Application of evolutionary algorithms to solve complex problems in quantitative genetics and bioinformatics

7. Managing constraints

Those that trespass will be penalized. Or otherwise fixed up.

Brian Kinghorn

University of New England
Armidale, Australia

Simple logic constraints: examples

- The number of mates to allocate to a bull should not be negative.
- The amount of feed to offer an animal should not be negative.
- The date to wean should not be before birth date.
- A natural mating bull can be used on a maximum of one farm at a time.

Stakeholder applied constraints: examples

- Maximum breeding herd size should be 200 females mated.
- The maximum acceptable coancestry among parents in this mating round is 0.1.
- The maximum number of individuals in a group for pooled tissue quantitative genotyping is ten.
- The maximum permissible number of genetic marker mismatches per animal (various applications …)

Ask your self: Do I need this constraint?

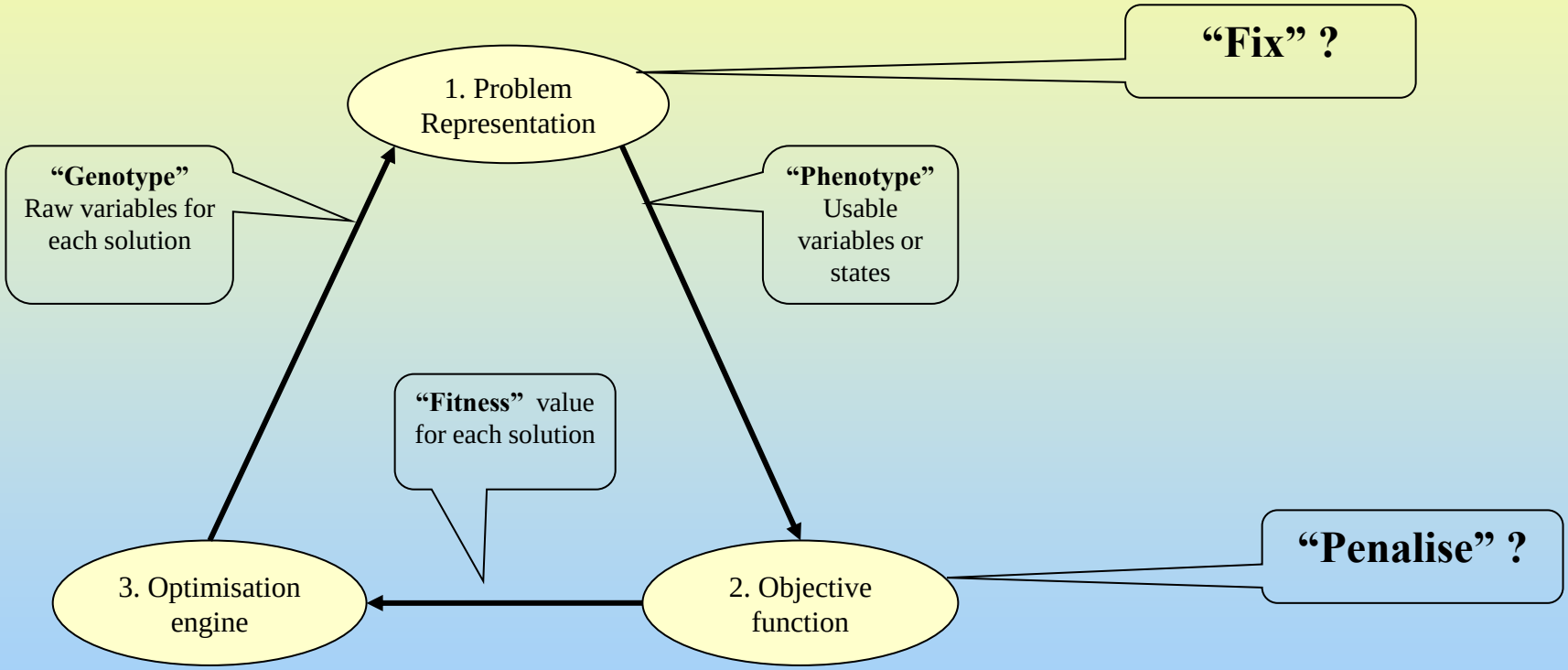
- Constraints often requested because of preconceived ideas about the optimal solution.
- This prevents the computer from “thinking outside the box”!
- You might get a surprise and discover an unusual solution ... (see “Let your computer make you famous.” in Chapter 1).
- Investigate the impact of removing constraints that are not needed to give a feasible solution.
- If these constraints were well-founded, then the optimal solution should not break them.

How to apply a constraint

Penalise: Apply penalty in the objective function to solutions that break constraints. The penalty should be sufficiently large to prevent the optimal solution being one that breaks the constraint.

Fix: If the constraint is on the variables passed to the objective function, you can choose to change these appropriately before passing them to the objective function.

Application of constraints



Simple Fix – make boundaries

- Simple fix is to set bounds for variables, as in DE_Demo

```
For j = 1 To loci
    If allele(j) < MinVal(j) Then allele(j) = MinVal(j)
    If allele(j) > MaxVal(j) Then allele(j) = MaxVal(j)
Next
```

- … before allele() gets evaluated in the Criterion subroutine. Setting values exactly to the boundaries is simple but not the best strategy.
- See “Assigning animals into groups” in Chapter 6.

Fix: Dealing cards

			Female →	1	2	3	4 …
Male↓	**No of uses**	Ranking criterion	Rank	1	0	1	1
1	2	**5.32** **2.16**	**2** **3**			✓	✓
2	0	-	-				
3…	1	**7.64**	**1**	✓			

Fixing up: comments

- "Fixing up" is the better strategy for most "logical" constraints. Fix them up before going to the objective function.
- Store the solution in its fixed state, especially if the fixing process is stochastic.
  - Misleading in notes: "However, you must do this in a repeatable manner. If two solutions have exactly the same "unfixed" set of parameters, they should have the same "fixed" set of parameters" … Not needed if store population member in the fixed state.
- Some danger that the strategy you use prevents good exploration of the solution space.
  - Preferably do not "Fix" parameters to the same sort of region, for example at the boundaries of constraints.

Penalising fitness

- "The big advantage of the "Penalise" strategy is that it so easy to apply…

```
If (ConstraintBroken) fitness = fitness – 999999
```

… that should cure the problem!

- Penalisation is easy to implement, but it makes for much slower convergence if most solutions are illegal.

Penalising: Hard constraints and Soft constraints

- In some such cases all solutions break one or more constraints in the first generation(s) of optimization.
- So, apply “softer” penalties – so that the optimization engine can give reward to solutions that break fewer constraints.
- Eventually it finds solutions that break no constraints.
- You might have to play with constraint penalties to help a complex optimization to “get off the ground”.
- A ‘hard constraint’ would allocate the smallest possible fitness. That is rarely needed ...

Penalising: Hard constraints and Soft constraints

- A hard constraint:

```
If (Para > ParaConstraint) fitness = minus infinity
```

- A softer constraint:

```
If (Para > ParaConstraint) fitness = -999
```

- A soft constraint with reward:

```
If (Para > ParaConstraint) fitness = fitness – 99*(Para – ParaConstraint)
```


The End