Задание 5.
Написание алгоритма решения задачи оптимизации надёжности (10 баллов).

Формулировка задания

Дано:

Дана программная система RelOpt для проведения экспериментов по оптимизации и статья с описанием алгоритма оптимизации надёжности.

Требуется:
1. **(7 баллов)** Разобраться в устройстве программной системы RelOpt. Реализовать алгоритм, описание которого приведено в статье на языке Python. Интегрировать реализацию в программную систему RelOpt.

2. **(1 балл)** Повторить экспериментальное исследование, проведённое в статье. Описать полученные экспериментальные результаты, сравнить их с результатами, приведёнными в статье. Обосновать их расхождение, если таковое имеется.

3. **(2 балла)** Написать эссе на тему "Как я провёл Практикум в 7м семестре". В этом эссе в свободной форме необходимо отразить примерное время, потраченное на выполнение Практикума, трудности с которыми Вы столкнулись при выполнении заданий, новые знания и технологии которым Вы научились при выполнении Практикума, критику и предложения по формулировкам и типам заданий.

Бонусное задание:
4. **(3 балла)** Провести полноценное экспериментальное исследование алгоритма при помощи метода статистических гипотез (*см. лекции по Имитационному Моделированию*). Обосновать количество запусков алгоритма и классы используемых в исследовании исходных данных. Описать полученные экспериментальные результаты, сравнить их с результатами, полученные в п.3. Обосновать их расхождение, если таковое имеется.

На выполнение задания отводится две недели. Последний срок сдачи задания – **25 декабря**.

Требования к файлам решения:

Сданная задача присылается на volkanov@lvk.cs.msu.su письмом с темой Reliability-Task5-Solution. Решение задачи должно включать в себя следующие файлы:

1. Статью с реализуемым алгоритмом year_group_surname_algorithm_<исходное_имя_файла>.pdf
   где:
   - year – год выполнения задания;
   - group – номер Вашей группы;
   - surname – Ваша фамилия;
   - algorithm – сокращённое имя алгоритма;
   - <исходное_имя_файла> – имя файла статьи, которое Вы получили при выдаче задания.

2. Файл с кодом алгоритма year_group_surname_algorithm.py
3. Файлы с описанием конфигураций систем, на которых проводились исследования, настроек алгоритмов (если таковые будут) и файлы со статистикой (они автоматически создаются средством). Имена всех этих файлов должны иметь префикс year_group_surname_algorithm

4. Файлы year_group_surname_algorithm_exps.pdf и исходный файл описания экспериментального исследования в произвольном формате с именем year_group_surname_algorithm_exps, а его расширение будет определяться типом.

5. Файл year_group_surname_esse.txt с текстом эссе “Как я провёл Практикум в 7м семестре”.

6. *Если Вы выполнили бонусное задание, то высылаете файлы аналогичные п.3 и п.4. Все эти файлы должны иметь суффикс _bonus в задании. Например, для п. 4 файл описания экспериментального исследования должны иметь имя year_group_surname_algorithm_exps_bonus.pdf, а исходный файл описания экспериментального исследования в произвольном формате должен иметь имя year_group_surname_algorithm_exps_bonus, а расширение будет определяться его типом.*

Присланная до **23:59 (московское время) 25 декабря** задача принимается (БЕЗ ШТРАФА), далее –5 балла за каждую неделю просрочки.

Методические указания по выполнению задания:

1. Архитектура программного средства RelOpt

Программное средство RelOpt для решения задачи оптимизации надёжности (ЗОН) написано на языке *Python*. В качестве примера алгоритмов приведена реализация нескольких алгоритмов. А именно эволюционного алгоритма (ЭА), адаптивного генетического эволюционного алгоритма (АГЭА) и жадного алгоритма (ЖА).

Разработанное программное средство включает следующие модули:

- модуль, содержащий классы, используемые всеми алгоритмами (представления отдельного решения, структуры РВС РВ, базовый класс алгоритма и др.);
- реализация классического эволюционного алгоритма (ЭА) и адаптивного генетического эволюционного алгоритма (АГЭА);
- реализация жадного алгоритма (ЖА);
- модуль, содержащий метамодели;
- графический пользовательский интерфейс (ГПИ).

Диаграмма классов, представляющих логику работы программного средства, приведена на рисунке 1

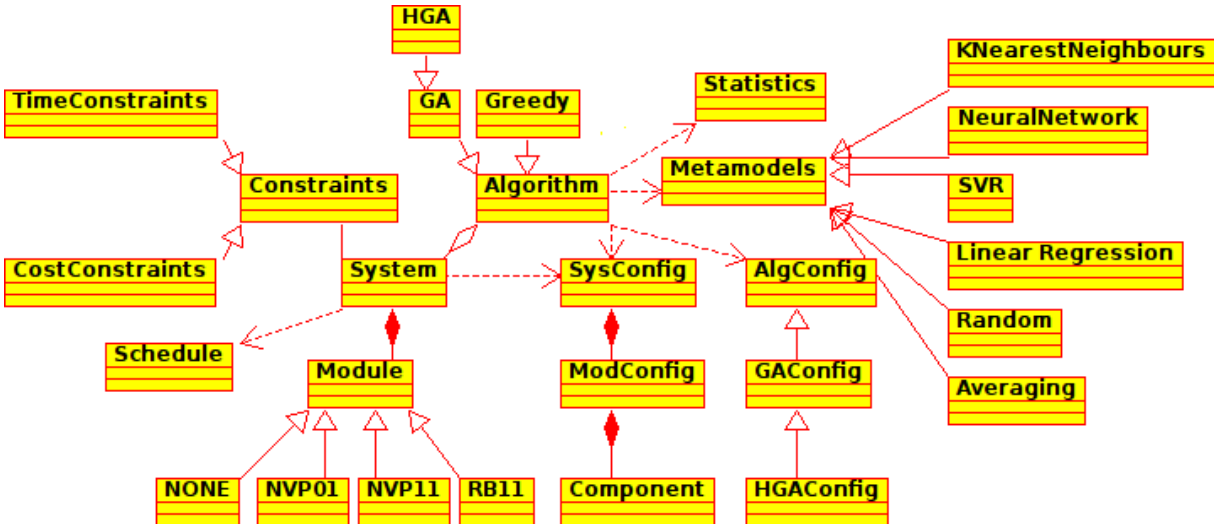

Рисунок 1. *Диаграмма классов, представляющих логику работы программного средства.*

Кратко опишем классы, представленные на диаграмме классов:

- *Algorithm, GA, HGA, Greedy* - базовый класс для оптимизационных алгоритмов и реализации ЭА, АГЭА, ЖА соответственно;
- *Module, NONE, NVP01, NVP11, RB11* - базовый класс, представляющий модуль РВС РВ, и реализации, соответствующие модулям с механизмами обеспечения отказоустойчивости (МОО) *None*, *NVP/0/1*, *NVP/1/1* и *RB/1/1* соответственно. Описание данных МОО дано в задании 3;
- *System* – представление вычислительной системы (ВС);
- *Component, ModConfig, SysConfig* - представление версии программного или аппаратного компонента, параметров конфигурации модуля и параметров конфигурации ВС;
- *Constraints, CostConstraints, TimeConstraints* - базовый класс, представляющий ограничения, и реализации ограничений на стоимость и времена окончания выполнения программных компонентов соответственно;
- *AlgConfig, GAConfig, HGAConfig* - базовый класс для настроек алгоритмов и реализации настроек классического эволюционного алгоритма и АГЭА;
- *Metamodel, Random, Averaging, LinearRegression, KNearestNeighbours, NeuralNetwork, SVR* - базовый класс метамодели и реализации случайной модели, линейной регрессии, метода k ближайших соседей, нейросетевой модели и регрессии методом опорных векторов соответственно.
- *Statistics* - класс для хранения, обработки и вывода в текстовый файл формата CSV статистики запусков алгоритма;
- *Schedule* - класс для преобразования конфигурации РВС РВ к расписанию общего вида.

2. Графический Пользовательский Иинтерфейс RelOpt

Для реализации Графического Пользовательского Интерфейса (ГПИ) используется PyQt4 - набор привязок библиотеки Qt4 для языка Python. ГПИ представляет собой несколько взаимосвязанных окон.

Главное окно программы (рисунок 2) представляет возможность задания всех параметров, необходимых для запуска алгоритма решения задачи ЗОН.

Окно генерации описания ВС (рисунок 3) представляет возможность задания параметров ВС и генерации её описания.

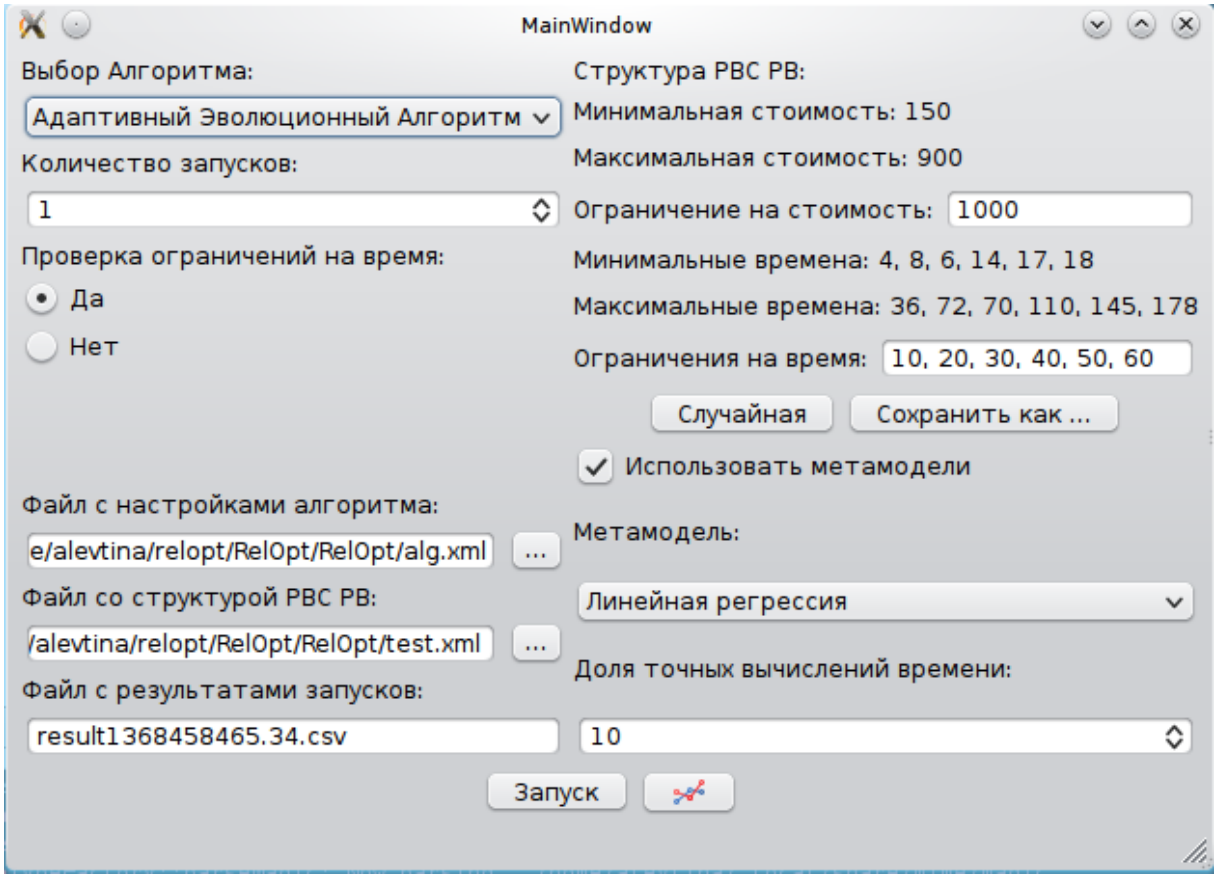

Рисунок 2. *Главное окно программы.*

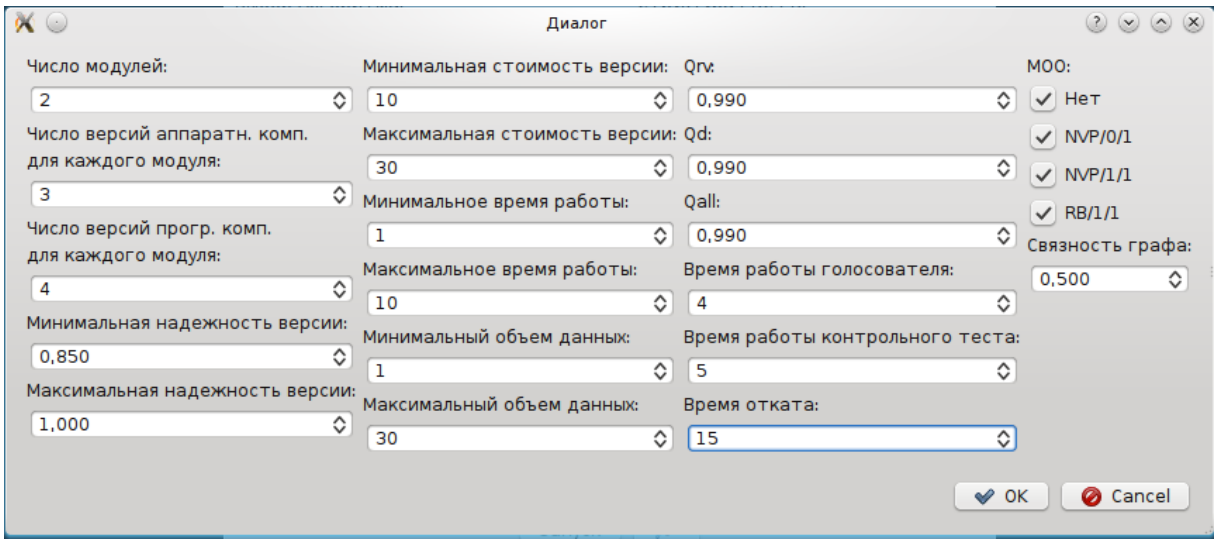

Рисунок 3. *Окно генерации описания ВС.*

3. Описание входных данных средства RelOpt

На вход средству RelOpt подаются XML-файлы с настройками разработанного алгоритма (если требуется) и описанием ВС.

Например, XML-файл с настройками АГЭА должен иметь следующую структуру:

- *<alg>* - корневой элемент в описании настроек АГЭА; имеет атрибуты *maxiter* - максимальное количество итераций алгоритма без изменения текущего лучшего решения, *maxgeniter* - максимальное количество попыток получить решение, удовлетворяющее ограничениям, при генерации начальной популяции, *popsize* - размер популяции; содержит тег *<par>*;
- <par> - элемент, соответствующий параметру АГЭА; имеет атрибут *name* - имя

параметра (*crosspercent*, *crossprob*, *mutpercent* или *mutprobe*), *min*, *norm*, *max* - минимальное, среднее и максимальное значения параметра.

XML-файл с описанием ВС должен иметь следующую структуру:

- <*system*> - корневой элемент в описании ВС; имеет атрибут *limitcost* - максимально допустимая стоимость системы; содержит теги <*module*> и <*link*>;
- <*module*> - описание модуля ВС; имеет атрибуты *num* - номер, *limittime* - директивный срок завершения работы программного компонента, *trecov* - время отката к контрольной точке, *ttest* - время работы контрольного теста, *tvote* - время работы голосователя, *qall*, *qd* и *qrv* соответствуют параметрам надёжности МОО; содержит теги <*hw*>, <*sw*>, <*tool*> и <*time*>;
- <*hw*>, <*sw*> - описание доступных аппаратных и программных компонентов; имеют атрибуты *num* - номер компонента, *cost* - стоимость, *rel* - надежность;
- <*tool*> - описание доступного МОО; имеет атрибут *name* - имя МОО (*none*, *nvp01*, *nvp11* или *rb11*);
- <*time*> - время работы для пары (аппаратный компонент, программный компонент); имеет атрибуты *hwnum* - номер аппаратного компонента, *swnum* - номер программного компонента, *t* - значение времени;
- <*link*> - описание зависимостей по данным между модулями; имеет атрибуты *src* - номер модуля, передающего данные, *dst* - номер модуля, получающего данные, *vol* - объём передаваемых данных.

4. Запуск и рекомендуемые настройки RelOpt

Для запуска средства достаточно просто воспользоваться командой *python RelOpt.py* . Необходимо, чтобы на компьютере, где установлена система RelOpt стоял **python** не ниже версии **2.7** и библиотека **PyQt4**. В качестве IDE под Windows рекомендуется использовать Visual Studio с Python Tools for Visual Studio (http://pytools.codeplex.com/)

Поскольку, в статьях с алгоритмами отсутствуют ограничения по времени, то не требуется оценивать эти ограничения и использовать метамодели.

Поэтому ГПИ настраиваем следующим образом:

- ставим “Нет” в меню “Проверка ограничений на время”;
- не ставим галочку в пункте “Использовать метамодели”.

5. Советы по интеграции реализации алгоритма с существующими файлами RelOpt

При интеграции реализации алгоритма из статьи существующие файлы RelOpt **менять не допускается.** Если такая необходимость возникает (ошибка в коде, не хватает чего-то для алгоритма), то по этим вопросам следует написать письмо по адресам volkanov@lvk.cs.msu.su и alevtina@lvk.cs.msu.su с темой письма содержащей, фразу *[RelOpt]*

При написании алгоритма следует придерживаться следующих правил:

- класс с реализацией алгоритма должен быть наследником класса *Algorithm*;
- класс с настройками алгоритма (если в нём будет необходимость) должен быть наследником наследником класса *AlgConfig*;
- для описания системы нужно использовать существующий класс *System*.
- реализация алгоритма должна быть прокомментирована в должной мере, достаточной для понимания работы алгоритма;
- код реализации алгоритма должен быть хорошо читаем (для этого используйте отступы и “говорящие имена” переменных).