Мета роботи

Ознайомитися з бібліотекою ECJ. Виконати індивідуальне завдання, використовуючи для розв'язання задачі бібліотеку ECJ.

Індивідуальне завдання

В якості завдання я вибрав дуже просту задачу – задачу газонокосарки ([1] Коза 1994). Повні описи цієї проблеми можна знайти за посиланнями [2][3]. Суть задачі полягає у тому, щоб знайти програму для газонокосарки, яка б направляла її по всій галявині. Газонокосарка може виконувати наступні інструкції:

- Уперед: перехід на 1 крок вперед і косити
- Поворот вліво: повернутися вліво на 90 градусів.
- Стрибок: перейти на позицію у полі без косіння трави.

Галявина – прямокутна ділянка розміром (n x m), і має тороїдальну форму (якщо газонокосарка вийде за межі поля, зв'явиться знову з боку). Газонокосарка мусить зкосити кожну клітинку поля, також вона може зкосити двічі одну клітинку, але це не дає ефекту.

Виконання індивідуального завдання

Першим кроком для реалізації вирішення задачі – це визначити набір команд, терміналів і функцій. Пропонується використовувати ті ж самі, за книжкою [3] Коза.

- Термінали:
    1. Зліва (*GPNode*): Повернути косарку наліво
    2. Уперед (*GPNode*): Перехід на один крок вперед з косінням
    3. Значення (*GPData*): Випадкова константа, містить вектор двох цілих чисел
- Функції:
    1. Сума: векторна сума двох цілочисельних векторів
    2. Стрибок: Перейти на нову позицію на галявині, де відносна відстань зазначена векторним аргументом.
    3. Програма: Виконує дві гілки послідовно і повертає результат останньої гілки.

Перше, реалізуємо дані (термінали, п.3):

```
[ECConfiguration("ec.app.lawnmower.LawnmowerData")]
public class LawnmowerData : GPData
{
    // return value
    public int x;
    public int y;
```

```
    public override void CopyTo(GPData gpd)
    {
        var d = (LawnmowerData)gpd;
        d.x = x;
        d.y = y;
    }
}
```

Стрибок:

```
[ECConfiguration("ec.app.lawnmower.func.Frog")]
public class Frog : GPNode
{
    public override string ToString() { return "frog"; }

    public override void CheckConstraints(IEvolutionState state,
        int tree,
        GPIndividual typicalIndividual,
        IParameter individualBase)
    {
        base.CheckConstraints(state, tree, typicalIndividual, individualBase);
        if (Children.Length != 1)
            state.Output.Error("Incorrect number of children for node " +
                ToStringForError() + " at " +
                individualBase);
    }

    public override void Eval(IEvolutionState state,
        int thread,
        GPData input,
        ADFStack stack,
        GPIndividual individual,
        IProblem problem)
    {
        var p = (Lawnmower)problem;
        var d = (LawnmowerData)input;

        Children[0].Eval(state, thread, input, stack, individual, problem);

        switch (p.Orientation)
        {
            case Lawnmower.O_UP:
                // counter-clockwise rotation
                p.PosX -= d.y;
                p.PosY += d.x;
                break;
            case Lawnmower.O_LEFT:
                // flipped orientation
                p.PosX -= d.x;
                p.PosY -= d.y;
                break;
            case Lawnmower.O_DOWN:
                // clockwise rotation
                p.PosX += d.y;
                p.PosY -= d.x;
                break;
            case Lawnmower.O_RIGHT:
                // proper orientation
                p.PosX += d.x;
                p.PosY += d.y;
                break;
```

```
        }

        p.PosX = ((p.PosX % p.MaxX) + p.MaxX) % p.MaxX;
        p.PosY = ((p.PosY % p.MaxY) + p.MaxY) % p.MaxY;

        p.Moves++;
        if (p.Map[p.PosX][p.PosY] == Lawnmower.UNMOWED)
        {
            p.Sum++;
            p.Map[p.PosX][p.PosY] = p.Moves;
        }
    }
}
```

Поворот вліво:

```
[ECConfiguration("ec.app.lawnmower.func.Left")]
public class Left : GPNode
{
    public override string ToString() { return "left"; }

    public override void CheckConstraints(IEvolutionState state,
        int tree,
        GPIndividual typicalIndividual,
        IParameter individualBase)
    {
        base.CheckConstraints(state, tree, typicalIndividual, individualBase);
        if (Children.Length != 0)
            state.Output.Error("Incorrect number of children for node " +
                ToStringForError() + " at " +
                individualBase);
    }

    public override void Eval(IEvolutionState state,
        int thread,
        GPData input,
        ADFStack stack,
        GPIndividual individual,
        IProblem problem)
    {
        var p = (Lawnmower)problem;
        var d = (LawnmowerData)input;

        switch (p.Orientation)
        {
            case Lawnmower.O_UP:
                p.Orientation = Lawnmower.O_LEFT;
                break;
            case Lawnmower.O_LEFT:
                p.Orientation = Lawnmower.O_DOWN;
                break;
            case Lawnmower.O_DOWN:
                p.Orientation = Lawnmower.O_RIGHT;
                break;
            case Lawnmower.O_RIGHT:
                p.Orientation = Lawnmower.O_UP;
                break;
        }
        d.x = 0;
        d.y = 0;
    }
}
```

Косіння:

```
[ECConfiguration("ec.app.lawnmower.func.Mow")]
public class Mow : GPNode
{
    public override string ToString() { return "mow"; }

    public override void CheckConstraints(IEvolutionState state,
        int tree,
        GPIndividual typicalIndividual,
        IParameter individualBase)
    {
        base.CheckConstraints(state, tree, typicalIndividual, individualBase);
        if (Children.Length != 0)
            state.Output.Error("Incorrect number of children for node " +
                ToStringForError() + " at " +
                individualBase);
    }

    public override void Eval(IEvolutionState state,
        int thread,
        GPData input,
        ADFStack stack,
        GPIndividual individual,
        IProblem problem)
    {
        var p = (Lawnmower)problem;
        var d = (LawnmowerData)input;

        switch (p.Orientation)
        {
            case Lawnmower.O_UP:
                p.PosY--;
                if (p.PosY < 0) p.PosY = p.MaxY - 1;
                break;
            case Lawnmower.O_LEFT:
                p.PosX--;
                if (p.PosX < 0) p.PosX = p.MaxX - 1;
                break;
            case Lawnmower.O_DOWN:
                p.PosY++;
                if (p.PosY >= p.MaxY) p.PosY = 0;
                break;
            case Lawnmower.O_RIGHT:
                p.PosX++;
                if (p.PosX >= p.MaxX) p.PosX = 0;
                break;
            default:  // whoa!
                state.Output.Fatal("Whoa, somehow I got a bad orientation! (" +
p.Orientation + ")");
                break;
        }

        p.Moves++;
        if (p.Map[p.PosX][p.PosY] == Lawnmower.UNMOWED)
        {
            p.Sum++;
            p.Map[p.PosX][p.PosY] = p.Moves;
        }

        // return [0,0]
```

```
        d.x = 0;
        d.y = 0;
    }
}
```

Обробка виходу за поле:

```
[ECConfiguration("ec.app.lawnmower.func.V8a")]
public class V8a : GPNode
{
    public const int MODULO = 8;  // odd that it doesn't change with map size

    public override string ToString() { return "v8a"; }

    public override void CheckConstraints(IEvolutionState state,
        int tree,
        GPIndividual typicalIndividual,
        IParameter individualBase)
    {
        base.CheckConstraints(state, tree, typicalIndividual, individualBase);
        if (Children.Length != 2)
            state.Output.Error("Incorrect number of children for node " +
                ToStringForError() + " at " +
                individualBase);
    }

    public override void Eval(IEvolutionState state,
        int thread,
        GPData input,
        ADFStack stack,
        GPIndividual individual,
        IProblem problem)
    {
        int resultx;
        int resulty;

        var rd = ((LawnmowerData)(input));

        Children[0].Eval(state, thread, input, stack, individual, problem);
        resultx = rd.x;
        resulty = rd.y;

        Children[1].Eval(state, thread, input, stack, individual, problem);
        rd.x = (resultx + rd.x) % MODULO;
        rd.y = (resulty + rd.y) % MODULO;
    }
}
```

Програма для послідовного виконання функцій:

```
[ECConfiguration("ec.app.lawnmower.func.Progn2")]
public class Progn2 : GPNode
{
    public override string ToString() { return "progn2"; }

    public override void CheckConstraints(IEvolutionState state,
        int tree,
        GPIndividual typicalIndividual,
        IParameter individualBase)
    {
        base.CheckConstraints(state, tree, typicalIndividual, individualBase);
```

```
        if (Children.Length != 2)
            state.Output.Error("Incorrect number of children for node " +
                ToStringForError() + " at " +
                individualBase);
    }

    public override void Eval(IEvolutionState state,
        int thread,
        GPData input,
        ADFStack stack,
        GPIndividual individual,
        IProblem problem)
    {
        // Evaluate both children.  Return the second one (done automagically).
        Children[0].Eval(state, thread, input, stack, individual, problem);
        Children[1].Eval(state, thread, input, stack, individual, problem);
    }
}
```

Модель поля задається як ERC [4] вузол:

```
[ECConfiguration("ec.app.lawnmower.func.LawnERC")]
public class LawnERC : ERC
{
    public int maxx;
    public int maxy;

    public int x;
    public int y;

    public override void Setup(IEvolutionState state, IParameter paramBase)
    {
        base.Setup(state, paramBase);
        // figure the coordinate base -- this will break if the underlying
        // base changes, oops
        var newbase = new
Parameter(EvolutionState.P_EVALUATOR).Push(Evaluator.P_PROBLEM);

        // obviously not using the default base for any of this stuff

        // load our map coordinates
        maxx = state.Parameters.GetInt(newbase.Push(Lawnmower.P_X), null, 1);
        if (maxx == 0)
            state.Output.Error("The width (x dimension) of the lawn must be >0",
                newbase.Push(Lawnmower.P_X));
        maxy = state.Parameters.GetInt(newbase.Push(Lawnmower.P_Y), null, 1);
        if (maxy == 0)
            state.Output.Error("The length (y dimension) of the lawn must be >0",
                newbase.Push(Lawnmower.P_X));
        state.Output.ExitIfErrors();
    }

    public override void ResetNode(IEvolutionState state, int thread)
    {
        x = state.Random[thread].NextInt(maxx);
        y = state.Random[thread].NextInt(maxy);
    }

    public override int NodeHashCode()
    {
        // a reasonable hash code
        return GetType().GetHashCode() + x * maxy + y;
```

```
    }

    public override bool NodeEquals(GPNode node)
    {
        // check first to see if we're the same kind of ERC --
        // won't work for subclasses; in that case you'll need
        // to change this to isAssignableTo(...)
        if (GetType() != node.GetType()) return false;
        // now check to see if the ERCs hold the same value
        var n = (LawnERC)node;
        return (n.x == x && n.y == y);
    }

    public override void ReadNode(IEvolutionState state, BinaryReader dataInput) //
throws IOException
    {
        x = dataInput.ReadInt32();
        y = dataInput.ReadInt32();
    }

    public override void WriteNode(IEvolutionState state, BinaryWriter dataOutput) //
throws IOException
    {
        dataOutput.Write(x);
        dataOutput.Write(y);
    }

    public override string Encode()
    { return Code.Encode(x) + Code.Encode(y); }

    public override bool Decode(DecodeReturn dret)
    {
        // store the position and the string in case they
        // get modified by Code.java
        int pos = dret.Pos;
        String data = dret.Data;

        // decode
        Code.Decode(dret);

        if (dret.Type != DecodeReturn.T_INT) // uh oh!
        {
            // restore the position and the string; it was an error
            dret.Data = data;
            dret.Pos = pos;
            return false;
        }

        // store the data
        x = (int)(dret.L);

        // decode
        Code.Decode(dret);

        if (dret.Type != DecodeReturn.T_INT) // uh oh!
        {
            // restore the position and the string; it was an error
            dret.Data = data;
            dret.Pos = pos;
            return false;
        }
```

```
        // store the data
        y = (int)(dret.L);

        return true;
    }

    public override string ToStringForHumans()
    { return "[" + x + "," + y + "]"; }

    public override void Eval(IEvolutionState state,
        int thread,
        GPData input,
        ADFStack stack,
        GPIndividual individual,
        IProblem problem)
    {
        var rd = ((LawnmowerData)(input));
        rd.x = x;
        rd.y = y;
    }
}
```

Для виконання програми необхідно описати задачу у файлі ***.params***:

```
gp.fs.size = 1
gp.fs.0.name = f0
gp.fs.0.size = 6
gp.fs.0.func.0 = ec.app.lawnmower.func.LawnERC
gp.fs.0.func.0.nc = nc0
gp.fs.0.func.1 = ec.app.lawnmower.func.Left
gp.fs.0.func.1.nc = nc0
gp.fs.0.func.2 = ec.app.lawnmower.func.Mow
gp.fs.0.func.2.nc = nc0
gp.fs.0.func.3 = ec.app.lawnmower.func.V8a
gp.fs.0.func.3.nc = nc2
gp.fs.0.func.4 = ec.app.lawnmower.func.Progn2
gp.fs.0.func.4.nc = nc2
gp.fs.0.func.5 = ec.app.lawnmower.func.Frog
gp.fs.0.func.5.nc = nc1
eval.problem = ec.app.lawnmower.Lawnmower
eval.problem.data = ec.app.lawnmower.LawnmowerData
eval.problem.x = 8
eval.problem.y = 8
```

У файлі задали термінали і фукнції, а також розміри задачі. До цього необхідно описати задачу генетичного програмування – тип фітнес-функції, тип кросоверу, мутації і репродукції, їхні ймовірності, тип селекції, її параметри (*нище наведені лише* ***найважливіші параметри****, додатково задаються обмеження, методи ведення статистики, стоп-умова та інше*):

```
pop.subpop.0.species.fitness = ec.gp.koza.KozaFitness
pop.subpop.0.species.pipe.num-sources = 2
pop.subpop.0.species.pipe.source.0 = ec.gp.koza.CrossoverPipeline
pop.subpop.0.species.pipe.source.0.prob = 0.9
pop.subpop.0.species.pipe.source.1 = ec.breed.ReproductionPipeline
pop.subpop.0.species.pipe.source.1.prob = 0.1
```

```
gp.koza.xover.source.0 = ec.select.TournamentSelection
gp.koza.xover.source.1 = same
gp.koza.xover.ns.0 = ec.gp.koza.KozaNodeSelector
gp.koza.xover.ns.1 = same
gp.koza.xover.maxdepth = 17
gp.koza.xover.tries = 1
gp.koza.mutate.source.0 = ec.select.TournamentSelection
gp.koza.mutate.ns.0 = ec.gp.koza.KozaNodeSelector
gp.koza.mutate.build.0 = ec.gp.koza.GrowBuilder
```

Виконання програми

```
Threads:  breed/1 eval/1
Seed: -164770117
Job: 0
Setting up
Processing GP Types
Processing GP Node Constraints
Processing GP Function Sets
Processing GP Tree Constraints
Initializing Generation 0
Subpop 0 best fitness of generation: Fitness: Standardized=56 Adjusted=0.01754386 Hits=8
Generation 1
Subpop 0 best fitness of generation: Fitness: Standardized=56 Adjusted=0.01754386 Hits=8
Generation 2
Subpop 0 best fitness of generation: Fitness: Standardized=54 Adjusted=0.01818182 Hits=10
Generation 3
Subpop 0 best fitness of generation: Fitness: Standardized=48 Adjusted=0.02040816 Hits=16
Generation 4
Subpop 0 best fitness of generation: Fitness: Standardized=48 Adjusted=0.02040816 Hits=16
Generation 5
Subpop 0 best fitness of generation: Fitness: Standardized=41 Adjusted=0.02380952 Hits=23
Generation 6
Subpop 0 best fitness of generation: Fitness: Standardized=34 Adjusted=0.02857143 Hits=30
Generation 7
Subpop 0 best fitness of generation: Fitness: Standardized=26 Adjusted=0.03703704 Hits=38
Generation 8
Subpop 0 best fitness of generation: Fitness: Standardized=18 Adjusted=0.05263158 Hits=46
Generation 9
Subpop 0 best fitness of generation: Fitness: Standardized=12 Adjusted=0.07692308 Hits=52
Generation 10
Subpop 0 best fitness of generation: Fitness: Standardized=3 Adjusted=0.25 Hits=61
Generation 11
Subpop 0 best fitness of generation: Fitness: Standardized=1 Adjusted=0.5 Hits=63
Generation 12
Subpop 0 best fitness of generation: Fitness: Standardized=1 Adjusted=0.5 Hits=63
Generation 13
Subpop 0 best fitness of generation: Fitness: Standardized=0 Adjusted=1 Hits=64
Found Ideal Individual
Subpop 0 best fitness of run: Fitness: Standardized=0 Adjusted=1 Hits=64

Done!
```

Результати виконання програми

Покоління (14):

```
Generation: 0
Best Individual:
Subpopulation 0:
Evaluated: T
Fitness: Standardized=56 Adjusted=0.01754386 Hits=8
 73 123 -89 -73 -127 6 -116 109 -86 -128 -98 47 125 -53 74 -33 101 57
Equivalent Tree:
```

```
Tree 0:
 (v8a mow (ADF1 (ADF1 (v8a ADF0 left))))
Tree 1:
 mow
Tree 2:
 (progn2 (progn2 mow (progn2 (v8a ADF0 ARG0)
     [4,5])) ADF0)

Generation: 1
Best Individual:
Subpopulation 0:
Evaluated: T
Fitness: Standardized=56 Adjusted=0.01754386 Hits=8
 73 123 -89 -73 -127 6 -116 109 -86 -128 -98 47 125 -53 74 -33 101 123 -89 -73 -127
Equivalent Tree:
Tree 0:
 (v8a mow (ADF1 (ADF1 (v8a ADF0 left))))
Tree 1:
 mow
Tree 2:
 (progn2 (progn2 mow (progn2 (v8a ADF0 ARG0)
     [4,5])) ADF0)

Generation: 2
Best Individual:
Subpopulation 0:
Evaluated: T
Fitness: Standardized=54 Adjusted=0.01818182 Hits=10
 73 123 -89 -73 -127 6 -116 109 -86 -128 -98 47 125 41 -104 -5 105 97 -21 119 37 -54
63 -18 37 24 114
Equivalent Tree:
Tree 0:
 (v8a mow (ADF1 (ADF1 (v8a ADF0 left))))
Tree 1:
 mow
Tree 2:
 (progn2 (progn2 mow (progn2 (v8a (v8a left
     [3,1]) mow) (v8a mow mow))) [4,4])

Generation: 3
Best Individual:
Subpopulation 0:
Evaluated: T
Fitness: Standardized=48 Adjusted=0.02040816 Hits=16
 49 1 -117 -24 -17 -28 99 50 -95 -74 -128 -77 127 119 115 87 -13 -66 127 119 115 29 -
109
Equivalent Tree:
Tree 0:
 (v8a (v8a mow (progn2 (ADF1 left) mow)) (frog
     (v8a ADF0 (progn2 mow (ADF1 (ADF1 mow))))))
Tree 1:
 (progn2 (progn2 mow (progn2 mow left)) mow)
Tree 2:
 ADF0

Generation: 4
Best Individual:
Subpopulation 0:
Evaluated: T
Fitness: Standardized=48 Adjusted=0.02040816 Hits=16
 49 1 -117 -24 -17 -28 99 50 -95 -74 -128 -77 127 119 115 87 -13 -66 127 119 115 29 -
109 127 119 115 87 -13 -66
```

```
Equivalent Tree:
Tree 0:
 (v8a (v8a mow (progn2 (ADF1 left) mow)) (frog
     (v8a ADF0 (progn2 mow (ADF1 (ADF1 mow))))))
Tree 1:
 (progn2 (progn2 mow (progn2 mow left)) mow)
Tree 2:
 ADF0

Generation: 5
Best Individual:
Subpopulation 0:
Evaluated: T
Fitness: Standardized=41 Adjusted=0.02380952 Hits=23
 33 123 -89 -73 -73 -127 6 -116 109 -86 -128 -98 47 125 123 -89 -73 -127 6 -116 109 -
86 -128 -98 47 125 -53 74 -33 47 125 41 -104 -5 105 97 -21 119
Equivalent Tree:
Tree 0:
 (v8a mow (ADF1 (ADF1 (ADF1 (v8a ADF0 left)))))
Tree 1:
 mow
Tree 2:
 (progn2 (progn2 mow (progn2 (v8a ARG0 [3,1])
     ARG0)) (v8a (v8a ADF0 ARG0) (progn2 (progn2
     mow (progn2 (v8a ADF0 ARG0) [4,5])) (progn2
     (v8a (v8a left [3,1]) mow) (v8a mow mow)))))

Generation: 6
Best Individual:
Subpopulation 0:
Evaluated: T
Fitness: Standardized=34 Adjusted=0.02857143 Hits=30
 73 123 -89 -73 -127 6 -116 123 -89 -73 -127 6 -116 109 -86 -128 -98 47 125 -53 74 6 -
116 109 -86 -128 -98 47 125 41 -104 -5 105 109 -86 -128 -98 47 125 -53 74 -33 -127 6 -
116 109 -86 -128 -98 47 125 41 -104 -5 105 97 -21 119 37
Equivalent Tree:
Tree 0:
 (v8a mow (ADF1 (ADF1 (v8a ADF0 left))))
Tree 1:
 (v8a [3,1] (progn2 (v8a [5,0] mow) mow))
Tree 2:
 (progn2 (progn2 mow (progn2 (v8a ADF0 ARG0)
     (v8a ADF0 ARG0))) (progn2 (progn2 mow (progn2
     (v8a (v8a left [3,1]) mow) ARG0)) (progn2
     (progn2 mow (progn2 (v8a ADF0 ARG0) [4,5]))
     (v8a (v8a ADF0 ARG0) (progn2 (progn2 mow
         (progn2 (v8a (v8a left [3,1]) mow) (v8a mow
             mow))) [4,4])))))

Generation: 7
Best Individual:
Subpopulation 0:
Evaluated: T
Fitness: Standardized=26 Adjusted=0.03703704 Hits=38
 49 1 -117 -24 -17 -28 99 50 -95 -74 -128 -77 127 119 115 87 -13 -66 127 58 -40 -53 79
-51 123 -89 -73 -127 6 -116 109 -86 -128 -98 47 125 50 -95 -74 -128 -77 -37 -25 -84
Equivalent Tree:
Tree 0:
 (v8a (v8a mow (progn2 (ADF1 left) mow)) (frog
     (v8a ADF0 (progn2 mow (ADF1 (ADF1 mow))))))
Tree 1:
 (progn2 (progn2 mow (progn2 (v8a left (progn2
```

```
     mow mow)) (v8a [3,1] (progn2 (v8a [5,0] mow)
     mow)))) mow)
Tree 2:
 (progn2 mow (progn2 (v8a left ADF0) ADF0))

Generation: 8
Best Individual:
Subpopulation 0:
Evaluated: T
Fitness: Standardized=18 Adjusted=0.05263158 Hits=46
 23 -63 65 103 87 30 73 123 -89 -73 -127 6 99 50 -95 -74 -128 -77 127 119 115 8 49 27
-63 58 -40 -53 79 -51 123 -89 -73 -127 6 -116 109 -86 -128 -98 47 125 -53 74 -33 -127
6 -116 109 -86 -128 -98 47 125 41 -104 -5 105 97 -21 119 -17
Equivalent Tree:
Tree 0:
 (ADF1 (v8a (v8a (ADF1 (ADF1 ADF0)) (v8a mow
     (ADF1 (ADF1 (v8a ADF0 mow)))))) (frog (v8a
     ADF0 (progn2 mow (ADF1 (ADF1 mow)))))))
Tree 1:
 (v8a mow (progn2 (progn2 (v8a left (progn2
     mow mow)) (v8a [3,1] (progn2 (v8a [5,0] mow)
     mow))) mow))
Tree 2:
 (progn2 mow (progn2 (v8a ADF0 ARG0) [4,5]))

Generation: 9
Best Individual:
Subpopulation 0:
Evaluated: T
Fitness: Standardized=12 Adjusted=0.07692308 Hits=52
 23 -63 65 103 87 30 73 123 -89 -73 -127 6 99 50 -95 -74 73 123 -89 -73 -127 6 -116
123 -89 -73 -127 6 -116 109 -86 -128 -98 47 125 -53 74 6 -116 109 -86 -128 -98 47 125
41 -104 -5 105 109 -86 -128 -98 47 125 -53 74 -33 -127 6 -116 109 -86 -128 -98 47 125
41 -104 -5 105 97 -21 119 37
Equivalent Tree:
Tree 0:
 (ADF1 (v8a (v8a (ADF1 (ADF1 ADF0)) (v8a mow
     (ADF1 (ADF1 (v8a ADF0 mow)))))) (frog (v8a
     ADF0 (v8a mow (ADF1 (ADF1 (v8a ADF0 left)))))))
Tree 1:
 (v8a [3,1] (progn2 (v8a [5,0] mow) mow))
Tree 2:
 (progn2 (progn2 mow (progn2 (v8a ADF0 ARG0)
     (v8a ADF0 ARG0))) (progn2 (progn2 mow (progn2
     (v8a (v8a left [3,1]) mow) ARG0)) (progn2
     (progn2 mow (progn2 (v8a ADF0 ARG0) [4,5]))
     (v8a (v8a ADF0 ARG0) (progn2 (progn2 mow
         (progn2 (v8a (v8a left [3,1]) mow) (v8a mow
             mow))) [4,4])))))

Generation: 10
Best Individual:
Subpopulation 0:
Evaluated: T
Fitness: Standardized=3 Adjusted=0.25 Hits=61
 49 1 -117 -24 -17 -28 99 50 -95 -74 -128 -77 127 119 115 87 -13 -66 127 58 -40 -53 79
-51 123 -89 -73 -127 6 -116 109 -86 -128 -98 47 125 50 -95 -127 6 -116 109 -86 -128 -
98 47 125 -53 74 6 -116 109 -86 -128 -98 47 125 41 -104 -5 105 109 -86 -128 -98 47 125
-53 74 -33 -127 6 -116 109 -86 -128 -98 47 125 41 -104 -5 105 97 -21 119 37
Equivalent Tree:
Tree 0:
 (v8a (v8a mow (progn2 (ADF1 left) mow)) (frog
```

```
     (v8a ADF0 (progn2 mow (ADF1 (ADF1 mow))))))
Tree 1:
 (progn2 (progn2 mow (progn2 (v8a left (progn2
     mow mow)) (v8a [3,1] (progn2 (v8a [5,0] mow)
     mow)))) mow)
Tree 2:
 (progn2 mow (progn2 (v8a left ADF0) (v8a
     (v8a ADF0 ARG0) (progn2 (progn2 mow (progn2
     (v8a ADF0 ARG0) (v8a ADF0 ARG0))) (progn2
     (progn2 mow (progn2 (v8a (v8a left [3,1])
         mow) ARG0)) (progn2 (progn2 mow (progn2 (v8a
     ADF0 ARG0) [4,5])) (v8a (v8a ADF0 ARG0) (progn2
     (progn2 mow (progn2 (v8a (v8a left [3,1])
         mow) (v8a mow mow))) [4,4]))))))))

Generation: 11
Best Individual:
Subpopulation 0:
Evaluated: T
Fitness: Standardized=1 Adjusted=0.5 Hits=63
 23 -63 65 103 87 30 73 123 -89 -73 -127 6 99 50 -95 -74 -128 -77 127 119 115 8 49 27
-63 58 -40 -53 79 -51 123 -89 -73 -127 6 -116 109 -86 -128 -128 -98 47 125 123 -89 -73
-127 6 -116 109 -86 -128 -98 47 125 123 -89 -73 -127 6 -116 109 -86 -128 -98 47 125 -
53 74 -33 47 125 41 -104 -5 105 97 -21 119
Equivalent Tree:
Tree 0:
 (ADF1 (v8a (v8a (ADF1 (ADF1 ADF0)) (v8a mow
     (ADF1 (ADF1 (v8a ADF0 mow))))) (frog (v8a
     ADF0 (progn2 mow (ADF1 (ADF1 mow)))))))
Tree 1:
 (v8a mow (progn2 (progn2 (v8a left (progn2
     mow mow)) (v8a [3,1] (progn2 (v8a [5,0] mow)
     mow))) mow))
Tree 2:
 (progn2 (progn2 mow (progn2 (v8a ARG0 [3,1])
     ARG0)) (v8a (v8a ADF0 ARG0) (progn2 (progn2
     mow (progn2 (v8a ARG0 [3,1]) ARG0)) (v8a
     (v8a ADF0 ARG0) (progn2 (progn2 mow (progn2
     (v8a ADF0 ARG0) [4,5])) (progn2 (v8a (v8a
     left [3,1]) mow) (v8a mow mow)))))))

Generation: 12
Best Individual:
Subpopulation 0:
Evaluated: T
Fitness: Standardized=1 Adjusted=0.5 Hits=63
 23 -63 65 103 87 30 73 123 -89 -73 -127 6 99 50 -95 -74 -128 -77 127 119 115 8 49 27
-63 58 -40 -53 79 -51 123 -89 -73 -127 6 -116 109 -86 -128 -128 -98 47 125 123 -89 -73
-127 6 -116 109 -86 -128 -98 47 125 123 -89 -73 -127 6 -116 109 -86 -128 -98 47 125 -
53 74 -33 47 125 41 -104 -5 105 97 -21 119 125 123 -89 -73 -127
Equivalent Tree:
Tree 0:
 (ADF1 (v8a (v8a (ADF1 (ADF1 ADF0)) (v8a mow
     (ADF1 (ADF1 (v8a ADF0 mow))))) (frog (v8a
     ADF0 (progn2 mow (ADF1 (ADF1 mow)))))))
Tree 1:
 (v8a mow (progn2 (progn2 (v8a left (progn2
     mow mow)) (v8a [3,1] (progn2 (v8a [5,0] mow)
     mow))) mow))
Tree 2:
 (progn2 (progn2 mow (progn2 (v8a ARG0 [3,1])
     ARG0)) (v8a (v8a ADF0 ARG0) (progn2 (progn2
```

```
     mow (progn2 (v8a ARG0 [3,1]) ARG0)) (v8a
     (v8a ADF0 ARG0) (progn2 (progn2 mow (progn2
     (v8a ADF0 ARG0) [4,5])) (progn2 (v8a (v8a
     left [3,1]) mow) (v8a mow mow)))))))

Generation: 13
Best Individual:
Subpopulation 0:
Evaluated: T
Fitness: Standardized=0 Adjusted=1 Hits=64
 23 -63 65 103 87 30 73 123 -89 -73 -127 6 99 50 -95 -74 -128 -77 127 119 115 87 -13 -
66 127 119 49 27 -63 58 -40 -53 79 -51 123 -89 -73 -127 6 -116 105 109 -86 -128 -98 47
125 -53 74 -33 -127 6 -116 -73 -127 6 -116 109 -86 -128 -98 47 125 -53 74 -73 -127 6 -
116 109 -86 -128 -98 47 125 -53 74 -33 -127 6 -116 109 -86 -128 -98 47 125 41 -104 -5
105 -73 -127 6 -116 109 -86 -128 -98 47 125 -53 -116 109 -86 -128 -98 47 125 -53 74 -
33 47 125 41 -104 -5 105 97 -21 119
Equivalent Tree:
Tree 0:
 (ADF1 (v8a (v8a (ADF1 (ADF1 ADF0)) (v8a mow
     (ADF1 (ADF1 (v8a ADF0 mow)))))) (frog (v8a
     ADF0 (progn2 mow (ADF1 (ADF1 mow)))))))
Tree 1:
 (progn2 (progn2 mow (progn2 mow mow)) (progn2
     (progn2 (v8a left (progn2 mow mow)) (v8a
         [3,1] (progn2 (v8a [5,0] mow) left))) mow))
Tree 2:
 (progn2 (progn2 mow (progn2 (v8a ADF0 ARG0)
     [4,5])) (v8a (v8a ADF0 ARG0) (v8a (v8a ADF0
     ARG0) (progn2 (progn2 mow (progn2 (v8a ADF0
     ARG0) ARG0)) (v8a (v8a ADF0 ARG0) (progn2
     (progn2 mow (progn2 (v8a ADF0 ARG0) [4,5]))
     (v8a (v8a ADF0 ARG0) (progn2 (progn2 mow
         (progn2 (v8a (v8a left [3,1]) mow) ARG0))
         (v8a (v8a ADF0 ARG0) (progn2 (progn2 mow
             (progn2 (v8a ADF0 ADF0) ARG0)) (progn2 (progn2
             mow (progn2 (v8a ADF0 ARG0) [4,5])) (progn2
             (v8a (v8a left [3,1]) mow) (v8a mow mow)))))))))))))
```

Найкраще покоління:

```
Best Individual of Run:
Evaluated: T
Fitness: Standardized=0 Adjusted=1 Hits=64
 23 -63 65 103 87 30 73 123 -89 -73 -127 6 99 50 -95 -74 -128 -77 127 119 115 87 -13 -
66 127 119 49 27 -63 58 -40 -53 79 -51 123 -89 -73 -127 6 -116 105 109 -86 -128 -98 47
125 -53 74 -33 -127 6 -116 -73 -127 6 -116 109 -86 -128 -98 47 125 -53 74 -73 -127 6 -
116 109 -86 -128 -98 47 125 -53 74 -33 -127 6 -116 109 -86 -128 -98 47 125 41 -104 -5
105 -73 -127 6 -116 109 -86 -128 -98 47 125 -53 -116 109 -86 -128 -98 47 125 -53 74 -
33 47 125 41 -104 -5 105 97 -21 119
Equivalent Tree:
Tree 0:
 (ADF1 (v8a (v8a (ADF1 (ADF1 ADF0)) (v8a mow
     (ADF1 (ADF1 (v8a ADF0 mow)))))) (frog (v8a
     ADF0 (progn2 mow (ADF1 (ADF1 mow)))))))
Tree 1:
 (progn2 (progn2 mow (progn2 mow mow)) (progn2
     (progn2 (v8a left (progn2 mow mow)) (v8a
         [3,1] (progn2 (v8a [5,0] mow) left))) mow))
Tree 2:
```

```
(progn2 (progn2 mow (progn2 (v8a ADF0 ARG0)
    [4,5])) (v8a (v8a ADF0 ARG0) (v8a (v8a ADF0
    ARG0) (progn2 (progn2 mow (progn2 (v8a ADF0
    ARG0) ARG0)) (v8a (v8a ADF0 ARG0) (progn2
    (progn2 mow (progn2 (v8a ADF0 ARG0) [4,5]))
    (v8a (v8a ADF0 ARG0) (progn2 (progn2 mow
        (progn2 (v8a (v8a left [3,1]) mow) ARG0))
        (v8a (v8a ADF0 ARG0) (progn2 (progn2 mow
            (progn2 (v8a ADF0 ADF0) ARG0)) (progn2 (progn2
            mow (progn2 (v8a ADF0 ARG0) [4,5])) (progn2
            (v8a (v8a left [3,1]) mow) (v8a mow mow)))))))))))))
```

Шлях газонокосарки найкращого покоління рішень:

```
Y ->
X+----+----+----+----+----+----+----+----+
|| 144| 129| 100| 273| 136| 121| 106| 487|
v+----+----+----+----+----+----+----+----+
 | 103| 102| 101|  64|  65|  66|  67| 104|
 +----+----+----+----+----+----+----+----+
 | 160|  51|   6|   7|   8|   9|  10|  11|
 +----+----+----+----+----+----+----+----+
 | 175|  50|   5|  20| 217|  56|  41|  12|
 +----+----+----+----+----+----+----+----+
 | 224|  49|   4|  19| 218|  57|  42|  13|
 +----+----+----+----+----+----+----+----+
 | 153|  48|   3|   2|   1|  16|  15|  14|
 +----+----+----+----+----+----+----+----+
 |  96|  97|  98|  75|  74|  73|  72|  95|
 +----+----+----+----+----+----+----+----+
 | 143| 128|  99|  90|  89|  88|  87| 454|
 +----+----+----+----+----+----+----+----+
```

Висновки

Ознайомився з бібліотекою ECJ. Виконав індивідуальне завдання – задача про оптимальних шлях газонокосарки, отримав розв’язок використовуючи для бібліотеку ECJ.

Список використаної літератури

1. Інтернет-ресурс: http://www.genetic-programming.com/jkpubs94.html.
2. Інтернет-ресурс: http://dev.heuristiclab.com/trac.fcgi/wiki/Documentation/Howto/Implement%20Genetic%20Programming%20Problems.
3. Інтернет-ресурс: https://books.google.com.ua/books?id=eu2JplnQdBkC&pg=PA101&lpg=PA101&dq=Lawn+mower+problem+Koza&source=bl&ots=XtlFigHjic&sig=n-8HLv0YsO8R3MKU-0bWe0kwjOw&hl=en&sa=X&ved=0CDsQ6AEwBGoVChMIvMjZ2-OQxgIVcVnbCh2I2gBP#v=onepage&q=Lawn%20mower%20problem%20Koza&f=false.
4. Інтернет-ресурс: https://cs.gmu.edu/~eclab/projects/ecj/docs/classdocs/ec/gp/ERC.html.