Scheduling Planned Maintenance of the National Grid

W. B. Langdon

email W.Langdon@cs.ucl.ac.uk

Dept of Computer Science

University College London.

Code available via anonymous ftp site cs.ucl.ac.uk directory genetic/four_node

- PhD student, Computer Science, UCL
- CASE Award with
- The National Grid Company Plc.'s Technology and Science Laboratories

National Grid

- Bulk Electricity transfer
- ≈ 300 nodes ≈ 400 lines
- Planned maintenance outages
- Year plan
- 52 weeks
- Plan for least cost

Maintenance Costs

- Major cost is cost of running generators
- Try to run cheapest generators
- Constraints prevent this
  - Generator characteristics
  - Network stability
  - Line loading
- Replacement generation, Major cost
- Other costs, maybe considered later

Fitness of Trial Schedules

- Fitness models cost of replacement generation
- assume cheapest generation, calculate power flows
- excessive flows can't happen

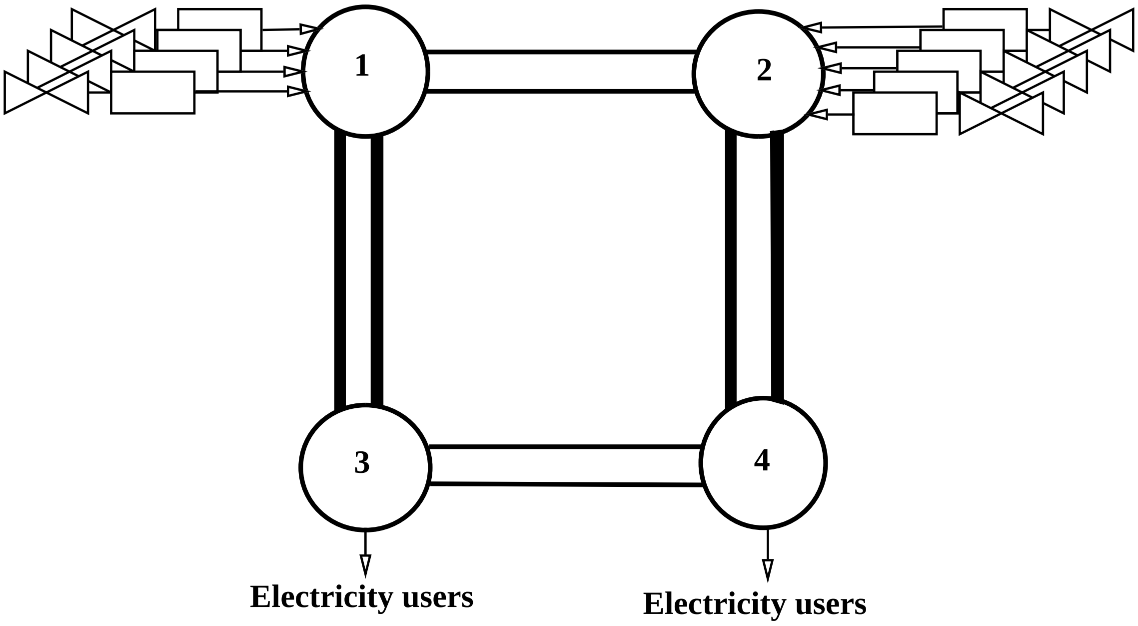
$\leadsto$ reduce flows by generating elsewhere, e.g. closer to demand

$\leadsto$ increase in generation costs

- model by sum excess flows

Fitness Function 2

1. Benefit for doing Maintenance
2. Penalty
    - for exceeding conductor limits
    - for isolating nodes
    - for splitting network



- Maintain all eight lines
- each takes a week
- Nine week plan
- 9 generators different prices
- 8 generators are also maintained

Week	Predicted Demand		
	Node 3	Node 4	Total
1	1200	1300	2500
2	1200	1600	2800
3	1000	2000	3000
4	1100	2300	3400
5	1100	2400	3500
6	1000	2500	3500
7	900	2600	3500
8	900	2700	3600
9	800	3000	3800

Generator Price and Capacity

	Node 1 Generators			
	G1.1	G1.2	G1.3	G1.4
Price	1.0	2.0	3.0	4.0
Cap	500	500	500	500

	Node 2 Generators				
	G2.1	G2.2	G2.3	G2.4	G2.5
Price	3.5	4.5	5.5	6.5	7.5
Cap	400	400	400	400	400

Week		Node 1			Node 2				
1	x	F	F	F	F	F	200	0	0
2	F	x	F	F	F	F	F	100	0
3	F	F	x	F	F	F	F	300	0
4	F	F	F	x	F	F	F	F	300
5	F	F	F	F	x	F	F	F	300
6	F	F	F	F	F	x	F	F	300
7	F	F	F	F	F	F	x	F	300
8	F	F	F	F	F	F	F	x	F
9	F	F	F	F	F	F	F	F	200

x generator not available

F generator fully loaded

0 generator used

generator partially loaded

Power input/output calculated from:

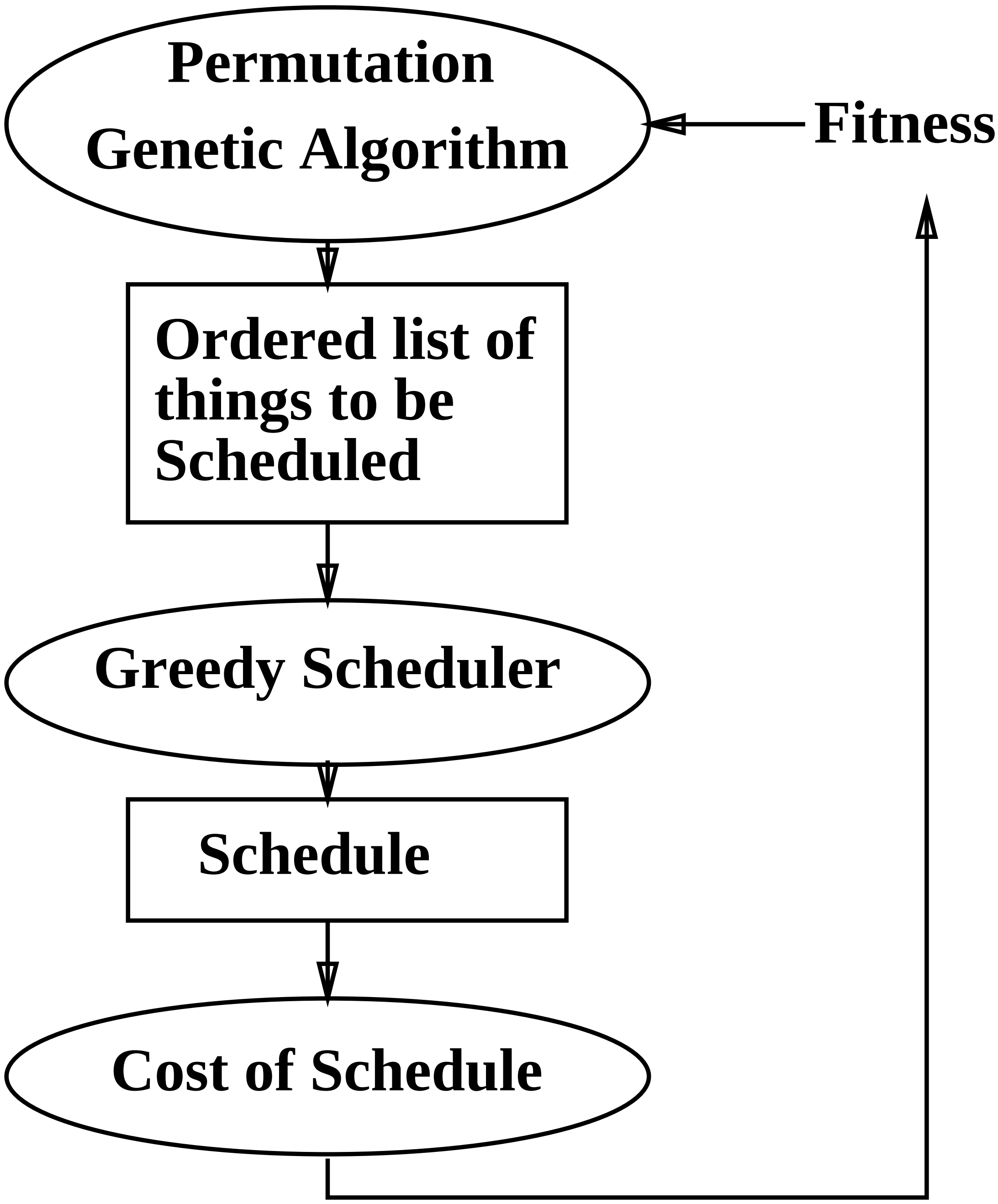
- predicted demand
- generator prices
- generator capacities
- generator availabilities

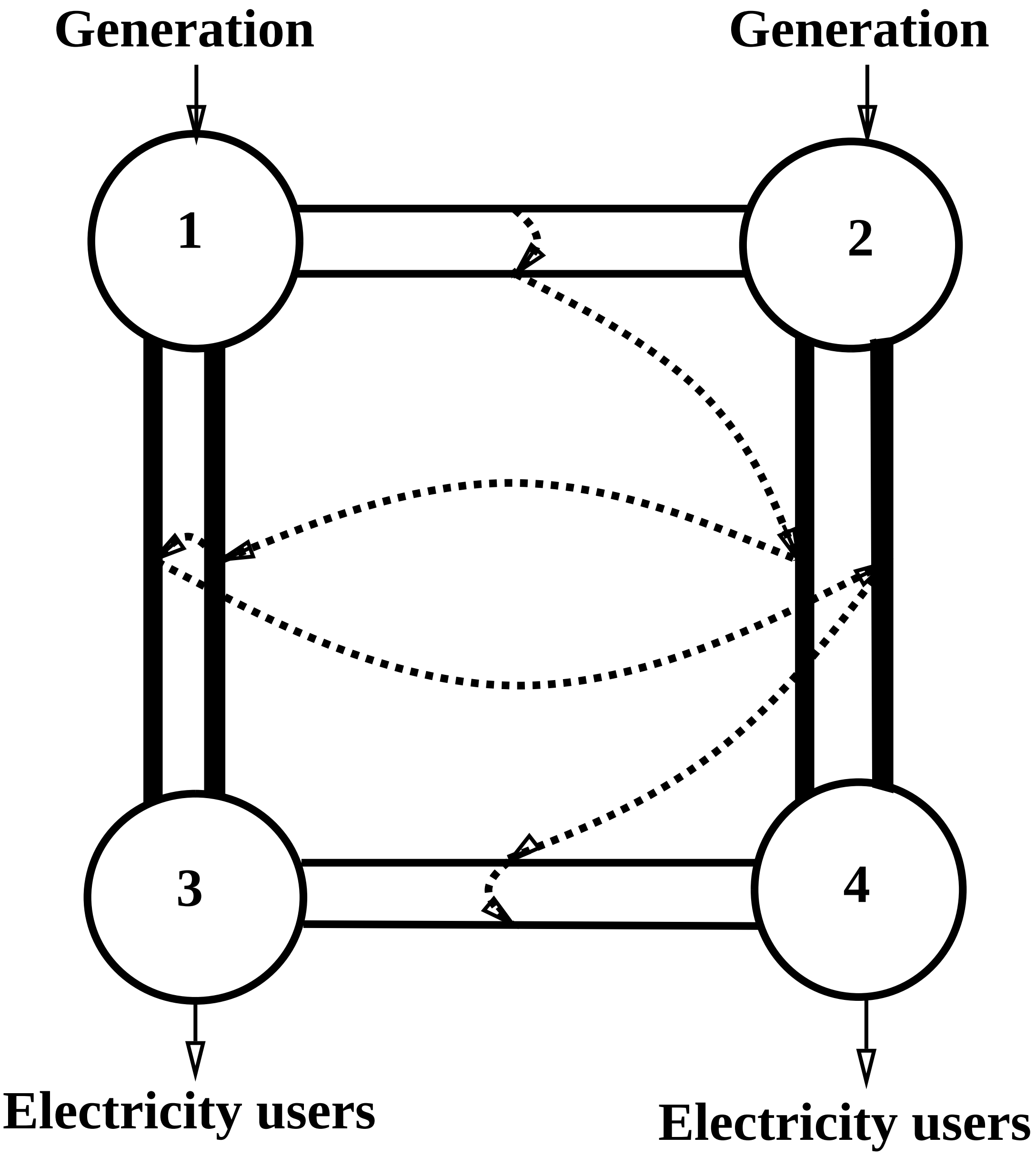
$\rightsquigarrow$ power flows for maintenance schedule

$\rightsquigarrow$ fitness of schedule

	1		0					1		1		
	0		1					0		0		

- Two good schema
- Crossover between them
- Both offspring likely to be poor:
  - Maintain 2 lines at same time
  - Fail to maintain either line


Permutation
Genetic Algorithm
Fitness
Ordered list of
things to be
Scheduled
Greedy Scheduler
Schedule
Cost of Schedule


Generation
Generation
1
2
3
4
Electricity users
Electricity users

In favour of Greedy Schedulers

- Schema prioritise work
- order to think about work **not** to do it
- Predictable?
  - Seed initial population with “good”
  - Specialised genetic operators
- Same next year? Reuse population next year or on reschedule?

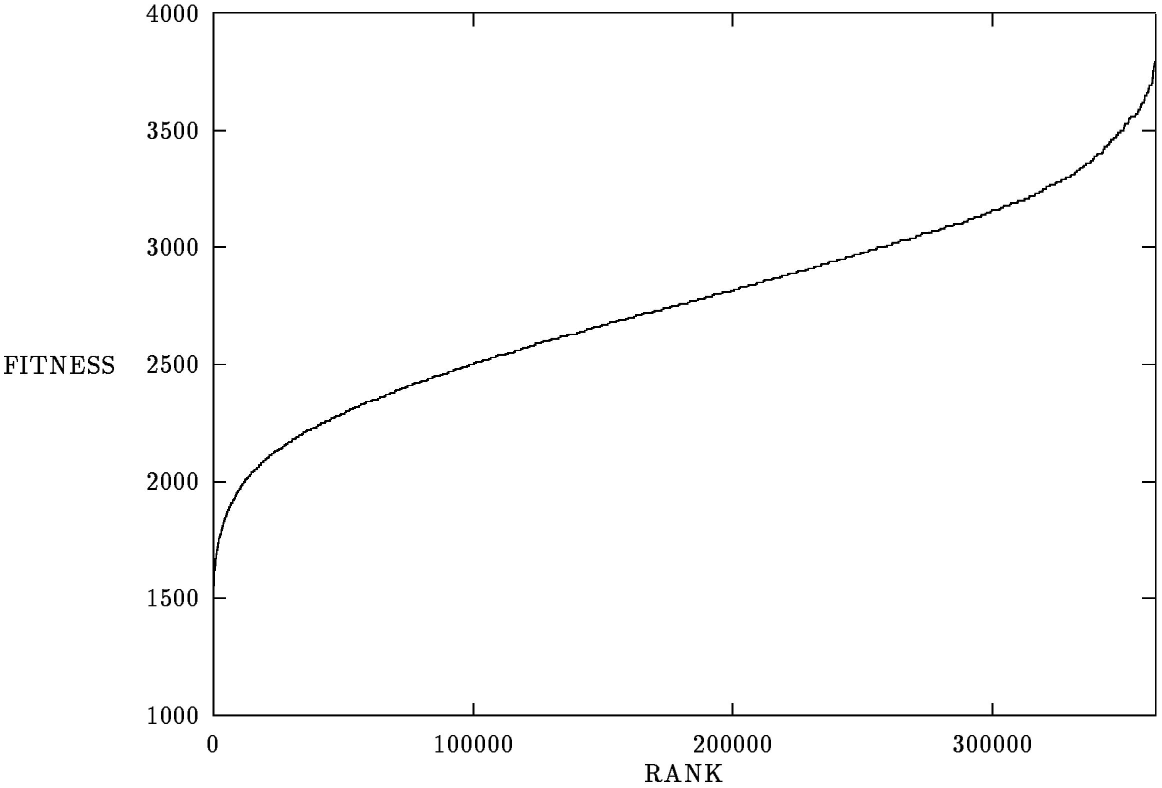
Against Greedy Scheduler

- CPU expense
- best solution limited by GS

One Week, One Line

- Every line maintained once
- None maintained simultaneously
- 9 to schedule (8 lines, 1 week with none)
- Best 1470 (e.g. 3,7,8,4,5,6,,1,2)
- Worst 3870
- Used to adjust fitness parameters

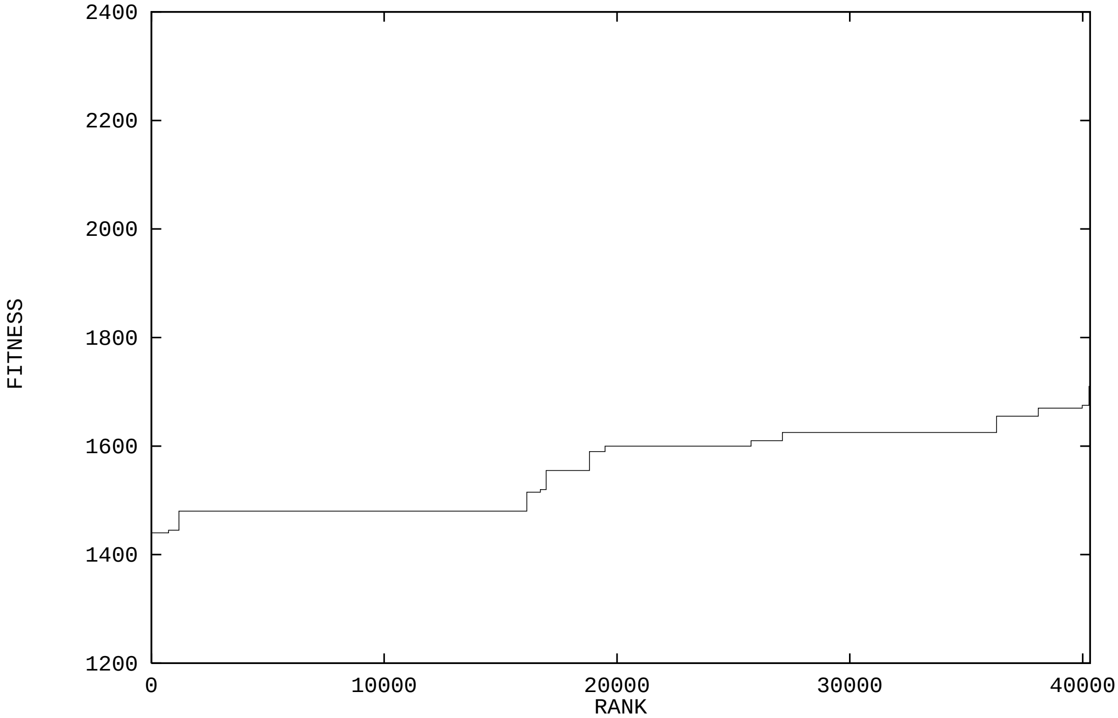
One Week, One Line



Minimum Power Flow

- Maintain all lines once
- choose week with min current
- Best 1440 (e.g. 4,3,567,128,,,,,)
- Worst 1710

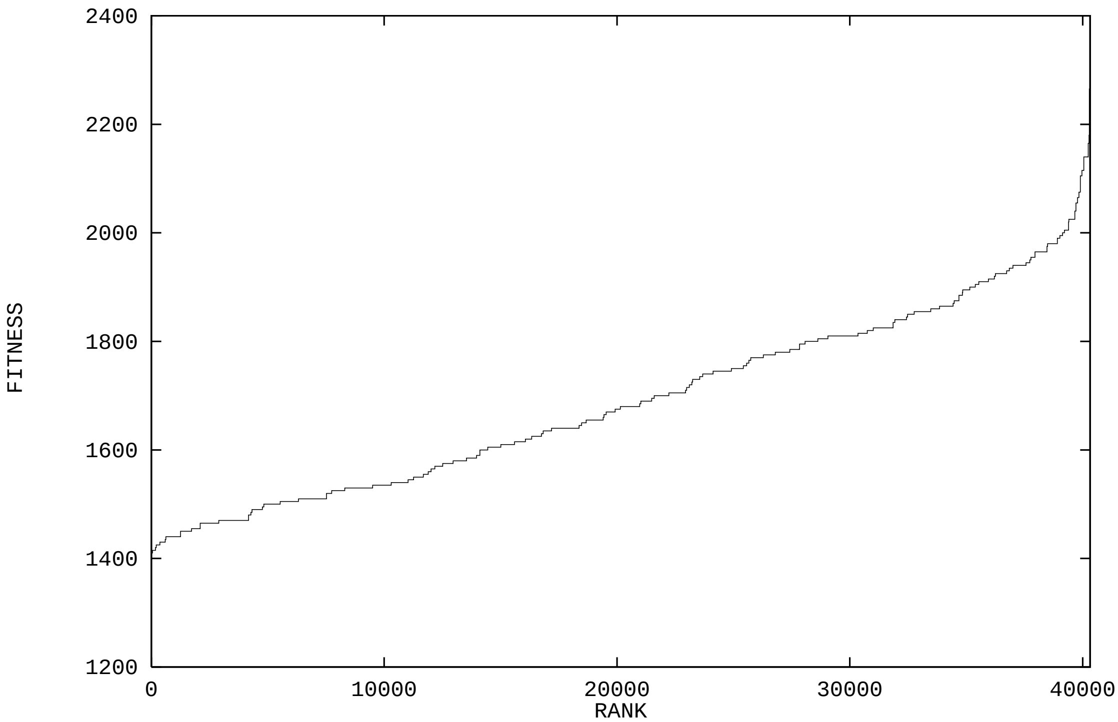
Minimum Power Flow



Minimum Line Cost

- Maintain line in week with min line cost
- As before but:
  don't differentiate between weeks where flow is below limit
- Best 1411.67 (e.g. 4,68,57,3,2,1,,,)
- Worst 2265

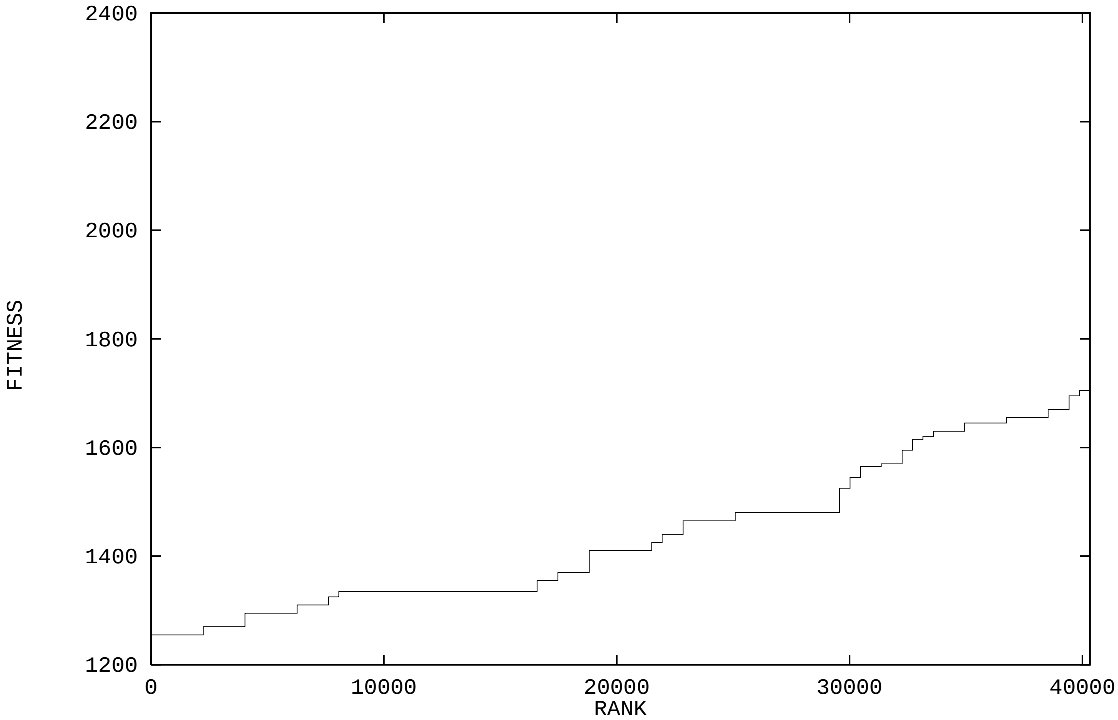
Minimum Line Cost

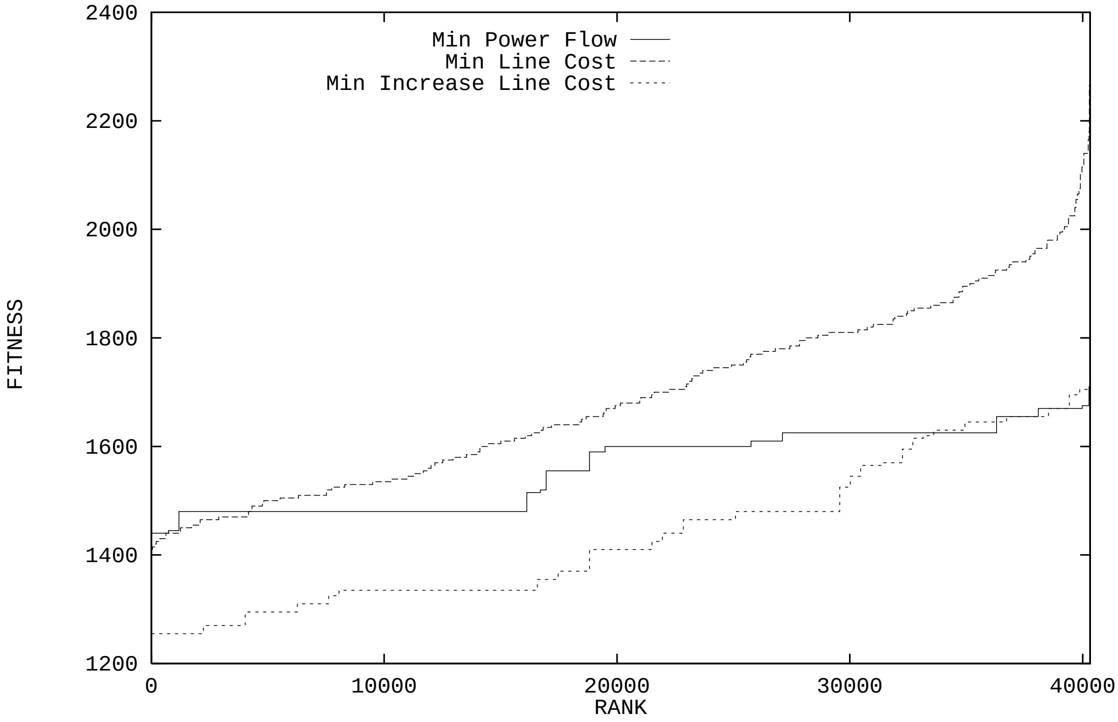


Minimum Increase in Line Cost

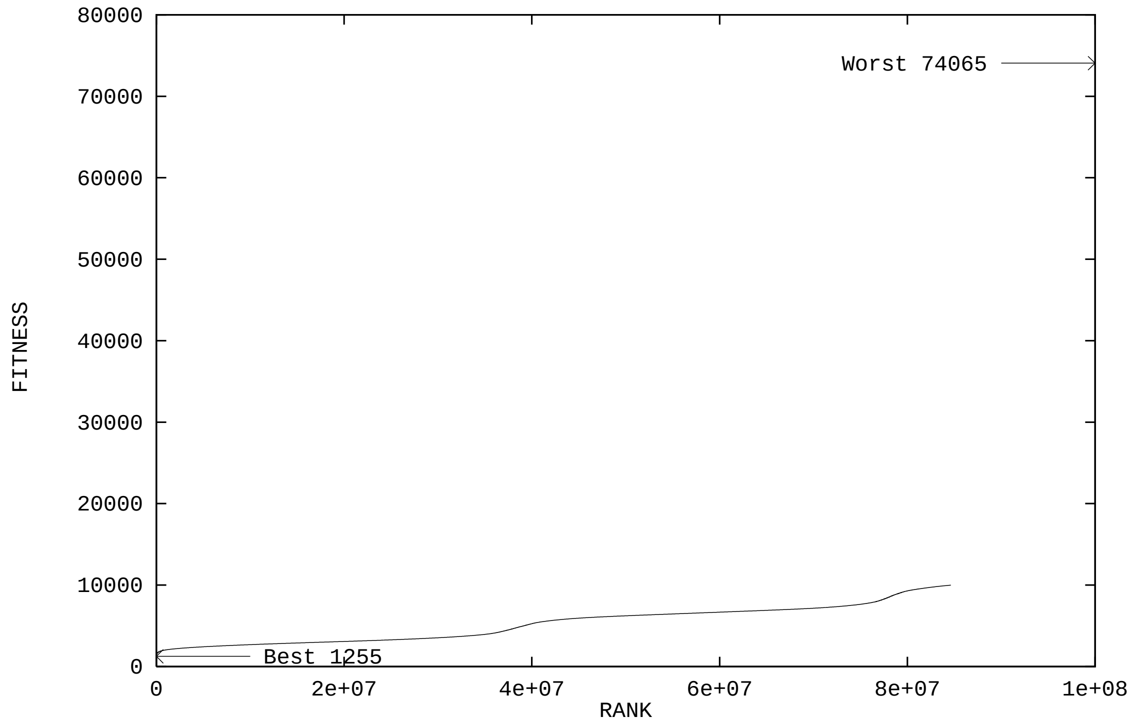
- Maintain line in week where it will cause least increase in line cost
- Scheduler now looks ahead
- Best 1255 (global optimum)
(e.g. 3,4,567,8,,,,1,2)
- Worst 1705

Minimum Increase in Line Cost




Min Power Flow
Min Line Cost
Min Increase Line Cost
FITNESS
RANK
2400
2200
2000
1800
1600
1400
1200
0
10000
20000
30000
40000

- Search space $2^{lines \times weeks} = 4.7\ 10^{21}$
- Graph shows most promising $2.1\ 10^{-14}$
- Each greedy scheduler only considers $8.5\ 10^{-18}$



No line maintained more than once

- QGAME + min increase in line cost
- population 20
- global optimum in generation 4
- a few seconds
- Expected!

- Conclusions, Greedy Optimiser viable
- Future Work
  - Expansive coding?
  - Genetic Programming?
  - Sparse matrix and other improvements
  - South Wales
  - Security
  - Other costs