Genetic Algorithms and Genetic Programming
Lecture 10

Gillian Hayes

27th October 2006

Genetic Planning

- What is planning?
- What is genetic planning?
- The basic algorithm
- Some results
- Better seeding using heuristics
- More results
- Concluding remarks

What is Planning?

- An initial state, e.g. `on(A,B), on(B,table)`

- A goal state, e.g. `on(B,A), on(A,table)`

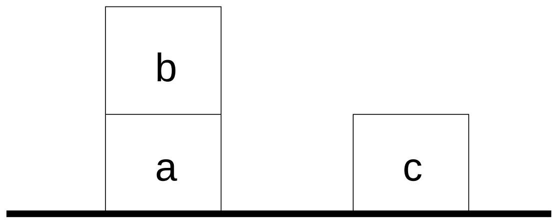
- A library of *planning operators*

- Problem: find a sequence of actions (operators) to transform the initial state into the goal state

- Example problem: Blocks World

The Blocks World

- Representation
  `table` represents the table
  `a ...z` represents the blocks
  `on(a,b)` represents that a is on b
  `clear(a)` represents that there is space for a block on a; there is unlimited space on the table
- So the situation



could be represented by

`on(a,table)`	`on(b,a)`	`clear(b)`
`on(c,table)`	`clear(c)`	`clear(table)`

Planning Operators

- To manipulate the blocks world, we need just one operator:

Name:	`move(A,B)`

Preconds:	`clear(A)`
	`clear(B)`
	`on(A,C)`

Add List:	`on(A,B)` `clear(C)`
Delete List:	`clear(B)` `on(A,C)`

- Special case: the table is *always* clear, and cannot be the first argument of `move(A,B)`

What is Genetic Planning?

- Creation of plans by genetic means
- An alternative to searching and backtracking
- Runs in constant memory – unlike A^*, etc.
- Easy to run on parallel machines
- More time $\rightarrow$ better solutions

How does Genetic Planning Work?

- Make a number of random guesses at the plan
- Evaluate, select, breed new plans, continue
- When happy with the best plan, stop

The Basic Algorithm

1. Initialise the populations with guesses

2. If best plan is acceptable, return it

3. Simulate each plan on the initial state

4. Evaluate each plan using the results of 3.

5. Select individuals for breeding

6. Create new population, goto 2.

Initialising the Populations

- For the given problem, make a list of all the possible actions

  e.g. for n-block problems using a single `move(X,Y)` operator there are n^2 possible actions

- To create a candidate plan of length l, make a list of l random choices from this list:

  e.g.
  `[move(9,6), move(1,table), move(2,10)]`

- Create p candidates as the initial population

Simulation

- Run plan forwards from initial state
- Ignore actions which cannot execute
- Continue until end of plan
- Record the final state achieved
- Do this for the whole population

Evaluation

- Evaluate each plan using a *fitness function*

- Simple fitness function: number of goals achieved

- Can reward other aspects of the plan, such as plans with a greater number of actions which execute, etc.

- Can also use problem-specific fitness functions

Selection and Breeding

- Selection is done using tournaments of size t
- Use 80% crossover, 20% reproduction
- Reproduction = copy individual into next generation
- Crossover = 1-point crossover, two children:

```
[a1, a2, a3] [b1, b2, b3, b4]
```

Mutation

- Create new population, then apply mutation to it
- Addition mutation: add a random action to 10% of the population

  e.g.
  `[move(9,6), move(1,table), move(2,10)]`

- Shrink mutation: delete a random action from 10% of the population

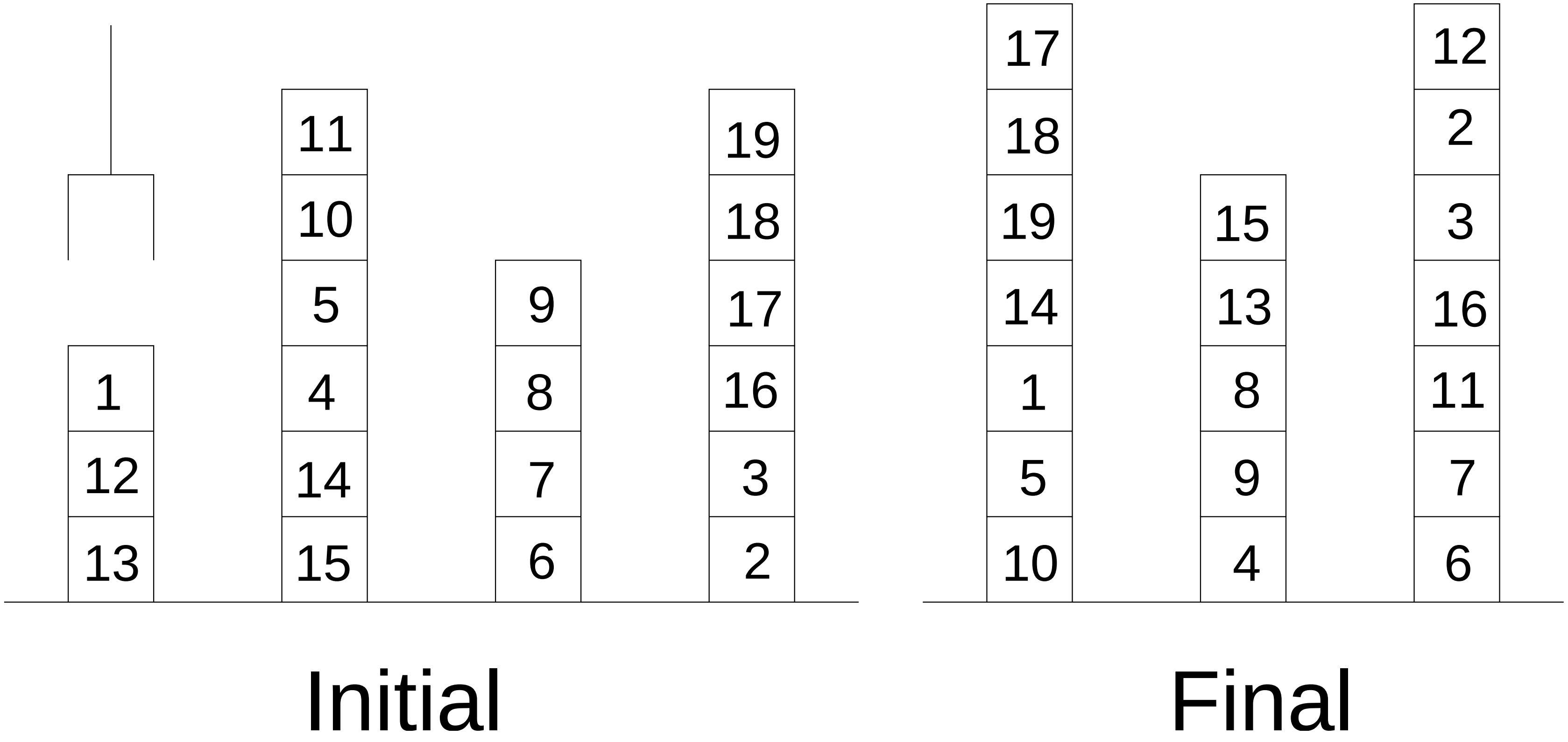
  e.g.
  `[move(9,6), move(1,table), move(2,10)]`

Some Results

- Parameters:
  - Population size = 1000
  - Tournament size = 2

  9 blocks, 6 actions: 15.6 generations
  11 blocks, 9 actions: 40.2 generations
  15 blocks, 14 actions: 210.5 generations
  19 blocks, 18 actions: 590.0 generations
- At the time (2001), these last two were unsolvable by one of the world's best planners (Blackbox)
- Newer planners do better than GP
- Current work to improve GP competitiveness

Blocks World – LargeD



Getting Optimal Plans

- Run in two phases:

  Phase 1: find *any* plan which works

  Phase 2: if plans are equally fit, select shorter plans

- So change the fitness function in phase 2:

- Combination of crossover and shrink mutation do the work (i.e. turn off/down addition mutation)

- Generates an optimal or near-optimal plan

Better Seeding Using Hill-Climbing

- Simple hill-climbing with a heuristic generates plans quickly . . .
- . . . but the plans are almost always sub-optimal
- Example heuristic: well-placed blocks – for LargeD (18 actions), this generates working but sub-optimal plans (25+ actions)
- Idea: use the non-optimal plans found by hill-climbing as the initial population for Phase 2 of genetic planning to find the optimal plan

More Results

- LargeD: optimal plan is 18 actions
- Population size = 10, 100 generations

  5 trials, 4 optimal plans, 1 19-action plan
- Population size = 20, 500 generations

  5 trials, 5 optimal plans
- A^* explores 1003 states using the same heuristic

Concluding Remarks

- Seems to be some promise in this method!
- Can generate plans with or without using a heuristic
- Can generate optimal plans
- Algorithm has some nice properties
- Future work:
  - More on *why* it works
  - More informed crossover, mutation
  - Learning of heuristics and short-cuts
  - Learning of domain-specific planners