# Assignment 2
# Selection Perturbative Hyper-Heuristics for Aircraft Landing

**Quinton Weenink, 13176545**[1]

[1]Department of Computer Science – University of Pretoria
Pretoria, South Africa

## 1. Initial solution creation

Initial solutions are created randomly and are immediately thrown out if invalid. Landing times were randomly selected within the earliest and latest times. After all the planes had a landing time the solution was verified. If the separation time between the plane and the corresponding plane was not met the generated solution was thrown out and a new solution was again attempted.

## 2. Low-level perturbative heuristics

### 2.1. Shift up

This low level heuristic sorts the planes in the order that they are currently to land in. In order the heuristic attempts to shift the planes earlier. If a plane can not be moved the next plane in the sorted list is considered if however it can be moved one plane is moved and the low-level heuristic ends. If no plane can be found to be moved up the heuristic will do nothing. The heuristic sorted the solutions in two ways worst to best and best to worst. Each of these were considered separate heuristics.

| t: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| before: | | $x_1$ | | | | | $x_2$ | |
| after: | | $x_1$ | | | $x_2$ | | | |

### 2.2. Shift down

Much like the shift up low-level heuristic above, the shift down heuristic shifts planes in there time if possible reducing the time between landings. The shift down heuristic moves planes later in time attempting to reduce the time between planes as far as possible. This heuristic does not change the order of planes just the time at which they land. Again, two different types of sorted lists were used.

| t: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| before: | | $x_1$ | | | | | $x_2$ | |
| after: | | | | $x_1$ | | | $x_2$ | |

### 2.3. Move worst to target

The low-level heuristic attempts to move a plane that is least fit to its preferred target landing time. If it fails the next plane in the list of planes is considered. If no plane is found to be able to be moved to its target time or is already at its target landing time the heuristic does nothing.

| t: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| before: | | $x_1$ | | | | t | | |
| after: | | | | | | $x_1$ | | |

### 2.4. Move best to target

Rather than attempting to move the worst solution to its target solution, as proposed above, this heuristic attempts to move the best solution to its target. It will not consider a plane that is already at the target landing time as a move. Again, if no plane is found to be able to move to its target landing time then the heuristic will do nothing.

| t: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| before: | | $x_1$ | | | | t | | |
| after: | | | | | | $x_1$ | | |

### 2.5. Swap

This heuristic attempts to swap two planes landing times based on an organized list of planes. One variant uses a list sorted from worst to best the other from best to worst. The heuristic attempts to swap each plane with the next in the list. If the solution is invalid the next plane in the list is attempted.

| t: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| before: | | $x_1$ | | | | $x_2$ | | |
| after: | | $x_2$ | | | | $x_1$ | | |

## 3. Genetic algorithm hyper-heuristic

### 3.1. Population creation

The population is created as a chromosome list of low level heuristics that will be applied in order to the initially randomly generated solution. The chromosomes can be any length from 20 to 30 heuristics in length and are not restricted from growing.

### 3.2. Chromosome operators

The hyper heuristic genetic algorithm used in this study makes use of genetic operators in order to manipulate the existing population. Two operators were used, namely, crossover at a rate of 30% and mutation at a rate of 50%. These operators only changed which heuristics were applied to the solution and did not mutate the solution itself. The remaining 20% was used as the reproduction rate, where nothing was done chromosome and candidates were just placed into the next generation.

### 3.3. Selection

Tournament selection was used to determine which individuals were the worst or best and which individuals the operators will be performed on. A tournament size of 3 was used throughout this research to reduce selective pressure.

The below function describes the fitness function that was used by both the single and multi point search algorithms.

$$f_{ALS} = minimize \sum_{i=1}^{P} (g_i \alpha_i + h_i \beta_i) \tag{1}$$

## 4. Experimental setup

An initial solution was generated and remained constant for that sample. The next sample a different starting solution was generated and the algorithm started over again with a new population.

For comparison random single-point search algorithms as well as a standard genetic algorithm was include in the results. The genetic algorithm directly manipulated the solution space by changing the values of the landing time with crossover and mutation techniques.

Each algorithm was run for 100 iterations/generations. A population size of 30 was used for both the hyper-heuristic genetic algorithm as well as the standard genetic algorithm.

**Table 1. Problems used to test the selective pertubative hyper-heuristic.**

| ***Problem*** | ***File*** | ***P*** |
|---|---|---|
| $P_1$ | airland1 | 10 |
| $P_2$ | airland2 | 15 |
| $P_3$ | airland3 | 20 |
| $P_4$ | airland4 | 20 |
| $P_5$ | airland5 | 20 |

## 5. Results

Table 2 indicates that a multi-point search is almost always better than a random heuristic selection hyper-heuristic. One can also observe that for some problems the hyper-heuristic performs suitably and often gets the the result at a much faster rate.

In problem $P_3$ the multi-point hyper heuristic can even be seen to outperform the standard genetic algorithm.

Without the ability to change previous heuristics in the solutions the single point random hyper-heuristics are not able to outperform and of the other algorithms. This speaks true to the benefit of multi-point searches but does not mean that a more informed single-point search couldn't do better.

**Table 2.** $P_{1-5}$ **after 100 generations. Means and standard deviations are reported over 30 samples.**

| | Algorithm | | | Performance | | |
|---|---|---|---|---|---|---|
| | ***Type*** | ***Algorithm*** | ***Heuristic Selection*** | **mean** | **std** | **min** |
| $P_1$ | hyper heuristic | random | accept all moves | 25702.000000 | 6791.333890 | 8240.0 |
| | | random | improving moves | 24824.666667 | 6048.198483 | 14780.0 |
| | | random | equal or improving moves | 24164.000000 | 4521.671962 | 17640.0 |
| | | genetic | hyper heuristic | 10173.333333 | 4119.201649 | 4660.0 |
| | standard heuristic | genetic | | 7428.000000 | 1497.037964 | 4280.0 |
| $P_2$ | hyper heuristic | random | accept all moves | 39964.000000 | 11948.348170 | 20170.0 |
| | | random | improving moves | 35636.000000 | 8896.806768 | 23750.0 |
| | | random | equal or improving moves | 32386.000000 | 8342.353625 | 16880.0 |
| | | genetic | hyper heuristic | 16870.666667 | 6366.444813 | 6260.0 |
| | standard heuristic | genetic | | 15302.000000 | 3441.449695 | 9180.0 |
| $P_3$ | hyper heuristic | random | accept all moves | 45847.333333 | 6945.592815 | 33880.0 |
| | | random | improving moves | 38452.666667 | 12605.453564 | 22210.0 |
| | | random | equal or improving moves | 32392.000000 | 5504.618909 | 22290.0 |
| | | genetic | hyper heuristic | 19162.000000 | 7617.512455 | 5200.0 |
| | standard heuristic | genetic | | 21955.333333 | 4046.571992 | 15120.0 |
| $P_4$ | hyper heuristic | random | accept all moves | 65648.0 | 7636.684446 | 52950.0 |
| | | random | improving moves | 64280.0 | 12241.595757 | 43920.0 |
| | | random | equal or improving moves | 66854.0 | 10398.229849 | 44160.0 |
| | | genetic | hyper heuristic | 53402.0 | 6752.316343 | 43250.0 |
| | standard heuristic | genetic | | 31278.0 | 3707.846095 | 24980.0 |
| $P_5$ | hyper heuristic | random | accept all moves | 73402.666667 | 13402.433345 | 45140.0 |
| | | random | improving moves | 60447.333333 | 10544.217035 | 39170.0 |
| | | random | equal or improving moves | 62593.333333 | 9824.264971 | 41280.0 |
| | | genetic | hyper heuristic | 54386.000000 | 3192.363388 | 49580.0 |
| | standard heuristic | genetic | | 31503.333333 | 4555.886546 | 24320.0 |

## 6. Comparison

As can be observed in the figures (1,2,3,4,5) the multi-point algorithm outshines the standard single-point random heuristic selection techniques. And initially outperforms a standard genetic algorithm.

In figure 3 one can observe the benefit and speed of the hyper-heuristic beating the standard genetic algorithm to 100 generations. It should be noted however that given more iterations that standard genetic algorithm would obtain a better performance.

Again the single-point hyper-heuristics fall short but further research should be done on other single-point hyper-heuristics that could potentially outperform the ones examined in this research.

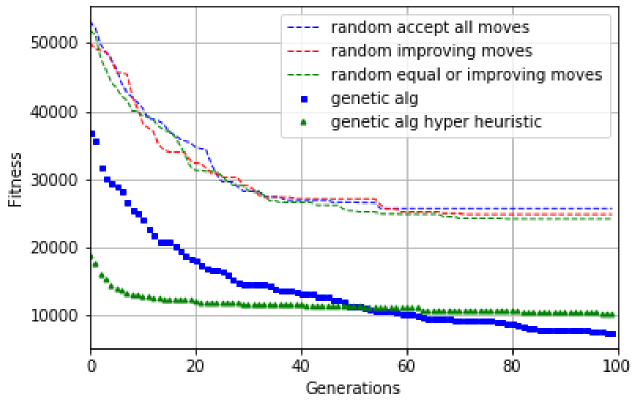


**Figure 1.** $airland1$ **mean fitness over 30 samples for 100 generations**

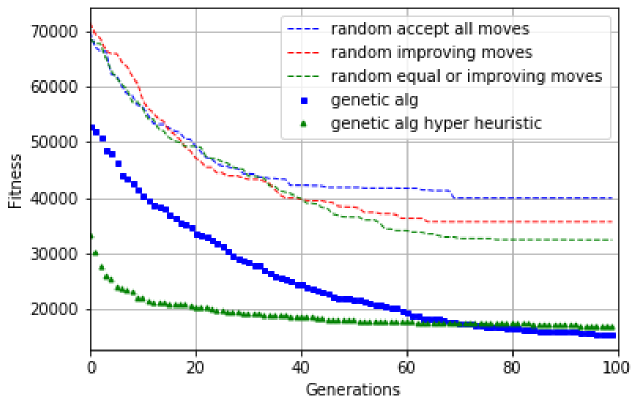


**Figure 2.** $airland2$ **mean fitness over 30 samples for 100 generations**

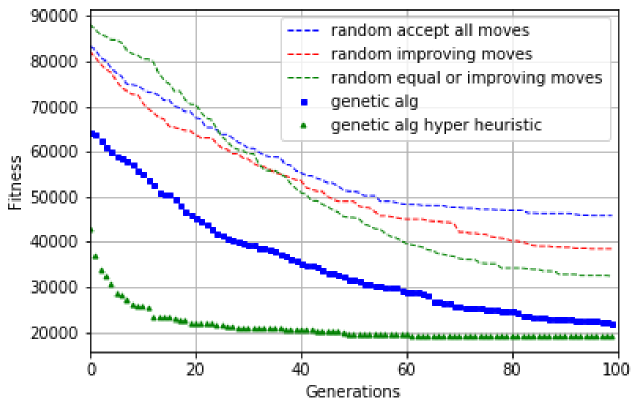


**Figure 3.** $airland3$ **mean fitness over 30 samples for 100 generations**

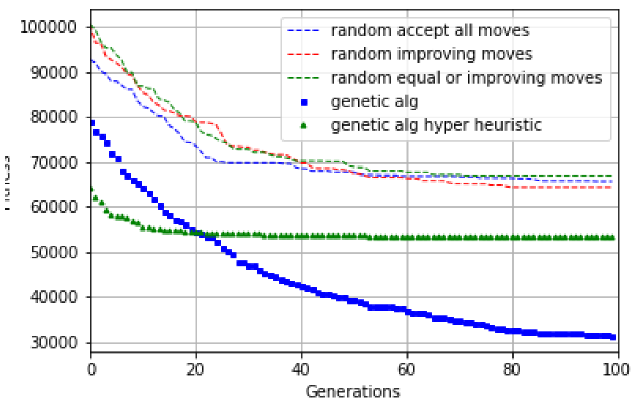


**Figure 4.** $airland4$ **mean fitness over 30 samples for 100 generations**

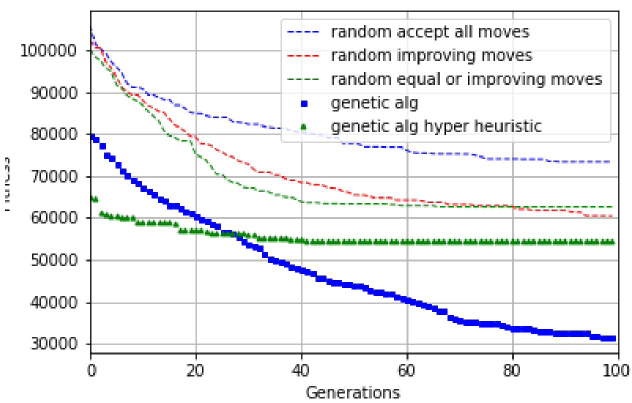


**Figure 5.** $airland5$ **mean fitness over 30 samples for 100 generations**