TUTORIAL: HYPER-HEURISTICS AND COMPUTATIONAL INTELLIGENCE

Nelishia Pillay
School of Mathematics, Statistics and Computer Science
University of KwaZulu-Natal
South Africa

TUTORIAL WEBSITE

- URL: http://titancs.ukzn.ac.za/SSCI2015Tutorial.aspx
- Resources available from site:
  - Breakdown of tutorial
  - References for the different parts/sections
  - Slides

OUTLINE

- Introduction to Hyper-Heuristics
- Low-Level Heuristics
- Classification of Hyper-Heuristics
  - Selection constructive
  - Selection selection perturbative
  - Generation constructive
  - Generation perturbative
- Cross Domain Challenge
- Frameworks
- Computational Intelligence Techniques
- Applications of Hyper-Heuristics
- Automated Design Using Hyper-Heuristics
- Challenges and Discussion

PART 1: INTRODUCTION

- Introduction to hyper-heuristics
- Low-level heuristics
- Classification of hyper-heuristics
- Cross domain challenge
- Frameworks

INTRODUCTION TO HYPER-HEURISTICS

- 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

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 enrolment (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: STATIC HEURISTIC

Clash matrix:

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

Static heuristics:

Examination	Largest Degree
E1	3
E2	1
E3	2
E4	

Timetable:

Period	Examinations
P1	E1
P2	E3 E2
P3	E4
P4	

EXAMPLE: DYNAMIC HEURISTIC

Clash matrix:

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

Static heuristics:

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

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

SELECTION CONSTRUCTIVE HYPER-HEURISTIC

- 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 HYPER-HEURISTICS

- 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 HYPER-HEURISTICS

- Create low-level heuristics.
- 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.

CROSS DOMAIN CHALLENGES

- CHeSC 2011 and CHeSC 2014
- HyFlex (Java implementation)
  - Provides method to create the initial solution
  - Provides low-level perturbative heuristics
  - Provides objective function
  - Researcher to create selective pertrubative hyper-heuristic
- Problem domains
  - Boolean satisfiability, one dimensional bin packing, flowshop scheduling, personnel scheduling.
  - Hidden domains: TSP and vehicle routing
- CHeSC 2014
  - Introduces “batching”.
  - Allows multithreading strategies

FRAMEWORKS

- Hyflex – created for the cross domain challenge (CheSC)
  - Development of selection perturbative hyper-heuristics
  - Low-level heuristics, method to create initial solution and objective function provided.
  - Implementation in Java
- HYPERION2 - For analyzing the trace performance of metaheuristics and hyper-heuristics
  - Implementation in Java
- Templar – Genetic programming toolkit for generation hyper-heuristics in Java.
- Survey of frameworks (Ryser-Welch & Miller 2014)

PART 2: TECHNIQUES & APPLICATIONS

- Selection constructive
- Selection perturbative
- Generation hyper-heuristics
- Case Study: selection hyper-heuristics
- Case Study: generation hyper-heuristics
- Hybrid hyper-heuristics

SELECTION CONSTRUCTIVE HYPER-HEURISTICS

- Methods used for this include:
  - tabu search
  - simulated annealing
  - variable neighbourhood search
  - genetic algorithms
  - case-based reasoning
- Applications: combinatorial optimization problems:
  - Educational timetabling
  - Production scheduling
  - Packing problems
  - Cutting stock problem
  - Workforce scheduling
  - Dynamic scheduling
  - Optimal discharge scheduling

SELECTION PERTURBATIVE HYPER-HEURISTICS: HEURISTIC SELECTION

- Simple random
- Random gradient
- Random permutation
- Random permutation gradient
- Reinforcement learning
- Greedy heuristic search
- Roulette wheel selection
- Markov chain
- Quantum-inspired learning strategy

SELECTION PERTURBATIVE HYPER-HEURISTICS: MOVE ACCEPTANCE

- Accept all moves
- Accept only improving moves
- Monte Carlo methods
- Great deluge
- Simulated annealing
- Late acceptance
- Adaptive limited list-based threshold acceptance

SELECTION PERTURBATIVE HYPER-HEURISTICS: MULTI-POINT SEARCH

- Do not have distinct phases
- Metaheuristics are usually employed to explore the heuristic space.
- In the context of GAs often referred to as a GA with an indirect representation.
- Metaheuristics used include genetic algorithms, ant colonization and particle swarm optimization.
- Methods used:
  - Genetic algorithms
  - Particle swarm optimization

SELECTION PERTURBATIVE HYPER-HEURISTICS: APPLICATIONS

- Personnel scheduling
- Educational timetabling
- Space allocation
- Vehicle routing
- Sports scheduling
- Grid scheduling
- Combinatorial interaction testing
- Water distribution network optimization
- Game playing strategies
- Dynamic optimization

CASE STUDY: GENETIC ALGORITHM 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: GENETIC ALGORITHM 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: GENETIC ALGORITHM 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* *lww*

    Crossover point: 2

    Offspring: *ssws* *lwde*

GENETIC ALGORITHM SELECTION PERTURBATIVE HYPER-HEURISTIC FOR THE ETP

- For a selection perturbative hyper-heuristic the low-level heuristics will change.
- 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*

GENERATION HYPER-HEURISTICS

- Genetic programming and variations of genetic programming, e.g. grammar-based genetic programming and grammatical evolution have been used.
- Generative construction heuristics combine:
  - variables representing the characteristics of the problem
  - existing low-level construction heuristics to create new heuristics.
- Generative perturbative hyper-heuristics combine different heuristic selection and move acceptance components to create new perturbative heuristics.

GENERATION HYPER-HEURISTICS: APPLICATIONS

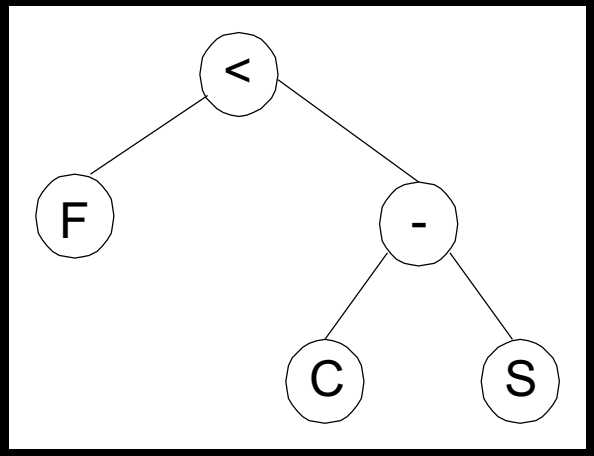
- Production scheduling
- Cutting and packing problems
- Boolean satisfiability
- Travelling salesman problem
- Function optimization
- Timetabling and scheduling

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



HYBRID HYPER-HEURISTICS

- Hybridizing heuristics – The hyper-heuristics search a space of constructive and perturbative heuristics.
- Hybridizing hyper-heuristics – the different types of hyper-heuristics are hybridized, e.g. generation and selection.
- Hybridizing heuristic space and solution space search
  - A hyper-heuristic searches the heuristic space
  - A local search or metaheuristic searches the solution space
  - The combination can happen in phases or sequentially
- Hybridizing of hyper-heuristics to firstly determine the set of low-level heuristics.

EXAMPLES: HYBRID HYPER-HEURISTICS

- Searching the constructive and perturbative heuristic space using tabu search to solve maritime service networks (Danach & Khalil 2015).
- Hybridizing a harmony search selection constructive hyper-heuristic and simulated annealing to solve the resource scheduling problem in cellular mobile networks (Dong et al. 2015).
- Combination of selection perturbative hyper-heuristic and local search to solve educational timetabling problems (Qu et al. 2009)
- Combination of a selection and generation hyper-heuristic to solve the capacitated vehicle routing problem (Hruska et al. 2015).
- Hybridization of two selection pertubative hyper-heuristics, the first determines the set of low-level heuristics to be used by the second hyper-heuristic (Kheiri & Ozcan 2015).

PART 3: AUTOMATED DESIGN

- Design decisions
- Parameter tuning
- Automating control flow
- Inducing operators/processes
- Induction algorithms
- Combining approaches
- Hyper Hyper-Heuristics

DESIGN DECISIONS

- Parameter tuning
- Automating control flow – operator/process selection
- Inducing operators/processes
- Inducing algorithms
- Combining approaches

PARAMETER TUNING

- Selecting parameter values and operators for a set algorithm.
- The parameters are treated as low-level heuristics.
- Selection hyper-heuristics are used to choose the best combination of parameter values.
- Choice function and multi-armed bandit to choose crossover and mutation in NSGA-II to solve the test order problem (2015).
- Self tuning for geometric semantic GP (2015).

PARAMETER TUNING

- Grammatical evolution
  - Vehicle routing problem (Marshall et al. 2014).
  - Selection of parameters for ant colonization to solve the travelling salesman problem (Tavares and Pereira 2012).
  - Selection of parameters for an evolutionary algorithm to solve the knapsack problem (Lourenco et al. 2013)
  - Parameter tuning and determining the stopping condition in an implementation of MOEA (Segredo et al. 2014).
  - Parameter tuning of PSO (Miranda & Prudencio 2015)

AUTOMATED CONTROL FLOW

- This is achieved by selecting operators/processes using a selection hyper-heuristic so as to induce an overall algorithm.
- The operators/processes are low-level heuristics.
- Examples
  - Selection of recombination operator and selection method in differential evolution (Tinco and Coello 2012).
  - Selection of crossover operator, mutation operator and selection method in an evolutionary algorithm (Kumari and Srinivas 2012; 2013).
  - Grammatical evolution to select operators for the vehicle routing problem (Marshall et al. 2014).

INDUCING OPERATORS/PROCESSES

- A generation hyper-heuristic is used to create a new operator or process.
- The operator or process is treated is a low-level heuristic.
- Examples
  - Genetic programming is used to generate mutation operators for evolutionary programming (Hong et al. 2013).
  - Local search is used to generate mutation operators in a genetic algorithm (Woodward & Swan 2012).
  - Induction of move operators using grammatical evolution for variable neighbourhood search to solve the vehicle routing problem (Drake et al. 2013).

INDUCING ALORITHMS/CONSTRUCTS

- Generation hyper-heuristics are used to create new algorithms or constructs for a technique.
- Decision tree induction and design
  - Grammar based genetic programming for decision tree induction (Barros et al. 2012).
  - Evolutionary algorithms for decision tree induction (Barros et al. 2013; 2014; 2015).
- Genetic programming to generate black box search algorithms to solve the deceptive trap problem (Martin & Tauritz 2014; 2015).
- Cartesian genetic programming for metaheuristic induction (Ryser-Welch et al. 2015).

INDUCING ALORITHMS/CONSTRUCTS

- Rule induction
  - Genetic programming and neural networks to solve the dynamic job shop scheduling problem (Branke et al. 2015).
  - Genetic programming evolution of dispatching rules to determine the next operation to be scheduled (Sim & Hart 2015).
- Evolution of algorithms to solve the knapsack problem (Harris et al. 2015)
- Genetic programming to generate black box search algorithms to solve the deceptive trap problem (Martin & Tauritz 2014; 2015).
- Inducing due date models for the job shop problem using genetic programming (Nguyen et al. 2015).

COMBINING APPROACHES

- This essentially involves creating an intelligent hybrid system.
- Each approach is treated as a low-level heuristic.
- A selection hyper-heuristic is used to select which approach to use when solving a problem.
- Alternatively a generation hyper-heuristic is used to combine approaches.
- Choice function hyper-heuristic to select one of three multi-objective evolutionary algorithms to solve at each point of solving the problem (Maashi et al. 2015).
- NSGA-II is used to combine neural networks into a stacked neural network (Furtuna et al. 2012).

HYPER HYPER-HEURISTICS

- Hyper-hyper-heuristics is the extension of the idea of hyper-heuristics to select or combine hyper-heuristics and generate new hyper-heuristics.
- Grammatical evolution is used to combine heuristic selection and move acceptance components (Sabar et al. 2013).
- Gene expression programming to evolve a selection perturbative hyper-heuristic which is used in the HyFlex framework to generalize over 6 problem domains (Sabar et al. 2014).
- Tensor analysis of the heuristic space to improve the performance of a hyper-heuristic (Astar and Ozcan 2015).

PART 4: CHALLENGES AND DISCUSSION

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
- Set of low-level heuristics to use
  - Random hyper-heuristic
  - Iterated local search
  - Co-evolution

DISCUSSION SESSION