Evolutionary Algorithms and Hyper-Heuristics

Nelishia Pillay

School of Mathematics, Statistics and Computer Science

University of KwaZulu-Natal

South Africa

NICOG

NATURE INSPIRED COMPUTING OPTIMIZATION GROUP

UNIVERSITY OF KWAZULU-NATAL

Congress on Evolutionary Computation 2017

Tutorial Site

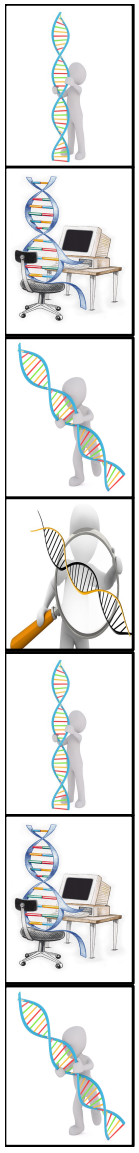
http://titancs.ukzn.ac.za/CEC2017.aspx

Outline

- Introduction
- An overview of hyper-heuristics
  - Low-level heuristics
  - Classification of hyper-heuristics
- Evolutionary algorithm hyper-heuristics
  - Overview
  - Selection hyper-heuristics
  - Generation hyper-heuristics
  - Challenges
- *EvoHyp*
- Evolutionary algorithm hyper-heuristics and design
- Hyper-heuristics for evolutionary algorithm design
- Discussion

Introduction

- Hyper-heuristics aim to provide a more generalized solution.
- Explore a heuristic space rather than a solution space.
- Heuristic space – low-level heuristics
  - Constructive
  - Perturbative
- Selection vs. Generation
- Role played by evolutionary algorithms



An Overview of Hyper-Heuristics

Low-Level Heuristics

Low-Level Heuristics

- Heuristics vs. low-level heuristics
- Construction heuristics – used to create a candidate solution
- Perturbative heuristics – used to improve a solution created randomly or using a construction heuristic

Case Study: Examination Timetabling (ETP)

- This problem involves the allocation of examinations to a set of specified timetable periods.
- Constraints
  - Hard constraints
  - Soft constraints
- Objective function
  - Hard constraint cost must be zero
  - Minimize soft constraint cost

Low-Level Construction Heuristics for the ETP

- Largest degree (l) - The examination with the most clashes is scheduled first.
- Largest enrollment (e) - The examination with the largest number of students is scheduled first.
- Largest weighted degree (w) - The examination with the largest number of students involved in clashes is scheduled first.
- Saturation degree (s) - The examination with the least number of feasible periods on the timetable is scheduled first.

Example of a Static Heuristic

Clash matrix:

	E1	E2	E3	E4	
E1	0	10	50	20	← 3
E2	10	0	0	0	← 1
E3	50	0	0	30	← 2
E4	20	0	30	0	← 2

Static heuristics:

Examination	Largest Degree
E1	3
E2	1
E3	2
E4	2

Timetable:

Period	Examinations
P1	E1
P2	E3 E2
P3	E4
P4	

Example of a Dynamic Heuristic

Clash matrix:

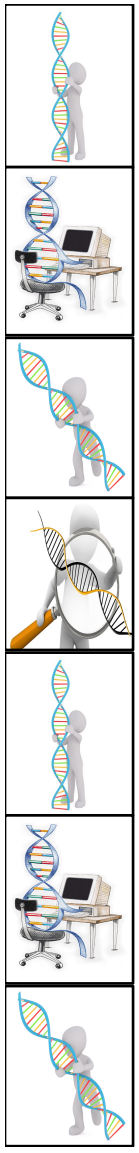
	E1	E2	E3	E4
E1	0	10	50	20
E2	10	0	0	0
E3	50	0	0	30
E4	20	0	30	0

Static heuristics:

Examination	Saturation Degree
~~E1~~	~~4~~
~~E2~~	~~4~~ ~~3~~
~~E3~~	~~4~~ ~~3~~
~~E4~~	~~4~~ ~~3~~ ~~2~~

Timetable:

Period	Examinations
P1	E1
P2	E2 E3
P3	E4
P4	

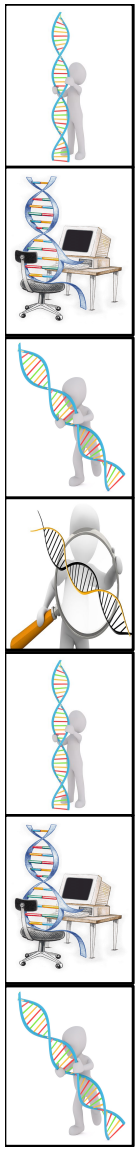


Low-Level Perturbative Heuristics for the ETP

- Swapping two randomly selected exams
- Swapping subsets of exams
- De-allocating exams
- Rescheduling exams
- Swapping timeslots of two randomly selected examinations

One-Dimensional Bin-Packing

- Define low-level construction heuristics for this problem.
- Define low-level perturbative heuristics for this problem.
- Can any of the heuristics used for the examination timetabling problem above be used for this problem?



An Overview of Hyper-Heuristics

Classification of Hyper-Heuristics

Hyper-Heuristic Classification

- Hyper-heuristics select or create low-level heuristics.
- Low-level heuristics are constructive or perturbative.
- Classification:
  - Selection constructive
  - Selection perturbative
  - Generation constructive
  - Generation perturbative
- Learning – offline vs. online

Selection Constructive

- Selects the constructive heuristic to use at each stage in constructing a solution.
- A problem specific objective function and low-level construction heuristics provide input to the hyper-heuristic.
- Applications: educational timetabling, production scheduling, bin packing and cutting stock problems.
- Methods used by the hyper-heuristic: case-based reasoning, tabu search, evolutionary algorithms, simulated annealing, variable neighbourhood search.

Selection Perturbative

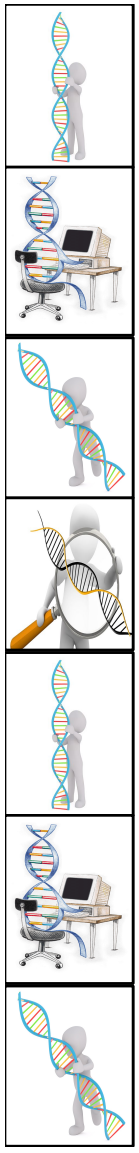
- Selection perturbative hyper-heuristics choose a low-level perturbative heuristic at each stage in the improvement.
- Multi-point vs. single-point search
- Single-point
  - Heuristic selection – random, choice function, roulette wheel, reinforcement learning
  - Move acceptance – deterministic vs. non-deterministic
- Multi-point search uses population based methods, e.g. genetic algorithms, ant colonization.
- Applications: educational timetabling, sports scheduling, personal scheduling and vehicle routing

Generation Constructive

- Create low-level constructive heuristics.
- Have previously been manually derived.
- Genetic programming has primarily been used for this.
- More recently variations of genetic programming: grammatical evolution and gene expression programming.
- Have produced good results for the domains of educational timetabling and packing problems.
- Disposable vs. reusable low-level heuristics.

Generation Perturbative

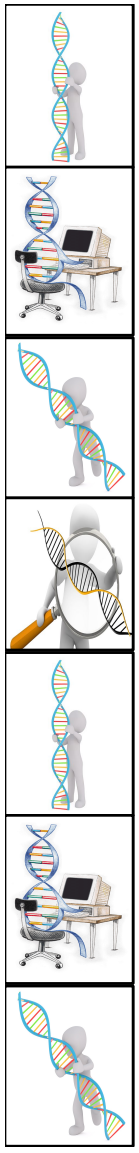
- Create low-level perturbative heuristics.
- The heuristics are local search operators, algorithms and metaheuristics.
- Variations of genetic programming have been used to evolve heuristics: grammar-based GP, linear GP and Cartesian GP.
- Disposable vs. reusable low-level heuristics.



Evolutionary Algorithm Hyper-Heuristics

Overview

- Evolutionary algorithms have been used for both the selection and generation of low-level construction and perturbative hyper-heuristics.
- Genetic algorithms have essentially been used for selection and genetic programming for generation.
- Variations of genetic programming have proven to be effective for generation hyper-heuristics.



Evolutionary Algorithm Hyper-Heuristics

Selection Hyper-Heuristics

Selection Constructive Hyper-Heuristics

- Selection construction – find a combination of heuristics that will produce a solution minimizing the objective function.
- The genetic algorithm explores the heuristic space.
- Each chromosome is composed of genes representing the low-level heuristics.
- Each heuristic is used to select one or more entities

Case Study: Selection Constructive Hyper-Heuristic for the ETP

- Low-level construction heuristics
  - Largest degree (l)
  - Largest enrollment (e)
  - Largest weighted degree (w)
  - Saturation degree (s)
- Chromosome representation and initial population generation
  - Example: *lwwse*
  - Chromosome length – longer than the number of examinations vs. shorter than number of examinations.

Case Study: Selection Constructive Hyper-Heuristic for the ETP

- Fitness calculation
  - Each chromosome is used to create a timetable
  - The fitness is a function of the hard and soft constraint cost.
  - Product of the hard constraints plus one and the soft constraints
- Example: *lwwse* to allocate the four examinations in the previous example:

  $l \rightarrow$ E1 $w \rightarrow$ E2 $w \rightarrow$ E3 $s \rightarrow$ E4

Case Study: Selection Constructive Hyper-Heuristic for the ETP

- Selection methods
  - Tournament selection
  - Fitness proportionate selection
- Genetic operators
  - Mutation – Parent: *ssde*
    Mutation point: 2
    Offspring: *swde*
  - Crossover – Parents: *ssde* *lwws*
    Crossover point: 2
    Offspring: *ssws* *lwde*

Examples

Approach	Application	Study
Messy GA	One and two dimensional bin-packing	Lopez-Camacho et al. (2014)
Genetic algorithm	Packing problems	Pillay (2012), Ross et al. (2002), Ross et al. (2003), Terashima-Marin (2010)
Genetic algorithm	Educational timetabling	Pillay (2010), Pillay(2011), Pillay (2012), Pillay (2013) Ross et al. (2014)
Genetic algorithms	Cutting stock problems	Terashima-Marin (2006)
Genetic algorithmc	Dynamic variable ordering problem	Terashima-Marin (2008)

Selection Perturbative Hyper-Heuristic for the ETP

- For a selection perturbative hyper-heuristic the low-level heuristics will change the initial solution.
- An initial solution will be created by using a low-level construction heuristic.
- Example low-level heuristics:
  - Swapping two randomly selected exams (e)
  - Swapping subsets of exams (s)
  - Deallocating exams (d)
  - Rescheduling exams (a)
  - Swapping timeslots of two randomly selected examinations (t)
- Example chromosome: *tdsse*

Examples

Approach	Application	Hyper-Heuristic	Study
Genetic algorithm	Trainer Scheduling	Selection perturbative	Cowling et al. (2002)
Genetic algorithm	Distributed staff trainer scheduling	Section perturbative	Han and Kendall (2003)

Evolutionary Algorithm Hyper-Heuristics

Generation Hyper-Heuristics

Generation Constructive Hyper-Heuristics

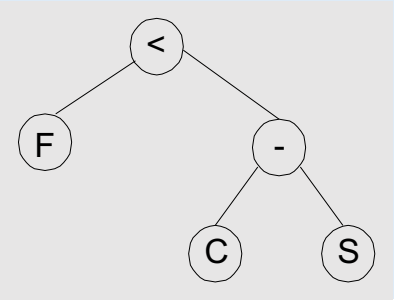
- Genetic programming and variations of genetic programming, e.g. grammar-based genetic programming and grammatical evolution have been used.
- Generation constructive heuristics combine:
  - variables representing the characteristics of the problem
  - existing low-level construction heuristics to create new heuristics.
- Applications: educational timetabling, packing problems and vehicle routing problems

Case Study: One-Dimensional Bin-Packing

- Function set:
  - Standard addition (+), subtraction (-) and multiplication (*).
  - % - protected division which will return a 1 if the denominator is zero.
  - < - will return a value of 1 if its first argument is less than or equal to its second and -1 otherwise.
- Terminal set
  - F - the fullness of a bin, i.e. the sum of the sizes of the elements in the bin.
  - C - the bin capacity
  - S - the size of the item to place next.

Case Study: One-Dimensional Bin-Packing

- Fitness
  - Each heuristic in the population is evaluated by using it to solve the problem instance.
  - The fitness is calculated to be the difference of the number of bins used and the ratio of the sum of the items to the sum of the capacity of all bins used.
- Example heuristic



Examples

Approach	Application	Study
Grammar-Based genetic programming	Examination timetabling	Bader-El-Den et al. (2009)
Grammatical evolution	Vehicle routing	Drake et al. (2013)
Genetic algorithms	One dimensional bin packing	Ozcan et al. (2011)
Single node genetic programming	One dimensional bin packing	Sim et al. (2013)
Genetic programming	Vehicle routing	Sim et al. (2016)
Grammar-based genetic programming	Constraint satisfaction problems	Sosa-Ascencio et al. (2015)

Examples

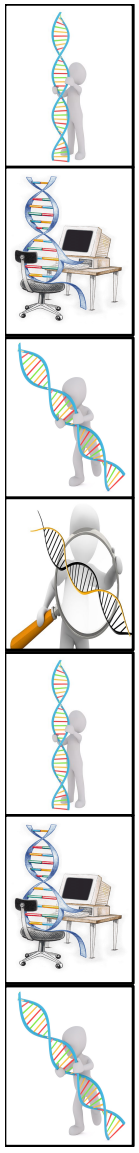
Approach	Application	Study
Genetic programming	Packing problems	Burke et al. (2007), Burke et al. (2010), Hyde (2010)
Genetic programming	Multidimensional knapsack problem	Drake et al. (2014)
Grammatical evolution	One dimensional knapsack problem	Sotelo-Figueroa 2013)
Genetic programming	Educational timetabling	Pillay (2009), Pillay(2011), Pillay (2016), Pillay and Banzhaf (2009)
Genetic programming and grammar-based genetic programming	Production scheduling	Branke et al. (2016)

Generation Perturbative Hyper-Heuristics

- Variations of genetic programming used
  - grammar-based genetic programming
  - linear genetic programming
  - Cartesian genetic programming
- Used to evolve local search operators, algorithms and metaheuristics.
- Components of existing perturbative heuristics are decomposed and combined with conditional branching and/or iterative constructs.

Examples

Approach	Application	Study
Genetic programming	3-SAT problem	Bader-El-Den et al. (2007)
Genetic programming	SAT problem	Fukanaga (2008)
Linear Genetic programming	Travelling salesman problem	Keller et al. (2003)
Cartesian genetic programming	Travelling salesman problem	Ryser-Welsh et al. (2015) Ryser-Welsh et al. (2016)
Genetic programming	Travelling salesman problem	Contreras-Bolton et al. (2007)
Genetic programming	Automatic clustering problem	Contreras-Bolton et al. (2007)

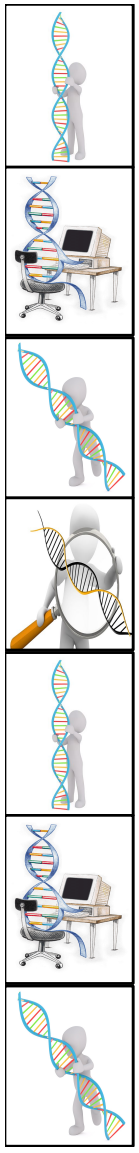


Evolutionary Algorithm Hyper-Heuristics

Challenges

Challenges

- High runtimes
  - Distributed architectures
  - Distributed frameworks
- Parameter tuning
  - Tuning tools, e.g iRace, ParamILS
  - Hyper-heuristics for design – using evolutionary algorithms with co-evolution



EvoHyp: A Java Toolkit for Evolutionary Algorithm Hyper-Heuristics

Overview

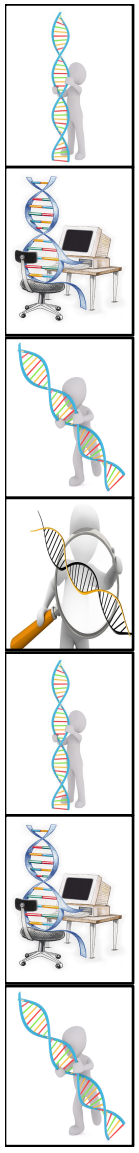
- *EvoHyp* is a Java evolutionary algorithm hyper-heuristic toolkit
- Allows the researcher to focus on the problem domain
- Packages provided by *EvoHyp*:
  - *GenAlg*
  - *GenProg*
  - *DistrGenAlg*
  - *DistrGenProg*
- Website: http://titancs.ukzn.ac.za/EvoHyp.aspx

GenAlg

- Implements a generational genetic algorithm.
- The genes of each chromosome are low-level heuristics
- The user has to implement the problem domain including a method to use the chromosome to construct or improve a solution.
- Fitness is a function of the objective value of the resulting solution.
- Tournament selection is used to choose parents.
- Mutation and crossover create offspring.
- Can be used to implement a selection constructive or selection perturbative hyper-heuristic.

GenProg

- Is a library for the implementation of a generation constructive hyper-heuristic.
- Implements a generational genetic programming algorithm.
- User must implement the problem domain including a method that uses an evolved heuristic to create a solution to the problem.
- The fitness is a function of the objective value of the resulting solution.
- Tournament selection is used to choose parents.
- Mutation and crossover create offspring.



DistrGenAlg and DistrGenProg

- Are distributed versions of *GenAlg* and *GenProg* aiming to reduce the runtimes of the hyper-heuristics.
- Uses a multicore architecture.
- The user has to specify the number of cores available for use.
- The creation and evaluation of the initial population is distributed over the number of cores.
- The creation and evaluation of offspring are distributed over the cores.

Evolutionary Algorithm Hyper-Heuristics for Design

Overview

- Selection and generative evolutionary algorithm hyper-heuristics have been used for design.
- Evolutionary algorithms used: genetic algorithms, genetic programming, grammatical evolution.
- Design aspects
  - Selecting parameter values
  - Selecting operators/methods and deciding control flow
  - Hybridization of methods
- Low-level heuristics are the parameter values, operators and methods.

Examples

Approach	Design Aspect	Study
Grammar based GP	Decision tree generation and design	Barros et al. (2012)
Evolutionary algorithms	Decision tree generation	Barros et al. (2013), Barros et al. (2014)
NSGA-II	Combine neural networks into a stacked neural network.	Furtuna et al. (2012)
Grammatical evolution	Selects operators and parameter values to solve a vehicle routing problem	Marshall et al. 2014)
Genetic programming	Generates black box search algorithms to solve the deceptive trap problem.	Martin and Tauritz (2014)

Examples

Approach	Design Aspect	Study
Grammatical evolution	Selects operators and parameter values for ant colonization to solve the travelling salesman problem.	Tavares and Pereira (2012)
Genetic programming	Generates mutation operators for evolutionary programming	Hong et al. (2013)
Grammatical evolution	Selects operators and parameter values for an evolutionary algorithm to solve the knapsack problem.	Lourenco et al. (2013)
Genetic algorithm	Designs a genetic programming algorithm to produce data classifiers	Nyathi et al. (2016)

Hyper-Heuristics for Evolutionary Algorithm Design

Overview

- Hyper-heuristics have also proven to be effective for the design of evolutionary algorithms.
- Has been used to:
  - decide on the control flow in an evolutionary algorithm by choosing when during the evolution process to use which operators
  - select of parameter values
  - hybridize evolutionary algorithms

Examples

Design Aspect	Study
Selection of recombination operator and selection method in differential evolution.	Tinoco et al. (2012)
Selects one of three multi-objective evolutionary algorithms to solve at each point of solving the problem	Maashi et al. (2014)
Parameter tuning and determining the stopping condition in and implementation of MOEA.	Segredo et al. (2014)
Selection of crossover operator, mutation operator and selection method in an evolutionary algorithm.	Kumari et al.(2012), Kumari et. al(2013)
Generation of mutation operators in a genetic algorithm	Woodward et al.(2012). Woodward et al. (2016)

Discussion Session: Future Research Directions