Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования «Московский государственный технический университет
имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ *Робототехника и комплексная автоматизация*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Студент *Новокшанов Евгений Андреевич*
(фамилия, имя, отчество)

Группа *РК6-76Б*

Тип практики Эксплуатационная

Название предприятия *DATADVANCE*

Студент РК6-76Б (Группа) *Новокшанов Е.А.* (подпись, дата)

Руководитель практики от кафедры *Оглоблин Д.И.* (подпись, дата)

Оценка ____________

Москва, 2023 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой *РК-6*
(Индекс)

______________ *А.П. Карпенко*
(И.О.Фамилия)

«____» ____________ *2023* г.

З А Д А Н И Е
на прохождение эксплуатационной практики

Студент *Новокшанов Евгений Андреевич, 4* курса, группы *РК6-76Б*
(фамилия, имя, отчество, № курса, индекс группы)

в период с *«03» июля 2023 г.* по *«03» августа 2023 г.*

Предприятие: *DATADVANCE*

Подразделение: ______________________
(отдел/сектор/цех)

Руководитель практики от предприятия (наставник):
Агасиев Талех Азер оглы
(фамилия имя отчество полностью, должность)

Руководитель практики от кафедры:
Оглоблин Дмитрий Игоревич
(фамилия имя отчество полностью, должность)

Задание.
1. Исследовать способы приведения задачи удовлетворение ограничениям к задаче безусловной оптимизации;
2. Исследовать эффективность стохастических методов оптимизации на приведенных задачах удовлетворения ограничениям;
3. Проверить работоспособность полученной программной реализации.

Дата выдачи задания *«03» июля 2023 г.*

Руководитель практики от предприятия ______________ *Агасиев Т.А.*
(подпись, дата)

Руководитель практики от кафедры ______________ *Оглоблин Д.И.*
(подпись, дата)

Студент РК6-76Б ______________ *Новокшанов Е.А.*
(группа) (подпись, дата)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Компания DATADVANCE, на базе которой проходит практика, является отечественным разработчиком программного обеспечения в области оптимизации, анализа данных, предиктивного моделирования и сред для совместной инженерной работы.

Одной из задач компании является разработка и применение методов условной оптимизации, которые применяются для повышения эффективности механических систем.

Оптимизационная задача – это задача нахождения экстремума (минимума или максимума) целевой функции в некоторой области конечномерного векторного пространства, ограниченной набором в общем случае нелинейных ограничений.

Задачи удовлетворения ограничениям (CSP) являются частным случаем типичных задач нелинейного программирования, целевые функции которых отсутствуют.

Подобные задачи оптимизации часто встречаются в инженерной отрасли, когда необходимо подобрать любой допустимый набор параметров модели, детали или механизма, для соответствия заданным требованиям.

Существует много открытых реализаций эффективных алгоритмов оптимизации, которые не поддерживают постановку задач оптимизации без целевых функций – поиск точки, удовлетворяющей всем функциональным ограничениям.

Цель практики - исследовать способы приведения такой задачи к задаче оптимизации с целевой функции, чтоб иметь возможность использовать эти библиотеки.

Было предложно использовать библиотеку Nevergrad, так как это популярная библиотека со стохастическими алгоритмами.

1. ПОСТАНОВКА ЗАДАЧИ

Дан X – вектор входных параметров, где $X_i^- \leq X_i \leq X_i^+$,где X_i^- – нижняя граница, X_i^+ – верхняя граница соответствующего i-го параметра, $X_0 -$ начальное приближение, и набор функциональных ограничений $C_1, C_2, \dots, C_n$, вида $C_j^- < C_j(X) < C_j^+$, где C(X) – функция от вектора параметров X, C_j^- – нижняя граница функционального ограничения $C_j(X)$, $C_j^+ -$ верхняя граница функционального ограничения $C_j(X)$.

Необходимо найти такой X, который принадлежит допустимой области и для которого выполняются все функциональные ограничения.

Необходимо исследовать:

- Варианты приведения задачи оптимизации CSP к задаче безусловной оптимизации;
- Эффективность стохастических методов оптимизации на приведенных задачах оптимизации.

По результатам исследований проверить работоспособность программной реализации задачи.

2. АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ

Задача CSP приводится к задаче безусловной оптимизации путем приведения к целевой функции, как функции от значений функциональных ограничений на каждом шаге работы алгоритма оптимизации.

Было предложено использовать целевую функцию вида

$$F(X) = -1 + \psi(X),$$

где $\psi(X)$ функция нарушения допустимой области.

Постановка задачи безусловной оптимизации имеет следующий вид:

$$\min_X F(X) = \min_X(-1 + \psi(X)).$$

$\psi_j(X)$ для j-го функционального ограничения рассчитывается по следующей формуле:

$$\psi_j(X) = \max\{\frac{C_j^- - C_j(X)}{\max(1, |C_j^-|)}, \frac{C_j(X) - C_j^+}{\max(1, |C_j^+|)}\},$$

Было выявлено несколько методов формирования целевой функции в приведенной задаче:

1. Рассматривать $\psi(X)$, как максимум от всех $\psi_j(X)$ для X на текущей итерации;

$$\psi(X) = \max\{\psi_j(X), \qquad j \in \overline{[1, m]}\}$$

2. Рассматривать $\psi(X)$, как среднее значение от всех нарушений допустимой области для X на текущей итерации;

$$\psi(X) = \frac{\sum_{j=1}^{m} \psi_j(X)}{m}$$

3. Рассматриваем $\psi(X)$ как вектор нарушений $[\psi_1(X), \psi_2(X), \dots, \psi_m(X)]$, тем самым формируя многоцелевую задачу безусловной оптимизации.

Был выбран 1-й вариант, где $\psi(X)$ рассчитывается следующим образом:

- Если вектор принадлежит допустимой области, тогда $\psi(X)$ рассчитывается, как нормированное расстояние между X и X_0. Формула для расчета приведена далее:

  $\rho = \max\left(\frac{|X - X_0|}{\max(1, |X_i^- - X_i^+| - X_0^i)}\right)$, где i – номер параметра(измерения).

- Иначе вычисляем значение $\psi(X)$ по формуле:

$$\psi(X) = \max\{\psi_j(X), \qquad j \in \overline{[1, m]}\}$$

3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

Для исследования приведения задачи удовлетворения ограничениям к задаче безусловной оптимизации были написаны программные функции, которые отображают различные зависимости. Для удобного хранения результатов была предусмотрена система имен.

Для тестирования задач разных типов была написана функция для генерции задач оптимизации, который создает класс `Problem` из модуля GTOpt p7core, который наследуется от класса `gtopt.ProblemGeneric`. Класс формируется на основе настроек, которые хранятся в файлах формата JSON. В классе реализованы методы `prepare_problem` и `evaluate`.

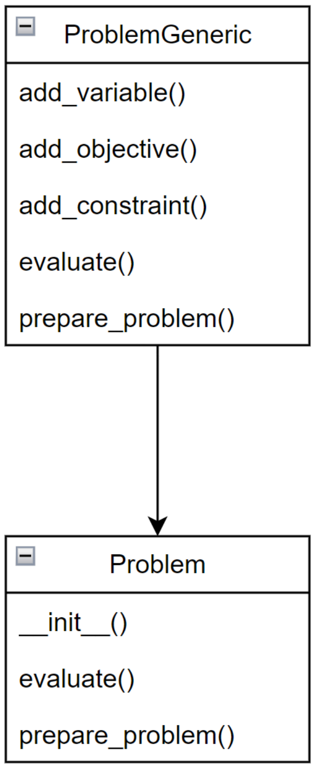

Рисунок 1. UML-диаграмма наследования.

Метод `prepare_problem` необходим для инициализации вектора параметров, целевых функций и функциональных ограничений. С помощью методов `add_variable` и `add_constraint` добавляются переменные и ограничения соответственно. Параметр `hints` позволяет задать тип переменных или ограничений.

Метод `evaluate` служит для вычисления значения функциональных ограничений или целевых функций в заданной точке. Так же в данном методе происходит запись времени в массив hist, в котором хранится время выполнения для каждой итерации, который необходим для вывода соответствующего графика.

Далее представлен листинг класса `Problem`:

```
class Problem(gtopt.ProblemGeneric):

    def __init__(self):
      self.time_history = [0]
      self.start_time = time.time()
      self.hist = []

    def prepare_problem(self):
      self.enable_history()
      for var in input_variables:
        if("@GT/VariableType" in var["hints"]):

          if(var["hints"]["@GT/VariableType"] == "Discrete" or var["hints"]["@GT/VariableType"] == "Categorical"):

            list_map = np.array([])
            if var["key"] == "Linspace":

              list_map = (np.linspace(var["bounds"][0], var["bounds"][1], var["bounds"][2]))
            elif var["key"] == "Random":
              list_map = np.random.uniform(var["bounds"][0], var["bounds"][1], var["bounds"][2])
            IG = list_map[var["initial_guess"]]

            self.add_variable(list_map, IG, hints=var["hints"])
          else:

            self.add_variable(var["bounds"], var["initial_guess"], hints=var["hints"])

        else:

          self.add_variable(var["bounds"], var["initial_guess"], hints=var["hints"])
      for constr in input_constraints:
        self.add_constraint(constr["bounds"], hints=constr["hints"])

      # self.add_objective()

    def evaluate(self, queryx, querymask):
      self.time_history = [*self.time_history, *[time.time() - self.start_time] * len(queryx)]

      functions_batch = []
      output_masks_batch = []
      for x, mask in zip(queryx, querymask):

        constraints = []
        for i in range(len(functions)):
          constraints.append(functions[i](x) if mask[i] else None)
        functions_batch.append(constraints)

        output_mask = mask
        output_masks_batch.append(output_mask)
        # print(functions_batch)
      return functions_batch, output_masks_batch
```

Решением поставленной задачи стала передача в оптимизатор ограничений под видом целевой функции и дальнейшая фильтрация результатов. Для этого был сформировал новый класс ConstraintsForNonObjectiveProblem, который наследуется от класса Constraints, где были переписаны методы __call__ и calc_batch для правильной работы в интерфейсе Nevergrad.

Листинг класса ConstraintsForNonObjectiveProblem приведен ниже:

```
class ConstraintsForNonObjectiveProblem(Constraints):
  def distance(self, x, problem):
    ig = problem.initial_guess()
    arr = [
      abs(x[i] - ig[i]) / max([1, ig[i] - problem.variables_bounds()[0][i], problem.variables_bounds()[1][i] - ig[i]])
      for i in range(self.problem.size_x())]
    dist = np.max(arr)
    return -1 + dist
  def constraints_to_objectives(self, x):
    problem = self.problem
    bounds_c = problem.constraints_bounds()
    n = len(x)
    m = len(x)
    x = np.array([x])
    mask_list = np.ones((1, m), dtype=np.int16)
    c, mask = problem.evaluate(x, mask_list)
    psi, flags = problem._evaluate_psi(np.array(c), np.array(bounds_c), 1e-5)
    res = psi[:, -1]
    res = np.where(res <= 0, self.distance(x[0], problem), np.abs(res))
    return res
  def __call__(self, *args):
    values = self._evaluate(input=args)
    res = self._calc_batch_violations(np.atleast_2d(values))[0]
    res = np.where(res <= 0, np.abs(res) * self.distance(args, self.problem), np.abs(res))
    return self.constraints_to_objectives(args)
  def calc_batch(self, *args):
    inputs = np.vstack(args).T
    batch_values = self._evaluate_batch(inputs=np.vstack(args).T)
    # TODO: Добавить проверку на наличие начального приближения.
    res = self._calc_batch_violations(batch_values)
    res = np.where(res <= 0, np.abs(res) * self.distance(inputs, self.problem), np.abs(res))
    return self.constraints_to_objectives(inputs[0])
```

4. ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

Эксперимент №1

В первом эксперименте с помощью генератора задач и входного файла с параметрами формируется задача CSP с целочисленным вектором входных параметров размерностью 2. Ограничения были подобраны таким образом, чтобы сформировать разрешенную область между двумя концентрическими областями. Для задания ограничений используется готовая функция Sargan, которая удовлетворяет вышеупомянутым условиям. Начальное приближение было задано в точке (1, -11). Далее представлено содержимое файла параметров, на основе которого формируется задача.

```
{"key": "test_1",
  "variables":
  [
    {"bounds": [-15, 15], "initial_guess": 1, "hints": {"@GT/VariableType": "Integer"}},
    {"bounds": [-15, 15], "initial_guess": -11, "hints": {"@GT/VariableType": "Integer"}}
  ],
  "constraints":
  [
    {"bounds": [10, 50], "function": "sargan", "hints": {}},
    {"bounds": [-100, 20], "function": "sargan", "hints": {}}
  ]
}
```

Решение, полученное с помощью Nevergrad, имеет лучший результат оптимизации по сравнению с результатом решения GTOpt, так как решение располагается ближе к начальному приближению(на 0.06 решение Nevergrad ближе к наччальному приближению, чем решение GTOpt), что видно на графиках (рис. 1 и рис.2). Можно заметить так же и то, что Nevergrad требует больше итераций для нахождения решения.

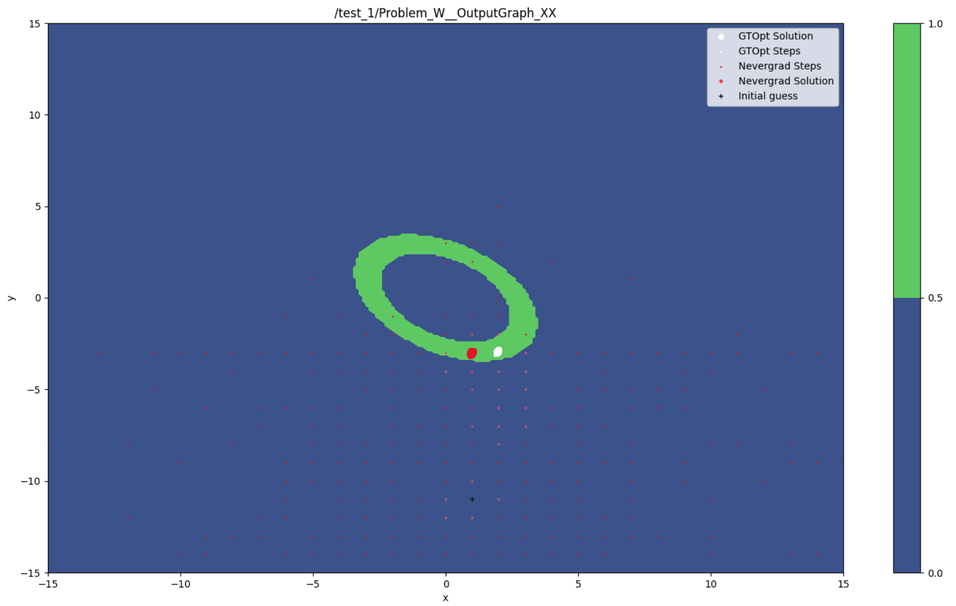

Рисунок 2. График для теста 1 в осях x1 и x2 с выводом допустимой и недопустимой области.

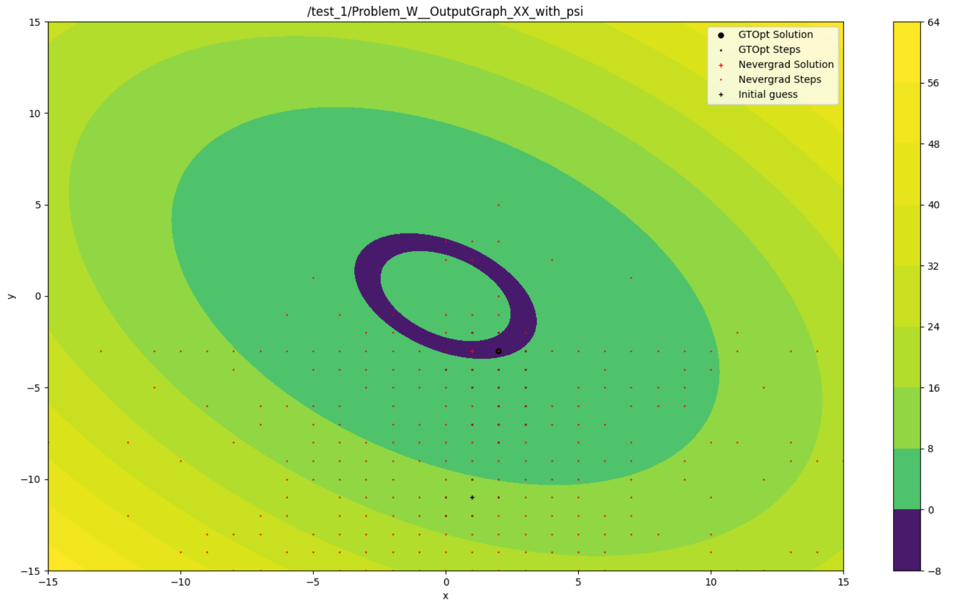

Рисунок 3. График для теста 1 в осях x1 и x2 с линиями уровня функциональных ограничений.

Эксперимент №2

Во втором эксперименте с помощью генератора задач и входного файла с параметрами формируется задача условной оптимизации с вещественным вектором входных параметров размерностью 2. Ограничения были подобраны таким образом, чтобы сформировать разрешенную область между двумя концентрическими областями. Для задания ограничений используется готовая функция Sargan, которая удовлетворяет вышеупомянутым условиям. Начальное приближение было выбрано в точке (1, -11). Далее представлено содержимое файла параметров, на основе которого формируется задача.

```
{"key": "test_2",
  "variables":
  [
    {"bounds": [-15, 15], "initial_guess": 1, "hints": {}},
    {"bounds": [-15, 15], "initial_guess": -11, "hints": {}}
  ],
  "constraints":
  [
    {"bounds": [10, 50], "function": "sargan", "hints": {}},
    {"bounds": [-100, 20], "function": "sargan", "hints": {}}
  ]
}

```

Результаты, полученные с помощью Nevergrad и GTOpt, сопоставимы между собой(отклонение меньше 0.0001), что видно из графиков (рис. 3 и рис.4). GTOpt так же использует меньше итераций для нахождения решения.

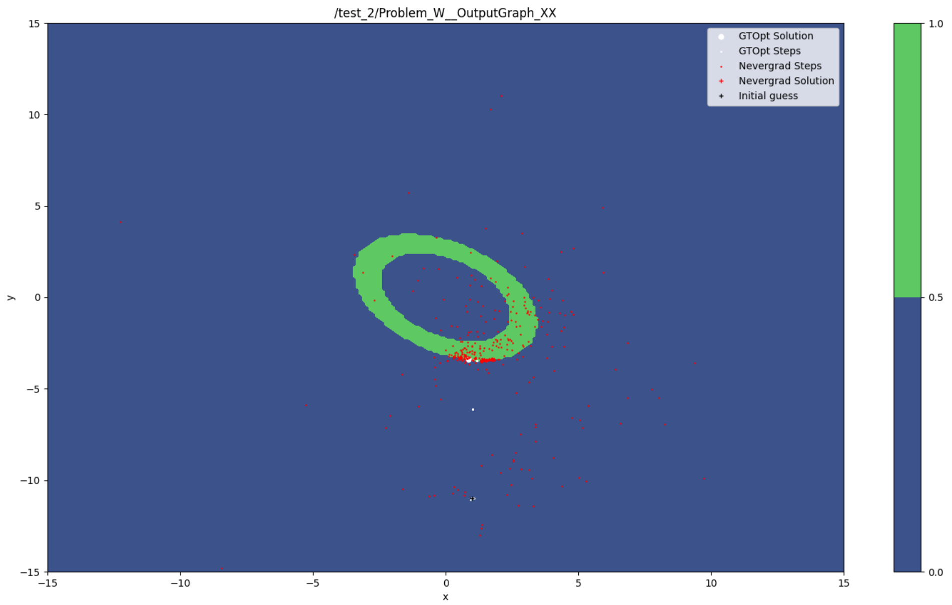

Рисунок 4. График для теста 2 в осях x1 и x2 с выводом допустимой и недопустимой области.

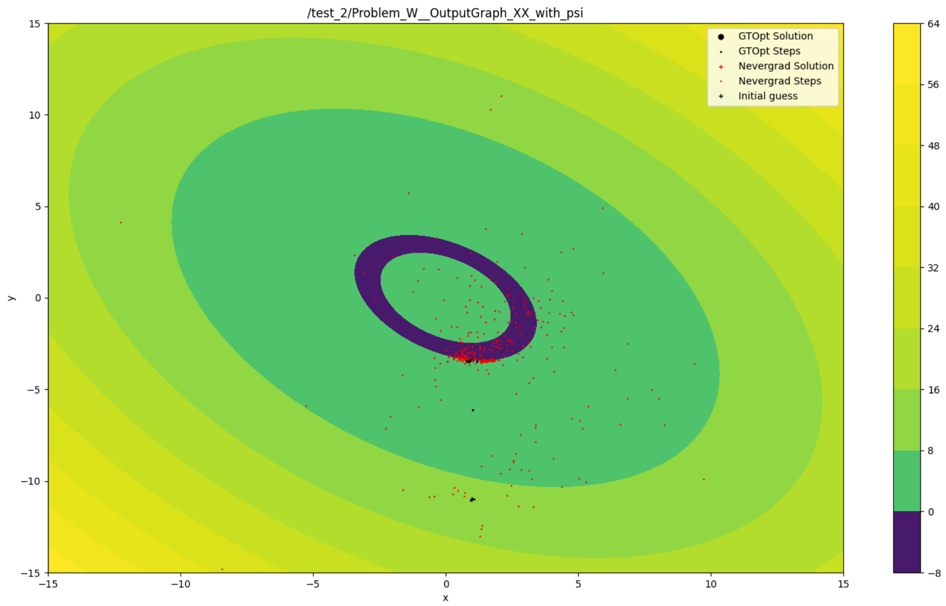

Рисунок 5. График для теста 2 в осях x1 и x2 с линиями уровня функциональных ограничений.

ЗАКЛЮЧЕНИЕ

В рамках прохождения практики были исследованы способы приведения задачи CSP к задаче безусловной оптимизации. Была написана программная реализация для поставленной задачи и проверена ее работоспособность для библиотеки Nevergrad. Также были проведены эксперименты для сравнения методов GTOpt и Nevergrad.

СПИСОК ЛИТЕРАТУРЫ

1. Документация библиотеки p7core [Электронный ресурс]. - URL: https://datadvance.ru/product/pseven-core/manual/6.47/index.html;
2. Документация Nevergrad [Электронный ресурс]. - URL: https://facebookresearch.github.io/nevergrad/index.html;
3. Сайт с набором функций разной размерности [Электронный ресурс]. - URL: http://infinity77.net/global_optimization/index.html.