MINFN Problem Specification

The *minfn* problem is a function optimisation problem. The *minfn* function takes values from an input vector with N vector-elements. The first N-1 vector-element values are used as coefficients $C_1..C_{N-1}$ in an equation of the form:

$$C_1*a + C_2*b + C_3*c + \ldots C_{N-1}*d = E$$

The N^{th} vector-element value is `E`.

Only one input vector is read for any given *minfn* problem, and all chromosomes in the population are candidate solutions for the same *minfn* problem. So, for example, if the vector input data file contained the following:

```
InputVector:0(1,3,4,2,40)
```

… that information would represent the equation:

$$a + 3b + 4c + 2d = 40$$

The task of *minfn* is to determine values for the variables (such as `a,b,c,d` in the above example) which solve the equation (for example `a = 9, b = 3, c = 2, d = 7` is one such solution for the above equation).

MINFN Chromosome representation

The task of *minfn* is to determine values for the variables which solve an equation of the form

$$C_1*a + C_2*b + C_3*c + \ldots C_{N-1}*d = E$$

If the input vector had N vector-elements, then the chromosome structure for *minfn* is N-1 integers, the first of which represents the first variable (`a`, in the above example) the second represents the next variable in the equation (`b`, in the above example) and so on.

For example, a *minfn* chromosome for `a = 9, b = 3, c = 2, d = 7` would be:

9	3	2	7

Each element of a *minfn* chromosome is called an *allele*.

Valid values for a *minfn* allele is the range 0..MINFN_MAX, inclusive.

Function for creating a random MINFN chromosome

The function for creating a 'random' *minfn* chromosom is `create_minfn_chrom`

MINFN Evaluation function

The *minfn* evaluation function calculates the raw score for a *minfn* chromosome.

The raw score for a *minfn* chromosome representing variables `a,b,c,d` is the value:

$$\texttt{abs}(C_1\texttt{*a} + C_2\texttt{*b} + C_3\texttt{*c} + C_4\texttt{*d} - \texttt{E})$$

… where coefficients $C_1..C_4$ and the value `E` are vector-element values from an input vector as described above.

So for example, for the *minfn* chromosome

9	1	2	7

… given an input vector for the *minfn* problem of

```
InputVector:0(1,3,4,2,40)
```

.. the raw score of the chromosome would be

```
abs(1*9 + 3*1 + 4*2 + 2*7 - 40)
```

… which is `6`

The function for implementing *minfn* evaluation is `eval_minfn`

Mutation operation for MINFN chromosome

Mutation of a *minfn* chromosome involves selecting an allele index at 'random' and changing its value to another valid 'random' *minfn* chromosome allele.

For example, mutation of the allele index 1 of chromosome:

9	3	2	7

… may alter its value from 3 to 11, resulting in a chromosome of:

9	11	2	7

The function for implementing *minfn* mutation is `mutate_minfn`

Crossover operation for MINFN chromosomes

Crossover of two 'parent' *minfn* chromosomes produces a new 'child' *minfn* chromosome as follows:

Assume two 'parent' *minfn* chromosomes, p1 and p2, as follows:

p1:

8	0	6	2

p2:

1	5	3	4

'Randomly' choose an allele index position (in this case index 1). Index positions 0..1 inclusive of p1 are copied to the child chromosome. Index positions 2..3 inclusive of p2 are copied to the child chromosome.

child:

8	0	3	4

The function for implementing *minfn* crossover is `crossover_minfn`