```cpp
#include "ionlib\log.h"
#include "ionlib\net.h"
#include "ionlib\genetic_algorithm.h"
#include <fstream>
#include <bitset>
#include <sstream>

int32_t signed_vector_to_int(std::vector<bool>::iterator first,
  std::vector<bool>::iterator end)
{
    LOGASSERT(end - first <= 32);
    uint32_t result = 0;
    for (std::vector<bool>::iterator it = first; it < end-1; ++it)
    {
        if (*it)
        {
            uint32_t offset = (uint32_t)(it - first);
            result |= 1 << offset;
        }
    }
    int32_t sign = (*(end-1)) ? -1 : 1;
    return sign * (int32_t)result;
}
class GANumOnes : public ion::GeneticAlgorithm
{
public:
    GANumOnes() = delete;
    GANumOnes(size_t num_members, size_t chromosome_length, double
      mutation_probability, double crossover_probability) : ion::GeneticAlgorithm
      (num_members, chromosome_length, mutation_probability,
      crossover_probability)
    {
        EvaluateMembers();
    }
    virtual void EvaluateMembers()
    {
        for (std::vector<std::vector<bool>>::iterator member_it = this-
          >population_.begin(); member_it != this->population_.end(); +
          +member_it)
        {
            double fitness = 0.0;
            for (std::vector<bool>::iterator gene_it = member_it->begin();
              gene_it != member_it->end(); ++gene_it)
            {
                if (*gene_it)
                {
                    fitness += 1.0 / member_it->size();
                }
            }
            this->fitness_[member_it - this->population_.begin()] = fitness;
            this->num_evaluations_++;
        }
    }
```

```
47  };
48
49  double dejong1(double x[3])
50  {
51      return x[0]*x[0] + x[1]*x[1] + x[2]*x[2];
52  }
53
54  class GADejong1 : public ion::GeneticAlgorithm
55  {
56  public:
57      GADejong1() = delete;
58      GADejong1(size_t num_members, double mutation_probability, double
          crossover_probability) : ion::GeneticAlgorithm(num_members,
          num_chromosomes_*chromosome_length_, mutation_probability,
          crossover_probability)
59      {
60          double worst_x[3];
61          worst_x[0] = worst_x[1] = worst_x[2] = -5.12;
62          worst_fitness_ = dejong1(worst_x);
63          EvaluateMembers();
64      }
65      virtual void EvaluateMembers()
66      {
67          for (std::vector<std::vector<bool>>::iterator member_it = this-
              >population_.begin(); member_it != this->population_.end(); +
              +member_it)
68          {
69              this->num_evaluations_++;
70              //convert to a value in range
71              double x[num_chromosomes_];
72              to_val(*member_it, x);
73              //evaluate
74              double raw_fitness = dejong1(x);
75              //scale to [0.0,1.0]
76              double fitness = (worst_fitness_ - raw_fitness) / worst_fitness_;
77              LOGASSERT(fitness <= 1.0 && fitness >= 0.0);
78              this->fitness_[member_it - population_.begin()] = fitness;
79          }
80      }
81      static const uint32_t num_chromosomes_ = 3;
82      static const uint32_t chromosome_length_ = 10;
83      double worst_fitness_;
84      void to_val(std::vector<bool> member, double x[num_chromosomes_])
85      {
86          for (uint32_t dim = 0; dim < num_chromosomes_; ++dim)
87          {
88              int32_t member_offset = signed_vector_to_int(member.begin() +
                  dim*chromosome_length_, member.begin() + (dim + 1)
                  *chromosome_length_);
89              x[dim] = (double)member_offset / 100.0;
90          }
91      }
92  };
```

```
 93 double dejong2(double x[2])
 94 {
 95     return 100 * pow(x[0]*x[0] - x[1], 2.0) + pow(1 - x[0], 2.0);
 96 }
 97
 98 class GADejong2 : public ion::GeneticAlgorithm
 99 {
100 public:
101     GADejong2() = delete;
102     GADejong2(size_t num_members, double mutation_probability, double
        crossover_probability) : ion::GeneticAlgorithm(num_members,
        num_chromosomes_*chromosome_length_, mutation_probability,
        crossover_probability)
103     {
104         double worst_x[2];
105         worst_x[0] = worst_x[1] = -2.048;
106         worst_fitness_ = dejong2(worst_x);
107         EvaluateMembers();
108     }
109     virtual void EvaluateMembers()
110     {
111         for (std::vector<std::vector<bool>>::iterator member_it = this-
            >population_.begin(); member_it != this->population_.end(); +
            +member_it)
112         {
113             this->num_evaluations_++;
114             //convert to a value in range
115             double x[num_chromosomes_];
116             to_val(*member_it, x);
117             //evaluate
118             double raw_fitness = dejong2(x);
119             //scale to [0.0,1.0]
120             double fitness = (worst_fitness_ - raw_fitness) / worst_fitness_;
121             LOGASSERT(fitness <= 1.0 && fitness >= 0.0);
122             this->fitness_[member_it - population_.begin()] = fitness;
123         }
124     }
125     static const uint32_t num_chromosomes_ = 2;
126     static const uint32_t chromosome_length_ = 12;
127     double worst_fitness_;
128     void to_val(std::vector<bool> member, double x[num_chromosomes_])
129     {
130         for (uint32_t dim = 0; dim < num_chromosomes_; ++dim)
131         {
132             int32_t member_offset = signed_vector_to_int(member.begin() +
                dim*chromosome_length_, member.begin() + (dim + 1)
                *chromosome_length_);
133             x[dim] = (double)member_offset / 1000.0;
134         }
135     }
136 };
137
138 double dejong3(double x[5])
```

```
139 {
140     return (double)((int32_t)x[0] + (int32_t)x[1] + (int32_t)x[2] + (int32_t)x[3]
        + (int32_t)x[4]);
141 }
142
143 class GADejong3 : public ion::GeneticAlgorithm
144 {
145 public:
146     GADejong3() = delete;
147     GADejong3(size_t num_members, double mutation_probability, double
        crossover_probability) : ion::GeneticAlgorithm(num_members,
        num_chromosomes_*chromosome_length_, mutation_probability,
        crossover_probability)
148     {
149         double worst_x[5];
150         worst_x[0] = worst_x[1] = worst_x[2] = worst_x[3] = worst_x[4] = 5.12;
151         worst_fitness_ = dejong3(worst_x);
152         EvaluateMembers();
153     }
154     virtual void EvaluateMembers()
155     {
156         for (std::vector<std::vector<bool>>::iterator member_it = this-
            >population_.begin(); member_it != this->population_.end(); +
            +member_it)
157         {
158             this->num_evaluations_++;
159             //convert to a value in range
160             double x[num_chromosomes_];
161             to_val(*member_it, x);
162             //evaluate
163             double raw_fitness = dejong3(x) + worst_fitness_;
164             //scale to [0.0,1.0]
165             double fitness = (worst_fitness_*2 - raw_fitness) /
                (2*worst_fitness_);
166             LOGASSERT(fitness <= 1.0 && fitness >= 0.0);
167             this->fitness_[member_it - population_.begin()] = fitness;
168         }
169     }
170     static const uint32_t num_chromosomes_ = 5;
171     static const uint32_t chromosome_length_ = 10;
172     double worst_fitness_;
173     void to_val(std::vector<bool> member, double x[num_chromosomes_])
174     {
175         for (uint32_t dim = 0; dim < num_chromosomes_; ++dim)
176         {
177             int32_t member_offset = signed_vector_to_int(member.begin() +
                dim*chromosome_length_, member.begin() + (dim + 1)
                *chromosome_length_);
178             x[dim] = (double)member_offset / 100.0;
179         }
180     }
181 };
182
```

```
183 double dejong4(double x[30])
184 {
185     double result = 0.0;
186     for (uint32_t i = 0; i < 30; ++i)
187     {
188         double random_number = ion::random_normal_distribution(0.0, 1.0);
189         result += i * pow(x[i], 4) + random_number;
190     }
191     return result;
192 }
193
194 class GADejong4 : public ion::GeneticAlgorithm
195 {
196 public:
197     GADejong4() = delete;
198     GADejong4(size_t num_members, double mutation_probability, double
        crossover_probability) : ion::GeneticAlgorithm(num_members,
        num_chromosomes_*chromosome_length_, mutation_probability,
        crossover_probability)
199     {
200         //note that we can't actually define a worst X for this function since it
            is random, however it is extremely unlikey we would exceed this value
201         double worst_x[30];
202         for (uint32_t x_index = 0; x_index < 30; ++x_index)
203         {
204             worst_x[x_index] = 1.28;
205         }
206         worst_fitness_ = dejong4(worst_x);
207         EvaluateMembers();
208     }
209     virtual void EvaluateMembers()
210     {
211         for (std::vector<std::vector<bool>>::iterator member_it = this-
            >population_.begin(); member_it != this->population_.end(); +
            +member_it)
212         {
213             this->num_evaluations_++;
214             //convert to a value in range
215             double x[num_chromosomes_];
216             to_val(*member_it, x);
217             //evaluate
218             double raw_fitness = dejong4(x) + worst_fitness_;
219             //scale to [0.0,1.0]
220             double fitness = (worst_fitness_ * 2 - raw_fitness) / (2 *
                worst_fitness_);
221             LOGASSERT(fitness <= 1.0 && fitness >= 0.0);
222             this->fitness_[member_it - population_.begin()] = fitness;
223         }
224     }
225     static const uint32_t num_chromosomes_ = 30;
226     static const uint32_t chromosome_length_ = 8;
227     double worst_fitness_;
```

```
228     void to_val(std::vector<bool> member, double x[num_chromosomes_])
229     {
230         for (uint32_t dim = 0; dim < num_chromosomes_; ++dim)
231         {
232             int32_t member_offset = signed_vector_to_int(member.begin() +
                    dim*chromosome_length_, member.begin() + (dim + 1)
                    *chromosome_length_);
233             x[dim] = (double)member_offset / 100.0;
234         }
235     }
236 };
237
238 void ExecuteGa(uint32_t population_size, double mutation_rate, double
      crossover_rate)
239 {
240
241     std::ofstream fout;
242     uint32_t dejong_num = 4;
243     std::stringstream filename;
244     filename << "DJ" << dejong_num << "_pop" << population_size << "_mut" <<
          mutation_rate << "_xover" << crossover_rate << ".csv";
245     fout.open(filename.str());
246     fout << "Generation,Min,Max,Mean,Evals" << std::endl;
247     double max_fitness[5000] = { 0 };
248     double min_fitness[5000] = { 0 };
249     double avg_fitness[5000] = { 0 };
250     double num_evals[5000] = { 0 };
251     double num_hits[5000] = { 0 };
252     for (uint32_t trial = 0; trial < 30; ++trial)
253     {
254         //Change this next line to switch between functions
255         GADejong4 algo(population_size, mutation_rate, crossover_rate);
256         uint32_t generation = 0;
257         max_fitness[generation] += algo.GetMaxFitness();
258         min_fitness[generation] += algo.GetMinFitness();
259         avg_fitness[generation] += algo.GetAverageFitness();
260         num_evals[generation] += algo.GetNumEvals();
261         num_hits[generation]++;
262         for (generation = 1; algo.GetMaxFitness() < 0.99999999 && generation <
              5000; ++generation)
263         {
264             algo.NextGeneration();
265             max_fitness[generation] += algo.GetMaxFitness();
266             min_fitness[generation] += algo.GetMinFitness();
267             avg_fitness[generation] += algo.GetAverageFitness();
268             num_evals[generation] += algo.GetNumEvals();
269             num_hits[generation]++;
270         }
271         LOGINFO("Completed trial %u", trial);
272     }
273     //scale all of the computed values
274     for (uint32_t generation_index = 0; generation_index < 5000; +
```

```
      +generation_index)
275     {
276         if (num_hits[generation_index] == 0)
277         {
278             break;
279         }
280         max_fitness[generation_index] /= num_hits[generation_index];
281         min_fitness[generation_index] /= num_hits[generation_index];
282         avg_fitness[generation_index] /= num_hits[generation_index];
283         num_evals[generation_index] /= num_hits[generation_index];
284         fout << generation_index << "," << min_fitness[generation_index] << ","
              << max_fitness[generation_index] << "," << avg_fitness
              [generation_index] << "," << num_evals[generation_index] << std::endl;
285     }
286     fout.close();
287 }
288
289 int main(int argc, char* argv[])
290 {
291     ion::Error result = ion::InitSockets();
292     ion::LogInit("genetic_algorithm");
293     //open a file for logging results
294     uint32_t population_set[3] = { 50, 100, 150 };
295     double mutation_set[3] = { 0.0001, 0.001, 0.01 };
296     double crossover_set[3] = { 0.2, 0.67, 0.99 };
297     for (uint32_t pop_choice = 0; pop_choice < 3; ++pop_choice)
298     {
299         for (uint32_t mutation_choice = 0; mutation_choice < 3; +
              +mutation_choice)
300         {
301             for (uint32_t crossover_choice = 0; crossover_choice < 3; +
                  +crossover_choice)
302             {
303                 ExecuteGa(population_set[pop_choice], mutation_set
                      [mutation_choice], crossover_set[crossover_choice]);
304                 LOGINFO("Completed pop %d, mutation %d, crossover %d",
                      pop_choice, mutation_choice, crossover_choice);
305             }
306         }
307     }
308     return 0;
309 }
```