**Отчет по модулю прогнозирования**

Содержание

1. Общая информация
2. Структура
3. Используемые технологии
4. Генетический алгоритм
5. План дальнейшей разработки

**Общая информация**

Модуль прогнозирования – это информационная подсистема, использующая алгоритмы ИИ и методы, основанные на эволюционных вычислениях, для обработки данных о повреждениях ветрогенераторов, выявления существующих неисправностей, оценки их критичности и прогнозирования вероятности их возникновения в будущем.

**Структура модуля прогнозирования и принцип работы**

Модуль прогнозирования состоит из следующих частей:

1. Базы данных
2. Программа с использованием генетического алгоритма
3. Нейронная сеть

Принцип работы:

База данных заполняется информацией о повреждениях ветрогенераторов, полученной с модуля распознавания неисправностей. Эта информация поступает в формате таблицы, в которой каждая строка описывает состояние одного ветрогенератора в определенную дату проверки. Столбцы описывают состояние показателей, таких как состояние лопастей, мачты, гондолы и т.д. Значения в данной таблице имеют формат целого числа от 0 до 5, где 0 – отсутствие повреждений, а 5 – критическая неисправность.

Из базы данных эта информация поступает в программу, в основе которой лежит генетический алгоритм. Он обрабатывает записи, используя метод, схожий с процессом эволюции, и находит наилучшее значение целевой функции и, соответственно, записи таблицы с наиболее высокими значениями показателей повреждения с учетом весовых коэффициентов важности критериев.

Генетический алгоритм позволяет произвести эффективный по времени и ресурсам поиск ветрогенераторов с наиболее серьезными неисправностями, но использует только данные, полученные после 1-2 проверок. Для того, чтобы иметь возможность спрогнозировать возникновение поломок в будущем, необходимо изучать данные, собранные за продолжительный период времени, чтобы проанализировать временной ряд, выявить закономерности, частоту возникновения различных неисправностей и вычислить вероятности, что некоторая неисправность повторится в будущем.

Для этой цели используется нейронная сеть, которая обучается на данных о неисправностях за несколько лет, и при очередной проверке формирует прогноз о поломках, которые вероятнее всего возникнут в ближайшее время. При этом учитывается время выработки генераторов, частота обслуживания, климат в регионе, сезон и прочие факторы.

По окончанию работы модуля прогнозирования формируется лист обслуживания, в котором обозначено текущее состояние ветрогенераторов с указанием поврежденных частей и степени серьезности повреждения, а также наиболее вероятные неисправности в ближайшем будущем. Помимо этого, лист обслуживания включает в себя список рекомендуемых действий по устранению и предотвращению неисправностей, а также их приоритет.

**Используемые технологии**

1. Базы данных – MS SQL Server;
2. Программа, использующая генетический алгоритм, написана на языке С++ в виде консольного приложения, с использованием библиотеки sql.h для связи с MS SQL ServerW;
3. Нейронная сеть реализуется на языке Python 3, с использованием готовых библиотек для работы с искусственным интеллектом (PyTorch, TensorFlow).

**Генетический алгоритм**

Генетический алгоритм (ГА) — это эвристический алгоритм поиска, основанный на применении методов естественного отбора и эволюции, и используемый для решения задач моделирования, оптимизации и прогнозирования.

Ключевые функции генетического алгоритма:

1) Создание первоначальной популяции;

2) Мутация;

3) Кроссигновер;

4) Вычисление приспособленности (fitness-функция);

5) Турнирный отбор.

Для более подробного описания способа реализации данных методов в разработанном алгоритме приведен псевдокод:

Псевдокод 1. Создание начальной популяции

vector<vector<NodeState>> initialize\_population(int size, int num\_nodes) {

vector<vector<NodeState>> population(size, vector<NodeState>(num\_nodes)); //Создаем вектор для хранения популяции

for (int i = 0; i < size; ++i) {

for (int j = 1; j < num\_nodes; ++j) {

population[i][j] = {

rand() % 5, // blades\_mechanical\_damage

rand() % 5, // flaps\_condition

rand() % 5, // corrosion

rand() % 5, // icing

rand() % 5, // rotary\_mechanism

rand() % 5, // wind\_sensor

};

}//Заполняем параметры случайными числами от 0 до 5

}

return population;

}

Первоначальная популяция формируется путем присвоения каждому показателю случайно выбранного целочисленного значения от 0 до 5.

Псевдокод 2. Мутация

void mutate(std::vector<NodeState>& individual, double mutation\_rate) {

for (size\_t i = 1; i < individual.size(); ++i) {

if (static\_cast<double>(rng()) / rng.max() < mutation\_rate) {

individual[i].flaps\_condition = 5 - individual[i].flaps\_condition;

individual[i].blades\_mechanical\_damage = 5 - individual[i].blades\_mechanical\_damage;

individual[i].corrosion = 5 - individual[i].corrosion;

individual[i].icing = 5 - individual[i].icing;

individual[i].rotary\_mechanism = 5 - individual[i].rotary\_mechanism;

individual[i].wind\_sensor = 5 - individual[i].wind\_sensor;

}

}

}

Функция мутации принимает в качестве параметров одного из “индивидов”, и значение mutation\_rate, от которого зависит количество мутирующих параметров данного индивида. Далее случайно выбранные показатели меняются.

Псевдокод 3. Кроссинговер

std::vector<NodeState> crossover(const std::vector<NodeState>& parent1, const std::vector<NodeState>& parent2) {

std::vector<NodeState> child(parent1.size());

for (size\_t i = 1; i < parent1.size(); ++i) {

child[i] = (rng() % 2) ? parent1[i] : parent2[i];

}

return child;

}

С помощью функции кроссинговера, представленной в псевдокоде 3, из двух родительских индивидов формируется индивид потомка. Случайно выбранные параметры в равном количестве наследуются от каждого из родителей.

Псевдокод 4. Fitness-функция

double fitness(const std::vector<NodeState>& individual, const std::vector<std::vector<int>>& G) {

double fitness\_value = 0.0;

for (int node = 1; node < G.size(); ++node) {

const auto& state = individual[node];

fitness\_value += state.flaps\_condition \* 3;

fitness\_value += state.blades\_mechanical\_damage \* 2;

fitness\_value += state.corrosion \* 1;

fitness\_value += state.icing \* 0.1;

fitness\_value += state.rotary\_mechanism \* 0.5;

fitness\_value += state.wind\_sensor \* 0.5;

}

return fitness\_value;

}

В псевдокоде 4 показана целевая функция, которая определяет уровень “приспособленности индивида”. В данном случае функция fitness определяет, насколько оптимальным является каждое найденное решение.

vector<NodeState> tournament\_selection(const vector<vector<NodeState>>& population, const vector<double>& fitnesses, int k) {

Псевдокод 5. Турнирный отбор

vector<int> selected;

for (int i = 0; i < k; ++i) {

int ind = rng() % population.size();

selected.push\_back(ind);

}

int best = selected[0];

for (int i = 1; i < k; ++i) {

if (fitnesses[selected[i]] > fitnesses[best]) {

best = selected[i];

}

}

return population[best];

}

После выполнения всех предыдущих операций и вычисления приспособленности для индивида последующая популяция формируется методом турнирного отбора наиболее приспособленных индивидов, т. е. решений, показавших наиболее высокие значения целевой функции. Остальные, менее приспособленные индивиды, отсеиваются таким образом, чтобы размер популяции всегда оставался постоянным.

**План дальнейшей разработки**

На данный момент реализована программа, принимающая данные из БД и обрабатывающая их с помощью генетического алгоритма.

Далее планируется расширить эту программу, включив в нее большее количество показателей для более точной оценки состояния ветрогенераторов, а также отладка и настройка алгоритма путем изменения весовых коэффициентов показателей. Также планируется написать нейронную сеть, которая на данный момент еще не реализована.

В дальнейшем необходимо будет объединить все части модуля в единую подсистему, которую, в свою очередь, объединить с аппаратным модулем, модулем маршрутизации и модулем распознавания неисправностей в одно приложение. Данное приложение, после тщательной настройки, отладки взаимодействия составных частей, исправления ошибок и достаточного обучения нейронных сетей, будет представлять собой систему, готовую к эксплуатации.