Solving Complex Problems with Coevolutionary Algorithms

Krzysztof Krawiec[1], Malcolm Heywood[2]
[1]Poznan University of Technology, Poland
[2]Dalhousie University, Canada

krawiec@cs.put.poznan.pl, mheywood@cs.dal.ca

http://www.sigevo.org/gecco-2016/

GECCO'16 Companion, July 20-24, 2016, Denver, CO, USA
ACM 978-1-4503-4323-7/16/07.
http://dx.doi.org/10.1145/2908961.2926989

Instructors

- **Krzysztof Krawiec** is an Associate Professor in the Laboratory of Intelligent Decision Support Systems at Poznan University of Technology, Poznań, Poland. His primary research areas are genetic programming and coevolutionary algorithms, with applications in program synthesis, modeling, image analysis, and games. Dr. Krawiec co-chaired the European Conference on Genetic Programming in 2013 and 2014, the ACM GECCO GP track in 2015 and 2016, and is an associate editor of Genetic Programming and Evolvable Machines journal. His work in the area of CoEAs includes problem decomposition using cooperative coevolution, learning strategies for Othello, Go, and other games using competetive CoEAs, and discovery of underlying objectives in test-based problems.

- **Malcolm Heywood** is a Professor of Computer Science at Dalhousie University, Canada. His has a particular interest in scaling up the tasks that genetic programming (GP) can potentially be applied to. His current research is attempting the appraise the utility of coevolutionary methods under non-stationary environments as encountered in streaming data applications, and coevolving agents for single and multi-agent reinforcement learning tasks. In the latter case the goal is to coevolve behaviours for playing soccer under the RoboSoccer environment (a test bed for multi-agent reinforcement learning). Dr. Heywood is a member of the editorial board for Genetic Programming and Evolvable Machines (Springer). He was a track co-chair for the GECCO GP track in 2014 and a co-chair for European Conference on Genetic Programming in 2015 and 2016.

Agenda

I. Introduction

Canonical assumptions made by EA

- An absolute measure of fitness is available and computable.
  - ‘Complete’ definition of task / environment
- Solutions are (more or less) monolithic.
  - Each individual encodes a complete solution to a problem
  - Tasks are not explicitly decomposed.
- Coevolutionary algorithms (CoEA) revise these assumptions.

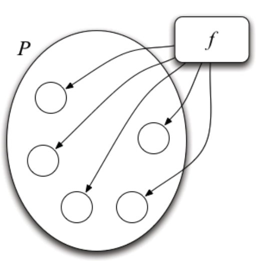
What is a coevolutionary algorithm?

- A variant of EC where fitness function mandates the individuals to engage into direct interactions.
  - Fitness cannot be computed for isolated individuals.
- Formally:
  - Evolutionary algorithm (EA): $f: X \rightarrow E$
  - Coevolutionary algorithm (CoEA): $f: X_1 \times X_2 \times \ldots \times X_n \rightarrow E$, where E is an evaluation codomain (typically R)
  - Interaction = a tuple from $X_1 \times X_2 \times \ldots \times X_n$

EA vs. CoEA

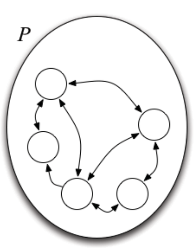
EA

Absolute measure of fitness f available and computable for each individual separately.



CoEA

Search gradient can be obtained only by letting individuals interact. Exact fitness may be not computable.



Consequences

- Individuals' performances depend on each other (fitness is contextual)
- The solution of a problem can be:
  - An element of X_i (as in an EA)
    - Typical for *competitive CoEA* (with exceptions)
    - Key questions: What to evolve against? Who is the best teacher?
  - A combination of elements from X_is
    - Typical for *cooperative CoEA* (with exceptions)
    - Key questions: How to encourage cooperation? Divide and conquer.
- Pertains to so-called solution concepts, see later
- Remember: individual ≠ solution

What is it good for?

- CoEAs lend themselves conveniently to a few classes of problems of theoretical and practical interest.
- Competitive CoEAs: test-based problems, games, interactive domains
  - Example: individual=game strategy, fitness=expected game outcome
- Cooperative CoEAs: problem decomposition, modularity, credit assignment
  - Example: individual=a rule in a classifier, fitness=overall accuracy of the classifier
- Class of problems: co-search, co-optimization, generalised optimisation (Wolpert and Macready 2005)

Other characteristics of CoEAs

- Operate under incomplete information (uncertainty)
- Focus on evaluation and interaction schemes (less so on search operators)
- Individuals often maintained in several populations.
- Biological analogs:
  - No global, static fitness function in Nature
  - Nature does not optimize for anything; EAs do.
  - Individual's fitness results from its interactions with environment, including other individuals of the same species

Measuring progress: Subjective vs. objective fitness

- Subjective fitness: f calculated using the currently available elements of X_is (a sample)
  - Typically those available in the current population,
  - Example: average game outcome against the opponents from the current population
- Objective fitness: f calculated with the elements chosen in a principled manner.
  Examples:
  - Average game outcome against all possible opponents
  - Game outcome against a human-crafted opponent.

II.1. Competitive coevolution

Class of problems tackled by competitive CoEAs

- Interactive domains
  - Sets of individuals (entities*)
  - Interaction function (payoff function) $g: X_1 \times X_2 \times \ldots \times X_n \rightarrow R$
  - When n=2, the second argument is an opponent.
- Note: g alone does not define the search goal.
- What is the solution to the problem?

(*) Sometimes, but not always, identified with candidate solutions

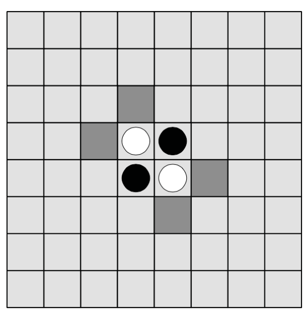
- Solution concept (cf. Ficici 2004, Popovici et al. 2012):
  - Criterion specifying whether a potential solution
    - is better than another solution (in co-optimization),
    - is solution to a problem (in co-search)
- Most popular SC: Maximization of Expected Utility (MEU): $f_o(x) = E[\, g(x_1, x_2)\,]$
  - A.k.a. generalization performance (Chong et al. 2008)
- Competitive CoEAs realize knowledge-free approach to solving problems posed in interactive domains.

Subjective fitness

- Challenge: calculation of f_o is often computationally infeasible.
  - Example: Othello: game tree complexity 10^{58}
  - Number of unique strategies typically much higher, due to many-to-one genotype-phenotype mapping
- Solutions:
  - 1. Fix the set of opponents.
    - For instance, well-performing known opponents (e.g., handcrafted by humans)
    - Strong bias, limited generalization
  - 2. Draw the opponents at random
    - What is the 'right' distribution of opponents?
    - Drawing uniformly in the genotypic space does not result in desired (e.g., uniform) distribution of skills/capabililties
  - 3. Competitive coevolution

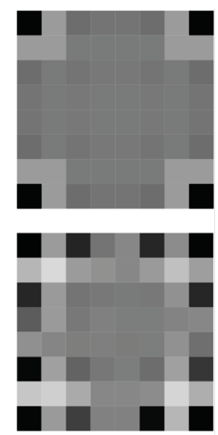
Example: Game of Othello

- Two-player, perfect-information, turn-based, zero-sum game
  - Still unsolved
  - Sudden changes of game state possible
- Strategy = individual (candidate solution)
- Common competitive CoEA approach:
  - Evolve board evaluation function $b()$
  - Use it in one-ply search: simulate all legal single moves from the current state and choose the one that maximizes b.
- Popular representations of board evaluation functions: weighted piece counter and n-tuples



Weighted Piece Counter (WPC)

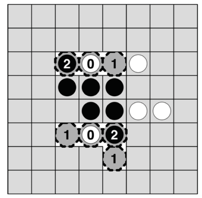
- Single linear neuron with 64 weights: $b(s) = \Sigma_i\, w_i s_i$
- Top: handcrafted Othello WPC board evaluation function (*standard WPC heuristics*)
- Bottom: a function evolved using one-population competitive CoEA, hybridized with TDL (Szubert, Jaśkowski, Krawiec 2009)

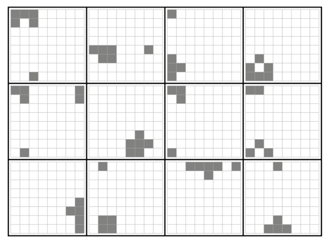


N-tuple networks

(Lucas 1997)

- Combinatorial network with lookup tables holding all combinations for (usually randomly selected) subsets of (usually adjacent) board locations
- 3^n weights for a single *n*-tuple for tri-state boards (for Othello: empty, black, white)
- Top: Exemplary 3-tuple and 4-tuple for base-3 numbers:
  - $2*3^2 + 0*3^1 + 1*3^0 = 19$
  - $1*3^3 + 0*3^2 + 2*3^1 + 1*3^0 = 34$
- Bottom: Examples of CTDL co-evolved *n*-tuples (Szubert, Jaśkowski, Krawiec, 2013)





One-population competitive CoEA

- The simplest setup to approach MEU problems.
  - Applicable when $X_1 = X_2 = \ldots = X_n = X$
  - E.g. symmetric games
  - Usually: $f_s(x) = \Sigma_{x' \in X'}\, g(x,x')$, where X' is some sample of X drawn from current population P
- An interaction = single game (symmetric games) or two games (asymmetric games)
- Interaction schemes:
  - Round-robin: $n(n-1)/2$ interactions ($X' = P \setminus \{x\}$)
  - *k*-random opponents: kn interactions ($|X'| = k$)
  - Single-elimination tournament (SET): n interactions
    - Pair the individuals at random. Winners pass to the next stage. Individual's fitness is the stage of tournament it reached.

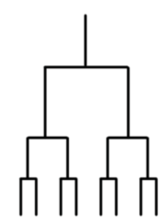
Highlights of one-pop competitive CoEAs

- Iterated Prisoner's Dilemma, IPD (Axelrod 1987)
- Backgammon (Pollack & Blair 1998)
- Checkers (Samuel 1959, Fogel 2002)
- NERO, Blackjack, Pong, Small-board GO, Tetris, …

Fitnessless Coevolution

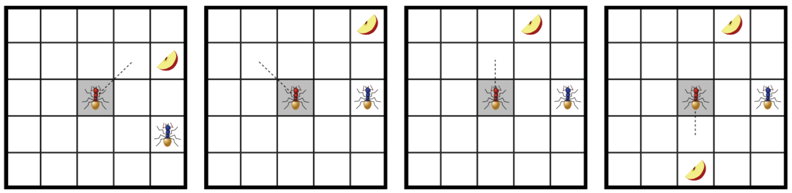
(Jaśkowski, Krawiec, Wieloch 2008)

- More specifically: fitnessless selection
  - Pick *k* individuals at random
  - Run a SET on them
  - The winner of SET is selected
- Does not rely on subjective fitness.



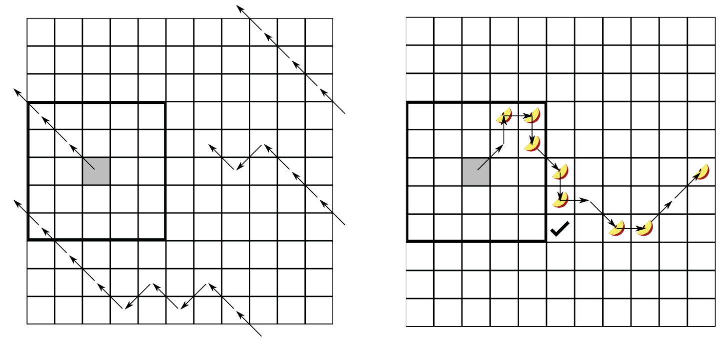
Fitnessless Coevolution for Ant Wars
(Jaśkowski, Krawiec, Wieloch 2008)

- Fitnessless Coevolution evolved the winner of the Ant Wars GECCO'08 contest
  - Two-player partially observable game
  - Agents (ants) see only a 5x5 fragment of the toroidal 11x11 board
  - The goal: collect more food pellets than the opponent (pellet locations are random).
  - Strategy representation: stateful GP program (maintains intra-game memory)



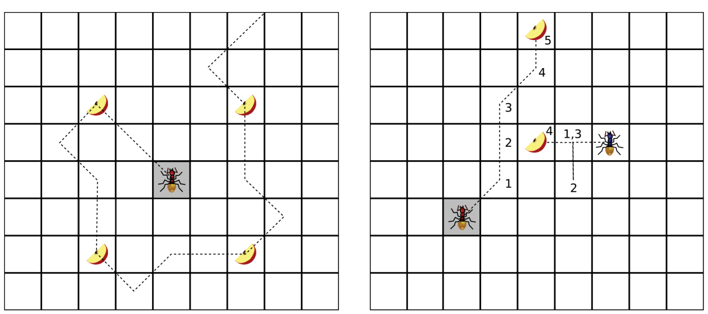
Example: Ant Wars

Complex behaviors emerged: systematic search, rational choice of trajectories, …



Example: Ant Wars

… memorizing locations of food pellets, opponent avoidance, pseudo-suicide, …



- Online demo: http://www.cs.put.poznan.pl/kkrawiec/antwars/

Digression: Importance of transitivity

- Fitnessless Coevolution is not equivalent to fitness-driven one-population coevolution if there are cycles in interactions in between individuals (Jaśkowski, Krawiec, Wieloch 2008)
- Example: Tic-tac-toe strategies A, B, C: place a mark in the numbered locations if free, otherwise in the location marked by asterisk (*)

A

1	2	3
		*

B

3		
2		
1		*

C

		1
	2	
3		*

- A beats B, and B beats C. But A does not beat C, just the opposite.
- Tic-tac-toe is intransitive.
- No scalar fitness function can model this (can realize only complete orders).

The philosophy behind one-pop competitive CoEA

- Individuals create search gradient for each other.
  - A form of (population-level) self-learning
  - Can be seen as an analog to self-play in RL (individual-level)
- Q: Is this sufficient to guarantee progress?
- A: No. Coevolutionary pathologies are lurking out there.

Münchhausen O. Herrfurth pinx

Coevolutionary pathologies

- Cycling: evolution keeps rediscovering the same solutions
  - Particularly likely if game is intransitive.
- Disengagement: opponents are either trivial or way too difficult to beat
- Overspecialization (focusing): mastering the skills of beating some opponents while neglecting the others.
- Forgetting: opponents defeated in the past turn out to be difficult again.
- See review and rigorous analysis in (Ficici 2004)
- Main causes:
  - No access to objective fitness
  - Population responsible for both search and providing search gradient for itself

Coevolutionary archive competitive CoEAs (one-population)

Archive = a container storing well-performing individuals, maintained alongside population.

- Provides long-term memory
- Builds search gradient
- Prevents some pathologies
- Maintains diversity and progress

Archives help maintaining historic progress (Miconi 2009)

- Not necessarily progress in the global, objective sense.

How it works:

- Search algorithm submits some individuals to the archive
- Archive accepts some of them
- Individuals in population interact with peers and archival individuals
- Outcomes of interactions augment the fitness
- Simplest archive: best-so-far individual
- Hall of fame (Rosin & Belew, 1997)
  - Stores all best-of-generation individuals found so far
  - Population members play against each other and against the opponents from HoF

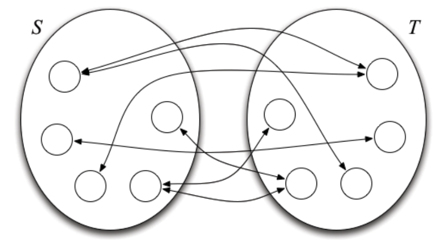
II.2. Two-population competitive CoEAs

Two-population competitive CoEAs

- One-pop competitive CoEA: Population responsible for both searching for good solutions and providing search gradient for itself.
  - Why not separate these functions?
- Two-pop competitive CoEAs: Maintain separate populations of:
  - candidate solutions $S \subset X_1$ – intended to solve the problem
  - tests $T \subset X_2$ – provide only search gradient for the individuals in S
- Applicable in symmetric ($X_1 = X_2$) and asymmetric setting ($X_1 \neq X_2$)

July 2016 Solving Complex Problems with Coevolutionary Algorithms 29

Two-population competitive CoEA



- Typical interaction scheme: all-to-all
- S and T co-evolve in parallel
- No transfer of individuals between S and T

July 2016 Solving complex problems with coevolutionary algorithms 30

What to reward the tests for?

- Individuals in S should maximize EU. How to reward the tests?
- Maximize EU as well?
  - Pathologies likely
  - Tests should be neither too easy nor to hard for the individuals in S
- Common reward schemes:
  - Distinctions: reward a test for every pair of solutions it distinguishes
  - Informativeness: reward a test for unique partitioning of S
  - Hybrids (e.g., with EU)

July 2016 Solving complex problems with coevolutionary algorithms 31

Test-based problems

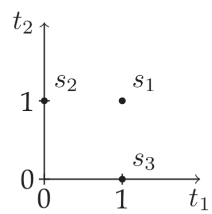
- With two populations, the tests can be conceptually different from candidate solutions.
- Formally: Test-based problem (S, T, G, Q) (Popovici et al., 2012)
- Examples:
  - Asymmetric games (strategies vs. opponents)
    - E.g., tic-tac-toe, Othello,
  - Control problems (controllers vs. initial conditions)
    - Pole balancing, car control, etc.
  - Learning from examples (hypotheses vs. examples)
  - Program synthesis with GP (programs vs. tests)
  - In general: co-optimization and co-search

July 2016 Solving complex problems with coevolutionary algorithms 32

Pareto-coevolution

(Ficici and Pollack, 2001; Noble and Watson, 2001)

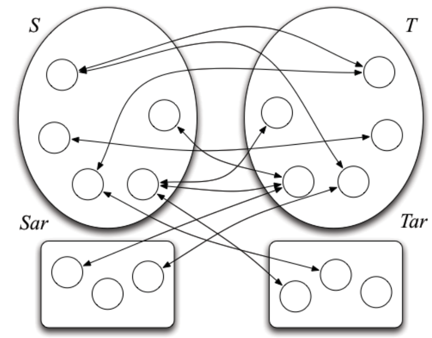
- Each test considered as a separate objective.
- Transforms a test-based problem into multiobjective optimization problem (or many-objective one).
- Example:
  - s_1 solves both tests t_1 and t_2
  - s_2 solves only t_2
  - s_3 solves only t_1



- Problem: large number of tests (and thus objectives).
- Sparse dominance relation.

Coevolutionary archives (two-pop)

- General scheme: individuals are submitted to archive and get accepted or rejected by it.
- Separate archives for solutions and tests



Coevolutionary archive algorithms (two-pop)

- Iterated Pareto-Coevolutionary Archive, IPCA (de Jong 2004)
  - A new solution s is added to S_{ar} if no $s' \in S_{ar}$ dominates s. In that case:
    - All $s'' \in S_{ar}$ dominated by s are removed from S_{ar}
    - The test t that made it possible for s to be added to S_{ar} is added to T_{ar}
  - Guarantees monotonous progress
  - Unlimited-size archive
  - Tests provide for distinctions between individuals
- Layered Pareto-Coevolutionary Algorithm, LAPCA (de Jong 2004)
  - Merges the current archive and the submitted elements and builds a Pareto ranking of solutions
  - The first k layers of the ranking remain in S_{ar}, the remaining ones are discarded
  - T_{ar} keeps the tests that support Pareto dominance in S_{ar}
  - No guarantee of monotonous progress, but (somehow) controllable size
- IPCA and LAPCA perform well only on small, usually artificial problems.

Coevolutionary archives

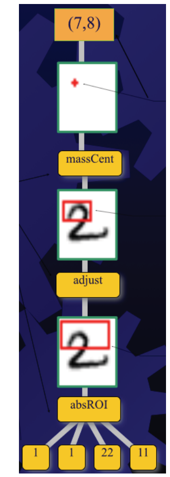
- Maintaining archives can be costly
  - Many interactions required to check if a solution should be added
- Mitigation: MaxSolve (De Jong 2005), for MEU solution concept
  - Keep in S_{ar} up to n solutions that solve the most tests (at least one), and in T_{ar} all tests that a solved by at least one $s \in S_{ar}$
  - [Behaviorally] duplicate tests are discarded
  - Monotonic: will not miss solutions that increase the number of solved tests
- When overhead of maintaining an archive counted in, non-archived algorithms can be equally efficient.
- Other types of archives (Jaśkowski & Krawiec 2010)

Related results and concepts

- Ideal evaluation and complete evaluation set (de Jong and Pollack 2004)
  - The set of tests that preserves dominance relation between the solutions in *S*
  - Determining the minimal complete evaluation set is NP hard (Jaśkowski & Krawiec 2011)

Genetic Programming: Program synthesis as a test-based problem

- Genetic programming
  - *S* = population of candidate programs
  - *T* = population of tests (fitness cases)
- Simple variant: Pairwise Comparison of Hypotheses (Krawiec 2001)
  - Dominance-based selection of hypotheses
  - Dominance-based maintenance of best solutions
  - Dominance-based selection of the best solutions (algorithm outcome)
- Applied to handwritten character recognition
- See also: (Arcuri & Yao 2014)

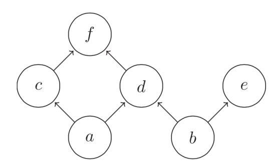


II.3. Advanced topics in competitive coevolution

Hybridization, coordinate systems, coevolutionary shaping

Coordinate systems

- An interaction matrix defines a dominance relation
- Dominance relation defines a partial order in the set of individuals ⇒ partially ordered set, *poset*

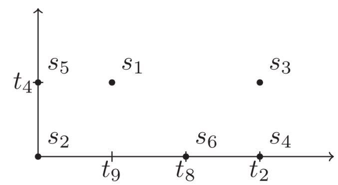


- A poset can be 'stretched' along multiple dimensions (underlying dimensions).
- Dimensions form a coordinate system (Bucci et al. 2004):
  - Axis = ordered list of tests
  - (alternative formulations exist)

Coordinate system: an example

- The game: Nim-1-3
  - Players in turns take sticks from two piles of size 1 and 3.
- Total of 144 strategies,
  - but only 6 behaviorally unique for the first player (*S*), and 9 for the second player (*T*).
- Minimal coordinate system
  - Some tests not needed to reproduce the dominance relation
- Game dimension: 2

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
s_1	1		1	1		1	1		1
s_2									
s_3	1	1	1	1	1	1	1	1	1
s_4	1	1	1				1	1	1
s_5	1			1			1		
s_6							1	1	1

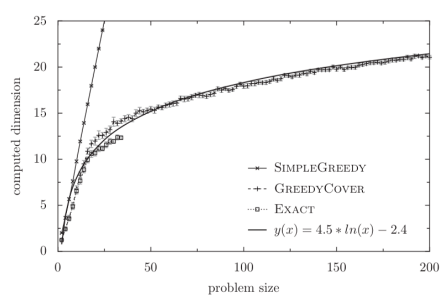


Coordinate systems: some results

- Benefits:
  - Can accelerate convergence and/or guarantee progress: Dimension Extraction Coevolutionary Algorithm, DECA (de Jong and Bucci 2006)
  - Reveal the internal structure of a problem and relate to problem difficulty
- Hypothesis: dimensionality of coordinate system is a yardstick of problem difficulty
- The set of all tests forms the complete evaluation set (de Jong & Pollack 2004)
- Game dimension = width of the poset (Jaśkowski & Krawiec 2011)
- The number of underlying objectives for an abstract problem seems to be limited by a logarithm of the number of tests.

Problems with exact coordinate systems

- Problem dimension may be underestimated when only samples of *S* and *T* are used.
- Finding minimal CS for a problem is NP-hard (Jaśkowski & Krawiec 2011)
- Heuristics exist but overestimate the number of dimensions
- Nontrivial test-based problems have very high dimensionality
- Q: Can we efficiently 'approximate' the underlying dimensions?



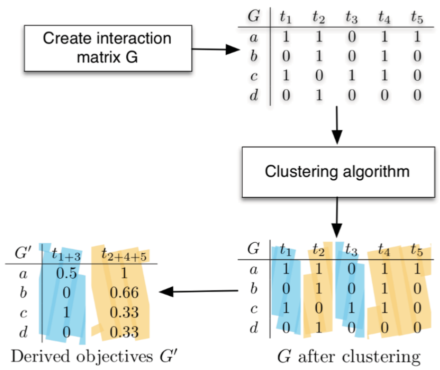
Heuristic discovery of underlying objectives

- Idea:
  - Construct efficiently approximate underlying objectives from the information available at the given stage of search process
  - Use the derived objectives in multiobjective EA setting
- Derived objectives rather than underlying objectives
  - Approximate (do not reproduce the original dominance)
  - Transient (depend on the current populations)
- Technical means: clustering of tests

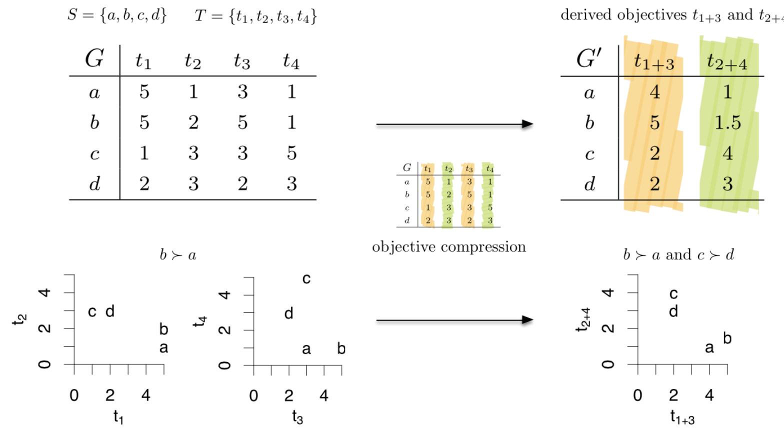
Heuristic discovery of underlying objectives

(Krawiec & Liskowski 2015, Liskowski & Krawiec 2016)

- ‘Batch evaluation’ of population (as in implicit fitness sharing)
- Example: four candidates: $S = \{a,b,c,d\}$, five tests: $T = \{t_1,t_2,t_3,t_4,t_5\}$
- No guarantee to reproduce the original dominance relation.
- ‘False positive’ dominance possible.
- ‘False negative’ – impossible.

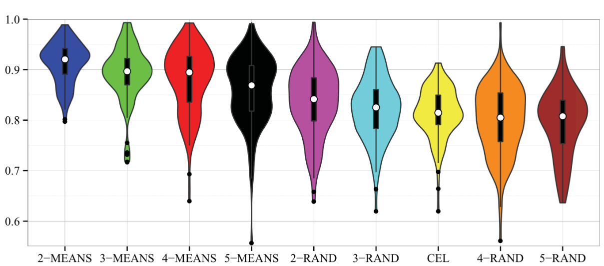


Heuristic discovery of underlying objectives



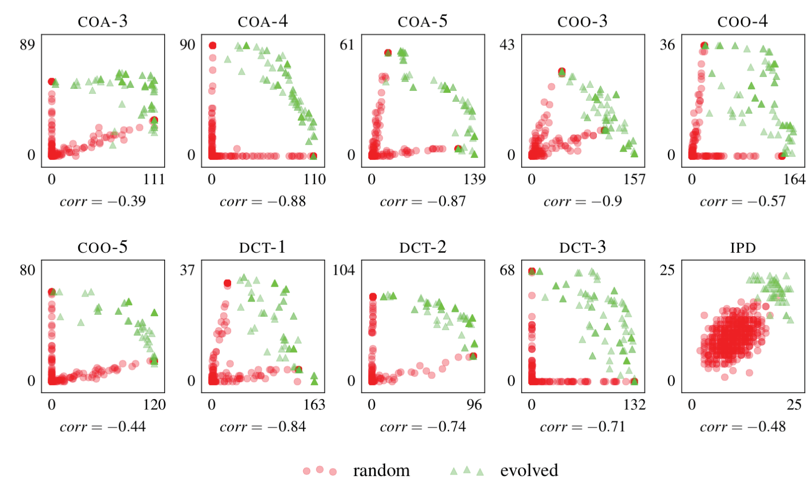
Heuristic discovery of underlying objectives

- Results for 9-choice iterated prisoner’s dilemma, IPD (MEU)
  - *k*-MEANS: k objectives derived using *k*-means clustering algorithm
  - *k*-RAND: objectives built by random partitioning of tests into k objectives
- Applied also in non-coevolutionary setting with GP, with k adjusted automatically (Krawiec & Liskowski 2015). Better than GP and RAND, comparable to IFS.



Heuristic discovery of underlying objectives

(Liskowski & Krawiec 2016)



Hybridization

- CoEAs are generate-and-test techniques (like EA)
  - In contrast, gradient-based methods provide 'directed' corrections/updates of parameters
  - Can be more efficient in high-dimensional problems
  - Complementary: CoEAs learn slower than TDL but eventually outperform it (Lucas & Runarsson 2006)
- Coevolutionary Temporal Difference Learning, CTDL (Krawiec & Szubert 2011, Szubert et al. 2013)
  - Interleave one-population coevolution (with round-robin) with TD(0)
  - CoEA picks the 'right' opponents, TDL tunes the solutions in a self-play mode
  - CoEA modifies the topology of n-tuples. TDL only affects the weights.
- A form of memetic algorithm (genetic local search) (Moscato 1989): individuals' interactions with the environment influence their genotypes (Lamarckian evolution).
- Related to: adversary reinforcement learning

Hybridization

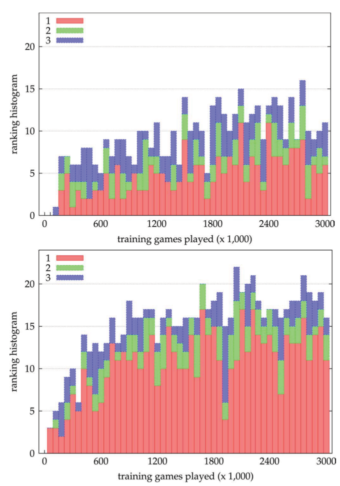
- Othello, *n*-tuples (Szubert, Jaśkowski, Krawiec 2013)
  - Compared also to ETDL= EA+TD(0)
- Othello Evaluation Function League
  - http://algoval.essex.ac.uk:8080/othello/html/Othello.html
  - Ranked according to average performance against so-called standard heuristic WPC (handcrafted strategy; moves partially randomized) (as of 2011)
- Players evolved by ETDL ranked higher than those produced by CTDL. Why?

Othello League Ranking

Name	Size	Played	Won	Drawn	Lost
epTDLmpx_12x6	12 × 6	100	89	1	10
prb_nt30_001	30 × 6	100	84	0	16
prb_nt15_001	15 × 6	100	83	3	14
epTDLxover	12 × 6	100	81	4	15
t15x6x8	15 × 6	100	79	3	18
SelfPlay15	12 × 6	100	77	0	23
tz278_2	278 × 2	100	76	3	21
Nash70	12 × 6	100	72	4	24
x30x6x8	30 × 6	100	71	4	25
pruned-pairs-56t	56 × 2	100	71	1	28

Hybridization: EA vs. CoEA

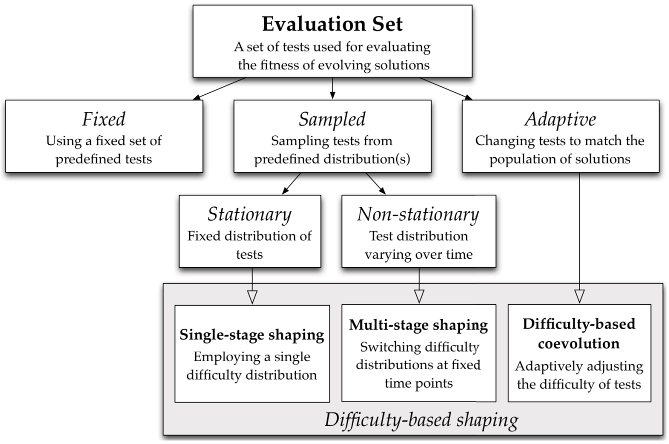
- Right: distribution of ranks obtained by ETDL (top) and CTDL (bottom) best-of-generation individuals in a round-robin competition with 24 top Othello League players.
  - ETDL better on predefined opponent (heuristic WPC)
  - CTDL better in face-to face confrontation with other opponents
- ETDL overfits on the WPC
- CTDL:
  - produces more versatile players
  - scales well with the number of parameters
  - effective interplay of combinatorial evolutionary search and gradient-based search in continuous space of *n*-tuple weights.



Coevolutionary shaping

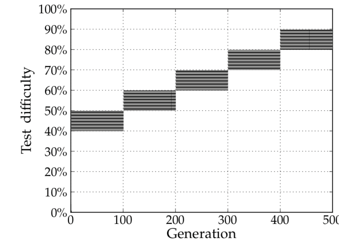
- Shaping = key concept in behavioral psychology (Skinner 1938)
  - Expose the learner to a series of training episodes of gradually increasing difficulty.
  - Motivation: Tasks can be too difficult to learn autonomously.
  - Example: To train a pigeon to strike a ball, first reward looking at it, then approaching it, and only then striking the ball with the beak.
- Used with success in Reinforcement Learning, e.g. pole balancing (Selfridge 1986)
  - Simplified version of tasks generated by relaxing/parameterizing the original one
  - E.g. change the length of the pole, increase the mass, etc.
- Related to: incremental evolution, staged evolution, environmental complexification
- Requires human intervention (decide how to relax the tasks, order them, etc.)

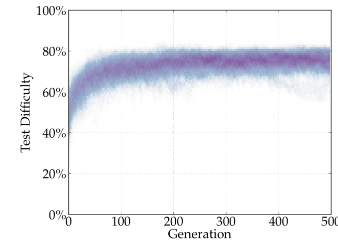
Coevolutionary shaping
(Szubert 2014)



Coevolutionary shaping

- ❖ Coevolution can be seen as a form of autonomous shaping
  - ❖ Training experience = the sequence of tests to interact with
- ❖ What should be the gauge to decide how to form the training experience?
- ❖ Test difficulty: (exact or estimated)
  - ❖ $d(t) = \Sigma_{s \in S} (1 - g(s,t))$
- ❖ Top chart: manual shaping ($d(t)$ ×100%).
- ❖ Bottom: coevolutionary shaping: distribution of test difficulty in a coevolving population of tests (Othello, WPC) (Szubert et al. 2013)
- ❖ Coevolutionary shaping works as well as the manual shaping, but requires less parameter tuning.





Competitive Coevolution: Take-home messages

- ❖ Population of tests (and archives) accumulate potentially useful knowledge about a problem
- ❖ Coordinate systems = a means of widening the ‘evaluation bottleneck’ and making search algorithm better-informed
  - ❖ Other means to opening the bottleneck exist (in GP: semantic GP, behavioral GP)
- ❖ Competitive CoEAs tend to overspecialize on the stronger opponents while forgetting how to deal with the weaker ones
- ❖ Importance of diversity (in particular diversity of tests)
- ❖ A competitive CoEA can guide itself towards the optimum more efficiently

Not covered in this tutorial

- ❖ Measuring and visualizing progress (e.g., CIAO plots)
- ❖ Artificial problems: number games. Strategies represented as vectors of n elements.
  - ❖ Compare-on-all: A solution wins if it is better on all elements
  - ❖ Compare-on-one: a test picks a dimension at random; the solution wins if it’s greater on that dimension
- ❖ Other solution concepts (Ficici 2004, Poppovici et al. 2011)
  - ❖ Simultaneous maximization of all outcomes, Nash equilibrium, Pareto-optimal set, Algorithms: (Ficici 2004) and review in (de Jong 2005)
- ❖ Deciding upon the final outcome of a CoEA: “output mechanism” (Popovici and Winston 2015)
- ❖ Random Sampling Evolutionary Algorithm (Chong et al. 2008) - no true coevolution, but hard to beat using competitive CoEAs.
- ❖ Coevolutionary free lunches (Wolpert & Macready 2005; Service and Tauritz 2008; Popovici and Winston 2015)
- ❖ Hybridization with CMA-ES (Jaśkowski & Szubert, 2015)

III. Cooperative Coevolution

Cooperative Coevolution

- Answers the question:
  - How to encourage collaboration?
- Metaphor:
  - Divide and conquer!
- Why (is it useful?): Promoting modularity / reuse
  - additional clarity in: (relative to a monolithic solution)
    - credit assignment
      - search space projected into multiple smaller search spaces
      - agents do not need to solve all the task
    - solution transparency
    - capacity to react to changes (Simon's parable of the two watch makers)
- Fitness: who to credit for what?
  - generalist pathology:
    - individuals rewarded for maximizing the number of collaborations
    - stable / mediocre solutions rather than optimal solutions

A Metaphor…

- "species [individuals] represent solution components. Each individual forms a part of a complete solution but need not represent anything meaningful on its own. The components are evolved by measuring their contribution to complete solutions and recombining those that are most beneficial to solving the task." [Gomez et al., (2008)]
- Central questions
  - How to:
    - compose a candidate solution (team)
    - distinguish between credit to the team versus that to team members
    - balance the exploration / exploitation tradeoff
      - Learning **context**

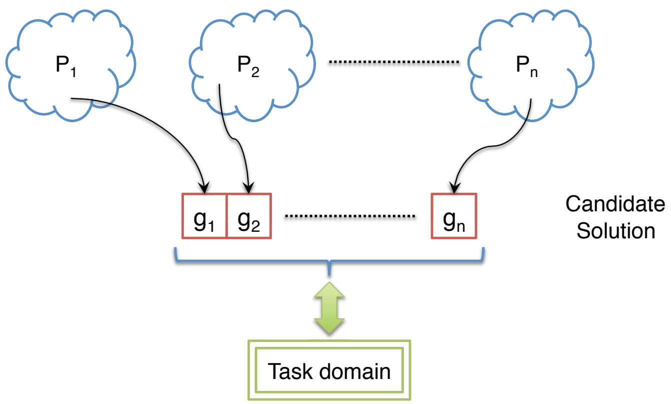
Cooperative Coevolution for complex systems : Some milestones

- Neural Networks
  - Moriarty, Miikkulainen (1998)
  - Potter & de Jong (2000)
  - Gomez et al. (2008)
  - Gomes et al. (2016)
- Genetic Programming
  - Krzystof & Bhanu (2006, 2007)
  - Thomason & Soule (2007), Rubini et al. (2009)
  - Lichodzijewski & Heywood (2008)
  - Wu & Banzhaf (2011)
- Formulating fitness functions
  - Panait et al. (2006, 2008)
  - Agogino & Tumar (2008), Knudson & Tumar (2010)
- Diversity maintenance
  - Lichodzijewski et al. (2011)
  - Doucette et al. (2012)
  - Kelly & Heywood (2014)
- Non-stationary tasks
  - Agogino & Tumar (2008)
  - Vahdat et al, (2015)
- Reinforcement Learning
  - Moriarty & Miikkulainen (1998)
  - Gomez et al. (2008)
  - Agogino & Tumar (2008), Knudson & Tumar (2010)
  - Rubini et al. (2009)
  - Doucette et al. (2012)
  - Kelly & Heywood (2014, 2015)

Cooperative Coevolution: An architecture

(Potter & De Jong, 2000)

Prior decomposition of the solution into ‘n’ independent populations (species)



Biased and Lenient cooperation

(Panait et al., 2006), (Panait et al., 2008)

Biased cooperation

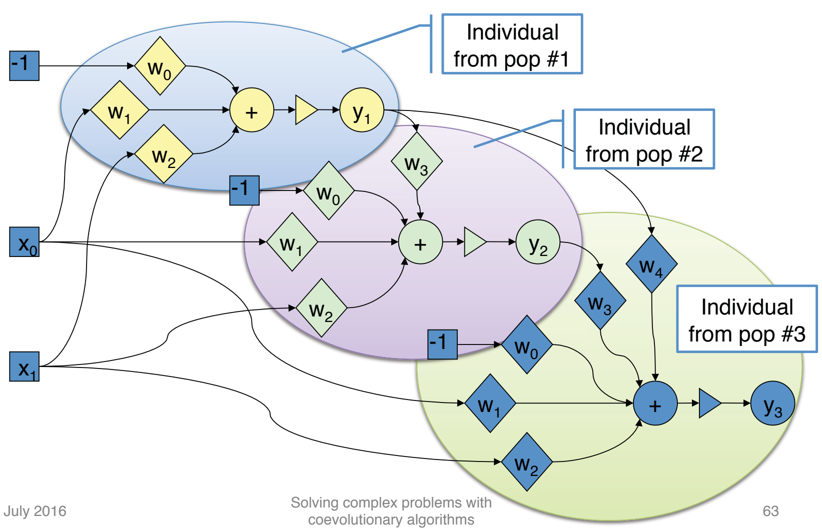
- Consider team versus individual fitness
  - Individuals receive avg. of fitness from teams
  - Promotes generalists
  - Hitchhiking
- Recommend defining individual fitness as
  - an *optimal* team of collaborators
  - Not clear how an *optimal* collaborator set is found in the general case

Lenient cooperation

- Individual fitness
  - $MAX_{i \text{ in team}}$ (team$_i$ fitness)
  - Hitchhiking still exists
  - Is hitchhiking all negative?
    - Enables individuals to find their niche
    - Provides a memory of previous / alternative policies

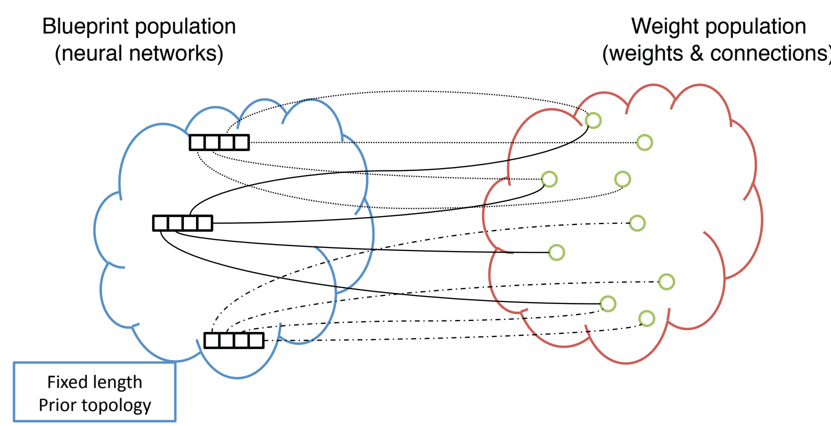
Coevolving a cascade network

(Potter & De Jong, 2000)



SANE with blueprints

(Moriarty & Miikkulainen, 1998)



Difference evaluation functions

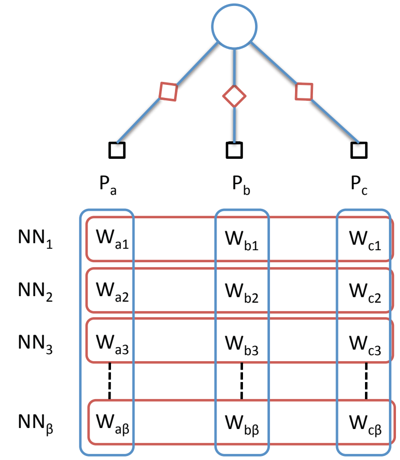
(Agogino & Tumar, 2008), (Knudson & Tumar, 2010), (Codly & Tumar, 2012)

- Global fitness
  - Performance of entire collective
  - Difficult to identify the contribution from each agent
- Local fitness
  - Performance of single agent
  - Difficult to encourage non-overlapping collective behaviours
- Difference evaluation function (D_i)
  - Explicitly estimate value added by agent 'i'
  - Global fitness needs to be locally 'decomposable'
  - Agents assigned w.r.t. physical locality to distributed sub-tasks
  - Form of 'spatial embedding'
- D_i formulation
  - $D_i = G(s) - G(s_{-i} + C_i)$
- G(s)
  - G(•) is the global evaluation function
  - 's' state of the system
- s_{-i}
  - States for which agent 'i' have no contribution
- C_i
  - Default vector of constants
- Observations
  - In the worst case s_{-i} is empty
    - Agent 'i' impacts on all states
  - D_i directly expresses the impact of agent 'i' not present
  - Limited by capacity to design appropriate `difference' expression

Cooperative Synapse NeuroEvolution

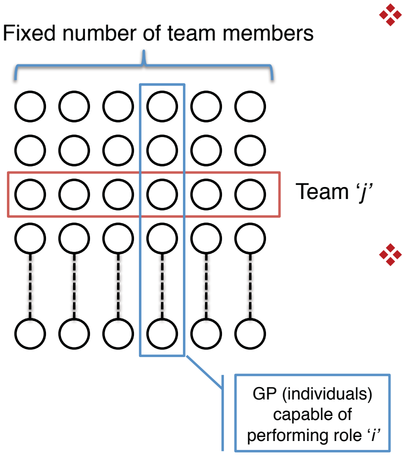
(Gomez et al., 2008)

- Select Parents
  - NNs (say, top 25%)
- Variation
  - 75% children
- Sort $\mathbf{P_i}$ w.r.t. $f(\boldsymbol{w_{ij}})$
  - $\mathbf{P_i}$: $f(\boldsymbol{w_{i1}}) > f(\boldsymbol{w_{i2}}) > \ldots f(\boldsymbol{w_{i\beta}})$
- Stochastic permutation of $\mathbf{P_i}$ content
  - $\mathbf{P_i}$: $f(\boldsymbol{w_{i1}})\ f(\boldsymbol{w_{i2}}) \ldots f(\boldsymbol{w_{i\beta}})$



Orthogonal evolution of (GP) teams (1)

(Thomason & Soule, 2007), (Rubini et al., 2009)



- Motivation
  - Team selection:
    - Good cooperation
    - Poor individual fitness
  - Island (individual) selection:
    - Poor cooperation
    - Strong individual fitness
- OET1 (OET2)
  - Select w.r.t individuals (teams)
  - Replace w.r.t. teams (individuals)

Orthogonal evolution of (GP) teams (2)

(Thomason & Soule, 2007), (Rubini et al., 2009)

OET1

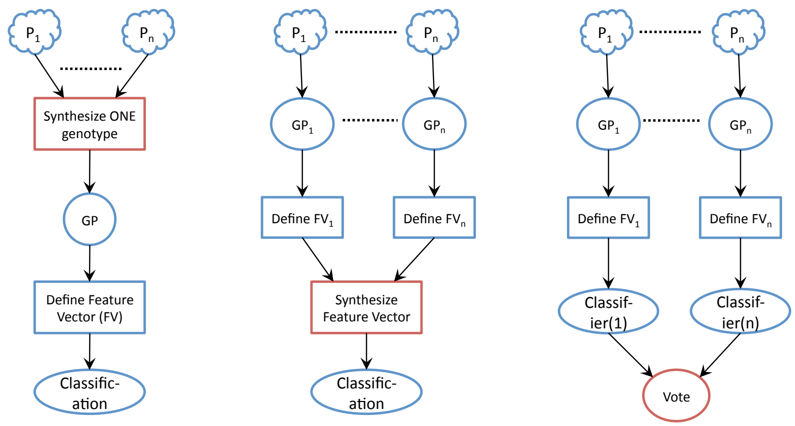
- Team = NULL
- *Select* best individual per role
- Create 2 such teams
- Apply variation operators
- Evaluate fitness
- *Replace* worst teams

OET2

- *Select* 2 best teams
- Apply variation operators
- Evaluate fitness
- Award fitness to individuals in same team
- *Replace* weakest individuals

Level of Decomposition

(Krawiec & Bhanu, 2005), (Krawiec & Bhanu, 2007)

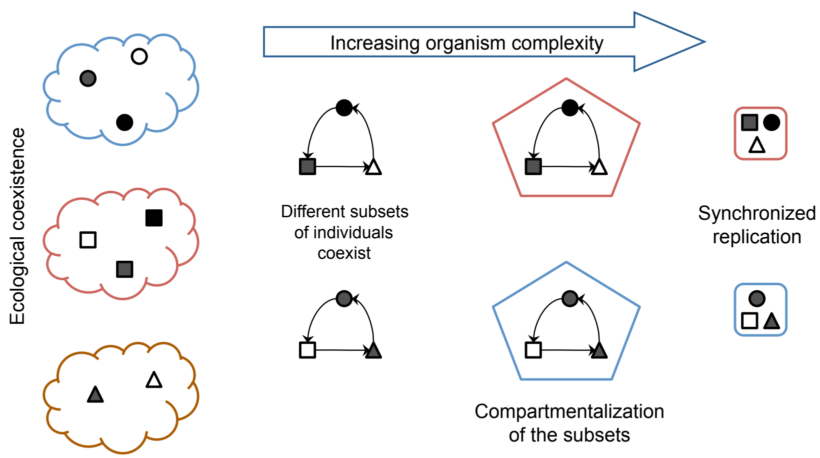


III.1 Case Study – Symbiotic bid-based GP

Variable GP teams, diversity maintenance, and separating action from context

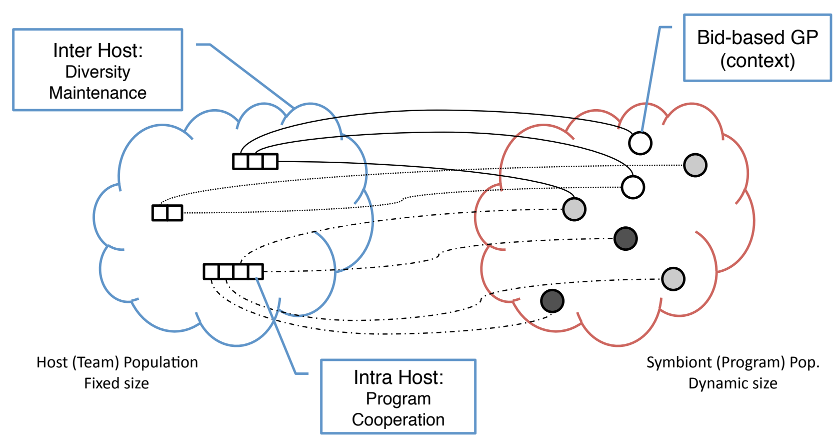
Abstract Model of Symbiosis

(Maynard Smith, 1991)



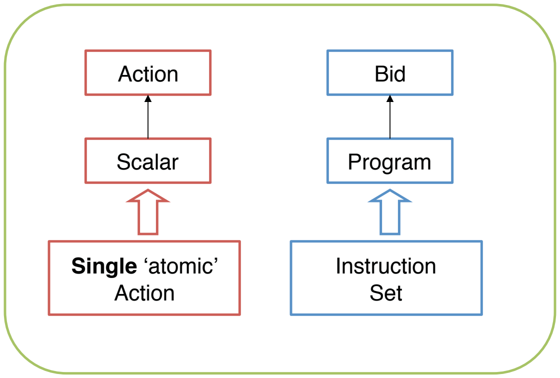
Symbiotic Bid-Based GP (SBB)

(Lichodzijewski & Heywood, 2008, 2010), (Lichodzijewski et al., 2011)



Achieving Symbiont Context

Bid-based GP



Bid-based GP individual

Host Fitness

- Outcome vector, G(•)
  - Point (p(k)) to Host (h(i)) Outcome

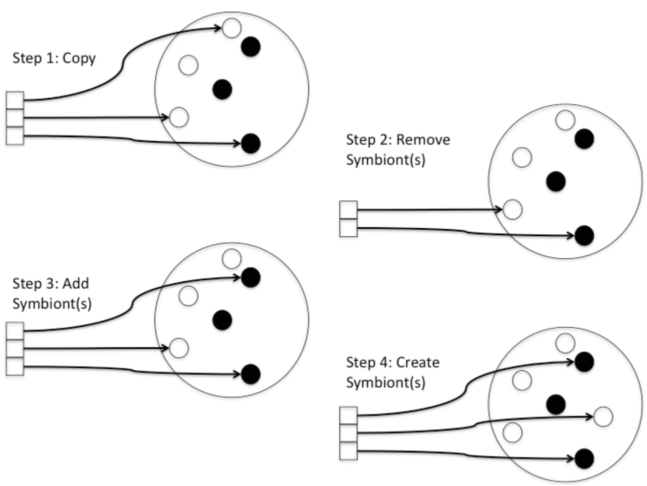
G(h(i), p(k)) =
- **Real valued reward (how close to target)**
- Domain specific

- Inter Host Diversity Maintenance
  - Fitness sharing (see also behavioural and novelty measures)

$$s_i = \sum_k \left(\frac{G(h_i, p_k)}{\sum_j G(h_j, p_k)} \right)^3$$

Asexual Reproduction

Species independence



III.2 Case Study – SBB under non-stationary streams

Supporting Evolvability / Plasticity through Cooperative Coevolution

Non-stationary Streaming data
(Vahdat et al., 2015)

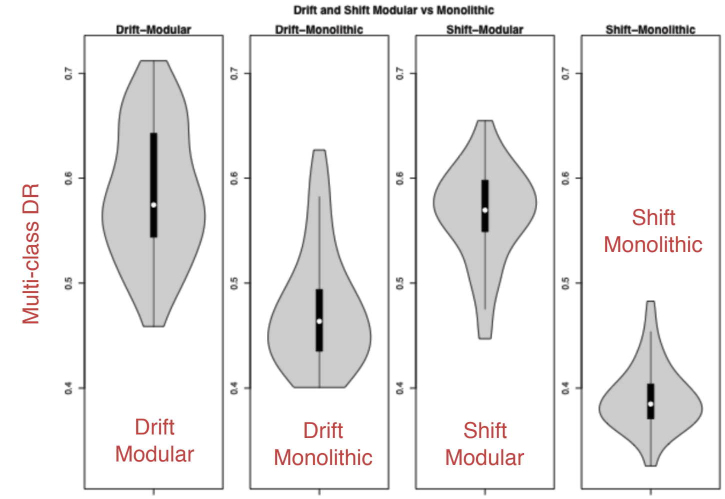
Drift – 'gradual' variation

- 150,000 exemplars over stream
- Window interface
  - 500 window locations
  - 20 exemplars sampled per window location
- 10 attributes
- 3 classes
  - 16%, 74%, 10%

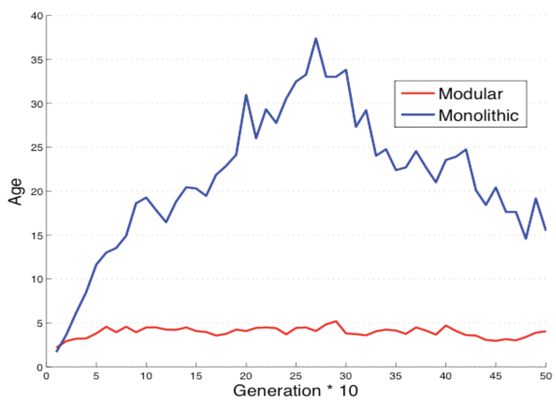
Shift – 'sudden' variation

- 6.5 million exemplars over stream
- Window interface
  - 1,000 window locations
  - 20 exemplars sampled per window location
- 6 attributes
- 5 classes
  - 36%, 49%, 6%, 0.5%, 1.5%, 3%, 4%

Accumulated multi-class detection rate
(Vahdat et al., 2015)

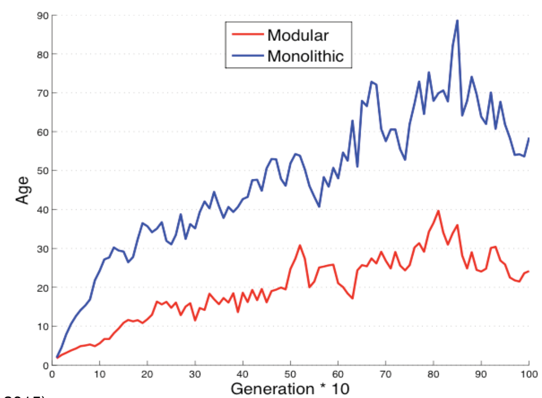


Age of champion individual
During course of stream – Drift



(Vahdat et al., 2015)

Age of champion individual
During course of stream – Shift



(Vahdat et al., 2015)

Observations

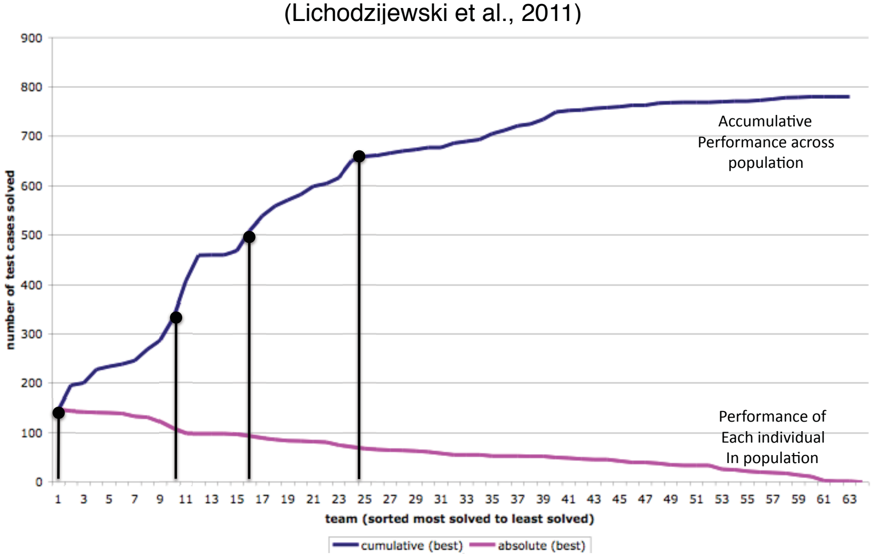
- Context for the symbiont programs must be evolved
  - Bidding mechanism
- Support for problem decomposition
  - Mix of symbiont programs per host an evolved trait
  - Inter host diversity encourages decomposition at host level
    - No prior knowledge on the nature of an appropriate decomposition
    - Provides capacity for reacting to change
- Lower 'age' of champion
  - Easier to switch in / out functional non-functional symbionts as contexts change

III.3 Case Study – Diversity maintenance and Policy reuse

Hierarchical organization of programs, program abstraction

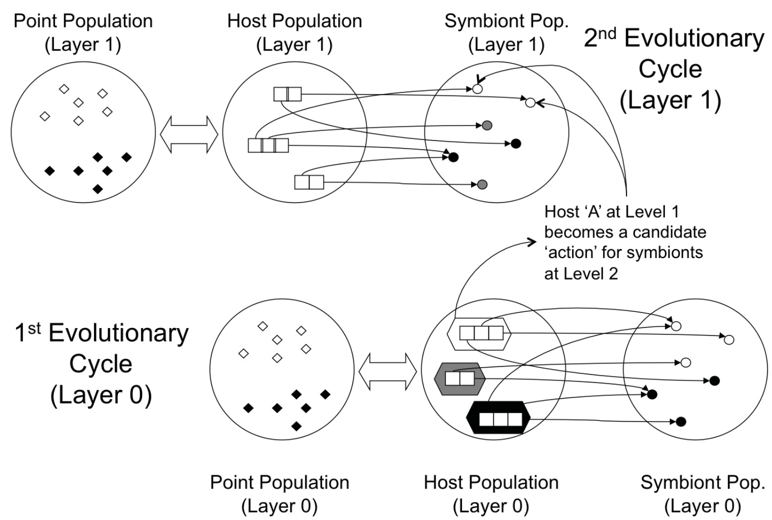
Motivation – Population fails in task

(Lichodzijewski et al., 2011)



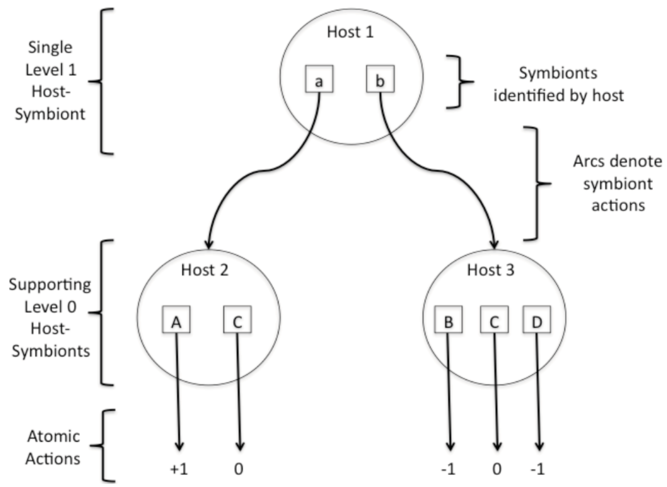
Evolving a policy tree

(Lichodzijewski et al., 2011), (Doucette et al., 2012), (Kelly & Heywood 2014, 2015)



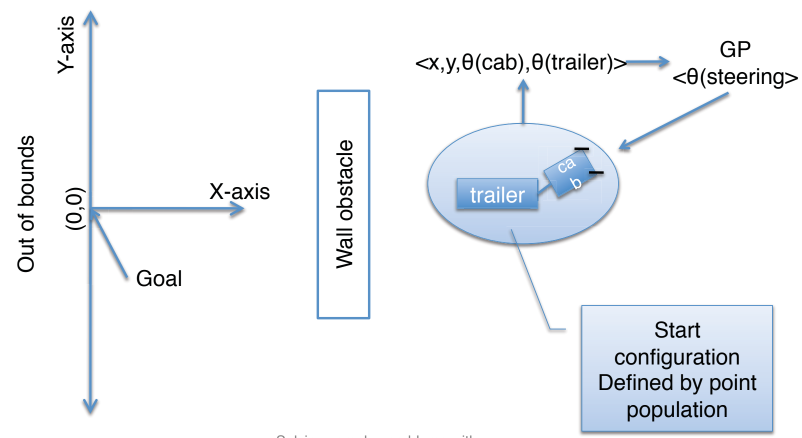
Evaluating a policy tree

(Lichodzijewski et al., 2011), (Doucette et al., 2012), (Kelly & Heywood 2014, 2015)



Hidden State Truck Backer-upper

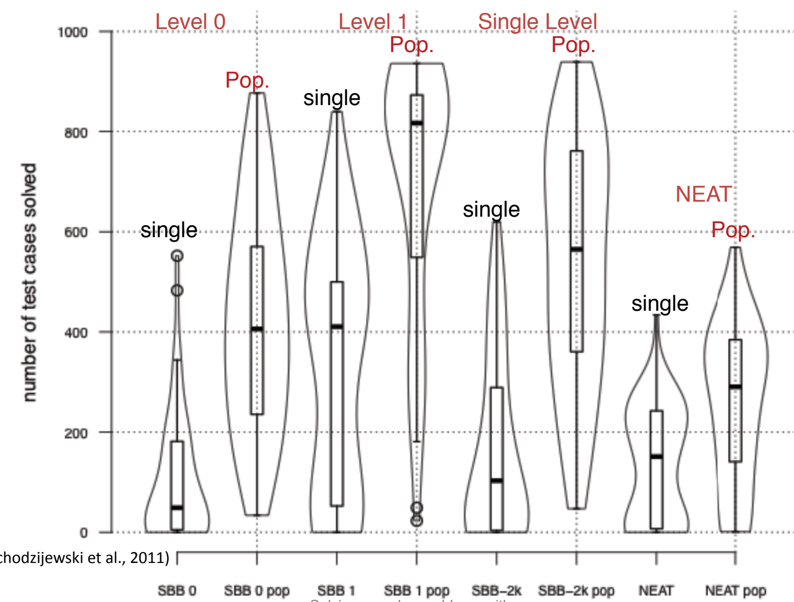
(Lichodzijewski et al., 2011)



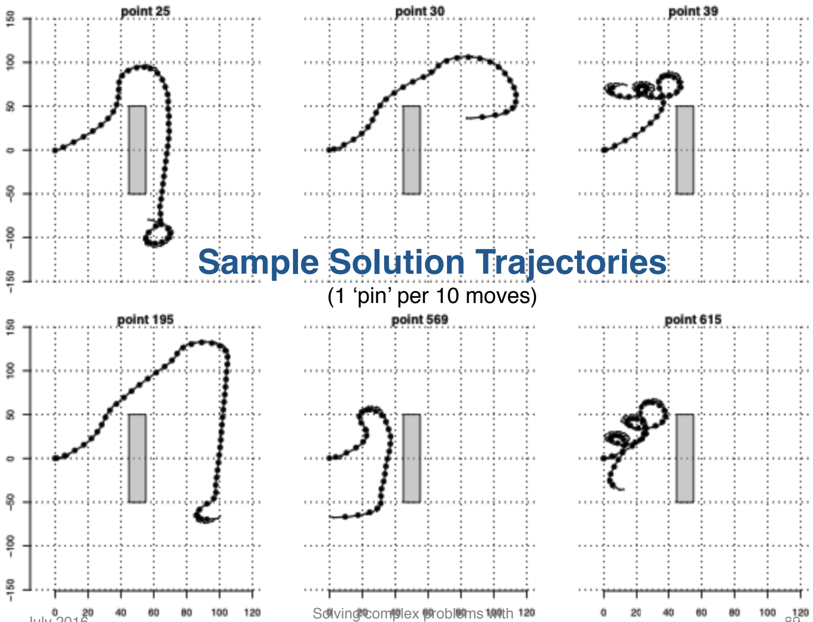
Parameterization

(Lichodzijewski et al., 2011)

- SBB
  - Max. Eval.: 16,800,000
    - 8,400,000 per layer
  - Max Host Size: 10
  - Host Pop.: 120
  - Host Gap: 60 (50% turnover)
  - (12 other parameters)
- Single layer SBB config.
  - 16,800,000 gen over 1 layer
  - Double Max host size
- SBB (generic)
  - Instruction set:
    - {+, −, ×, ÷, cos, ln, exp, if R[x] < R[y] THEN sign(R[x])}
- NEAT
  - Max. Eval.: 16,812,000
  - NN Pop.: 150
  - (17 other parameters)
- Common
  - Point pop.: 120
    - Point Gap: 20 (17% turnover)
    - Uniform sampling (x, y, θ_c)
  - Atomic actions (steering)
    - 0°, +30°, -30°
    - Movement fixed at constant rate



(Lichodzijewski et al., 2011)

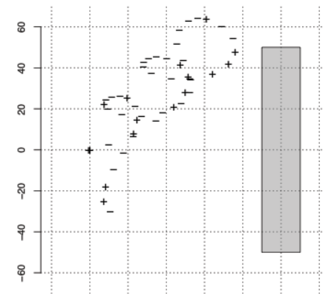


Sample Solution Trajectories

(1 'pin' per 10 moves)

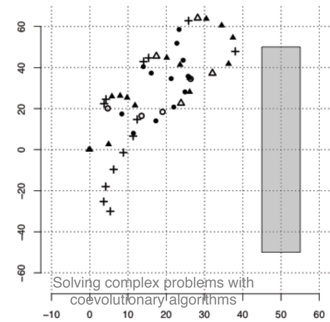
(Lichodzijewski et al., 2011)

Key:
+ denotes 30 degrees
- denotes -30 degrees



Sequencing of \`Atomic' Actions

Key:
Each symbol represents (1 of 5) different layer 0 Hosts

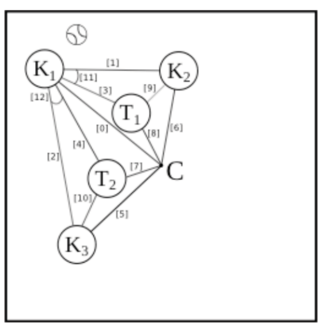


Deployment of Layer 0 hosts

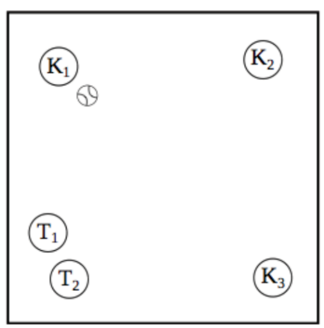
(Lichodzijewski et al., 2011)

Keepaway soccer

Task definition (Stone et al, 2005)



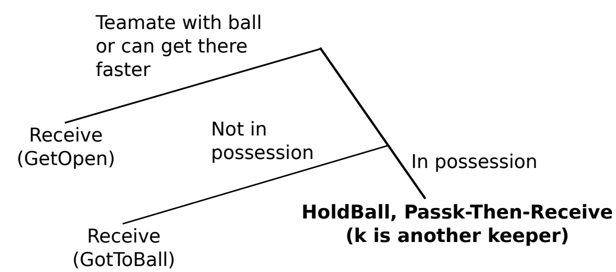
Game initial state
-- Stochastically defined
-- Robocup server



State variables
-- takers to keepers
-- ball assumes similar description

Interface to policy learner

Prior 'keeper' decision tree
Stone et al, (2005)



‘Novelty’ style diversity metric

Kelly & Heywood (2014)

- All start states the ‘same’
- Encourage diversity in failure (novelty)

Reward of individual ‘h_i’ on game ‘e_j’

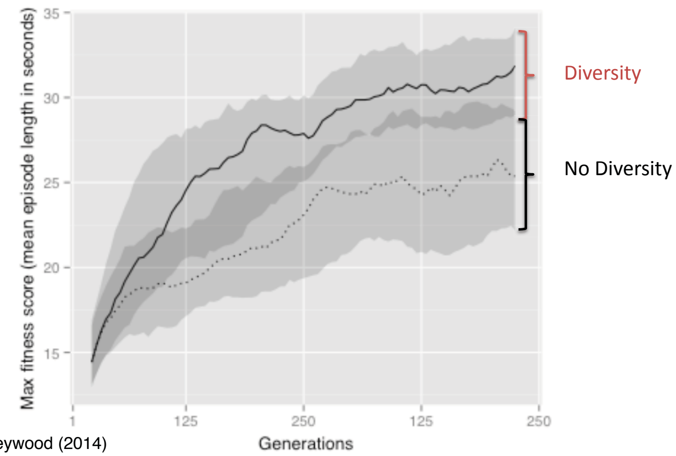
$$s_i = \sum_{j \in h_{hist}} \left(\frac{G(h_i, e_j)}{\sum_{k \neq i}(1 - dist(e_j, e_{hist}))G(h_k, e_{hist})} \right)$$

Distance between current game (e_j) and ‘closest’ historical game (e_{hist}) for alternate solution (h_k)

Reward of alternate individ. (h_k) in historical game (e_{hist})

Keepaway TRAINING performance

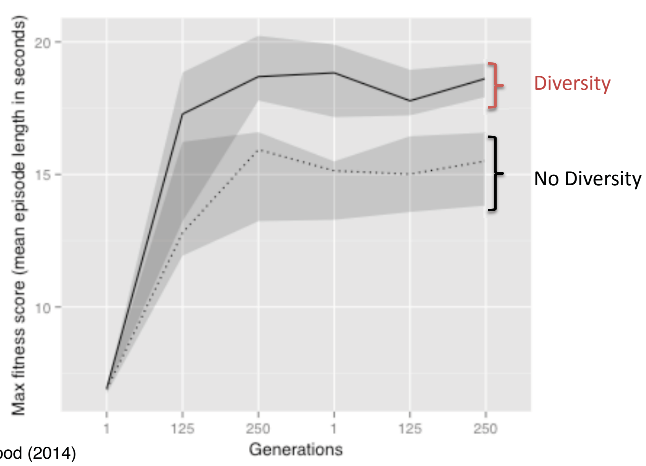
With / Without diversity



Kelly and Heywood (2014)

Keepaway TEST performance

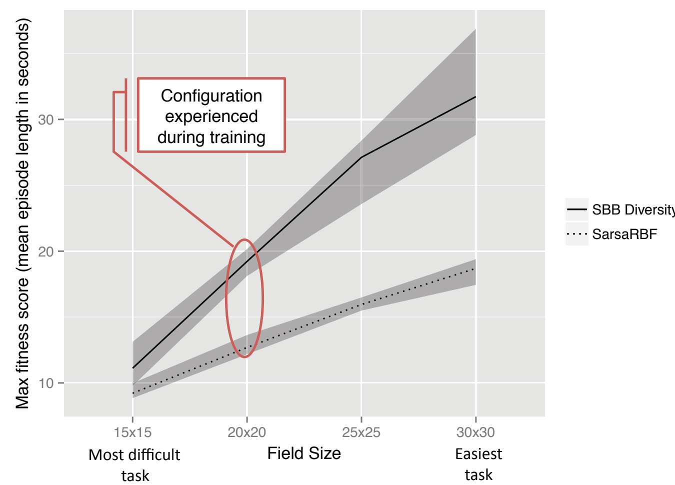
1000 games, Sampled at intervals of 125 generations



Kelly and Heywood (2014)

Keepaway TEST performance

1000 games, Different field sizes



Cooperative Coevolution
Concluding Comments (1 of 2)

- Highlights
  - Separation of context and action
    - Arbitrary team sizes under GP
  - Maintaining Diversity significant
    - Making diversity metrics 'task free'? (see below)
  - Reuse of previous policies leverages diversity for generalization
    - Organization of code hierarchically
  - Solutions generally simpler than monolithic models
  - Easier to react to changing environments

Cooperative Coevolution
Concluding Comments (1 of 2)

- Some open questions (a non exhaustive list!)
  - Credit for collective versus individuals
  - What learning bias are most appropriate for diversity maintenance
    - Task specific metrics
      - E.g., (Nelson et al. 2009)
    - … versus task independent metrics
      - Novelty as an objective (Gomes, Christensen 2013), (Gomes et al., 2016)
      - Compression distance (Gomez, 2009)
      - Connectivity biases (Clune et al., 2013)
      - Intra Team diversity (Kelly, Heywood, 2015), (Gomes et al., 2016)
    - … versus how to 'present' diversity
      - Pareto Multi-objective versus **switching** between multiple diversity metrics (Donieux, Mouret, 2013)
  - Cooperative coevolution and code reuse
    - Supervised learning (Jaskowski et al., 2014)
    - Reinforcement learning (Kelly, Heywood, 2015), (Didi and Nitschke, 2016)
  - Specialization versus generalization
    - Heterogeneous versus Homogeneous deployment of policies within teams (Waibel et al., 2009), (Nitschke et al., 2012)

Cooperative Coevolution
Example Benchmark task domains

- Feature identification to classification
  - K. Krawiec, B. Bhanu (2006, 2007); W. Jaskowski et al., (2014)
    - Constructing hierarchal models for feature extraction and classification
- Double inverted pendulum / cart pole
  - Gomez et al, (2008)
    - Capacity for solving the task
- Truck reversal with obstacle
  - Lichodzijewski et al, (2011)
    - Capacity for solving the task / generalization
- Acrobot
  - Doucette et al, (2012)
    - Capacity for solving the task / generalization
- Predator-prey strategies
  - Nitschke et al., (2012); Yong and Miikkulainen (2009); Rawael et al., (2010); Gomes et al., (2016)
    - Task decomposition and collective problem solving
- Distributed multi-object location
  - Agogino, Tumar (2008); Knudson, Tumar (2010); Colby, Tumar (2012)
    - Task decomposition and (heterogeneous) collective problem solving
- Keepaway or Half field offense (soccer)
  - Kelly, Heywood (2014, 2015), (Didi and Nitschke, 2016)
    - Task decomposition and (homogeneous) collective problem solving
    - Capacity for task / generalization through hierarchical code reuse
- Strategies for solving the Rubik's Cube
  - Smith et al., (2016)
    - Task decomposition and capacity for task / generalization through hierarchical code reuse

IV. Closing remarks

Closing remarks

- Coevolutionary algorithms = conceptually interesting and oftentimes efficient paradigm for solving complex problems
- Addresses key aspects of computational intelligence:
  - What/who to learn from?
  - How to drive the search/optimization?
  - What is solution to my problem?
  - How do I decompose my problem?
  - How do I make some entities cooperate?
- Many interesting results,
  - … even more open questions!

Acknowledgements

- The content of this tutorial has benefited from a host of collaborations over the years including, but not limited to:

  John Doucette, Wojciech Jaśkowski, Stephen Kelly, Peter Lichodzijewski, Paweł Liskowski, Marcin Szubert, Ali Vahdat, Bartosz Wieloch
- MIH would like to acknowledge funding for aspects of research reported on in this tutorial from the NSERC Discovery and CRD programs (Canada).
- KK would like to acknowledge funding for aspects of research reported on in this tutorial from the National Science Centre and National Centre for Research and Development in Poland (Poland).

References

Competitive Coevolution (1 of 3)

- A. Arcuri, X. Yao, Co-evolutionary automatic programming for software development, Information Sciences, 259:412-432, February 2014.
- R. Axelrod (1987) The evolution of strategies in the iterated prisoner's dilemma. In L. Davis, editor, Genetic Algorithms in Simulated Annealing, 32–41. Pitman, London.
- A. Bucci, J.B. Pollack, E. de Jong (2004) Automated extraction of problem structure. In K. Deb et al. (Eds.), Genetic and Evolutionary Computation, GECCO-2004, Part I. Lecture Notes in Computer Science, Vol. 3102, 501–512. Berlin: Springer-Verlag
- S. Y. Chong, P. Tino, and X. Yao (2008) Measuring generalization performance in coevolutionary learning, IEEE Trans. Evol. Comput., vol. 12, no. 4, 479–505
- S.G. Ficici (2004) Solution concepts in coevolutionary algorithms, Ph.D. thesis, Brandeis University, Waltham, MA.
- S.G. Ficici, J.B. Pollack (2001) Pareto optimality in coevolutionary learning. In J. Kelemen and P. Sosık (Eds.), Advances in Artificial Life, 6th European Conference, ECAL'01. Lecture Notes in Computer Science, Vol. 2159, 316–325. Berlin: Springer-Verlag
- D.B. Fogel (2002) Blondie24: Playing at the Edge of AI, Morgan Kaufmann Publishers Inc., San Francisco, CA.
- W. Jaśkowski, K. Krawiec and B. Wieloch (2008) Evolving Strategy for a Probabilistic Game of Imperfect Information using Genetic Programming. Genetic Programming and Evolvable Machines, 9(4):281-294
- W. Jaśkowski, K. Krawiec (2010) Coordinate System Archive for coevolution. In IEEE Congress on Evolutionary Computation.
- W. Jaśkowski, K. Krawiec (2011) How many dimensions in co-optimization. In GECCO (Companion), 829-830.
- W. Jaśkowski, K.Krawiec (2011) Formal Analysis, Hardness, and Algorithms for Extracting Internal Structure of Test-Based Problems. Evolutionary Computation, 19(4):639-671.
- E.D. de Jong (2004) Towards a Bounded Pareto-Coevolution Archive. In Proceedings of the IEEE Congress on Evolutionary Computation, volume 2, 2341– 2348, Portland, Oregon, USA.

References

Competitive Coevolution (2 of 3)

- E.D. de Jong (2004) The Incremental Pareto-Coevolution Archive. In K. Deb et al., editor, Genetic and Evolutionary Computation–GECCO 2004. Proceedings of the Genetic and Evolutionary Computation Conference. Part I, 525–536, Seattle, Washington, USA, Springer-Verlag, Lecture Notes in Computer Science Vol. 3102.
- E.D. de Jong, J.B. Pollack (2004) Ideal evaluation from coevolution. Evolutionary Computation, 12(2):159–192.
- E. D. de Jong (2004) The MaxSolve algorithm for coevolution, in GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation, 2005, 483–489. dissertation, Waltham, MA, USA.
- E.D. de Jong, A. Bucci (2006) DECA: Dimension extracting coevolutionary algorithm. In Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO 2006, 313–320
- K. Krawiec, (2001) Pairwise Comparison of Hypotheses in Evolutionary Learning. In Machine Learning. Proceedings of the Eighteenth International Conference, ICML 2001. Morgan Kaufmann Publishers, 266-273.
- K. Krawiec, P. Liskowski (2015) Automatic Derivation of Search Objectives for Test-Based Genetic Programming, in P. Machado, M. Heywood, J. McDermott (eds.), 18th European Conference on Genetic Programming, Springer
- K. Krawiec and M. Szubert (2011) Learning N-tuple networks for Othello by coevolutionary gradient search, in Proc. Genetic Evol. Comput. Conf., ACM 355–362.
- P. Liskowski, K. Krawiec, Online Discovery of Search Objectives for Test-based Problems, Evolutionary Computation Journal, MIT Press, 2016 (accepted).
- T. Miconi (2009) Why coevolution doesn't work: Superiority and progress in coevolution, In: L. Vanneschi, et al. (eds.), EuroGP 2009, Springer-Verlag, Berlin Heidelberg New York, 49–60.
- G.A. Monroy, K.O. Stanley, and R. Miikkulainen (2006) Coevolution of neural networks using a layered Pareto archive. In M. Keijzer et al., editors, GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation, volume 1, 329–336, Seattle, Washington, USA, 8-12 July 2006. ACM Press.
- P. Moscato (1989) On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms, Caltech Concurrent Computation Program C3P Rep., vol. 826.
- J. Noble, R.A. Watson (2001) Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for Pareto selection. In L. Spector et al. (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001, 493–500.
- J.B. Pollack, A.D. Blair (1998) Co-evolution in the successful learning of backgammon strategy. Mach. Learn. 32(3), 225–240.

References
Competitive Coevolution (3 of 3)

- E. Popovici, A. Bucci, R.P. Wiegand, and E.D. de Jong (2012) Coevolutionary Principles. In Rozenberg, G., Baeck, T., and Kok, J. N., editors, Handbook of Natural Computing, 987–1033. Springer.
- E. Popovici, E. Winston (2015) A framework for co-optimization algorithm performance and its application to worst-case optimization, Theoretical Computer Science, Volume 567, Pages 46-73
- C.D. Rosin and R. K. Belew (1997) New methods for competitive coevolution, Evolutionary Computation, vol. 5, no. 1, 1–29.
- A.L. Samuel (1959) Some studies in machine learning using the game of checkers. IBM Journal of Research and Development, 3(3):211–229.
- O.G. Selfridge, R.S. Sutton, A.G. Barto (1985) Training and Tracking in Robotics. In Joshi, A. K., editor, Proceedings of the 9th International Joint Conference on Artificial Intelligence, IJCAI, 670–672, Los Angeles, CA. Morgan Kaufmann.
- T.C. Service, D.R. Tauritz (2008) A no-free-lunch framework for coevolution, in: Proceedings of the Genetic and Evolutionary Computation Conference, ACM, 371–378.
- M. Szubert, Coevolutionary (2014) Shaping for Reinforcement Learning, Phd Thesis, Institute of Computing Science, Poznan University of Technology.
- M. Szubert, W. Jaśkowski, K. Krawiec (2009) Coevolutionary Temporal Difference Learning for Othello. In IEEE Symposium on Computational Intelligence and Games. 104-111.
- M. Szubert, W. Jaśkowski, P. Liskowski, K. Krawiec (2013) Shaping Fitness Function for Evolutionary Learning of Game Strategies. In Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference, GECCO '13, 1149–1156, New York, NY, USA. ACM.
- M. Szubert, W. Jaśkowski, K. Krawiec (2013) On Scalability, Generalization, and Hybridization of Coevolutionary Learning: A Case Study for Othello. Computational Intelligence and AI in Games, IEEE Transactions on, 5(3):214-226.
- B. F. Skinner (1938) The behavior of organisms: An experimental analysis. Appleton-Century.
- D. Wolpert, W. Macready (2005) Coevolutionary free lunches, IEEE Trans. Evol. Comput. 9: 721–735.

References
Cooperative Coevolution (1 of 3)

- A. K. Agogino, K. Tumar (2008) Efficient evaluation functions for evolving coordination. Evolutionary Computation 16(2): 257–288
- J. Clune, J.-B. Mouret, H. Lipson (2013) The evolutionary origins of modularity. Proceedings of the Royal Society – B 280 20122863
- S. Didi, G. Nitschke (2016) Multi-agent behavior based policy transfer. EvoApplications. LNCS 9598: 181—197
- M. Colby, K. Tumer (2012) Shaping fitness functions for coevolving cooperative multiagent systems. ACM AAMAS 425–432
- S. Doncieux, J.-B. Mouret (2013) Behavioral diversity with multiple behavioral distances. IEEE CEC 1–8
- J. Gomes, P. Mariano, A. L. Christensen (2016) Novelty-driven cooperative coevolution. Evolutionary Computation. To appear. (2016)
- J. Gomes, A. L. Christensen (2014) Generic behaviour similarity measures for evolutionary swarm robotics. ACM GECCO 199—206
- F. Gomez, J. Schmidhuber, R. Miikkulainen (2008) Accelerated neural evolution through cooperatively coevolved synapses. Journal of Machine Learning Research 9:937–965
- F. Gomez (2009) Sustaining diversity using behavioural information distance. ACM GECCO 113–120
- W. Jaskowski, K. Krawiec, B. Wieloch (2014) Cross-task code reuse in genetic programming applied to visual learning. Applied Mathematics and Computer Science 24(1): 183—197
- M. Knudson, K. Tumar (2010) Coevolution of heterogeneous multi-robot teams. ACM GECCO 127–132
- K. Krawiec, B. Bhanu (2007) Visual learning by evolutionary and coevolutionary feature synthesis. IEEE Transactions on Evolutionary Computation 11(5): 635–650
- K. Krawiec, B. Bhanu (2006) Visual learning by coevolutionary feature synthesis. IEEE Transactions on Systems, Man and Cybernetics. Prt B. 35: 409–425
- J. Maynard Smith (1991) A Darwinian view of symbiosis. Chapter 3 in Symbiosis as a source of evolutionary innovation. (eds) L. Margulis and R. Fester (MIT Press)
- D. E. Moriarty, R. Miikkulainen (1998) Forming neural networks through efficient and adaptive coevolution. Evolutionary Computation 5(4):373–399

References
Cooperative Coevolution (2 of 3)

- A. L. Nelson, G. J. Barlow, L. Doitsidis (2009) Fitness functions in evolutionary robotics: A survey and analysis. Robotics and Autonomous Systems 57: 345–370
- G. S. Nitschke, A. E. Eiben, M. C. Schut (2012) Evolving team behaviors with specialization. Genetic Programming and Evolvable Machines 13(4): 493—536
- L. Panait, S. Luke, R. P. Wiegand (2006) Biasing coevolutionary search for optimal multiagent behaviors. IEEE Transactions on Evolutionary Computation 10(6): 629–645
- L. Panait, K. Tuyls, S. Luke (2008) Theoretical advantages of lenient learners: An evolutionary game theoretic perspective. Journal of Machine Learning Research 9: 423–457
- M. A. Potter, K. A. De Jong (2000) Cooperative coevolution: An architecture for coevolving coadapted subcomponents. Evolutionary Computation 8(1): 1–29
- A. Rawal, P. Rajagoplan, R. Miikkulainen (2010) Constructing competitive and cooperative agent behavior using coevolution. IEEE CIG 107—114
- J. Rubini, R. B. Heckendorn, T. Soule (2009) Evolution of team composition in multi-agent systems. ACM GECCO 1067–1072
- P. Stone, R. Sutton, G. Kuhlmann (2005) Reinforcement learning for RoboCup soccer Keepaway. Adaptive Behavior 13: 165–188
- R. Thomason, T. Soule (2007) Novel ways of improving cooperation and performance in ensemble classifiers. ACM GECCO 1708–1716
- C. H. Yong and R. Miikkulainen (2009) Coevolution of role-based cooperation in multi-agent systems. IEEE Transactions on Autonomous Mental Development 1(3): 170—186
- M. Waibel, L. Keller, D. Floreano (2009) Genetic team composition and level of selection in the evolution of cooperation. IEEE Transactions on Evolutionary Computation. 13(3):648—660
- S. Wu, W. Banzhaf (2011) Rethinking multilevel selection in genetic programming. ACM GECCO. 1403 – 1410

References
Cooperative Coevolution (3 of 3)

- J. A. Doucette, P. Lichodzijewski, M. I. Heywood (2012) Hierarchical task decomposition through symbiosis in reinforcement learning. ACM GECCO 97–104
  - Finding optimal solutions to the Acrobot 'handstand' task
  - Complexity through hierarchical code reuse
- S. Kelly, M.I. Heywood (2014) On diversity, teaming, and hierarchical policies: Observations from the Keepaway soccer task. EuroGP LNCS 8599:75–86
  - Diversity maintenance, modularity and generalization under keepaway
  - Complexity through hierarchical code reuse
  - https://web.cs.dal.ca/~skelly/keepaway-gecco-2015/
- S. Kelly, M.I. Heywood (2015) Knowledge transfer from keepaway soccer to half- field offense through program symbiosis: Building simple programs for a complex task. ACM GECCO.
  - Task free diversity metrics, scaling to more difficult problems with task transfer
- P. Lichodzijewski, M. I. Heywood (2008) Managing team-based problem solving with symbiotic bid-based genetic programming. ACM GECCO 363–370
  - Basic architecture, no hierarchy, supervised learning; benchmark with multi-class classification and LCS
- P. Lichodzijewski, M. I. Heywood (2010) Symbiosis, Complexification and Simplicity under GP. ACM GECCO 853–860
  - Simplified basic architecture, no hierarchy, supervised learning; benchmark against monolithic GP solutions
- P. Lichodzijewski, J.A. Doucette, M. I. Heywood (2011) A symbiotic framework for hierarchical policy search. FCS, Dalhousie University. Tech. Report CS-2011-06.
  - Truck reversal domain tutorial
  - http://www.cs.dal.ca/research/techreports/cs-2011-06
- R. J. Smith, S. Kelly, M. I. Heywood (2016) Discovering Rubik's Cube Subgroups using Coevolutionary GP -- A Five Twist Experiment. ACM GECCO.
  - Scaling to more difficult tasks with diversity maintenance and task transfer
  - Complexity through hierarchical code reuse
- A. Vahdat, J. Miller, A. McIntyre, M. I. Heywood, N. Zincir-Heywood (2015) Evolving GP classifiers for streaming data tasks with concept change and label budgets. Handbook of GP Applications. (Springer)
  - Significance of coevolving task decomposition under non-stationary streaming data