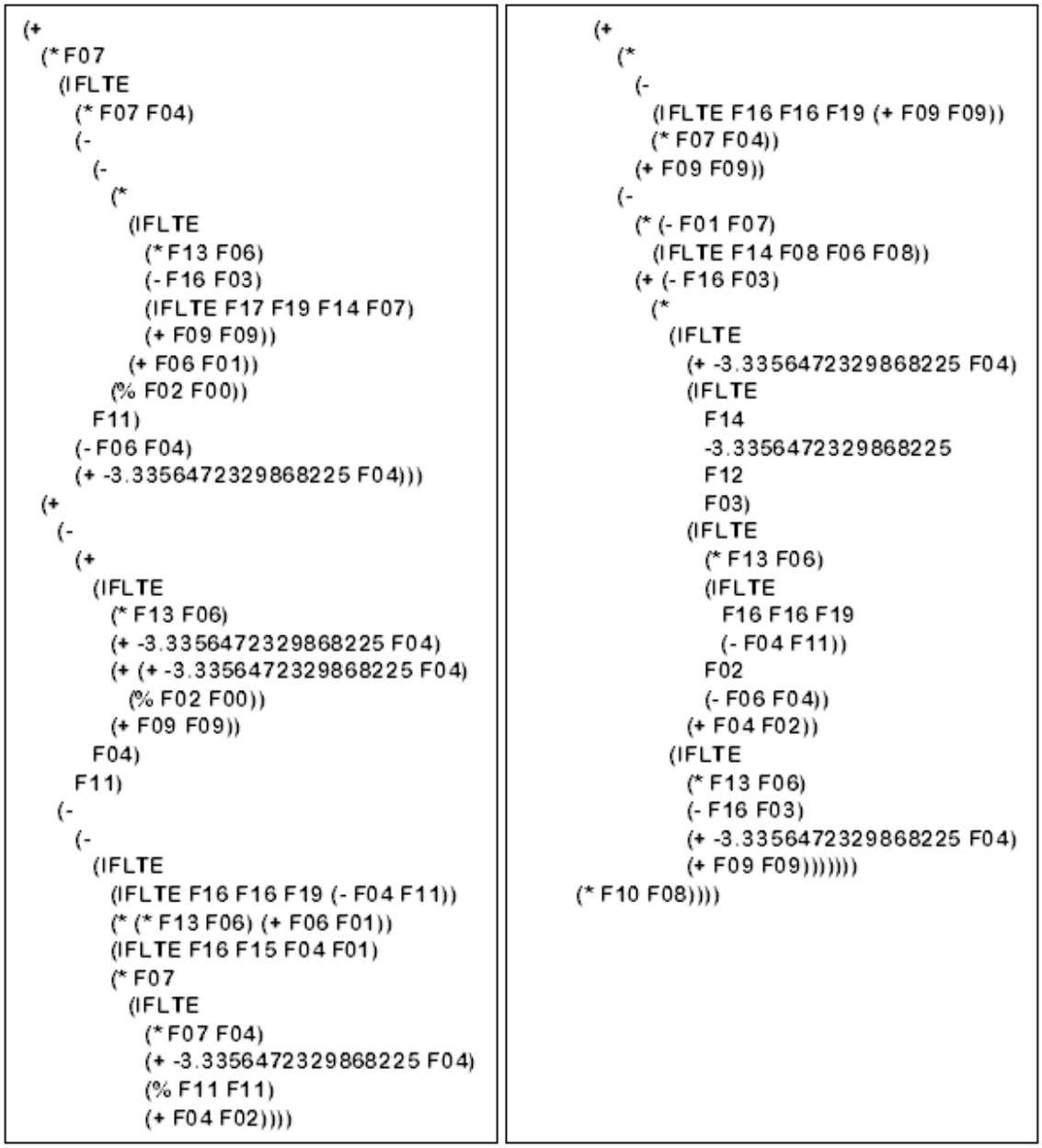
Որոշումների բինար ծառի միջոցով դասակարգումը փորձարկվել է օգտագործելով բոլոր 3 հարաբերությունները (աղմուկ/թիրախ): Ծառի կառուցման որոշ համակարգային պարամետրեր գեներացվում են մշակման ցիկլի ընթացքում:

Նկար 2.4-ում ներկայացված է երեք դասակարգողների փորձնական արդյունքները : Երեք կետերը հանդիսանում են ԲՊՆՑ-ի երեք հարաբերությունների համար կիրառելուց ստացված լավագույն արդյունքները: Վեց կետերը բինար ծառ դասակարգողի արդյունքներ են, որոնք համապատասխանում են երկու մուտքային առանձնահատկություններով երեք C/T գործակիցներին: Գենետիկ ծրագրավորման արդյունքները ներկայացնող 7 կետերը համապատասխանում են C/T 0.5 գործակցի համար մեկ էվոլուցիան աշխատանքը ընթացքում ընտրված սերունդների լավագույն անհատներին: Բոլոր երեք մեթոդներն էլ հասնում են $P(D)$-ի 74%-75% արդյունքի: Սակայն, գենետիկ ծրագրավորումը հասնում է 31% $p(FA)$ –ի, որը 8%-ով ավելի քիչ է քան ԲՊՆՑ -ի $p(FA)$-ն 39% նույն $P(D)$-ի դեպքում և 30%-ով քիչ, քան բինար ծառ դասակարգողի դեպքում $p(FA)$-ն 61% [20, 6]:



Նկար 2.4. ԳԾ-ի, նեյրոնային ցանցի և բինար ծառի արդյունավետության համեմատություն: Նպատակը $P(D)$-ն մաքսիմալ ին և $p(FA)$-ն մինիմալ ին հասցնել նէ [6]:

Նկար 2.5-ը ցուց է տալիս 48-րդ սերնդի լավագույն անհատը արտահայտված LISP ծրագրավորման լեզվի օպերատորներով և ֆունկցիաներով: Այս ծրագիրը հասնում է 74.2% ճշգրիտ $P(D)$ և 30.8% կեղծ աղմուկ $p(FA)$ արդյունքների: Կարող ենք տեսնել, որ կիրառվում են 55 մաթեմատիկական ֆունկցիաներ և 15 տրամաբանական գործողություններ: Մինչդեռ, նեյրոնային ցանցի դեպքում պահանջվում են 440 մաթեմատիկական գործողություններ և 12 ոչ գծային գործողություն:

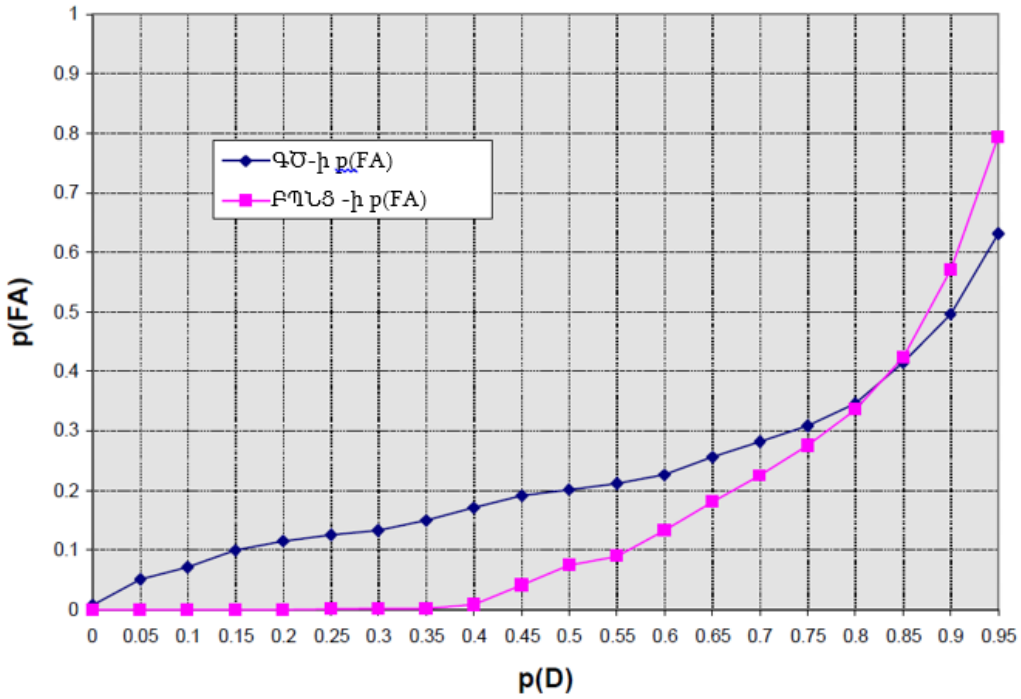


Նկար 2.5. ԳԾ-ի առաջին փորձի ժամանակ ստացված լավագույն անհատը [6]:

1.1.5 MTAP ծրագրայինհամակարգով իրականացված երկրորդ փորձի նկարագրությունը

Երկու մեթոդների համար էլ՝ գենետիկ ծրագրավորման և նեյրոնային ցանցերի, լավագույն արդյունքը ստացվել է C/T 0.5:1 գործակցի դեպքում: նկար 2.6-ում ներկայացվում է ԲՊՆՑ -ի և գենետիկ ծրագրավորման արդյունավետության գրաֆիկները: Գրաֆիկները ցույց են տալիս $P(D)$-ի ցանկալի արժեքները 96%-ի դեպքում: $P(D)$-ի լավագույն արժեքի համար գենետիկ ծրագրավորման

դեպքում ստացվում է $p(FA) = 62.8\%$, մինչդեռ ԲՊՆՑ-ի լավագույն $P(D)$-ի համար $p(FA)$-ն ստացվում է 82,6%: Փոփոխելով $P(D)$-ի ստացման շեմային արժեքները $p(FA)$ -ի համար ստացվում են նոր արժեքներ: Գենետիկ ծրագրավորման կիրառած ֆիտնես ֆունկցիան այնպես է կառուցված որ հնարավորություն կա $p(FA)$-ն փոփոխել շեմային արժեքով, իսկ ԲՊՆՑ-ի դեպքում շեմային արժեքը անկախ է: Այսինքն, գենետիկ ծրագրավորումը ընտրում է ֆունկցիա, որը փորձում է հասնել մեծ $P(D)$-ի և միաժամանակ փոքր $p(FA)$ -ի:

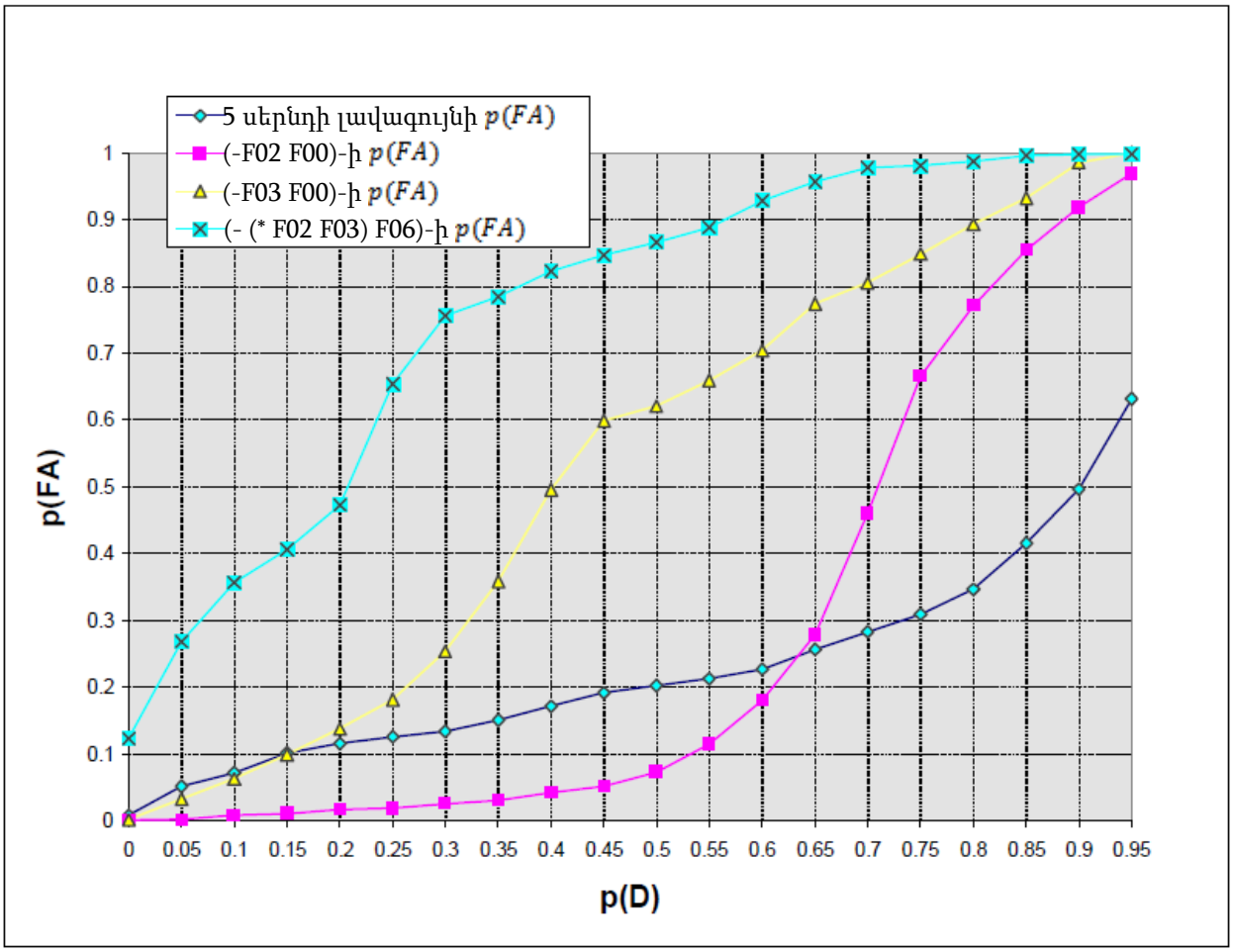


Նկար 2.6. Թիրախի հայտնաբերման և կեղծ աղմուկի միջև կարելի է հասնել փոխզիջումային տարբերակի շեմային արժեքը փոփոխելով: Երկրորդ փորձի նպատակն է $P(D)$-ի 96% - դեպքում նվազեցնել $p(FA)$-ն ինտենսիվության փոփոխության շնորհիվ:

Անիատը, որի աշխատանքը ներկայացված է նկար 2.6-ում, որպես LISP-ի արտահայտություն բերված է նկար 2.7-ում: Այն պահանջում է 12 մաթեմատիկական գործողություններ, իսկ ԲՊՆՑ-ը՝ 72 մաթեմատիկական գործողություններ և 6 ոչ գծային ֆունկցիաներ: Անիատը ունի մի քանի լավ հատկություններ՝ այն չի ներառում IFLTE պայմանական օպերատորը և չի օգտագործում F01, F04 և F05 առանձնահատկությունները, բացի այդ (-F02 F00) և (- F03 F00) առանձնահատկությունները կրկնվում են:

```
(% (- F02 F00)
  (+ (+ (% (- (* F02 F03) F06)
           (+ (- F02 F00) (- F03 F00)))
        (- F03 F00))
     (- F03 F00)))
```

Նկար 2.7. Երկրորդ փորձում ԳԾ-ի 5-րդ սերնդում ստացված լավագույն անհատը [6]:



Նկար 2.8. Ենթածառերի և ընդհանուր ծառի կիրառման արդյունքները:

Այս անհատի փոքր չափերը թույլ է տալիս քննարկել նրա հանգույցները առանձին առանձին: Նկար 2.8-ում բերված են Նկար 2.7 ծառից առանձնացված բաղադրիչ ենթածառերի կիրառման արդյունքները: Բաղադրիչ ենթածառերը կոչվում են գենետիկ ծրագրավորման կառուցման բլոկեր [18], որոնք տվյալ դեպքում հետևյալներն են՝ (- F02 F00), (- F03 F00) և (- (* F02 F03) F06): Ինչպես տեսնում ենք երեք ենթածառերի դեպքում էլ $P(D)$-ի 96% արժեքների դեպքում $p(FA)$-ն ստացվում է 100% արժեք: Այսինքն նրանք առանձին

ոչ արդյունավետ են տվյալ խնդրի համար, արդյունավետ է նրանց համակցությունը:

2.1.6 Փորձերի վիճակագրությունը

Մեթոդներից արդյունավետը ճշգրտորեն որոշելու համար պետք է հաշվի առնվեն նաև այլ փաստեր: ԳԾ-ն օգտագործում է ավելի բարդ ֆիտնես ֆունկցիա, քան ԲՊՆՑ, որի դեպքում այն ուղղակի սխալի միջին քառակուսին է: Սա չի կարելի անտեսել, քանի որ ԳԾ-ն սխալ չափելու կամ ֆիտնես ֆունկցիան հաշվելու միջոցներ չունի և ամեն անգամ պետք է իրականացվի ալգորիթմը, որով հնարավոր կլինի ստանալ ֆիտնեսային արժեքը: Սա կարող է դիտվել որպես առավելություն, քանի որ ԳԾ-ն ենթադրում է տվյալ խնդրի համար հատուկ ալգորիթմի իրականացում, իսկ ԲՊՆՑ-ի դեպքում ընդհանուր ալգորիթմ:

Որպեսզի ստացվեն խելամիտ տվյալներ և խնդրի լուծման ժամանակը և ռեսուրսները կրճատվեն, ԳԾ-ն ու ԲՊՆՑ -ը աշխատացվել են տասներկուական անգամ, որոնցից միայն չորսում է օգտագործվել բոլոր C/T գործակիցները: Ցանկալի կլիներ ալգորիթմները C/T գործակիցով կիրառել շատ անգամներ, միջին ե ստանդարտ շեղումը ստանալու և արդյունավետությունը բարձրացնելու համար: Չնայած վիճակագրական նման մեծ տվյալների բացակայությանը՝ ԳԾ-ն հասնում է ավելի մեծ արդյունավետության և հաշվարկի բարդության նվազեցման:

ԳԾ-ի դեպքում հաշվարկման ժամանակը զգալի մեծ է: Մեկ անհատով մեկ C/T գործակցով ուսուցման իրականացման դեպքում Sun SparcStation/2 մեքենայի վրա ԲՊՆՑ -ի համար պահանջվում է 20 րոպե, իսկ նմանատիպ փորձը ԳԾ-ի համար՝ 40-50 ժամ: Ավելի ուշ ստեղծվել է ԳԾ-ի C լեզվով իրականացումը: Այս ծրագրում ծառերը ներկայացվում են, այսինքն, գրվում և կարդացվում են LISP-ի արտահայտությունների տեսքով: Երկրորդ փորձը իրականացվել է այս ծրագրով և համեմատվել LISP-ով իրականացված տվյալների հետ:

Այն փորձի համար ծախսել է 2 ժամ, որը p6 անգամ քիչ է, քան LISP-ով իրականացված դեպքում, բայց 6 անգամ շատ, քան ԲՊՆՑ-ի դեպքում: Այսպիսով, կարող ենք եզրակացնել, որ ԳԾ-ը ԲՊՆՑ-ից ավելի ճշգրիտ է, սակայն ԲՊՆՑ –ն ավելի արագագործ է:

2.2 Գենետիկ ծրագրավորման կիրառությունը նախապես սահմանված քանակի դասերին պատկանող օբյեկտների հայտնաբերման խնդրում

Պատկերում նախապես սահամանված քանակի դասերին պատկանող օբյեկտների հայտնաբերման նպատակով, գենետիկ ծրագրավորման միջոցով գեներացվել և կիրառվել են քոմփյութերային ծարգրեր՝ ԳԾ-ի անհատներ: Այդ անհատները բաղկացած են տերմինալներից և օպերատորներից, այն կարելի է ներկայացնել ծառի տեսքով, որի արմատը և ներքին հանգույցները օպերատորներ են, տերևները՝ տերմինալներ:

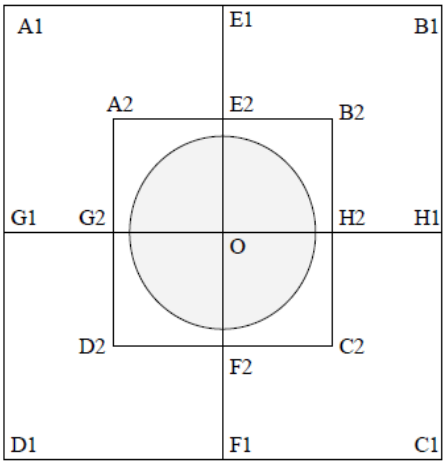
2.2.1 **Տերմինալների և օպերատորների բազմության նկարագրությունը**

Տերմինալների բազմություն: [46] աշխատանքում օգտագործվել են 20 տերմինալներ (տարրական առանձնահատկություններ) (տես՝ աղյուսակ 1.3-ը), որոնք ստացվում են մուտքային պատկերի այն քառակուսուց, որն ընդգրկում է հետաքրքրության հատվածը ամբողջովին [46, 47, 48] (տես՝ նկար 1.9-ը): Նկար 1.9-ում շրջանագիծը հանդիսանում է հետաքրքրության հատվածը, մյուս քառակուսիները ընդգծվում են առանձնահատկություններ ստանալու համար: Նկար 1.9-ում պատկերված են 6 քառակուսիներ, որոնց համար հաշվվում են պիքսելների միջին և միջին քառակուսային արժեքները, և ստացվում են աղյուսակ 2.3-ի 20 առանձնահատկությունները: Նկար 2.9-ում ցույց տրված առանցքներով ընդգրկված փիքսելների միջոցով ստացվում են F13-F20 առանձնահատկությունները [46]:

Օպերատորների բազմություն: [46]-ում օգտագործվում են add, sub, mul և div գործողությունները: Սրանք երկու արգումենտանոց օպերատորներ են, որոնցից առաջին երեքը աշխատում են իրենց

սովորական իմաստով, իսկ div օպերատորի դեպքում, եթե բաժանվում է 0-ի, ընդունվում է 0:

Կոպեկներ հայտանբերող բաղադրյալ օպերատորի օրինակ է նկար 2.10-ում պատկերված բաղադրյալ օպերատորը:



Նկար 2.9. Պատկերից առանձնատկությունների ստացումը հիմնված պատկերի ներքին հատվածի ինտեսիվության վրա[46].

Աղյուսակ 2.3. Քսան տարրական առանձնահատկություններ

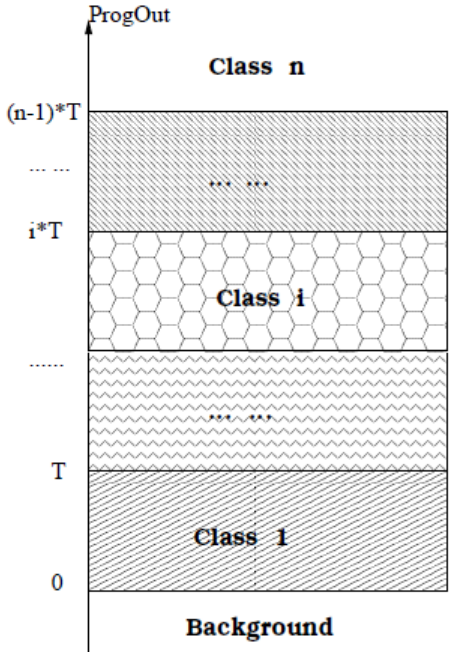
Առանձնա-հատկություն		Հետաքրքրության հատվածների տիրույթները և առանցքները
mean	standard deviation	
F1	F2	Մեծ քառակուսի A1-B1-D1-C1
F3	F4	Փոքր կենտրոն. քառակուսի A2-B2-D2-C2
F5	F6	Վերևի ձախ քառակուսի A1-E1-O-G1
F7	F8	Վերևի աջ քառակուսի E1-B1-H1-O
F9	F10	Ներքևի ձախ քառակուսի G1-O-F1-D1
F11	F12	Ներքևի աջ քառակուսի O-H1-C1-F1
F13	F14	Մեծ քառակուսու կենտրոնական տող G1-
F15	F16	Մեծ քառակուսու կենտրոնական սյուն
F17	F18	Փոքր քառակուսու կենտրոնական տող
F19	F20	Փոքր քառակուսու կենտրոնական սյուն

```
(+ (- (+ (+ (/ f16 f14) f5) (+ (/ (/ f11 (* f14 f20)) f11) (- f12
f14))) (- (* ( - (* (* (* f9 f11) f1) f10) (* f9 f17)) (/ f5 f18))
(- (+ (+ f17 (* (+ f11 f12) f20)) (* (- (+ f2 145.765) (/ f6 f11))
(- 133.082 f17))) (/ f11 (* f14 f20))))) (* (- (* (- (- f6 f5) (*
f3 f6)) (/ (+ (+ f1 145.765) (* f16 f10)) f18)) f12) (+ (+ f17 (*
(+ f17 f12) f20)) (* (+ f14 f12) (- (+ f1 f12) f17)))))
```

Նկար 2.10: Գեներացված ծրագիր (բաղադրյալ օպերատոր) պատկերից մետատադրամ հայտնաբերելու համար [46].

Յուրաքանչյուր գեներացված ծրագիր վերադարձնում է 1-ից փոքր թիվ, որը պետք է ցույց տա, թե տվյալ հետաքրքրության օբյեկտը որ դասին է պատկանում: Այդ նպատակին կարող ենք հասնել նկ. 2.11-ում ներկայացված քայլերով:

```
begin
    if (ProgOut ∈ (-∞, 0)) then
        the object is classified as background
    else if (ProgOut ∈ [0, T))
        the object is classified as class 1
    ... ... ... ...
    else if (ProgOut ∈ [(i-1)×T, i×T))
        the object is classified as class i
    ... ... ... ...
    else if (ProgOut ∈ [(n-1)×T, +∞))
        the object is classified as class n
    endif
end
```



Նկար 2.11: Հետաքրքրության օբյեկտների դասակարգումը, որտեղ n-ը հետաքրքրության օբյեկտների դասերի քանակն է, rogOut-ը գեներացված ծրագրի վերադարձվող արժեքն է, T-ն հաստատուն թիվ է [46]:

Այսինքն, նախապես որոշվում են n հատ T լայնության միջակայքեր, և, եթե ծրագրի վերադարձրած արժեքը պատկանում է i-րդ միջակայքին, ապա այն համարվում է i-րդ դասին պատկանող պիքսել (կետ):

2.2.2 Ֆիտնես ֆունկցիա

Յուրաքանչյուր սերնդի մեջ ծրագրի (անհատի) գնահատման համար օգտագործվող ֆիտնես ֆունկցիան հաշվում են օգտագործելով հայտնաբերման և աղմուկի գործակիցները, որոնք գտնվում են հետևյալ ձևով [46, 49, 29]՝

1. Յուրաքանչյուր մուտքային պատկերի ամբողջ մակերեսով անցկացվում է $n \times n$ չափանի պատուհան և այդ պատուհանով ծածկված մակերեսի վրա կիրառվում է տվյալ ծրագիրը (բաղադրյալ օպերատորը), և յուրաքանչյուր փիքսելի (այն ընթացիք փիքսելի, որը գտնվում է $n \times n$ չափանի պատուհանով ծածկված մակերեսի մեջտեղում) համար ստացվում է ծրագրի վերադարձվող արժեքը: Փիքսելների համար ստացված այդ արժեքները դասակարգվում են նկար 2.11–ում նկարագրված ալգորիթմով: Այս դասակարգումը կոչվում է *հայտնաբերման քարտեզ* [46]:

2. Գտնել հետաքրքրության օբյեկտների կենտրոնները նշանակում է հետազոտել հայտնաբերման քարտեզը որևէ օբյեկտի համար, երբ գտնվի այդ օբյեկտի փիքսել, այն նշել որպես կենտրոն և շարունակել փնտրումը n/2 փիքսել հետո:

3. Հայտնաբերված օբյեկտները համապատասխանեցնել տվյալ օբյեկտների իրական դասերի և դիրքերի հետ: Փիքսելը կհամարվի համընկած, եթե այն գտնվի իրական կամ նախապես սահմանված չափով շեղված (Tolerance pixels) կորդինատներում: Այսպիսով՝ հաշվվում են հայտնաբերման (Dr) և աղմուկի գործակիցները (Fr):

4. Ֆիտնեսը հաշվվում է հետևյալ ձևով [46]

$$fitness(Fr, Dr) = A * Fr + B * (1 - Dr), \qquad (2.1)$$

որտեղ A-ն և B-ն հայտնաբերման և աղմուկի համեմատական կարևորությունը ցույց տվող հաստատուններ գործակիցներն են:

Տվյալ դեպքում ֆիտնեսի ավելի փոքր արժեքի համար հայտնաբերման սխալանքը ավելի քիչ է:

2.2.3 Իրականացված փորձերի նկարագրությունը

Պարամետրերը: Փորձում օգտագործված տարբեր համակարգային պարամետրները բերված են աղյուսակ 2.4-ում [46-51]: Population-Size-ը ցույց է տալիս տվյալ սերնդում անհատների քանակը; Elitism-PCNT-ը ցույց է տալիս, թե տվյալ սեռնդում լավագույն անհատների քանի տոկոսն են կազմում, այդ անհատները փոփոխության չեն

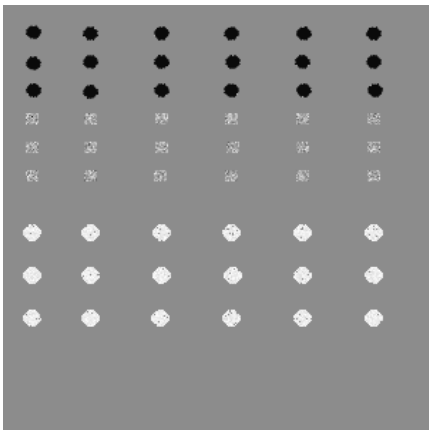
ենթարկվում և տեղափոխվում է հաջորդ սերունդ, Crossover-Rate-ը նշում է հաջորդ սերնդում անհատների քանակը, որոնք պետք է խաչասերվեն, Mutation_Rate-ը անհատների քանակն է, որոնք պետք է մուտացիայի ենթարկվեն հաջորդ սերնդում (Crossover-Rate + Mutation_Rate=100%), Cross-Chance_Term-ը ցույց է տալիս խաչասերման ժամանակ անահատներում (ծառերում) երկու հանգույցների տեղափոխության հավանականությունը, Cross-Chance_Func դա խաչասերման գործողության ընթացքում պատահական ծառերի ենթածառերի տեղափոխության հավանականությունն է (Cross-Chance_Term + Cross-Chance_Func=100%), Initial-Max_Depth-ը ընտրվում է որպես պատահականորեն գեներացված սկզբնական ծառերի խորություն, Max_Depth-ը դա ծառերի մաքսիմալ խորությունն է, Max-Generation-ով նախապես սահմանվում է սերունդների քանակը, այն ապահովում է ծրագրի ավարտը, T-ն, A-ն և B-ն արդեն նկարագրվել են:

Աղյուսակ 2.4. ԳԾ-ում օգտագործված պարամետրները [46]:

Parametrs	Հեշտ պատկերներ	Մետաղադրամի պատկերներ	Ցանցաթաղանթի պատկերներ
Population-Size	100	500	500
Elitism-PCNT	10%	1%	2%
Crossover-Rate	65%	74%	73%
Mutation_Rate	25%	25%	25%
Cross-	15%	15%	15%
Cross-	85%	85%	85%
Initial-Max_Depth	5	5	5
Max_Depth	8	12	20
Max-Generations	10	200	250
T	100	100	100
A	50	50	50
B	1000	1000	3000
Tolerance	2	2	2

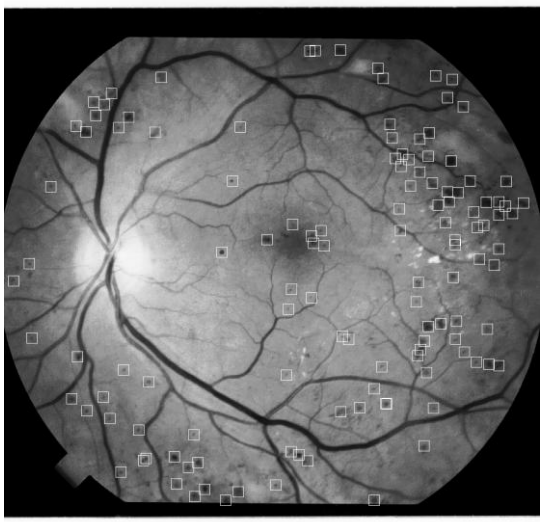
Պատկերների բազան: Փորձերը կատարվել են նկար 2.12-ում, 2.13-ում և 2.14-ում բերված պատկերների վրա: Նմանատիպ պատկերներ ընտրվել

են փորձերի բարդությունը մեծացնելու համար: Բազա 1-ը (պարզ) գեներացվել է լավ առանձնացված օբյեկտների և հաստատուն գունավորմամբ հետևի ֆոնի համար: Օբյեկտների փիքսելները գեներացվել են Գաուսյան գեներատորով [p46]: Մետաղադրամների պատկերը (բազա 2) ներառվել է խնդիրը բարդացնելու համար: Այն ստացվել է իրական ֆոտոխցիկով արված նկարներից: Նկարները արվել են տարբեր օրերում գրեթե նույն լուսավորության տակ: Այս պատկերի տարբեր հատվածներում հետևի ֆոնը ինչոր չափով տարբերվում է, սակայն, առկա է որոշակի համաչափություն: Ցանցաթաղանթի պատկերը (բազա 3) նկարվել է արհեստավարժ նկարողի կողմից իրական հիվանդանոցում հատուկ սարքավորումով: Արդյունքում ստացել է անհամաչափ օբյեկտներ և շատ աղմուկ պարունակող հետևի ֆոն:



Նկար 2.12. Պարզ պատկեր, դասերի քանակը 3, պատկերի չափը 700x700 [46]:

Նկար 2.13. Մետաղադրամներ պատկեր, դասերի քանակը 4, պատկերի չափը 640x480 [46]:



Նկար 2.14. Ցանցաթաղանթի պատկեր, դասերի քանակը 2 Պատկերի չափը 1024x1024 [46]:

Փորձի արդյունքները: Ներկայացված են օբյեկտի հայտնաբերման և ճանաչման մի քանի փորձեր: Արդյունքները համեմատվել են նեյրոնային ցանցերով ճանաչման արդյունքների հետ: Գենետիկ ծրագրավորումը և նեյրոնային ցանցերը կիրառվել են միևնույն բազաների վրա [46, 52, 53]:

Պարզ պատկերներ. Աղյուսակ 2.5-ում բերված են երկու մեթոդների փորձնական արդյունքները: Առաջի և երրորդ դասերի համար երկու մեթոդներն էլ ունեցել են հայտնաբերման 100% և կեղծ աղմուկի 0% գործակիցներ: Սակայն, երկրորդ դասի համար գենետիկ ծրագրավորումը նորից ունի 100% հայտնաբերման և 0% կեղծ աղմուկի գործակիցներ, իսկ նեյրոնային ցանցերը՝ 92% աղմուկի գործակից և 100% հայտնաբերման գործակից:

Աղյուսակ 2.5. Պարզ պատկերներում երեք դասերի օբյեկտների հայտնաբերման արդյունքների համեմատությունը:

Հեշտ պատկերներ		Օբյեկտների դասեր		
		Դաս 1	Դաս 2	Դաս 3
Հայտնաբերման լավագույն արդյունք (%)		100	100	100
Կեղծ աղմուկի գործակից (%)	Նեյրոնային ցանց	0	92	0
	ԳԾ	0	0	0

Մետաղադրամների պատկերներ. Մետաղադրամների պատկերների փորձերի արդյունքները բերված են աղյուսակ 2.6-ում: 5 ցենտ մետաղադրամի գլուխները և մետաղադրամի հակառակ կողմը համեմատաբար հեշտ հայտնաբերել: Եվ՛ նեյրոնային ցանցերով, և՛ գենետիկ ծրագրավորմամբ հայտնաբերման գործակիցը 100% է, իսկ կեղծ աղմուկի գործակիցը 0%: 20 ցենտանոցի գլուխները և հակառակ կողմերը համեմատած ավելի բարդ է հայտնաբերել, որոնց դեպքում նեյրոնային ցանցերը ունեցել են կեղծ աղմուկի մեծ գործակից: Գենետիկ ծրագրավորման դեպքում կատարվել է իդիալական հայտնաբերում:

Աղյուսակ 2.6. Մետաղադրամների պատկերներում չորս դասերի օբյեկտների հայտնաբերման արդյունքների համեմատությունը:

Մետաղադրամի պատկերներ		Օբյեկտների դասեր			
		Գլուխ5	Հակ.Կողմ 5	Գլուխ 20	Հակ.Կողմ 20
Հայտնաբերման լավագույն արդյունք (%)		100	100	100	100
Կեղծ աղմուկի գործակից (%)	Նեյրոնային ցանց	0	0	182	37.5
	ԳԾ	0	0	0	0

Ցանցաթաղանթ. Ցանցաթաղանթի պատկերների փորձերը բերված են Աղյուսակ 2.7-ում: Նախորդ փորձերի համեմատ այս դեպքում արդյունքները շատ ավելի վատ են, սակայն գենետիկ ծրագրավորման դեպքում կեղծ աղմուկի գործակիցը ավելի փոքր է:

Աղյուսակ 2.7. Ցանցաթաղանթի պատկերներում երկու դասերի օբյեկտների հայտնաբերման արդյունքների համեմատությունը:

Ցանցաթաղանթի պատկերներ		Օբյեկտների դասեր	
		Գամմա	Միկրո
Հայտնաբերման լավագույն արդյունք (%)		70	100
Կեղծ աղմուկի գործակից (%)	Նեյրոնային ցանց	2698	10104
	ԳԾ	1357	588

Արդյունքներից կարող ենք ենթադրել, որ գենետիկ ծրագրավորումը նույն հայտնաբերման գործակցի դեպքում ունի

շատ ավելի փոքր կեղծ աղմուկի գործակից քան նեյրոնային ցանցերի դեպքում:

2.3 Օբյեկտների ճանաչման խնդրում բաղադրյալ առանձնահատկությունների ուսուցում կոէվոլուցիոն գենետիկ ծարագրավորմամբ

Օբյեկտների ճանաչման հիմնական ուղություններից մեկն է պատկերներից առանձնահատկությունները առանձնացնելը: Առանձնահատկությունները պահում են պատկերների հատկությունները, որոնք տրվում են դասակարգողին ճանաչումը իրականացնելու համար [54, 55]: Բաղադրյալ առանձնահատկություններ ստանալու համար կոէվոլուցիոն գենետիկ ծարագրավորմամբ (ԿԳԾ) գեներացվում են բաղադրյալ օպերատորների զանգված, որի տարրերը բաղադրյալ օպերատորներ են: Բաղադրյալ օպերատորները ներկայացվում են ծառի տեսքով, որոնց ներքին հանգույցները տարրական օպերատորներ են, իսկ տերևները՝ տարրական առանձնահատկություններ: Սա տարրական առանձնահատկություններ համախբելու ձև է: Յուրաքանչյուր սերնդում բաղադրյալ ԿԳԾ-ը ստանալով բաղադրյալ օպերատորների զանգվածը համալրվում է նոր անդամով: Կիրառելով բաղադրյալ օպերատորների զանգվածը ստացվում է բաղադրյալ առանձնահատկությունների զանգվածը, որը տրվում է դասակարգողին ճանաչումն իրականացնելու համար:

2.3.1 Տերմինալների (տարրական առանձնահատկություններով պատկերներ) և Տարրական օպերատորների բազմություններ

Տերմինալների բազմություն: *կոէվոլուցիոն* գենետիկ ծարագրավորմամբ օբյեկտների ճանաչման խնդրում տերմինալների բազմությունը բաղկացած է 20 տարրական առանձնահատկություններով պատկերներից [30, 56]: Առաջին 10 հատը ստեղծվել են MIT Lincoln լաբորատորիայի կողմից [30, 56]: Նրանք իրենց

մեջ պարունակում են արհեստական արբանյակի նկարող սարքի բնութագրեր: Մյուս 10-ը ընդհանուր առանձնահատկություններ են և սովորաբար օգտագործվում են պատկերների մշակման խնդիրներում: (1) պատկերի ստանդարտ շեղումն է, (2) ֆրակտալային (fractal) չափ, (3) ամենապայծառ բաշխիչների (scatterers) լրացման ռանգի գործակցի հաշվարկն է, (4) բլոբի զանգված (blob mass), (5) բլոբի տրամագիծ, (6) բլոբի իներցիա (inertia), (7) բլոբի փիքսելների մաքսիմալ արժեքը, (8) բլոբի փիքսելների միջին արժեքը, (9) բլոբի կոնտրաստի պայծառությունը, (10) քանակ, բաշխիչների պրոկեցիաները, (11) հորիզոնական, (12) ուղղահայաց, (13) մաժոր անկյունագիծ, (14) մինոր անկյունագիծ, բաշխիչների (15) մաքսիմում, (16) մինիմում և (17) միջին հեռավորություններն են իրենց կենտրոնից, (18) մոմենտ μ20, (19) մոմենտ μ02 և (20) մոմենտ μ22:

Տարրական օպերատորների բազմություն: *Կոէվոլուցիոն* գենետիկ ծարագրավորմամբ օբյեկտների ճանաչման խնդրում տարրական օպերատորները որպես արգումենտ ընդունում են մեկ կամ երկու իրական թվեր և վերադարձնում են գործողության արդյունքը: Օգտագործվել են 12 տարրական օպերատորներ, որոնք բերված են Աղյուսակ 2.8-ում:

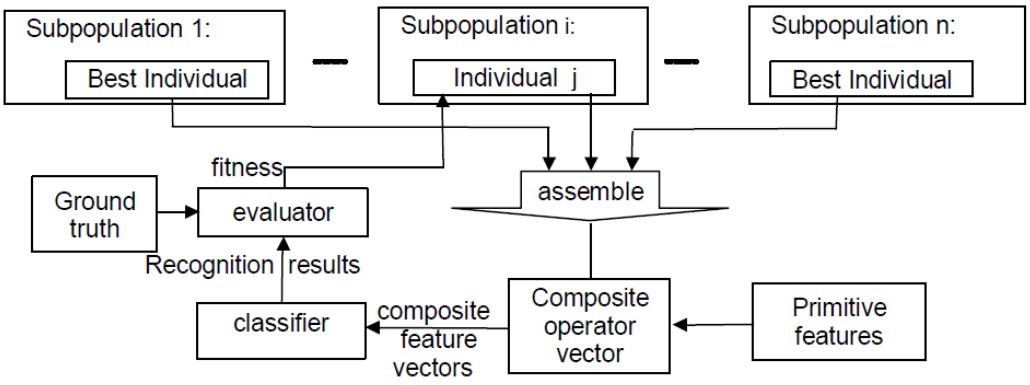
Աղյուսակ 2.8. Տասներկու տարրական օպերատորներ [30, 56]:

Primitive Operator	Description	Primitive Operator	Description
ADD (a, b)	Add a and b.	ADDC (a, c)	Add constant value c to a.
SUB (a, b)	Subtract b from a.	SUBC (a, c)	Subtract constant value c from a.
MUL (a, b)	Multiply a and b.	MUL (a, c)	Multiply a with constant value c.
DIV (a, b)	Divide a by b.	DIVC (a, c)	Divide a by constant value c.
MAX2 (a, b)	Get the larger of a and b.	MIN2 (a, b)	Get the smaller of a and b.
SQRT (a)	Return $\sqrt{a}$ if a ≥ 0; otherwise, return $-\sqrt{-a}$.	LOG (a)	Return log(a) if a ≥ 0; otherwise, return – log(-a).

2.3.2 Ֆիտնեսի հաշվումը

Բաղադրյալ օպերատորների զանգվածի ֆիտնեսը հաշվվում է հետևյալ կերպ՝ բաղադրյալ օպերատորների զանգվածը կիրառում է ուսուցման պատկերից ստացված տարրական առանձնահատկությունների վրա, որպեսզի ստացվի ուսուցման պատկերի բաղադրյալ առանձնահատկությունների զանգվածը և

զանգվածի տարրերը տրվում են բաեսյան դասակարգողին [56, 57]: Դասակարգողի ճանաչման գործակիցն էլ կլինի բաղադրյալ օպերատորների զանգվածի ֆիտնեսային արժեք:



Նկար. 2.15. Բաղադրյալ օպերատորների զանգվածի j-րդ դիրքի բաղադրյալ օպերատորի ֆիտնեսի հաշվումը [30, 56]:

Ցանկացած բաղադրյալ օպերատորի ֆիտնեսի [30, 56] հաշվման բլոկ սխեման բերված է Նկար 2.15-ում: Ենթասերունդների ընթացիք լավագույն բաղադրյալ օպերատորները համախմբվում են տվյալ բաղադրյալ օպերատորի հետ, կազմելով բաղադրյալ օպերատորների զանգվածը, որում i-րդ ենթասերնդի լավագույն բաղադրյալ օպերատորը գտնվում է i-րդ դիրքում: Բաղադրյալ օպերատորների զանգվածի ֆիտնեսն էլ կլինի տվյալ անհատի ֆիտնեսը: Այդ զանգվածի մյուս անհատները մնում են անփոփոխ: Ամեն սերնդում հաջորդաբար հաշվվում են անհատների ֆիտնեսները: Երբ սկզբնական ենթասերունդները գեներացվում են, նրանցում յուրաքանչյուր անհատի ֆիտնեսը հաշվվում է առանձին՝ առանց մյուս ենթասերունդներից վերցրած անհատների հետ համախմբելու: Յուրաքանչյուր սերունդ ստանալուց հետո առաջինը հաշվվում են առաջին ենթասերնդի անհատների ֆիտենսները, այնուհետև երկրորդ ու այսպես շարունակ: ԿԳԾ-ի աշխատանքի ավարտից հետո յուրաքանչյուր ենթասերնդից ընտրվում են լավագույն բաղադրյալ օպերատորները և ստացվում է բաղադրյալ օպերատորների զանգվածը:

2.3.3 Գենետիկ ծրագրավորման գործողությունները

Կոէվոլուցիոն գենետիկ ծարագրավորմը հաջորդ սերնդի բաղադրյալ օպերատորների զանգվածը ստանալու համար

իրականացնում է որոշ գործողություններ օգտագործելով տվյալ սերնդի բաղադրյալ օպերատորների բազմությունը: Այդ գործողություններն են՝ ընտրում, խաչասերում և մուտացիա:

Ընտրում: Ընտրում գործողությունը ընթացիք ենթասերնդից բաղադրյալ օպերատոր է ընտրում: Կիրառվել է մրցակցային ընտրում [28, 30]՝ որքան մեծ է բաղադրյալ օպերատորի ֆիտնեսային արժեքը այնքան մեծ է հավանականությունը, որ այն կընտրվի: Ընտրումը գործողությամբ ընտրվում են բաղադրյալ օպերատորներ գենետիկ ծրագրավորման հետագա գործողությունները իրականացնելու համար:

Խաչասերում: Երկու բաղադրյալ օպերատորներ, որոնք կոչվում են ծնողներ, ընտրվում են կախված իրենց ֆիտնեսային արժեքից: Այդ երկու ծնողներում (որոնք նեկայացվում են ծառի տեսքով) ընտրվում են մեկական հանգույց, և այդ հանգույցներից սկսվող ենթածառերը տեղերով փոխվում են ծնողների միջև: Տարբեր ենթասերունդների բաղադրյալ օպերատորների միջև խաչասերում չի իրականացվում [28,58, 30]:

Մուտացիա: Ենթասերունդում նմանօրինակությունից խուսափելու համար ենթասերնդի որոշ անահատներ ենթարկվում են փոփոխությունների: Բաղադրյալ օպերատորում պատահականորեն ընտրվում է հանգույց և այն փոխարնվում է մեկ ուրիշով կամ այդ հանգույցից սկսվող ենթածառը փոխարինվում է մեկ այլ պատահականորեն գեներացված ենթածառով, կամ բաղադրյալ օպերատորում ընտրվում են երկու տարբեր հանգույցներ և այդ հանգույցներից սկսվող ենթածառերը տեղերով փոխվում են [30]:

2.3.4 Կոէվոլուցիոն գենետիկ ծրագրավորման ալգորիթմի նկարագրությունը

Կոէվոլուցիան գենետիկ ծրագրավորման (ԿԳԾ) ալգորիթմը օգտագործվում է բաղադրյալ օպերատորների էվոլուցիայի համար [30, 56]: Այն հերթով իրականացնում է գենետիկ ծրագրավորման

գործողությունները՝ ընտրում, խաչասերում և մուտացիա: Սկզբում երկու բաղադրյալ օպերատորները ընտրվում են ըստ իրենց ֆիտնեսային արժեքի և խաչասերումից հետո ընթացիք սերնդում չեն մասնակցում հետագա խաչասերման գործողությանը: Այս գործողությունը շարունակվում է մինչև խաչասերման գործակցի բավարարումը: Այնուհետ կիրառվում է մուտացիայի գործողությունը ընթացիք ենթասրնդի անհատների վրա: Այնուհետև կիրառվում է ենթասերնդից բաղադրյալ օպերատորներ ընտրելու համար *ընտրում* գործողությունը, ընտրված բաղադրյալ օպերատորները համախմբվում են *խաչասերում* գործողությունից հետո ստացված անհատների հետ և կազմում են նոր ենթասերունդ (նոր ենթասերունդը պիտի ունենա նույն չափը, ինչ իր նախորդը): Վերջում կիրառվում է *ընտրում* գործողությունը, լավագույն բաղադրյալ օպերատորներն ընտրելու համար:

ԿԳԾ-ի աշխատանքի համար կիրառված հիմնական պարամետրերն են՝ ենթասերունդների քանակը N, սերնդի չափը M, սերունդների քանակը G, ֆիտնեսի շեմային արժեք, ԿԳԾ-ն ավարտում է աշխատանքը, երբ ֆիտնեսի արժեքը ստացվում է ավելի մեծ քան շեմային արժեքը:

Ստորև բերված են կոէվոլուցիոն գենետիկ ծրագրավորման ալգորիթմի փուլերը

1. Պատահականորեն գեներացնել N հատ M չափանի ենթասերունդ և առանձին առանձին գնահատել բոլոր ենթասերունդների բաղադրյալ օպերատորները (հաշվել ֆիտնեսները);
2. for gen = 1 to *generation_num* do
    3. for i =1 to N do
        4. Պահել լավագույն բաղադրյալ օպերատորը P_i ենթասերնդում
        5. Իրականացնել խաչասերում գործողությունը P_i ենթասերունդ բաղադրյալ օպերատորների վրա մինչ խաչասերման գործակցի բավարարումը, և պահել խաչասերումից ստացված բոլոր ժառանգները
        6. Իրականացնել մուտացիայի գործողությունը P_i ենթասերունդ բաղադրյալ օպերատորների վրա և

խաչասերումից ստացված ժառանգների վրա մուտացիայի գործակցի հավանականությամբ

7. Կիրառել ընտրում գործողությունը P_i-ի և խաչասերումից ստացված ժառանգների վրա ձևավորելով նոր ենթասերունդ P_i՝

8. Գնահատել յուրաքանչյուր բաղադրյալ օպերատոր C_j Pi՝-ում, C_j գնահատելու համար ընտրվում են ընթացիք մյուս ենթասերունդներից լավագույն բաղադրյալ օպերատորները և C_j-ի հետ կազմելով բաղադրյալ օպերատորների վեկտորը, ընդ որում, k-րդ ենթասերնդի բաղադրյալ օպերատորը զբաղեցնում է k-րդ դիրքը այդ զանգվածում: Բաղադրյալ օպերատորների վեկտորը կիրառել տարրական առանձնահատկությունների վրա արդյունքում կստացվի բաղադրյալ առանձնահատկությունների զանգվածը: Բաղադրյալ օպերատորների զանգվածը և բաղադրյալ առանձնահատկությունների զանգվածը տրվում են Բեզյան դասակարգողին: Ճանաչման գործակիցն էլ կհամարվի C_j-ի ֆիտնեսային արժեքը:

9. Pi-ի լավագույն բաղադրյալ օպերատորը փոխարինել Pi՝-ի՝ վատագույնով և իրականացնել $Pi = Pi$՝

10. Ձևավորել բաղադրյալ օպերատորների զանգվածը ենթասերունդների լավագույն բաղադրյալ օպերատորներով և եթե զանգվածի ֆիտնես արժեքը մեծ լինի շեմայինից, ապա

11. endfor// առաջին ցիկլի ավարտը

12. endfor // երկրորդ ցիկլի ավարտը

13. Արտածել բաղադրյալ օպերատորների զանգվածը:

2.3.5 Իրականացված փորձերն ու փորձերի արդյունքները

Հինգ տիպի օբյեկտների պատկերներ (BRDM2 զրահամեքենա, D7 տրակտոր, T62 տանկ, ZIL131 զինվորական բեռնատար և ZSU հակաօդային պաշտպանության մեքենա) են օգտագործվել փորձերում:

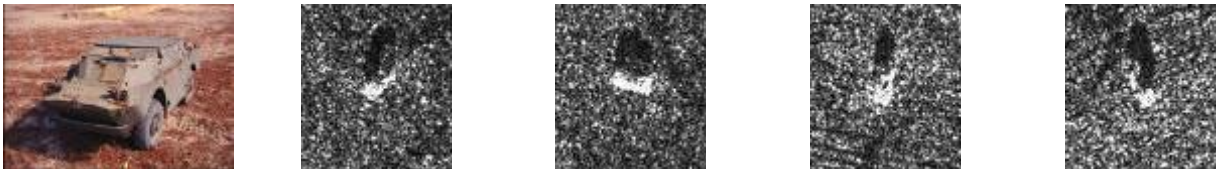
Յուրաքանչյուր օբյեկտի համար վերցվել են 15^0 անկման տակ տարբեր 0^0-359^0 անկյան տակ արված SAR պատկերներ տես նկար 2.16:

Քանի որ SAR պատկերները մեծապես կախված են նկարելու և ազիմուտային անկյուններից, հետևաբար, յուրաքանչյուր օբյեկտի համար պահվում է տարբեր ազիմուտ ունեցող նկարներ:

Իրականացվել են երկու տիպի փորձեր՝

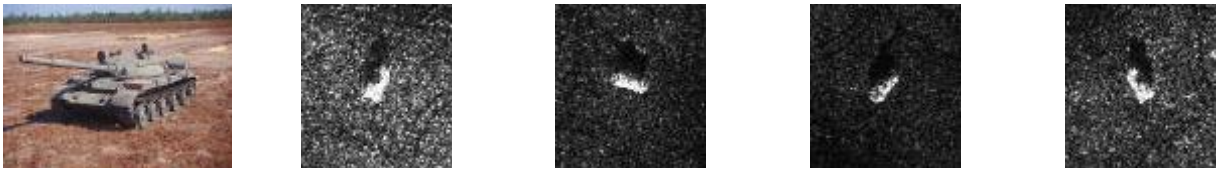
Փորձ 1. ԿԳՃ –ի միջոցով գեներացվել են բաղադրյալ օպերատորների զանգված երեք տիպի օբյեկտներ ճանաչելու համար (BRDM2 զրահամեքենա, D7 տրակտոր, T62 տանկ):

Երեք տիպի փորձեր են կատարվել, որոնցում ԿԳՃ-ն օգտագործել է 3, 5 ը 8 ենթաերուններ, որպեսզի զարգացվի համապատասխանաբար 3, 5 և 8 չափանի բաղադրյալ օպերատորների զանգված: Փորձերի մի մասում օգտագործվել են բոլոր տարրական առանձնահատկությունները, մյուս մասում միայն վերջին 10-ը: Փորձերի արդյունքները բերված են Աղյուսակ 2.9-ում, Աղյուսակ 2.10-ում և Աղյուսակ 2.11-ում:

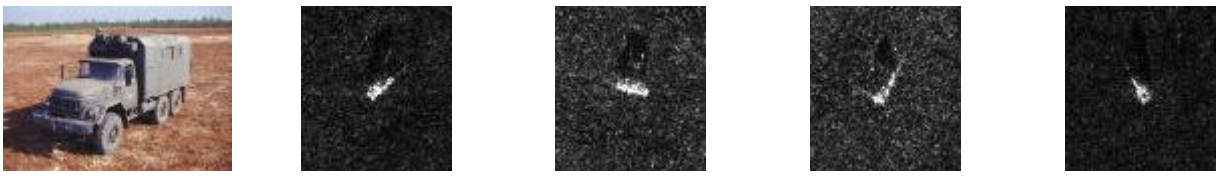


BRDM2 զրահամեքենայի օպտիկական և SAR պատկերներ

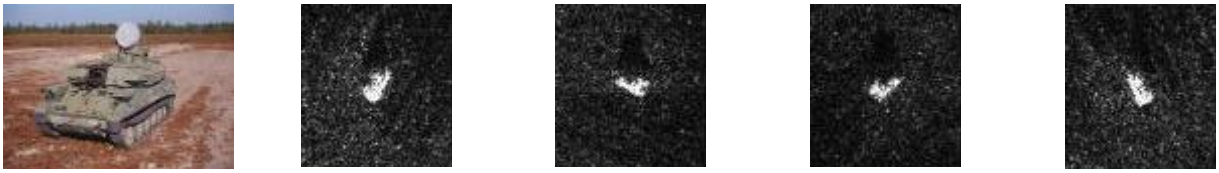
D7 տրակտորի օպտիկական և SAR պատկերներ



T62 տանկի օպտիկական և SAR պատկերներ



ZIL131 զինվորական բեռնատարի օպտիկական և SAR պատկերներ



ZSU հակաօդային պաշտպանության մեքենայի օպտիկական և SAR պատկերներ

Նկար 2.16. Ճանաչման մեջ օգտագործված հինգ ռազմական օբյեկտներ [30]:

Աղյուսակ 2.9. Երեք ենթատարածությունների դասերով փորձերի արդյունքները [30]:

Runs	Recognition Rate			
	10f		20f	
	Training	Testing	Training	Testing
1	0.895	0.836	0.981	0.967
2	0.886	0.829	0.967	0.948
3	0.867	0.848	0.952	0.921
4	0.910	0.831	0.957	0.945
5	0.857	0.843	0.962	0.926
6	0.910	0.843	0.967	0.936
7	0.852	0.862	0.981	0.938
8	0.871	0.838	0.976	0.936
9	0.876	0.860	0.967	0.952
10	0.871	0.843	0.981	0.956
Averag	0.880	0.843	0.969	0.943

Աղյուսակ 2.10. Հինգ ենթատարածությունների դասերով փորձերի արդյունքները [30]:

Runs	Recognition Rate			
	10f		20f	
	Training	Testing	Training	Testing
1	0.919	0.864	0.990	0.969
2	0.914	0.860	0.981	0.955
3	0.914	0.845	0.995	0.964
4	0.943	0.852	0.990	0.943
5	0.910	0.845	0.990	0.962
6	0.919	0.843	0.990	0.952
7	0.919	0.881	0.995	0.960
8	0.919	0.874	0.986	0.967
9	0.919	0.845	0.986	0.960
10	0.929	0.857	0.995	0.974
Averag	0.921	0.857	0.990	0.961

Աղյուսակ 2.11. Ութ ենթատարածությունների դասերով փորձերի արդյունքները [30]:

Runs	Recognition Rate			
	10f		20f	
	Training	Testing	Training	Testing
1	0.976	0.888	1.0	0.967

2	0.962	0.860	1.0	0.971
3	0.952	0.9	1.0	0.976
4	0.967	0.876	1.0	0.964
5	0.967	0.871	1.0	0.967
6	0.952	0.845	0.995	0.979
7	0.967	0.852	1.0	0.979
8	0.957	0.874	0.995	0.95
9	0.957	0.867	1.0	0.964
10	0.967	0.869	0.995	0.983
Averag	0.962	0.870	0.999	0.97

Արդյունքները թույլ են տալիս հասկանալու, որ երբ օգտագործվել են բոլոր տարրական առանձնահատկությունները ֆիտնեսային արժեքները ավելի մեծ են, ինչպես նաև 8 չափանի ենթասերունդների կիրառման դեպքում: Բոլոր առանձնահատկությունները օգտագործող 5 չափանի լավագույն բաղադրյալ օպերատորների զանգվածը բերված է Նկար 2.17-ում:

Փորձ 2. Շատ տիպի օբյեկտների դեպքում ճանաչումը ավելի դժվար է: Երկու տիպի փորձեր են իրականացվել 5 չափանի և 8 չափանի բաղադրյալ օպերատորներով, որոնցում կա՛մ 10 տարրական օպերատոր է օգտագործվել, կա՛մ բոլորը: Բոլոր առանձնահատկությունները օգտագործող 8 չափանի լավագույն բաղադրյալ օպերատորների զանգվածը բերված է Նկար 2.18-ում: Արդյունքները բերված են Աղյուսակ 2.12-ում և Աղյուսակ 2.13-ում: Արդյունքներից երևում է, որ ճանաչման ֆիտնեսային արժեքները ավելի փոքր են, քան 3 տիպի օբյեկտների դեպքում էր:

Աղյուսակ 2.12. Հինգ ենթասերունդներով փորձերի արդյունքները [30]:

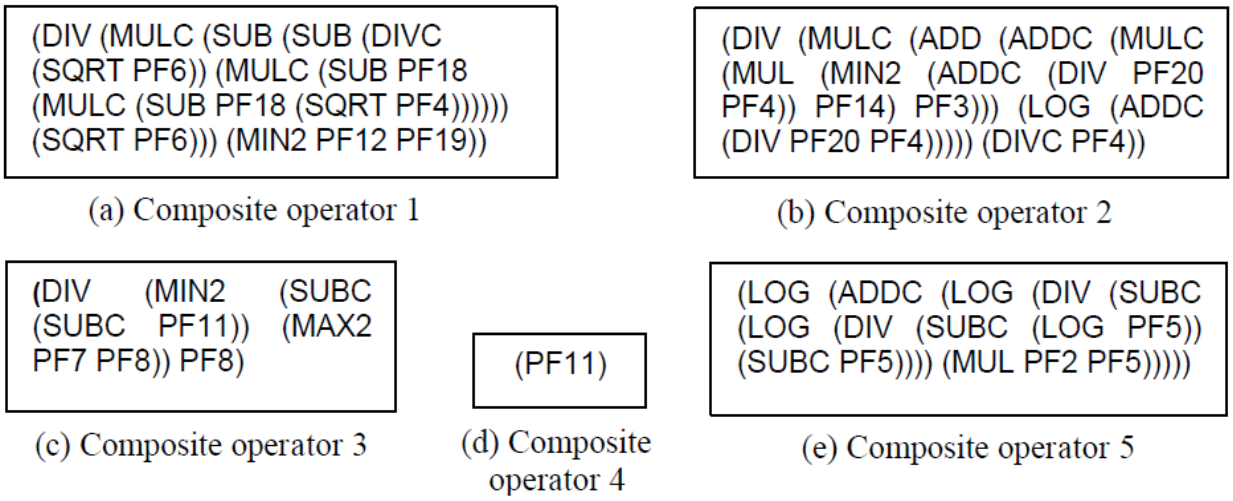
Runs	Recognition Rate			
	10f		20f	
	Training	Testing	Training	Testing
1	0.691	0.594	0.84	0.727
2	0.674	0.581	0.831	0.741
3	0.706	0.627	0.866	0.75

Աղյուսակ 2.13. Ութ ենթասերունդներով փորձերի արդյունքները [30]:

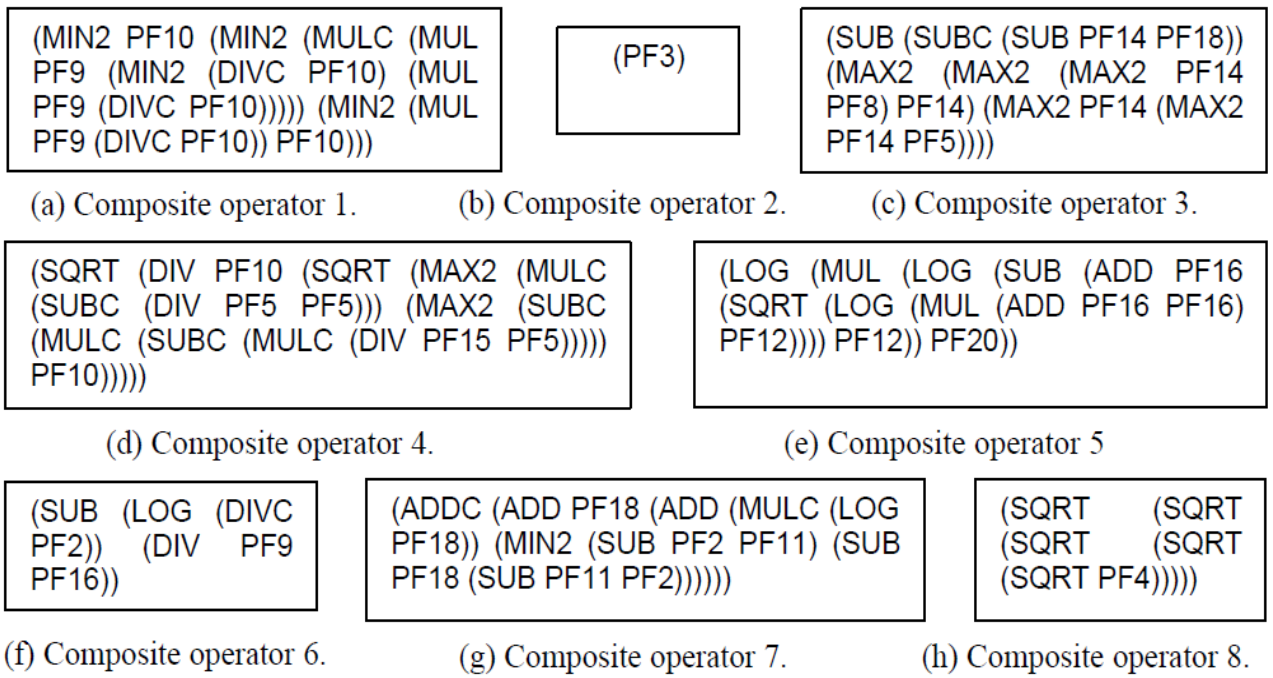
Runs	Recognition Rate			
	10f		20f	
	Training	Testing	Training	Testing
1	0.769	0.621	0.929	0.824
2	0.791	0.63	0.926	0.816
3	0.794	0.63	0.929	0.803

4	0.697	0.594	0.846	0.756
5	0.666	0.563	0.869	0.777
6	0.671	0.549	0.863	0.787
7	0.654	0.546	0.829	0.747
8	0.683	0.577	0.837	0.753
9	0.691	0.573	0.874	0.794
10	0.674	0.587	0.88	0.803
Averag	0.681	0.579	0.854	0.764

4	0.791	0.624	0.941	0.845
5	0.774	0.614	0.931	0.809
6	0.774	0.647	0.911	0.809
7	0.794	0.64	0.903	0.832
8	0.774	0.623	0.943	0.842
9	0.754	0.641	0.94	0.847
10	0.763	0.631	0.909	0.823
Averag	0.778	0.630	0.926	0.825



Նկար 2.17. Հինգ չափանի բաղադրյալ օպերատորի զանգվածի օրինակ [30, 56]:



Նկար 2.18. Ութ չափանի բաղադրյալ օպերատորի զանգվածի օրինակ [30, 56]:

1.4 Հիշատակված աշխատանքներում նկատված թերությունները

1. **Առաջին աշխատանքում նկատված թերությունները**: Այս մոտեցման դեպքում հնարավոր չի պարզել ճանաչվող օբյեկտի տիպը, ինչպես նաև նաև օբյեկտների չափերը նախապես հայտնի են:

2. **Երկրորդ աշխատանքում նկատված թերությունները**: Քանի որ այս մոտեցում չի դիտարկում հայտաբերված պատկերի ձևը, հետևաբար այն նույնատիպ ինտեսիվություն ունեցող տարբեր օբյեկտներ կդասակարգի որպես նույն դասին պատկանող: Ինչպես նաև տվյալ աշխատանքում կատարված փորձերի արդյունքները թույլ են տալիս պնդել, որ բարդ պատկերների ճանաչումը կատարվում է բավականին մեծ սխալանքով: Հետևաբար այս մոտեցումը արյունավետ չի բարդ պատկերներում օբյեկտներ ճանաչելու համար:

3. **Երրորդ աշխատանքում նկատված թերությունները**: Այս աշխատանքում որոշ տարրական պատկերներ գեներացնելու համար օգտագործվել են նկարող սարքերի բնութագրերը: Դրանից հետևում է, որ համակարգը աշխատում է միայն SAR պատկերների համար, որոնցում օբյեկտներին պատկանող փիքսելները ավելի պայծառ են: Որպես գենետիկ ծրագրավորման ալգորիթմ օգտագործվել է Կոէվոլուցիոն գենետիկ ծրագրավորման ալգորիթմը, որը բավականին բարդ ալգորիթմ է: Բաղադրյալ օպերատորների զանգվածը կիրառվում է մուտքին տրվող պատկերի վրա, ինչը

ենթադրում է ճանաչման ավելի մեծ ժամանակ, քանի որ զանգվածում գտնվող յուրաքանչյուր բաղադրյալ օպերատոր կիրառվում է ճանաչվելիք պատկերի վրա: Փորձերի արդյունքներից երևում է, որ **կախված ճանաչվող օբեկտների տիպերի քանակից, ճանաչման սխալանքը աճում է**:

ԳԼՈՒԽ 3

ԹՎԱՅԻՆ ՊԱՏԿԵՐՆԵՐՈՒՄ ՕԲՅԵԿՏՆԵՐԻ ՀԱՅՏՆԱԲԵՐՄԱՆ ԵՎ ՃԱՆԱՉՄԱՆ ԾՐԱԳՐԱՅԻՆ ՀԱՄԱԿԱՐԳԻ ՄՇԱԿՈՒՄԸ

Այս գլխում նկարագրված են գենետիկ ծրագրավորմամբ թվային պատկերներում օբյեկտներ հայտնաբերող և ճանաչող ծրագրային համակարգի մշակման փուլերը:

3.1 Օբյեկտների հայտնաբերում և ճանաչում

Օբյեկտների ճանաչումը իրականացվում է երկու փուլով՝ օբյեկտների հայտնաբերում և ճանաչում: Օբյեկտների հայտնաբերման փուլի հիմնական խնդիրն է գտնել և առանձնացնել պատկերի այն հատվածը, որն հավանական է, որ ընդգրկում է հետաքրքրության օբյեկտը: Սա կարևոր քայլ է օբյեկտների ճանաչման խնդրում: Առանձնացված տարածքը կոչվում է հետաքրքրությունների տիրույթ (ՀՏ - ROI) [20, 30, 6]: Այս քայլը կարևոր է օբյեկտների ճանաչման համար մի շարք պատճառներով:

Առանձնացնելով ՀՏ-ն մենք նվազեցնում ենք հաշվարկների ժամանակը, քանի որ պատկերի չափերը ավելի մեծ են քան այն տիրույթները, որում գտնվում է օբյեկտը:

Առանձնացնելով ՀՏ-ն ճանաչող համակարգը հնարավորություն է ստանում կենտրոնանալ այդ տարածքի վրա, ինչը կարող է դրական անդրադառնալ օբյեկտների ճանաչման ճշգրտության վրա:

Օբյեկտների ճանաչման փուլի խնդիրն է հետազոտել ՀՏ-ն և պարզել ինչ օբյեկտ է պարունակում այն:

Որպեսզի կարողանանք իրականացնել հայտնաբերում և ճանաչում պետք է կարողանանք պատկերից, որում գտնվող օբյեկտը պետք է ճանաչենք, տարանջատել որոշ առանձնահատկություններ (սկզբնական պատկերից ստացված պատկերներ, որոնք ստացվում են սկզբնական պատկերի վրա որոշակի ֆիլտրեր կիրառելուց): Այս առանձնահատկությունները կոչվում են տարրական առանձնահատկություններ: Հայտնաբերումն ու ճանաչումը մեծապես կախված են այն առանձնահատկություններից, որոնք կկարողանանք տարանջատել: Հետևաբար, չենք կարող հասնել ճանաչման և հայտնաբերման լավ արդյունքների (անկախ նրանից թե, ինչ ալգորիթմ է օգտագործված), առանց համապատասխան առանձնահատկություններ ստանալու: Իրական պատկերից առանձնահատկություններ ստանալը հեշտ չէ, քանի որ իրական պատկերում աղմուկները բազմազան են, ինչպես նաև առանձնահատկություններն ու դրանց տիպերն բազմազան, հետևաբար, առանձնահատկությունները տարանջատելու և դրանցից համապատասխանները ընտրելու համար հարկավոր է հաշվարկային մեծ քանակի գործողություններ: Սովորաբար որպեսզի կարողանանք իրականացնել ճանաչումն ու հայտնաբերումը մեկ տարրական առանձնահատկությունը բավարար չի, մենք պետք է կարողանանք առանձնացնել առանձնահատկությունների խումբ, որի միջոցով էլ կիրականացվի ճանաչումն ու հայտնաբերումը: Առանձնահատկությունների այդ խումբն էլ կկոչենք բաղադրյալ առանձնահատկություններ: Ավանդաբար բաղադրյալ առանձնահատկությունները սինթեզվել են մասնագետների կողմից, ովքեր հիմնվելով իրենց փորձի, գիտելիքների, ինտուիցիայի և մասնագետին բնորոշ այլ որակների վրա կարողացել են այդ խնդիրը լուծել: Քանի որ, ինչպես արդեն նշեցինք, այդ

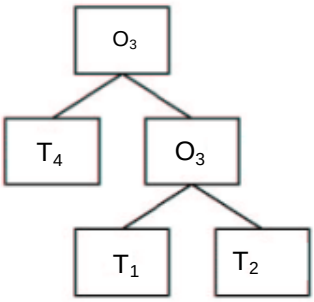
առանձնահատկությունները շատ բազմազան են, հետևաբար մարդուց շատ երկար ժամանակ կպահանջվի օբյեկտի ճանաչման համար պիտանի բաղադրյալ առանձնահատկություն սինթեզելը: Ակնհայտ է, որ մարդու համար շատ դժվար է (որոշ դեպքերում անհնար է) առանձնահատկությունների բազմության միջից ընտրել որևէ ենթաբազմություն, որը արդյունավետ կլինի տվյալ օբյեկտի հայտնաբերման և ճանաչման համար: Հենց այս խնդրի լուծման համար էլ օգտագործում ենք գենետիկ ծրագրավորումը (ԳԾ), որպեսզի կարողանանք առանձնահատկությունների բազմությունից ընտրել որևէ ենթաբազմություն (բաղադրյալ առանձնահատկություն), որը արդյունավետ կկիռառվի տվյալ օբյեկտի հայտնաբերման և ճանաչման համար:

3.2 Գենետիկ ծրագրավորումը օբյեկտների հայտնաբերման և ճանաչման խնդրում

Ինչպես արդեն նշել ենք օբյեկտների հայտնաբերման և ճանաչման համար անհրաժեշտ է պատկերից առանձնացնել տարրական առանձնահատկություններ, և դրանց բազմությունից ընտրել որևէ ենթաբազմություն, որը էֆեկտիվ կլինի այդ օբյեկտը ճանաչելու համար: Եվ քանի որ տարրական առանձնահատկությունները բազմազան են ու նրանցով կազմված կոմբինացիաները (բաղադրյալ առանձնահատկություններ) գրեթե անսահմանափակ են, ապա այդ բազմությունից ցանկալի ենթաբազմություն ընտրելը շատ ժամանակատար է, և փորձագետը կարող է փորձել շատ սահմանափակ քանակի կոմբինացիաներ: Հենց այդ նպատակով էլ օգտագործում ենք գենետիկ ծրագրավորումը, քանի որ այն հնարավորություն է տալիս ոչ բոլոր հնարավոր կոմբինացիաները դիտարկելով գտնել որևէ ենթաբազմություն, որը կբավարարի նախապես սահմանված պայմաններին: Բացի այդ ԳԾ-ի զուգահեռությունը և ժամանակակից քոմփյութերների արագությունները հնարավորություն են տալիս շատ ավելի մեծ բազմությունում փնտրել, քան մասնագետի կողմից դիտարկվող բազմությունը: Այսինքն՝ ԳԾ-ն պատկերների մեջ

օբյեկտներ հայտնաբերելու և ճանաչելու խնդրում կիրառում ենք բաղադրյալ առանձնահատկություններ սինթեզելու համար: ԳԾ-ի իրականացրած փնտրումը պատահական փնտրում չի, այն պատահականորեն ստեղծում է սկզբնական սերունդ հետո սերնդե-սերունդ զարգացնելով սերնդի ներկայացուցիչներին (ԳԾ-ի անհատներին) որևէ սերնդում ստանում է նախապես սահմանված պահանջներին բավարարող անհատ՝ բաղադրյալ օպերատոր կամ բաղադրյալ օպերատորների խումբ:

Օբյեկտների հայտնաբերման և ճանաչման խնդրում ԳԾ-ի կիրառելիս նրա անհատներ են հանդիսանում բաղադրյալ օպերատորները, որոնք իրենց հերթին հանդիսանում են պատկերի բաղադրյալ առանձնահատկություններ: Բաղադրյալ օպերատորները՝ անհատները կազմված են տարրական օպերատորներից և տարրական պատկերներից (տարրական առանձնահատկություններ): Բաղադրյալ օպերատորները կարող են ներկայացվել ծառի տեսքով (տես՝ Նկար 3.1), O-ով նշված հանգույցները՝ օպերատորներ են, իսկ T-ով նշվածները՝ տերևները, տարրական պատկերներ (գենետիկ ծրագրավորման թեմփլեյթներ):



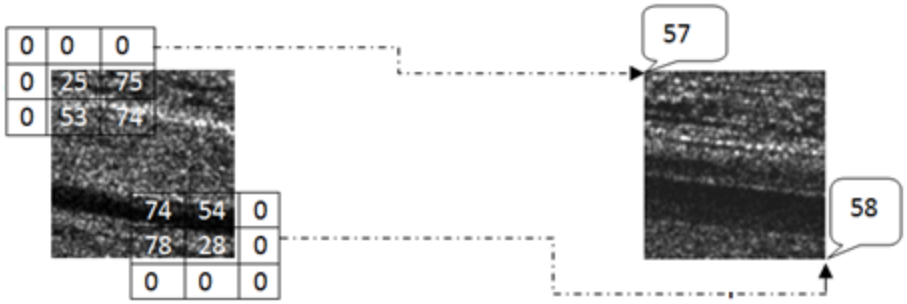
Նկար 3.1 Բաղադրյալ օպերատոր ներկայացված ծառի տեսքով:

ԳԾ-ի ալգորիթմի աշխատանքը կախված չի տարրական օպերատորների և տարրական պատկերների տեսակներից [28, 30]: Այսինքն՝ մենք կարող ենք ընտրել կամայական տարրական օպերատորներ և տարրական պատկերներ:

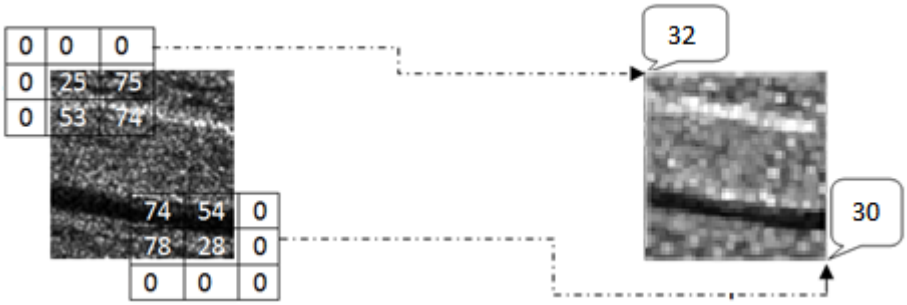
3.3 Տարրական պատկերներ և օպերատորներ

Տարրական պատկերներ: Աշխատանքում օգտագործված տարրական պատկերների բազմությունը բաղկացած է տարրական առանձնահատկություններ ունեցող տասնվեց պատկերներից, որոնցից առաջինը սկզբնական (օրիգինալ) պատկերն է, մյուսները՝ միջին, շեղված, մաքսիմում, մինիմում և մեդիան պատկերներն են (տես՝ Աղյուսակ 3.1.) [30], որոնք ստացվում են սկզբնական պատկերի վրա որոշակի գործողություններ կիրառելով: Միջին պատկերի ստացման նկարագրությունը բերված է ստորև:

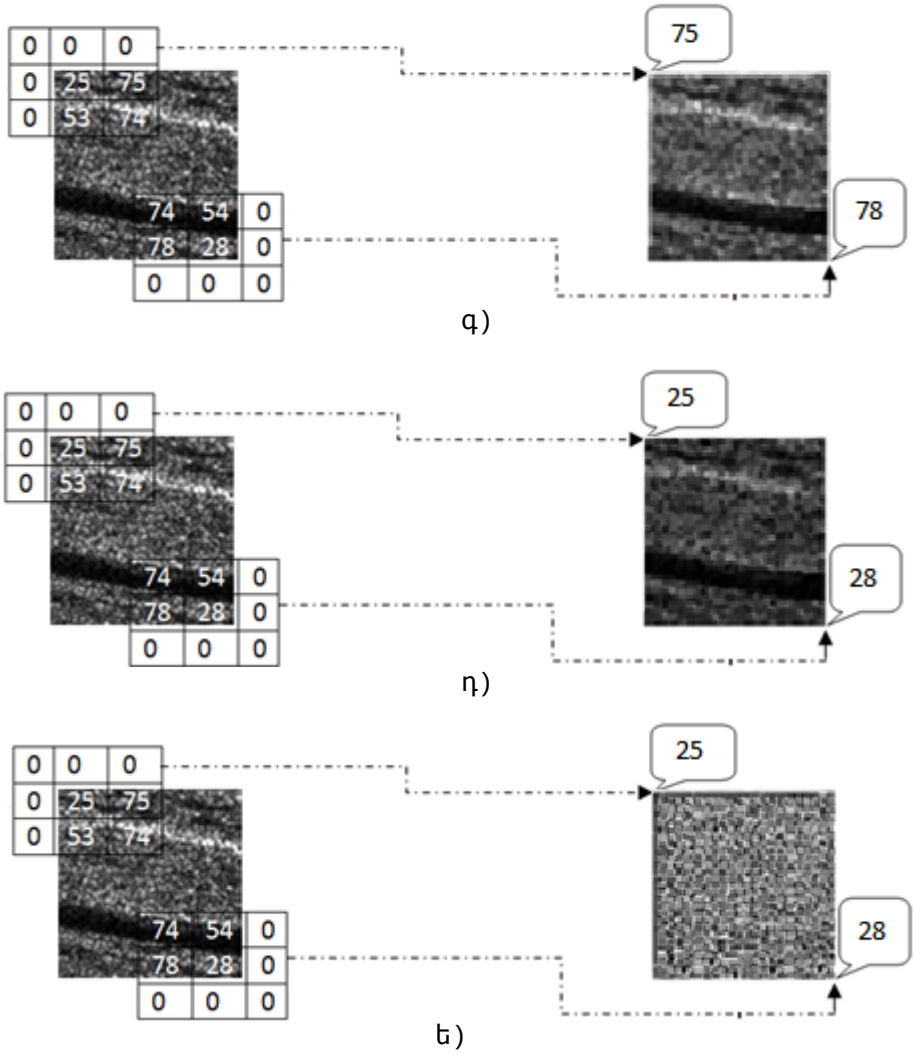
Սկզբնական պատկերի մատրիցայի վրայով անցկացվում է $3x3$ չափանի պատուհան, հետո հաշվվում են պատկերի այն մասի փիքսելների միջին թվաբանական արժեքը, որը ծածկված է այդ պատուհանով և արդյունքը տեղադրում ենք նոր գեներացվող (միջին $3x3$) պատկերի այն դիրքում, որտեղ համապատասխանաբար գտնվում էր սկզբնական պատկերի պատուհանով ծածկված տարածքի մեջտեղի կետը (տես՝ Նկար 3.2): Միջին $5x5$ և 7x7 պատկերների ստացման համար պատուհանի չափերը համապատասխանաբար ընտրվում են $5x5$ և $7x7$:



ա)



բ)



Նկար 3.2. Միջին $3x3$ պատկերի ստացման սխեման: Ձախում սկզբնական պատկերն է, աջում՝ միջին $3x3$ պատկերը:

Շեղված, մաքսիմում, մինիմում և մեդիան պատկերները ստացվում են նույն սկզբունքով, տարբերությունը կայանում է նրանում, որ այս դեպքում նոր գեներացվող պատկերի փիքսելի արժեքը ոչ թե պատուհանի միջին արժեքն է, այլ շեղված պատկերի դեպքում այդ արժեքը սկզբնական պատկերի և միջին պատկերի համապատասխան փիքսելների արժեքների տարբերությունն է (տես՝ նկար 3.2 բ-ն), մաքսիմում պատկերի դեպքում պատուհանի ընգրկած փիքսելների մաքսիմումն է (տես՝ նկար 3.2 գ-ն), մինիմումի դեպքում՝ մինիմումն է (տես՝ նկար 3.2 դ-ն), մեդիան ֆիլտրի դեպքում պատուհանի ընդգրկած փիքսելների մեդիանային արժեքը (տես՝ նկար 3.2 ե-ն): Մեր օգտագործած տարրական պատկերների ամբողջական ցուցակը ներկայացված է Աղյուսակ 3.1-ում:

Աղյուսակ 3.1. Աշխատանքում օգտագործված 16 տարրական պատկերները:

No.	Տարրական առանձնահատ. պատկերներ	Մեկնաբանություն	No.	Տարրական առանձնահատ. պատկերներ	բացատրություն
0	PFIM0	Սկզբնական պատկեր (Նկ. 3.2ա)	8	PFIM8	$5x5$ մաքսիմում պատկեր
1	PFIM 1	$3x3$ միջին պատկեր	9	PFIM9	$7x7$ մաքսիմում պատկեր
2	PFIM2	$5x5$ միջին պատկեր	10	PFIM10	$3x3$ մինիմում պատկեր
3	PFIM3	$7x7$ միջին պատկեր	12	PFIM11	$5x5$ մինիմում պատկեր
4	PFIM4	$3x3$ շեղված պատկեր	13	PFIM12	$7x7$ մինիմում պատկեր
5	PFIM5	$5x5$ շեղված պատկեր	14	PFIM13	$3x3$ մեդիան պատկեր
6	PFIM6	$7x7$ շեղված պատկեր	15	PFIM14	$5x5$ մեդիան պատկեր
7	PFIM7	$3x3$ մաքսիմում պատկեր	16	PFIM15	$7x7$ մեդիան պատկեր

Տարրական օպերատորներ. Տարրական օպերատորները ֆունկցիաներ են, որոնք որպես արգումենտ ստանում են մեկ կամ երկու պատկերներ, իրականացնում են որոշակի գործողություններ այդ պատկերների վրա և արդյունքը պահում են ելքային պատկերում: Այս աշխատանքում մենք օգտագործել ենք 17 տարրական օպերատորներ (տես՝ Աղյուսակ 3.2) [30, 56], որտեղ A և B-ն որոշակի չափ ունեցող մուտքային պատկերներ են: C-ն ամբողջ թիվ է, որը կարող է ընդունել [-20 ... 20] արժեքներ: $MAX(A)$, $MIN(A)$, $MED(A)$, $MEAN(A)$ և $STDV(A)$ տարրական օպերատորների $3x3$, $5x5$ և $7x7$ *հարևանները* օգտագործվում են հավասար հավանականությամբ, այսինքն՝ նրանցից ոչ մեկին առավելություն վերապահված չի:

Աղյուսակ 3.2. 17 տարրական օպերատորները:

Հ.հ.	Օպերատոր	Նկարագրությունը

1	$ADD(A,B)$	A և B տարրական պատկերների գումարը
2	$SUB(A,B)$	A և B պատկերների տարբերությունը
3	$MUL(A,B)$	A և B պատկերների արտադրյալը
4	$DIV(A,B)$	A և B պատկերների հարաբերությունը (եթե B-ն ունի 0 արժեքով փիքսել, այն փոխարինվում է A-ի մաքսիմալ փիքսելով).
5	$MAX2(A,B)$	Նոր պատկերի փիքսելին վերագրվում է A և B-ի համապատասխան փիքսելների մեծագույն արժեքը
6	$MIN2(A,B)$	Նոր պատկերի փիքսելին վերագրվում է A և B-ի համապատասխան փիքսելների նվազագույն արժեքը
7	$ADDC(A)$	Յուրաքանչյուր փիքսելին գումարվում է c թիվը
8	$SUBC(A)$	Յուրաքանչյուր փիքսելից հանվում է c թիվը
9	$MULC(A)$	Յուրաքանչյուր փիքսել բազմապատկվում է c թվով
10	$DIVC(A)$	Յուրաքանչյուր փիքսել բաժանվում է c թվի վրա
11	$SQRT(A)$	$v \geq 0$ արժեքով ցանկացած փիքսել փոխարինվում է $\sqrt{v}$ -ով, հակառակ դեպքում՝ $-\sqrt{-v}$-ով
12	$LOG(A)$	Եթե փիքսելի արժեքը 0 է, այն թողնվում է անփոփոխ արժեքով փիքսելը փոխարինվում է $\ln(v)$ -ով՝ հակառակ դեպքում՝ $-\ln(-v)$
13	$MAX(A)$	Պիքսելի արժեքը փոխարինվում է 3x3, 5x5 կամ 7x7 հարևանների մաքսիմալ փիքսելի արժեքով
14	$MIN(A)$	Պիքսելի արժեքը փոխարինվում է 3x3, 5x5 կամ 7x7 հարևանների մինիմալ փիքսելի արժեքով
15	$MED(A)$	Փիքսելի արժեքը փոխարինվում է 3x3, 5x5 կամ 7x7 հարևանների մեդիան փիքսելի արժեքով
16	$MEAN(A)$	Փիքսելի արժեքը փոխարինվում է 3x3, 5x5 կամ 7x7 հարևանների միջին փիքսելի արժեքով
17	$STDV(A)$	Փիքսելի արժեքը փոխարինվում է 3x3, 5x5 կամ 7x7 հարևանների միջին քառակուսային արժեքով

3.4. Համընկման ֆունկցիան և գենետիկ ծրագրավորման պարամետրերը

Համընկման ֆունկցիա (ֆիտնես) ԳԾ-ի անհատները (բաղադրյալ օպերատորները) գնահատելու համար, թե որքանով է այն համապատասխանում նախապես սահմանված պահանջներին, օգտագործվում է համընկման (fitness) ֆունկցիան: Բաղադրյալ օպերատորը գնահատելու համար օգտագործում ենք հետևյալ տեսքի fitness ֆունկցիան՝

$$fitness(G,G') = \frac{n(G \cap G`)}{n(G \cup G`)}, \quad (3.1)$$

որտեղ n(X)-ը X պատկերի փիքսելների քանակն է, G-ն սկզբնական պատկերի վրա բաղադրյալ օպերատորի կիրառման արդյունքն է (բաղադրյալ առանձնահատկություն-ներով պատկեր), G'-ը օբյեկտի սպասվող Ground Truth [30] պատկերն է, որը ուսուցման փուլում ստացվում է “ձեռքով”, $n(G \cap G`)$-ը երկու պատկերների համադրման արդյունքում համընկնող փիքսելների քանակն է, $n(G \cup G`)$-ը պատկերների ընդհանուր փիքսելների քանակն է:

ԳԾ-ի պարամետրեր. ԳԾ-ի աշխատանքը և նրա ավարտը կառավարելու համար օգտագործվում են որոշակի մուտքային պարամետրեր, որոնք բերված են ստորև՝

- *Սերնդում անհատների քանակը*՝ M, դա յուրաքանչյուր սերնդում բաղադրյալ օպերատորների քանակն է: Գենետիկ ծրագրավորման աշխատանքի սկզբում պատահականորեն գեներացվում են M բաղադրյալ օպերատորներ: Յուրաքանչյուր հաջորդ սերնդում այդ քանակը պահպանվում է:
- *Սերունդների քանակը*՝ *N*: Գենետիկ ծրագրավորման ալգորիթմի աշխատանքի արդյունքում գեներացված սերունդների քանակը չի գերազանցում N-ը:

- *Անհատի մաքսիմալ խորությունը*: Անհատների համար նախապես սահմանվում է խորության մաքսիմում արժեք, որը չպետք է գերազանցվի սկզբնական անհատների գեներացման, խաչասերման կամ մուտացիայի ժամանակ:
- *Խաչասերման գործակիցը* ցույց է տալիս, թե սերնդում գտնվող բաղադրյալ օպերատորների որ մասի վրա պետք է իրականացվի խաչասերում գործողությունը:
- *Մուտացիայի գործակիցը* ցույց է տալիս, թե սերնդում գտնվող բաղադրյալ օպերատորների որ մասի վրա պետք է իրականացվի մուտացիագործողությունը,
- *Ֆիտնեսի շեմային արժեքը* անահատների այն նպատակային ֆիտնես արժեքն է, որին փորձում ենք հասնել: Եթե սերնդում լավագույն անդամի (ամենամեծ ֆիտնեսային արժեք ունեցող բաղադրյալ օպերատորը) ֆիտնեսը մեծ է շեմային արժեքից փնտրումը ավարտվում է:

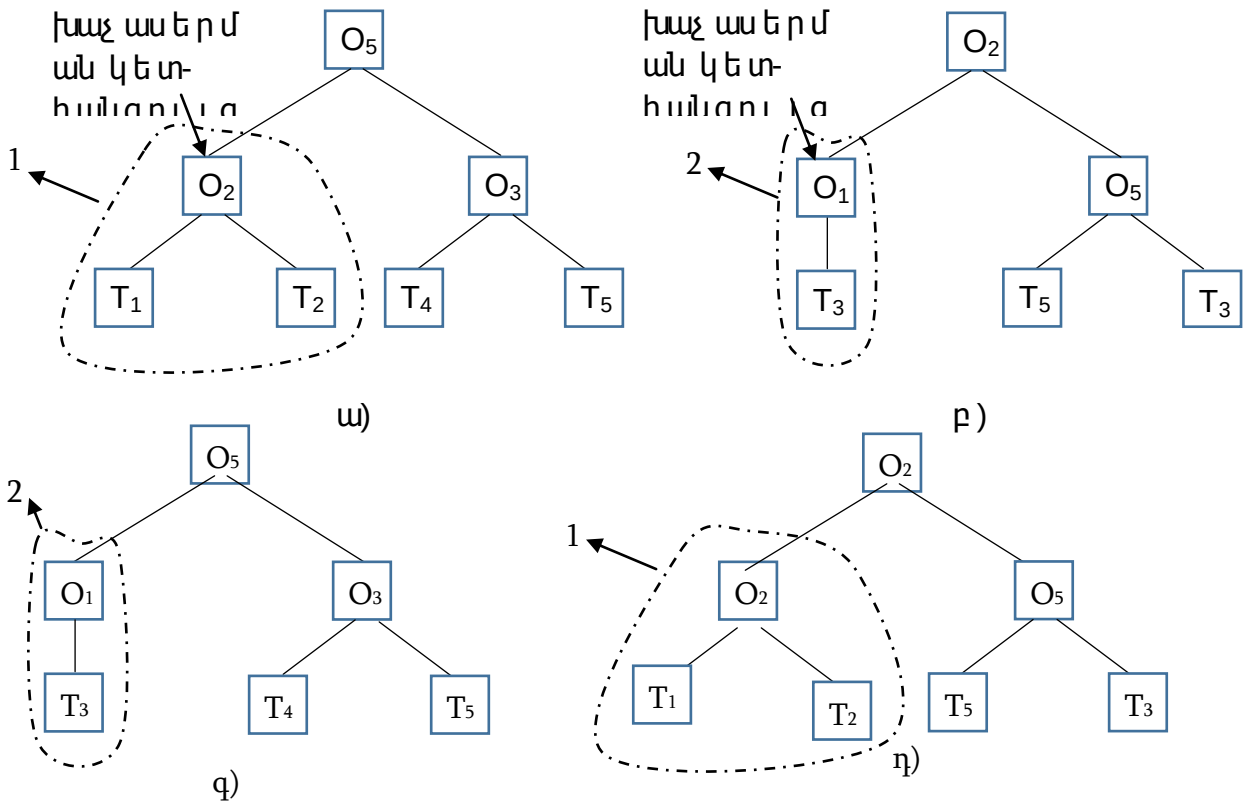
3.5 Աշխատանքում կիրառվող գենետիկ ծրագրավորման գործողությունները

Ինչպես արդեն նշել ենք, գենետիկ ծրագրավորումն իրականացնելու համար կիրառվում են երեք տիպի գործողություններ՝ ընտրում, խաչասերում և մուտացիա: Այս գործողությունների կիրառությունների նկարագրությունները բերված են ստորև:

Ընտրումը կիրառում ենք սերնդից անհատ (բաղադրյալ օպերատոր) ընտրելու համար: Այն կատարվում է կախված անհատի fitness-ային արժեքից, այսինքն՝ իրականացվում է մրցակցային ընտրման ալգորիթմը (տես՝ Գլուխ 1 էջ 13)՝ որքան մեծ է անհատի fitness-ը, այնքան հավանական է, որ այն կընտրվի:

Խաչասերում. Որպեսզի իրականացնենք խաչասերում պետք է սերնդից (բաղադրյալ օպերատորների բազմությունից), ընտրել երկու անհատ (բաղադրյալ օպերատորներ), որոնք կընտրվեն կախված

իրենց fitness-ի արժեքից, այսինքն, ընտրելու համար օգտագործվում է մրցակցային ընտրումը: Ընտրված այս երկու բաղադրյալ օպերատորները կոչվում են ծնողներ: Ընտրված բաղադրյալ օպերատորներից յուրաքանչյուրում պատահականորեն ընտրվում են ներքին հանգույցներ (խաչասերման կետեր) և ծնողների մեջև տեղերով փոխվում են այն ենթածառերը, որոնք սկսվում են ընտրված ենթահանգույցներից (տես՝ Նկար 3.3-ի ա-ն և բ-ն):



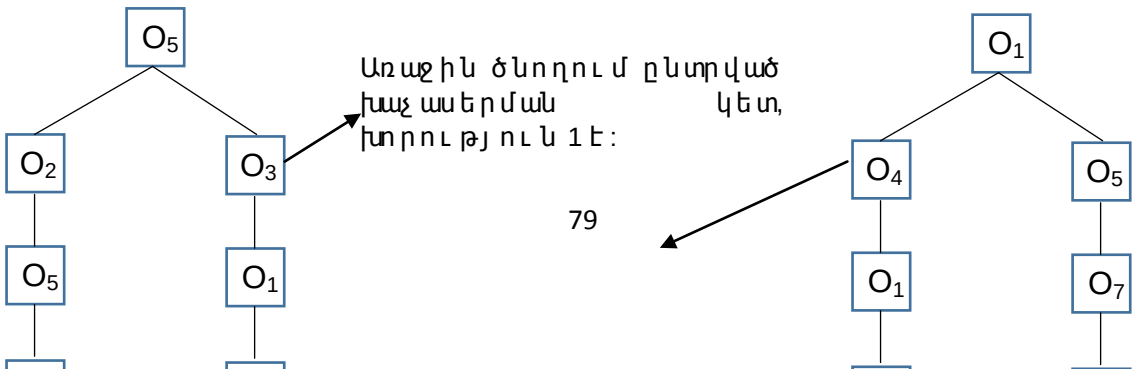
Նկար 3.3. Խաչասերում գործողություն: **ա)** և **բ)** խաչասերման համար ընտրված անհատներ, իսկ **գ)** և **դ)** նրանց ժառանգները:

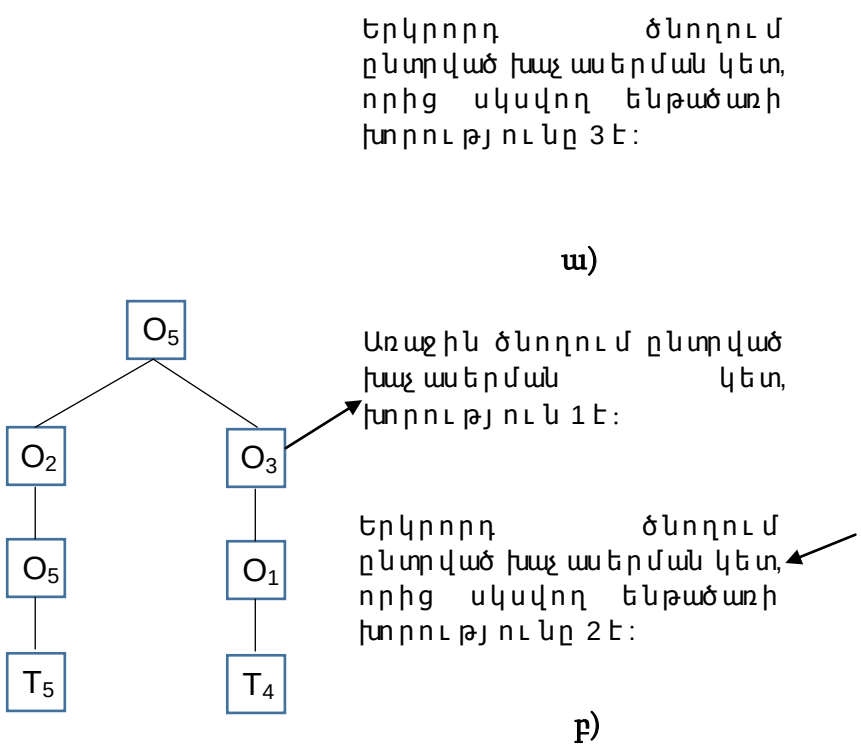
Արդյունքում ստացվում է երկու նոր բաղադրյալ օպերատոր, որոնք կոչվում են ժառանգներ (տես՝ Նկար 3.3-ի գ-ն և դ-ն): Այս երկու բաղադրյալ օպերատորները կարող են ավելի արդյունավետ լինել, քան նրանց ծնողները, քանի որ երկու արդյունավետ բաղադրյալ օպերատորների պատահական ընտրված ենթածառեր միավորվում են, ինչի արդյունքում կարող է ստացվել առավել արդյունավետ բաղադրյալ օպերատոր: Ակնհայտ է, որ ժառանգների չափերը կարող են ավելի մեծ լինել քան ծնողների չափերը: (եթե անհատը (բաղադրյալ օպերատորը) ներկայացնենք ծառի տեսքով (օրինակ,

Նկար 3.3 գ-ն), ապա այդ ծառի հանգույցների քանակը կհամարվի տվյալ անհատի չափը): Այսինքն՝ խաչասերման դեպքում, եթե բաղադրյալ օպերատորների չափերը չսահմանափակենք, ապա բաղադրյալ օպերատորները անընդհատ կարող են աճել:

Սագենետիկ ծրագրավորման մեջ հայտնի խնդիր է և կոչվում է *կոդի ընդլայնում* (code bloat): Սա լուրջ խնդիր է, քանի որ որքան մեծանում են բաղադրյալ օպերատորի չափերը, դրան զուգահեռ մեծանում է նրանց կատարման տևողությունը: Նշենք, որ մեծ բաղադրյալ օպերատորը պահանջում է նաև մեծ ծավալի հիշողություն: Այդ պատճառով հաճախ սահմանափակում են բաղադրյալ օպերատորների չափերը: Սահամանափակումը կատարվում է հետևյալ ձևով՝ ծնող օպերատորում փոխարինման համար պատահական հանգույց ընտրելու ժամանակ ստուգում ենք այն ենթածառի խորությունը, որը սկիզբ է առնում այդ հանգույցից: Եթե այդ ենթածառի խորության և մյուս ծնողում փոխարինման համար ընտրված հանգույցից սկսվող ենթածառի խորության գումարը փոքր է նախապես սահմանված ԳԾ-ի պարամետր հանդիսացող անհատի առավելագույն խորությունից, ապա խաչասերումը կիրականացվի: Հակառակ դեպքում՝ պատահականորեն կընտրվեն այլ հանգույցներ այնքան ժամանակ, մինչ այդ հանգույցներով իրականացված խաչասերման արդյունքում ստացված անհատի խորությունը չի գերազանցի անհատի առավելագույն խորությանը (տես՝ Նկար 3.4):

Այս մեթոդով մենք կարող ենք սահմանափակել բաղադրյալ օպերատորների չափերը, սակայն խիստ սահմանափակում ենք նաև ԳԾ-ի փնտրման տիրույթը, այսինքն՝ փոքրանում է լավ բաղադրյալ օպերատոր գտնելու հավանականությունը:



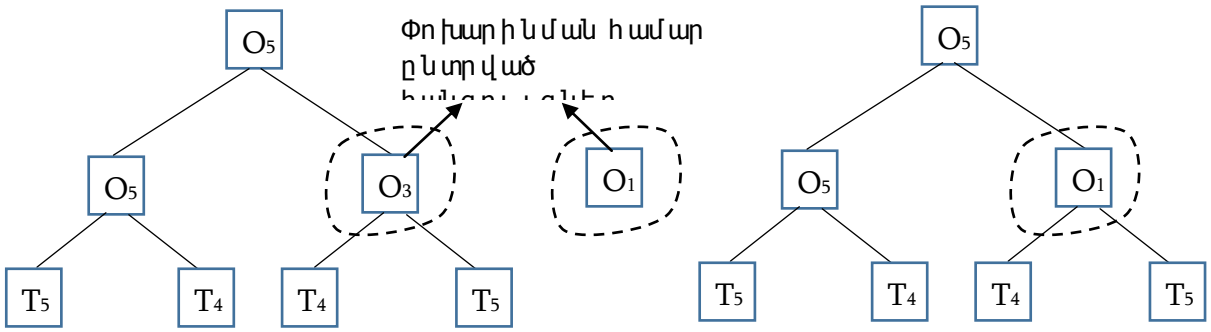


Նկար 3.4. Բաղադրյալ օպերատորների խաչասերման օրինակ, որտեղ առավելագույն խորությունը 3է: ա) Խաչասերումը չի թուլատրվում, **բ**) Խաչասերումը թուլատրվում է:

3.5.1 Մուտացիա

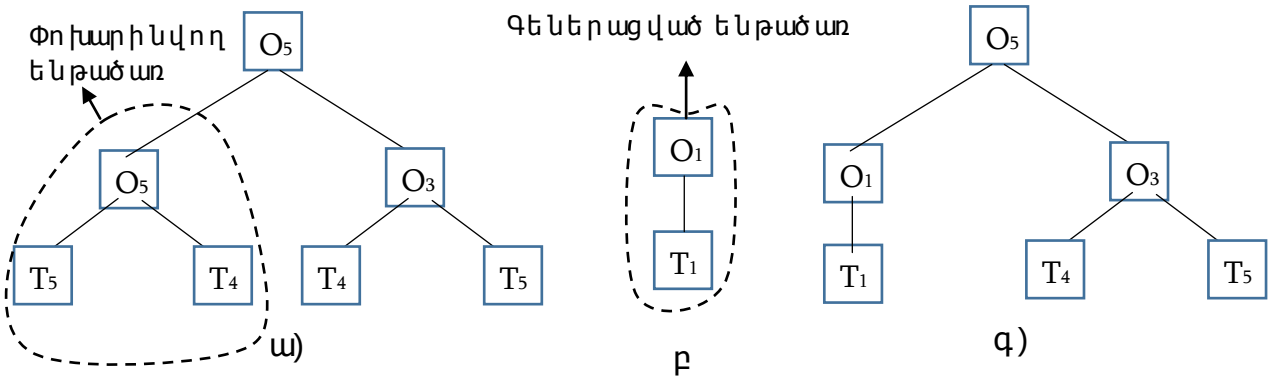
Մուտացիան իրականացվում է անհատի վրա փոփոխելով նրա կառուցվածքը: Այս գործողությունը իրականացվում է սերնդի բազմազանությունը ապահովելու նպատակով: Մենք կիրառում ենք երեք տիպի մուտացիաներ՝

1) Անհատում պատահականորեն ընտրվում է մեկ հանգույց (տարրական պատկեր կամ տարրական օպերատոր) և այն փոխարինվում է մեկ այլ համարժեք հանգույցով (տարրական պատկեր կամ տարրական օպերատոր): Եթե փոխարինման համար ընտրվել է տարրական օպերատոր, ապա նրան փոխարինող տարրական օպերատորի արգումենտների քանակը պետք է նույնը լինի (տես՝ Նկար 3.5):



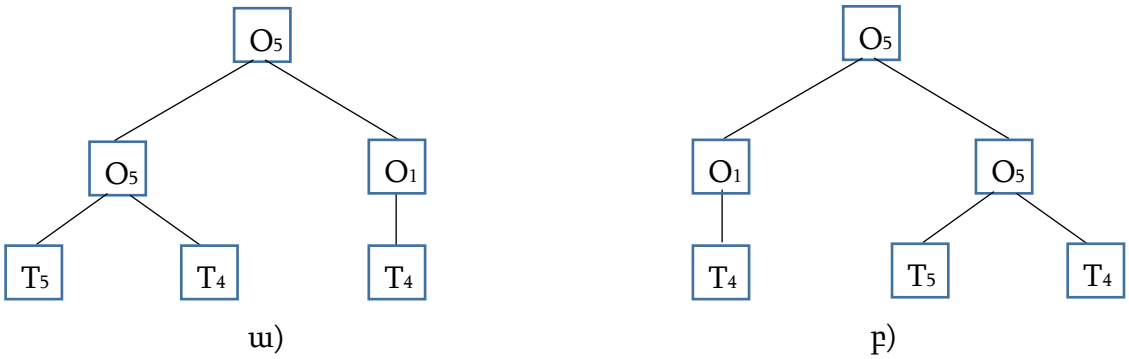
Նկար 3.5. Մուտացիան առաջին դեպքում: **ա)** Անհատ; **բ)** Պատահական ընտրված տարրական օպերատոր; **գ)** Մուտացիայի արդյունքը:

2) Անհատում պատահականորեն ընտրվում է ենթածառ և այն փոխարինվում մեկ այլ պատահականորեն գեներացված ենթածառով, ներառելով այն հանգույցը, որն ընտրվել էր (տես՝ նկար 3.6):



Նկար 3.6. Մուտացիան երկրորդ դեպքում. **ա)** Սերնդի ներկայացուցիչ; **բ)** Պատահականորեն գեներացրած ենթածառ; **գ)** Մուտացիայից հետո:

3) Անհատում պատահականորեն ընտրվում են ենթածառեր և տեղերով փոխվում են այդ ենթածառերը (տես՝ Նկար 3.7): Նշենք, որ նրանցից ոչ մեկը չի կարող լինել մյուսի ենթածառը:



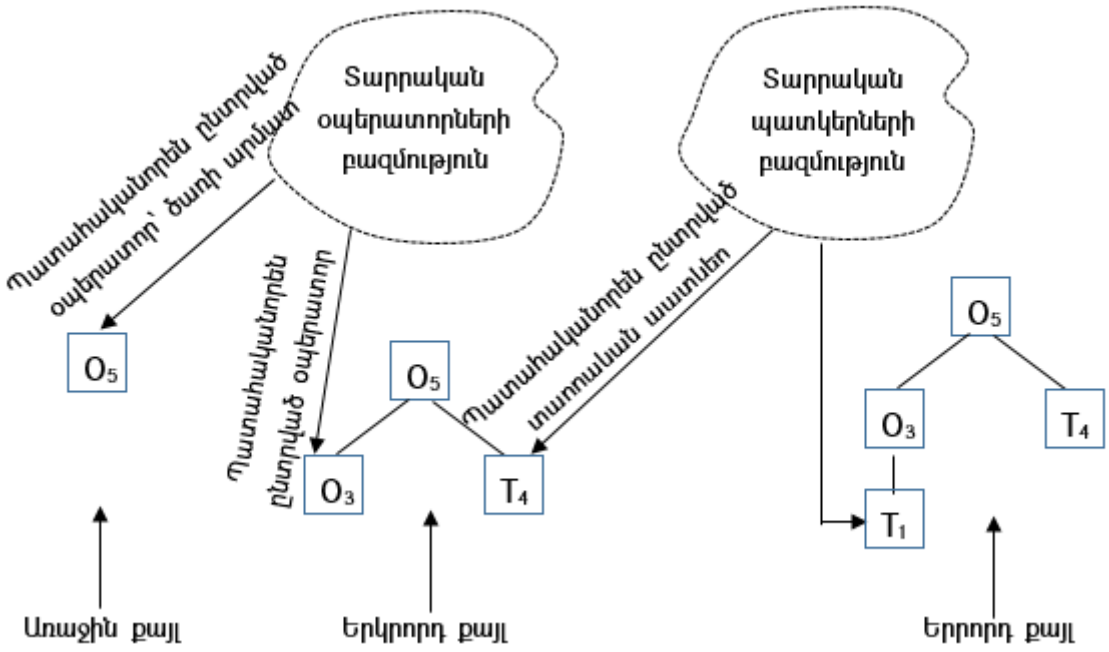
Նկար 3.7. Մուտացիան երրորդ դեպքում. **ա)** Անհատ; **բ)** Անհատը մուտացիայից հետո:

3.6 Բաղադրյալ օպերատորների առաջին սերնդի գեներացումը

Գենետիկ ծրագրավորման առաջին քայլը սկզբնական սերնդի գեներացումն է: Սկզբնական սերնդի անհատները գեներացվում են պատահականորեն: Անհատների քանակը սկզբնական սերնդում հանդիսանում է Գենետիկ ծրագրավորման համար մուտքային պարամետր: Աշխատանքում կիրառել ենք սկզբնական սերնդի գեներացման աճման, ամբողջական և խառը ալգորիթմներ, որոնք նկարագրված են Գլուխ1-ում (էջ 10-12):

Աճման ալգորիթմի օգնությամբ ամեն անգամ գեներացնում է սերնդի մեկ ներկայացուցիչ: Այս ալգորիթմով գեներացված սերնդի ներկայացուցիչը կարող է լինել նախապես սահմանված m կամ m-ից փոքր խորությամբ բինար ծառ, այսինքն՝ տվյալ անհատի առավելագույն խորությունը m է: Ստորև բերված է աճման ալգորիթմի նկարագրությունը քայլ առ քայլ (տես՝ նկար 3.8).

1. Սերնդի ներկայացուցիչի գեներացումը սկսվում է սկզբնական հանգույցի (ծառի արմատի) գեներացումով: Տարրական օպերատորների ցուցակից պատահականորեն ընտրվում է կամայական օպերատոր: Այդ օպերատորի համար պատահականորեն ընտրվում են արգումենտները, որոնք հանդիսանում են հաջորդ խորության հանգուցները:
2. Եթե ծառի տվյալ հանգույցը (նախորդ խորության հանգույցների պատահական ընտրված արգումենտը/արգումենտները) պետք է լինի տարրական պատկեր (տերմինալ կամ կախված հանգույց), ապա տերմինալների ցուցակից պատահականորեն ընտրվում է տարրական պատկեր (կախված հանգույց):
3. Եթե ծառի տվյալ հանգույցը օպերատոր է, ապա տարրական օպերատորների ցուցակից պատահականորեն ընտրվում է մի օպերատոր: Այս դեպքում տվյալ հանգույցի երեխաների քանակը հավասար կլինի ընտրված օպերատորի արգումենտների թվին:



Նկար 3.8. Ծառի գեներացումը աճման ալգորիթմով:

Եթե ծառի խորությունը հավասարվում է m-ի, ապա որպես ծառի տերևներ ընտրվում են տարրական պատկերներ:

Ամբողջական ալգորիթմի դեպքում ծառի խորությունը պարտադիր պետք է հասնի նախապես սահմանված m խորության: Այս դեպքում որպես ծառի արմատ պատահականորեն ընտրվում է տարրական օպերատոր և նրա երեխաներին պետք է ընտրել տարրական օպերատորների ցուցակից: Եվ այսպես ծառի յուրաքանչյուր խորությունում ընտրում ենք տարական օպերատորներ: Այս պրոցեսը կատարվում է մինչ ծառի խորությունը հասնում m-ի, որից հետո բոլոր հանգույցները պետք է ընտրվեն տարրական պատկերների ցուցակից:

Խառը ալգորիթմի դեպքում սերնդի ներկայացուցիչների կեսը գեներացվում է աճման, իսկ մյուս կեսը ամբողջական ալգորիթմով:

3.7 Գենետիկ ծրագրավորման *կայուն վիճակի* և *սերնդային* ալգորիթմներ

Այս աշխատանքում մենք կիրառել ենք գենետիկ ծրագրավորման երկու տիպի ալգորիթմներ՝ *կայուն վիճակի* (steady-state) և *սերնդային* (generational) [30]:

Հաստատուն վիճակի գենետիկ ծրագրավորման ալգորիթմի դեպքում, կախված իրենց ֆիտնեսային արժեքից, խաչաձևման համար ընտրվում են երկու ծնող անհատներ (բաղադրյալ օպերատորներ): Խաչաձևումից հետո ստացված երկու ժառանգները փոխարինում են սերնդում ամենափոքր ֆիտնես արժեք ունեցող բաղադրյալ օպերատորներին: Այդ երկու բաղադրյալ օպերատորները անմիջապես մասնակցում են խաչաձևման պրոցեսին մինչ խաչաձևման գործակցի բավարարումը: Այնուհետև կատարվում է մուտացիա և գնահատվում են մուտացիայի ենթարկված սերնդի բաղադրյալ օպերատորները:

Ալգորիթմը հերթական քայլերով բերված է ստորև՝

1. Սկզբնական սերունդ գեներացնող խառը ալգորիթմով գեներացնել M անհատ ունեցող բազմություն (P) և գնահատել նրա յուրաքանչյուր անհատը (բաղադրյալ օպերատորը):

2. *for gen = 1 to N* / / N-ը սերունդների քանակն է.

3. P-ում լավագույն բաղադրյալ օպերատորը պահել.

4. ***repeat***.

5. Ֆիտնեսային արժեքի հիման վրա P-ից ընտրել 2 բաղադրյալ օպերատորներ խաչաձևման համար.

6. P-ից ընտրել 2 ամենափոքր ֆիտնեսային արժեք ունեցող բաղադրյալ օպերատորները փոխարինման համար.

7. Իրականացնել խաչաձևում գործողությունը և թույլ տալ 2 նորաստեղծ ժառանգներով փոխարինել փոխարինման համար ընտրված բաղադրյալ օպերատորները.

8. Գնահատել այդ երկու ժառանգները (հաշվել նրանց ֆիտնեսային արժեքները).

9. ***until*** *crossover rate is met //* մինչ խաչասերման գործակցի բավարարումը.

10. Իրականացնել խաչասերում մինչև մուտացիայի գործակցի բավարարումը (իրականացնել մուտացիա յուրաքանչյուր բաղադրյալ օպերատորի վրա մուտացիայի գործակցի հավանականությամբ) և գնահատել մուտացիայի ենթարկված բաղադրյալ օպերատորը.

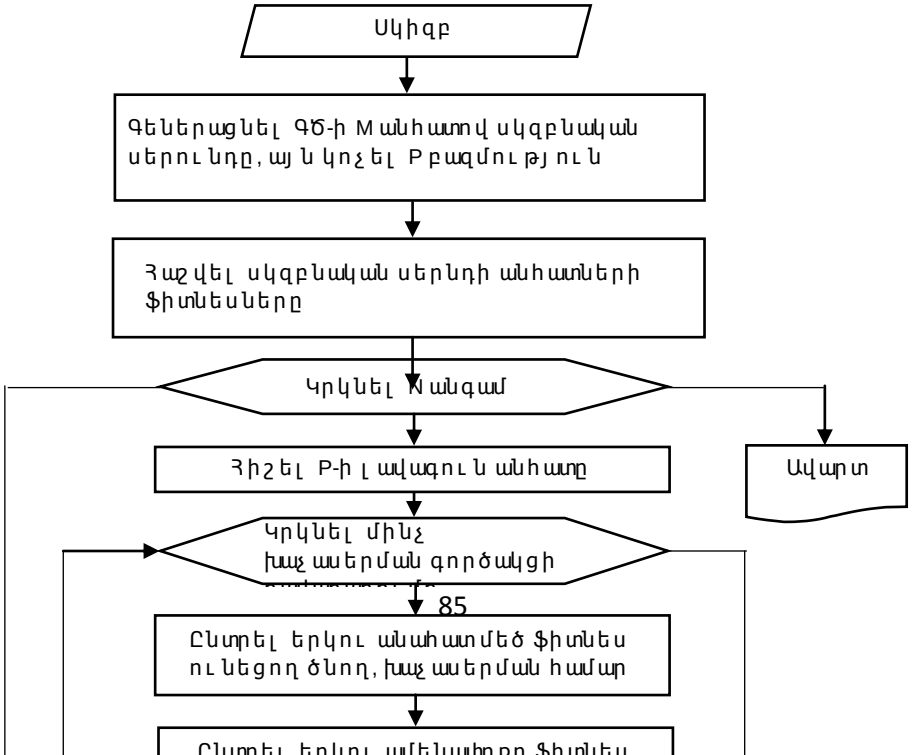
11. Մուտացիայից և խաչասերումից հետո գեներացվում է նոր անհատների P' բազմություն (պոպուլացիա).

12. P'-ում վատագույն բաղադրյալ օպերատորը փոխարինել P-ի լավագույն բաղադրյալ օպերատորով, և նախորդ սերնդին վերագրել հաջորդ սերունդը՝ P = P'.

13. Եթե P-ի լավագույն բաղադրյալ օպերատորի ֆիտնեսային արժեքը մեծ է ֆիտնեսային արժեքի շեմային արժեքից, ապա կատարել անցում 14 քայլին.

14. *Stop. Endfor.*

Կայունվիճակի գենետիկ ծրագրավորման ալգորիթմի բլոկ-սխեման բերված է Նկար 3.9-ում:



Նկար 3.9. Գենետիկ ծրագրավորման կայուն վիճակի ալգորիթմի բլոկ-սխեման:

Սերնդային գենետիկ ծրագրավորման ալգորիթմը աշխատանքը սկսում է սկզբնական սրնդի գեներացմամբ: Այնուհետև կախված իրենց ֆիտնեսային արժեքներից ընտրվում են երկու ծնող բաղադրյալ օպերատոր, խաչասերում իրականացնելու նպատակով: Խաչասերման արդյունքում գեներացվում են երկու ժառանգ բաղադրյալ օպերատորներ, որոնք չեն մտցվում ընթացիք սերնդի մեջ և չեն մասնակցում ընթացիք սերնդի հետագա խաչասերման գործողություններին: Խաչասերումը շարունակվում է մինչև խաչասերման գործակցի բավարարումը: Հետո սերնդի անհատների և խաչասերումից ստացված ժառանգների վրա կիրառվում է մուտացիա գործողությունը մինչև մուտացիայի գործակցի բավարարումը: Մուտացիայից հետո իրականացվում է մրցակցային ընտրում գործողությունը՝ ընթացիկ սերնդից ընտրվում են այնքան թվով բաղադրյալ օպերատորներ, որ ընտրվածների և խաչասերումից

ստացված ժառանգների ընդհանուր քանակը չգերազանցի ընթացիկ սերնդի անհատների քանակին: Հաջորդ քայլում գնահատվում են նոր ստացված սերնդի անհատները: Ալգորիթմը հերթական քայլերով բերված է ստորև՝

1. Սկզբնական սերունդ գեներացնող խառը ալգորիթմով գեներացնել M անհատ ունեցող բազմություն (P) և գնահատել նրա յուրաքանչյուր անհատը (բաղադրյալ օպերատորը):

2. *for gen = 1 to N* // N-ը սերունդների քանակն է.

    3. P-ում լավագույն բաղադրյալ օպերատորը պահել.

    4. P-ի անահատների վրա իրականացնել խաչասերում մինչև խաչասերման գործակցի բավարարումը և ստացված ժառանգները պահել առանձին.

    5. Կիրառել մուտացիան P-ի վրա և խաչասերումից ստացված բոլոր ժառանգների վրա, մինչև մուտացիայի գործակցի բավարարումը.

    6. P-ի վրա իրականացնել ընտրում գործողությունը, որոշ անահատներ ընտրելու համար: Ընտրված անհատների քանակը պետք է լինի M-ից այնքան պակաս, ինչքան ժառանգ որ ստացվել էր խաչասերումից.

    7. Խաչասերումից ստացված ժառանգներ համախմբել P-ից ընտրված բաղադրյալ օպերատորների հետ, որպեսզի ստանանք հաջորդ P-ի չափանի P’ սերունդ.

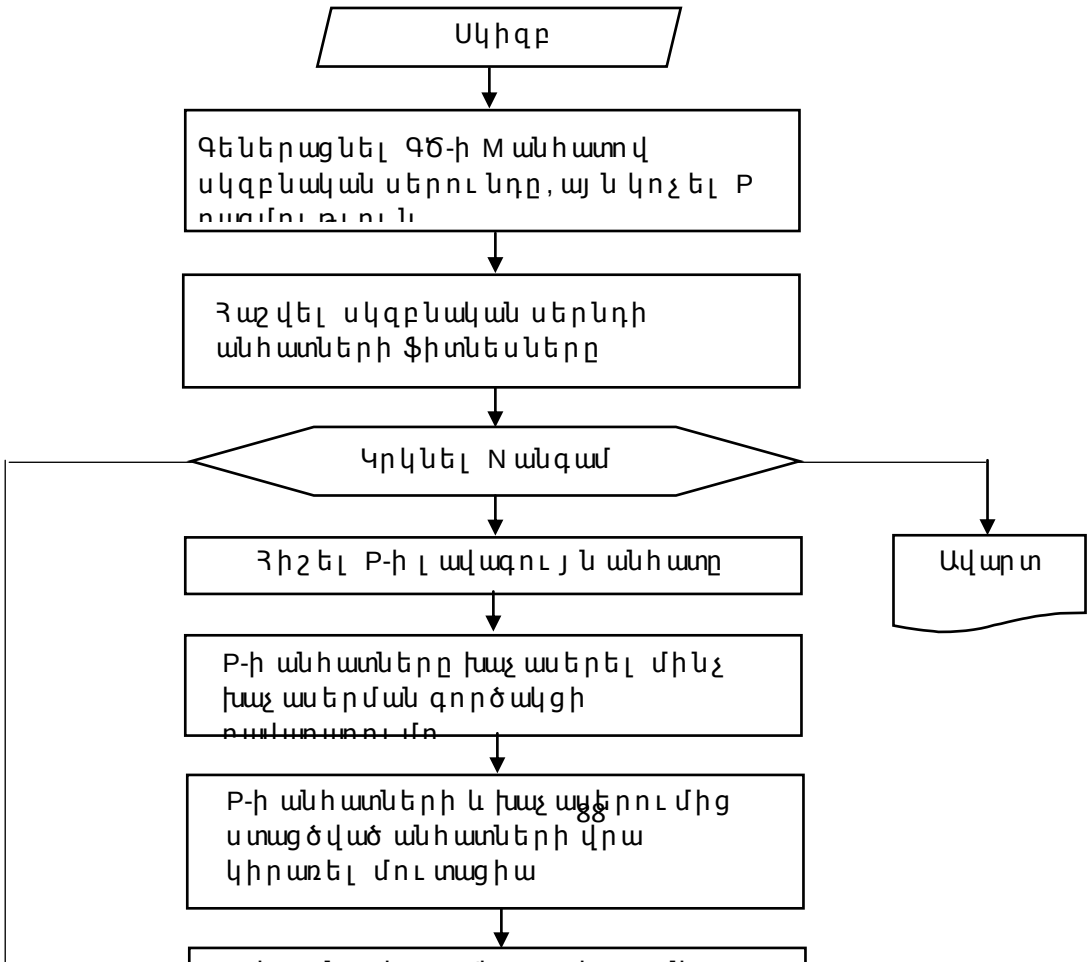
    8. Գնահատել P’ սերնդի բաղադրյալ օպերատորները.

    9. P’ –ի վատագույն բաղադրյալ օպերատորը փոխարինել P-ի լավագույն բաղադրյալ օպերատորով և նախորդ սերնդին վերագրել հաջորդ սերունդը՝ P = P’.

    10. Եթե P-ի լավագույն բաղադրյալ օպերատորի ֆիտնեսային արժեքը մեծ է ֆիտնեսային արժեքի շեմային արժեքից, ապա կատարել անցում 11 քայլին.

11. *Stop. Endfor*

Կայուն վիճակի գենետիկ ծրագրավորման ալգորիթմի բլոկ-սխեման բերված է Նկար 3.10-ում:



Նկար 3.10. Գենետիկ ծրագրավորման *սերնդային* ալգորիթմի բլոկ-սխեման:

Նշված ալգորիթմների աշխատանքի արդյունքը բաղադրյալ օպերատոր է, որի ֆիտնեսային արժեքը մեծ է ֆիտնեսի շեմային արժեքից: Նման բաղադրյալ օպերատորի օրինակ բերված է ստորև՝

FSUB(FADDC(FMAX(FADDC(FDIVC(PFIM10)))), FMIN(FMAX(FADDC (FDIVC(PFIM10))))): (3.2)

Այս բաղադրյալ օպերատորի մուտքին տալով Նկար 3.11 ա) պատկերը ելքում ստանում ենք Նկար 3.11 գ) պատկերը: Օպերատորի ֆիտնես արժեքը հաշվելու համար բաղադրյալ օպերատորի ելքում ստացված պատկերը սեգմենտացվում (տես նկար Նկար 3.11 դ-ն) և համընկման ֆունկցիայով (տես (3.1) բանաձևը) համեմատվում է Ground Truth (տես՝ Նկար 3.11 բ-ն) պատկերի հետ:

ա) բ) գ) դ)

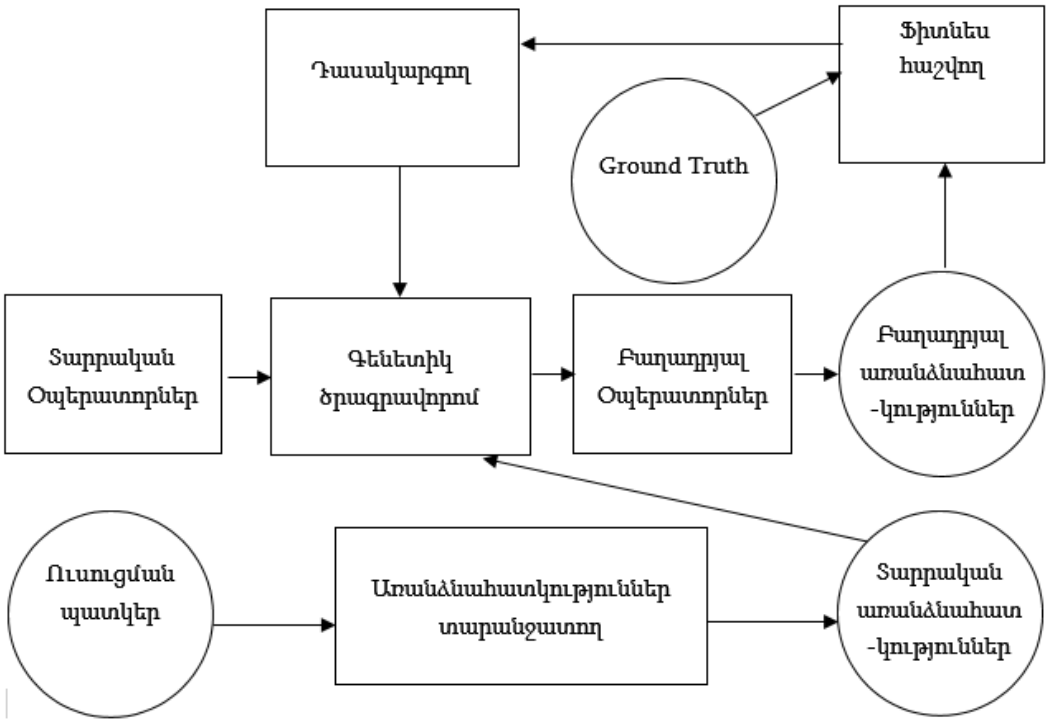
Նկար. 3.11. Սկզբնական պատկերի վրա բաղադրյալ օպերատոր կիրառելուց ստացված պատկերի օրինակ: **ա)** սկզբնական պատկեր, **բ)**

Ground Truth պատկեր, **գ)** բաղադրյալ օպերատորից ստացված պատկեր, **դ)** սեգմենտացիայից հետո ստացված պատկեր:

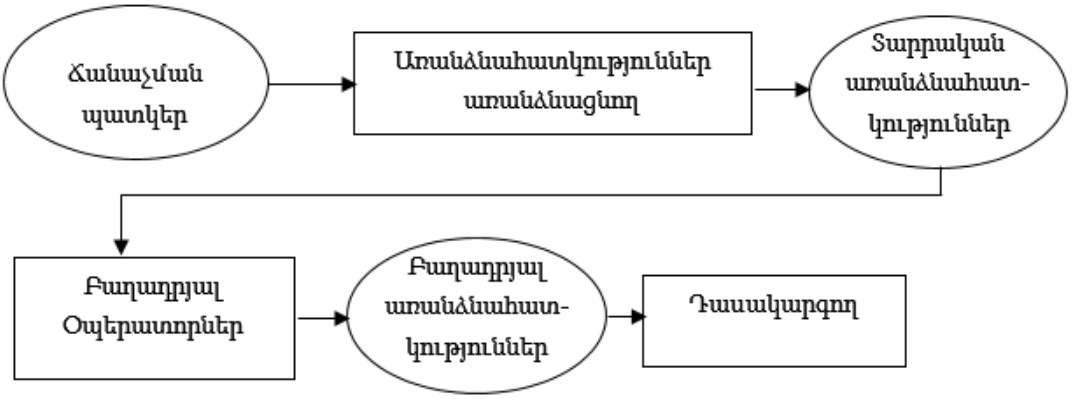
3.8 Մշակված ծրագրային համակարգի նկարագրությունը

3.8.1 Ծրագրային համակարգի մոդուլների բլոկային նկարագրությունը

Գենետիկ ծրագրավորման կայուն վիճակի և սերնդային ալգորիթմներով (տես՝ պարագրաֆ 3.6) ստեղծվել է ծրագրային համակարգ, որը հնարավորություն է ընձեռնում, մասնավորապես SAR պատկերներում օբյեկտների հայտնաբերումը և ճանաչումը: Համակարգը բաղկացած է երկու հիմնական մոդուլներից՝ուսուցման (տես՝ Նկար. 3.12) և ճանաչման (տես՝ Նկար. 3.13):



Նկար 3.12. Համակարգի ուսուցման մոդուլի բլոկ-սխեման:



Նկար 3.13. Համակարգի ճանաչման մոդուլի բլոկ-սխեման:

Ուսուցման մոդուլի բլոկ-սխեմայում *ուսուցման պատկեր* բլոկը բնութագրում է մուտքային պատկերները: Համակարգը պետք է ուսուցանվի այդ պատկերներում գտնվող օբյեկտների դասերի ճանաչման համար: *Առանձնահատկություններ տարանջատող* բլոկում հանդես են գալիս տասնվեց տարրական առանձնահատկություններով պատկերները, որոնք բերված են Աղյուսակ 3.1-ում: *Տարրական օպերատորների* բլոկում՝ 17 տարրական օպերատորներն են, որոնք բերված են Աղյուսակ 3.2-ում: *Գենետիկ ծրագրավորման* բլոկում հանդես են գալիս գենետիկ ծրագրավորման վերը նշված երկու ալգորիթմները՝ կայուն վիճակի և սերնդային: *Բաղադրյալ օպերատոր* բլոկը բնութագրում է գենետիկ ծրագրավորման

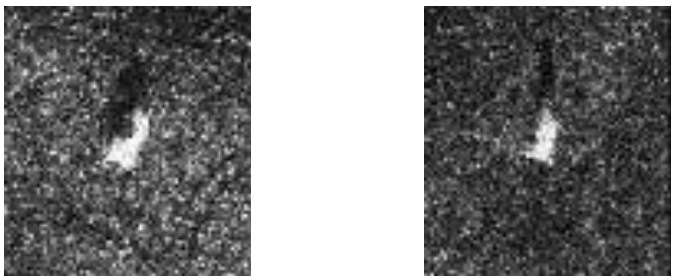
աշխատանքի արդյունքում գեներացված բաղադրյալ օպերատոները: *Ֆիտնես հաշվարկող* բլոկը, համընկման ֆունկցիան է, որն օգտագործելով Ground Truth պատկերը և բաղադրյալ առանձնահատկությունները գնահատում է բաղադրյալ օպերատորները: *Դասակարգման բլոկում* որոշվում է, թե բավարար է արդյոք գենետիկ ծրագրավորման աշխատանքի արդյունքում գեներացված բաղադրյալ օպերատորը, ուսուցման պատկերում գտնվող օբյեկտի նման օբյեկտները ճանաչելու համար:

Ճանաչման մոդուլի բլոկ-սխեմայում ճանաչման *պատկեր* բլոկը բնութագրում է համակարգին տրվող ճանաչման պատկերները, որոնցում գտնվող օբյեկտները պետք է ճանաչվի ծրագրային համակարգի կողմից: *Դասակարգման բլոկում* իրականացվում է ճանաչում:

Այսպիսով, ուսուցման մոդուլը պատասխանատու է օբյեկտների տվյալ դասի բաղադրյալ օպերատորներ գեներացնելու համար: Ճանաչման մասը պատասխանատու է ուսուցման ժամանակ գեներացված բաղադրյալ օպերատորների միջոցով օբյեկտները ճանաչելու համար:

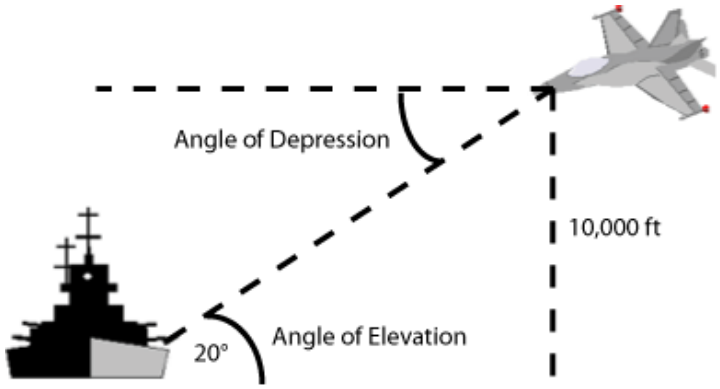
3.8.2 Համակարգի ուսուցման փուլի նկարագրությունը

Ծրագրային համակարգը թվային պատկերներում գտնվող օբյեկտները ճանաչում է ըստ դասերի: Այսինքն՝ նախապես օբյեկտները դասակարգվում են, այնուհետև տվյալ օբյեկտը ճանաչելու համար համակարգը պետք է պարզի տվյալ օբյեկտի պատկանելիությունը այս կամ այն դասին: Ստորև բերված մի քանի դասերի օրինակներ՝ *տանկը դաշտում*, որը գտնվում է 15° անկման անկյան (depression angle; տես՝ Նկար 3.15) տակ նկարահանող արբանյակի նկատմամբ: Դասին պատկանող պատկերի օրինակ է բերված Նկար 3.14ա-ում: *Տրակտորը դաշտում* ՝ նույնպես 15° անկման անկյան տակ բերված է Նկար 3.14 բ-ում, և այլն:



ա) բ)

Նկար 3.14. Արհեստական արբանյակից 15° անկման անկյան տակ արված տրանսպորտային միջոցների SAR պատկերներ; **ա)** դաշտում գտնվող տանկ; **բ)** դաշտում գտնվող տրակտոր [30]:



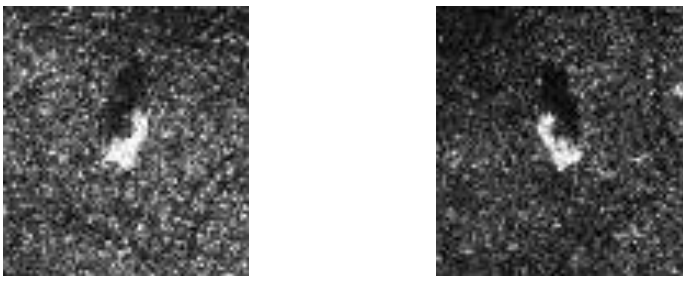
Նկար 3.15. Նկար, որում ցույց է տրված անկման անկյունը (depression angle):

Համակարգը օբյեկտները ճանաչում է ("պարզում" է, թե իր կողմից ճանաչվող որ դասին է պատկանում օբյեկտը) երկու բաղադրիչներով, դրանք են՝ բաղադրյալ օպերատոր և պատկերների բազմություն: Այս երկու բաղադրիչները ստացվում են ուսուցման մոդուլում (տես՝Նկար 3.7):

Բաղադրյալ օպերատորը գեներացվում է դասի մեկ ներկայացուցիչի միջոցով: Օրինակ, *տանկը դաշտում* դասի համար բաղադրյալ օպերատոր գեներացնելու նպատակով կարող ենք օգտագործել Նկար 3.14-ում բերված պատկերը: Դասի համար բաղադրյալ օպերատոր գեներացնելուց ուսուցման պատկերը (տես՝ Նկար 3.7 բլոկ-սխեման) հանդիսանում է տվյալ դասի ներկայացուցիչ: Ուսուցման պատկերի վրա կիրառվում են տարրական առանձնահատկություններ տարանջատող ֆունկցիաները, արդյունքում ստանում ենք այդ պատկերի տարրական առանձնահատկություններով տասնվեց պատկերներ (տես՝ Աղյուսակ 3.1): Օգտագործելով տարրական առանձնահատկություններով պատկերները և տարրական օպերատորները՝ սկզբնական սերունդ գեներացնող խառը

ալգորիթմով գեներացվում է բաղադրյալ օպերատորների առաջին սերունդը: Հաջորդ քայլում գնահատվում են առաջին սերնդի բաղադրյալ օպերատորները: Գնահատումը կատարվում է ըստ ֆիտնեսի (ֆիտնեսը հաշվվում է օգտագործելով Ground Truth պատկեր, որը ստանում ենք “ձեռքով”): Եթե առաջին սերնդի բաղադրյալ օպերատորներից լավագույնը (որի ֆիտնես արժեքը ամենամեծն է) չի բավարարում նախապես սահմանված պահանջներին (ֆիտնես արժեքը փոքր է գենետիկ ծրագրավորման համար մուտքային պարամետր հանդիսացող ֆիտնեսի շեմային արժեքից), ապա գենետիկ ծրագրավորումը շարունակում է աշխատանքը: Գենետիկ ծրագրավորումը, օգտագործելով բաղադրյալ օպերատորների առաջին սերունդը, իրականացնում է ընտրում, խաչասերում և մուտացիա, որոնց արդյունքում ստանում է բաղադրյալ օպերատորների հաջորդ սերունդը: Յուրաքանչյուր հերթական սերնդի բաղադրյալ օպերատորները գնահատվում են: Եթե նրանցից լավագույնը բավարարում է նախապես սահմանված պահանջներին, ապա գենետիկ ծրագրավորման աշխատանքը ընդհատվում է և այդ լավագույն օպերատորը հիշվում է որպես տվյալ դասի բաղադրյալ օպերատոր: Այսինքն՝ այդ բաղադրյալ օպերատորը, կիրառելով տվյալ դասի այլ պատկերների վրա, կկարողանանք հայտնաբերել այդ դասի բոլոր օբյեկտները: Բաղադրյալ օպերատորի օրինակ է (3.2)-ում բերված արտահայտությունը:

Ինչպես արդեն նշեցինք դասը բնութագրող մյուս բաղադրիչը ***պատկերների բազմությունն*** է: Այս համակարգը նպատակաուղղված է բարձրությունից նկարահանված պատկերներում օբյեկտների հայտնաբերմանն ու ճանաչմանը: Բարձրից նկարահանված պատկերներում օբյեկտների տեսքը մեծապես կախված է այն անկման անկյունից, որից կատարվել է նկարահանումը, և պատկերում օբյեկտի թեքության անկյունից: Օրինակ, տանկը դաշտում կարող գտնվել $[0, 1, ..., 359^0]$ անկյան տակ թեքված նկարահանող համակարգի նկատմամբ: Նկար 3.16-ում ցույց է տրված 15° անկման անկյունից նկարահանված տարբեր թեքություն ունեցող միևնույն տանկի պատկերներ:



Նկար 3.16. Արհեստական արբանյակից 15° անկման անկյան տակ արված միևնույն տիպի տանկերի SAR պատկերներ, որոնք նկարելու պահին ունեցել են տարբեր աստիճանի թեքվածություն նկարող սարքի նկատմամբ:

Դասը ամբողջությամբ ճանաչելու համար $[0...359^0]$ միջակայքից յուրաքանչյուր 5 աստիճան անկյան համար բազայում պահում ենք մեկ պատկեր (օրինակ՝ 0^0-4^0-ի համար համակարգի բազայում պահում ենք այն պատկերը, որը 2^0 թեքվածություն ունի, 5^0-9^0-ի համար համակարգի բազայում պահում ենք այն պատկերը, որը 7^0 թեքվածություն ունի և այլն): Հետևաբար, բազայում մեկ դասը բնութագրելու համար պահվում են 72 պատկերներ: Բազայում պահվող պատկերները բինար պատկերներ են: Այդ պատկերները ստացվում են տվյալ դասի համար ուսուցման փուլում, նախորոք գտնված բաղադրյալ օպերատորը կիրառելով 2^0, 7^0, ..., 257^0 թեքվածություն ունեցող պատկերների վրա: Այնուհետև բաղադրյալ օպերատորից ստացված պատկերները սեգմենտացվում են (սեգմենտացիայից ստացվում են բինար պատկերներ, որոնցում ընդգրկված օբյեկտների փիքսելները ստանում են սպիտակ գույն, իսկ հետևի ֆոնը սև կամ հակառակը): Սեգմենտացված պատկերներից առանձնացվում և պահվում են որպես առանձին պակերներ այն նվազագույն ուղղանակյան տիրույթները, որոնք ընդգրկում են օբյեկտները: Այս պատկերները անվանել ենք դասը բնութագրող ձևանմուշ (template) պատկերներ: Օրինակ՝ Նկար 3.16-ում բերված տանկերը ճանաչելու համար համակարգում պահվել են Նկար 3.17-ում ներկայացված բինար պատկերները մեծացված չափերով (Նկար 3.16-ի աջ պատկերի համար պահվում է Նկար 3.17-ի աջ պատկերը, իսկ Նկար 3.16-ի ձախ պատկերի համար՝ Նկար 3.17-ձախ պատկերը):

Նկար 3.17. Նկար 3.16-ում պատկերված տանկերի արհեստական արբանյակից ստացված SAR պատկերները ճանաչելու համար համակարգի բազայում պահվող բինար պատկերներ:

3.8.3 Համակարգի ճանաչման փուլի նկարագրություն

Ինչպես արդեն նշեցինք համակարգը կկարողանա ճանաչել պատկերում օբյեկտը, եթե այն դասը որին պատկանում է այդ օբյեկտը նկարագրված է համակարգում (տվյալ դասի համար գտնված է բաղադրյալ օպերատորը և ձևանմուշ պատկերները): Համակարգը պետք ունենա դասերի բազմազանություն, որպեսզի կարողանա ճանաչել այդ դասերին պատկանող օբյեկտները:

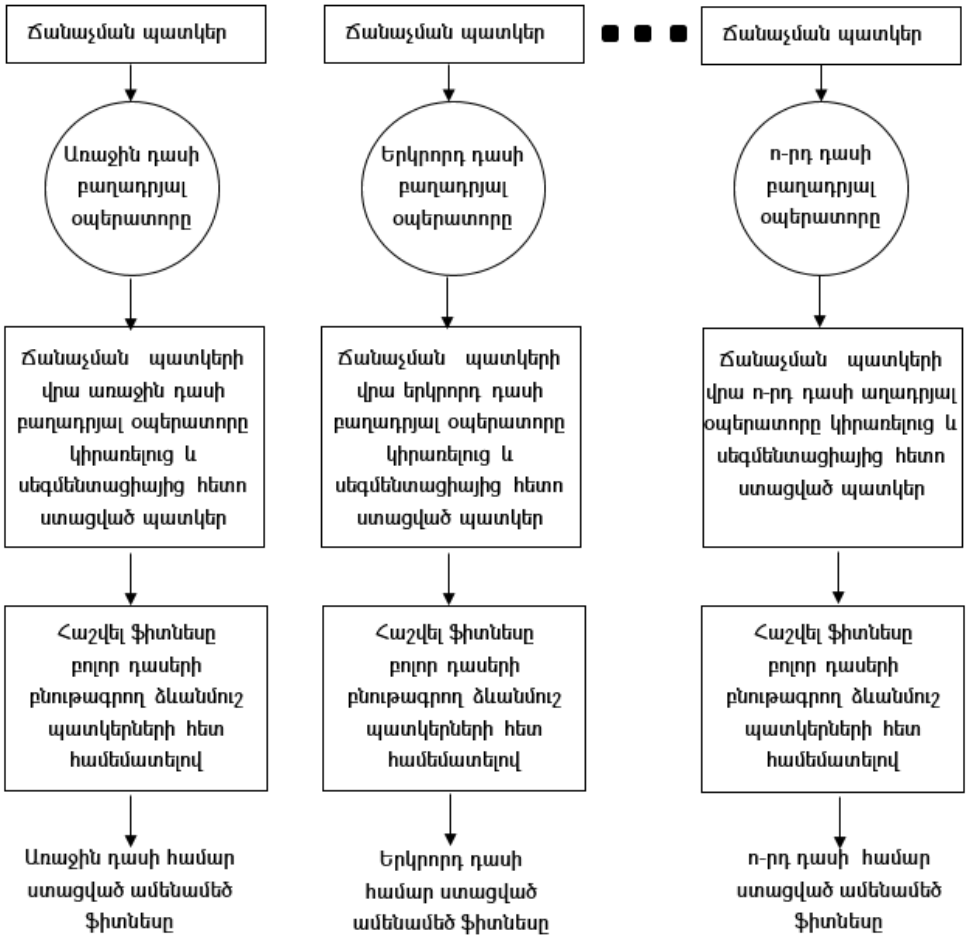
(Հետևություն. համակարգը ինչքան շատ դասեր ունենա նկարագրված, այնքան շատ տիպի օբյեկտներ կկարողանա ճանաչել):

Համակարգը օբյեկտները ճանաչում է երկու եղանակով՝

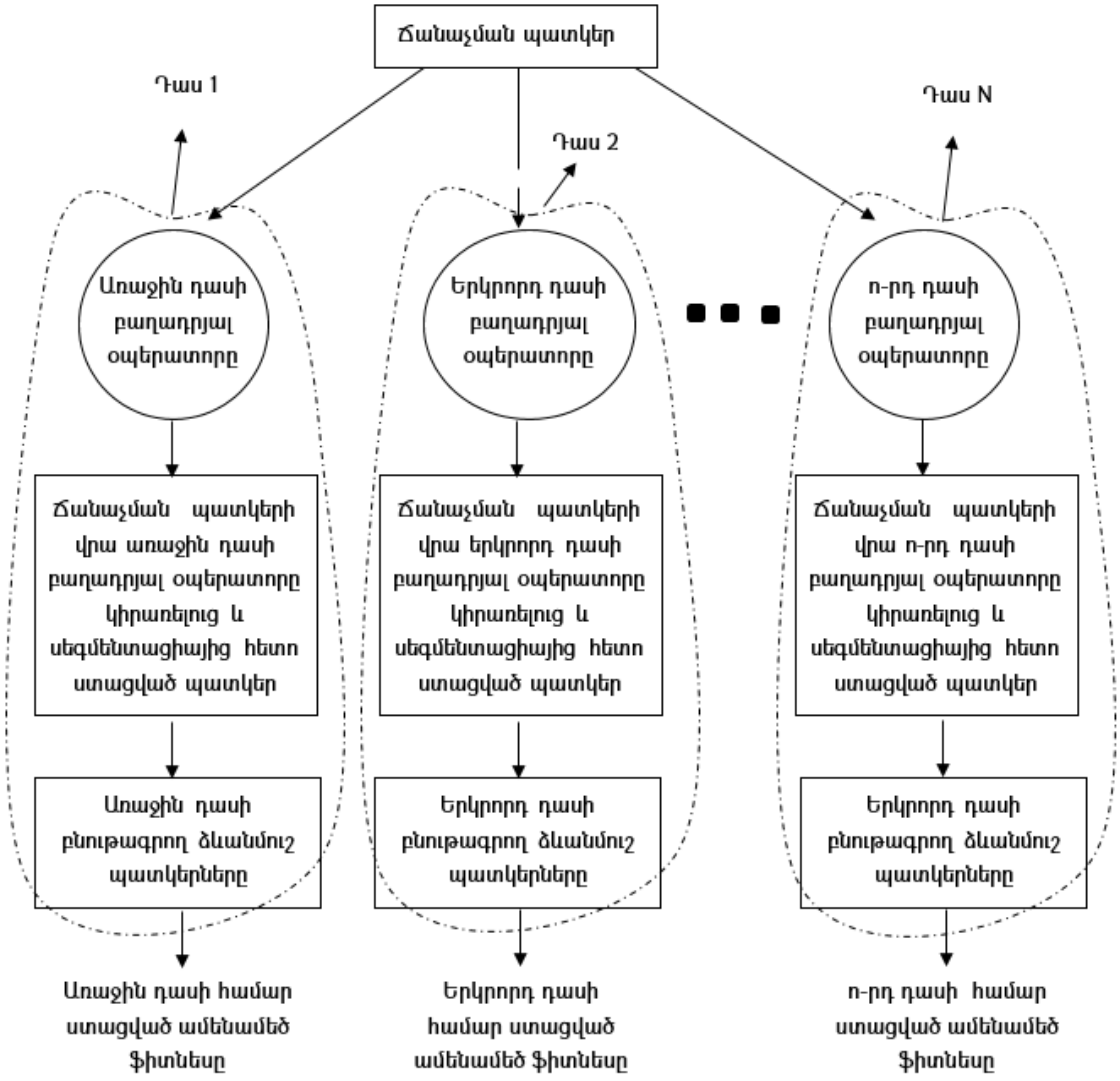
- Համակարգի կողմից օբյեկտների ճանաչման այս եղանակը կոչել ենք *հաջորդաբար ճանաչման եղանակ*: Ճանաչման այս եղանակի դեպքում, օբյեկտը ճանաչելու համար, պատկերի վրա հաջորդաբար կիրառվում են բոլոր դասերի բաղադրյալ օպերատորները: Օպերատորը կիրառելուց հետո ստացված բաղադրյալ առանձնահատկություններով պատկերը սեգմենտացվում է և համեմատվում է այդ դասը բնութագրող պատկերների հետ (համեմատությունը կատարվում է ըստ ֆիտնեսի): Օբյեկտը կհամարվի այն դասից, որի հետ համընկնումը ամենամեծն էր եղել: Ֆիտնես հաշվելու ժամանակ որպես Ground Truth պատկեր հանդիսանում են տվյալ դասը բնութագրող պատկերները (տես՝ Նկար 3.18):
- Համակարգի կողմից օբյեկտների ճանաչման երկրորդ եղանակը կոչել ենք *զուգահեռ ճանաչման եղանակ*: Այս դեպքում Ճանաչումը կատարվում է զուգահեռ՝ ըստ դասերի: Համակարգի կողմից ճանաչվող դասերից

յուրաքանչյուրի համար մեկնարկվում է պրոցեսի առանձին հոսք (thread) (օրինակ, եթե համակարգը ճանաչում է 5 տիպի օբյեկտ, ապա մեկնարկվում են 5 thread-ներ), այդ thread-ները աշխատում են զուգահեռ: Յուրաքանչյուր thread-ում տվյալ դասը բնութագրող բաղադրյալ օպերատորը կիրառվում է ճանաչվող պատկերի վրա: Ստացված բաղադրյալ առանձնահատկություններով պատկերը սեգմենտացվում է և համեմատվում այդ դասը բնութագրող պատկերների հետ, և արդյունքները հիշվում են: Բոլոր thread-ների աշխատանքի ավարտից հետո, հիշված արդյունքներից գտնվում է մեծագույնը, և ճանաչվող օբյեկտը համարվում է այն դասից, որի համար ստացվել էր մեծագույն արժեքը:

Համեմատություն կատարելուց եթե բաղադրյալ պատկերից ստացված բինար պատկերից առանձնացված օբյեկտ պարունակող ուղղանկյան չափերը չեն համընկնում Ground Truth պատկերի չափերի հետ, ապա այդ ուղղանկյան չափերը փոխվում են՝ օգտագործելով պատկերի չափերը փոփոխող երկգծային ալգորիթմը:



Նկար 3.18. Ծրագրային համակարգի հաջորդաբար ճանաչման եղանակի աշխատանքի բլոկ-սխեման:



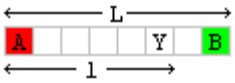
Նկար 3.18. Ծրագրային համակարգի զուգահեռ ճանաչման եղանակի աշխատանքի բլոկ-սխեման:

3.8.4. Պատկերների չափերի փոփոխում երկգծային մեթոդով

Պատկերի երկգծային մաշտաբավորումը ներառում է ներդրման սկզբունքը, հիմնված շրջապատող փիքսելների վրա, որն ապահովում է պատկերի ավելի քիչ աղավաղում: Բիլինար ալգորիթմը չի համարվում բարդ ալգորիթմ, բայց այն բաղադրյալ ալգորիթմ է հիմնված հիմնարար ֆունկցիաների վրա: Անուևից կարելի է ենթադրել, որ այն բաղկացած է երկու գծային ներդրման ալգորիթմների վրա[59]:

Գծային ներդրում. Այս մեթոդը հնարավորություն է ընձեռում պատահական կետ տեղադրել երկու կետերի միջև: Ենթադրենք ունենք երկու գույները՝ կանաչ և կարմիր, և ուզում ենք նոր գույն (կետ) տեղադրել այդ երկուսի միջև: Գծային ներդրման ալգորիթմը կիրառելով կարող ենք գտնել ամենահարմար գույնը, որը կարելի տեղադրել այդ երկուսի միջև:

Նկար 3.19-ում ներկայացված է *Y* անհայտ գույնի ներդրումը *A* (կարմիր) և *B* (կանաչ) գույների միջև. Նկար 3.19 –ից երևում է, որ հայտնի են միայն *A*-ի և *B*-ի հեռավորությունը՝ L և *A*-ի և *Y*-ի հեռավորությունը՝ l: Այս ինֆորմացիան բավական է գծային ներդրման ֆունկցիայի կառուցման համար (տես՝ արտահայտություն 3.2) [59]:

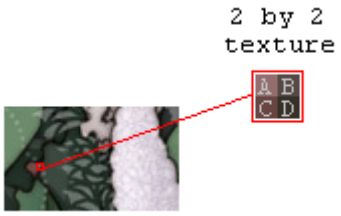


Նկար 3.19. Գծային ներդրման սխեման [59]:

$$\frac{Y-A}{l} = \frac{B-A}{L}$$

$$Y = A + \frac{l(B-A)}{L} \qquad (3.2)$$

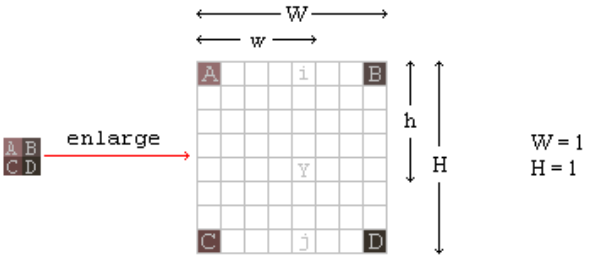
Տեքստուրա պատկերի չափերի փոփոխման բիլինար ալգորիթմում կարևոր տերմին է նաև տեքստուրան, որն իրենից ներկայացնում է պատկերի որևէ փոքր մաս: Հասարակ բիլինար մաշտաբավորման ժամանակ տեքստուրան պարունակում է 4 փիքսել (տես՝ Նկար 3.20):



Նկար 3.20. 2 x 2 տեքստուրայի օրինակ [59]:

3.8.4.1 Երկգծային ալգորիթմի նկարագրությունը

Պատկերի մաշտաբավորումը լինում է երկու տիպի չափերի մեծացում և փոքրացում: Պատկերի մեծացման դեպքում ներդրման համար ստեղծվում են նոր փիքսելներ: Կարելի է ենթադրել, որ փոքրացման ժամանակ ճիշտ համարվող փիքսելները պահվում են, իսկ միյուսները դեն նետվում. իրականում այդպես չի այս ալգորիթմով փոքրացման դեպքում փոքրացված պատկերը ստացվում է սկզբնական պատկերը փոքր չափերով: Այս դեպքում որոշ մանրամասնություններ կորում են և պատկերը չի բխում անմիջապես սկզբնական պատկերից այլ ստացվում են ինտերպոլացիայով (interpolation), ուղղակիորեն պահելով կորցված փիքսելների հատկությունները [59]: Կեսից ավելի փոքրացման և մեծացման դեպքում պատկերը նշանակալիորեն աղավաղվում է: Պատկերի մեծացումը սկսում ենք տեքստուրայի մեծացումով, ինչպես ցույց է տրված Նկար 3.21-ում: Պետք է գտնել բոլոր բացատների համար i, j, և Y արժեքները:



Նկար 3.21. Պատկերի մեծացում բացատների ներդրումով [59]:

Սկզբում պետք է գտնել A-ի, i-ի, և B-ի կախվածություն կիրառելով գծային ներդրման (3.2) արտահայտությունը կստացվի (3.3) արտահայտությունը [59]՝

$$\frac{i-A}{w}=\frac{B-A}{W}$$

$$\mathrm{i}=A+\frac{w(B-A)}{W}$$

$$\mathrm{i}=A+w(B-A)$$

Նույնը անում ենք նաև C -ի, j -ի, և D –ի համար ու ստանում ենք հետևյալ արտահայտությունը [57]՝

(3.4)

$$\frac{j-C}{w}=\frac{D-C}{W}$$

$$j=C+w(\mathrm{D}-\mathrm{C})$$

Արդյունքում ստանում ենք երկու գծային ներդրման հավասարումներ՝ (3.3) և (3.4) արտահայտությունները: Միացնելով այդ հավասարումները կստանանք մեկ բիլինար արտահայտություն [57]՝

$$\frac{Y-i}{h}=\frac{j-i}{H} \tag{3.5}$$

$$\mathrm{Y}=i+h(j-i)$$

Հավասարում (3.4)-ից հանում ենք (3.3)-հավասարումը և տեղադրում (3.5)-ի մեջ: Արդյունքում ստացվում է (3.6) հավասարումը:

$$\begin{gathered}Y=A+w(\mathrm{B}-\mathrm{A})+\mathrm{h}(\mathrm{C}+\mathrm{w}(\mathrm{D}-\mathrm{C})-(\mathrm{A}+\mathrm{w}(\mathrm{B}-\mathrm{A})))\\ Y=A(1-w)(1-h)+B(w)(1-h)+C(h)(1-w)+D(wh)\end{gathered} \tag{3.6}$$

Օգտագործելով (3.6) արտահայտությունը կարող են լրացնել Նկար 3.21-ի բոլոր բացատները:

3.9 Ծրագրային համակարգի մշակման նկարագրությունը

Ծրագրային համակարգը մշակվել է C# .NET միջավայրում: Արդեն նշվել է, որ գենետիկ ծրագրավորման մեթոդը (ալգորիթմը) իրականացնելու համար նպատակահարմար է օգտագործել այնպիսի ծրագրային լեզուներ և միջավայրեր, որոնցում

(a) աշխատում է ավտոմատ աղբ հավաքող ծրագիրը (automatic garbage collection);
(b) որպես տվյալների հիմնական տիպ առկա է դինամիկ ցուցակը (dynamic list);
(c) հնարավորություն կա հեշտությամբ նկարագրել ծառատիպ օբյեկտները;
և այլն [28]:

Աղբի հավաքումը (Garbage Collection).

Աղբի հավաքումը մի տեխնոլոգիա է, որը մի կողմից հեշտացնում է ծրագրավորողի աշխատանքը, ավտոմատ հեռացնելով դինամիկ հիշողությունում ստեղծված օբյեկտը, մյուս կողմից կարողանում ենք խուսափել ծրագրավորողի կողմից իրականացվող հիշողության կառավարման սխալներից: Աղբ հավաքողի խնդիրն է ծրագրից հեռացնել դինամիկ հիշողությունում ստեղծված այն օբյեկտները, որոնց ծրագիրը չի օգտագործում: Ծրագրավորողը միայն ստեղծում է օբյեկտ, օգտագործում այն և նա չի մտածում օբյեկտի հեռացման մասին, քանի որ միջավայրը այդ անում է ավտոմատ: Աղբի հավաքումն իրականացնելու համար միջավայրը պարունակում է լրացուցիչ աղբ հավաքող մոդուլ, որը ժամանակ առ ժամանակ աշխատում է: Այն պարզում է, թե որ օբյեկտն այլևս չի օգտագործվում և հեռացնում է հիշողությունից: [60]:

Ծրագրի կառուցվածքը: Ծրագիրը բաղկացած է մի քանի հիմնական դասերից (class)՝ տարրական պատկերների դաս (PrimitiveImages), տարրական օպերատորների դաս (PrimitiveOpertors), բաղադրյալ օպերատորներ գեներացնող դաս (Composite OperatorHelper) և օբյեկտների հայտնաբերման ու ճանաչման դաս (ObjectRecognize):

3.9.1. Տարրական պատկերների դասի (PrimitiveImages) նկարագրությունը

Տարրական պատկերների դասը ունի 18 անդամ զանգվածներ, որոնցից 16-ում պահվում են տարրական պատկերների փիքսելների արժեքները: Մյուս երկուսում՝ ImageArray-ում և TrutGroundArray-ում, համապատասխանաբար պահվում են ուսուցման կամ ճանաչման համար ծրագրային համակարգի մուտքին տրված պատկերի և TrutGround պատկերի փիքսելային արժեքները:

Վերոհիշյալ դասում նկարագրված և իրականացված են հետևյալ ֆունկցիաները.

- *LoadImage ֆունկցիան* արգումենտ չունի և արժեք չի վերադարձնում: Այն, ուսուցման կամ ճանաչման համար ծրագրային համակարգի մուտքին տրված պատկերից տվյալների մասը պահում է ImageArray-ում (դասի անդամ զանգված): Այն մուտքային պատկերի ֆայլի վերնագիրը մշակում է, նրանից վերցնում է պատկերի բարձրությունը ու լայնությունը և դրանք վերագրում է ImageHight և ImageWidth դասի անդամ փոփոխականներին համապատասխանաբար:
- *LoadTrutGround* ֆունկցիան TrutGround պատկերների տվյալները պահում է TrutGroundArray դասի զանգվածում:
- *MaxFim, MinFim, MeanFim, STDVFim, MedFim* ֆունկցիաները որպես արգումենտ ընդունում են ամբողջ տիպի փոփոխական, որն ընդունում 3, 5 կամ 7 արժեքներ (3, 5 և 7 թվերը ցուց են տալիս այն պատուհանի չափը, որն անցկացվում է սկզբնական պատկերի վրա՝ տվյալ տարրական պատկերը ստանալու համար (տես՝

պարագրաֆ 3.3)): Վերոհիշյալ ֆունկցիաները գործողությունները կիրառում են ImageArray զանգվածի վրա և արդյունքները պահում են վերոնշված 16 դասի անդամ զանգվածներում, որոնք հանդիսանում են գենետիկ ծրագրավորման կողմից օգտագործվող տարրական պատկերները: Համակարգում տարրական պատկերները համարակալված և բերված են Աղյուսակ 3.4-ում:

Աղյուսակ 3.4. Համակարգում տարրական պատկերների համարները:

Տարրական օպերատորներ	PFIM0	PFIM1	PFIM2	PFIM3	PFIM4	PFIM5	PFIM6	PFIM7
Տարրական օպերատորներ համարներ	17	18	19	20	21	22	23	24
Տարրական օպերատորներ	PFIM8	PFIM9	PFIM10	PFIM11	PFIM12	PFIM13	PFIM14	PFIM15
Տարրական օպերատորներ համարներ	24	25	26	27	28	29	30	31

3.9.2. Տարրական օպերատորների դասի (PrimitiveOpertors) նկարագրությունը

Այս դասում նկարագրված և իրականացված են աշխատանքում գենետիկ ծրագրավորման մեջ օգտագործվող 17 տարրական օպերատորները (Աղյուսակ.3.2):

- Դասի *FAdd, FSub, FMul, FDiv, FMax2 և FMin2* ֆունկցիաները որպես արգումենտ ընդունում են երեք փոփոխականներ, որոնցից առաջին երկուսը զանգվածներ են (զանգվածները պարունակում են երկու պատկերների փիքսելների արժեքները, որոնց վրա պետք է կիրառվի համապատասխան գործողությունը), երրորդը

տարրական պատկերների դասի (PrimitiveImages) օբյեկտ է, որում պահվում է պատկերի մասին անհրաժեշտ ինֆորմացիան: Նրանք վերադարձնում են զանգված: Ֆունկցիաները որպես արգումենտ փոխանցված զանգվածների համապատասխան (զանգվածում նույն ինդեքս ունեցող) արժեքների վրա կիրառում են համապատասխան թվաբանական գործողությունները և արդյունքները պահում մի այլ զանգվածում, որոնք հանդիսանում են ֆունկցիաների վերադարձվող արժեքները:

- Դասի *FAddC, FSubC, FMulC և FDivC* ֆունկցիաները ընդունում են երեք արգումենտներ: Արգումենտներից առաջինը պատկերի (այն պատկերի, որի վրա պետք է իրականացվի համապատասխան գործողությունը) փիքսելների արժեքների զանգված է, երկրորդը՝ տարրական պատկերների դասի օբյեկտ, երրորդը՝ ամբողջ (int) տիպի փոփոխական է, որը [-20,20] միջակայքից որևէ ամբողջ թիվ է: Այս ֆունկցիաների վերադարձվող արժեքները զանգված են, որոնց էլեմենտները ստացվում են հետևյալ կերպ: Ֆունկցիաները իրենց որպես առաջին արգումենտ փոխանցված պատկերի փիքսելների արժեքների (զանգվածի յուրաքանչյուր անդամի) և որպես արգումենտ փոխանցված C ամբողջ թվի վրա կիրառում են համապատասխան թվաբանական գործողություններ: Թվաբանական գործողության արդյունքը պահվում է մի այլ զանգվածում, որն էլ ֆունկցիայի վերադարձվող արժեքն է:
- Դասի *FSqrt և Flog* ֆունկցիաները ընդունում են երկու արգումենտներ: Առաջինը պատկերի փիքսելների արժեքների զանգված է, երկրորդը՝ տարրական պատկերների դասի օբյեկտ: Նրանք վերադարձնում են զանգվածներ: Ֆունկցիաները իրենց որպես առաջին արգումենտ փոխանցված պատկերի փիքսելների արժեքների վրա կատարում են համապատասխանաբար քառակուսի արմատ և լոգարիթմ գործողությունները: Գործողության արդյունքը պահվում է մի այլ զանգվածում, որոնք էլ ֆունկցիաների վերադարձվող արժեքներն են: Նախ վերցվում

են այդ թվերի բացարձակ արժեքները, այնուհետև կատարվում են համապատասխան գործողությունները:

- Դասի *FMax, FMin, FMed, FMean և FSTDV* ֆունկցիաները ընդունում են երեք արգումենտներ: Արգումենտներից առաջինը պատկերի փիքսելների արժեքների զանգվածն է, երկրորդը՝ տարրական պատկերների դասի օբյեկտ, երրորդը՝ int տիպի փոփոխական է, որն ընդունում 3, 5 կամ 7 արժեքներ: Այն ցույց է տալիս թե պատկերի 3, 5 թե 7 հարևաններից որն օգտագործել (տես՝ Պարագրաֆ 3.3; տարրական օպերատորներ): Այս ֆունկցիաները վերադարձնում են զանգված տիպի օբյեկտներ: Ֆունկցիաները իրենց որպես առաջին արգումենտ փոխանցված պատկերի փիքսելների արժեքների վրա կիրառում են համապատասխան գործողությունները: Այդ գործողությունները կիրառելուց, 3, 5 կամ 7 հարևաների ստացման պատուհանի չափերը վերցվում են ֆունկցիայի երրորդ արգումենտի չափով: Գործողությունները իրականացնելուց հետո ստացվում են նոր զանգվածներ, որոնք հանդիսանում են ֆունկցիաների վերադարձվող արժեքները:

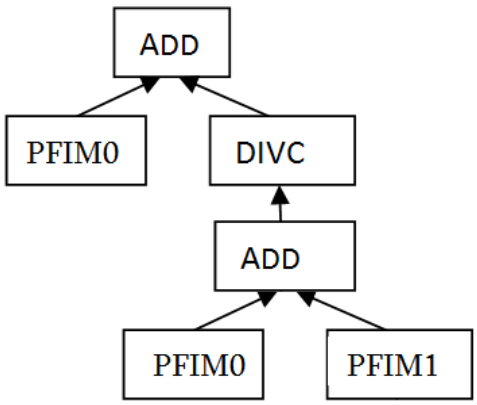
Ծրագրային համակարգում այս ֆունկցիաները համարակալված և բերված են Աղյուսակ 3.5-ում:

Աղյուսակ 3.5. Համակարգում տարրական օպերատորների համարները:

Տարրական օպերատորներ	FAdd	FSub	FMul	FDiv	FMax2	FMin2	FAddC	FSubC	
Տարրական օպերատորների համարներ	0	1	2	3	4	5	6	7	
Տարրական օպերատորներ	FMulC	FDivC	FSqrt	Flog	FMax	FMin	FMed	FMean	FSTDV
Տարրական օպերատորներ համարներ	8	9	10	11	12	13	14	15	16

3.9.3. Բաղադրյալ օպերատորներ գեներացնող դասի (CompositeOperatorHelper) նկարագրությունը

Ծրագրային համակարգում գենետիկ ծրագրավորման անհատները (բաղադրյալ օպերատորները) հանդիսանում են System.Xml.XmlDocument դասի տիպի օբյեկտներ: Այսինքն՝ բաղադրյալ օպերատորը, որը ներկայացվում էր ծառի տեսքով (տես՝ պարագրաֆ 3.2), նկարագրվում է Xml տեքստային ֆորմատով [61]: Բաղադրյալ օպերատորը ծառի տեսքով և Xml նկարագրությամբ բերված է Նկար 3.22-ում:




```
<operator>
    <root number='0'>
        < functiontoin number='17' />
        <functoin number='9' random='-7'>
            < functoin number='0'>
                < functoin number='17' />
                < functoin number='18' />
            </ functoin >
        </ functoin >
    </root>
<operator>
```

Նկար 3.22. Բաղադրյալ օպերատորը ներկայացված ծառի և Xml դոկումենտի տեսքով:

Նկար 3.22-ում operator էլեմենտը Xml-ի արմատ էլեմենտն է: root էլեմենտը բաղադրյալ օպերատորը ներկայացնող ծառի արմատն է, որն ունի մեկ *number* անունով ատրիբուտ: *numberber* ատրիբուտը ցույց է տալիս տարրական օպերատորի համարը, որը կարող է ընդունել 0-16 արժեքներ (արդեն նշել ենք, որ տարրական օպերատորները ծրագրային համակարգում ունեն 0-16 համարներ): Xml-ի մյուս *function* էլեմենտները նկարագրում են բաղադրյալ օպերատորի /ծառի/ համապատասխան հանգույցները: Այդ էլեմենտները ունեն մեկ կամ երկու ատրիբուտ: numberber ատրիբուտը ցույց է տալիս տվյալ հանգույցի տարրական օպերատորի կամ տարրական պատկերի համարը: *random* ատրիբուտ ունեն այն էլեմենտները, որոնց numberber ատրիբուտի արժեքը կամ՝ [6-9] միջակայքից են (այս դեպքում ատրիբուտի արժեքին համապատասխան

համարով տարրական օպերատորներում օգտագործված հաստատունի արժեքն է), կամ՛ [12-16] (այս դեպքում ցույց է տալիս, թե ատրիբուտի արժեքին համապատասխան համարով տարրական օպերատորների 3x3, 5x5 և 7x7 հարևաններից որն է օգտագործված):

Ծրագրային համակարգում գենետիկ ծրագրավորման սերունդն իրենից ներկայացնում է System.Xml.XmlDocument տիպի դինամիկ ցուցակ (System.Collections.Generic. List<System.Xml.XmlDocument>): Այն հայտարարված է CompositeOperatorHelper դասում Population անունով:

Համակարգում սկզբնական սերունդը գեներացվում է խառը ալգորիթմով, որի իրականացման նպատակով CompositeOperatorHelper դասում նկարագրվել և իրականացվել են երկու ֆունկցիաներ՝ FirstPopulFull և FirstPopulGrow: Վերջիններս պատասխանատու են ածման և ամբողջական ալգորիթմներով սկզբնական սերնդի գեներացման համար:

- *FirstPopulFull* ֆունկցիան արգումենտներ չունի և այն վերադարձնում է System.Xml.XmlDocument տիպի օբյեկտ (գենետիկ ծրագրավորման անհատ՝ բաղադրյալ օպերատոր): Այս ֆունկցիան սկզբնական սերունդ գեներացնող ամբողջական ալգորիթմի իրականացումն է: Սերնդի անհատ գեներացնելու համար, այն ստեղծում է System.Xml.XmlDocument տիպի օբյեկտ՝ compositeOperator անունով և նրան ավելացնում է operator անունով Xml էլեմենտ: Այնուհետև operator-ին որպես երեխա-էլեմենտ ավելացվում է root անունով էլեմենտ, որն ունի number ատրիբուտ: number ատրիբուտին վերագրվում է [0-16] միջակայքից պատահական թիվ (որն տարրական օպերատորի համարն է): Ֆունկցիայում հայտարարվում են երկու զանգվածներ: Զանգվածներից առաջինում պահվում է root անունով էլեմենտը: Այնուհետև ցիկլով գեներացվում են տվյալ անհատի՝ ծառի հերթական խորության հանգույցները: Ցիկլը կրկնվում է սկզբնական սերնդի անհատների խորության (ԳԾ-ի համար մուտքային պարամետր) քանակով: Ցիկլի յուրաքանչյուր քայլում կատարվում է մի այլ ցիկլ, որը դիտարկում է առաջին զանգվածի անդամների numberber ատրիբուտները, և կախված այդ ատրիբուտների արժեքներից գեներացվում են նոր Xml

էլեմենտներ, որոնք նույնպես ունեն [0-16] միջակայքից numberber ատրիբուտ: Այդ էլեմենտները ավելացվում են առաջին զանգվածի դիտարկվող անդամին որպես երեխա: Երկրորդ ցիկլի ավարտից հետո առաջին զանգվածի տարրերը փոխարինվում են երկրորդ զանգվածի տարրերով: Այլ կերպ ասած՝ զանգվածներից առաջինում պահվում են ընթացիք խորության հանգույցները, երկրորդում՝ հաջորդ խորության հանգույցները: Երբ առաջին ցիկլերի քանակը հասնում է սկզբնական սերնդի անհատների խորությանը, ապա երկրորդ ցիկլի ընթացքում ստեղծվող նոր Xml Էլեմենտների number ատրիբուտի արժեքները պատահականորեն ընտրվում են [17-32] միջակայքից: Արդյունքում ստանում ենք Xml դոկումենտի տեսքով նկարագրված բաղադրյալ օպերատոր, որին ֆունկցիան վերադարձնում է որպես System.Xml.XmlDocument տիպի օբյեկտ:

- *FirstPopulGrow* ֆունկցիան հանդիսանում է սկզբնական սերունդ գեներացնող ածման ալգորիթմի իրականացումը, որն աշխատում է նույն սկզբունքով, ինչ FirstPopulFull ֆունկցիան: Տարբերությունն այն է, որ երկրորդ ցիկլի ընթացքում ստեղծվող Xml էլեմենտների number ատրիբուտների արժեքները պատահականորեն ընտրվում են [0-32] միջակայքից, իսկ սկզբնական սերնդի անհատների նախապես սահմանված խորության հասնելու դեպքում ընտրվում են [17-32] միջակայքից:

Այս երկու ֆունկցիաները սկզբնական սերնդի անդամների քանակությամբ հաջորդաբար աշխատացվում են, և նրանց վերադարձրած անհատները ավելացվում են Population զանգվածին:
Համակարգում գենետիկ ծրագրավորման գործողությունները (ընտրում, խաչասերում և մուտացիա) իրականացված են Selection, Crossover և Mutation ֆունկցիաների տեսքով, որոնք հանդիսանում են CompositeOperatorHelper դասի անդամներ:

- *Selection* ֆունկցիան կիրառվում է անհատ ընտրելու համար: *Selection* ֆունկցիան մրցակցային ընտրում է իրականացնում: Ընտրումը կատարվում է սերնդից (Population) խաչասերման կամ մուտացիայի համար անհատ ընտրելու նպատակով: Մրցակցային

ընտրության դեպքում անհատները ընտրվում են պատահականորեն՝ կախված իրենց ֆիտնեսի արժեքներից (մեծ ֆիտնես ունեցողները առավել հավանական են): Այդպիսի ընտրում կատարելու համար Population զանգվածը դասավորվում է ըստ ֆիտնեսի նվազման կարգով: Խաչասերման համար երկու ծնող անհատ ընտրելուց, նրանցից առաջինը ընտրվում է Population-ի առաջին մասի անհատներից, երկրորդ ընտրելու համար՝ պատահական գեներացվում է [0 - խաչասերման գործակից] միջակայքից որևէ թիվ և Population զանգվածից վերցվում է այդ թվի ինդեքսով անդամը: Մուտացիայի դեպքում ընտրվում են մուտացիայի գործակցի քանակի անհատներ, որոնց կեսը ընտրվում են դասավորված Population զանգվածի առաջին մասից, միւս կեսը՝ [0 - մուտացիայի գործակից] միջակայքից:

- *Crossover* ֆունկցիան ընտրված երկու անհատների միջև իրականացնում է խաչասերում: Ֆունկցիային որպես արգումենտ փոխանցվում են խաչասերման համար ընտրված երկու անհատներ՝ծնողներ: Ֆունկցիան ստեղծում է երկու նոր XmlDocument տիպի օբյեկտներ, որոնք հանդիսանում են խաչասերումից ստացվող ժառանգներ: Այնուհետև, ֆունկցիան իրեն փոխանցված ծնողների պատճենները (clone) վերագրում է երկու ժառանգներին: Այնուհետև ժառանգներից յուրաքանչյուրում պատահականորեն ընտրվում են Xml էլեմենտներ (խաչասերման կետեր) և ընտրված էլեմենտներն ու նրանց երեխաները տեղերով փոխվում են: Արդյունքում ստացվում են երկու նոր անհատներ՝ժառանգներ:

Մուտացիա

Համակարգում կատարվում են երեք տիպի մուտացիաներ, որոնց համար իրականացվել են Mutation1, Mutation2 և Mutation3 ֆունկցիաները: Մուտացիայի համար անհատ ընտրելուց հետո որոշվում է, թե որ մուտացիան է պետք կատարել: Որոշումը կատարվում է պատահականության սկզբունքով՝ 0-2 թվերից պատահական ընտրվում է որևէ մեկը. եթե ընտրվում է 0-ն, ապա կանչվում է Mutation1, 1-ի

դեպքում՝ Mutation2-ը, 2-ի դեպքում՝ Mutation3-ը: Մուտացիայի համար կիրառվող ֆեւնկցիաները նկարագրված են ստորև՝

1. Mutation1. ֆունկցիան որպես արգումենտ ընդունում է int տիպի փոփոխական, որը Population զանգվածում անհատի ինդեքսն է: Տվյալ անհատի մեջ պատահականորեն ընտրվում է որևէ հանգույց (Xml էլեմենտ) և դիտարկվում նրա number ատրիբուտը: Եթե ատրիբուտի արժեքը [0-5] միջակայքից է (երկու արգումենտանոց օպերատոր է), ապա այդ միջակայքից ընտրվում է մի այլ պատահական թիվ և ընտրված հանգույցի number ատրիբուտի արժեքը փոխարինվում է այդ թվով: Եթե ատրիբուտի արժեքը [6-16] միջակայքից է, ապա պատահական թիվը ընտրվում է այդ միջակայքից: Նույնը կատարվում է նաև [17-32] միջակայքի համար:
2. Mutation2. այս դեպքում մուտացիայի համար ընտրված անհատի որևէ ենթածառ փոխարինվում է մի այլ պատահականորեն գեներացված ենթածառով: Ֆունկցիան գեներացնում է նոր անհատ (System.Xml.XmlDocument օբյեկտ) և կանչում է խաչասերում ֆունկցիան, որին որպես արգումենտ փոխանցվող ծնողներից առաջինը Population-ի այն անհատն է, որը պետք է մուտացիայի ենթարկվի, երկրորդը՝ պատահականորեն գեներացված անհատը: Խաչասերումից ստացված ժառանգներից առաջինը հանդիսանում է Population զանգվածի մուտացիայի համար ընտրված անահատի մուտացիան:
3. Mutation3. այս դեպքում մուտացիայի համար ընտրված անհատի երկու ենթածառեր տեղերով փոխվում են: Ֆունկցիան անհատի մեջ ընտրում է նույն խորության երկու պատահական հանգույցներ (Xml էլեմենտներ) և նրանց տեղերով փոխում:

Անհատների գնահատումը իրականացվում է հետևյալ ֆունկցիաներով՝ Fitness, Segmentation, CalculateMember: Վերջիններս CompositeOperatorHelper դասի անդամեն:

CalculateMember ֆունկցիան նախատեսված է գենետիկ ծրագրավորման անհատը հաշվարկելու համար: Մեր համակարգում անհատը բաղադրյալ օպերատոր է, որի բաղկացուցիչ մասերը տարրական առանձնահատկություններով պատկերներ են և այդ պատկերների վրա

կիրառված տարրական օպերատորներ: Հետևաբար, բաղադրյալ օպերատորը հաշվարկելու արդյունքում ստացվում է բաղադրյալ առանձնահատկություններով պատկեր: Ֆունկցիային որպես արգումենտ փոխանցվում է System.Xml.XmlElement տիպի օբյեկտ, այն վերադարձնում է զանգված, որը պատկերի փիքսելների արժեքներն են: *CalculateMember* ֆունկցիան ռեկուրսիվ ֆունկցիա է: Նախնական փոխանցվում անհատի root էլեմենտը: Այնուհետև *CalculateMember* ֆունկցիան դիտարկում է իրեն փոխանցված root էլեմենտի number ատրիբուտը: Եթե ատրիբուտի արժեքը [0-16] միջակայքից է, ապա կանչվում է այդ էլեմենտի համապատասխան տարրական օպերատորը (որն ունի մեկ կամ երկու արգումենտ՝ երեխա): Այնուհետև նրա արգումենտների (երեխաների) համար կանչվում է *CalculateMember* ռեկուրսիվ ֆունկցիան: Եթե [17-32] միջակայքից է ֆունկցիան ուղղակի վերադարձնում է այդ էլեմենտին համապատասխան տարրական առանձնահատկություններով պատկերը:

- *Segmentation* ֆունկցիան կիրառում ենք CalculateMember ֆունկցիայի վերադարձրած զանգվածի՝ պատկերի վրա, որպեսզի ստանանք բինար պատկեր: Այն որպես արգումենտ ընդունում է զանգված, threshold, maxValue, minValue int տիպի փոփոխականներ: Որպես արգումենտ փոխանցված զանգվածը հանդիսանում է այն պատկերը, որը պետք է սեգմենտացիայի ենթարկվի: Threshold-ը այն շեմային արժեքն է, որից փոքր արժեքները պատկերում պետք համարել 0, իսկ մեծերը՝ 1: maxValue և minValue զանգվածում (պատկերում) փիքսելների մեծագույն և փոքրագույն արժեքներն են: Եթե maxValue-ն մեծ է 1000-ից, ապա կատարվում է փիքսելների արժեքների 0-1000 միջակայքի նորմալացում (տես՝ (3.7)-բանաձևը), որտեղ org_pixval-ն զանգվածի ներթական անդամն է, new_pixval փիքսելի նոր արժեքը [30]:

$$new\_pixval = (org\_pixval - minValue) / (maxValue - minValue\,1000) \qquad (3.7)$$

Նորմալացում կատարելուց հետո այն արժեքները, որոնք մեծ են շեմայինից համարվում են 1 (255), իսկ փոքրերը՝ 0: Ֆունկցիան վերադարձնում է սեգմենտացիայի ենթարկված պատկերը:

- *Fitness* ֆունկցիան անհատի ֆիտնեսն հաշվելու համար է: Այն որպես արգումենտ ընդունում է զանգված (պատկերի փիքսելների զանգվածը) և վերադարձնում է float տիպ՝ անհատի ֆիտնեսը: Այն կատարում է համեմատություն իրեն որպես արգումենտ փոխանցված զանգվածի և TrutGround զանգվածների անդամների միջև: Համեմատությունը կատարվում է ըստ (3.1) բանաձևի:

3.9.4. Օբյեկտների հայտնաբերման և ճանաչման (ObjectRecognize) դասի նկարագրությունը

Ինչպես նշեցինք վերևում օբյեկտը ճանաչելու համար (պարզելու թե որ դասին է պատկանում օբյեկտը), համակարգը օգտագործում է դասը բնութագրող ձևանմուշ պատկերների բազմությունը (յուրաքանչյուր դաս իրենից ներկայացնում է դասի ներկայացուցիչի 72 տարբեր դիրքերի բինար պատկերներ) և դասը բնութագրող բաղադրյալ օպերատորները: Ծարագրային համակարգում ճանաչվող օբյեկտների դասը բնութագրող բաղադրյալ օպերատորները և ձևանմուշ պատկերները պահպանվում են առանձին թղթապանակներում և յուրաքանչյուր դասի համար ստեղծված է իր անունով թղթապանակ:

Ծրագրային համակարգը օբյեկտները ճանաչում է երկու եղանակով՝ հաջորդական և զուգահեռ (տես՝ պարագրաֆ 3.7.3): Ճանաչման եղանակները իրականացնելու համար ObjectRecognize դասում նկարագրված և իրականացված են Recognize և RecognizeMulty ֆունկցիաները, որոնք համապատասխանաբար կիրառվում են հաջորդական և զուգահեռ ճանաչման դեպքում:

Առավել մանրամասն դիտարկենք RecognizeMulty ֆունկցիան: Այն արժեք չի վերադարձնում և ունի 4 արգումենտ՝

- դասի թղթապանակի ամբողջական հասցեն: Թղթապանակում պահվում են դասը բնութագրող բաղադրյալ օպերատորներն ու ձևանմուշ պատկերները;

- սկզբնական պատկերի ամբողջական հասցեն, որում գտնվող օբյեկտը պետք է ճանաչել;
- Dictionary<string, float> տիպի classesFitneses անվանումով դինամիկ ցուցակ;
- List<Thread> տիպի threadsArray անվանումով դինամիկ ցուցակ:

RecognizeMulty ֆունկցիան աշխատանքը սկսում է սկզբնական պատկերի բեռնումով, այնուհետև ստեղծում է PrimitiveImages դասի տիպի օբյեկտ, որը գեներացնում է տարրական պատկերներ: Այնուհետև բեռնում է բաղադրյալ օպերատորն ու հաշվարկում է ստացվող բաղադրյալ առանձնահատկություններով պատկերը, որը պահվում է CompositeFeaturesImage զանգվածում: Պատկերը ստանալուց հետո, ֆունկցիան ցիկլով անցնում է ճանաչվող պատկերների վրայով՝ հերթով հաշվում է նրանց ֆիտնեսները և պահում Fetneses զանգվածում: Ցիկլի ավարտից հետո Fetneses զանգվածի մեծագույն արժեքը և դասի անվանումը գրանցվում են classesFitneses ցուցակի մեջ:

Ծրագրային համակարգը ճանաչվող յուրաքանչյուր դասի համար մեկական Thread է մեկնարկում, որոնք աշխատացնում են RecognizeMulty ֆունկցիան: Thread-ների ցուցիչները պահվում են threadsArray զանգվածում, որը հանդիսանում է RecognizeMulty ֆունկցիայի փոխանցվող արգումենտ: Ֆունկցիայի աշխատանքի ավարտից հետո Thread-ը հեռացնում է իր ցուցիչը threadsArray զանգվածից: Սա արվում է աշխատող Thread-երը ղեկավարելու համար:

Thread-երի աշխատանքի ավարտից հետո classesFitneses ցուցակից ընտրվում է մեծագույն ֆիտնես ունեցող անդամը:

Պատկերում ընդգրկված օբյեկտները ընդգրկող տիրույթների (ուղղանկյունների) կոորդինատները գտնելու և այդ տիրույթների որպես առանձին պատկերի չափերի, փոփոխման համար ObjectRecognize դասում իրականացված են GetObjectsRectangle, GetMinRectObjects և ObjectResize ֆունկցիաները:

- GetObjectsRectangle. CompositeFeaturesImage պատկերի սեգմենտացիայից հետո, նրանում գտնվող օբյեկտները

ընդգրկող տիրույթների կոորդինատները հայտնի չեն, այդ տիրութները կարող են տարբեր չափերի լինել և կարող են կեղծ օբյեկտներ ընդգրկել (այսինքն՝ պատկերի այդ հատվածում օբյեկտ չլինի): Այս ֆունկցիան սեգմենտացիայից հետո պատկերում առանձնացնում է այն ուղղանկյունները, որոնք ենթադրաբար ընգրկում են պատկերի օբյեկտները: Ֆունկցիան որպես արգումենտ ընդունում է մի զանգված (պատկեր) և երկու int տիպի width և height փոփոխականներ և վերադարձնում է զանգված, որի անդամները System.Drawing.Rectangel տիպի օբյեկտներ են: Ֆունկցիայում հայտարարվում է System.Collections.Generic.List<System.Drawing.Rectangel> տիպի ObjetsRectangeles անունով ցուցակ (ֆունկցիայի աշխատանքի արդյունքում, ցուցակի անդամները պետք է լինեն պատկերի մեջ այն ուղղանկյուն կոորդինատները, որոնք ընդգրկում են օբյեկտները): Ֆունկցիան սկսում է աշխատել դիտարկելով պատկերի հերթական բլոկերը: Այդ բլոկների երկարությունները width չափի են, իսկ բարձրությունները՝ height: Եթե այդ բլոկներում սպիտակ փիքսելների քանակը մեծ է 95%-ից, ապա բլոկը համարում ենք օբյեկտին պատկանող բլոկ: Տվյալ բլոկը, որպես օբյեկտ պարունակող բլոկ, փորձում ենք ավելացնել ObjetsRectangeles ցուցակին.

- Եթե ցուցակում անդամների քանակը 0 է, ապա ցուցակում բլոկի կոորդինատներով և չափերով System.Drawing.Rectangel տիպի օբյեկտ է ավելացվում;
- Եթե ցուցակում անդամների քանակը 0 չէ, ապա հերթով դիտարկվում են ObjetsRectangeles զանգվածի անդամները, եթե այդ բլոկի գագաթներից մեկը ընկած է ObjetsRectangeles զանգվածի անդամներից մեկի մեջ, ապա զանգվածի այդ անդամի չափերը և կոորդինատները լրացվում են դիտարկվող բլոկի չափերով և կոորդինատներով: Եթե այդ բլոկի գագաթները զանգվածի անդամներից ոչ մեկին չեն պատկանում, ապա զանգվածին ավելանում է բլոկի կոորդինատներով և չափերով նոր System.Drawing.Rectangel տիպի օբյեկտ:

ObjetsRectangeles ֆունկցիան ավարտում է աշխատանքը, երբ դիտարկվում են պատկերի բոլոր բլոկները: Ֆունկցիան վերադարձնում է ObjetsRectangeles զանգվածը:

- *GetMinRectObjetc*. GetObjectsRectangle-ի վերադարձրած ուղղանկյուններից յուրաքանչյուրը կարող է պարունակել պատկերի հավելյալ ֆոն, սյսինքն, այդ ուղղանկյունները կարող են ավելի մեծ լինել, քան անհրաժեշտ է նրանում գտնվող օբյեկտը ամբողջությամբ ընդգրկելու համար: Քանի որ դասի ճանաչման պատկերները ընտրված են այնպես, որ նրանց բարձրությունն ու լայնությունը համապատասխանում են նրանցում գտնվող օբեկտի բարձրության ու լայնության, ապա *GetMinRectObjetc* ֆունկցիան ուղղանկյուններից հեռացնում է նշված ավելորդ մասերը (տես՝ Նկար 2.23): Այն որպես արգումենտ ստանում է System.Drawing.Rectangel տիպի օբյեկտ՝ rectang անունով, և վերադարձնում է նույն տիպի օբյեկտ: Որպես արգումենտ փոխանցված System.Drawing.Rectangel տիպի օբյեկտը GetObjectsRectangle ֆունկցիայի վերադարձրած ուղղանկյուններից է: GetObjectsRectangle ֆունկցիան պատկերի rectang-ի կոորդինատներով և չափերով ընդգրկված հատվածից գտնում է ամենաձախ, ամենաաջ, ամենավերին և ամենաստորին սպիտակ փիքսելների կոորդինատները և rectangle ուղղանկյան չափերը սեղմվում են մինչ այդ կոորդինատները, որի արդյունքում ձևավորվում է նոր ուղանկյուն, որն էլ հանդիսանում է *GetMinRectObjetc-ի* վերադարձվող արժեքը:

ա) բ)

Նկար 3.23: Պատկերից առանձնացված օբյեկտպարունակող տրիույթի օրինակ. **ա)** GetObjectsRectangle ֆունկցիայի վերադարձրած օբյեկտընդգրկող ուղանկյան օրինակ, **բ)** GetMinRectObjetc ա) պատկերի վրաաշխատացնելուց ստացված ուղղանկայն պատկերը:

- *ObjectResize* ֆունկցիան իրականացված է պատկերների չափերը փոփոխելու համար: Ֆունկցիան որպես արգումենտ ընդունում զանգված (պատկերի տիրույթի փիքսելների արժեքները) և երկու int տիպի width և height փոփոխականներ: Ֆունկցիան իրեն որպես արգումենտ փոխանցված պատկերի (զանգվածի) չափերը փոխում է երկգծային (տես՝ պարագրաֆ 3.7.4) ալգորիթմով՝ երկարությունը ստացվում է width արգումենտին հավասար և բարձրությունը՝ height արգումենտին:

ԳԼՈՒԽ 4

ՄՇԱԿՎԱԾ ԾՐԱԳՐԱՅԻՆ ՀԱՄԱԿԱՐԳԻ ՄԻՋՈՑՈՎ ԿԱՏԱՐՎԱԾ ՓՈՐՁԱՐԿՈՒՄՆԵՐԻ ԱՐԴՅՈՒՆՔՆԵՐԸ

Այս գլխում նկարագրված են մշակված ծրագրային համակարգով իրականացված փորձերը: Փորձերի նկարագրության մեջ դիտարկվում են համակարգի ուսուցման և ճանաչման փուլերը: Փորձերի արդյունքում ծրագրային համակարգը ռադիոտեղորոշիչ պատկերներից հայտնաբերել և ճանաչել է տարատեսակ օբյեկտներ: Գլխում բերված է նաև ծրագրային համակարգի օգտագործողի ուղեցույցը:

4.1 Ծրագրային համակարգով օբյեկտների հայտնաբերումը և ճանաչումը

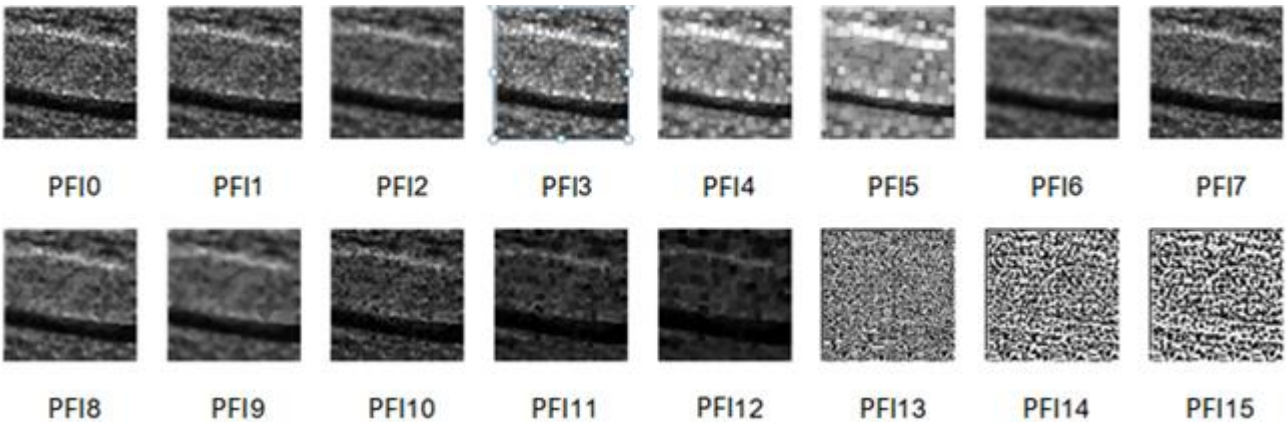
Ծրագրային համակարգով կատարվել են պատկերներից օբյեկտների հայտնաբերման և ճանաչման փորձարկումներ: Փորձերը իրականացվել են արհեստական արբանյակից ստացված SAR պատկերների վրա[62, 30]:

Առաջին փորձի նպատակն է գեներացնել քոմփյութերային ծրագիր (բաղադրյալ օպերատոր(ներ)), որի միջոցով հնարավոր կլինի հայտնաբերել *դաշտում* առկա *ճանապարհ* տիպի օբյեկտը (հետաքրքրության հատվածները կամ օբյեկտները՝ ճանապարհները): Բաղադրյալ օպերատոր գեներացնելու համար օգտագործվել է Նկար 4.1 ա)-ում ներկայացված պատկերը:

ա) բ) գ) դ)

Նկար 4.1: ա) սկզբնական պատկեր, **բ)** ground truth պատկեր, **գ)** բաղադրյալ օպերատորից ստացված պատկեր, **դ)** սեգմենտացիայից հետո ստացված պատկեր:

4.1) պատկերի համար գեներացվել են տասնվեց տարրական պատկերներ (տես՝ Աղյուսակ 3.1), որոնք ներկայացված են Նկար 4.2-ում:



Նկար 4.2. Նկար 4.1ա) պատկերից ստացված տարրական առանձնահատկություններով տասնվեց պատկերները:

Ծրագրային համակարգը տվյալ դասի բաղադրյալ օպերատոր գեներացնելու համար աշխատել է 4 անգամ: 4-րդ անգամ աշխատելու ժամանակ ծրագրային համակարգը տարրական օպերատորների խմբից պատահականորեն ընտրել է MIN, ADD, MIN2, ADDC, MUL, ADD և MAX տարրական օպերատորները և տարրական պատկերներից PFI10 և PFI2

տարրական պատկերները: Աշխատանքի ընթացքում գենետիկ ծրագրավորման 8-րդ սերնդում ստացվել է անհատ (բաղադրյալ օպերատոր), որի ֆիտնեսի արժեքը 0.9 է: Որպես Ground Truth պատկեր օգտագործվել է նկար 4.1բ պատկերը, որը ստացվել է նկար 4.1ա պատկերից: Ստորև բերված է նշված բաղադրյալ օպերատորը՝

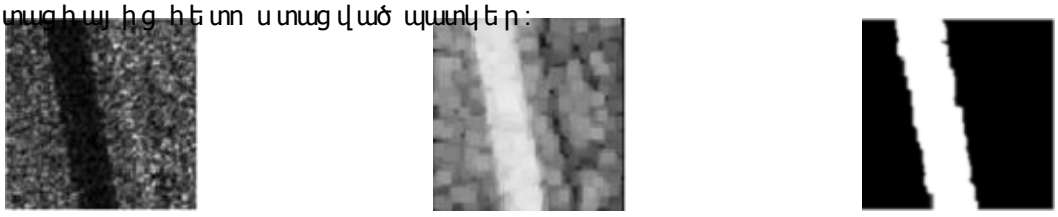
MIN(MIN(ADD(MIN2(ADDC(MUL(ADD(PFI10, PFI2), ADD(PFI10, FI2)))), MAX(LOG(MIN2(MULC(PFI10)))))): (4.1)

Այս բաղադրյալ օպերատորը ունի 17 հանգույց, որոնցից 5-ը տերևներ են (տարրական պատկերներ), իսկ 12-ը տարրական օպերատորներ: Բաղադրյալ օպերատորի բոլոր հանգույցներում ստացված պատկերները ծառի տեսքով բերված են Նկար 4.5-ում, իսկ նրանց սեգմենտացված պատկերները և ֆիտնեսի արժեքները բերված են 4.6-ում:

Վերոնշված (4.1) բաղադրյալ օպերատորը կիրառվել է նկար 4.3ա) և Նկար 4.4ա) պատկերների վրա, որոնց համար բաղադրյալ օպերատորի վերադարձրած պատկերները բերված են համապատասխանաբար Նկար 4.3բ) և Նկար 4.4բ)-ում: Բաղադրյալ օպերատորից ստացված պատկերի սեգմենտացնելով ստացվել են 4.3գ) և 4.4գ) բինար պատկերները համապատասխանաբար 0.937 և 0.943 ֆիտնեսային արժեքներով:

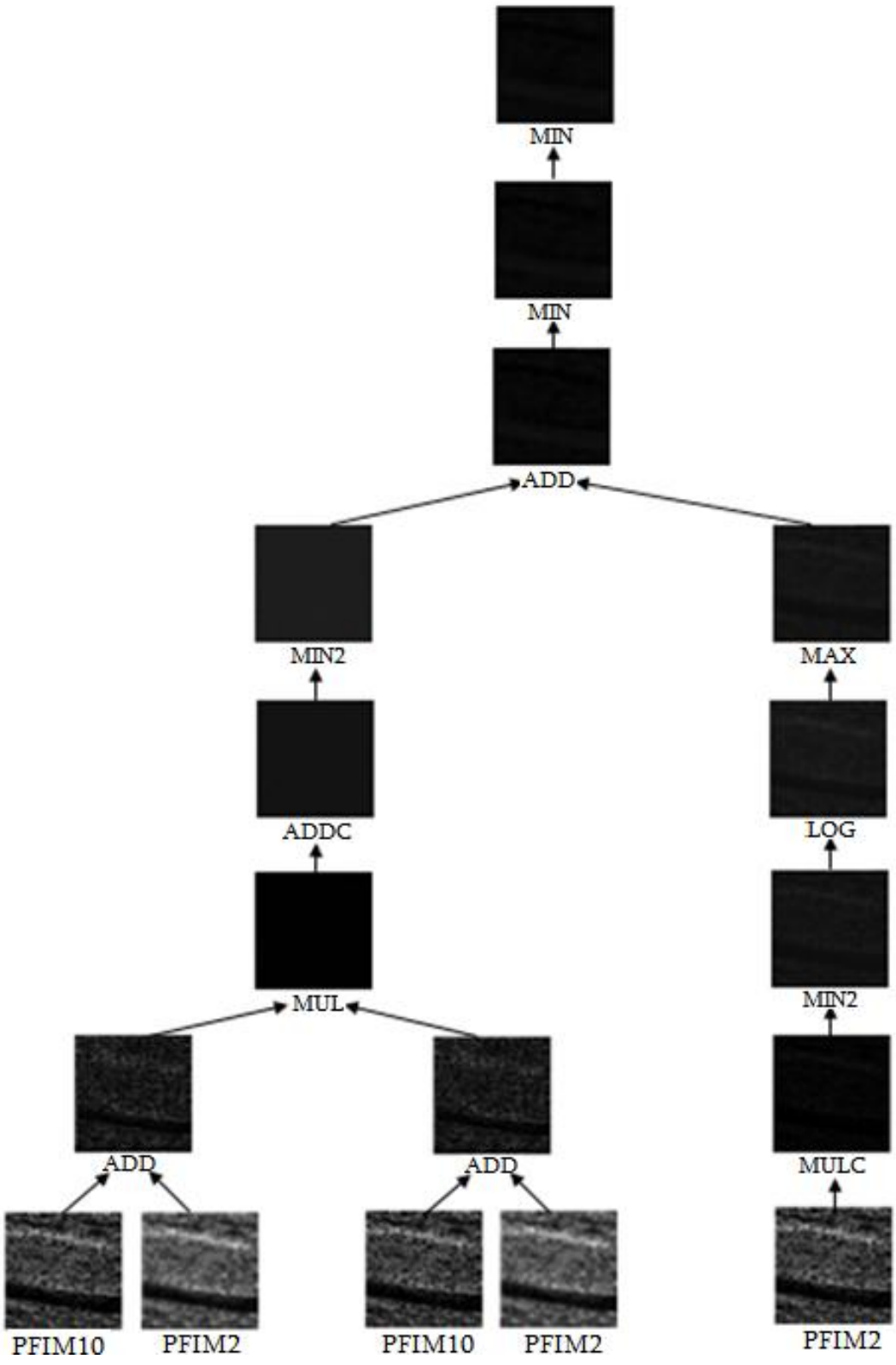
ա) բ) գ)

Նկար. 4.3. Փորձի համար օգտագործված հորիզոնական ճանապարհի դաշտում պատկեր. **ա)** սկզբնական պատկեր, **բ)** բաղադրյալ օպերատորից ստացված պատկեր, **գ)** սեգմենտացիայից հետո ստացված պատկեր:

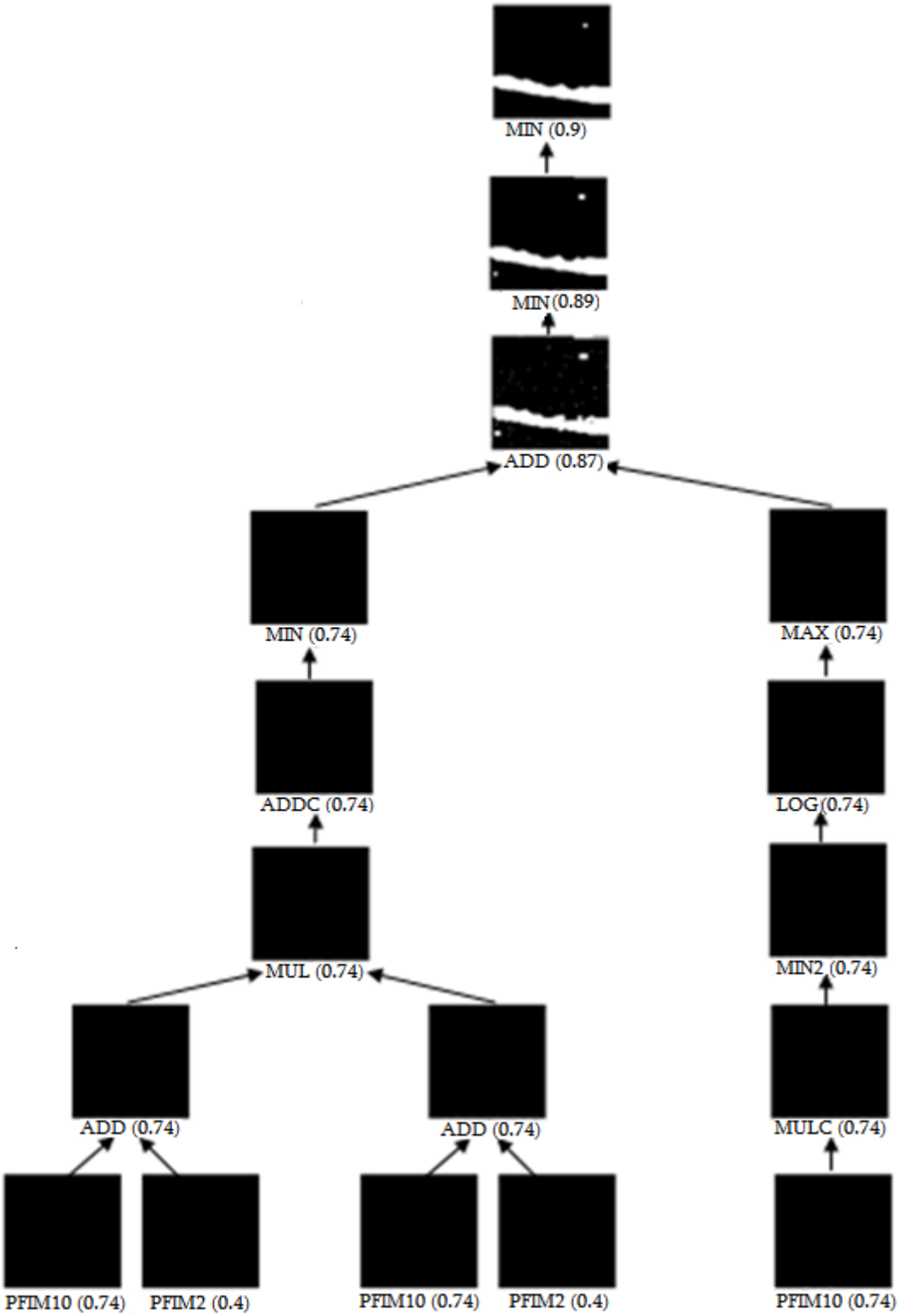


ա) բ) գ)

Նկար. 4.4. Փորձի համար օգտագործված ուղղահայաց ճանապարհի դաշտում պատկեր. **ա)** սկզբնական պատկեր, **բ)** բաղադրյալ օպերատորից ստացված պատկեր, **գ)** սեգմենտացիայից հետո ստացված պատկեր:



Նկար 4.5. Ճանապարհ-դաշտիդենտֆիկացնող բաղադրյալ օպերատորը բոլոր հանգուցներում ստացված պատկերների ծառի տեսքով:



Նկար 4.6. Բաղադրյալ օպերատորը հանգուցներում ստացված սեգմենտացված պատկերների ծառի տեսքով:

Երկրորդ փորձ. Փորձերը կատարվել են արհեստական արբանյակից ստացված տանկերի և լճերի SAR պատկերների վրա:

Փորձի առաջին փուլում, օգտագործելով Նկար 4.7ա) պատկերը [30] (որը հանդիսանում է արհեստական արբանյակից ստացված դաշտում գտնվող տանկի SAR պատկեր) գեներացվել է բաղադրյալ օպերատոր, որի կիրառմամբ *տանկը դաշտում* պատկերից հայտնաբերում ենք օբյեկտը՝ տանկը:

ա) բ) գ)

Նկար 4.7. ա) փորձի համար օգտագործված տանկի սկզբնական պատկեր, **բ)** բաղադրյալ օպերատորից ստացված պատկեր, **գ)** սեգմենտացիայից հետո ստացված պատկեր:

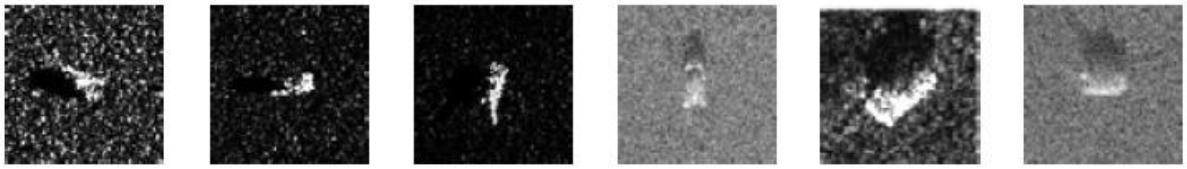
Բաղադրյալ օպերատոր գեներացնելու նպատակով գենետիկ ծրագրավորման սկզբնական սերունդը ստանալու համար օգտագործվել է 4 տարրական պատկեր և 6 տարրական օպերատոր: Սկզբնական սերնդում անհատների քանակը 10 է և յուրաքանչյուրի խորությունը՝ 5: Ստորև բերված է գեներացված բաղադրյալ օպերատորը.

ADD (ADD (MAX(DIV (ADD (MAX2 (MAX2 (PFII7, PFI13), DIVC (PFI7)),ADD (DIVC (PFI0), ADD(PFI0, PFI13))), SQRT (MAX2 (SQRT (PFI13), ADD(PFI9, PFI13))))), DIVC (MAX (MAX2 (DIV (SQRT (PFI13), DIV (PFI13,PFI0)), SQRT (DIVC (PFI7)))))), MAX2 (MAX2 (ADD (DIV (DIV (PFI13,PFI13), SQRT (PFI9)), ADD (MEAN (PFI9), MAX2 (PFI9, PFI7))), DIVC(DIVC (DIVC (PFI13)))), MEAN (MAX2 (MAX2 (DIV (PFI13, PFI7), DIVC(PFI0)), MAX (DIVC (PFI13)), SQRT (DIVC (DIVC (PFI0))))))): (4.2)

“*Տանկը դաշտում*” դասի համար բաղադրյալ օպերատոր գեներացնելու նպատակով մեր ծրագրային համակարգը աշխատացվել է 21 անգամ (առաջին 20-ի ժամանակ նախնական պահանջներն բավարարող օպերատոր չի ստացվել): 21-րդ անգամ աշխատացնելու ընթացքում գենետիկ ծրագրավորման 8-րդ սերնդում գեներացվել է բաղադրյալ

օպերատոր, որն ունի 0.9 ֆիտնեսային արժեք: Նույն բաղադրյալ օպերատորը կիրառվել է դաշտում գտնվող արհեստական արբանյակից ստացված տանկի այլ պատկերների վրա, որի արդյունքները ներկայացված են Նկար 4.8-ում:

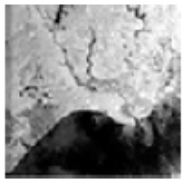
ա)



բ)

Նկար 4.8. Տանկի SAR պատկերներ. ա) սկզբնական պատկերներ, բ) բաղադրյալ օպերատորից և սեգմենտացիայից հետո ստացված պատկեր:

Փորձի երկրորդ փուլում գեներացվել է լիճ հայտնաբերող բաղադրյալ օպերատոր, որի գեներացման համար օգտագործվել է Նկար 4.9ա) պատկերը [30]:



ա)

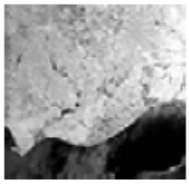
բ)

Նկար 4.9. ա) Լճի սկզբնական պատկեր, **բ)** բաղադրյալ օպերատորից և սեգմենտացիայից հետո ստացված պատկեր:

Գեներացված բաղադրյալ օպերատորը ունի հետևյալ տեսքը՝

ADD (MED (DIV (SUB (PFI8, PFI8), SUB (PFI7, PFI9))), SUB (ADD (MED (DIV (MAX2 (PFI13, PFI13), ADD (PFI8, PFI9)))), SUBC (DIV (PFI9, PFI9)))): (4.3)

Այն կիրառվել է Նկար 4.10ա)-ում պատկերված լճի վրա [30]: Արդյունքը բերված է նկար 4.10բ)-ում:



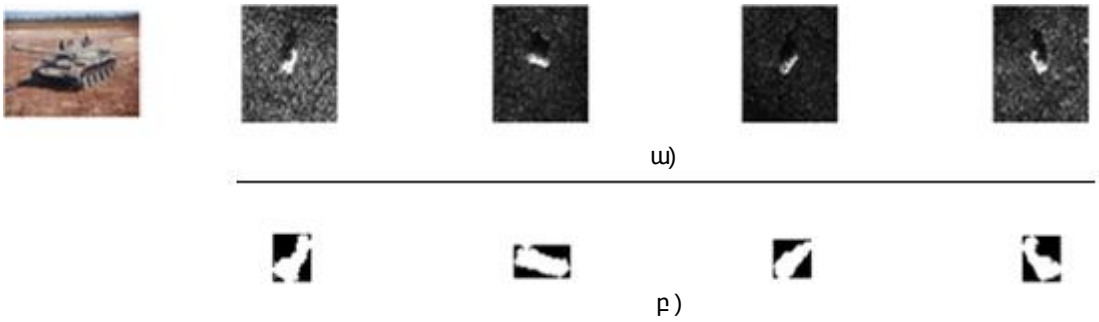
ա)

բ)

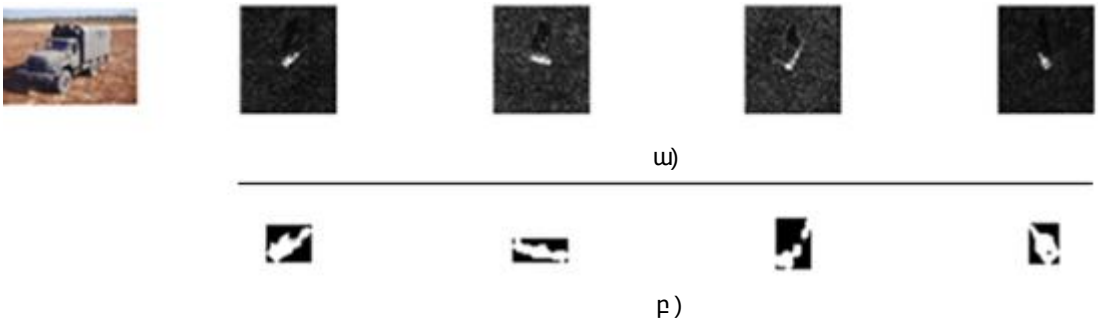
Նկար 4.10. ա) Լճի սկզբնական պատկեր, **բ)** բաղադրյալ օպերատորից և սեգմենտացիայից հետո ստացված պատկեր:

Երրորդ փորձ. Այս փորձում օգտագործվել են ծրագրի և՛ ուսուցման, և՛ ճանաչման մոդուլները: Ծրագրային համակարգի միջոցով արբանյակից ստացված երեք պատկերների համար (Նկար 4.11ա), 4.12ա) և 4.13ա) [30]) նկարագրվել են դասեր, որոնց

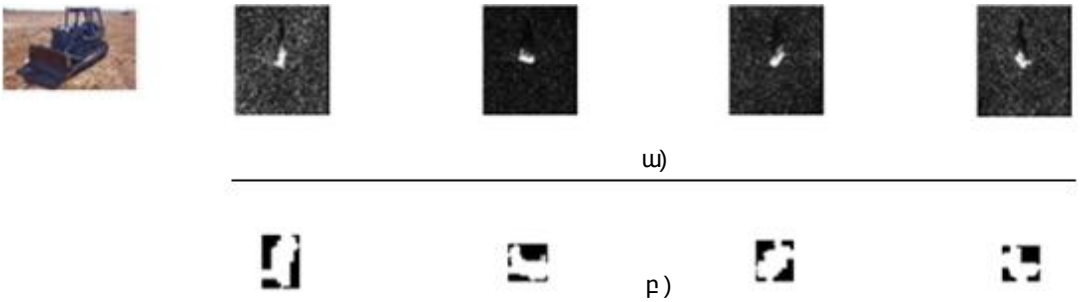
միջոցով այդ պատկերների դասին պատկանող այլ պատկերներում օբյեկտները հայտնաբերվում և ճանաչվում են համակարգի կողմից: Ինչպես արդեն նշել ենք համակարգում դասերը բնութագրվում են երկու բաղադրիչներով՝ պատկերներ և բաղադրյալ օպերատորներ: Օգտագործելով 4.11ա), 4.12ա) և 4.13ա) պատկերները գեներացնում են այդ դասերի համար ճանաչելիության լավ արդյունք ապահովող բաղադրյալ օպերատորներ: Կիրառելով այդ օպերատորները նշված պատկերների վրա ստացվում են այդ դասերը բնութագրող պատկերները 4.11բ), 4.12բ) և 4.13բ):



Նկար 4.11. ա) T62 տանկի օպտիկական, SAR և բինար պատկերներ, **բ)** տանկի դասը բնութագրող ձևանմուշ պատկերներ:



Նկար 4.12. Զիլ բեռնատարի Օպտիկական, SAR և բինար պատկերներ, **բ)** զիլ բեռնատարի դասը բնութագրող ձևանմուշ պատկերներ:



Նկար 4.13. D7 տրակտորի օպտիկական, SAR և բինար պատկերներ **բ)** D7 տրակտորի դասը բնութագրող ձևանմուշ պատկերներ:

Ստորև բերված է կատարված փորձերից մեկը՝ փորձ է արվել 4.11 նկարում բերված առաջին դիրքի տանկի 2^0-ով պտտած պատկերի ճանաչման համար (տես՝ Նկար 3.36): Համակարգը պատկերը ճանաչել է 0.84 ֆիտնես արժեքով:

Նկար 4.14. Նկար 4.11-ում բերված առաջին դիրքի տանկի 2^0-ով պտտած պատկեր:

Չորրորդ փորձ. Փորձը բաղկացած է երկու մասից: Փորձի առաջին մասում MSTAR պատկերների տվյալների բազայից [62] վերցրած արհեստական արբանյակից ստացված ռադիոլոկացիոն պատկերից առանձնացվել է մեքենա պարունակող հատված (տես՝ Նկար 4.15ա) և փորձ է արվել տվյալ տիպի մեքենաների համար գեներացնել բաղադրյալ օպերատոր, որի միջոցով համակարգը կկարողանա ճանաչել տվյալ տիպի մեքենաները: Օգտագործելով նկար 4.15ա) պատկերը, ծրագրային համակարգը գեներացրել է հետևյալ բաղադրյալ օպերատորը՝

ADD(PFIM0,FMAX2(FDIV(ADD(FLOG(FDIVC(PFIM0)), FADDC(FDIV(FMUL (PFIM5, PFIM5), ADD(PFIM13, PFIM0)))), ADD(FSUBC(FADDC(PFIM10)), PFIM0)), FLOG(FDIV(PFIM10, FSUBC(FSUBC(FSUBC(PFIM0)))))): (4.4)

Նշենք, որ բաղադրյալ օպերատորը գեներացնելու համար որպես Ground Truth պատկեր օգտագործվել է նկար 4.15բ) պատկերը:

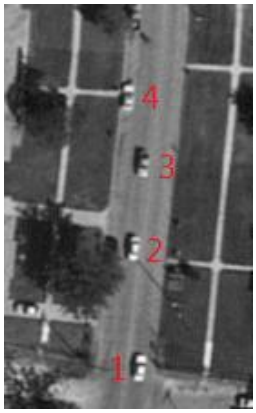
ա) բ) գ)

Նկար 4.15. ա) ավտոմեքենայի սկզբնական SAR պատկերը, **բ)** մեքենայի ground truth պատկերը, **գ)** օպերատորի հետո ստացված և սեգմենտացված պատկերը:

(4.4) բաղադրյալ օպերատորը Նկար 4.15ա) պատկերի վրա կիրառելուց և սեգմենտացիայից հետո ստացվում է Նկար 4.15գ) պատկերը: Նկար 4.16 պատկերը համակարգում հիշվում է որպես տվյալ տիպի և տվյալ դիրքի մեքենայի դաս բնութագրող ձևանմուշ պատկեր:

Նկար 4.16. նկար 4.15ա) ավտոմեքենայի դասի ձևանմուշ պատկերը համակարգում:

Նկար 4.15ա) ավտոմեքենայի դասի համար գեներացված բաղադրյալ օպերատորը կիրառվել է Նկար 4.17-ում բերված համար 1 ավտոմեքենայի պատկերի վրա (նկար 4.18 ա)-ում ներկայացված է համար 1 մեքենայի Ground Truth պատկերը) և տվյալ անկյան տակ տվյալ տիպի մեքենաների համար որպես դաս բնութագրող պատկեր է ստացվել նկար 4.18բ)-ում բերված պատկերը՝ մեծացրած չափերով:



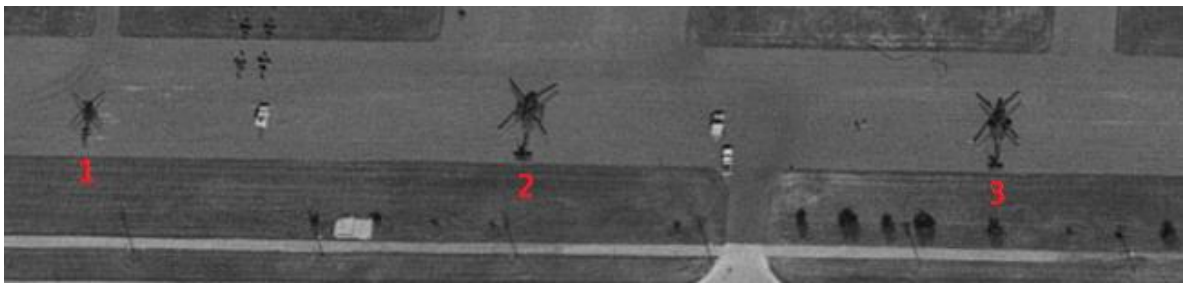
Նկար 4.17. MSTAR պատկերների տվյալների բազայից SAR պատկերի ընտրված հատված:

ա) բ)

Նկար 4.18. ա) Համար 1 մեքենայի Ground Truth պատկերը, **բ)** այս դասը բնութագրող պատկեր:

Օգտագործելով (3.11) բաղադրյալ օպերատորը և դաս բնութագրող Նկար 4.18բ) պատկերը, ծրագրային համակարգը Նկար 3.30-ի 1, 2 և 4 ավտոմեքենաները ճանաչում է համապատասխանաբար 0.95, 0.7 և 0.69 ֆիտնեսային արժեքներով 3 համարի ավտոմեքենան շփոթել է այլ օբյեկտի՝ տանկի հետ, 0.79 ֆիտնես արժեքով:

Փորձի երկրորդ մասում նկար 4.19-ում պատկերված MSTAR պատկերների տվյալների բազայից [62] վերցրած արհեստական արբանյակից ստացված պատկերում գտնվող ուղղաթիռների համար գեներացվել է բաղադրյալ օպերատոր և տվյալ դասը բնութագրող ձևանմուշ պատկեր: Վերջինս համակարգին հնարավորություն են տվել ճանաչել ու այս և նմանատիպ ուղղաթիռները:



Նկար 4.19. MSTAR պատկերների տվյալների բազայից պատկերի ուղղաթիռներ պարունակող հատված [62]:

Օգտագործելով Նկար 4.19-ում պատկերված համար 2 ուղղաթիռը գեներացվել է 0.94 ֆիտնեսային արժեքով հետևյալ բաղադրյալ օպերատորը՝

ADD(PFIM14, ADD(FADDC(FMAX2(FSQRT(PFIM12), FMIN(FMIN (FDIV (PFIM12,PFIM12))))), FSUB(PFIM7, FMEAN(FMAX2(PFIM2, PFIM14))))): (4.5)

Որպես Ground Truth օգտագործվել է նկար 4.20ա)-ում բերված պատկերը, Նկար 4.21բ)–ում պատկերված է Նկար 4.19-ի 2-րդ համարի ուղղաթիռի

վրա բաղադրյալ օպերատորը կիրառելուց և սեգմենտացիայից հետո ստացված պատկերը: Ծրագրային համակարգում որպես տվյալ տիպի ուղղաթիռների դասը բնութագրող ձևանմուշ պահվել է նկար 4.20բ)-ում բերված պատկերը:

ա)

բ)

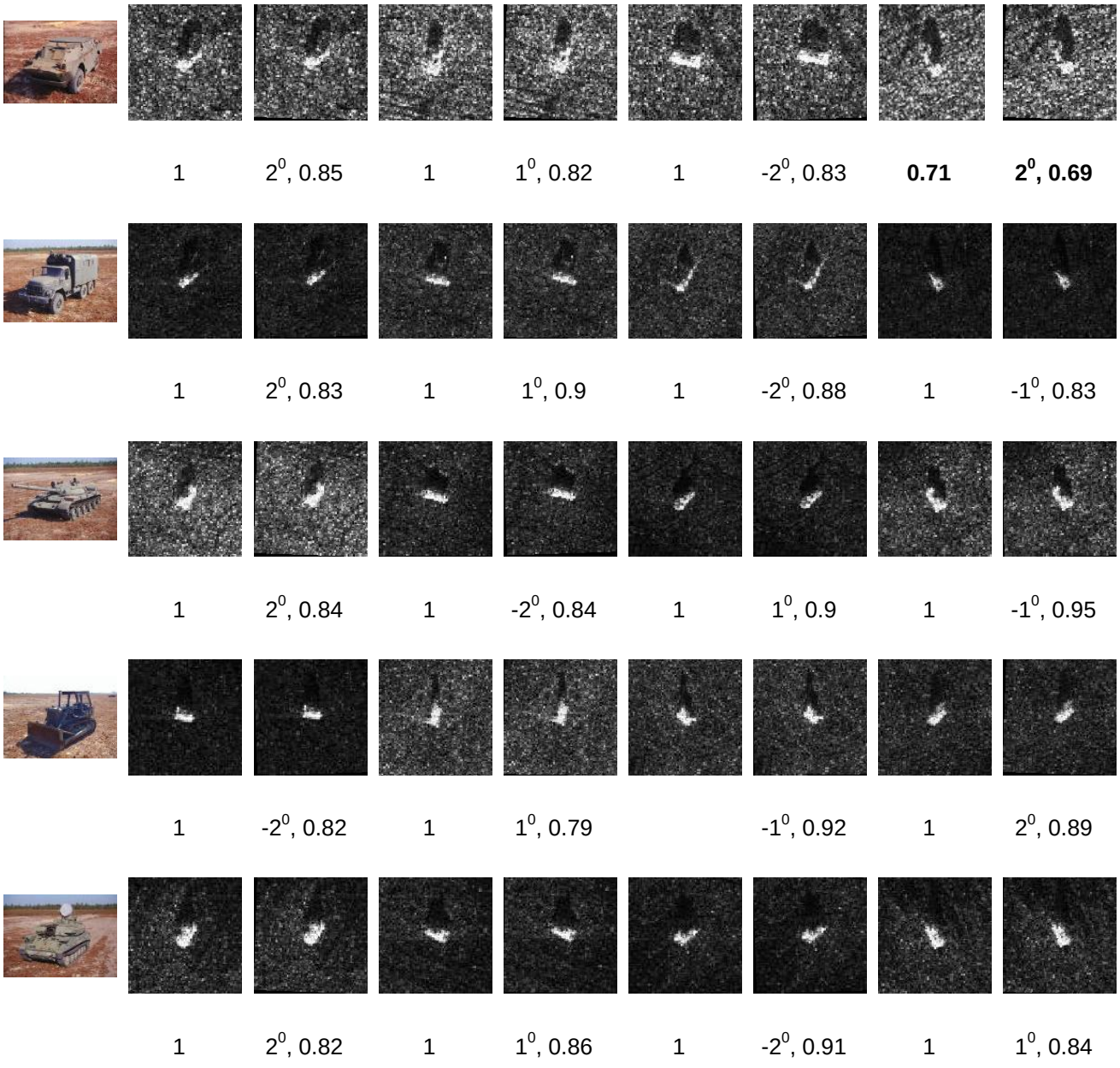
Նկար 4.20. ա) ուղղաթիռի Ground Truth պատկերը, **բ)** օպերատորի հետո ստացված և սեգմենտացված պատկերը:

Օգտագործելով (4.5) բաղադրյալ օպերատորը և դասը բնութագրող Նկար 3.42բ ձևանմուշը, ծրագրային համակարգը Նկար 4.19-ում պատկերված 1, 2 և 3 ուղղաթիռները ճանաչում է համապատասխանաբար 0.73, 0.94 և 0.73 ֆիտնեսային արժեքներով:

Հինգերորդ փորձ. Փորձերը իրականացվել են արհեստական արբանյակից ստացված հինգ տարբեր տիպի օբյեկտ ընգրկող (տես՝ Նկար 4.21) չորս տարբեր անկյան տակ նկարահանված ինֆրակարմիր պատկերների վրա [30]: Ծրագրային համակարգը ուսուցանվել է այդ հինգ տիպի օբյեկտները ճանաչելու համար: Այսինքն՝ այդ հինգ տիպի դասերի համար գտնվել են բաղադրյալ օպերատորներն ու դասը բնութագրող ձևանմուշ պատկերները և ներառվել են համակարգում: Փորձ է արվել ճանաչել նշված պատկերները և նրանցից ստացված -2^0 - $+2^0$ պտտված պատկերները (տես՝ նկար 4.21): Նկար 4.21-ում ցույց է տրված այդ պատկերների պտտման անկյունները և ճանաչման ֆիտնեսային արժեքները: Նկատենք, որ համակարգը առաջին օբյեկտի 4-րդ դիրքի պատկերը շփոթել է այլ օբյեկտի հետ:

Փորձի հաջորդ փուլում համակարգի տվյալների բազայում ավելացվել են նախորդ փորձի ընթացքում ստացված ավտոմեքենայի և ուղղաթիռի ճանաչման համար ստացված բաղադրյալ օպերատորները և այդ դասերը բնութագրող ձևանմուշ պատկերները, որպեսզի համակարգը կարողանա ճանաչել յոթ տիպի օբյեկտներ տարբեր դիրքերով: Փորձի արդյունքում ստացվել է, որ համակարգը տվյալ

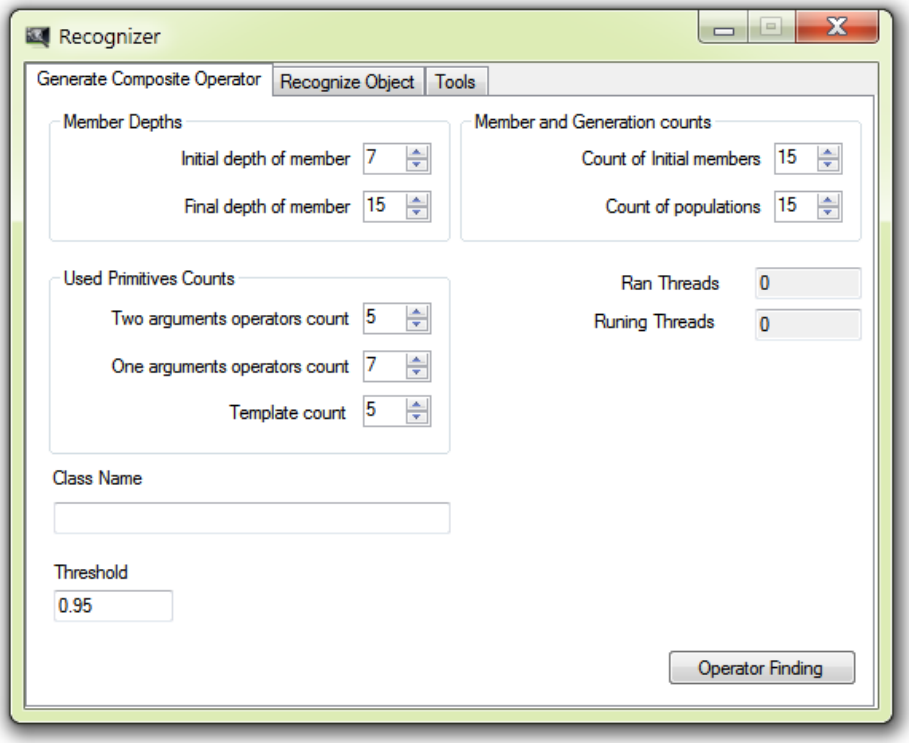
օբյեկտները ճանաչում է նույն ֆիտնեսային արժեքներով, որոնք ստացվել էին նաև առանձին առանձին ճանաչման փորձերի ժամանակ: Այս արդյունքը ապացուցում է այն, որ համակարգի աշխատանքը կախված չէ օբյեկտների դասերի (դասերում օբյեկտների դիրքերի) քանակից:



Նկար 4.21. Հինգ տիպի օբյեկտների օպտիկական և SAR պատկերները, նրանց պատման անկյունները և ճանաչման ֆիտնեսային արժեքները:

4.2 Ծրագրային համակարգի օգտագործողի ուղեցույցը

Ծրագրային համակարգը մշակված է C# լեզվով .Net միգավայրում և աշխատում է windows օպերացիոն համակարգում .Net framework-4.5 և բարձր վերսիաների առկայության դեպքում: Ծրագրային համակարգը օգտագործողի ինտերֆայսը (User interface) բաղկացած է մեկ պատուհանից, որը բերված է Նկար 4.22-ում: Այն բաղկացած է երեք ենթաբաժիններից՝ Generate Composite Operator, Recognize Object և Tools.



Նկար. 4.22. Ծրագրային համակարգի պատուհանը, երբ ակտիվ է առաջին

Generate Composite Operator բաժինը պատասխանատու է դասի համար բաղադրյալ օպերատոր գեներացնելու համար: Այս բաժնում օգտագործողին հնարավորություն է տրվում փոփոխել գենետիկ ծրագրավորման աշխատանքը կառավարող պարամետրերը: Գենետիկ ծրագրավորման պարամետրերի փոփոխման ղեկավարիչները (controls) դասավորված են ըստխմբերի.

Member depths. Այս խմբում ղեկավարվում է գենետիկ ծրագրավորման անհատների՝ ծառերի, խորությունները, այսինքն, նախանշվում է այն ամենամեծ խորությունը անահատիների համար, որից մեծ չպետք է լինի յուրաքանչյուր անհատի խորությունը:

Initial depth of member դաշտում տրվում է գենետիկ ծրագրավորման պատահականորեն գեներացված սկզբնական սերնդի անհատի խորությունը:

Final depth of member դաշտում տրվում է անհատի մաքսիմալ խորությունը:

Member and generation counts. Այս խմբում ղեկավարվում են անհատների քանակը սերնդում և սերունդների քանակը:

Count of initial memberes դաշտում տրվում է գենետիկ ծրագրավորման յուրաքանչյուր սերնդում անհատների քանակը:

Count of generationes դաշտում տրվում է գենետիկ ծրագրավորման սերունդների քանակը: ԳԾ-ի ավարտը նախանշող պարամետր:

Used primitive counts. Այս խմբում ղեկավարվում է տվյալ փորձի ընթացքում օգտագործվող տարրական պատկերների և տարրական օպերատորների տիպերի քանակը:

Two arguments operators count դաշտում տրվում է օգտագործվելիք երկու արգումենտ ունեցող տարրական օպերատորների քանակը:

One arguments operators count դաշտում տրվում է օգտագործվելիք մեկ արգումենտ ունեցող տարրական օպերատորների քանակը:

Template count դաշտում տրվում է օգտագործվելիք տարրական պատկերների քանակը:

Finding Operator կոճակով սկսվում է փորձը: Մուտքին պետք է տալ թրենինգի պատկերը և այդ պատկերին համապատասխան Ground Truth պատկերը:

Ծրագիրը հնարավորություն է ընձեռնում միաժամանակ կատարել մեկից ավելի փորձեր բաղադրյալ օպերատորներ գտնելու համար, քանի որ յուրաքանչյուր փորձում մեր պայմանների բավարարող բաղադրյալ օպերատոր գտնելը երաշխավորված չ1, ապա միաժամանակ մի քանի փորձ իրականացնելով մեր նախանշած պայմաններին բավարարող բաղադրյալ օպերատոր գտնելու ժամանակը կարող է զգալի կրճատվել: Միաժամանակ մի քանի փորձ կատարելու գաղափարը իրականացվել է օգտագործելով *multy-threading* (բազմահոսքային) ծրագրավորման հնարավորությունները:

Ran Threads դաշտը ցույց է տալիս թողարկված փորձերի քանակը, իսկ *Running Threads* դաշտը՝ տվյալ պահին աշխատանքի մեջ գտնվող Thread-երի (կատարվող փորձերի) քանակը:

Class Name դաշտը նախատեսված է գտնվող օպերատորը և հետևաբար դասը անվանելու համար:

Threshold-ով փոխում ենք ֆիտնես ֆունկցիայի շեմային արժեքը:

Recognize Object ենթաբաժինը պատասխանատու է օբյեկտների ճանաչման համար (տես՝ Նկար 4.23):

Class name. Ճանաչում գործողությունը ավարտելուց հետո ծրագիրը այս դաշտում գրում է այն դասի անունը, որին որ պատկանում է ճանաչված օբյեկտը:

Fitness. Ճանաչում գործողությունը ավարտելուց հետո ծրագիրը այս դաշտում ցույց է տալիս այն ֆիտնեսային արժեքը, որով ճանաչվել է օբյեկտը:

Class image. Սա այն ձևանմուշ պատկերն է, որի համար ամենամեծ ֆիտնեսային արժեքն է ստացվել, այսինքն, ճանաչված օբեկտիվ ձևանմուշ պատկերն է:

Clasess. Այս ցուցակում բերվում են համակարգի կողմից ճանաչվող օբյեկտները, և Ճանաչում գործողությունը ավարտելուց հետո նշվում է ճանաչված դասի անունը:

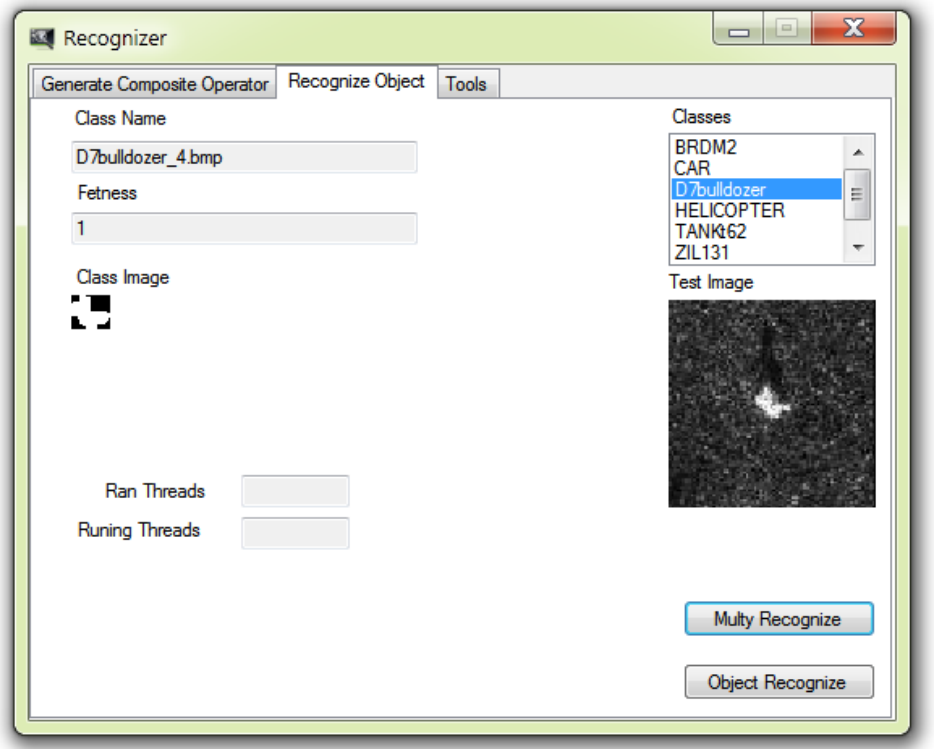
Test Image. Այստեղ պատկերվում է մուտքին տրվող ճանաչման պատկերը:

Համակարգը ճանաչում գործողությունը սկսում է երկու կոճակներով՝ *Multy Recognize* և *Object Recognize*: Ճանաչման համար համակարգին պետք է տալ Ճանաչման պատկերը:

Multy Recognize ճանաչման դեպքում համակարգը օբյեկտի ճանաչումը սկսում է յուրաքանչյուր իր կողմից ճանաչվող դասի համար առանձին և զուգահեռաբար: Յուրաքանչյուր դասի համար մեկնարկում է առանձին հոսք (thread), և յուրաքանչյուր դասի համար ճանաչումը կատարվում է առանձին: Այսինքն՝ տվալ դասը բնութագրող բաղադրյալ օպերատորը(ները) կիրառվում է ճանաչվող պատկերի վրա, այնուհետև նրանից առանձնացվում է հայտնաբերված

օբյեկտը, այնուհետ հայտնաբերված օբյեկտը համեմատվում է տվյալ դասը բնութագրող ձևանմուշ պատկերների հետ(հաշվում են ֆիտնեսները): Համեմատությունից ստացված լավագույն ֆիտնես արժեքը thread-ը վերադարձնում է որպես ճանաչման լավագույն արդյունքը: Համակարգի ճանաչման արդյունքը լինում է բոլոր thread-երի վերադարձրած արդյունքներից լավագույնը: *Ran Threads* դաշտը ցույց է տալիս մեկնարկած thread-երի քանակը: *Running Threads* դաշտը ցույց է տալիս տվյալ պահին աշխատանքի մեջ գտնվող thread-երի քանակը:

Object Recognize ճանաչման դեպքում Ճանաչման պատկերի վրա կիրառվում են բոլոր բաղադրյալ օպերատորները և արդյունքները համեմատվում են բոլոր դասերի ձևանմուշ պատկերների հետ: Լավագույն համընկնումը համարվում է ճանաչման արդյունքը:



Նկար 4.23. Ծրագրային համակարգի երկրորդ ենթաբաժնի

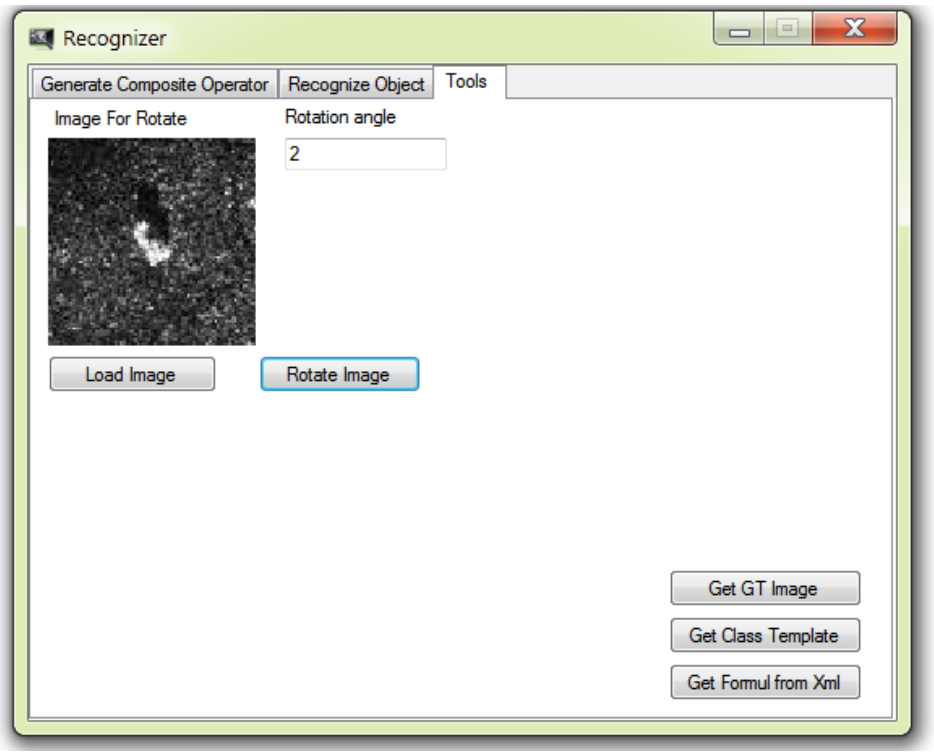
Tools ենթաբաժնում մշակված են մի քանի գործառույթներ, որոնք հեշտացնում են ծրագրային համարի շահագործումը (տես՝ Նկար 4.24):

Rotate Image կոճակի օգնությամբ պտտում ենք *Load Image* կոճակի օգնությամբ բեռնված պատկերը *Rotation angle* դաշտում տրված անկյան չափով:

Get GT Image կոճակի օգնությամբ, բեռնված բինար պատկերի, ձեռքով ստացված *Ground Truth* պատկերի և բեռնված բաղադրյալ օպերատորի կիրառմամբ ստացվում է իրական Ground Truth պատկեր:

Get Class Template կոճակի օգնությամբ բեռնված պատկերի, նրա Ground Truth պատկերի և բեռնված բաղադրյալ օպերատորի կիրառմամբ ստացվում է բեռնված պատկերի դասը բնութագրող ձևանմուշ պատկեր: *Get Formul From Xml* կոճակի օգնությամբ բաղադրյալ օպերատոր ներկայացնող Xml-ից ստացվում է մաթեմատիկական արտահայտություն (ինչպես արդեն նշել ենք, Xml-ը բնութագրում է ծառ, որի հանգույցները կամ՛ տարրական օպերատորներ, կամ՛ տարրական պատկերներ):

Get Fromul Frome Xml կոճակի օգնությամբ Xml դոկումենտի տեսք ունեցող բաղադրյալ օպերատերը վերածում ենք արտահայտության տեսք ունեցող տողի:



Նկար 4.24. Ծրագրային համակարգի գործիքների ենթաբանժինը:

Գեներացված բաղադրյալ օպերատորները

Ուղաթիռների դասը բնութագրող բաղադրյալ օպերատորներ՝

1. FMAX2(PFIM13, FLOG(FMULC(PFIM5))):
2. ADD(PFIM14, ADD(FADDC(FMAX2(FSQRT(PFIM12), FMIN(FMIN (FDIV (PFIM12,PFIM12))))), FSUB(PFIM7, FMEAN(FMAX2(PFIM2, PFIM14))))):

ZIL131 զինվորական բեռնատարի դասը բնութագրող բաղադրյալ օպերատոր՝

ADD(FMED(FDIV(FSUB(FSUB(FDIVC(FDIV(PFIM9, PFIM5)), ADD(ADD(PFIM7, PFIM9), FDIVC(PFIM14))),FDIVC(FMUL(FDIVC(PFIM7),FDIVC(PFIM5)))),FADDC(FLOG(ADD(FADDC(PFIM5), ADD(PFIM7, PFIM5)))))), FDIV(FMAX(FDIVC(FDIVC(ADD(FADDC(PFIM14), FDIV(PFIM5,PFIM7)))),FDIVC(ADD(FSUB(FMED(FMED(PFIM14)),FLOG(FMAX(PFIM14))), FDIVC(FMAX(FDIV(PFIM14, PFIM14))))))) :

T62 տանկի դասը բնութագրող բաղադրյալ օպերատոր՝

ADD (ADD (MAX(DIV (ADD (MAX2 (MAX2 (PFII7, PFI13), DIVC (PFI7)),ADD (DIVC (PFI0), ADD(PFI0, PFI13))), SQRT (MAX2 (SQRT (PFI13), ADD(PFI9, PFI13)))), DIVC (MAX (MAX2 (DIV (SQRT (PFI13), DIV (PFI13,PFI0)), SQRT (DIVC (PFI7)))), MAX2 (MAX2 (ADD (DIV (DIV (PFI13,PFI13), SQRT (PFI9)), ADD (MEAN (PFI9), MAX2 (PFI9, PFI7))), DIVC(DIVC (DIVC (PFI13)))), MEAN (MAX2 (MAX2 (DIV (PFI13, PFI7), DIVC(PFI0)), MAX (DIVC (PFI13)), SQRT (DIVC (DIVC (PFI0))))))):

ZSU հակաօդային պաշտպանության մեքենայի դասը բնութագրող բաղադրյալ օպերատոր՝

FSUB(FMUL(FLOG(FMED(FLOG(FMAX(FMUL(FMAX(PFIM13), FSUB(PFIM13, PFIM0)))))), FMED(FLOG(FMAX(FDIV(PFIM7, FMAX(FMUL(PFIM7, PFIM13))))), FLOG(PFIM13)):

BRDM2 զրահամեքենայի դասը բնութագրող բաղադրյալ օպերատոր՝

FSUB(FSUB(FMIN(FSQRT(FMED(FSUBC(ADD(PFIM3,PFIM13)))),FSQRT(FSUBC(FSQRT(FSQRT(ADD(PFIM3, PFIM13)))),

FSQRT(FSQRT(ADD(FSQRT(FSQRT(FSUBC(PFIM11))),
FMAX2(FMED(FSUBC(PFIM3)), FSQRT(FMIN(PFIM13))))))):

D7 տրակտորի դասը բնութագրող բաղադրյալ օպերատոր՝

FMAX2(FMIN(FMAX(FMAX2(FDIV(FMAX(FMAX(PFIM5)), FMAX(FMUL(PFIM9, PFIM9))), FDIVC(FMUL(FMED(PFIM5),FDIV(PFIM9,PFIM5))))),FDIV(FMUL(FDIVC(FMUL(FMAX(FMIN(PFIM9)), FMAX(FMUL(PFIM9, PFIM9)))), FDIV(FMIN(FMAX(FMIN(PFIM9))), FDIVC(FMAX(FMUL(PFIM9, PFIM9))))), FMAX(FMIN(FMAX(FMED(FDIV(PFIM9, PFIM9))))))):

Մարդատար ավտոմեքենայի դասը բնութագրող բաղադրյալ օպերատոր՝

ADD(PFIM0, FMAX2(FDIV(ADD(FLOG(FDIVC(PFIM0)), FADDC(FDIV(FMUL(PFIM5, PFIM5), ADD(PFIM13, PFIM0)))), ADD(FSUBC(FADDC(PFIM10)), PFIM0)), FLOG(FDIV(PFIM10, FSUBC(FSUBC(FSUBC(PFIM0))))))):

Լիճ հայտնաբերող բաղադրյալ օպերատոր՝

ADD (MED (DIV (SUB (PFI8, PFI8), SUB (PFI7, PFI9))), SUB (ADD (MED (DIV (MAX2 (PFI13, PFI13), ADD (PFI8, PFI9)))), SUBC (DIV (PFI9, PFI9)))):

Ճանապարհ հայտնաբերող բաղադրյալ օպերատոր՝

MIN(MIN(ADD(MIN2(ADDC(MUL(ADD(PFI10, PFI2), ADD(PFI10, FI2)))), MAX(LOG(MIN2(MULC(PFI10))))))):

Հիմնական արդյունքների եզրահանգումը

- Մշակվել է ռադիոլոկացիոն պատկերներում օբյեկտների հայտնաբերման և ճանաչման մեթոդ, որն անկախ է պատկերների ստացման սարքի բնութագրերից:
- Մշակվել է օբյեկտների դասերի նկարագրման մեթոդ, որը թույլ է տալիս ճանաչվող դասերի քանակի ավելացում՝ առանց ճանաչման արդյունավետության նվազման:
- Մշակվել է ռադիոլոկացիոն պատկերներում օբյեկտների ճանաչումն ապահովող նոր և արդյունավետ ծրագրային համակարգ:

Գրականություն

1 https://en.wikipedia.org/wiki/Synthetic_aperture_radar#cite_ref-2.

2 L. J. Cutrona: "Synthetic Aperture Radar": Chapter 23 (25 pp) of the McGraw Hill "Radar Handbook", 1970.

3 Конникова В.К., Лехт Е.Е., Силантьев Н.А. Практическая радиоастрономия. — М.: Издательство МГУ, 2011. — 304 с.
4 http://www.crisp.nus.edu.sg/~research/tutorial/sar_int.htm.
5 R. J. Hampo and K. A. Marko. Application of Genetic Programming to Control of Vehicle Systems. In Proceedings of the Intelligent Vehicles '92 Symposium, pages 191-195, Detroit, Mi, USA, 1992.
6 Walter Alden Tackett, Genetic Programming for Feature Discovery and Image Discrimination, University of Southern California, Dept. of EE Systems and Hughes Missile Systems Co..
7 A. Tackett. Genetic generation of “dendritic” trees for image classification. In Proceedings of WCNN93, pages IV 646–649. IEEE Press, July 1993.
8 Daniel Howard and Simon C. Roberts and Richard Brankin. Evolution of Ship Detectors for Satellite SAR Imagery. Genetic Programming, Proceedings of EuroGP'99, volume 1598, pages 135-148, Goteborg, Sweden, 1999.
9 Howard, S. C. Roberts, and C. Ryan. Pragmatic genetic programming strategy for the problem of vehicle detection in airborne reconnaissance. Pattern Recognition Letters, 27(11):1275–1288, August 2006.
10 M. Daida, J. D. Hommes, T. F. Bersano-Begey, S. J. Ross, and J. F. Vesecky. Algorithm discovery using the genetic programming paradigm: Extracting low-contrast curvilinear features from SAR images of arctic ice. In P. J. Angeline and K. E. Kinnear, Jr., editors, Advances in Genetic Programming 2, chapter 21, pages 417–442. MIT Press, Cambridge, MA, USA, 1996.
11 J. M. Daida and J. D. Hommes and S. J. Ross and J. F. Vesecky. Extracting curvilinear features from SAR images of arctic ice: Algorithm discovery using the genetic programming paradigm. Proceedings of IEEE International Geoscience and Remote Sensing, pages 673-675, Florence, Italy, 1995.
12 Yu and B. Bhanu. Evolutionary feature synthesis for facial expression recognition. Pattern Recognition Letters, 27(11):1289–1298, August 2006.
13 Jiangang Yu, Bir Bhanu, Evolutionary feature synthesis for facial expression recognition, Center for Research in Intelligent Systems, University of California, Riverside, 2006.
14 http://en.wikipedia.org/wiki/Genetic_programming.

15 Lawrence J. Fogel, David B. Fogel: A Preliminary Investigation on Extending Evolutionary Programming to Include Self-Adaptation on Finite State. Informatica (Slovenia) 18(4) (1994).
16 Nichael Lynn Cramer, A Representation for the Adaptive Generation of Simple Sequential Programs, Proceedings of an International Conference on genetic Algorithms and Their Applications, pp 183-187, 1985.
17 Koza, John R. Genetic Programming. Cambridge, MA: MIT Press, 1992.
18 Holland, John H. Adaptation in Natural and Artificial Systems. Cambridge, MA: MIT Press, 1992.
19 http://en.wikipedia.org/wiki/Fitness_landscape/.
20 Walter Alden Tackett, Recombination, Selection, and the Genetic Construction of Computer Programs, April 1994.
21 Korf, R.: Depth-First Iterative-Deepening: An Optimal Admissible Tree Search. Artificial Intelligence 27(1), 97–109 (1985).
22 Knuth, E. Donald, The Art Of Computer Programming Vol 1. 3rd ed., Boston: Addison-Wesley, 1997.
23 T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein Introduction to Algorithms. — 3rd edition. — The MIT Press, 2009.
24 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, Section 22.3: Depth-first search, pp. 540–549.
25 Jon Louis Bentley. Writing Efficient Programs. Prentice Hall. p. 11. 1982.
26 T. C. Hu, Combinatorial Algorithms, Addison Wesley, 1982.
27 Bruce Lowerre and Raj Reddy. The Harpy speech understanding system, in Trends in Speech Recognition, W.A. Lea, Ed. Prentice Hall, EngleWood Cliffs NJ. 1980.
28 A Field Guide to Genetic Programming: c Riccardo Poli, William B. Langdon, and Nicholas F. McPhee, 2008.
29 Koza, J.R.: Genetic programming: on the programming of computers by means of natural selection. Cambridge, Mass. MIT Press, London, England (1992).
30 Bir Bhanu, Yingqiang Lin, and Krzysztof Krawiec, Automatic Design AND Synthesis OF Automatic Target Recognition (ATR) Systems Using Learning Paradigms, University of California Bourns College of Engineering Center for Research in Intelligent Systems, Final Report for 23 October 1999 – 22 October 2003.

31 Bäck, Thomas, Evolutionary Algorithms in Theory and Practice, Oxford Univ. Press, p. 120, 1996.
32 http://watchmaker.uncommons.org/manual/ch03s02.html.
33 B. Langdon. The evolution of size in variable length representations. In 1998 IEEE International Conference on Evolutionary Computation, pages 633–638, Anchorage, Alaska, USA, 5-9 May 1998. IEEE Press.
34 E. Kinnear, Jr. Fitness landscapes and difficulty in genetic programming. In Proceedings of the 1994 IEEE World Conference on Computational Intelligence, volume 1, pages 142–147, Orlando, Florida, USA, 27-29 June 1994a. IEEE Press. ISBN 0-7803 1899-4.
35 J. Angeline. An investigation into the sensitivity of genetic programming to the frequency of leaf selection during subtree crossover. In J. R. Koza, et al., editors, Genetic Programming 1996: Proceedings of the First Annual Conference, pages 21–29, Stanford University, CA, USA, 28–31 July 1996. MIT Press.
36 Kanal, LN "Problem-solving models and search strategies for pattern recognition," IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-1, pp. 194-201, Apr. 1979.
37 Bandwidth Reduction Intelligent Target-Tracking / Multi-function Target Acquisition Processor (BRITT / MTAP): Final Technical Report, CDRL B0001. El Segundo, CA: Hughes Aircraft Co., May 1990.
38 Hertz, J., Krogh, A., Palmer, RG Introduction to the Theory of Neural Computation. Redwood City, CA: Addison Wesley, 1991.
39 David Rumelhart, James L. McClelland, Parallel Distributed Processing: explorations in the microstructure of cognition. MIT Press, Cambridge, 1986.
40 Goldberg, David E. Genetic Algorithms in Search, Optimization, and Machine Learning. Reading MA: Addison Wesley, 1989.
41 http://en.wikipedia.org/wiki/Posterior_probability.
42 Goldberg, D., E. Genetic Algorithms and Walsh functions: Part II, deception and its analysis. Complex Systems 3 pp 153-171, 1989.
43 http://en.wikipedia.org/wiki/Multi-objective_optimization.
44 http://en.wikipedia.org/wiki/Multilayer_perceptron.
45 R. Rojas: Neural Networks, Springer-Verlag, Berlin, 1996.
46 Mengjie Zhang and Victor Ciesielski, Genetic Programming for Multiple Class Object Detection, Department of Computer Science Royal Melbourne Institute of Technology GPO Box 2476V, Melbourne Victoria 3001, Australia.

47 Casasent, D.P., Neiberg, L.M.: Classier and Shift-invariant Automatic Target Recognition Neural networks. Neural Networks, Vol.8. 7/8 (1995) 1117-1129.
48 Roitblat, H.L., Au, W.W.L., Nachtigall, P.E., Shizumura, R., Moons, G.: Sonar Recognition of Targets Embedded in Sediment. Neural Networks, Vol. 8. 7-8 (1995) 1263-1273.
49 Gader, P.D., Miramonti, J.R., Won Y., Co Æeld P.: Segmentation free shared weight neural networks for automatic vehicle detection. Neural Networks, Vol.8. 9 (1995) 1457-1473.
50 Koza, J.R.: Genetic programming: on the programming of computers by means of natural selection. Cambridge, Mass. MIT Press, London, England (1992).
51 Winkeler, J.F., Manjunath, B.S.: Genetic Programming for Ob ject Detection. In: Koza, J.R., Deb, K., Dorigo, M., Fogel, D.B., Garzon, M., Iba, H., Riolo, R.L. (eds.): Genetic Programming 1997: Proceedings of the Second Annual Conference, Morgan Kaufmann, Stanford University, (1997) 330-335.
52 Zhang, M.: A pixel-based approach to recognising small objects in large pictures using neural networks. In: Proceedings of the Annual RMIT Computer Science Post-graduate Students' Conference, Department of computer science, RMIT. TR 97-51, Melbourne (1997) 57-68.
53 Zhang, M., Ciesielski, V.: Centred Weight Initialization in Neural Networks for Object Detection. In: Proceedings of the Twenty Second Australasian Computer Science Conference, Auckland (1999).
54 R. Poli, "Genetic programming for feature detection and image segmentation," in Evolutionary Computation, T.C. Forgarty (Ed.), pp. 110-125, 1996.
55 B. Bhanu and Y. Lin, "Learning composite operators for object detection," Proc. Genetic and Evolutionary Computation Conference, pp. 1003-1010, July 2002.
56 Yingqiang Lin and Bir Bhanu, Learning Features for Object Recognition, Center for Research in Intelligent Systems University of California, Riverside, CA, 92521, USA.
57 S. Theodoridis and K. Koutroumbas, Pattern Recognition, Academic Press, 1999.
58 Bäck, Thomas, Evolutionary Algorithms in Theory and Practice, Oxford Univ. Press, p. 120, 1996.
59 http://tech-algorithm.com/articles/bilinear-image-scaling/.
60 https://msdn.microsoft.com/en-us/library/0xy59wtx(v=vs.110).aspx.
61 http://ru.wikipedia.org/wiki/XML.
62 https://www.sdms.afrl.af.mil/index.php?collection=mstar&page=targets.

63 http://cswww.essex.ac.uk/staff/poli/papers/Poli-ECJ1998.pdf.

64 Riccardo Poli Peter Nordin William B. Langdon Terence C. Fogarty, Genetic Programming: Second European Workshop, EuroGP'99 Goteborg, Sweden, May 26-27, 1999.