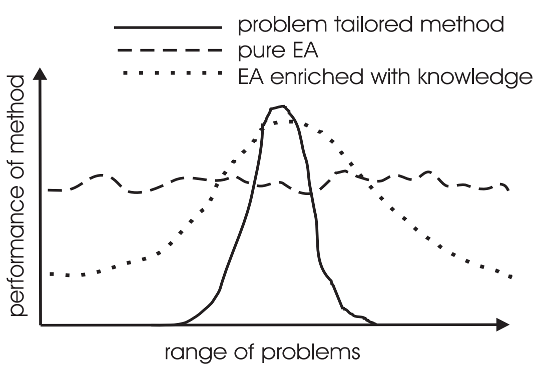
Nature-Inspired Computing

Hybridization

Dr. Şima Uyar
September 2006

Why Hybridize

- NIH may be a part of a larger system
- to improve on existing techniques
- to improve NIH search
- ...



How to Hybridize?

- creation of initial solutions
- local improvement of candidate solutions
- intelligent decoders
- intelligent / heuristic variation operators

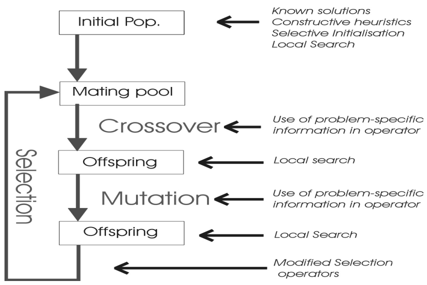
How to Hybridize?

- parallel populations
  - same / different heuristic
  - same / different metaheuristic
  - different parameter settings
  - different fitness functions
- approximate models
  - costly fitness evaluation
  - use approximate model for some evaluations
  - use different levels of approximation for sub-populations
  - hierarchical model

How to Hybridize?

- modify problem instance
  - e.g. decrease search space size
- partition into sub-problems
- interactive iterations
  - for local tuning
  - for constraints

Where to Hybridise: Example EA



Initialization

- initialise population with
  - previously known solutions
  - solutions found by other technique
- "inject" population with
  - solutions from previous runs
  - solutions found by other algorithms

Heuristics for Initializing Population

- *n*-way tournament among randomly created solutions
- multi-start local search: pick *N* points randomly to climb from
- constructive heuristics often exist

Heuristics for Initialising Population

- diversity is important
- advantage: good solutions found quickly
- disadvantage: possible to get stuck at local optima (strong bias)

Intelligent Operators

- incorporating problem or instance specific knowledge within operators
- usually with problem specific representations
- usually fast

Local Search Acting on Offspring

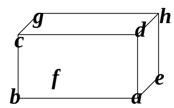
- to speed-up the NIH
- NIH for exploration, LS for eploitation
- makes search around good solutions more systematic
- fast local optimizer needed
- smoothes fitness landscape
- introduces redundancy and plateaus
- very successful in practise

Example: Memetic Algorithms

- combination of EAs with local search in EA loop: Memetic Algorithms
- also EAs using instance specific info in operators
- shown to be faster and more accurate than EAs on some problems
- are the "state of the art" on many problems: e.g. scheduling and timetabling problems

Local Search

- *neighbourhood* concept
- *N(x):* set of points that can be reached from *x* with one application of a move operator
  - e.g. bit flipping search on binary problems



⇐ ***N(d) = {a,c,h}***

Local Search

- *degree* of graph: max. no of edges coming into/out of a single point
  - size of biggest neighbourhood
- local search look at points in neighbourhood of a solution
  - complexity related to degree of graph
    - bit-wise mutation on binary

Local Search

- is neighbourhood searched randomly, systematically or exhaustively ?
- does search stop as soon as a fitter neighbour is found (*Greedy Ascent*)
- or is whole set of neighbours examined and the best chosen (*Steepest Ascent*)
- ....

Local Search

- local search in representation or solution space ?
- how many iterations of local search ?
- local search applied to whole population?
  - or just the best ?
  - or just the worst ?

Two Models of Lifetime Adaptation

- Lamarkian
  - traits acquired by individual during lifetime transmitted to offspring
  - e.g. replace individual with fitter neighbour
- Darwinian
  - traits acquired by individual during lifetime not transmitted to offspring
  - e.g. individual receives fitness (but not genotype) of fitter neighbour

Two Models of Lifetime Adaptation

- Baldwinian effect: individual learning improves evolutionary learning by changing fitness landscape (both models)
- Lamarckian good in stationary environments
- Darwinian good in dynamic environments

Intelligent Decoders

- indirect representation
- use a decoding function
  - decoding function uses problem specific info
- representations
  - permutations
  - random keys
  - weight codings
- good with handling constraints
- time consuming
- locality problem

Hybrid Algorithms Summary

- hybridize especially for real world problems
- hybridization may involve
  - use of operators from other algorithms
  - incorporation of domain-specific knowledge
- hyrid algorithms
  - shown to be much faster and more accurate on some problems
  - the “state of the art” on many problems
- more problem specific
- requires more parameter settings
- possible loss of creativity