Metaheuristic Design Pattern: Surrogate Fitness Functions

Alexander E.I. Brownlee\textsuperscript{1}
John R. Woodward\textsuperscript{1}
Jerry Swan\textsuperscript{2}

\textsuperscript{1} University of Stirling, UK
\textsuperscript{2} University of York, UK
Outline

- Problem statement
- Solution
- Consequences
- Implementation
- Examples
Problem statement

• Metaheuristics need some notion of “fitness”

We’re done – how did we do? (objectives)

• Two purposes:
  1. Measure quality (wrt the objectives+constraints)
  2. Guide the search

• (1) is not necessarily suitable for (2)...
Problem statement

The *true* fitness function might:

1. Be costly
2. Be noisy
3. Not have a useful search gradient
Solution

• *Surrogate* fitness function in place of “true” FF
  – Still need to refer to “true” fitness occasionally
• a.k.a. meta-model, proxy, fitness model or approximation
  – typically one of the above for costly problems, but less so for noisy problems or reshaping landscape

• Two types:
  – Static
  – Dynamic
Solution

- Static surrogates
  - part of problem definition
  - can include domain knowledge
  - typically guides search towards partial solutions

- Constraint relaxation, multi-objective weights
  - *might* be classed as surrogates

- Often used already!
  - We don’t usually directly search the real world
Solution

• Dynamic surrogates
  – Regression or machine learning: polynomials, Kriging, artificial neural networks, interpolations
  – Fitness inheritance

• Trained using samples of “true” fitness

• Updated or replaced over time
  – bridge / handle / body pattern

• Ensembles combine strengths of many
  – “composite” pattern
Consequences

• Search landscape altered
• Approximation errors
  – Must make reference to objective function
  – e.g. Surrogate filters new solutions before full evaluation, or switch between surrogate & true
• Can offer speed up – but balance with overhead
• Surrogate explicitly models fitness: mine it to support decision making
Example 1

- Long-running simulations of building energy performance (mins to hrs)
- RBFN surrogate uses population as training data
- Filters offspring before evaluation with full simulation
- Many similar examples

Example 2

• Eternity II puzzle
• Objective: maximise matched adjacent edges
• Surrogate objectives:
  – Completed 2x2 squares
  – Completed 3x3 squares
  – Completed 4x4 squares
  – Tiles with all 4 edges matched
• Search iterates over two stages: surrogate, then objective

Example 3

- MFM-GA uses undirected PGM (Markov network) to approximate fitness
- PGM initialised with dependencies between 5-bit blocks in problem, coefficients estimated using randomly generated population
- Fewer evals wrt GA, but more overhead

Mining a surrogate model

• Examine the surrogate model to gain insight into the problem

• Model here shows where glass is preferred (blue) on the façade
Summary

• “true” fitness not always suited to guiding search
• Use surrogates to improve search efficiency
• Static surrogates often used already!
• More reading...

www.cs.stir.ac.uk/~sbr  sbr@cs.stir.ac.uk
Class diagram

- `ObjectiveFunction`
- `Problem`
- `Solver`
- `EvaluationFunction`
- `SurrogateFunction`
- `EnsembleSurrogate`
- `DynamicSurrogate`

`Problem` is associated with `ObjectiveFunction` with a cardinality of 1:1. Both `Problem` and `SurrogateFunction` are associated with `ObjectiveFunction` with a cardinality of 1:1. `SurrogateFunction` is associated with `EnsembleSurrogate` and `DynamicSurrogate` with cardinalities of 1 and *, respectively. `EvaluatedFunction` has a method `evaluate()`.