Island Evolutionary Algorithm

Filip Gąciarz

Igor Ratajczyk

Dominik Żurek

Credentials

Anna Ostowska

Hanna Jarlaczyńska

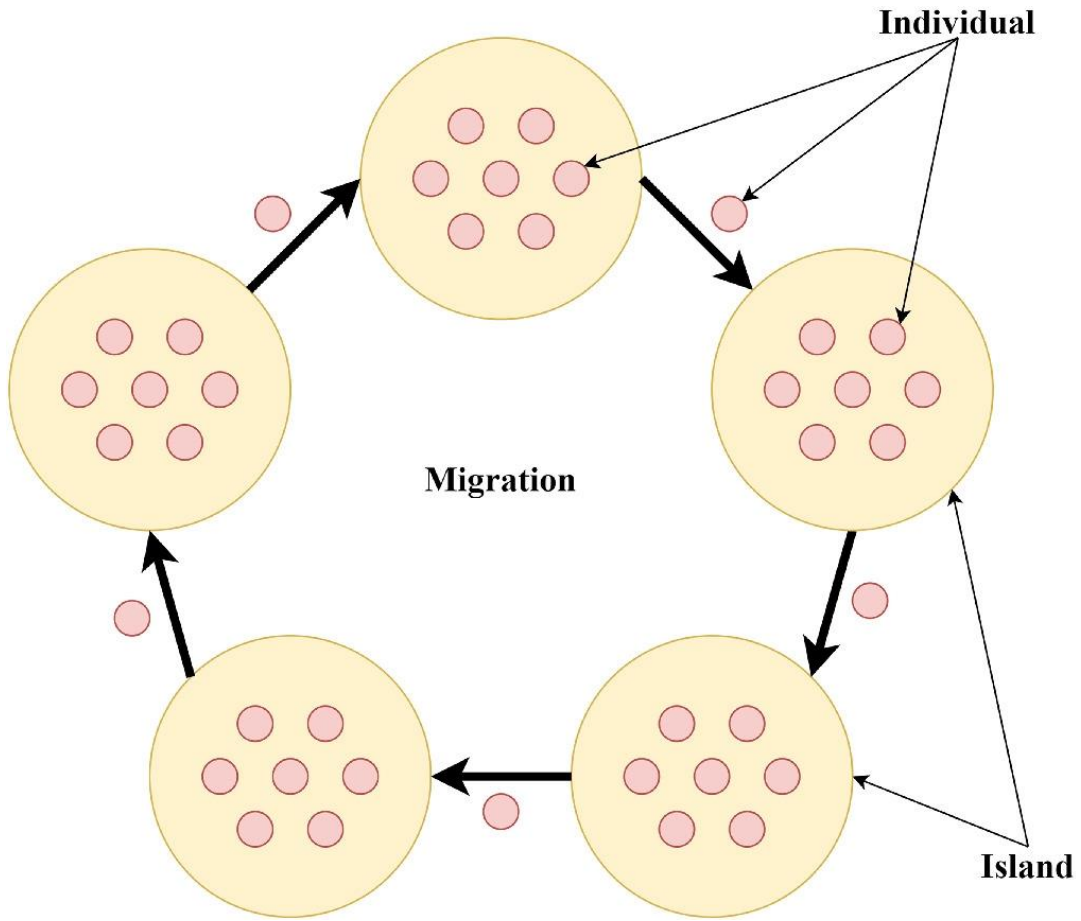
Mateusz Ślażyński

IEA Overview

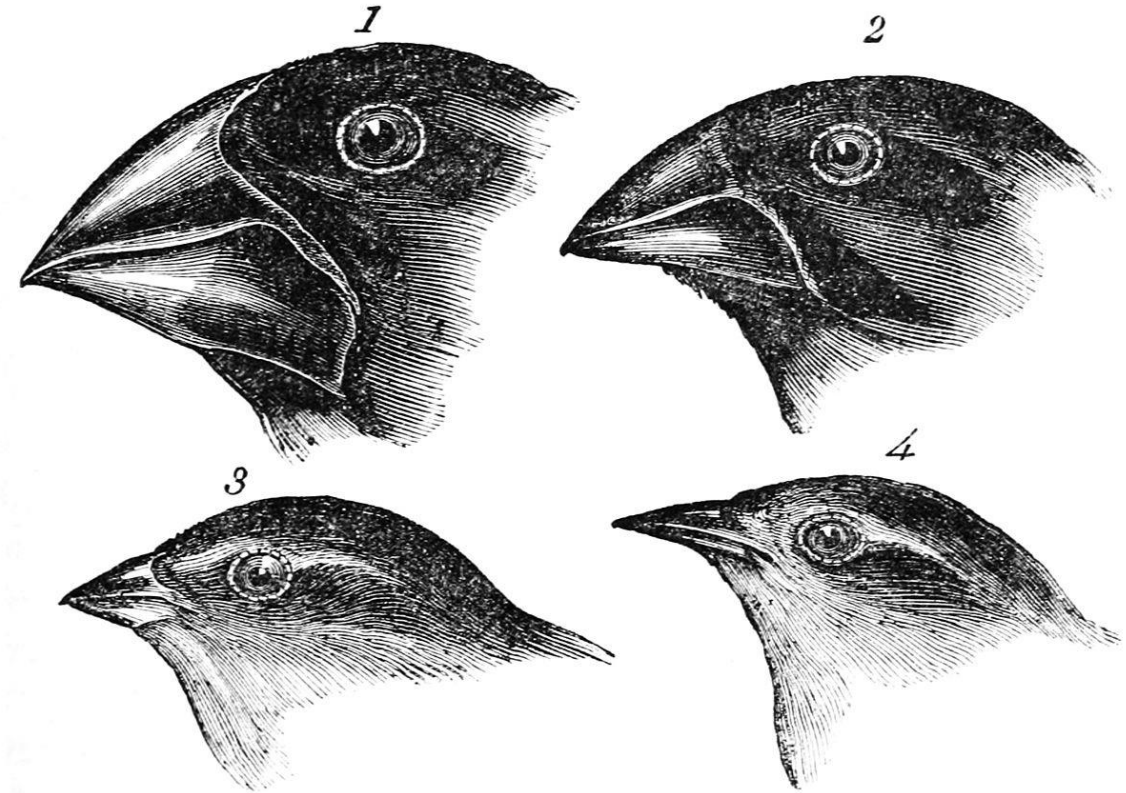
Island Evolutionary Algorithms models are popular extension of classical EAs. The very idea behind IEA is to divide the population of candidate solutions into several subpopulations, each evolving independently on different islands. Not only is it not required for all islands to posses the same Mutation and Crossover operators but also there is no need to apply the same Fitness Function.

As each Island evolves in different manner, they do not „blend” their solutions. Knowledge sharing is performed via migration operator. In every epoch, it is not possible for all solutions to cross with every other one.

IEA



IEA Darvins Finches



Theories of Evolution in EAs

Darwinian Evolution– classical EA – fitness is calculated based on phenotype (genotype)

$$\big(X, f(X)\big)$$

Larmarkian Evolution – genotype of individual is optimized and then optimized genotype is preserved and used to calculate fitness

$$\big(X^*, f(X^*)\big)$$

Balwinian Evolution – genotype of individual is optimized and then optimized genotype used to calculate fitness while genotype remains unchanged

$$\big(X, f(X^*)\big)$$

Problem to be solved – Timetable enroll

Problem to be solved is to assign S students to G possible groups that belong to C classes.

Problem is entirely modelled as discrete optimization problem. Similar approaches of modeling are to be met in AGH for certain faculties (although methods to solve these models are different).

Both obligatory and elective courses are possible to include.

Timetable Enroll - constraints

Problem is heavily constrained. Constraints are:

- All students have to participate obligatory courses
- If student has chosen an elective course, he must attend these classes
- Student cannot attend two groups of the same classes
- Group sizes are limited and cannot be exceeded.

Timetable Enroll - constraints

- Groups that overlap in time cannot be attended at the same time
- Groups that are distant (space) cannot be attended if time required to travel too short
- Complex dependencies between groups can exclude simultaneous participation.

Timetable Enroll - Objective

For the calculation of the objective, we introduce additional factors.

- Group/Break preference – student may decide whether he prefers to enroll his preferred groups or just spend as little time as possible at the University.
- Group Preference – every students mark his preferences when it comes to group assignment
- Exclusion – students can mark groups that they cannot be assigned

Timetable Enroll – Objective

For each student and each day, we calculate the time they have to spend at the university and how much they spend on learning. Difference between these number is the time they have wasted that we aim to minimize. We call it **break disappointment**.

For each student and each class, we calculate difference between their favorite group and the one they have been assigned. We sum those values for each student and will call this value **preference disappointment**

Technology

Technology has been chosen to support distributed paradigm. Language should also be capable of HPC (High Performance Computing) at least roughly.

Language should also be equipped with proper libraries for both computation and distributed management.

Java

Java:

- Multi platform
- JVM
- *Lingua Franca* of IT
- Verbose?
- Extremely influential
- Supports OOP
- JIT

Dart

Dart

- Multi platform
- Designed for UI
- Concise syntax
- Null safety design
- Mainly for Mobile apps

- Immature solution:
  - Interface?
  - JNI binder
- Channeling
- Purely Objective Language
- Both Compiled and Interpreted Language at the same time

C#

- Modern design
- CRL
- Multi-platfrom
- Advanced null checking designs
- .NET environment
- Visual Studio
- JIT
- Well documented

Problem Embodiment

Our solution may be classified as Island Evolutionary Algorithm with Larmarkian Optimization on subspace of solutions.

Every Island has its own Algorithm, with its own parameters.

Between Islands only solutions (without Island parameters) are migrated.

Connections between Islands are directed and do not change during the run.

Problem modelling in C#

Each solution cosist of:

- Genotype
- Fitness
- Feasibility Flag
- Flag whether solution has changed since it has been measured with fitness Function
- Violated Constraints Count

Channeling

Channeling is a modeling method to change the representation of a model while constructing the solution.

Different method may enforce some constraint to be satisfied. Channeling usually should be reversible.

Trivial example of channeling is e.g. One-Hot-encoding:

$$2 \leftrightarrow [0,0,1,0,0]$$

In this example, boolean representation enforce variable to be bounded, and integer representation keeps the number defined.

Solution Masking

Each Island has its own constant mask. Mask suggests which values of solution have to be greedily subspace – optimized, while the rest of solutions remains unchanged.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 4 & 5 \\ 6 & -1 & 6 \\ 7 & 8 & -1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & \_ \\ 4 & \_ & 5 \\ 6 & \_ & 6 \\ \_ & 8 & -1 \end{bmatrix}$$

Semi Feasible Cast

Integer representation of a problem barely can keep the variables bounded.

As assigning group to a wrong class do not introduce anything to a problem we check whether every group has been assigned correctly.

In case of incorrectly assigned groups we reassign them to a randomly selected proper one.

Genotype

Genotype is a $S \times C$ matrix that keeps the information which group given student attends.

$$S = 4$$
$$C = 3$$
$$G = 10$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 4 & 5 \\ 6 & -1 & 6 \\ 7 & 8 & -1 \end{bmatrix}$$

Phenotype

Phenotype of a solution is its boolean representation:

$$S = 4$$
$$C = 3$$
$$G = 6$$

$$\begin{bmatrix} 0 & 3 & 5 \\ 1 & 3 & -1 \\ 0 & -1 & 4 \\ 1 & 2 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Fitness Function

Fitness function is measuread as already mentioned objective.

We apply penalty for every constraint violated and leave penalty coefficient as parameter.

Mutation

Mutation randomly changes selected variables by adding or subtracting given integer.

After mutation the solution is casted to Semi - Feasible.

Crossover

Crossover operator cuts the Genotype in N points and exchange the cut parts between offspring.

Cut is placed along students, as it is more probable that two students share the same classes that class to share common students (and to preserve amount of conflicts).

Selection

Selection has been implemented as Tournament Selection.

- Selection of a group for tournament
- Selection of a winned

Migration

Migration operator randomly selects the iteration of algorithm to select the migrants and then until proper amount of migrant has not been selected it selects best solution with probability of p, another one with probability of p^2 and so on.

How to approach Enroll problem?

SOTA:

Usually Enroll problem occurs in education sector. It is common, that human expert is incapable of producing even feasible solution.

Some Faculties (IET) makes use of certain solvers for timetable assignment.

AGH UST made approach to implement these solvers for entire university.

Sadly – usually the details of used solvers are not unveiled for public.

SATisfiability problem

Formula Ψ is satisfiable if and only if there exists an interpretation I, such that Ψ is satisfied with it:

$$\vDash_I \Psi$$

Satisfiability problem is the problem of determining whether given formula Ψ is satisfiable. Sometimes SAT requires folmuae given in CNF (conjunctive normal form, product of sums). In general SAT problem is NP-Complete, and it is the first problem to be announced such. Were one to elaborate list of NP-C problem: TSP, Knapsack problem to the name of few.

Constraint Programming

The very set of logical formulae can be considered as constraints, and it is a model satisfying these constrains one may look for.

CP is a paradigm of solving combinatorial problems. Usually CP program consists of:

- Variables
- Domains of variables (discrete and preferably limited)
- Constraints <-> RULES

Constraint Programming Constraints

CP Modelling Languages are equipped with multiple pre-defined constraints, some of them are dedicated for certain problems.

Pre-defined constraints efficiently coexist with CP solvers.

- Constraint that all values in vector are different
- Conditional constraints
- Constraints dedicated for packing and scheduling problems

Constraint Programming

Constraint Programming as a paradigm is dedicated to discrete, heavily constrained problems, where feasible representation is impossible to achieve and finding feasible solution is hard.

Optimization in CP Solvers is performed by comparison feasible solutions.

CP and similar algorithms.

- CP in comparison to B&B requires no prior knowledge of a given problem.
- CP in comparison to Wave Function Collapse algorithm can backtrack the solutions.
- CP in comparison to LKH proves optimality of a solution and is applicable to wider spectrum of problems.

Constraint Programming Solvers

CP solvers work usually via algorithm called Constraint Propagation for both Solve Satisfy (ß) and Solve Optimize. They result in OPTIMAL solutions given enough time. For unsatisfiable problem CPSolver outputs information about it.

Advanced CP compilers utilize algorithm called Arc Consistency.

CP Solver Flex Section

Let us consider time required for CP Solver to schedule feasible plan for 40 students and 30 groups.

How much time do CP Solver require to find a feasible solution?

CP Solver Flex Section

Let us consider time required for CP Solver to schedule feasible plan for 40 students and 30 groups.

How much time do CP Solver require to find a feasible solution?

476 [ms]

CP Modelling Techniques

CP rather utilizes modeling techinques based mainly on redundant constraints and symmetry breaking rather than imperative algorithm coding.

For redundant constraints and symmetry breaking problem of minimizing the area to pack N = 15 squares drops from

33s 700 ms

To:

238 ms

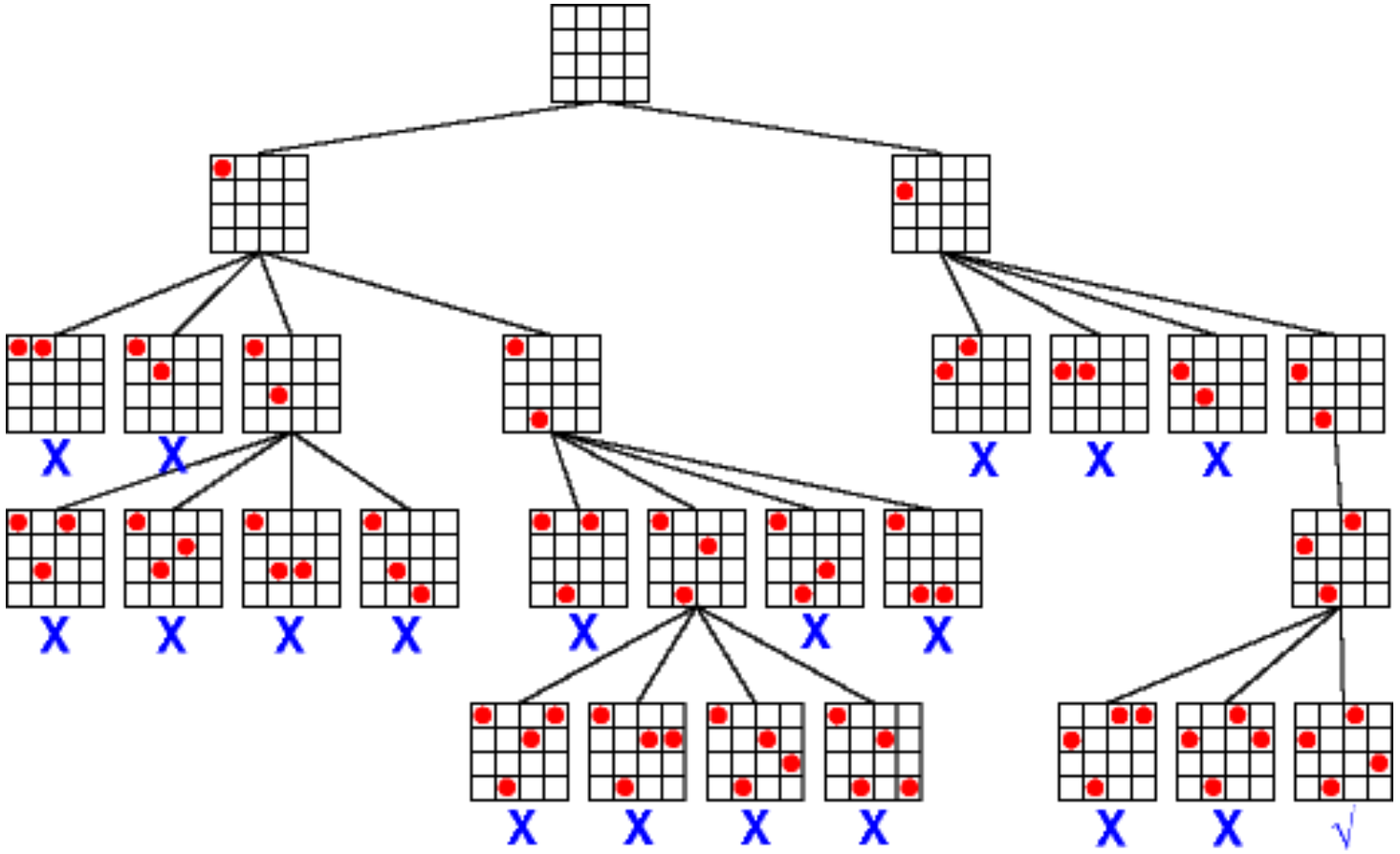
Constraint Propagation

Constraint propagation is a fundamental technique used in constraint satisfaction and optimization problems. It involves iteratively applying constraints to reduce the search space and derive new information that can be used to further refine the problem solution. The goal is to efficiently eliminate inconsistent or infeasible assignments and guide the search towards valid solutions.

Constraint Propagation

1. Initialization: Assign an initial value to each variable and create an initial domain for each variable.
2. Constraint propagation: Iterate through the constraints and update the domains of the variables based on the current assignments. This step is often repeated until no further changes occur.
3. a. For each constraint, check if it is satisfied or violated based on the current assignments. If a constraint is violated, it means the current assignment is inconsistent and should be revised.
4. b. If a constraint is violated, prune the inconsistent values from the domains of the variables involved in that constraint. This pruning operation restricts the variables' domains to only contain values that are consistent with the constraints.
5. Solution search: Once constraint propagation has been applied, the algorithm may perform a search for a valid solution based on the updated domains of the variables. This search can be performed using various techniques, such as backtracking or local search.
6. Backtracking: If a solution is not found or additional solutions are desired, the algorithm can employ backtracking to explore different assignments. Backtracking involves systematically undoing previous assignments and trying alternative values until a valid solution is found or all possibilities have been exhausted.

Constraint Propagation



Modelling for CP

$$q \; where \; p$$
$$p \rightarrow q$$
$$\neg p \vee q$$
$$\neg q \rightarrow \neg p$$
$$\neg p \; where \; \neg q$$

All of these formulae are equivalent, but for CP solvers they may not be the same in means of performance. Effectiveness may even differ between solvers!

CP in terms of Theory of Compilation

Some Modern schemas of executing the programs are:

Code → compilation → +data → execution (+ JIT?)

Constraint programming solvers:

Code + Data → compilation → execution

Utilizing Prior Knowledge

Although CP do not require any prior knowledge about the problem, it may be utilized when it comes to order of search of variables.

Basic CP Solver always chooses the variable with smallest (in means of numer of variables) domain.

With prior knowledge it is possible to help Solver to fail as soon as possbile to backtrack and reduce significantly seach space.

OR-Tools

Operational Research Tools is a framework created at Google for solving multiple problems typical for Operational Research.

ORTools CP Solver have been winning Gold Medal in Constraint Programming Solvers annually since 2013.

ORTools CP solvers utilize two backends – CP-SAT solver and CP-MIP solver.

In oposite to other SotA CP solvers, ORTools CP Solver supports paralel search.

ORTools are interfaced for: Python, C++, Java and C#.

Masking

As optimization of entire problem may be complex and time consuming every Island posses a mask to mark these variables that will be preserved during CP Optimization.

This approach fulfills the Larmarkian evolution paradigm and may be considered as subspace optimization.

CP Solver and IEA - Summary

Our presented approach fulfills the paradigm of distributed computing.

We first search solutions to be optimized and then optimize them in exhaustive manner.

Island separation allows to thrive in independent direction for every population.

We consider this solution to be meet the standard of leading industrial/commertial solutions in real world applications as not only decently HPC language and best possible solvers have been chosen but also computations may be performed in distributed manner.