**ВВЕДЕНИЕ**

Современные техника, наука, экономика, финансы существенно используют экстремальные свойства процессов и систем. Поэтому достижения в теории оптимизации – в математическом программировании, теории управления – находят многие важные области применения. Специалист должен уметь составлять математические модели практических экстремальных задач, проводить их теоретический анализ, разрабатывать самостоятельно или использовать известные методы решения, реализовать эти методы на электронно-вычислительных машинах и делать выводы по изучаемой задаче.

Оптимальные решения позволяют достичь поставленной цели с минимальными материальными и трудовыми затратами. Поэтому в настоящее время большое внимание уделяется анализу и методам принятия оптимальных решений. Чтобы процент риска при принятии решения был максимально близок к нулю, нужно подходить к исследованию со всех возможных сторон. Необходимо создать модель для одновременной оптимизации решений по нескольким критериям, что усложняет задачу оптимизации, особенно если она проводится вручную: количество альтернатив может достигать очень большого значения. Следовательно, возникает необходимость в оптимизации процесса принятия решений, что может сузить количество предлагаемых альтернатив и облегчить задачу выбора оптимальной альтернативы.

Оптимизация большого числа альтернатив по многим критериям – это достаточно трудоёмкий процесс, требующий достаточных вычислительных мощностей. Так как компьютеры тесно вошли в нашу жизнь, то логично предположить, что существуют алгоритмы, которые выполняют оптимизации различного рода автоматически на основании каких-либо экспертных оценок и, самое главное, делают это намного быстрее, чем если бы это делали люди без использования компьютеров [1].

Целью курсового проекта является написание приложения, которое позволяет упростить выбор тепловых завес для установки в механическом цеху среди множества альтернативных вариантов. В результате выполнения курсового проекта должны быть реализованы однокритериальная и многокритериальная оптимизации, определение наилучшей альтернативы производственной линии с учетом ограничений на стоимость, электрическую мощность, часовой расход воздуха, необходимость водяного пара, а также годовое время работы тепловых завес.

# **1 ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ОДНОКРИТЕРИАЛЬНОЙ И МНОГОКРИТЕРИАЛЬНОЙ ОПТИМИЗАЦИИ**

## **1.1 Основные понятия оптимизации проектных решений**

Термин «оптимизация» относится к процессу выбора наилучшего варианта из всех возможных. С точки зрения инженерных расчётов методы оптимизации позволяют выбрать лучший вариант конструкции, лучшее распределение ресурсов.

Оптимизация осуществляется с помощью алгоритмов математического программирования и может быть структурной, параметрической и структурно-параметрической. В процессе оптимизации конструкции оптимизируется структура объекта, в процессе параметрической оптимизации оптимизируются параметры элементов, составляющих структуру. Эти задачи решаются с использованием алгоритмов дискретного, непрерывного и дискретно-непрерывного математического программирования соответственно.

В зависимости от количества критериев, по которым оптимизируется объект, можно различать однокритериальную и многокритериальную оптимизацию.

По наличию ограничений в целевой функции и операционных параметрах различают оптимизацию без ограничений и при наличии ограничений. Таким образом, если при синтезе усилителя необходимо, чтобы коэффициент усиления был не меньше определенного значения, то говорят о наложении ограничения на соответствующий критерий. Если в этом случае необходимо использовать названия элементов, значения которых должны находиться в определенном диапазоне, то мы имеем дело с ограничениями на рабочие параметры.

В случае параметрического синтеза с известной структурой объекта параметры элементов выбираются таким образом, чтобы минимизировать (максимизировать) целевую функцию.

Следует отметить, что существующие алгоритмы оптимизации обычно не гарантируют, что глобальный оптимум будет найден, но это не критично. Например, чтобы увеличить вероятность нахождения глобального оптимума, можно значительно увеличить количество итераций, использовать несколько алгоритмов и многократно выполнять соответствующие алгоритмы [2]. Современные передовые системы автоматизированного проектирования включают в себя модули для параметрического синтеза и оптимизации, которые, в свою очередь, очень помогают человеку в выборе решений.

Если помимо подбора параметров необходимо еще и определить структуру объекта, то мы будем уже иметь дело со структурно-параметрическим синтезом, который решается при помощи алгоритмов дискретно-непрерывного математического программирования. Если задача параметрической оптимизации сейчас решается практически для любых объектов, то развитие структурно-параметрической оптимизация сейчас находится лишь на начальной стадии развития. С теорией оптимизации тесно связаны математическое программирование, теория исследования операций, теория принятия решений, динамическое программирование. Дальнейшее развитие теории и практики оптимизации является очень важным для развития науки и техники.

Выбор оптимального решения или сравнение двух альтернативных решений осуществляется с использованием некоторой зависимой величины (функции), которая определяется проектными параметрами. Это значение называется целевой функцией или критерием качества. В процессе решения задачи оптимизации необходимо найти такие значения проектных параметров, при которых целевая функция имеет максимум (или минимум). Таким образом, целевая функция является глобальным критерием оптимальности математических моделей, с помощью которых описываются инженерные задачи.

В случае конструктивного параметра, это функция переменной, а его график представляет собой определенную кривую на плоскости. Для двоих целевая функция – это функция двух переменных, а её график – это поверхность.

Следует отметить, что целевую функцию не всегда можно представить в виде формулы. Иногда он может принимать только отдельные значения, указываться в виде таблицы и т.д. Однако в каждом случае это должна быть уникальная функция параметров проекта.

Различают основные этапы решения задачи оптимизации:

– постановка задачи. На этом этапе аналитик должен превратить слова клиента в четко сформулированную задачу;

– построить математическую модель задачи. Здесь чётко поставленная и сформулированная жизненно важная проблема формализуется математически;

– решение математической модели задачи;

– принимая решения. На этом этапе аналитик, основываясь на предыдущих шагах, должен принять лучшее решение. Принятие решений – это разновидность управленческой задачи.

**1.2 Классификации методов оптимизации**

Классификация – это общенаучный метод систематизации знаний, направленный на упорядочение определенного набора изучаемых объектов из разных областей реальности, знаний и деятельности в систему подгрупп (классов), в которых эти объекты расположены на основе их сходства по определенным существенным свойствам.

Методы оптимизации классифицируют в соответствии с задачами оптимизации.

Локальные методы сходятся в некотором локальном экстремуме целевой функции. В случае унимодальной целевой функции этот экстремум уникален и будет глобальным максимумом или минимумом [3]. Глобальные методы имеют дело с многоэкстремальными целевыми функциями. В глобальном поиске основная задача – выявить тенденции глобального поведения целевой функции.

По критерию размерности допустимого множества, методы оптимизации делят на методы одномерной оптимизации и методы многомерной оптимизации. По виду целевой функции и допустимого множества, задачи оптимизации и методы их решения можно разделить на следующие классы:

– задачи оптимизации, в которых целевая функция и ограничения являются линейными функциями, разрешаются так называемыми методами линейного программирования;

– задачи нелинейного программирования. В свою очередь из них выделяют две частные задачи. Если выпуклые функции, то такую задачу называют задачей выпуклого программирования. В противном случае имеем дело с задачей целочисленного (дискретного) программирования.

Практически все методы минимизации функции n переменных основаны на многократном повторении следующих двух действий:

– выбор в области параметров некоторого направления спуска;

– спуск к минимуму вдоль выбранного направления.

Если единичный вектор выбранного направления в точке, то уравнение прямой, проходящей через эту точку в направлении, записывается в виде, где параметр *z*, соответствующий точкам на прямой (модуль *z* есть расстояние от текущей точки).

Значения функции вдоль этой прямой можно описать функцией одной переменной. Изменяя *z* производится движение вдоль этой прямой, находим точку, в которой функция имеет меньшее значение, чем в точке. Обычно производится нахождение минимума функции одной переменной.

Все многообразие методов минимизации функции n переменных определяется множеством способов выбора направлений и методов спуска в выбранном направлении.

Классификация методов многомерной оптимизации [4]:

– методы нулевого порядка, где при выборе направления спуска требуют вычисления только значений функции;

– методы первого порядка, где требуются вычисления (точного или приближенного) градиента функции;

– методы второго порядка, где требуются вычисления как градиента, так и матрицы вторых производных.

Также различают методы однокритериальной оптимизации (поиск оптимума единственной целевой функции) и многокритериальной оптимизации (принятия решения при многих критериях, в частности сводят векторную задачу к последовательности скалярных задач).

Рациональные методы ориентированы на некоторые математические модели оптимизируемой функции. Эвристические алгоритмы обычно не используют какую-либо модель целевой функции, а основывают процесс оптимизации на формализованной человеческой интуиции и других расплывчатых предположениях.

Методы оптимизации также подразделяются на детерминированные и стохастические. Стохастические алгоритмы используют случайные элементы при выборе направления или длины шага в процессе оптимизации. Обратите внимание, что методы стохастической оптимизации применяются к детерминированным задачам (случайность намеренно вводится в алгоритм, чтобы обеспечить достижение цели).

Стохастические алгоритмы, в свою очередь, подразделяются на (прямые) алгоритмы случайного поиска и алгоритмы аппроксимации стохастического градиента.

**1.3 Однокритериальная оптимизация**

Цель оптимизации по одному критерию – найти лучшее решение для определенного критерия или показателя, такого как время работы, мощность и сочетание этих показателей с показателями энергопотребления или потерь мощности. Можно объединить несколько критериев в одну оптимизацию критерия, определив функцию стоимости одного критерия как взвешенную сумму нормализованных затрат, связанных с каждой метрикой.

Задачи одномерной минимизации представляют собой простейшую математическую модель оптимизации, в которой целевая функция зависит от одной переменной, а допустимым множеством является отрезок вещественной оси.

Общая постановка задачи одномерной оптимизации сводится к следующему: имеется некоторая функция *f(x)* от одной переменной *x*, необходимо найти такое значение *x\**, при котором функция *f(x)* принимает экстремальное значение. Под этим значением понимают минимальное или максимальное значения.

В общем, функция может иметь одну или несколько крайних точек. Поиск таких точек с определенной точностью обычно делится на два этапа. На первом этапе отделяются крайние точки. Это означает, что определяются участки, содержащие одну крайнюю точку, а затем уточняются до требуемой точности *ε*.. Максимизация целевой функции эквивалентна минимизации противоположного значения; поэтому могут рассматриваться только задачи минимизации.

Для решения задачи минимизации функции *f(x)* на отрезке *[a, b]*, в большинстве случаев, используют приближенные методы. С их помощь можно найти решения этой задачи с необходимой точностью в результате нахождения конечного числа значений функции *f(x)* и ее производных в некоторых точках отрезка *[a, b]*. Методы, использующие только значения функции и не требующие вычисления ее производных, называются прямыми методами минимизации. Достоинством прямых методов является то, что от целевой функции не требуется дифференцируемости и, более того, она может быть не задана в аналитическом виде. Единственное, на чем основаны алгоритмы методов прямой минимизации – это возможность определять значения *f(x)* в определенных точках. Среди задач математического программирования наиболее распространенными и изученными являются так называемые задачи линейного программирования (линейная оптимизация). Давайте рассмотрим этот тип задач. Они характеризуются тем, что целевая функция линейно зависит от параметров, а также тем, что ограничения, накладываемые на независимые переменные, принимают форму линейного равенства или неравенства по этим переменным. Эти задачи нередко используются на практике – например, при решении проблем, связанных с распределением ресурсов, планированием производства, организацией работы транспорта и т.д.

Общая задача линейной оптимизации заключается в нахождении максимума (минимума) линейной целевой функции. Вид целевой функции и ограничений описываются формулами (1.1-1.4).

|  |  |  |
| --- | --- | --- |
|  |  | (1.1) |
|  |  |  |

при ограничениях:

|  |  |  |
| --- | --- | --- |
|  |  | (1.2) |
|  |  | (1.3) |
|  |  | (1.4) |

где −целевая функция, критерий оптимальности или линейная форма;

*x* − вектор неизвестных;

 − коэффициенты целевой функции;

 − коэффициенты ограничений;

 − величины правых частей ограничений.

Вектор значений неизвестных *x = (x*1*, x*2*, …, xn)*, удовлетворяющих условию задачи, называется допустимым решением или допустимым планом задачи линейной оптимизации. Совокупность всех допустимых планов называется множеством допустимых планов. Допустимое решение *x\* = (x*1\**, x*2\**, …, xn\*)* называется оптимальным, если оно обеспечивает максимальное (или, в зависимости от условий задачи, − минимальное) значение целевой функции.

Решение задач линейной оптимизации может быть получено без особых трудностей (разумеется, при правильной постановке задачи). Классическим методом решения таких задач является симплексный метод. Если используются только две переменные, можно также успешно использовать графический метод решения, который имеет преимущество ясности. Очевидно, что, если *n* больше двух, графический метод не может быть применен.

При решении ряда оптимизационных задач требуется, чтобы значения неизвестных *x = (x*1*, x*2*, …, xn)* выражались в целых числах. К задачам подобного типа относятся те, в которых требуется определить необходимые для принятия решений значения физически цельных объектов (машин, агрегатов различного типа, людей, транспортных единиц и т.д. и т.п.).

Такие задачи называются задачами целочисленной оптимизации. Математическая модель задачи линейной целочисленной оптимизации также определяется формулами, но в этом случае существует дополнительное требование, чтобы все (или некоторые) неизвестные были целочисленными. Если требование целого числа применяется только к некоторым из неизвестных значений проблемы, то проблема частично называется целым числом.

## **1.4 Многокритериальная оптимизация**

Многокритериальная оптимизация или программирование – это процесс одновременной оптимизации двух или более конфликтующих целевых функций в заданной области определения [5].

Любая проблема практического решения имеет несколько желательных альтернатив, но традиционные подходы к оптимизации отдельных критериев ограничиваются одним решением. Также может быть ряд ограничений, но во всех случаях структура модели требует единого заранее определенного набора взаимных уступок. Оптимизация по нескольким критериям предлагает множество различных подходов и методов, которые можно использовать для поиска подходящих альтернатив наиболее эффективным и высококачественным образом.

Многокритериальная оптимизация используется во многих областях науки, включая инженерию, экономику и логистику, где оптимальные решения должны приниматься, когда существует компромисс между двумя целями. В таких задачах нет общего понятия оптимальности, поскольку существуют разные объективные критерии, которые могут противоречить друг другу. На практике часто не существует единого критерия оптимальности для измерения качества решения. Скорее, последнее определяется ценностью решения по каждому объективному критерию. Фактически, практикующий специалист ищет решение, имеющее приемлемую ценность, и на практике могут быть разные допуски к качеству решения, предусмотренного для разных целевых функций: для некоторых объективных критериев решения, далекие от оптимальных, могут быть приемлемыми. Традиционный подход оптимальности направлен на создание всех возможных решений без доминирования над всеми критериями оптимальности. Часто это занимает неприемлемое количество времени. Кроме того, не очевидно, как выбрать подходящее решение из оптимального набора возможных решений, который может быть очень большим. Здесь мы предлагаем новый подход и называем его настройкой многопороговой оптимизации, которая учитывает различные требования для разных объективных критериев и, следовательно, является более гибкой и часто может быть решена более эффективно.

Задача многокритериальной оптимизации – это задача оптимизации, которая включает в себя несколько целевых функций [6]. С математической точки зрения задача многокритериальной оптимизации может быть сформулирована следующей функцией (формула 1.5):

|  |  |  |
| --- | --- | --- |
|  |  | (1.5) |

Относительная важность целей обычно неизвестна до тех пор, пока не будут определены все основные свойства системы и полностью не поняты все возможные взаимосвязи.

По мере увеличения числа возможных целей становится ясно, что эти отношения образуют сложную структуру, и их становится труднее идентифицировать. В этом случае многое зависит от интуиции исследователя и его способности точно выразить определенные предпочтения в процессе оптимизации.

Таким образом, стратегия построения многокритериальной оптимизации состоит, прежде всего, в умении адекватно определить постановку задачи так, чтобы эта задача допускала свое решение, а также выразить необходимые предпочтения в виде числовых зависимостей, при сохранении реальности поставленной задачи.

Задача многокритериальной оптимизации формулируется следующим образом:

|  |  |  |
| --- | --- | --- |
|  |  | (1.6) |
|  |  |  |

где  – это *n* целевых функций.

Векторы решений приведены в формуле (1.7) и относятся к не пустой области определения *S*:

|  |  |  |
| --- | --- | --- |
|  |  | (1.7) |

Задача многокритериальной оптимизации – найти вектор целевых переменных, удовлетворяющий наложенным ограничениям, и оптимизировать вектор-функцию, элементы которой соответствуют целевым функциям. Эти функции образуют математическое описание удовлетворительного критерия и обычно противоречат друг другу. Таким образом, «оптимизация» означает поиск решения, при котором значение целевых функций для проблемы было бы приемлемым.

Итальянский экономист В. Парето сформулировал один из самых распространенных экономических критериев оптимальности. Он формулируется очень просто: «Следует считать, что любое изменение, которое никому не причиняет убытков и которое приносит некоторым пользу, является улучшением».

Критерий Парето имеет очень широкий экономический смысл и очень часто используется для решения сложных экономических задач. Его можно использовать, когда оптимизация одного набора показателей, характеризующих объект, не должна ухудшать другой набор не менее важных показателей. Критерий Парето не применяется к ситуациям, когда экономический эффект одних связан с потерями других. Область оптимальных решений Парето называется областью компромисса.

Алгоритм выделения области Парето проводится в несколько следующих шагов:

1) выбирается проект *Пi*, полагая *i* = 1;

2) проект *Пi* сравнивается с остальными по всем показателям и отмечаются те из них, которые строго хуже, чем *Пi*;

3) отмеченные проекты не могут принадлежать области Парето и из дальнейшего рассмотрения исключаются;

4) полагается *i* = *i* + 1;

5) если  и проект *Пi* уже был отмечен на предыдущих итерациях, то выполняется переход к четвёртому шагу;

6) если  и проект еще не помечен, то производится переход ко второму шагу;

7) если , где *n* – число сравниваемых проектов, то осуществляется переход к восьмому шагу;

8) оставшиеся не отмеченными проекты образуют множество эффективных решений (область Парето).

Если использовать дополнительную информацию, например, об относительной важности показателей, то можно получить дальнейшее уточнение мест проектов.

Часто, когда необходимо сделать выбор из альтернативы, возникает необходимость в измерениях с использованием специальных шкал. Такие шкалы используются в случаях, когда оценки субъективны. Эти шкалы называются психофизическими и позволяют формализовать систему предпочтений эксперта. Психофизические шкалы задаются функциями особого вида - функциями желательности и устанавливают соответствие между естественными значениями показателей в физических шкалах и психологическими параметрами - субъективными оценками «ценности» этих значений. К физическим показателям объекта могут относиться экономические, технические, технологические, экологические, социальные, эстетические и другие параметры.

Самым известным и широко используемым является метод Харрингтона. Введение шкалы желательности позволяет свести исходную многокритериальную задачу решения с критериями разных измерений к многокритериальной задаче с критериями, измеряемыми по той же шкале. Возможность широкого применения этого метода в задачах оценки и оптимизации подтверждается исследованиями ученых в различных областях.

Многообразие областей применения метода указывает на возможность его применения при оценке уровня техносферной безопасности на территории. Метод Харрингтона для описания частных критериев и ограничений при решении многокритериальных задач позволяет установить соответствие между лингвистическими оценками удобства значений оценочного показателя и числовыми диапазонами предпочтений. Интервалы предполагают значения, которые непрерывно увеличиваются от нуля до единицы, так как соответствующий параметр качества изменяется от наименьшего до наиболее желательного. Конкретный вид функций желательности задается лицом, принимающим решения, исходя из его субъективных представлений. Затем путем свертки частных функций желательности строится глобальный критерий качества процесса, максимизация которого составляет оптимум.

Метод Харрингтона обладает следующими преимуществами:

– является количественным;

– выражается одним числом, т.е. является единой функцией;

– является однозначным, т.е. заданному набору значений частных параметров оценки соответствует одно значение функции;

– универсальный и может использоваться в различных областях;

– всесторонне характеризует объект, т.е. соответствует требованию полноты;

– обеспечивает простой путь преобразования показателей с помощью одного графика для всех критериев;

– «нейтральна» при обобщении, на конечный результат при построении шкалы желательности влияют только личные предпочтения;

– является адекватным методом. Адекватность частичных и обобщенных функций желательности следует понимать, как их эквивалентность измеренным значениям параметров оптимизации.

Процесс оценки состоит из следующих этапов:

– определение перечня частных параметров оценки, т.е. критериев, по которым будет оцениваться объект (количественных, качественных);

– установление границы допустимых значений. Это удобнее, когда есть четкие ограничения. Однако из-за нового характера исследования или отсутствия данных допустимые значения для отдельных параметров могут быть недоступны. Тогда пределы для таких показателей устанавливает эксперт, исходя из опыта и интуиции экспериментатора, то есть оценка субъективна. Устанавливая предел допустимых значений, следует учитывать, что ограничения могут быть установлены как с одной, так и с двух сторон. Односторонние ограничения устанавливаются, если улучшение функции желательности происходит только при однонаправленном изменении показателя (уменьшении или увеличении).

– выбор шкалы желательности. Для получения шкалы желательности удобно использовать таблицу соответствий, разработанную между отношениями предпочтения в эмпирической и числовой (психологической) системах. Масштаб должен быть одинаковым для всех объединенных параметров (ответов). Однако стандартные значения шкалы желательности не являются обязательными;

– определение уровня желательности в соответствии со шкалой Харрингтона для каждого параметра оценки.

**2 АЛГОРИТМИЧЕСКИЙ АНАЛИЗ ЗАДАЧИ**

**2.1 Постановка задачи**

Для решения поставленной задачи требуется разработать приложение для механического цеха, которое планирует установить на входных воротах тепловую завесу в количестве не менее 3. Для начала установки нужно определить наиболее оптимальные варианты для приобретения. При этом присутствуют следующие ограничения:

– годовое время работы завес (не более 5000 часов);

– общая стоимость завес (не более 200 миллионов рублей);

– суммарная электрическая мощность (не более 120 кВт).

Исходными данными является информация с характеристиками тепловых завес, которая должна быть найдена в интернете (таблица 2.1).

Таблица 2.1 – Информация о тепловых завесах

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название** | **Стоимость**  **(руб.)** | **Мощность (кВт)** | **Расход воздуха (м^3)** | **Необходимость водяного пара** |
| 1 | 2 | 3 | 4 | 5 |
| Ballu BHC-L06S03-S | 265 | 120 | 350 | Нет |
| Ballu BHC-L08-T03 | 408 | 90 | 600 | Есть |
| Ресанта ТЗ-3С (67/6/1) | 222,79 | 200 | 390 | Нет |
| Ballu BHC-CE-3L | 236 | 160 | 250 | Есть |
| Ballu BHC-CE-3T | 341 | 180 | 400 | Нет |
| BHC-L06-S03 | 302 | 110 | 350 | Есть |
| Ballu BHC-L10T05 | 571 | 300 | 750 | Есть |

Продолжение таблицы 2.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Ballu BHC-L08-S05 | 439 | 190 | 450 | Есть |
| Ballu BHC-L15-S09-M | 890 | 185 | 1050 | Есть |
| Ballu BHC-M10T06-PS | 988 | 142 | 1500 | Нет |
| Ballu BHC-M10T09-PS | 1008 | 100 | 1500 | Есть |
| Ballu BHC-L09S05-ST | 453 | 90 | 480 | Есть |
| Ballu BHC-CE-3 | 268 | 140 | 300 | Есть |
| Ballu BHC-B20T12-PS | 1398 | 150 | 2200 | Есть |
| Timberk THC AERO II | 360 | 180 | 480 | Нет |
| Ballu BHC-B10T06-PS | 816 | 170 | 1100 | Есть |
| Ballu BHC-L15S09-SP | 831 | 160 | 1050 | Нет |

Для решения поставленной задачи необходимо разработать приложение, которое выполнит однокритериальную и многокритериальную оптимизацию для выбора наилучшего варианта установки завес.

Приложение должно иметь следующие функции:

– производить однокритериальную оптимизацию по удельному расходу электроэнергии;

– производить проверку решения методом полного перебора;

– производить многокритериальную оптимизацию.

Задача многокритериальной оптимизация делится на следующие подзадачи:

– перевод качественных оценок в числовую форму по шкале Харрингтона;

– определение значимости (веса) критериев методом экспертного анализа (метод Ранга) для двух экспертов;

– отбор бесперспективных альтернатив с помощью множества Парето;

– определение лучших альтернатива методом анализа иерархий.

Для оценки эффективности работы приложения, необходимо увеличить число альтернатив.

## **2.2 Описание алгоритмов решения задач**

***2.2.1*** Задача однокритериальной оптимизации решается методом полного перебора. Полный перебор – это один из наиболее простых методов оптимизации, который универсален и подходит не только для задачи оптимизации, но и для многих других. Суть данного метода заключается в последовательном переборе всех элементов, имеющих различные значения некоторых характеристик. В ходе перебора находится элемент, у которого значение параметра лучше всего подходит под изначальные требования к результату. Недостаток данного метода – это долгое время выполнения при больших количествах обрабатываемых элементов, поэтому данный способ применяется в первую очередь для однокритериальной оптимизации, где параметром выступает единственный критерий. Основная сложность при реализации метода перебора – это реализация максимально эффективного алгоритма для наиболее быстрого решения.

Перед использованием метода полного перебора в задаче однокритериальной оптимизации нужно сначала сделать фильтрацию альтернатив по заданным параметрам:

– годовое время работы не должно превышать определённое значение (в данном случае 5000 часов);

– общая стоимость не должна превышать определённое значение (в данном случае 200 миллионов рублей);

– суммарная электрическая мощность не должна превышать определённое значение (в данном случае 120 киловатт).

После фильтрации альтернатив наступает этап нахождения лучшей альтернативы по какому-то заданному критерию, в данном случае им является удельный расход электроэнергии. Для этого нужно список альтернатив отсортировать по этому критерию и получится список, в котором в начале находятся альтернативы с наиболее низким удельным расходом электроэнергии, а в конце оставшиеся альтернативы с большим расходом электроэнергии.

***2.2.2*** Для решения задачи многокритериальной оптимизации нужно выполнить несколько этапов, подзадач.

Как и в случае с однокритериальной оптимизацией сначала требуется провести фильтрацию альтернатив по заданным параметрам, описанным выше, для получения альтернатив, которые будут удовлетворять заданным к ним требованиям.

Принятие решений как научное направление характеризуется тем, что его практически невозможно полностью формализовать, так как в процессе принятия решений используются процедуры многокритериального анализа альтернатив в условиях неполной информации о внешних факторах, влияющих на эффективность решений, учитывается субъективное мнение экспертов. Основные проблемы, возникающие при многокритериальной оптимизации:

– противоречивость критериев. Как правило, улучшение по одному критерию приводит к ухудшению по другому критерию. Например, более производительный компьютер стоит дороже;

– при многокритериальной оптимизации используются разнородные оценки: числовые, качественные («быстрый процессор», «отличная машина» и т.п.), оценки вида «да-нет», балльные оценки, оценки в виде ранжирования и т.д., которые к тому же отличаются по размерности (например, цена – в рублях, производительность компьютера – в миллиардах операций в секунду, объём и т.п.);

– критерии могут сильно различаться по важности;

– крайне сложно (иногда и невозможно) определить аналитическую зависимость между критериями.

Один из вариантов перевода качественных оценок в числовую форму – это перевод по шкале Харрингтона (таблица 2.2).

Таблица 2.2 − Шкала Харрингтона

|  |  |
| --- | --- |
| **Качественная оценка** | **Числовая оценка** |
| Очень плохо | 0,0 – 0,2 |
| Плохо | 0,2 – 0,36 |
| Удовлетворительно | 0,36 – 0,63 |
| Хорошо | 0,63 – 0,8 |
| Отлично | 0,8 – 1,0 |

В случае, если две альтернативы имеют одинаковую оценку «хорошо», но по мнению эксперта или ЛПР вторая альтернатива лучше, то первой альтернативе можно назначить оценку 0,7, а второй – 0,8.

Для оценок, имеющих вид «да-нет», используют следующую шкалу: «да» – 0,67; «нет» – 0,33.

Оценки различного рода, представленные различными системами измерений, заменяются экспертными оценками, представленными в виде балльных оценок, в долях единицы, в виде парных сравнений, в виде ранжирования и т.д.

Нормирование критериев. Числовые оценки из произвольного диапазона приводят к единому масштабу, то есть оценкам, лежащим в диапазоне [0;1]. При этом, как правило, лучшей оценке соответствует большее значение. Формула нормализации:

|  |  |  |
| --- | --- | --- |
|  |  | (2.1) |

где *x* – список всех значений;

*z* – нормализованные значения, лежащие в диапазоне [0;1];

*i* – номер значения в списке.

При решении сложных и сложных задач, особенно в условиях неопределенности и неполноты информации, широко используются методы экспертного анализа. Идея экспертизы заключается в том, что к получению оценок привлекаются люди, разбирающиеся в этой области, эксперты, которые проводят логико-интуитивный анализ любого вопроса, чтобы вынести решение. Экспертные заключения обрабатываются определенным образом с использованием специальных математических процедур. В результате получаются так называемые экспертные оценки. Следует отметить, что экспертная оценка не является решением. Это только информация, которая поможет ЛПР найти оптимальное решение. В целом предпочтения лиц, принимающих решения, могут не совпадать с предпочтениями экспертов. Однако суждения экспертов и их советы помогают ЛПР, критически понимать различные точки зрения, уточнить или изменить свою систему предпочтений и тем самым снизить вероятность принятия неоптимальных решений.

Используя эти методы, для каждой пары альтернатив определяется рейтинг превосходства одной альтернативы над другой; Этот рейтинг может быть выставлен непосредственно человеком или рассчитан на основе оценок альтернатив по отдельным критериям. На основе этих сравнений определяется лучшая альтернатива. Метод попарного сравнения основан на попарном сравнении альтернатив. Для каждой пары альтернатив эксперт указывает, какая из альтернатив предпочтительнее (лучше, важнее и т.д.). Существует ряд алгоритмов, реализующих метод парных сравнений: они различаются по количеству используемых экспертных оценок (индивидуальные и коллективные оценки), по шкалам сравнения альтернатив и т.д.

Метод Ранга основан на балльных оценках альтернатив, указываемых несколькими экспертами. Каждый из экспертов (независимо от других) оценивает альтернативы по некоторой шкале (обычно — 10-балльной). Чем более предпочтительной (по мнению эксперта) является альтернатива, тем более высокий балл для нее указывается.

Каждый эксперт указывает оценки альтернатив по 10-балльной шкале. Оценки, указанные экспертами, сводятся в матрицу размером *MxN*, где *M* — число экспертов, *N* — число альтернатив. Обозначим эти оценки как *Xij*, *i=1,…,M*, *j=1*,…,*N*.

Затем находятся суммарные оценки альтернатив всеми экспертами, после чего находится сумма всех оценок.

Далее можно приступать к нахождению весов альтернатив по формуле:

*Vj = Cj/C, j=1,…,N*  (2.2)

Наиболее предпочтительной, по мнению экспертов, является альтернатива, имеющая максимальный вес.

Для данного метода также возможна проверка экспертных оценок на согласованность. Для этого рассчитываются дисперсии (оценки разброса) оценок для каждого эксперта и для каждой альтернативы. Для анализа результатов применяются различные методы математической статистики. Причем, они могут комбинироваться и варьироваться в зависимости от типа задачи и необходимого результата.

Для формирования обобщенной оценки группы экспертов чаще всего используются средние величины. Например, медиана, за которую принимается такая оценка, по отношению к которой число больших оценок равняется числу меньших.

Иногда требуется определить, насколько тот или иной фактор (объект) важен (существенен) с точки зрения какого-либо критерия. В этом случае говорят, что нужно определить вес каждого фактора. Отличается от формирования обобщенной оценки тем, что определяется не общая оценка объекта, а оценка для каждого его признака.

В случае участия в опросе нескольких экспертов расхождения в их оценках неизбежны, однако величина этого расхождения имеет важное значение. Групповая оценка может считаться достаточно надежной только при условии хорошей согласованности ответов отдельных специалистов.

Для анализа разброса и согласованности оценок применяются статистические характеристики – меры разброса или статистическая вариация.

Способы вычисления меры разброса:

а) вариационный размах

(2.3)

б) среднее линейное отклонение

(2.4)

в) среднеквадратическое отклонение

(2.5)

г) дисперсия

(2.6)

д) коэффициента ранговой корреляции Спирмэна

(2.7)

Коэффициент (величина ) может изменяться в диапазоне от –1 до +1. При полном совпадении оценок коэффициент равен единице. Равенство коэффициента минус единице наблюдается при наибольшем расхождении в мнениях экспертов.

– ранг (важность), присвоенный *i*-му объекту *j*-ым экспертом, – ранг, присвоенный *i*-му объекту *k*-ым экспертом, *di* – разница между рангами, присвоенными *i*-му объекту.

Следующее действие в алгоритме многокритериальной оптимизации – это выбор множества Парето-оптимальных решений, который представляет собой отбор перспективных альтернатив, из которых затем отбираются лучшие альтернативы.

Выбор множества Парето производится следующим образом. Все альтернативы попарно сравниваются друг с другом по всем критериям. Если при сравнении каких-либо альтернатив (обозначим их как *Yi* и *Yj*) оказывается, что одна из них не лучше другой ни по одному критерию, то её можно исключить из рассмотрения.

Представим рассмотренную процедуру отбора в виде алгоритма. Пусть множество возможных альтернатив *Y* состоит из конечного числа альтернатив и имеет вид: . Необходимо сформировать новое множество альтернатив *P*(*Y*) путём исключения альтернатив, которые не превосходят ни по одному критерию остальные альтернативы.

На первом шаге положим . То есть текущее множество Парето-оптимальных альтернатив совпадает с исходным множеством.

На втором шаге нужно проверить неравенства  по всем критериям. Если оно истинно, то нужно удалить из *P*(*Y*) альтернативу  и перейти к четвёртому шагу. В противном случае перейти к третьему шагу.

На третьем шаге проверяются неравенства  по всем критериям. Если оно истинно, то из *P*(*Y*) удаляется из .

На четвёртом шаге если , то  и нужно вернуться на второй шаг. В противном случае нужно перейти к пятому шагу.

На последнем шаге алгоритма, если , то ,  и затем вернуться к второму шагу. В противном случае нужно завершить вычисления, так как множество Парето-оптимальных альтернатив сформировано.

Как правило, во множество Парето входит несколько альтернатив. Поэтому выбор множества Парето не обеспечивает принятия окончательного решения (выбора одной лучшей альтернативы), однако позволяет сократить количество рассматриваемых альтернатив, т.е. упрощает принятие решения.

Далее следует метод анализа иерархий (МАИ) (*Analytic Hierarchy Process - AHP*), или подход аналитической иерархии предполагает декомпозицию проблемы на простые составляющие части и обработку суждений лица, принимающего решения (ЛПР). В результате определяется относительная значимость исследуемых альтернатив для всех критериев, находящихся в иерархии. Относительная значимость выражается численно в виде векторов приоритетов. Полученные таким образом значения векторов являются оценками в шкале отношений и соответствуют так называемым жестким оценкам.

МАИ – это структурированный метод организации и анализа сложных решений, основанный на математике и психологии. Алгоритм был разработан Томасом Л. Саати в 1970-х годах и представляет собой точный подход к количественной оценке весов критериев принятия решения. Опыт отдельных экспертов используется для оценки относительной величины факторов посредством парных сравнений. Каждый респондент должен сравнить относительную важность между двумя пунктами специальной анкеты.

МАИ имеет особое применение при групповом принятии решений и используется во всем мире в самых разных ситуациях принятия решений в таких областях, как правительство, бизнес, промышленность, здравоохранение, судостроение и образование.

Иерархическая структура – это графическое представление проблемы в виде перевёрнутого дерева, где каждый элемент, за исключением самого верхнего, зависит от одного или более выше расположенных элементов. Часто в различных организациях распределение полномочий, руководство и эффективные коммуникации между сотрудниками организованы в иерархической форме.

Иерархические структуры используются для лучшего понимания сложной реальности: мы раскладываем исследуемую проблему на составные части; затем разбиваем на составные части получившиеся элементы и т.д. На каждом шаге важно фокусировать внимание на понимании текущего элемента, временно абстрагируясь от всех прочих компонентов. При проведении подобного анализа приходит понимание всей сложности и многогранности исследуемого предмета. В качестве примера можно привести иерархическую структуру, которая используется при обучении в медицинских вузах.

В рамках изучения анатомии отдельно рассматривается костно-мышечная система (которая включает такие элементы, как руки и их составляющие: мышцы и кости), сердечно-сосудистая система (и её множественные уровни), нервная система (и её компоненты и подсистемы) и т.д. Степень детализации доходит до клеточного и молекулярного уровня. В конце изучения приходит понимание системы организма в целом, а также осознание того, какую роль в нем занимает каждая часть. С помощью подобного иерархического структурирования студенты приобретают всесторонние знания об анатомии.

Аналогичным образом, когда мы решаем сложную проблему, мы можем использовать иерархию как инструмент для обработки и восприятия больших объёмов информации. По мере проектирования этой структуры формируется всё более полное понимание проблемы.

Вместо того, чтобы предписывать «правильное» решение, МАИ помогает лицам, принимающим решения, найти то, которое лучше всего соответствует их цели и их пониманию проблемы. Он обеспечивает всеобъемлющую и рациональную основу для структурирования проблемы решения, для представления и количественной оценки ее элементов, для соотнесения этих элементов с общими целями и для оценки альтернативных решений.

В МАИ разлагается проблема решения на иерархию более легко понимаемых подзадач, каждую из которых можно анализировать независимо. Элементы иерархии могут относиться к любому аспекту проблемы принятия решения – материальному или нематериальному, тщательно измеренному или грубо оценённому, хорошо или плохо понятому – ко всему, что относится к рассматриваемому решению.

После того, как иерархия построена, систематически оцениваются различные элементы, сравниваются друг с другом по два за раз, с точки зрения их влияния на элемент, находящийся над ними в иерархии. При проведении сравнений, есть возможность использовать конкретные данные об элементах, но обычно используются суждения об относительном значении и важности элементов. Суть заключается в том, что человеческие суждения, а не только лежащая в основе информация, могут использоваться при выполнении оценок.

МАИ преобразует эти оценки в числовые значения, которые можно обрабатывать и сравнивать во всем диапазоне проблемы. Для каждого элемента иерархии выводится числовой вес или приоритет, что позволяет рационально и согласованно сравнивать различные и часто несоизмеримые элементы друг с другом. Эта возможность отличает МАИ от других методов принятия решений.

На заключительном этапе процесса численные приоритеты рассчитываются для каждой из альтернативных решений. Эти числа представляют относительную способность альтернатив по достижению цели решения, поэтому они позволяют напрямую рассматривать различные варианты действий.

Несколько фирм поставляют компьютерное программное обеспечение для помощи в использовании этого процесса. Хотя он может использоваться отдельными лицами, работающими над прямыми решениями, МАИ наиболее полезен там, где группы людей работают над сложными проблемами, особенно теми, которые с высокими ставками, включая человеческое восприятие и суждения, решения которых имеют долгосрочные последствия. Алгоритм имеет уникальные преимущества, когда важные элементы решения трудно количественно оценить или сравнить, или, когда общение между членами команды затруднено из-за их различной специализации, терминологии.

Принятия решения, к которым может применяться МАИ, включают:

– выбор одной альтернативы из заданного набора альтернатив, обычно при наличии нескольких критериев принятия решения;

– упорядочение набора альтернатив от наиболее к наименее желательным;

– определение относительной ценности членов набора альтернатив, в отличие от выбора одного или простого их ранжирования;

– распределение ресурсов между набором альтернатив;

– сравнение процессов в своей собственной организации с процессами других лучших в своем классе организаций;

– работа с многомерными аспектами качества и повышения качества;

– разрешение споров между сторонами с явно несовместимыми целями или позициями.

Приложения МАИ к сложным ситуациям принятия решений исчисляются тысячами и дали обширные результаты в решении проблем, связанных с планированием, распределением ресурсов, установкой приоритетов и выбором среди альтернатив. Другие области включают прогнозирование, общее управление качеством, реинжиниринг бизнес-процессов, развертывание функций качества и сбалансированную систему показателей. Многие приложения МАИ никогда не сообщаются всему миру, поскольку они размещаются на высоких уровнях крупных организаций, где соображения безопасности и конфиденциальности запрещают их раскрытие.

МАИ иногда используется при разработке узкоспециализированных процедур для конкретных ситуаций, таких как рейтинг зданий по историческому значению.

Использование МАИ включает математический синтез многочисленных суждений о решаемой проблеме. Нередко число таких приговоров исчисляется десятками или даже сотнями. Хотя математические вычисления можно выполнять вручную или с помощью калькулятора, гораздо более распространено использование одного из нескольких компьютеризированных методов для ввода и синтеза суждений. Самые простые из них включают стандартное программное обеспечение для работы с электронными таблицами, в то время как самые сложные используют специализированное программное обеспечение, часто дополняемое специальными устройствами для сбора мнений лиц, принимающих решения, собравшихся в конференц-зале.

Процедуру использования МАИ можно разделить на следующие подзадачи:

– моделирование проблемы как иерархию, содержащую цель решения, альтернативы ее достижения и критерии для оценки альтернатив;

– установка приоритетов между элементами иерархии, сделав серию суждений, основанных на попарных сравнениях элементов;

– синтез суждений, чтобы получить набор общих приоритетов для иерархии;

– проверка последовательности суждений;

– приход к окончательному решению по результатам этого процесса.

# **3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ЗАДАЧИ**

## **3.1 Интерфейс приложения для однокритериальной оптимизации**

Разработанное приложение является программой с графическим пользовательским интерфейсом. Программа написана на языке программирования *C#*, для графического интерфейса была использована платформа *Window Form*. Доступ к базе данных был осуществлен с помощью специальных классов *Linq to SQL*. Язык *C#* выбран, потому что с его помощью можно создавать кроссплатформенные приложения без изменений под конкретную платформу или операционную систему. Приложение разработано с использованием объектно-ориентированной парадигмы. Разработанное приложение позволяет проводить однокритериальную и многокритериальную оптимизации. Текст программы представлен в приложении Б.

Программное средство однокритериальной оптимизации входит в состав комплекса Многокритериальной оптимизации при проектировании тепловых завес механического цеха. К реализации однокритериальной оптимизации относятся пункты меню для ввода исходных данных:

– файл / 20 завес;

– файл / 50 завес;

– файл / 100 завес;

– выход.

Меню «Файл» показано на рисунке 3.1 в режиме проектирования среды «*Visual Studio Community* 2022».

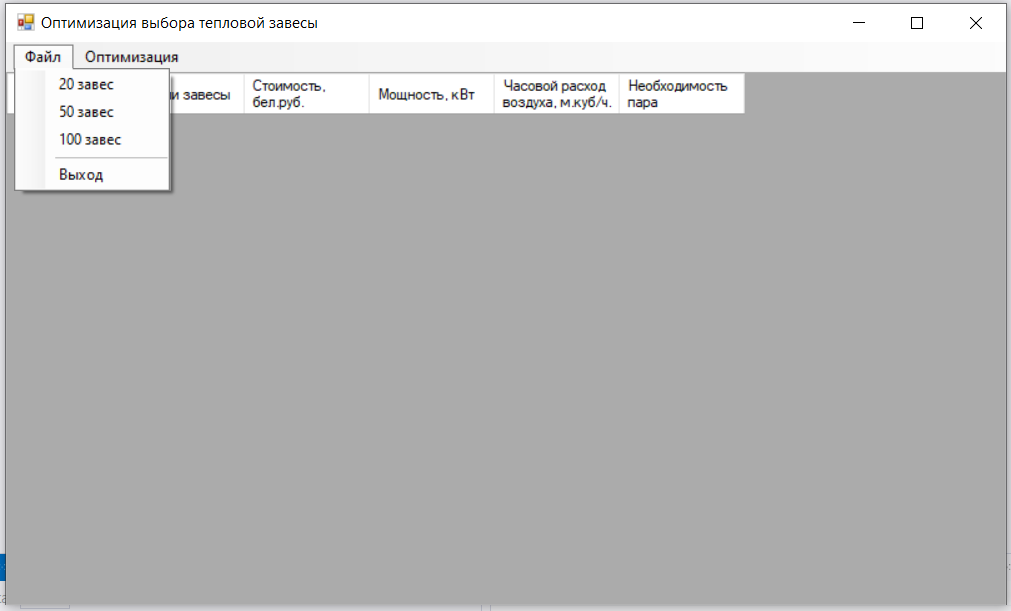


Рисунок 3.1 – Меню «Файл» главного окна программы

Также к реализации однокритериальной оптимизации относится пункт меню «Оптимизация / Однокритериальная», который показан на рисунке 3.2 в режиме проектирования.

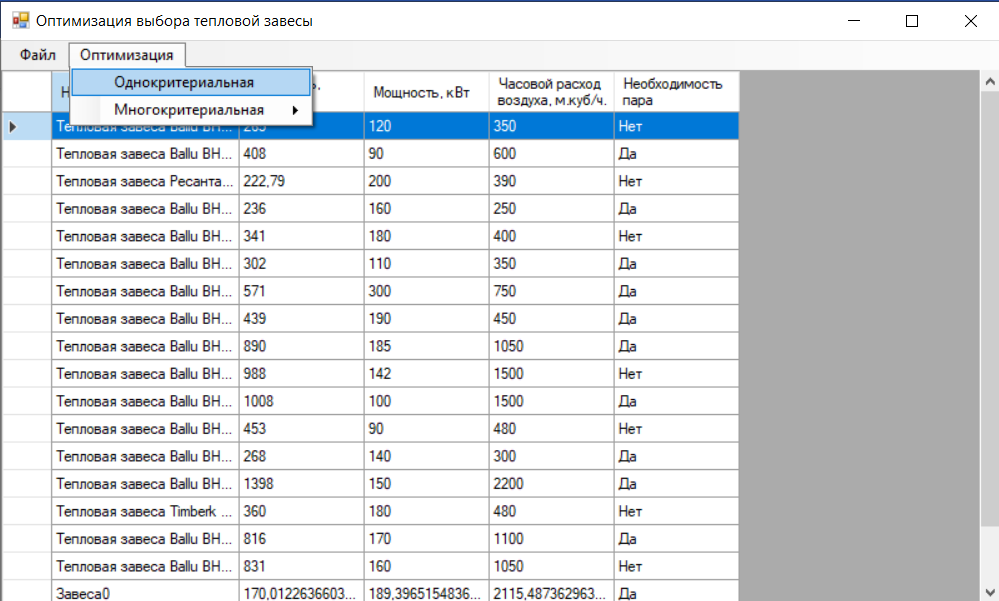


Рисунок 3.2 – Меню «Оптимизация» главного окна программы

Однокритериальная оптимизация может быть реализована для 20, 50 и 100 завес. Результаты однокритериальной оптимизации для 20 вариантов завес показаны на рисунках 3.3 и 3.4.

В начале необходимо вывести альтернативы из меню «Файл / 20 завес» на главное окно программы:

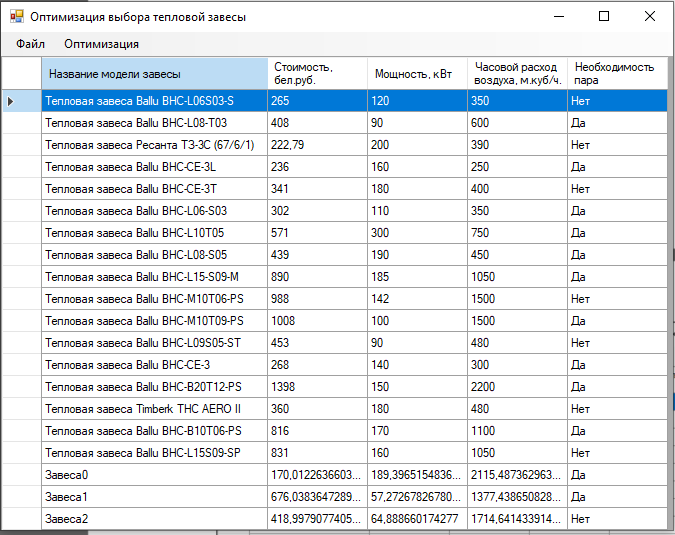


Рисунок 3.3 – Вывод 20 альтернатив

Далее выбираем в меню «Оптимизация» пункт «Однокритериальная»:

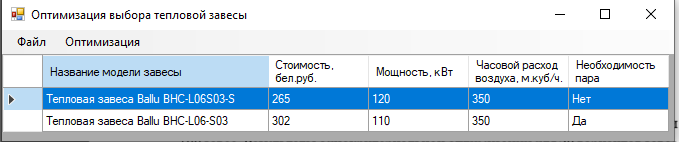


Рисунок 3.4 – Результат однокритериальной оптимизации для 20 альтернатив тепловых завес

Аналогичным образом производится вывод результатов для 50 альтернатив, как показано на рисунках 3.5 и 3.6. Вначале необходимо вывести альтернативы:

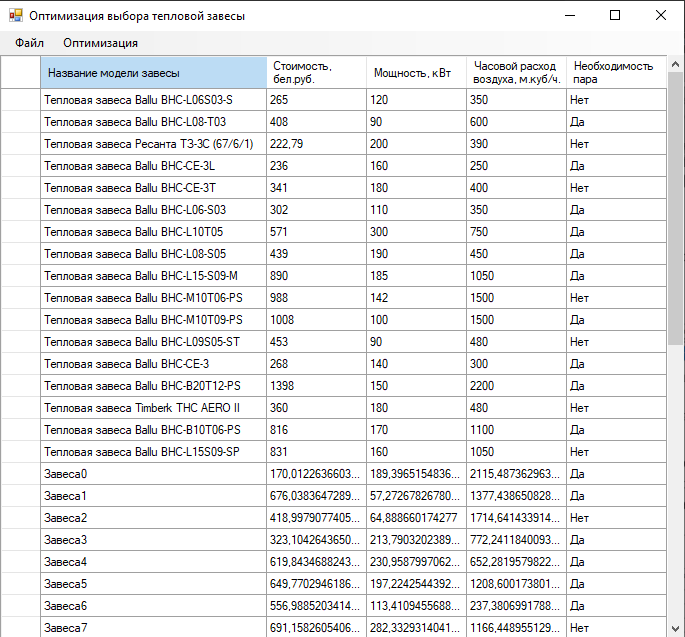


Рисунок 3.5 – Вывод 50 альтернатив

Далее выбираем в меню «Оптимизация» пункт «Однокритериальная» и получаем результат для 50 альтернатив:

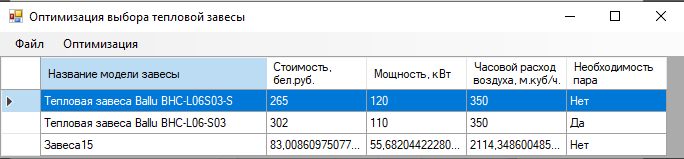


Рисунок 3.6 – Результат однокритериальной оптимизации для 50 альтернатив

Для 100 альтернатив проводим те же операции, показанные на рисунках 3.7, 3.8. В начале необходимо вывести альтернативы из меню «Файл / 100 завес» на главное окно программы:

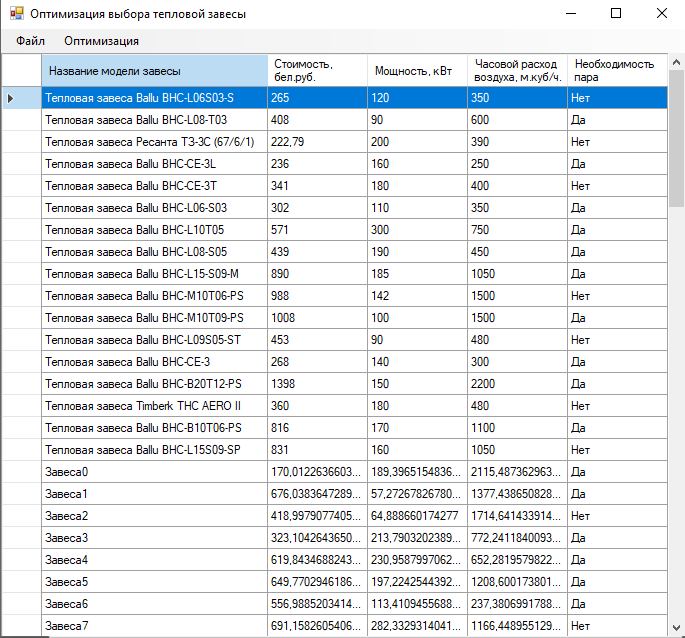


Рисунок 3.7 – Вывод 100 альтернатив

Далее выбираем в меню «Оптимизация» пункт «Однокритериальная» и получаем результат для 100 альтернатив:

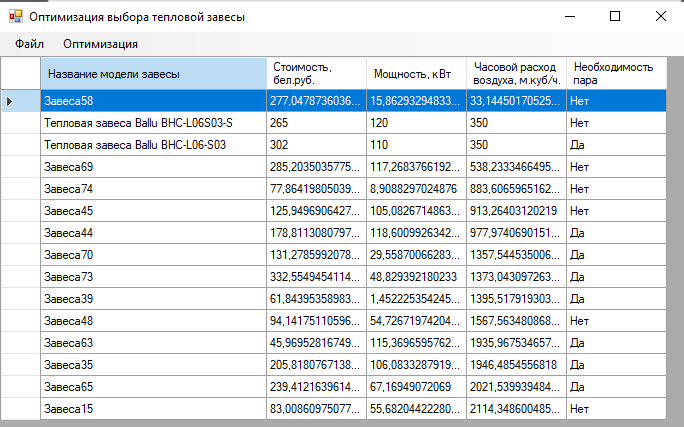


Рисунок 3.8 – Результат однокритериальной оптимизации для 100 альтернатив

В результате расчёта производится фильтрация по установленным в момент нажатия значениям ограничений параметров, далее с помощью метода полного перебора из оставшихся альтернатив будут найдены лучшие альтернативы по критерию расхода электроэнергии и показан результат в таблице, где альтернативы будут отсортированы от лучше к худшей, сверху вниз в списке.

**3.2 Интерфейс приложения для многокритериальной оптимизации**

Задача многокритериальной оптимизация делится на следующие подзадачи:

– перевод качественных оценок в числовую форму по шкале Харрингтона;

– определение значимости (веса) критериев методом экспертного анализа (метод Ранга) для двух экспертов;

– отбор бесперспективных альтернатив с помощью множества Парето;

– определение лучших альтернатива методом анализа иерархий.

На рисунке 3.9 представлен перевод оценок в числовую форму по шкале Харрингтона.

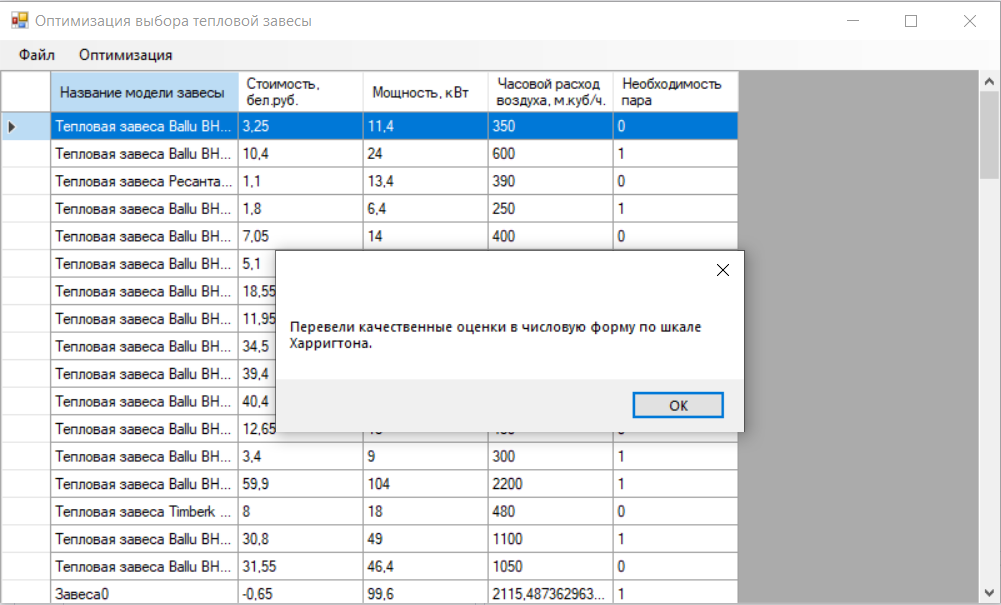


Рисунок 3.9 – Перевод качественных оценок в числовую форму по шкале Харрингтона

Далее выводятся бесперспективные альтернативы с помощью множества Парето, как показано на рисунке 3.10.

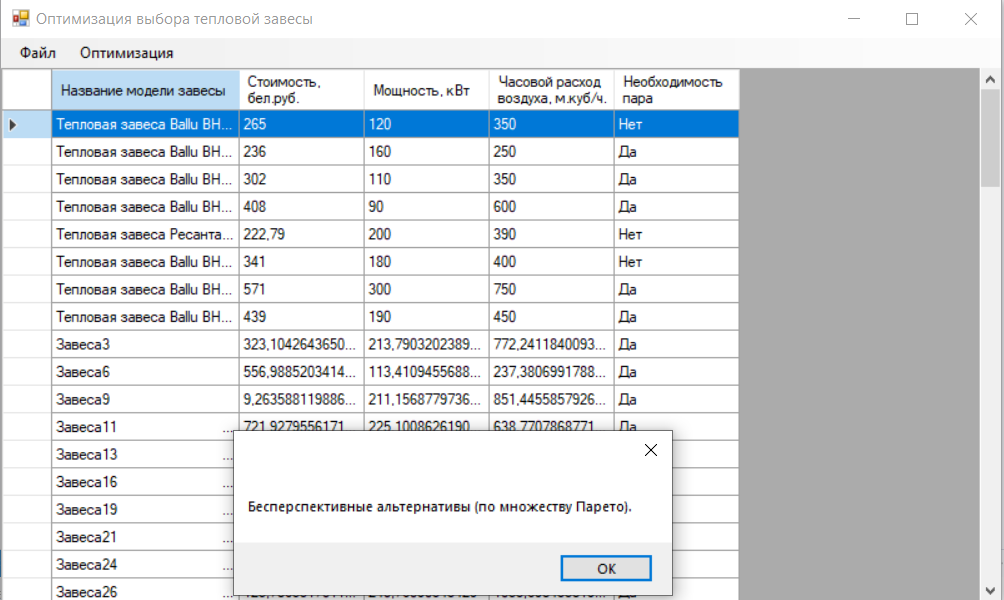


Рисунок 3.10 – Отбор бесперспективных альтернатив с помощью множества Парето

Следующим шагом необходимо определить значимости (веса) критериев методом экспертного анализа (метод Ранга) для двух экспертов. Для этого необхолимо обозначить параметры экспертов. Параметры экспертов представлены на рисунке 3.11.

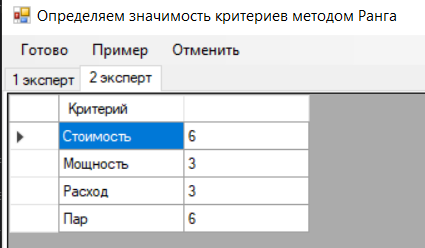
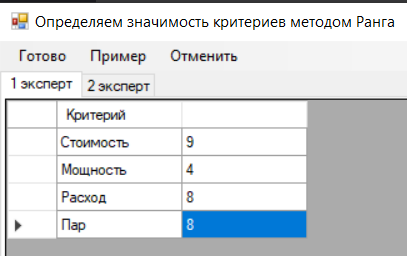


Рисунок 3.11 – Значения критериев экспертов для определения весов

Далее необходимо выбрать пункт меню «Готово». После выбора будет доступен результат вычисления весов критериев. Результаты вычисления представлены на рисунке 3.12.

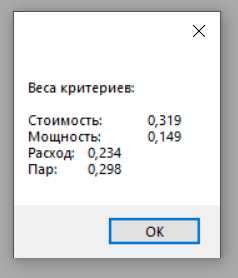


Рисунок 3.12 – Веса критериев

Далее доступен вывод результатов весов альтернатив по критериям «Стоимость», «Мощность», «Расход» и «Пар».

На рисунке 3.13 показаны веса альтернатив по критерию «Стоимость» и «Мощность».

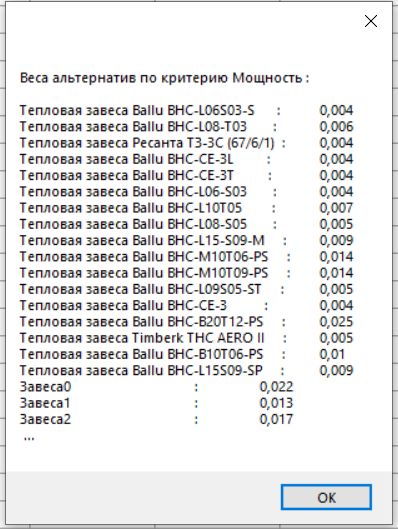
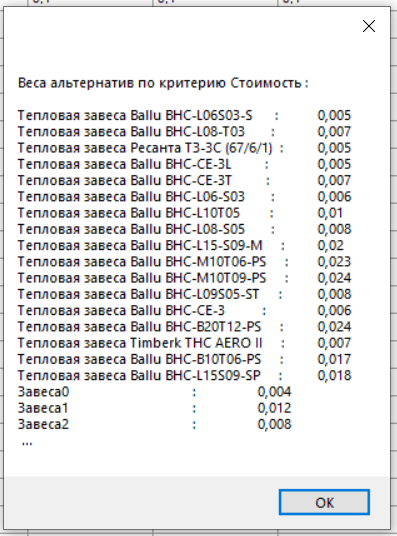


Рисунок 3.13 – Веса альтернатив по критерию «Стоимость» и «Мощность».

На рисунке 3.14 показаны веса альтернатив по критерию «Расход».

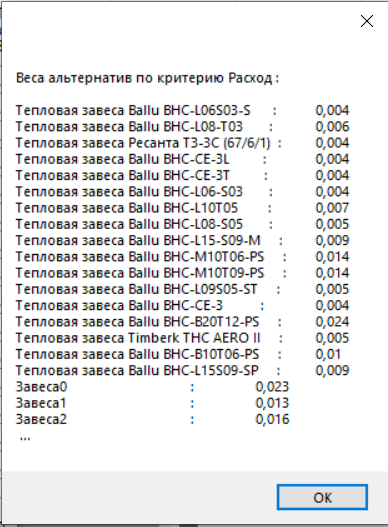


Рисунок 3.14 – Веса альтернатив по критерию «Расход».

На рисунке 3.15 показаны веса альтернатив по критерию «Пар».

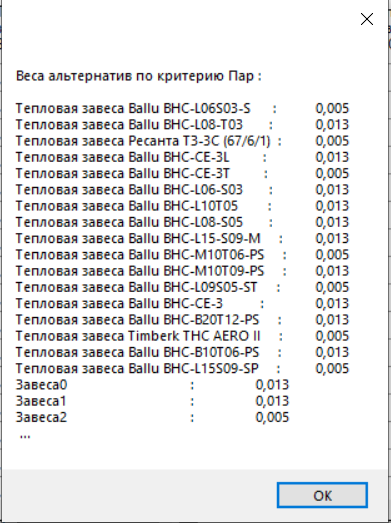


Рисунок 3.15 – Веса альтернатив по критерию «Пар»

После вывода весов по всем критериям, проводится экспертный анализ. Результат вычисления представлен на рисунке 3.16.

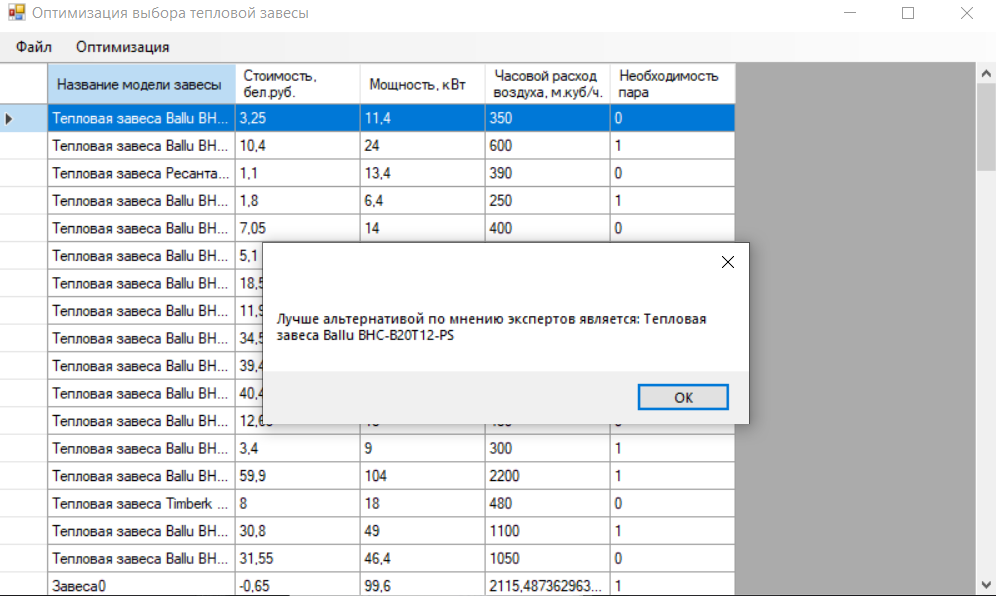


Рисунок 3.16 – Результат нахождения лучшей альтернативы методом Ранга для двух экспертов

На рисунке 3.17 показан результат вычисления лучшей альтернативы методом анализа иерархий.

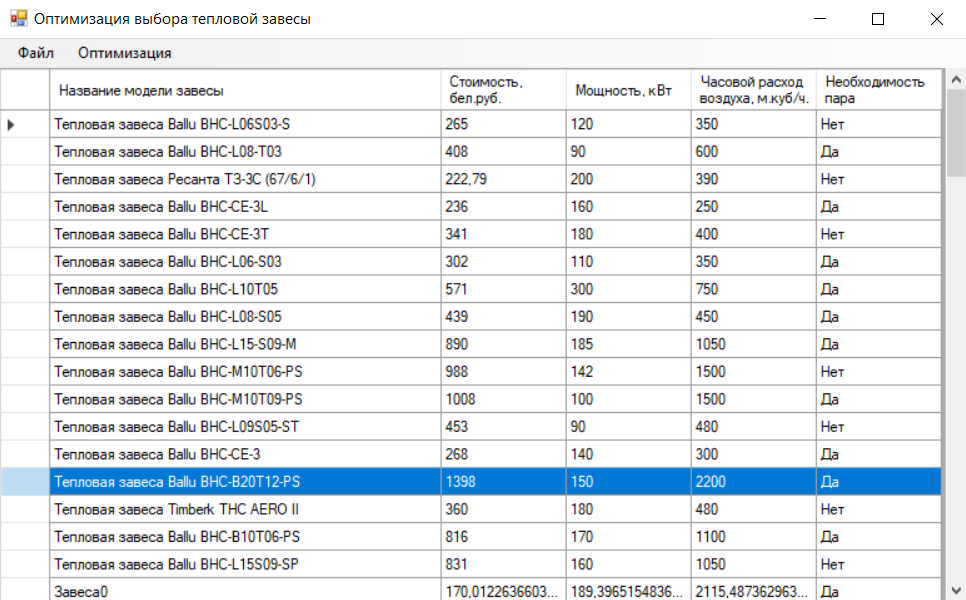


Рисунок 3.17 – Лучшая альтернатива, вычисленная с помощью МАИ

# **ЗАКЛЮЧЕНИЕ**

Разработано программное обеспечение для однокритериальной и многокритериальной оптимизаций при проектировании тепловых завес механического цеха на языке программирования *C#*, для графического интерфейса была использована платформа *Window Form*. Особенностью приложения является кроссплатформенность, то есть возможность запустить программу на любой операционной системы, а также удобный и интуитивно понятный пользовательский интерфейс. Получившаяся программа полностью удовлетворяет всем поставленным требованиям и выполняет все необходимые задачи.

Разработанная программа позволяет пользователю на основе вводимых данных используя метод полного перебора, метода экспертного анализа (метод Ранга) для двух экспертов, множеством Парето отобрать бесперспективные альтернативы, определять лучшую альтернативу методом анализа иерархий и эффективно решить задачи по нахождению лучших вариантов тепловых завес (альтернатив) среди списка представленных и использовать результат на практике, как дополнительное средство при тепловых завес для установки в механическом цеху. Также есть возможность использовать два вида оптимизации, что может дать дополнительную информацию пользователю при выборе.

В результате выполнения курсового проекта был приобретён опыт работы с однокритериальной и многокритериальной оптимизациями. Также была изучена предметная область – проектирование установки тепловых завес, ведь данная мера является весьма эффективной, когда речь идет о сохранении тепла в помещении, устранении сквозняков или сбережении охлажденного воздуха в холодильных установках промышленного назначения.

# **Список использованных источников**

1.  МНОГОКРИТЕРИАЛЬНАЯ ОПТИМИЗАЦИЯ. ПАРЕТО-ОПТИМАЛЬНЫЕ РЕШЕНИЯ – Студенческий научный форум. – Электрон. данные. – Режим доступа: https://scienceforum.ru/2021/article/.18024620. – Дата доступа: 01.01.2022.

2. Колемаева В. А. Методы оптимальных решений. Практикум : учеб. Пособие / В. А. Колемаева, В. И. Соловьёва. – М. : КНОРУС, 2017. – 194 с.

3. Кочкарова А. А. Методы принятия управленческих решений: количественный подход / А. А. Кочкарова. – М. : КНОРУС, 2016. – 146 с.

4. Лобачевский Н. И., Методы оптимизации в примерах и задачах / Н. И. Лобачевский, Бирюков Р. С. – Н. Новгород : Нижегородский госуниверситет, 2015. – 101 с.

5. Набатова Д. С. Математические и инструментальные методы поддержки принятия решений / Д. С. Набатова. – М. : Издательство Юрайт, 2015. – 292 с.

6. Тема 3.5. Многокритериальная оптимизация – Студопедия. – Студенческий научный форум. – Электрон. данные. – Режим доступа: https://studopedia.ru/39903\_tema--mnogokriterialnaya-optimizatsiya.html. – Дата доступа: 07.01.2022.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Чертёж схемы алгоритма**

**ПРИЛОЖЕНИЕ Б**

(обязательное)

**Текст программы**

**Program.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Proj

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

**Dbo.Zavesi:**

CREATE TABLE [dbo].[Zavesi] (

[Id] INT IDENTITY (1, 1) NOT NULL,

[Name] NCHAR (40) NOT NULL,

[Cost] FLOAT (53) NOT NULL,

[Power] FLOAT (53) NOT NULL,

[Rashod] FLOAT (53) NOT NULL,

[Par] INT NOT NULL,

PRIMARY KEY CLUSTERED ([Id] ASC)

)

**Form1.Designer.cs**

namespace Proj

{

partial class Form1

{

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором форм Windows

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

private void InitializeComponent()

{

this.grid = new System.Windows.Forms.DataGridView();

this.Id = new System.Windows.Forms.DataGridViewTextBoxColumn();

this.Name0 = new System.Windows.Forms.DataGridViewTextBoxColumn();

this.Cost = new System.Windows.Forms.DataGridViewTextBoxColumn();

this.Udel = new System.Windows.Forms.DataGridViewTextBoxColumn();

this.Pak = new System.Windows.Forms.DataGridViewTextBoxColumn();

this.Air = new System.Windows.Forms.DataGridViewTextBoxColumn();

this.menuStrip1 = new System.Windows.Forms.MenuStrip();

this.файлToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.MenuBegin = new System.Windows.Forms.ToolStripMenuItem();

this.Menu100 = new System.Windows.Forms.ToolStripMenuItem();

this.toolStripMenuItem1 = new System.Windows.Forms.ToolStripSeparator();

this.MenuClose = new System.Windows.Forms.ToolStripMenuItem();

this.оптимизацияToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.однокритериальнаяToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.многокритериальнаяToolStripMenuItem1 = new System.Windows.Forms.ToolStripMenuItem();

this.бесперспективныеАльтернативыпоМножествуПаретоToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.лучшаяАльтернативаToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.завесToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

((System.ComponentModel.ISupportInitialize)(this.grid)).BeginInit();

this.menuStrip1.SuspendLayout();

this.SuspendLayout();

//

// grid

//

this.grid.AllowUserToAddRows = false;

this.grid.AllowUserToDeleteRows = false;

this.grid.BorderStyle = System.Windows.Forms.BorderStyle.None;

this.grid.ColumnHeadersHeightSizeMode = System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;

this.grid.Columns.AddRange(new System.Windows.Forms.DataGridViewColumn[] {

this.Id,

this.Name0,

this.Cost,

this.Udel,

this.Pak,

this.Air});

this.grid.Dock = System.Windows.Forms.DockStyle.Fill;

this.grid.Location = new System.Drawing.Point(0, 24);

this.grid.Name = "grid";

this.grid.ReadOnly = true;

this.grid.SelectionMode = System.Windows.Forms.DataGridViewSelectionMode.FullRowSelect;

this.grid.Size = new System.Drawing.Size(800, 426);

this.grid.TabIndex = 0;

this.grid.CellValueChanged += new System.Windows.Forms.DataGridViewCellEventHandler(this.grid\_CellValueChanged);

//

// Id

//

this.Id.HeaderText = "Column1";

this.Id.Name = "Id";

this.Id.ReadOnly = true;

this.Id.Visible = false;

//

// Name0

//

this.Name0.HeaderText = "Название модели завесы";

this.Name0.Name = "Name0";

this.Name0.ReadOnly = true;

this.Name0.SortMode = System.Windows.Forms.DataGridViewColumnSortMode.NotSortable;

this.Name0.Width = 150;

//

// Cost

//

this.Cost.HeaderText = "Стоимость, бел.руб.";

this.Cost.Name = "Cost";

this.Cost.ReadOnly = true;

this.Cost.SortMode = System.Windows.Forms.DataGridViewColumnSortMode.NotSortable;

//

// Udel

//

this.Udel.HeaderText = "Мощность, кВт";

this.Udel.Name = "Udel";

this.Udel.ReadOnly = true;

this.Udel.SortMode = System.Windows.Forms.DataGridViewColumnSortMode.NotSortable;

//

// Pak

//

this.Pak.HeaderText = "Часовой расход воздуха, м.куб/ч.";

this.Pak.Name = "Pak";

this.Pak.ReadOnly = true;

this.Pak.SortMode = System.Windows.Forms.DataGridViewColumnSortMode.NotSortable;

//

// Air

//

this.Air.HeaderText = "Необходимость пара";

this.Air.Name = "Air";

this.Air.ReadOnly = true;

this.Air.SortMode = System.Windows.Forms.DataGridViewColumnSortMode.NotSortable;

//

// menuStrip1

//

this.menuStrip1.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.файлToolStripMenuItem,

this.оптимизацияToolStripMenuItem});

this.menuStrip1.Location = new System.Drawing.Point(0, 0);

this.menuStrip1.Name = "menuStrip1";

this.menuStrip1.Size = new System.Drawing.Size(800, 24);

this.menuStrip1.TabIndex = 1;

this.menuStrip1.Text = "menuStrip1";

//

// файлToolStripMenuItem

//

this.файлToolStripMenuItem.DropDownItems.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.MenuBegin,

this.завесToolStripMenuItem,

this.Menu100,

this.toolStripMenuItem1,

this.MenuClose});

this.файлToolStripMenuItem.Name = "файлToolStripMenuItem";

this.файлToolStripMenuItem.Size = new System.Drawing.Size(48, 20);

this.файлToolStripMenuItem.Text = "Файл";

//

// MenuBegin

//

this.MenuBegin.Name = "MenuBegin";

this.MenuBegin.Size = new System.Drawing.Size(180, 22);

this.MenuBegin.Text = "20 завес";

this.MenuBegin.Click += new System.EventHandler(this.MenuBegin\_Click);

//

// Menu100

//

this.Menu100.Name = "Menu100";

this.Menu100.Size = new System.Drawing.Size(180, 22);

this.Menu100.Text = "100 завес";

this.Menu100.Click += new System.EventHandler(this.Menu100\_Click);

//

// toolStripMenuItem1

//

this.toolStripMenuItem1.Name = "toolStripMenuItem1";

this.toolStripMenuItem1.Size = new System.Drawing.Size(177, 6);

//

// MenuClose

//

this.MenuClose.Name = "MenuClose";

this.MenuClose.Size = new System.Drawing.Size(180, 22);

this.MenuClose.Text = "Выход";

this.MenuClose.Click += new System.EventHandler(this.MenuClose\_Click);

//

// оптимизацияToolStripMenuItem

//

this.оптимизацияToolStripMenuItem.DropDownItems.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.однокритериальнаяToolStripMenuItem,

this.многокритериальнаяToolStripMenuItem1});

this.оптимизацияToolStripMenuItem.Name = "оптимизацияToolStripMenuItem";

this.оптимизацияToolStripMenuItem.Size = new System.Drawing.Size(94, 20);

this.оптимизацияToolStripMenuItem.Text = "Оптимизация";

//

// однокритериальнаяToolStripMenuItem

//

this.однокритериальнаяToolStripMenuItem.Name = "однокритериальнаяToolStripMenuItem";

this.однокритериальнаяToolStripMenuItem.Size = new System.Drawing.Size(194, 22);

this.однокритериальнаяToolStripMenuItem.Text = "Однокритериальная";

this.однокритериальнаяToolStripMenuItem.Click += new System.EventHandler(this.MenuOne\_Click);

//

// многокритериальнаяToolStripMenuItem1

//

this.многокритериальнаяToolStripMenuItem1.DropDownItems.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.бесперспективныеАльтернативыпоМножествуПаретоToolStripMenuItem,

this.лучшаяАльтернативаToolStripMenuItem});

this.многокритериальнаяToolStripMenuItem1.Name = "многокритериальнаяToolStripMenuItem1";

this.многокритериальнаяToolStripMenuItem1.Size = new System.Drawing.Size(194, 22);

this.многокритериальнаяToolStripMenuItem1.Text = "Многокритериальная";

//

// бесперспективныеАльтернативыпоМножествуПаретоToolStripMenuItem

//

this.бесперспективныеАльтернативыпоМножествуПаретоToolStripMenuItem.Name = "бесперспективныеАльтернативыпоМножествуПаретоToolStripMenuItem";

this.бесперспективныеАльтернативыпоМножествуПаретоToolStripMenuItem.Size = new System.Drawing.Size(258, 22);

this.бесперспективныеАльтернативыпоМножествуПаретоToolStripMenuItem.Text = "Бесперспективные альтернативы";

this.бесперспективныеАльтернативыпоМножествуПаретоToolStripMenuItem.Click += new System.EventHandler(this.множествоПаретоToolStripMenuItem\_Click);

//

// лучшаяАльтернативаToolStripMenuItem

//

this.лучшаяАльтернативаToolStripMenuItem.Name = "лучшаяАльтернативаToolStripMenuItem";

this.лучшаяАльтернативаToolStripMenuItem.Size = new System.Drawing.Size(258, 22);

this.лучшаяАльтернативаToolStripMenuItem.Text = "Лучшая альтернатива";

this.лучшаяАльтернативаToolStripMenuItem.Click += new System.EventHandler(this.лучшаяАльтернативаМетодомАнализаИерархийToolStripMenuItem\_Click);

//

// завесToolStripMenuItem

//

this.завесToolStripMenuItem.Name = "завесToolStripMenuItem";

this.завесToolStripMenuItem.Size = new System.Drawing.Size(180, 22);

this.завесToolStripMenuItem.Text = "50 завес";

this.завесToolStripMenuItem.Click += new System.EventHandler(this.завесToolStripMenuItem\_Click);

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(800, 450);

this.Controls.Add(this.grid);

this.Controls.Add(this.menuStrip1);

this.MainMenuStrip = this.menuStrip1;

this.Name = "Form1";

this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

this.Text = "Оптимизация выбора тепловой завесы";

this.Load += new System.EventHandler(this.Form1\_Load);

((System.ComponentModel.ISupportInitialize)(this.grid)).EndInit();

this.menuStrip1.ResumeLayout(false);

this.menuStrip1.PerformLayout();

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.DataGridView grid;

private System.Windows.Forms.MenuStrip menuStrip1;

private System.Windows.Forms.ToolStripMenuItem файлToolStripMenuItem;

private System.Windows.Forms.ToolStripMenuItem MenuClose;

private System.Windows.Forms.ToolStripMenuItem MenuBegin;

private System.Windows.Forms.ToolStripMenuItem Menu100;

private System.Windows.Forms.ToolStripSeparator toolStripMenuItem1;

private System.Windows.Forms.ToolStripMenuItem оптимизацияToolStripMenuItem;

private System.Windows.Forms.ToolStripMenuItem однокритериальнаяToolStripMenuItem;

private System.Windows.Forms.ToolStripMenuItem многокритериальнаяToolStripMenuItem1;

private System.Windows.Forms.ToolStripMenuItem бесперспективныеАльтернативыпоМножествуПаретоToolStripMenuItem;

private System.Windows.Forms.ToolStripMenuItem лучшаяАльтернативаToolStripMenuItem;

private System.Windows.Forms.DataGridViewTextBoxColumn Id;

private System.Windows.Forms.DataGridViewTextBoxColumn Name0;

private System.Windows.Forms.DataGridViewTextBoxColumn Cost;

private System.Windows.Forms.DataGridViewTextBoxColumn Udel;

private System.Windows.Forms.DataGridViewTextBoxColumn Pak;

private System.Windows.Forms.DataGridViewTextBoxColumn Air;

private System.Windows.Forms.ToolStripMenuItem завесToolStripMenuItem;

}

}

**Form2.Designer.cs:**

namespace Proj

{

partial class Form2

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.tabs = new System.Windows.Forms.TabControl();

this.menuStrip1 = new System.Windows.Forms.MenuStrip();

this.MenuReady = new System.Windows.Forms.ToolStripMenuItem();

this.MenuExample = new System.Windows.Forms.ToolStripMenuItem();

this.MenuCancel = new System.Windows.Forms.ToolStripMenuItem();

this.menuStrip1.SuspendLayout();

this.SuspendLayout();

//

// tabs

//

this.tabs.Dock = System.Windows.Forms.DockStyle.Fill;

this.tabs.Location = new System.Drawing.Point(0, 24);

this.tabs.Name = "tabs";

this.tabs.SelectedIndex = 0;

this.tabs.Size = new System.Drawing.Size(800, 426);

this.tabs.TabIndex = 0;

//

// menuStrip1

//

this.menuStrip1.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.MenuReady,

this.MenuExample,

this.MenuCancel});

this.menuStrip1.Location = new System.Drawing.Point(0, 0);

this.menuStrip1.Name = "menuStrip1";

this.menuStrip1.Size = new System.Drawing.Size(800, 24);

this.menuStrip1.TabIndex = 1;

this.menuStrip1.Text = "menuStrip1";

//

// MenuReady

//

this.MenuReady.Name = "MenuReady";

this.MenuReady.Size = new System.Drawing.Size(57, 20);

this.MenuReady.Text = "Готово";

this.MenuReady.Click += new System.EventHandler(this.MenuReady\_Click);

//

// MenuExample

//

this.MenuExample.Name = "MenuExample";

this.MenuExample.Size = new System.Drawing.Size(64, 20);

this.MenuExample.Text = "Пример";

this.MenuExample.Click += new System.EventHandler(this.MenuExample\_Click);

//

// MenuCancel

//

this.MenuCancel.Name = "MenuCancel";

this.MenuCancel.Size = new System.Drawing.Size(73, 20);

this.MenuCancel.Text = "Отменить";

this.MenuCancel.Click += new System.EventHandler(this.MenuCancel\_Click);

//

// Form2

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(800, 450);

this.Controls.Add(this.tabs);

this.Controls.Add(this.menuStrip1);

this.MainMenuStrip = this.menuStrip1;

this.Name = "Form2";

this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

this.Text = "Определяем значимость критериев методом Ранга";

this.menuStrip1.ResumeLayout(false);

this.menuStrip1.PerformLayout();

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.TabControl tabs;

private System.Windows.Forms.MenuStrip menuStrip1;

private System.Windows.Forms.ToolStripMenuItem MenuReady;

private System.Windows.Forms.ToolStripMenuItem MenuExample;

private System.Windows.Forms.ToolStripMenuItem MenuCancel;

}

}

**Form3.Designer.cs**

namespace Proj

{

partial class Form3

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.grid = new System.Windows.Forms.DataGridView();

this.menuStrip1 = new System.Windows.Forms.MenuStrip();

this.MenuNext = new System.Windows.Forms.ToolStripMenuItem();

((System.ComponentModel.ISupportInitialize)(this.grid)).BeginInit();

this.menuStrip1.SuspendLayout();

this.SuspendLayout();

//

// grid

//

this.grid.AllowUserToAddRows = false;

this.grid.AllowUserToDeleteRows = false;

this.grid.ColumnHeadersHeightSizeMode = System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;

this.grid.Dock = System.Windows.Forms.DockStyle.Fill;

this.grid.Location = new System.Drawing.Point(0, 24);

this.grid.Name = "grid";

this.grid.Size = new System.Drawing.Size(800, 426);

this.grid.TabIndex = 0;

//

// menuStrip1

//

this.menuStrip1.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.MenuNext});

this.menuStrip1.Location = new System.Drawing.Point(0, 0);

this.menuStrip1.Name = "menuStrip1";

this.menuStrip1.Size = new System.Drawing.Size(800, 24);

this.menuStrip1.TabIndex = 1;

this.menuStrip1.Text = "menuStrip1";

//

// MenuNext

//

this.MenuNext.Name = "MenuNext";

this.MenuNext.Size = new System.Drawing.Size(63, 20);

this.MenuNext.Text = "Дальше";

this.MenuNext.Click += new System.EventHandler(this.MenuNext\_Click);

//

// Form3

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(800, 450);

this.Controls.Add(this.grid);

this.Controls.Add(this.menuStrip1);

this.MainMenuStrip = this.menuStrip1;

this.Name = "Form3";

this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

this.Text = "Сравнение альтернатив по критерию:";

((System.ComponentModel.ISupportInitialize)(this.grid)).EndInit();

this.menuStrip1.ResumeLayout(false);

this.menuStrip1.PerformLayout();

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.DataGridView grid;

private System.Windows.Forms.MenuStrip menuStrip1;

private System.Windows.Forms.ToolStripMenuItem MenuNext;

}

}