Шифр «Еглет»

Наукова робота на конкурс за напрямом:
Комп’ютерні науки

на тему:
«Оптимізація діяльності операторів продуктових компаній»

2020

ЗМІСТ

ВСТУП

Актуальність. Оператори продуктових компаній працюють, як правило, в умовах щільного потоку заявок,часових обмежень і стресів. Незважаючи на велику кількість наукових досліджень з питань організації діяльності продуктових компаній, питання ергономічного забезпечення діяльності операторів, які працюють в умовах обмеженого часу, досліджені недостатньо.

Предмет. Діяльність операторів продуктових компаній.

Об'єкт. Оптимізація діяльності операторів продуктових компаній, які працюють в умовах обмеженого часового ресурсу.

Мета. Розробка інформаційної системи оптимізації алгоритму функціонування людино-машинних систем з ймовірнісним обмеженням на час.

Наукова новизна. До цього часу не реалізовані алгоритми оптимізації, які сприймають час як ймовірнісну величину, що в свою чергу підвищить ефективність.

Практична цінність. Вирішення задачі оптимізації, що враховує ймовірнісний характер часу виконання діяльності оператора людино-машинних систем, що, в свою чергу, зможе надати користувачу рішення, яке гарантує необхідну своєчасність.

Публікації. За матеріалами дослідження опубліковано 9 наукових робіт.

Апробації. Результати доповідались на 8 наукових конференціях:

– International Scientific Conference «UNITECH 2017» (17-18 November 2017, Gabrovo, Bulgaria)

– на науково-практичній конференції «Цифровые технологии в образовании, науке, обществе» (Петрозаводськ, 27-30 листопада 2017 року);

– на науково-практичній конференції «Цифровые технологии в образовании, науке, обществе» (Петрозаводськ, 4-6 грудня 2018 року);

– на науковій конференції «Інформатика, математика, автоматика» ІМА 2018 (м.Суми, 5-9 лютого 2018 року);

– на студентській конференції «Перший крок у науку» (м.Суми, 24 лютого 2019 року);

– на науковій конференції «Інтелектуальний потенціал – 2018» (м.Хмельницький, 14-16 листопада 2018 року);

– на науковій конференції «Інформатика, математика, автоматика» ІМА 2019 (м.Суми, 23-26 квітня 2019 року);

– на науковій конференції «Інтелектуальний потенціал – 2019» (м.Хмельницький, 20-22 листопада 2019 року).

Впровадження: результати впроваджено:

- у навчальний процес Сумського державного університету.

1 АНАЛІЗ МЕТОДІВ МОДЕЛЮВАННЯ ТА ОПТИМІЗАЦІЇ ЛЮДИНО-МАШИННИХ СИСТЕМ

1.1 Аналіз методів опису та оцінки функціонування людино-машинних систем

Будь-яка продуктова компанія є складною людино-машинною системою з великою кількістю різноманітного персоналу, в тому числі, операторів, зайнятих безпосередньо обробкою заявок та в середньому заданим часом на їх виконання. Для того, щоб описати людино-машинні системи (ЛМС) з урахуванням "людського фактору" на даний момент використовується широкий спектр наук – від психології і біомеханіки до теорії управління та математичної логіки. Процеси функціонування ЛМС описуються за допомогою формальних систем [2,3,4,5]:

- логічні системи (формальні граматики, мережі Петрі і ін.);
- алгебраїчні системи (марковскі і напівмарковські процеси,

напівмарковські мережі обслуговування та ін.);

- мовно-алгебраїчні системи (мережі передування, PERT, GERT, МКП мережі, функціональні мережі).

У додатку А виконаний за матеріалами наукових досліджень з питань ергономіки [6-25] детальний опис методів описання та оцінки ЛМС, а порівняльний аналіз – у додатку Б.

Як видно, найбільш пристосованим для моделювання людини-оператора є метод побудови функціональної моделі професора Губінського[16].

Перелік типових функцій одиниць (ТФО), що що використовує цей метод, наведено у додатку В, а типових функцій - у додатку Г.

1.2 Аналіз методів оптимізації людино-машинних систем

Завдання оптимізації процесу функціонування (ПФ) ЛМС класифікуються за різними ознаками. Головними класифікаційними ознаками є:

а) наявність обмежень, згідно з якими безліч оптимізаційних задач ділиться на два класи (умовних і безумовних задач);

б) тип показника функціонування ЛМС (скалярні і векторні задачі оптимізації);

в) спосіб опису ПФ ЛМС (оптимізація ПФ ЛМС на основі напівмарковських процесів з використанням «графа подій», або на основі функціональних мереж із застосуванням «графа робіт»).

Зв'язок між постановками задач оптимізації ПФ ЛМС на функціональних мережах (ФМ) і на напівмарковських процесах (НМП) визначається зв'язком ФМ і НМП. Суттєвим є те, що НМП отримують після ФС. Процес отримання НМП наступний:

а) процес функціонування ЛМС описується в просторі функцією;

б) за допомогою ФМ будується повний граф подій (опис процесу функціонування ЛМС в просторі станів);

в) шляхом об'єднання вершин повного графа подій переходять до укрупнених графу подій, у якого на відміну від повного графа подій відсутній післядія по станам, в яких може перебувати ПФ ЛМС;

г) від об'єднаного графа подій переходять до НМП.

Таким чином, граф робіт може бути перетворений в граф подій (його вершини – початку, завершення подій; дуги – спосіб виконання робіт), і навпаки. Приклад таких графів для робочої операції з функціональним контролем наведено на рис.

При оптимізації ПФ ЛМС на ФМ є три можливості зміни структури процесу функціонування ЛМС: зміна структури функцій (F-структура) при фіксованій структурі елементів (S-Структура);

Оскільки в змістовній постановки задачі використовуються поняття F і S-структури, то при вирішенні задач оптимізації в просторі станів змістовний сенс значно втрачається. Крім того, часто одним станом системи відповідає результат

виконання декількох операцій на ФМ. Ця обставина приводить, по-перше, до скорочення числа способів виконання ПФ ЛМС при постановці завдань на НМП в порівнянні з постановкою на ФМ; по-друге, до неможливості врахування на НМП різних обмежень на структуру процесу функціонування ЛМС.

Однак постановка завдань на НМП має і свої переваги, а саме: універсальність опису завдань на НМП, в той час як при постановці завдань на ФМ доводиться визначати, яка структура ФМ, наприклад послідовно організована, з глобальними зворотними зв'язками і т. п.; можливість вирішення задачі оптимізації ПФ ЛМС без поглинаючих станів з безперервним часом і з урахуванням впливу післядії за часом на функціонування системи, яка на ФМ виглядає неприродньо.

Найбільш поширеними показниками, використовуваними в якості критеріїв оптимізаційних задач, є наступні: β – ймовірність правильного виконання способу функціонування; T – середній час виконання; V – середній дохід від виконання.

Спосіб опису показників тісно пов'язаний з вирішенням задачі і повинен відповідати найбільш ефективному алгоритму оптимізації. Так, наприклад, необхідно враховувати, що при аналітичному описі задача може бути вирішена методами цілочисельного і комбінаторного програмування, а при алгоритмічній – тільки комбінаторного. Скалярні задачі оптимізації характеризуються використанням в якості критерію оптимальності одного з показників функціонування β, T, V. Локальними критеріями векторних постановок задач є показники β, T і V.

Принцип (критерій) оптимальності задається двома основними способами:

1. Побудова узагальненого критерію F(Ф) = φ(β (Ф), T (Ф), V (Ф)), що є монотонною функцією локальних критеріїв.

2. Визначення алгоритму послідовного звуження безлічі альтернатив D_A з метою виділення непорожньої підмножини бажаних рішень.

Незалежно від способу визначення принципу оптимальності, рішення векторної задачі $Ф^*$ має належати множині ефективних (за Парето) рішень $D_n \in D_A$.

Оптимізація ПФ ЛМС на ФМ проводиться переважно методами дискретного програмування. Домінуюче становище в дискретному програмуванні займають комбінаторні методи. До них відносяться, в першу чергу, метод гілок і меж, а також

методи послідовного аналізу варіантів, послідовні схеми, локальні алгоритми, метод послідовних розрахунків, апроксимаційно-комбінаторний метод, принцип розширення Шоха, метод динамічного програмування та ін. Другу групу методів дискретного програмування, під загальною назвою «методи відсікання», складають алгоритми Гоморі, алгоритм Дальтона, β-алгоритм Фінкельштейна і ін.

З огляду на всі наведені вище дані, ми маємо вирішення задачі оптимізації людино-машинних систем яке не враховує ймовірнісні події, що виникають у оператора, а час розглядався як статична середня величина. Для досягнення необхідної своєчасності був розроблений алгоритм, який враховує ймовірнісний характер часового ресурсу.

1.3 Висновки

— Найбільш зручним апаратом опису функціонування людино-машинних систем є апарат функціонування мереж, що розроблено у функціонально-структурній теорії ерготехнічних систем проф. А.І. Губінського

— Оцінку надійності функціонування людино-машинних систем зручно проводити з використанням бібліотек типових функціональних одиниць та типових функціональних структур.

— Для оптимізації людино-машинної взаємодії розроблено сукупність постановок задач та методів їх вирішення.

— Серед задач оптимізації людино-машинних систем виділяють однокритеріальні та багатокритеріальні.

— В практиці ергономічного проетування найбільш часто використовується задача максимізації вірогідності безпомилкового виконання при обмеженні на математичне сподівання часу виконання, але нажаль, розв'язок такої задачі не забезпечує гарантування обмеження на вірогідність своєчасного виконання.

— У зв’язку з вищесказаним практика ергономічного проетування вимагає розробки моделей для вирішення нової задачі, в якій в якості обмеження використовується обмеження на вірогідність своєчасного виконання.

2 ПОСТАНОВКА ЗАДАЧІ ТА ОБГРУНТУВАННЯ МЕТОДУ ЇЇ ВИРІШЕННЯ

2.1 Мета та задачі

Метою є розробка інформаційної системи оптимізації алгоритму функціонування людино-машинних систем з ймовірнісним обмеженням на час. Проблемою є відсутність якісної інформаційної технології, яка допомогла б користувачу підібрати оптимальні способи виконання роботи, для вчасного виконання.

У ході проекту має бути створена універсальна інформаційна система вибору оптимальних способів виконання робіт, що дозволить набути максимального результату та гарантувати вчасність їх виконання.

Продукт – інформаційна система оптимізації алгоритму функціонування.

Результат – інформаційна система, яка дозволить знаходити оптимальні способи виконання робіт, які гарантовано виконаються вчасно, та виведення їх користувачу.

2.2 Обгрунтування методу розв’язання задачі

Детермінована оптимізаційна задача для АФ ЕТС зводиться до пошуку оптимальної стратегії поглинає ПМП за розподілом конкретних способів виконання операцій, що забезпечують збільшення ймовірності поглинання процесу в заданий поглинаючий стан s, відповідне безпомилкового результату виконання АФ, яка вирішується як завдання частково-цілочисельного лінійного програмування. Використовуємо цей висновок при постановці і рішенні стохастичною завдання з обмеженням по ймовірності на час реалізації АФ.

В якості базової теорії, що використовується для рішення задачі оптимізації вибираємо функціональні структури ерготехнічних систем професора А.І.Губінського. Обгрунтування вибору наведено в розділі 1.

Описання альтернативних підходів до розв’зання задачі оптимізації алгоритму функціонування людино-машинних систем наведено в додатку Б.

Описання типових функціональних одиниць та типових функціональних структур наведено в додатку В.

3 РОЗРОБКА МАТЕМАТИЧНОЇ МОДЕЛІ ТА АЛГОРИТМУ ОПТИМІЗАЦІЇ АЛГОРИТМУ ФУНКЦІОНУВАННЯ ЛЮДИНО-МАШИННИХ СИСТЕМ

3.1 Розробка математичної моделі задачі в загальному вигляді

Виходячи з змістовного аналізу задачі приведеного в розділі 1, можна сформулювати задачу в загальному вигляді

$$\begin{cases} B(\mathrm{X}) \to max \\ P\{T(\mathrm{X}) \le T_0\} \ge \theta_0 \\ X \in X' \\ U(\mathrm{X}) \le U_0 \end{cases} \quad (3.1)$$

$B(\mathrm{X}) \to max$,тобто максимізація ймовірності безпомилкового виконання. X – спосіб виконання операції.

Заміна обмеження на математичне сподівання на ймовірність своєчасного виконання: $P\{T(\mathrm{X}) \le T_0\} \ge \theta_0$, $T(\mathrm{X})$ – випадкова величина часу виконання, θ_0 - мінімальна ймовірність своєчасного виконання.

U(X) – витрати ресурсів при даному способі виконання. U_0 – задана кількість ресурсів.

X’ – ОДР задачі оптимізації.

Найбільш зручний інструмент для описання діяльності операторів людино-машинних систем є функціональні мережі.

Функціональна мережа – це орієнтований граф, що описує логічні і часові зв’язки між діями оператора і операціями технічних засобів. Оптимізація алгоритму функіонування може проводитися на основі двох типів моделей: граф робіт (функціональні мережі) та граф подій (використання напівмарковських процесів).

3.2 Розробка математичної моделі задачі оптимізації на графі подій

Для отримання графа подій кожній вершині графа необхідно зіставити події, що означають початок та закінчення виконання певної роботи чи операції. Результатом виконання операції може бути декілька, в простому випадку два: помилкове виконання, безпомилкове виконання. Кожному з результатів ставиться у відповідність окрема дуга на графі. Кожний наступний стан характеризується як правильний чи неправильний.

Приклад переходу від графа робіт до графа подій показаний на рис.3.1

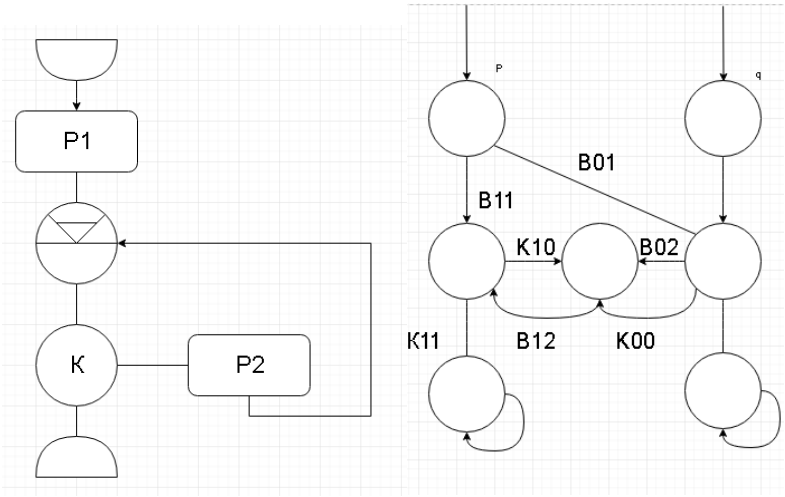


Рисунок 3.1 –Перехід від графу робіт(зліва) до графу подій(справа)

Кожному варіанту закінчення функціонування ЕТС на графі подій ставимо у відповідність своє поглинаючий стан, наприклад, «безпомилкове виконання» процесу і його «помилкове виконання». Поглинають вершини нумеруємо першими τ натуральними числами (r - кількість поглинаючих вершин). Для початкових вершин, які нумеруються числами з числової послідовності після перших τ поглинаючих вершин, необхідно задати вектор початкових ймовірностей, тобто ймовірності знаходження ЕТС в початкових станах у відповідних вершинах графа подій:

$$a = (a_{r+1}, a_{r+2}, \dots, a_m), \sum_{i=r+1}^{m} a_i = 1 \qquad (3.2)$$

Введемо наступні змінні: P_{ij}^k - ймовірність переходу ПМП з вершини і в вершину j при к-му способі виконання роботи, N - загальна кількість вершин, з яких перші r - поглинають, t_i^k - випадкова величина часу перебування процесу в вершині і

при виборі к-го рішення, $x_i^k > 0$ в тому випадку, якщо для і-ї вершини вибрано к-е рішення, то дорівнює 0 в іншому випадку, k_l - безліч допустимих рішень в -й вершині.

При таких умовах з урахуванням постановок вихідна задача формалізується наступним чином:

$$\begin{cases} \sum_{i=r+1}^{N} \sum_{k \in K_i} P_{is}^k x_i^k \to max \\ P\left\{ \sum_{i=r+1}^{N} \sum_{k \in K_i} t_i^k \le T_0 \right\} \ge \theta_0 \\ \sum_{k \in K_i} x_j^k - \sum_{i=r+1}^{N} \sum_{k \in K_i} x_i^k P_{ij}^k = a_j, \qquad j = \overline{r+1, N} \\ x_j^k \ge 0; j = \overline{r+1, N};\ k \in K_j \\ \sum_{k \in K_j} \delta_j^k = 1 \\ x_j^k - M\delta_j^k \le 0,\ \ j = \overline{r+1, N};\ k \in K_j \\ \delta_l^k = \delta_m^k = \cdots = \delta_n^k \end{cases}$$

де l, m, ..., n - залежні стану, відповідні одній ТФЕ (на графі подій однієї ТФЕ може відповідати кілька вершин, природно, в них повинні прийматися однакові рішення) або різними ТФЕ, які повинні виконуватися однаковими способами, δ_j^k - булева змінна, що приймає значення 0 або 1; M - досить велике число.

3.3 Розробка алгоритму оптимізації алгоритму функціонування людино-машинних систем

Найбільш проста процедура такого рішення, орієнтована на наявний метод вирішення детермінованою завдання виглядає наступним чином.

1. Пошук оптимального рішення на детермінованою моделі АФ (рішення задачі без урахування імовірнісного обмеження на час виконання АФ, а з обмеженням на математичне очікування часу виконання АФ):

$$\sum_{i=r+1}^{N} \sum_{k \in k_i} \overline{t_i^k} \le T_0 \qquad (3.3)$$

2. Визначення функції розподілу часу виконання АФ і ймовірності своєчасного виконання АФ для отриманого рішення і, якщо вона відповідає умові - зупиниться (отримано оптимальне рішення), якщо немає - коригування ОДР (образно кажучи - знаходження «кордону відходу»).

3. Рішення детермінованою завдання в новій ОДР.

Кордон відходу можно можна знайти наступним чином:

$$T_{\text{ро}} = \bar{T}(x_1) + T_0 - \Phi_{x_1}^{-1}(\theta_0) \qquad (3.4)$$

Разом з тим, в розробленому вигляді не всі можливості розглянутого походу вичерпані. Це пов'язано з прийнятим припущення про незмінність функції розподілу часу реалізації АФ для рішень, отриманих на етапі 1 і етапі 3. Таке припущення може викликати або недотримання , образно кажучи, «недоліт» при визначенні $T_{\text{ро}}$; або при дотриманні виявлення рішення, що не гарантує максимальну ймовірність безпомилкового виконання АФ («переліт»).

Зняти це припущення можна за допомогою послідовного уточнення кордону функціонального обмеження, коли кожен раз після знаходження рішення детермінованою завдання досліджується функція розподілу часу виконання АФ і знаходиться «кордон відходу» і так до отримання гарантованого із заданою точністю результату. Така послідовна корекція «кордону відходу» ускладнює пошук оптимуму, але дозволяє виявляти додаткові резерви підвищення безпомилковості функціонування ЕТС. З міркувань практики для скорочення кількості ітерацій необхідно передбачити завершення процедури не тільки в разі отримання оптимального рішення, але і в разі отримання допустимого рішення зі значенням цільової функції, що відрізняється від максимально можливого при заданих обмеженнях не більше, ніж на деяку певну величину ε. Таким чином, гарантується здобуття оптимального рішення.

Алгоритм можна представити у вигляді схеми(рис.3.2), яка демонструє які дії повторюються, число умов та циклів, критерій зупинки роботи та для полегшення створення інформаційної системи, при написанні основного коду програми.

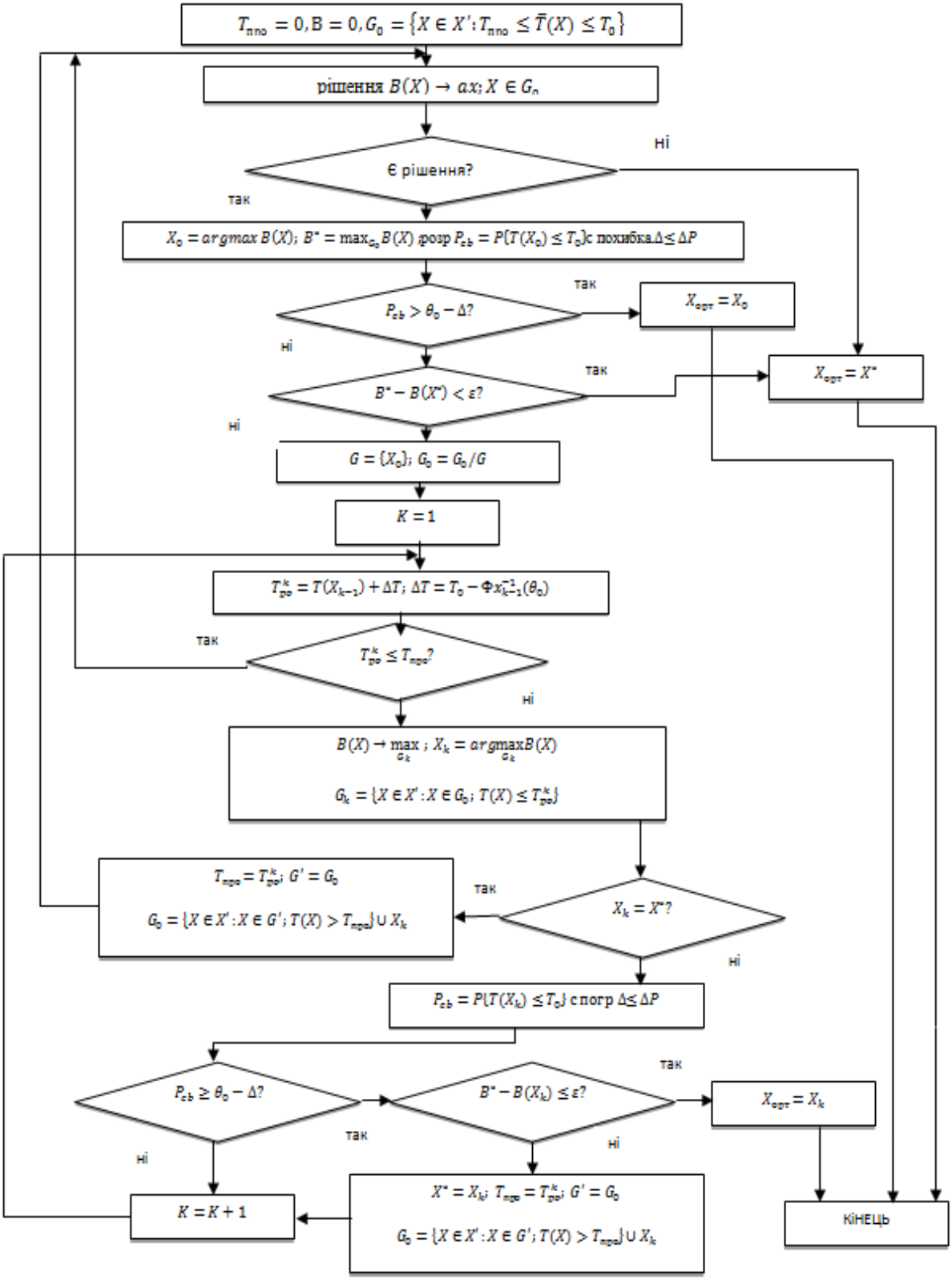


Рисунок 3.2 – Загальна схема алгоритму оптимізації

3.4 Висновки

— Оптимізація алгоритму діяльності при вірогідносних обмеженнях на час зручно проводити з використанням опису діяльності у вигляді функціональних мереж.

— Задачу з обмеженням по вірогідності доцільно зводити до детермінованої задачі.

— Алгоритм оптимізації з обмеженням по вірогідності доцільно організовувати таким чином, щоб: вирішувати детерміновану задачу, оцінювати вірогідність своєчасного виконання, корегувати область допустимих рішень.

— Зручність такої процедури полягає до вирішення сукупності детермінованих задач, рішення яких на разі відомі.

4 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1 Розробка загальних вимог до інформаційної системи

4.1.1 Опис інформаційної системи

Інформаційна система оптимізації людино-машинних систем призначена для знаходження оптимальних способів виконання операцій діяльності операторів, яка здатна оброблювати будь-яку кількість операцій з будь-яким числом варіантів виконання.

4.1.2 Цільова аудиторія

Інформаційна система може використовуватися менеджерами виробничих організацій, тобто заводів, фабрик тощо, а також в навчальних цілях студентами та викладачами.

4.1.3 Сценарії взаємодії

Дана інформаційна система повинна працювати на більшості платформ. Всі користувачі мають однакові можливості щодо використання функціоналу програми.

При потраплянні користувача на робоче вікно він може:

1. Ввести необхідну кількість операцій АФ людино-машинних систем.
2. Задати кількість способів виконання операцій.
3. Задати параметри кожного способу виконання операцій.
4. Підтверджувати введення та редагування даних за допомогою кнопки «Сохранить».
5. Експортувати чи імпортувати дані.

Натиснувши на відповідну кнопку розрахунку, результати роботи мають бути відображені на наступній вкладці «Результати», де відображуються локальні розрахунки на кожній ітерації та остаточний результат.

Розробка вимог безпосередньо до процесу функціонування інформаційної системи наведена у додатку Г.

4.2 Реалізація інформаційної системи

4.2.1 Створення інтерфейсу

Інтерфейс користувача – різновид інтерфейсів, в якому одна сторона представлена людиною (користувачем), інша - машиною / пристроєм. Являє собою сукупність засобів і методів, за допомогою яких користувач взаємодіє з різними, найчастіше складними, машинами, пристроями та апаратурою.

Логотип – оригінальне графічне зображення найменування компанії або продукту, покликане виділити фірму серед конкурентів.

Так як програма працює з ймовірнісними величинами, було вирішено зробити логотипом саме гральний куб червоного кольору.

Дана програма має класичні кольори елементів інтерфейсу: класичний колір фону та кнопок – сірий, який характерний для діалогових вікон операційної системи Windows. При запуску відкривається стартове вікно, в якому користувачу надається інформація про даний продукт, його задачі, а по натисканню кнопки відбувається перехід до робочого вікна, який розділений на дві вкладки: на першій користувач вводить та корегує свої вхідні дані, експортує чи імпортує їх, а на другій вкладці, яка називається «Результати» – відображено результати роботи на кожній ітерації.

Робоче вікно містить 2 основних поля для введення даних:

- Операції.
- Способи.

За допомогою спеціальних кнопок відбувається додавання чи видалення виділеного в цей момент способу чи операції. Тому можна виділити наступні можливості:

- При видаленні операції вцілому – видаляються всі способи її виконання.
- При додаванні операції за замовчуванням вона має 2 способи виконання, які потім можна редагувати.
- Для задавання параметрів необхідно внести їх у відповідні поля вводу, для підтвердження натиснути кнопку «Сохранить».

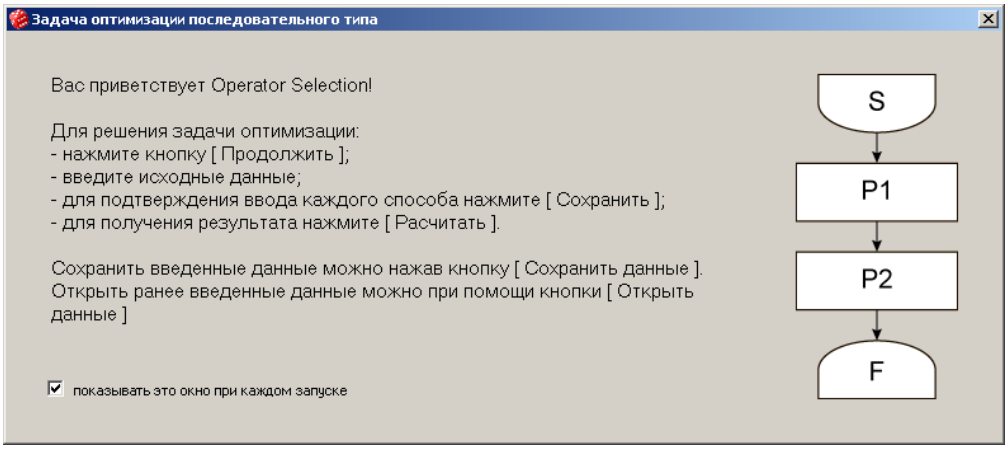


Рисунок 4.1 – Стартове вікно

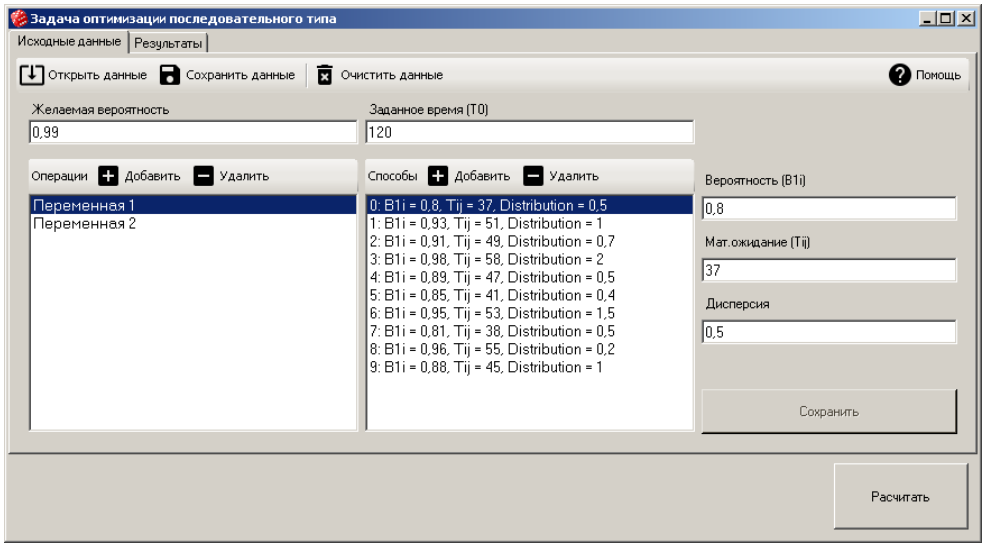


Рисунок 4.2 – Робоче вікно

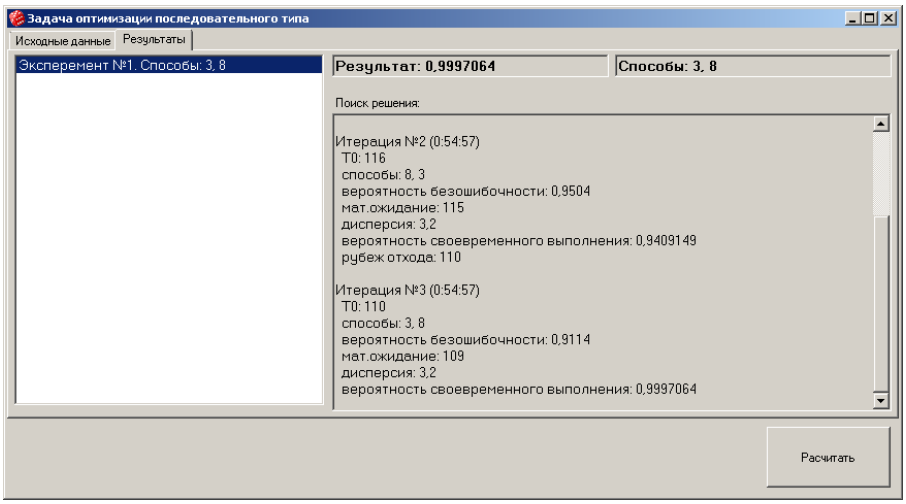


Рисунок 4.3 – Результати

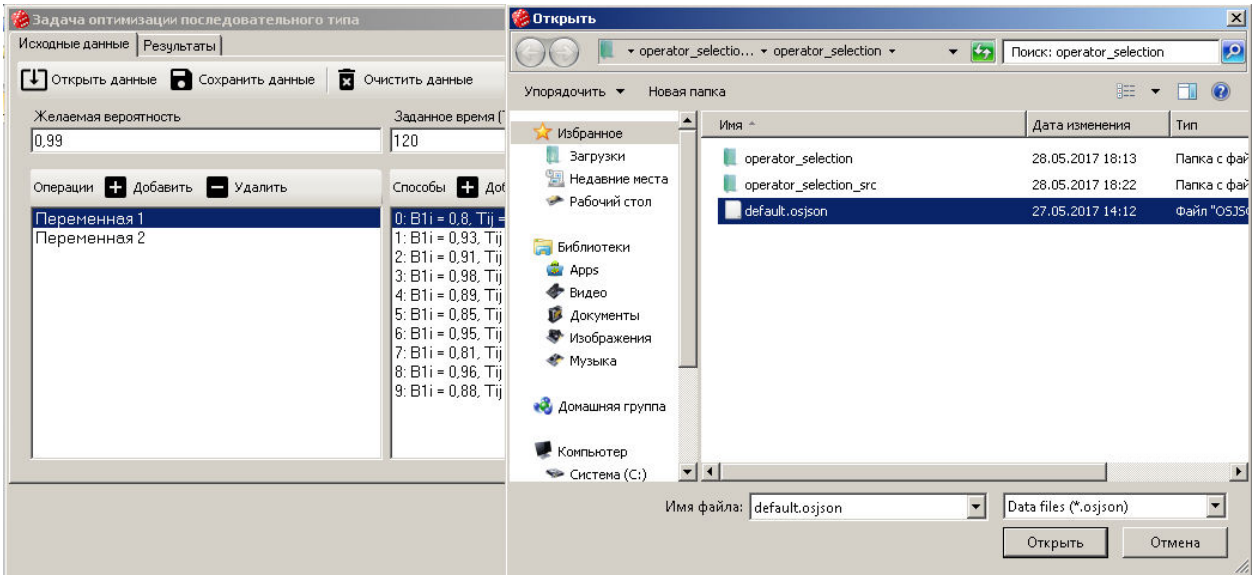


Рисунок 4.4 – Експорт, імпорт даних

4.2.2 Розробка технології для реалізації алгоритму оптимізації

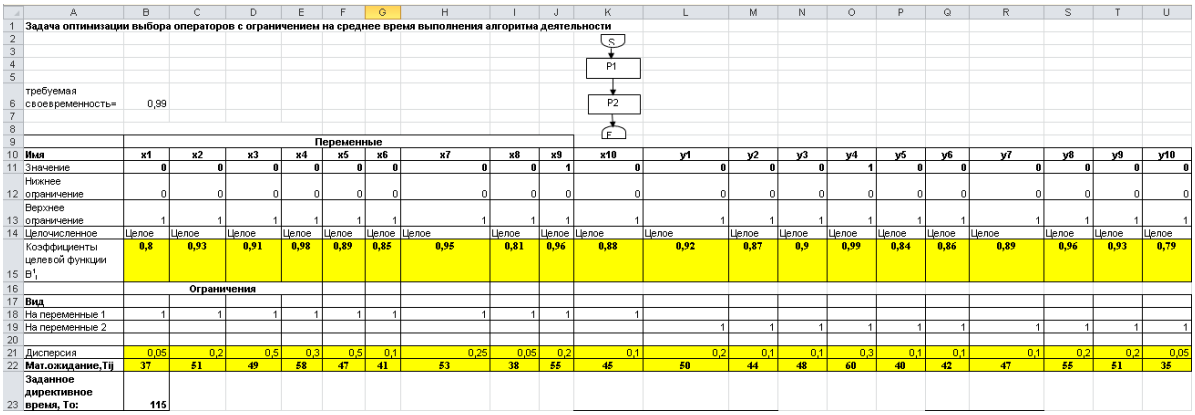
Для розробки інформаційної системи використаємо Microsoft VisualStudio 2012, яка дозволяє створювати програми будь-якого типу, створювати власний інтерфейс та запрограмувати алгоритми різної складності.

Для збереження даних було обрано технологію JSON – це текстовий формат обміну даними між комп'ютерами. Він базується на тексті, що може бути прочитаним людиною. Формат дозволяє описувати об'єкти та інші структури даних. Перевагою JSON перед XML є те, що він дозволяє складні структури в атрибутах, що займає менше місця і прямо інтерпретується за допомогою JavaScript в об'єкти.

Програмна реалізація алгоритму відбувалась мовою програмування C# на базі рішення задачі за допомогою MS Excel.

Розробка програми відбувалась на основі рішення задачі оптимізації людино-машинних систем з двома операціями по 10 способів виконання кожної з них на прикладі у MS Excel. Задано бажану своєчасність – 0,99.

На початку роботи відбувається введення даних, в даному випадку 2 операції по 10 способів виконання, кожен з яких має математичне сподівання, ймовірність безпомилковості, середнє відхилення. Також початковий час та бажана ймовірність своєчасності.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	**Задача оптимизации выбора операторов с ограничением на среднее время выполнения алгоритма деятельности**																				
2											S										
3																					
4											P1										
5																					
6	требуемая своевременность=	0,99									P2										
7																					
8											F										
9		Переменные																			
10	**Имя**	**x1**	**x2**	**x3**	**x4**	**x5**	**x6**	**x7**	**x8**	**x9**	**x10**	**y1**	**y2**	**y3**	**y4**	**y5**	**y6**	**y7**	**y8**	**y9**	**y10**
11	Значение	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
12	Нижнее ограничение	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	Верхнее ограничение	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
14	Целочисленное	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое
15	Коэффициенты целевой функции B',	**0,8**	**0,93**	**0,91**	**0,98**	**0,89**	**0,85**	**0,95**	**0,81**	**0,96**	**0,88**	**0,92**	**0,87**	**0,9**	**0,99**	**0,84**	**0,86**	**0,89**	**0,96**	**0,93**	**0,79**
16		Ограничения																			
17	**Вид**																				
18	На переменные 1	1	1	1	1	1	1	1	1	1	1										
19	На переменные 2											1	1	1	1	1	1	1	1	1	1
20																					
21	Дисперсия	0,05	0,2	0,5	0,3	0,5	0,1	0,25	0,05	0,2	0,1	0,2	0,1	0,1	0,3	0,1	0,1	0,1	0,2	0,2	0,05
22	**Мат.ожидание,Tij**	**37**	**51**	**49**	**58**	**47**	**41**	**53**	**38**	**55**	**45**	**50**	**44**	**48**	**60**	**40**	**42**	**47**	**55**	**51**	**35**
23	**Заданное директивное время, To:**	**115**																			

Рисунок 4.5 – Вхідні дані

На першій ітерації необхідно підібрати оптимальні способи, при яких ймовірність безпомилковості буде максимальною, а час виконання не вище заданого. У MS Excel перебір даних виконується за допомогою інструменту «Пошук рішення», а в програмі – за допомогою спеціальних циклів, які перебираюсь всі варіанти та зупиняються, при задоволені умов.

Після знаходження оптимальних способів виконується розрахунок ймовірності своєчасного виконання. Для реалізації було підключено бібліотеки, які містять спеціальні математичні формули, реалізовані у вигляді методів класів. У MS Excel знаходження ймовірності своєчасного виконання потребує спеціальної функції НОРМ.РАСП, параметри якої є заданий час виконання, математичне сподівання, середнє відхилення та має наступний вигляд.

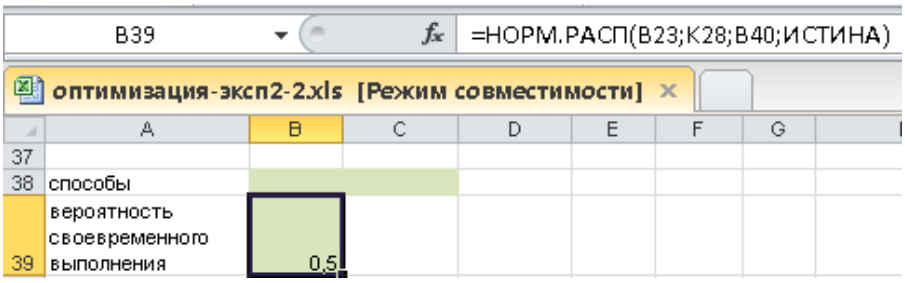


Рисунок 4.6 – Запис формули у MS Excel

У програмі викликається спеціальний метод StatisticFormula.NormalDistribution(), атрибутами якого є заданий час виконання, математичне сподівання, середнє відхилення. В цьому методі розраховується формула нормального розподілу у інтегральній формі та дозволяє отримати ймовірність по заданим величинам.

Після цього відбувається порівняння отриманого результату з бажаною ймовірністю та виконуються наступні дії:

- Отримана ймовірність більше або дорівнює заданої, тому рішення оптимальне, виведення результату користувачу
- Отримана ймовірність менше заданої, розрахунок «кордону відходу»

Для розрахунку кордону відходу використовується формула наведена в розділі 3, а в Excel – з використанням функції оберненого нормального розподілу НОРМ.ОБР, загальний синтаксис якої має наступний вигляд.

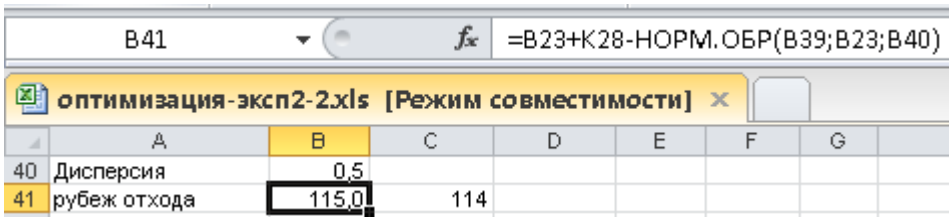


Рисунок 4.7 – Запис формули кордону відходу у MS Excel

У програмі дана функція реалізована за допомогою спеціального методу StatisticFormula.InverseDistribution(), аргументами якої є ймовірність своєчасного виконання, заданий середній час виконання, середнє відхилення. Кордон відходу уявно зміщує область допустимих рішень для вирішення задачі на своєчасність.

Після його знаходження ми замінюємо заданий середній час виконання на значення кордону відходу.

На початку другої ітерації програма виконує новий пошук рішення, з обмеженням у часі вже на значення кордону відходу. Після отримання оптимальних значень відбувається новий розрахунок ймовірності своєчасного виконання. Якщо на даному етапі знову не досягнуто значення ймовірності своєчасності, то починається наступна ітерація.

Результат выбора:										
		Вероятность безошибочного функционирования:								**0,9408**
дисперсия:	0,5									
вероятность своевременного выполн	0,97725									
рубеж отхода	112,0	112								

Рисунок 4.14 – Результати обчислень другої ітерації

Тому програма виконує вище описані дії доки отримане рішення ймовірності своєчасного виконання буде більше чи дорівнювати бажаному.

Під час роботи програми всі проміжні значення розрахунків кожної ітерації виводяться на вкладці «Результати».

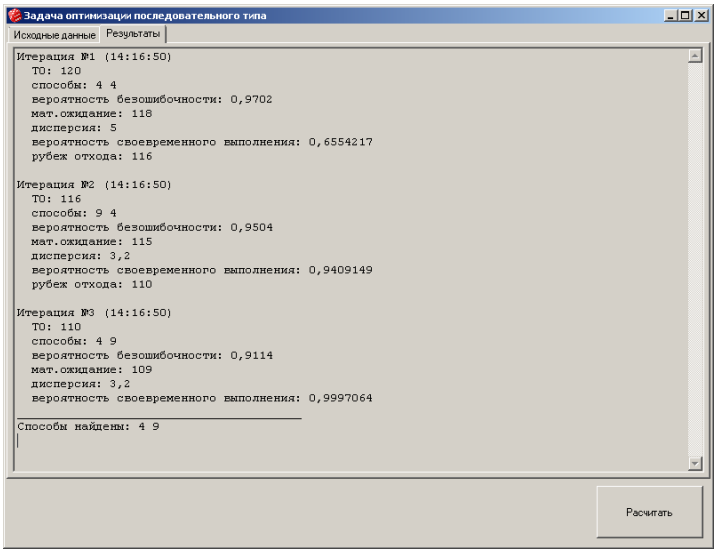


Рисунок 4.8 – Результат роботи програми

Тестування інформаційної системи та комп’ютерної комп’ютерні експерименти наведені у додатку Д.

ВИСНОВКИ

Дуже стрімкий розвиток технологій все частіше призводить до ускладнення роботи операторів людино-машинних систем. Прчинами є: збільшення числа керованих об’єктів, кількості інформації на основі якої необхідно приймати рішення, збільшення відстані до об’єктів, а тому витрачається певний час на комунікації.

Основна проблема існуючих алгоритмів оптимізації – вирішення задачі з обмеженням на середній час виконання. Дослідження показали, що при даних умовах вже не можна розглядати час як постійну величину, і тому вони дають помилковий результат. Був запропонований та реалізований алгоритм оптимізації з урахуванням ймовірнісного характеру часу.

Було проведено серію комп’ютерних експериментів, метою яких була перевірка роботи алгоритму при різних вхідних даних, велику увагу яких мала дисперсія. Перевірка відбувалась на основі математичних розрахунків та наведення відповідних графіків. В результаті було доведено оптимальність та ефективність роботи алгоритму оптимізації. Тому розроблений продукт дозволяє вирішити задачу будь-якої складності гарантуючи своєчасність.

СПИСОК ВИКОРИСТАНОЇ ЛТЕРАТУРИ

1. Надежность и эффективность комплексных систем «человек—техника». Ч. 3. Под ред. А. И. Губинского. Л., ЛДНТП, 1970.
2. Ахьюджа Х. Сетевые методы управления в проектировании и производстве.Пер. с англ. /Под. ред. В. Н. Калашникова М.:Мир, 1979. – 638 с.
3. Губинский А.И. Надежность и качество функционирования эргатических систем. Л.: Наука, 1982. 270с.
4. Поспелов Д.А. Логико-лингвистические модели в системах управления. – М.: Энергоиздат, 1981. 232 с.
5. Котов В.Е. Сети Петри. – М.: Наука, 1984. – 160 с.
6. Филлипс Д., Гарсиа-Диас А. Методы анализа сетей / Д. Филлипс, А. Гарсиа-Диас; пер. с англ. / Под ред. Б.Г. Сушкова. – М.: Мир, 1984. – 496
7. Суходольский Г.В. Инженерно психологический анализ и синтез профессиональной деятельности. – Автореф. дисс. На соиск. ученой степени докт. психол. наук. – Л.: 1982. – 40 с.
8. Суходольский Г.В. Структурно-алгоритмический анализ и синтез деятельности. Л.: Изд-во ЛГУ, 1976. – 120 с.
9. Кузин Л.Т. Основы кибернетики. Т.2. – М.: Энергия, 1979, 584 с.
10. Майн Х., Осаки С. Марковские процессы принятия решений. М.: Наука, 1977. – 176 с.
11. Сильвестров Д.С. Полумарковские процессы с дискретным множеством состояний (основы расчета функциональных надежностных характеристик стохастических систем). – М.: Сов.радио, 1980. – 272 с.
12. Филлипс Д., Гарсиа-Диас А. Методы анализа сетей / Д. Филлипс, А. Гарсиа-Диас; пер. с англ. / Под ред. Б.Г. Сушкова. – М.: Мир, 1984. – 496 с.
13. Зараковский Г.М. Психофизиологический анализ трудовой деятельности. – М.: Наука, 1966. – 112 с.
14. Зараковский Г.М.. Павлов В.В. Закономерности функционирования эргатических систем. – М.: Радио и связь, 1987. – 232 с.

15. Попович П.Р., Губинский А.И., Колесников Г.М. Эргономическое обеспечение деятельности космонавтов. – М.: Машиностроение, 1985. – 272 с.

16. Губинский А.И., Гриф М.Г., Цой Е.Б. О некоторых алгоритмах оптимизации систем «человек-техника» // Применение ЭВМ в оптимальном планировании и проектировании. – Новосибирск: НГУ, 1981, - С. 148-154.

17. Ашеров А.Т. Обеспечение эффективности функционирования АСУ на этапе проектирования // Экономика и математические методы, 1975, №1. – С.178-179.

18. Ашеров А.Т., Исаенко С.Г., Лавров Е.А. Разработка моделей для оценки надежности системы управления гибкими производственными системами (СУ ГПС). // III Конференция молодых ученых и специалистов приборостроительной промышленности. – М., 1986. – С. 64-66.

19. Ашеров А.Т., Лавров Е.А., Исаенко С.Г. Подход к оценке надежности системы управления ГПС. // Расчет и управление надежностью больших механических систем. Информационные материалы VI Всесорюзной школы. – Тернополь-Свердловск, 1986. – С. 69-71.

20. Ашеров А.Т., Лавров Е.А., Исаенко С.Г. Метод формализованного описания процесса функционирования элементов ГПС для оценки показателей надежности и типизации АСУ ГПС. Тезисы докладов Всесоюзного научно-технического семинара (г. Харьков, ноябрь 1987 г.) Часть 1. – М., 1987. –С 14-15.

21. Ашеров А.Т., Лавров Е.А., Исаенко С.Г., Морарь Ю.И. Оценка своевременности реализации дискретных процессов в человеко-машинных системах управления и проектирования // Тезисы докладов Всесоюзной научно-технической конференции «Проблемы стандартизации и повышения технического уровня автоматизированных систем различного назначения». – Минск, 1988. – С.53-54.

22. Лавров Е.А., Ашеров А.Т., Бланк Н.М. Оценка времени выполнения задания системой «человек-машина» при различных законах распределения времени выполнения отдельных операций // III всесоюзная научно-техническая конференция «Методы синтеза типовых модульных систем обработки данных». Кишинев. 18-21 октября 1988. тезисы докладов. –М., 1988. –С. 59-60.

23. Лавров Е.А., Волонцевич А.А. Пути оценки надежности программного обеспечения банка эргономических данных // Всесоюзная конференция «Автоматизация эргономических исследований, проектирования и испытаний систем «человек-машина». – Л., 1984. – С27-29.

24. Информационно-управляющие человеко-машинные системы: Исследование, проектирование, испытания: Справочник/ Адаменко А.Н., Ашеров А.Т., Лавров Е.А. и др.. под общ. ред. Губинского А.И. и Евграфова Е.Г.- М., Машиностроение, 1993. – 528с.

25. Избачков Ю.С. Информационные системы: учебник [Текст]: – 2-е изд. – СПб: Питер, 2008. – 656 с.

ДОДАТОК А
АНАЛІЗ МЕТОДІВ ОПИСАННЯ ТА ОЦІНКИ ЛЮДИНО-МАШИННИХ СИСТЕМ

Формальні граматики і відповідні їм автомати (машини Тюрінга, лінійно-обмежені автомати, кінцеві автомати) дозволяють описувати процес функціонування послідовної алгоритмічної системи як процес зміни станів в залежності від зовнішніх впливів.

Мережі Петрі [6] призначені для опису систем з паралельно функціонують і асинхронно взаємодіючими елементами. Даний апарат призначений в основному для вирішення завдань "якісного" характеру при проектуванні дискретних систем з паралелізмом: виявлення аварійних ситуацій і потенційно вузьких місць (взаємне блокування процесів), спрощення системи без порушення її загального функціонування і т.п.

Алгоритмічні моделі (граф-схеми алгоритмів, логічні схеми алгоритмів, алгоритмічні алгебри Глушкова, схеми Янова) [2,7] дозволяють описувати тільки послідовні алгоритмічні системи і принципово не придатні для опису систем з паралельно функціонуючими елементами.

Структурно-алгоритмічний метод аналізу і синтезу діяльності професора Суходольського Г.В. [8,9] дозволяє синтезувати узагальнену модель індивідуальної і колективної діяльності у вигляді абстрактного графа діяльності, що має вигляд стохастичного мультиграфа. Процедура синтезу передбачає елементи формалізації шляхом представлення графів кожної реалізації в матричної формі, що дозволяє здійснювати раціоналізацію панелі пульта управління і вирішувати інші завдання ергономічного проектування. До обмежень методу відносяться орієнтація на формалізацію тільки дій людини, а не всього процесу функціонування системи, а також відсутність можливості опису паралельних процесів.

Методи теорії ситуаційного управління [10,11] призначені для опису процесів функціонування складних систем на семантичному рівні, дозволяють описувати також процеси прийняття рішень.

Основним недоліком всіх перерахованих методів є неможливість кількісної оцінки показників функціонування ерготехнічних систем(ЕТС).

У класі алгебраїчних систем найбільш придатні для опису алгоритмів функціонування ЛМС марковські і напівмарковські процеси [12,13]. На відміну від попередніх, ці методи дозволяють знаходити кількісні оцінки ЛМС, проте обмежені в описі логіки алгоритму функціонування: не дозволяють описувати системи з паралельно функціонуючими елементами, циклічні процеси з обмеженим числом циклів.

Мовно-алгебраїчні системи містять властивість описовості та оцінки модельованих процесів. Загальним для мереж передування (СП), МКП, PERT і GERT [2,14] є те, що описуваний процес представляється у вигляді графа подій: послідовність виконуваних робіт представляється дугами орграфа, а події початку і закінчення роботи – його вершинами. Основною кількісною характеристикою на мережах СП, МКП, PERT і GERT є час виконання описуваного процесу, причому в СП і МКП час є детермінованою величиною, а в мережах PERT і GERT – випадковою, і оцінюється математичне очікування і дисперсія часу виконання процесу. Мережі МКП і PERT мають найбільш слабку логіку взаємозв'язку виконання окремих операцій: що виходять і входять в вершини дуги пов'язані логічною функцією "І". Мережа GERT має великі логічні можливості і дозволяє реалізувати на вході функції алгебри логіки "І", "АБО-вкл." і "АБО-викл.", а не по виходу "АБО-викл." [2]. У мережах МКП і PERT не допускаються цикли, в СП можливі повторювані групи операцій, в GERT допускаються цикли, петлі і неодноразове виконання операцій. Очевидно, що найбільшими можливостями з перерахованих мережевих методів має GERT, проте його використання обмежується оцінкою тільки тимчасових витрат на реалізацію процесу функціонування значної обчислювальної складності.

Операційно-психологічний метод професора Зараковського Г.М. [15,16] призначений для оцінки завантаження людини за показниками операційної напруженості. Даний метод орієнтований на оцінку часових показників процесу функціонування і має слабо розвинений апарат для опису логіки АФ.

Узагальнений структурний метод (УСМ), що лежить в основі функціонально-структурної теорії (ФСТ) ЕТС, як апарату опису АФ використовує функціональну мережу (ФМ) [17]. ФС дозволяє описувати не тільки процеси виконання, але і процеси прийняття рішень. На відміну від мережевих методів в апараті ФМ процес функціонування може представлятися як графом подій, так і графом робіт: вершин орграфа відповідають операції, а дугам - відносини між операціями.

Для опису АФ в апараті ФМ використовуються типові функціональні одиниці (ТФО). До їх числа відносяться функціонери, які відповідають реальним операціям і діям, необхідні для встановлення логіко-функціональних зв'язків між ними.

Введення в алфавіт апарату ФМ операцій контролю функціонування і працездатності дозволяє, на відміну від інших мережевих методів, моделювати процеси втрати стійкості процесу функціонування через помилки і відмов. ФМ можуть описувати циклічні процеси, петлі, як з обмеженням на число повторень, так і без них. Логіка виконання паралельних операцій ширше, ніж в GERT, і дозволяє реалізувати функції алгебри логіки "І", "АБО-вкл.", "АБО-викл." як по входу, так і по виходу паралельної структури. Для проведення кількісної оцінки показників якості і надійності функціонування ЕТС в апараті ФМ отриманий набір аналітичних виразів для найбільш часто зустрічаються структур, так званих типових функціональних структур (ТФС), що спрощує процедуру оцінки ФМ. Номенклатура описуваних показників в апараті ФМ включає цілий ряд системних показників, що характеризують, крім тимчасових, також надійнісні і вартісні показники.

Із аналізу зрозуміло, що найбільшою мірою вимоги опису будь-якого АФ в СОІУ та можливості оцінки показників його ефективності, якості і надійності відповідає УСМ.

Використання УСМ дозволяє не тільки отримувати оцінки варіантів АФ ЕТС, а й у разі великої їх кількості вирішувати задачу оптимального вибору. На даний момент у роботах А.І. Губинського, Є.Б. Цоя, М.Г. Грифа [3,18] поставлені і вирішені однокритеріальні завдання, в яких екстремується один з показників, ймовірність безпомилкового виконання АФ (максимізує), середній час виконання АФ (мінімізується), а на два інших накладається обмеження . У роботах Ашерова А.Т. [19] і Ільченко Є.В. [20] вперше поставлена і вирішена одна з

багатокритеріальних задач оптимізації ЕТС стосовно розподілу функцій між людиною і технікою в інформаційних технологіях (у разі явно заданої множини альтернатив і наявності оцінок за всіма критеріальними функціями для всіх варіантів). У роботах Лаврова Є.А. розроблені моделі для оцінки якості виконання АФ АТК при різних варіантах організації, а також поставлено і вирішено комплекс оптимізаційних завдань ергономічного проектування [21,22,23,24,25].

Загальною метою розрахунків ефективності, якості і надійності ергатичних систем (систем «людина-техніка») є оцінка рівня ефективності, якості і надійності системи на основі обчислення кількісних значень показників, встановлених або обраних для даного типу системи.

Попередньо обчислені значення показників ефективності, якості і надійності ергатичних системи можуть застосовуватися:

- для перевірки відповідності рівня ефективності, якості і надійності проектованої системи встановленим нормам;
- для оцінки впливу ергатичних ланок на ефективність, якість і надійність системи в цілому;
- для оцінки доцільності введення організаційно-технічних заходів щодо підвищення ефективності, якості, надійності ергатичних ланок або системи в цілому;
- для обгрунтованого вибору організаційно-технічної структури ергатичній системи і інших задач ергомічного проектування;
- як одна зі складових при обчисленні показників ефективності, якості і надійності системи, в яку входить розглянута ергатичній система.

Розроблені в функціонально-структурної теорії професора Губинського А.І. [17] методики оцінки показників ефективності, якості і надійності є універсальними для широкого класу ергатичних систем, що допускають опис їх функціонування у вигляді дискретних кінцевих алгоритмів, відповідних схем алгоритмів першого і другого роду. При цьому єдиним алгоритмом описується функціонування ергатичних системи в цілому, тобто діяльність фахівця або групи фахівців і операції, що виконуються технічними засобами (ЕОМ і ін.).

Математичними моделями цього класу систем є стохастичні мережі з петлями і циклами (напівмарковські ланцюги з довільними законами розподілу часу між переходами).

Подання алгоритмів здійснюється у вигляді сукупності спеціально введених типових функціональних одиниць (ТФО), що складають основу алгоритмічної мови типових структур (АМТС). Обчислення показників ефективності, якості і надійності проводиться за допомогою формул, наведених для найбільш часто зустрічаються комбінацій.

Номенклатура визначаються показників ефективності, якості і надійності задається в технічному завданні на систему, в керівних документах або вибирається на розсуд розробника (дослідника, експлуатальника) ергатичній системи.

Показники якості функціонування (час виконання функції і ін.) і показники функціональної надійності (ймовірність безпомилкового виконання функції, ймовірність своєчасного виконання функції) обчислюються для кожної функції (цілі, завдання, операції) ергатичній системи. Показники ефективності (дохід від виконання функції і ін.) -як для окремої функції, так і для ергатичній системи в цілому.

При оцінці показників ефективності, якості і надійності ергатичних систем відповідно до методиками справжніх «Рекомендацій» доводиться мати справу з трьома групами показників:

Група результуючих показників, які є результатом розрахунку і служать оціночними для ергатичній системи в цілому або окремої функції:

π – ймовірність безпомилкового виконання функції;

θ – ймовірність своєчасного виконання функції;

φ – ймовірність правильного безпомилкового і своєчасного виконання функції;

T – час виконання функції;

U – дохід, отриманий від виконання функції;

Якщо при розрахунку результуючих показників використовуються випадкові величини (випадкове число операцій, циклів, що змінюються значення вихідних характеристик і т.п.), як наслідок, перераховані підсумкові показники стають самі

випадковими величинами, необхідно обчислювати їх математичні очікування і дисперсії.

Група вихідних показників, які відносяться до конкретної 1-й операції, що входить до складу функції, і на основі яких виробляється обчислення результуючих показників, визначаються експериментально.

β_i – можливість безпомилкового (або помилкового) виконання і-й операції;

T_i – час виконання і-й операції;

r_i – дохід (витрата), одержуваний від виконання і-й операції.

Якщо вихідні показники є випадковими величинами (від впливу зовнішніх факторів на людину або але інших причин), то в розрахунок необхідно вводити їх математичні очікування і дисперсії.

Група проміжних показників, що з'являється в процесі виконання розрахунків показників для типових груп операцій (типових функціональних структур) по редукції [1,3,18]:

β_s- еквівалентна ймовірність безпомилкового (або помилкового) виконання, відповідна типовий функціональною структурою (ТФС);

T_s- еквівалентне час виконання типової функціональної структури;

U_s- еквівалентний дохід, відповідний типовий функціональною структурою.

Для ТФС, що мають цикли (повтори однієї або декількох операцій, що входять в типову функціональну структуру), при розрахунках доводиться вводити ряд їх модифікацій:

а) накопичені ймовірності – для випадку, коли число циклів фіксоване: $\beta_s(l)$, $T_s(l)$, $U_s(l)$;

б) фінальні ймовірності (випадки «а» при $l = \infty$: $\beta_s(\infty)$, $T_s(\infty)$, $U_s(\infty)$

в) осереднення від циклів ймовірності – для випадку, коли граничне число циклів обмежене (так само від m), а закінчення повторення відбувається при першому ж виході, визнаному вдалим: M $[\beta_s(m)]$ і D $[\beta_s(m)]$, M $[T_s(m)]$, і D $[T_s(m)]$, M $[U_s(m)]$ і D $[U_s(m)]$.

На ранніх стадіях проектування ергатичних систем інформація про деякі характеристики алгоритмів функціонування може бути недостатньо точна. У зв'язку з цим деякі характеристики приймаються спочатку наближеними (орієнтовними),

що знижує точність результатів розрахунку. Звідси допустимі і певні спрощення в самій методиці розрахунку (орієнтовна методика розрахунку показників ефективності, якості і надійності). На наступних стадіях після уточнення необхідних характеристик, щоб не втратити точність і самі математичні моделі, які використовуються для розрахунку показників, повинні бути більш адекватними реальному процесу функціонування (уточнена методика розрахунку показників ефективності, якості і надійності).

ДОДАТОК Б
ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ ОПИСАННЯ ТА ОЦІНКИ ПРОЦЕСУ ФУНКЦІОНУВАННЯ ЛЮДИНО-МАШИННИХ СИСТЕМ

Таблиця Б.1 – порівняння методів оптимізації

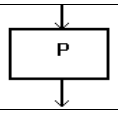
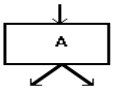
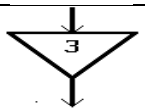
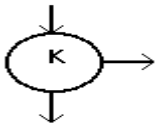
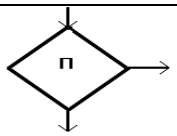
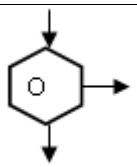
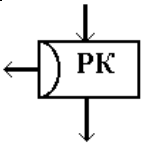
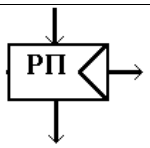
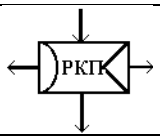
Метод характеристика	Граф-схеми алгоритмів	PERT	GERT	Імітаційні моделі	Функціональні мережі	
					Граф-схеми	Алгебраїчні ФМ
Загальна направленість методу	Опис алгоритмів	Опис і оцінка комплексів робіт в промисловості і будівництві	Опис довільних процесів	Опис та оцінка надійності процесів	Універсальний підхід на описання будь-яких процесів. Опис та оцінка ефективності функціонування ЕТС	Опис та оцінка недефектних ЛМС
Засоби автоматизації	Ручні	Різні ЕОМ	ЕОМ	ЕОМ	ПК	ПК MS DOC
Форма представлення	Граф робіт	Граф подій (Вершини – події, дуги - роботи)	Граф подій (Вершини – події, дуги - роботи)	Граф подій (Вершини – події, дуги - роботи)	Граф робіт-подій(верішини роботи чи події)	Граф подій (Вершини – події, дуги - роботи)
Зміст символів алфавіту	Дві роботи: оператор та лог. умови	Тільки одновихідні роботи	Одновихідні та двовихідні роботи	Тільки одновихідні роботи	15 функціонерів 14 композицій	3 оператора робіт та 2 умови
Можливість представлення послідовних робіт	Так	Так	Так	Так	Так	Так
Паралельних робіт на вході І	Ні	Так	Так	Так	Так	Так
Паралельних робіт на виході І	Ні	Так	Так	Так	Так	Так
Цикли	Так	Ні	Так	Ні	Так	Так
Цикли допрацювання	Ні	Ні	Так	Ні	Так	Так

Продовження таблиці Б.1

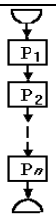
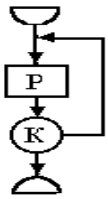
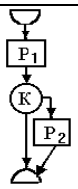
Метод характеристика	Граф-схеми алгоритмів	PERT	GERT	Імітаційні моделі	Функціональні мережі	
					Граф-схеми	ФМ алгебраїзму
Можливість врахування збоїв, помилок	Ні	Ні	Ні	Ні	Так	Так
Можливість врахування непомічених збоїв, помилок	Ні	Ні	Ні	Ні	Так	Так
Основні недоліки	Слабкі можлтвості	Неврахування зупинки роботи через збої, помилки	Неврахування зупинки роботи через збої, помилки	Неврахування зупинки роботи через збої, помилки	Висока складність методу	Висока складність методу
Загальна оцінка можливостей	Примітивна	Слабка	Гарна	Слабка	Найкраща серед існуючих	Висока
Рівень	Нуль	Тільки часові характеристики	Часові з ймовірнісними величинами	Часові з показником своєчасності	Часові та показники бездефектності, врахування збоїв, помилок	Часові та показники бездефектності, врахування збоїв, помилок
Загальна оцінка методу	Придатний тільки для опису структур	Придатний для оцінки часових характеристик з обмеженою логікою	Придатний для оцінки часових характеристик з розширеною логікою	Придатний для оцінки часових характеристик та своєчасності	Придатний для оцінки часових та надійнісних характеристик будь-яких процесів з урахуванням збоїв, помилок	Придатний для оцінки часових та надійнісних характеристик будь-яких процесів з урахуванням збоїв, помилок

ДОДАТОК В
ТИПОВІ ФУНКЦІОНАЛЬНІ ОДИНИЦІ ТА СТРУКТУРИ

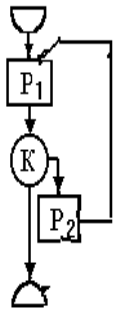
Таблиця В.1 Типові функціональні одиниці

Вид	Назва	Позначення
Основні	Робітник	
	Логічний	
	Затримка	
	Контроль	
Допоміжні	Діагностичний контроль	
	Організаційний контроль	
	Робітник з керуванням результатів	
	Робітник з діагностикою техніки	
	Робітник з діагностикою техніки і керування результатів	

Таблиця В.2 – Типові функціональні структури

Зміст ТФС	Схема	Показник	Формула
Послідовне виконання операцій *RR*		Ймовірність безпомилкового виконання операцій	$B = \prod_{i=1}^{n} B_i$
		Математичне сподівання часу виконання операцій	$M(X) = \sum_{i=1}^{n} M(X_i)$ $X = \{T, W, C\}$
		Дисперсія часу виконання операцій	$D(X) = \sum_{i=1}^{n} D(X_i)$ $X = \{T, W, C\}$
Циклова функціональна схема "Робоча операція з контролем функціонування без обмеження на кількість циклів", *RK*		Ймовірність безпомилкового виконання операцій	$B = B^1 * K^{11} * \frac{1}{1-(B^1 * K^{10} + B^0 * K^{00})}$
		Математичне сподівання часу виконання операцій	$M(X) = (M(X_p) + M(X_к)) * M(L)$ $M(L) = \frac{1}{1-(B^1 * K^{10} + B^0 * K^{00})}$ $X = \{T, W, C\}$
		Дисперсія часу виконання операцій	$D(X) = D(L) * (M(X_p) + M(X_к))^2 +$ $(D(X_p) + D(X_к)) * M(L)$ $D(L) = \frac{B^1 * K^{10} + B^0 * K^{00}}{(1-(B^1 * K^{10} + B^0 * K^{00}))^2}$ $X = \{T, W, C\}$
Робоча операція з контролем функціонування і виправленням помилки без циклів, *RKR1*		Ймовірність безпомилкового виконання операцій	$B = B_1^1 * K^{11} +$ $(B_1^0 * K^{00} + B_1^1 * K^{10}) * B_2^1$
		Математичне сподівання часу виконання операцій	$M(X) = M(X_{p1}) + M(X_к) +$ $(B_1^0 * K^{00} + B_1^1 * K^{10}) * M(X_{p2})$ $X = \{T, W, C\}$
		Дисперсія часу виконання операцій	$D(X) = D(X_{p1}) + D(X_к) +$ $(B_1^0 * K^{00} + B_1^1 * K^{10}) * D(X_{p2}) +$ $(B_1^0 * K^{00} + B_1^1 * K^{10}) *$ $(B_1^1 * K^{11} + B_1^0 * K^{01}) * M^2(X_{p2})$

Продовження таблиці В.2

Циклова ФС «Робоча операція з контролем функціонування, виправленням і повторенням робочої операції без обмеження на кількість циклів», RKR		Ймовірність безпомилкового виконання операцій	$B = \frac{B_1^1 * K^{11}(1 - K^{00} * B_2^0)}{K^{01} + B_1^1 * B_2^1(K^{11} - K^{01})}$
		Математичне сподівання часу виконання операцій	$M(X) = M(X_{p1}) + M(X_k) + [M(X_{p1}) + M(X_{p2}) + M(X_k)] * * \frac{B_1^1 * K^{10} + B_1^0 * K^{00}}{1 - (B^1 K^{10} + B^0 * K^{00})}$, де $B^1 = B_1^1 * B_2^1; \quad B^0 = 1 - B^1$
		Дисперсія часу виконання операцій	$D(T) = D(X_{p1}) + D(X_k) + [D(X_{p1}) + D(X_{p2}) + D(X_k)] * * \frac{B_1^1 * K^{10} + B_1^0 * K^{00}}{1 - (B^1 K^{10} + B^0 * K^{00})} + + \frac{B_1^1 * K^{10} + B_1^0 * K^{00}}{(1 - (B^1 K^{10} + B^0 * K^{00}))^2} * * \frac{B_1^1 * K^{10} + B_1^0 * K^{00}}{1 - (B^1 K^{10} + B^0 * K^{00})} * * \frac{B_1^1 * K^{11} + B_1^0 * K^{01}}{(1 - (B^1 K^{10} + B^0 * K^{00}))^2} * * [M(X_{p1}) + M(X_{p2}) + M(X_k)]^2$ де $X = \{T, W, C\}$

ДОДАТОК Г
РОЗРОБКА ВИМОГ ДО ПРОЦЕСУ ФУНКЦІОНУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Побудова контекстної діаграми у нотації IDEF0. IDEF0 – Function Modeling – методологія функціонального моделювання і графічного описання процесів, призначена для формалізації і опису бізнес-процесів. Особливістю IDEF0 є її акцент на ієрархічне представлення об'єктів, що значно полегшує розуміння предметної області. В IDEF0 розглядаються логічні зв'язки між роботами, а не послідовність їх виконання в часі (WorkFlow).

Так само відображаються всі сигнали управління. Така модель є однією з найпрогресивніших моделей і використовується в організації бізнес проектів і проектів, що базуються на моделюванні всіх процесів як адміністративних, так і організаційних.

Контекстна діаграма має рівень A0. Це найвищий рівень абстракції для даного завдання. Основною точкою зору була взята точка зору замовника, а також було визначено мету – аналіз процесу побудови програмного засобу.

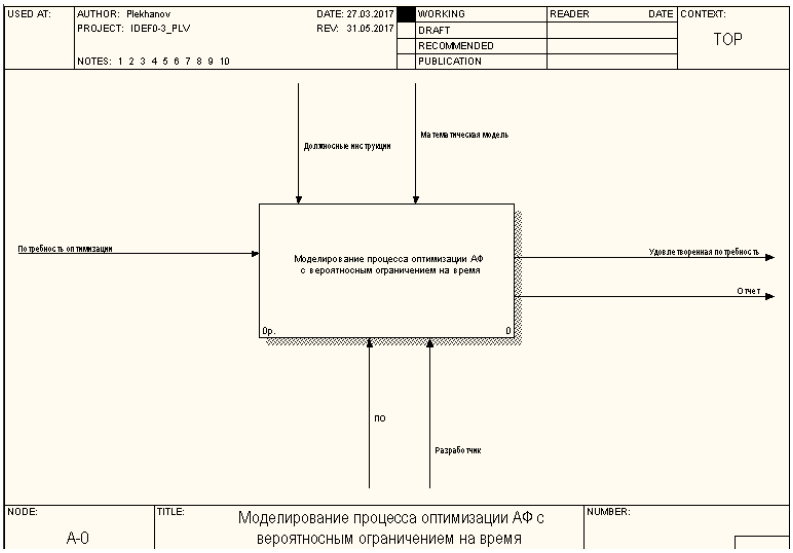


Рисунок Г.1 – Контекстна діаграма у нотації IDEF0

Після опису системи в цілому проводиться її розбиття на великі фрагменти. Цей процес називається функціональної декомпозицією, а діаграми, які описують кожен фрагмент і взаємодію фрагментів, називаються діаграмами декомпозиції. Діаграма декомпозиції контекстного представлення моделі зображена на рис.Г.2 .

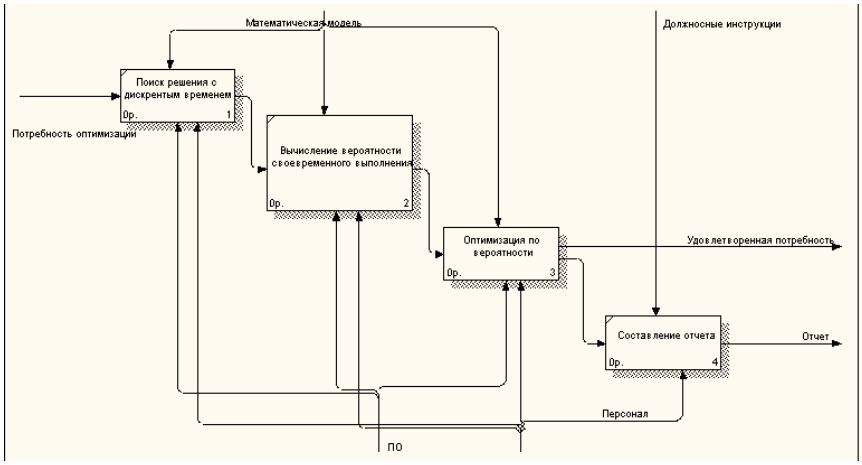


Рисунок Г.2 – Діаграма декомпозиції у нотації IDEF0

Побудова Basic Flowchart. Нотація Процес (Basic Flowchart) використовується для представлення алгоритму (сценарію) виконання процесу і дозволяють задати причинно-наслідкові зв'язки і тимчасову послідовність виконання дій процесу.

Свій початок Basic Flowchart бере в 1921 р, коли Франк Банкер Гилбрет (Frank Bunker Gilbreth) представив перший структурований метод для документування потоків процесу (flow process chart) для членів Американського товариства інженерів-механіків (ASME).

Блок-схема використовується в різних областях знань. Багатьом вона може бути відома з шкільних уроків інформатики.

Процес (Basic Flowchart) складається з прямокутників (бізнес-процеси), в які входять і виходячи стрілки (потоки інформації, документів, ТМЦ). Так само в нотації використовуються елементи типу «рішення», які дозволяють робити розгалуження. Для позначення початку виконання всього бізнес-процесу і його закінчення можуть бути використані фігури типу «подія» (елементи, схожі на овали).

Кожен бізнес-процес на нотації може бути декомпозований (розбитий на

детальні бізнес-процеси) в нотаціях Процес, Процедура і EPC.

Переваги Процесу (Basic Flowchart) в простоті і наочності. З її допомогою можна швидко описати кроки бізнес-процесу. Використання Процесу (Basic Flowchart) не вимагає спеціальних знань, тому що легко сприймається співробітниками з різним рівнем підготовки.

Недоліки Процесу (Basic Flowchart) теж в простоті. Набір графічних елементів дуже обмежений для передачі інформації про бізнес-процесі. Наприклад, на діаграмі ніяк не позначені учасники бізнес-процесу.

У нашому випадку розглядаємо предметну область створення інформаційної технології оптимізації. Нотація «Процес» (Basic Flowchart), застосована до даної предметної області, представлена на рис.Г.3 .

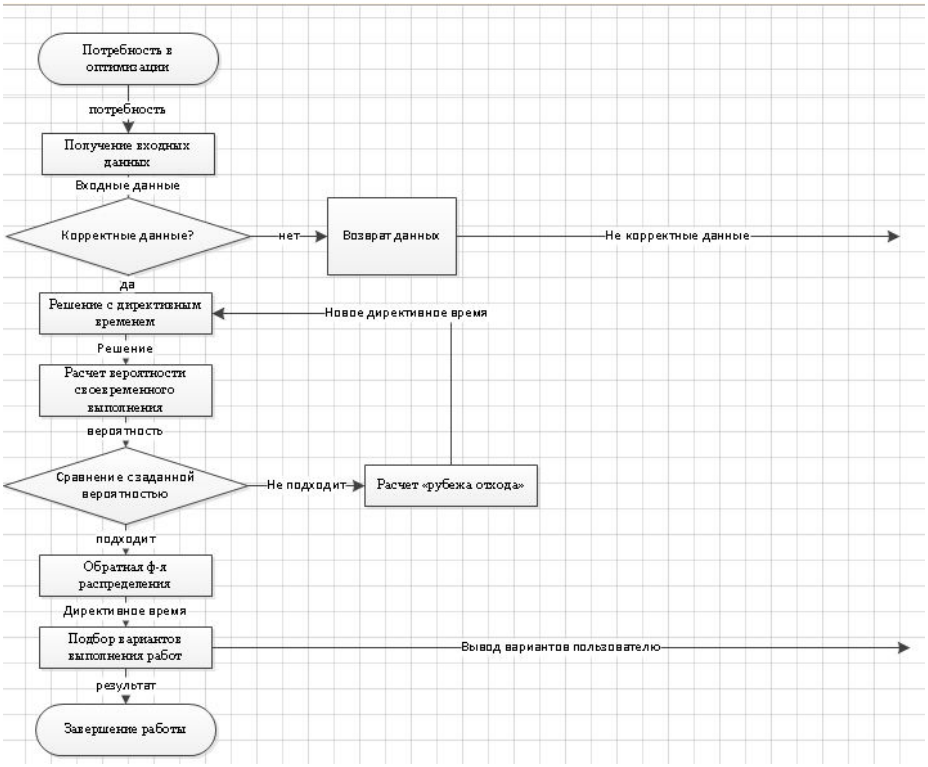


Рисунок Г.3 – Процес Basic Flowchart

Побудова Use Case Diagram. Візуальне моделювання в UML можна представити як деякий процес поуровневого спуску від найбільш загальної і абстрактної концептуальної моделі вихідної системи до логічної, а потім і до фізичної моделі відповідної програмної системи. Для досягнення цих цілей спочатку будується модель у формі так званої діаграми варіантів використання (use case

diagram). Діаграма варіантів використання – діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також називається діаграмою прецедентів.

Розробка діаграми варіантів використання переслідує мети:

1. Визначити загальні межі і контекст модельованої предметної області на початкових етапах проектування системи.

2. Сформулювати загальні вимоги до функціонального поведінки проектованої системи.

3. Розробити вихідну концептуальну модель системи для її подальшої деталізації у формі логічних і фізичних моделей.

4. Підготувати вихідну документацію для взаємодії розробників системи з її замовниками і користувачами.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей або акторів, що взаємодіють з системою за допомогою так званих варіантів використання. При цьому актором (actor) або дійовою особою називається будь сутність, що взаємодіє з системою ззовні. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка може служити джерелом впливу на моделируемую систему так, як визначить сам розробник. У свою чергу, варіант використання (use case) служить для опису сервісів, які система надає акторові.

Варіанти використання можуть бути специфіковані у вигляді тексту, а в подальшому - за допомогою операцій і методів разом з атрибутами, у вигляді графа діяльності, за допомогою автомата або будь-якого іншого механізму опису поведінки, що включає предусловия і постумови. Взаємодія між варіантами використання і акторами може уточнюватися на діаграмі кооперації, коли описуються взаємозв'язки між сутністю, що містить ці варіанти використання, і оточенням або зовнішнім середовищем цієї сутності.

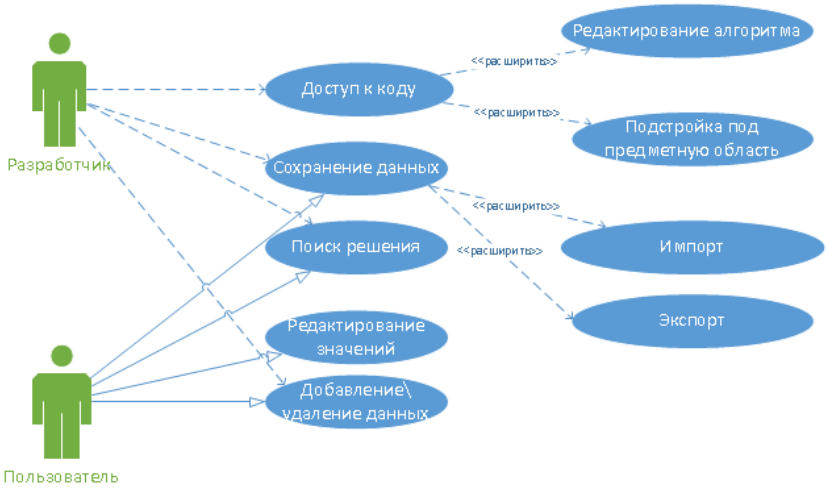


Рисунок Г.4 – Головна модель варіантів використання

*Побудова business process model and notation.*BPMN (англ. Business Process Model and Notation, нотація і модель бізнес-процесів) - система умовних позначень (нотація) для моделювання бізнес-процесів. Розроблено Business Process Management Initiative (BPMI.org) і підтримується Object Management Group, після злиття організацій в 2005 році. Остання версія BPMN - 2.0

Специфікація BPMN описує умовні позначення для відображення бізнес-процесів у вигляді діаграм бізнес-процесів. BPMN орієнтована як на технічних фахівців, так і на бізнес-користувачів. Для цього мову використовує базовий набір інтуїтивно зрозумілих елементів, які дозволяють визначати складні семантичні конструкції. Крім того, специфікація BPMN визначає, як діаграми, що описують бізнес-процес, можуть бути трансформовані в виконувані моделі на мові BPEL. Специфікація BPMN 2.0 також є виконуваною і яку переносять (тобто процес, намальований в одному редакторі від одного виробника, може бути виконаний на движку бізнес-процесів абсолютно іншого виробника, за умови, якщо вони підтримують BPMN 2.0).

Основна мета BPMN – створення стандартного набору умовних позначень, зрозумілих всім бізнес-користувачам. Бізнес-користувачі включають в себе бізнес-аналітиків, що створюють і покращують процеси, технічних розробників, відповідальних за реалізацію процесів і менеджерів, що стежать за процесами і керуючих ними. Отже, BPMN покликана служити сполучною ланкою між фазою дизайну бізнес-процесу і фазою його реалізації.[34]

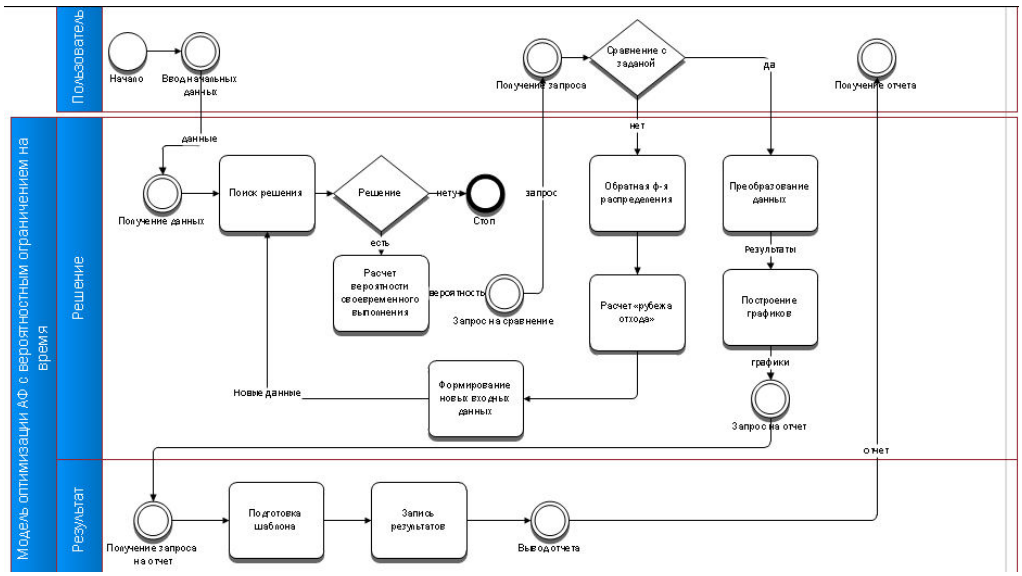


Рисунок Г.5 – Процес користування інформаційною системою у нотації BPMN

ДОДАТОК Д
ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА КОМП'ЮТЕРНІ ЕКСПЕРИМЕНТИ

Тестування інформаційної системи. Після розробки будь-якого програмного продукту відбувається етап тестування, на якому перевіряють якість розробленого продукту . Тестування буває функціональне та не функціональне. Під час функціонального тестування проводиться перевірка на коректність роботи усіх функцій, які були описані у технічному завданні. На етапі не функціонального тестування проводиться перевірка характеристик розробленого продукту.

Спочатку відбувалась перевірка правильності введення даних. В ситуації, коли користувач вводить невірні чи недопустимі символи, програма виводить спеціальне вікно про помилку. По натисканні кнопки «Продолжить», користувач має змогу виправити введене значення, а якщо натиснув «Выход», то відбувається вихід з програми.

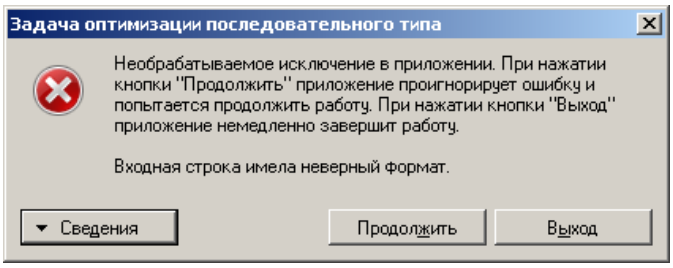


Рисунок Д.1 – Повідомлення про помилку

Після цього відбулась перевірка створеної програми на правильність отриманих значень. Основним способом підтвердження було порівняння отриманих значень з тими, що були отримані вручну за допомогою MS Excel. Для прикладу було обрано типові вхідні дані та вирішено задачу оптимізації за допомогою MS Excel та створеного додатку.

Результат выбора:			
		Вероятность безошибочного функционирования:	0,912
способы	7	8	
вероятность своевременного выполнения	0,69146		
Дисперсия	2		
рубеж отхода	107,0	107	

Рисунок Д.2 – Результат першої ітерації

Результат выбора:			
		Вероятность безошибочного функционирования:	0,8928
дисперсия:	2		
вероятность своевременного выполн	0,933192799		
рубеж отхода	103,0	103	
способы:	9	9	

Рисунок Д.3 – Результат другої ітерації

Результат выбора:			
		Вероятность безошибочного функционирования:	0,874
дисперсия:	2		
вероятность своевременного выполн	0,998650102		
рубеж отхода	97,0	97	
способы:	2	9	

Рисунок Д.4 – Остаточний результат

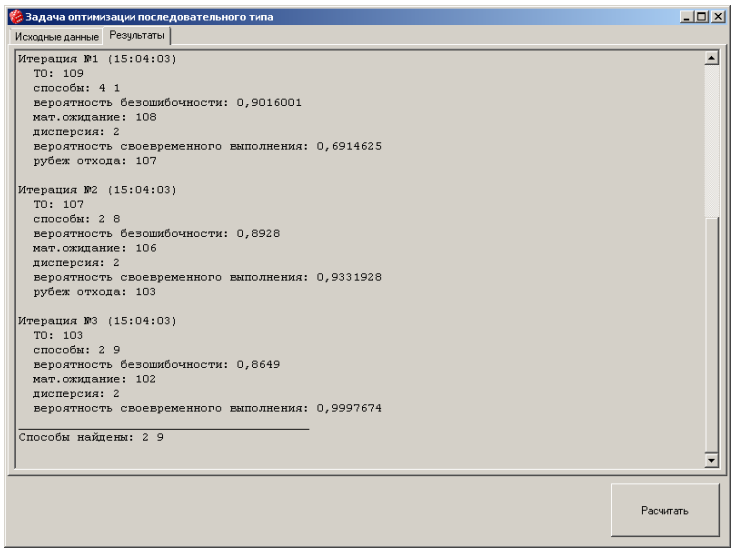


Рисунок Д.5 – Результати обчислень програмою

У більшості випадків створена інформаційна система має однаковий результат із ручною реалізацією. Існує похибка обчислень, яка пов’язана з реалізацією методів нормального та оберненого нормального розподілу всередині програми та MS Excel. Але також результати роботи програми можуть не співпасти з отриманими значеннями в Excel. Такі випадки відбуваються через недосконаліть інструменту пошук рішення, який не завжди дає оптимальний результат, навіть якщо перевіряти деякі рішення самостійно, можна знайти способи оптимальніші.

Було перевірено збереження даних, а саме експорт даних, та імпорт. Також перевірена можливість завантажувати вхідні дані.

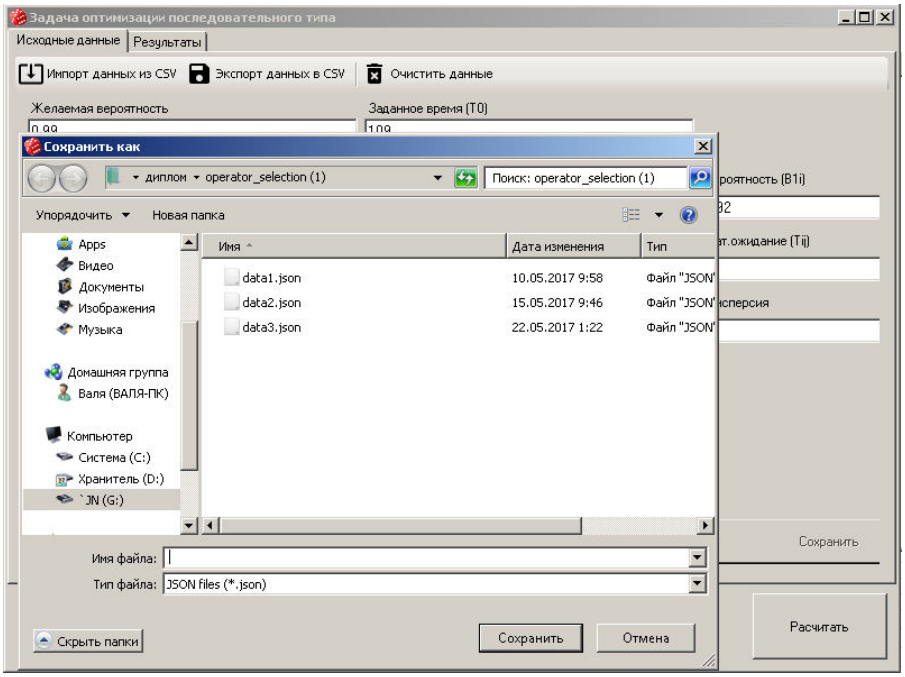


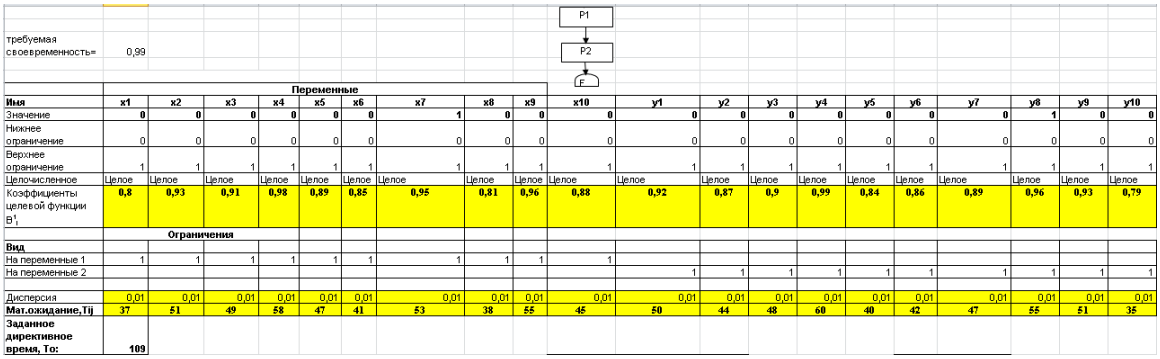
Рисунок Д.6 – Експорт та імпорт

Отже, інформаційна система працює коректно.

*Комп’ютерні експерименти.*Комп’ютерні експерименти проводились шляхом використання інформаційної технології для вирішення задачі оптимізації при різних вхідних даних, перевіряючи якість виконання окремих операцій. Всього було проведено 58 експериментів.

На початку було перевірено вплив середнього відхилення на результати, наведено графіки зміни області допустимих рішень.

Для цього на прикладі двох операцій по 10 способів виконання та за допомогою MS Excel проводилось дослідження. Були введені вхідні дані, які не змінювались, а саме математичне сподівання та ймовірність безпомилкового виконання. Змінювалось лише значення дисперсії. В першому дослідженні було обрано значення середнього відхилення 0,01.



требуемая своевременность=	0,99																			
	Переменные																			
Имя	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	y1	y2	y3	y4	y5	y6	y7	y8	y9	y10
Значение	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
Нижнее ограничение	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Верхнее ограничение	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Целочисленное	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое	Целое
Коэффициенты целевой функции B^1_i	0,8	0,93	0,91	0,98	0,89	0,85	0,95	0,81	0,96	0,88	0,92	0,87	0,9	0,99	0,84	0,86	0,89	0,96	0,93	0,79
	Ограничения																			
Вид																				
На переменные 1	1	1	1	1	1	1	1	1	1	1										
На переменные 2											1	1	1	1	1	1	1	1	1	1
Дисперсия	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01
Мат.ожидание,Tij	37	51	49	58	47	41	53	38	55	45	50	44	48	60	40	42	47	55	51	35
Заданное директивное время, To:	109																			

Рисунок Д.7 – Вхідні дані

В цьому випадку задача зводиться до рішення детермінованої задачі на математичне сподівання. Тому було вирішено її однією ітерацією.

		Левая часть	Знак	Правая
		1	=	1
Вероятность безошибочного функционирования:	0,912	1	=	1
		108,0	≤	109

Рисунок Д.8 – Результат

Після цього було змінено значення середнього відхилення до 0,5 та вирішено задачу оптимізації. Для цього значення знадобилось дві ітерації. Вхідні дані ідентичні до рисунку В.1, змінено до 0,5 дисперсія. На першій ітерації було отримано наступний результат.

Результат выбора:			
	Вероятность безошибочного функционирования:		0,912
способы			
вероятность своевременного выполнения	0,84134		
Дисперсия	1		
рубеж отхода	107,0	107	

Рисунок Д.9 – Результат першої ітерації

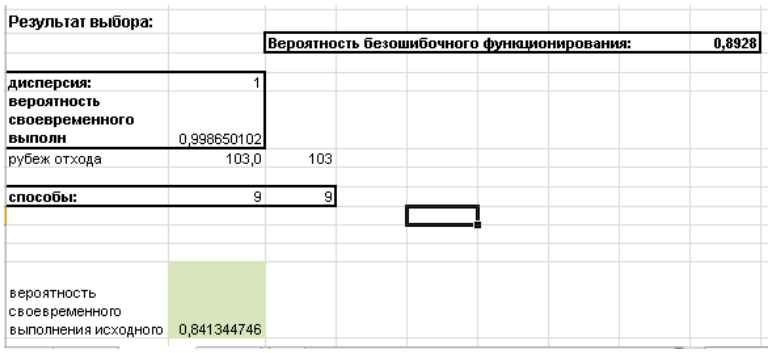


Рисунок Д.10 – Результат другої ітерації

Наступним кроком було підвищення дисперсії до 0,75. На цьому етапі не досягнуто оптимальне рішення вже на другій ітерації, знадобилась третя ітерація.

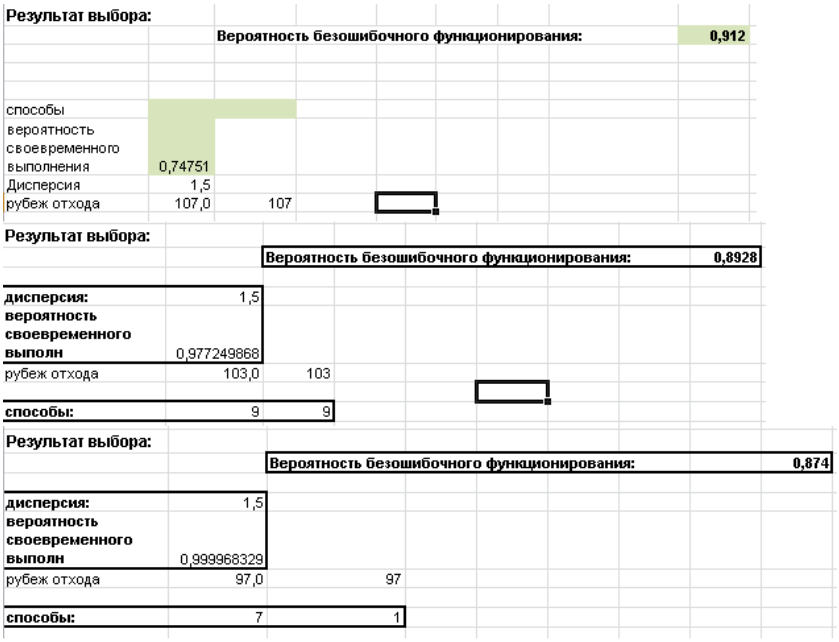


Рисунок Д.11 – Результати задачі з відхиленням 0,75

Наступним кроком було підвищення відхилення до 1, але отримані значення майже не відрізняються від результатів зі значенням 0,75, тому відразу перехід до 1,5. При цих значеннях знадобилось 4 ітерації та супроводжувалось дослідження графіками, на яких продемонстрована зміна області допустимих рішень.

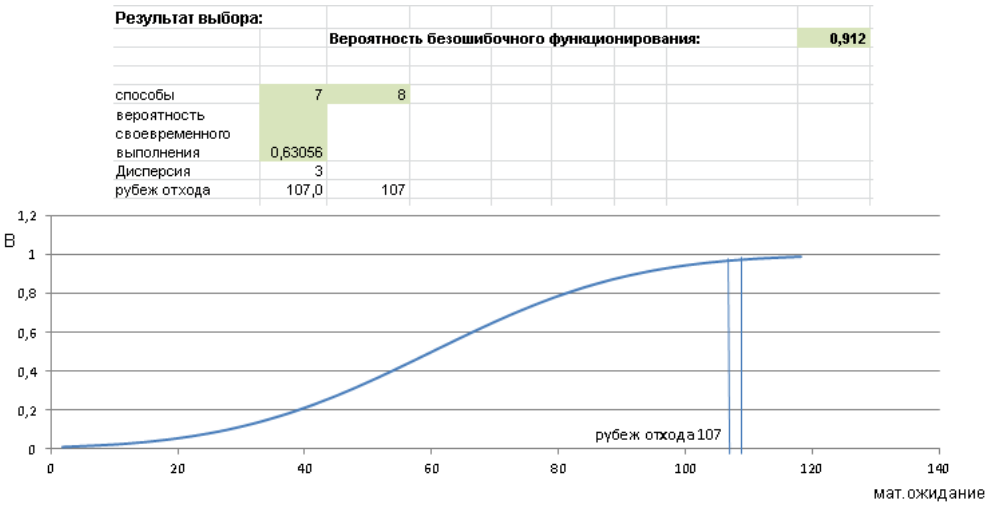


Рисунок Д.12 – Результат першої ітерації з відхиленням 1,5

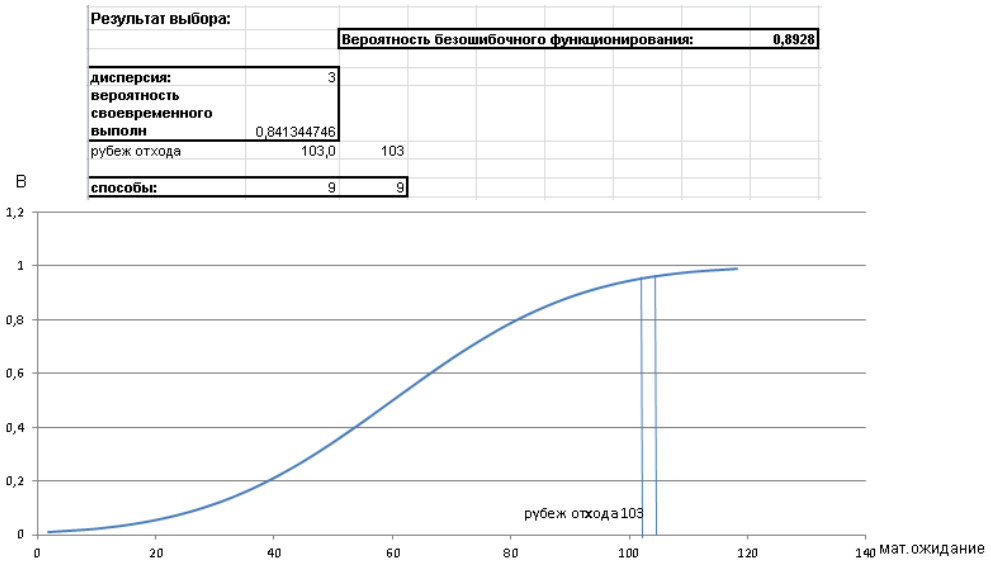


Рисунок Д.13 – Результат другої ітерації з відхиленням 1,5

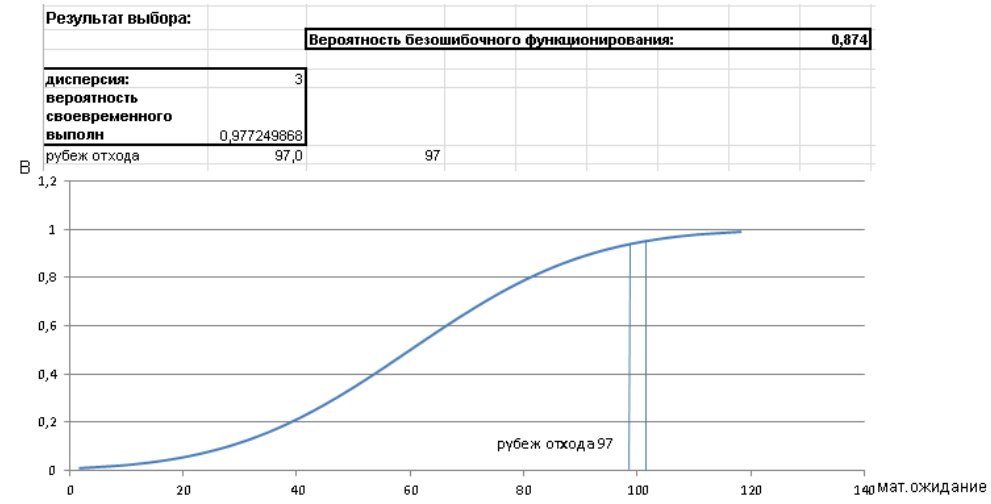


Рисунок Д.14 – Результат третьої ітерації з відхиленням 1,5

Результат выбора:			
		Вероятность безошибочного функционирования:	**0,817**
дисперсия:	3		
вероятность своевременного выполн	0,999998469		
рубеж отхода	81,0	81	
способы:	7	6	

Рисунок Д.15 – Результат четвертої ітерації з відхиленням 1,5

Після одержання кінцевого результату наведено графік залежності ймовірності безпомилковості та ймовірності своєчасності на кожній ітерації для того, щоб наглядно переконатись у оптимальності отриманого рішення.

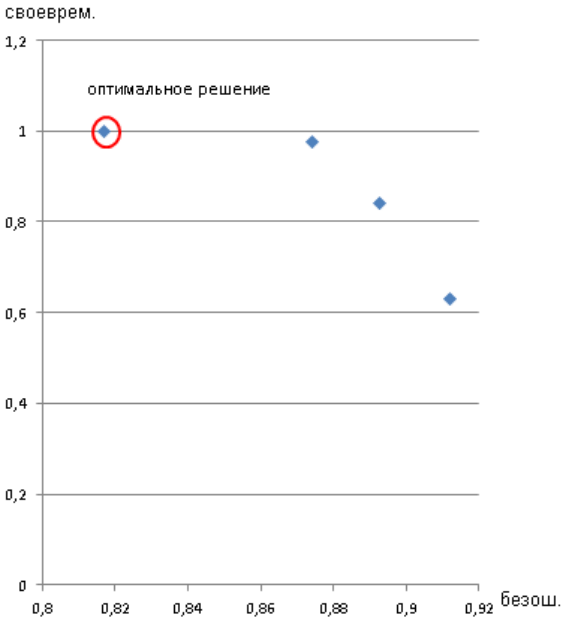


Рисунок Д.16 – Положення оптимального рішення

Дані дії були проведені ще з декількома вхідними даними, головною метою яких була перевірка отриманих рішень на оптимальність. Нижче наведено графік залежності своєчасності та безпомилковості при більшому числі корегування ОДР.

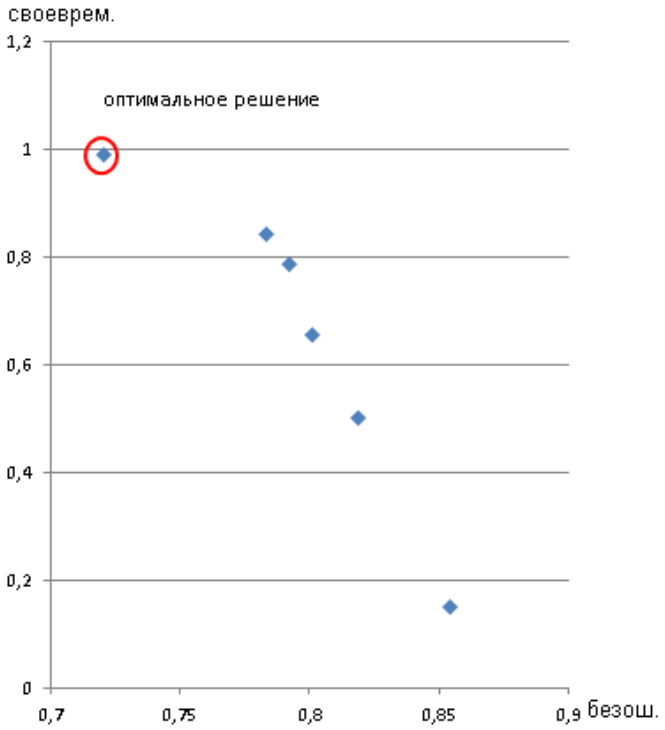


Рисунок Д.17 – Положення оптимального рішення

Далі продемонстровано інший спосіб дослідження зміни ОДР, цього разу в залежності від способів виконання першої та другої операції.

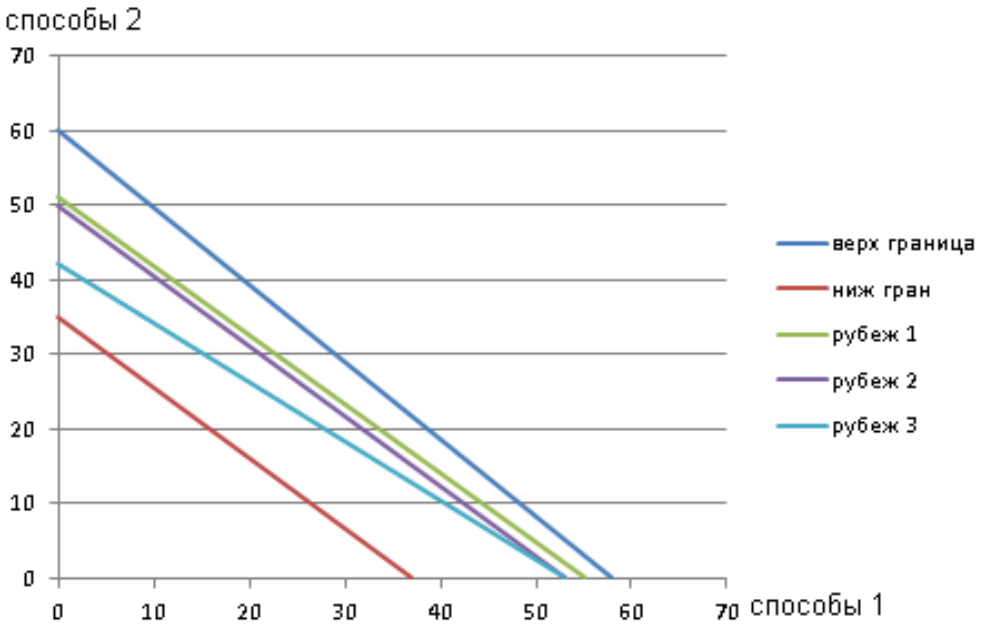


Рисунок Д.18 – Положення кордонів відходу

На графіку відображено положення верхньої межі, тобто максимальне значення математичного сподівання обох операцій, та нижню межу, яка є мінімальним часом виконання операцій. Так як на прикладі було вирішена задача оптимізації за чотири ітерації, то було розраховано три положення кордону відходу, в межах яких знаходиться оптимальне рішення.

В ході перевірки роботи алгоритму можна зробити наступні висновки:

- Значення дисперсії близьке до нуля зводить задачу оптимізації до детермінованої.
- Зі збільшенням значення дисперсії підвищується кількість корегування області допустимих рішень.
- На результат впливає загальна дисперсія виконання операцій.
- Зі збільшенням дисперсії при однакових вхідних даних значення ймовірності безпомилкового виконання буде зменшуватись, при виконанні умови своєчасності.

Отже, так як задача має обмеження на своєчасне виконання, та на максимальну безпомилковість, в ході експериментів було доведено оптимальність отриманих рішень. Вони мали ймовірність своєчасного виконання більше заданої, одночасно з цим, ймовірність безпомилкового виконання була максимальною. Наведено графіки, що наглядно демонструють пошук оптимума в задачі оптимізації.

ДОДАТОК Е
ЛІСТИНГ ПРОГРАМИ ОПТИМІЗАЦІЇ АЛГОРИТМУ ФУНКЦІОНУВАННЯ ЛЮДИНО-МАШИННИХ СИСТЕМ

Файл form1.cs

```
<!DOCTYPE html>
<html> using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using TAlex.MathCore.Statistics.Distributions;
using Meta.Numerics.Statistics.Distributions;
using Newtonsoft.Json;
using System.IO;
using operator_selection;

namespace norm_dsitrib
{
   public partial class frmMain : Form
   {
      public frmMain()
      {
         InitializeComponent();
      }
      private void Form1_Load(object sender, EventArgs e)
      {
         Init();
      }
      private void Init()
      {
         txtTargetProbability.Text = "" + 0.99f;
         txtT0.Text = "" + 120f;
         data.Clear();
         //генерация тестовых данных
         //GenerateData();
         List<StepPart> step = new List<StepPart>();
         step.Add(new StepPart());
         step.Add(new StepPart());
         data.Add(step);
         data.Add(step);
         UpdateVariablesList();
      }
      private void UpdateVariablesList()
      {
         int selected = lstVariables.SelectedIndex;
         lstVariables.Items.Clear();
         for (int i = 0; i < data.Count; i ++)
         {
            lstVariables.Items.Add("Переменная " + (i + 1));
         }
         lstVariables.SelectedIndex = Math.Min(0, lstVariables.Items.Count - 1);
         UpdateMethodsList();
      }
      private void UpdateMethodsList()
```

```
{
    int selected = lstMethods.SelectedIndex;
    lstMethods.Items.Clear();
    if (lstVariables.SelectedIndex != -1)
    {
        for (int i = 0; i < data[lstVariables.SelectedIndex].Count; i ++)
        {
            lstMethods.Items.Add(i + ": " + data[lstVariables.SelectedIndex][i].ToDisplay());
        }
        if (selected == -1 && lstMethods.Items.Count > 0)
            selected = 0;
        lstMethods.SelectedIndex = Math.Min(selected, lstMethods.Items.Count - 1);
    }
    DisplayMethod();
}
private void DisplayMethod()
{
    if(lstVariables.SelectedIndex != -1 && lstMethods.SelectedIndex != -1)
    {
        txtMultiplier.Text = "" + data[lstVariables.SelectedIndex][lstMethods.SelectedIndex].multiplier;
        txtWaiting.Text = "" + data[lstVariables.SelectedIndex][lstMethods.SelectedIndex].waiting;
        txtDipression.Text = "" + data[lstVariables.SelectedIndex][lstMethods.SelectedIndex].disperse;
        pnlMethodData.Enabled = true;
        cmdSave.Enabled = false;
    }
    else
    {
        txtMultiplier.Text = "";
        txtWaiting.Text = "";
        txtDipression.Text = "";
        pnlMethodData.Enabled = false;
    }
}
private List<List<StepPart>> data = new List<List<StepPart>>();
private List<StepResult> results = new List<StepResult>();
private float target;
private float t0;
private float init_t0;
private int iteration = 0;
private void cmdCalc_Click(object sender, EventArgs e)
{
    cmdCalc.Enabled = false;
    init_t0 = float.Parse(txtT0.Text);
    t0 = init_t0;
    target = float.Parse(txtTargetProbability.Text);
    iteration = 1;
    tabPnl.SelectedIndex = 1;
    results.Add(new StepResult());
    CalculateData();
}
private void CalculateData()
{
    float tMax = 0;
    float multiplierMax = 0;
    List<int> ids = new List<int>();
    List<int> index = new List<int>();
    for (int i = 0; i < data.Count; i++)
    {
        ids.Add(0);
        index.Add(0);
    }
    while (index.Count != 0)
    {
        float t = 0;
        float multiplier = 1;
```

```
                for (int i = 0; i < index.Count; i++)
                {
                    multiplier *= (float)data[i][index[i]].multiplier;
                    t += (float)data[i][index[i]].waiting;
                }
                if (multiplier > multiplierMax && multiplier < target &&
                  t < t0 && t > tMax)
                {
                    multiplierMax = multiplier;
                    tMax = t;
                    ids.Clear();
                    for (int i = 0; i < index.Count; i++)
                        ids.Add(index[i]);
                }
                index = IndexUpdate(index);
            }
            float dispers = 0;
            for (int i = 0; i < ids.Count; i++)
                dispers += data[i][ids[i]].disperse;
            float result = CumulativeDistribution(init_t0, tMax, dispers);
            String idsToPrint = "";
            for (int i = 0; i < ids.Count; i++)
                idsToPrint += (ids[i] + 1) + " ";
            results[results.Count - 1].iteration.Add(new ResultItem(iteration++,
                                        DateTime.Now.ToLongTimeString(),
                                        t0,
                                        ids,
                                        multiplierMax,
                                        tMax,
                                        dispers,
                                        result));
                if (result < target)
            {
                t0 = (float)Math.Round(t0 + tMax - DistributionInvert(result, t0, dispers), 2);
                results[results.Count - 1].iteration[results[results.Count - 1].iteration.Count - 1].AddT0(t0);
                CalculateData();
            }
            else
            {
                lstResults.Items.Add("Эксперемент №" + results.Count + ". Способы: " + results[results.Count -
1].GetResultIds());
                lstResults.SelectedIndex = lstResults.Items.Count - 1;
                cmdCalc.Enabled = true;
            }
        }
        private List<int> IndexUpdate(List<int> index)
        {
            for (int i = 0; i < data.Count; i ++)
            {
                if (index[i] < data[i].Count - 1)
                {
                    index[i]++;
                    return index;
                }
                else
                {
                    index[i] = 0;
                }
            }
            index.Clear();
            return index;
        }
        private void GenerateData()
        {
            if (data.Count > 0) return;
```

```
            data.Clear();
            List<StepPart> x = new List<StepPart>();
            x.Add(new StepPart(0.80f, 37f, 0.5f));
            x.Add(new StepPart(0.93f, 51f, 1.0f));
            x.Add(new StepPart(0.91f, 49f, 0.7f));
            x.Add(new StepPart(0.98f, 58f, 2.0f));
            x.Add(new StepPart(0.89f, 47f, 0.5f));
            x.Add(new StepPart(0.85f, 41f, 0.4f));
            x.Add(new StepPart(0.95f, 53f, 1.5f));
            x.Add(new StepPart(0.81f, 38f, 0.5f));
            x.Add(new StepPart(0.96f, 55f, 0.2f));
            x.Add(new StepPart(0.88f, 45f, 1.0f));
            data.Add(x);
            List<StepPart> y = new List<StepPart>();
            y.Add(new StepPart(0.92f, 50f, 1.0f));
            y.Add(new StepPart(0.87f, 44f, 1.0f));
            y.Add(new StepPart(0.90f, 48f, 1.0f));
            y.Add(new StepPart(0.99f, 60f, 3.0f));
            y.Add(new StepPart(0.84f, 40f, 0.8f));
            y.Add(new StepPart(0.86f, 42f, 0.5f));
            y.Add(new StepPart(0.89f, 47f, 1.0f));
            y.Add(new StepPart(0.96f, 55f, 1.1f));
            y.Add(new StepPart(0.93f, 51f, 1.2f));
            y.Add(new StepPart(0.79f, 35f, 0.5f));
            data.Add(y);
        }
        private float CumulativeDistribution(float x, float mu, float sigma)
        {
            TAlex.MathCore.Statistics.Distributions.NormalDistribution distr = new
TAlex.MathCore.Statistics.Distributions.NormalDistribution(mu, sigma);
            return (float)distr.CumulativeDistributionFunction(x);
        }
        private float DistributionInvert(float x, float mu, float sigma)
        {
            Meta.Numerics.Statistics.Distributions.Distribution distr = new
Meta.Numerics.Statistics.Distributions.NormalDistribution(mu, sigma);
            return (float)distr.InverseLeftProbability(x);
        }
        private void lstVariables_SelectedIndexChanged(object sender, EventArgs e)
        {
            UpdateMethodsList();
        }
        private void lstMethods_SelectedIndexChanged(object sender, EventArgs e)
        {
            DisplayMethod();
        }
        private void DataChanged(object sender, EventArgs e)
        {
            cmdSave.Enabled = true;
        }
        private void cmdSave_Click(object sender, EventArgs e)
        {
            if (lstVariables.SelectedIndex != -1 && lstMethods.SelectedIndex != -1)
            {
                data[lstVariables.SelectedIndex][lstMethods.SelectedIndex].multiplier = float.Parse(txtMultiplier.Text);
                data[lstVariables.SelectedIndex][lstMethods.SelectedIndex].waiting = float.Parse(txtWaiting.Text);
                data[lstVariables.SelectedIndex][lstMethods.SelectedIndex].disperse = float.Parse(txtDipression.Text);
                UpdateMethodsList();
            }
        }
        private void cmdVariableAdd_Click(object sender, EventArgs e)
        {
            List<StepPart> x = new List<StepPart>();
            x.Add(new StepPart());
            x.Add(new StepPart());
```

```
            data.Add(x);
            UpdateVariablesList();
            lstVariables.SelectedIndex = lstVariables.Items.Count - 1;
        }
        private void cmdMethodAdd_Click(object sender, EventArgs e)
        {
            if (lstVariables.SelectedIndex != -1) {
                data[lstVariables.SelectedIndex].Add(new StepPart());
                UpdateMethodsList();
                lstMethods.SelectedIndex = data[lstVariables.SelectedIndex].Count - 1;
                txtMultiplier.Focus();
            }
        }
        private void cmdVariableDelete_Click(object sender, EventArgs e)
        {
            if (lstVariables.SelectedIndex != -1)
            {
                if (MessageBox.Show("Удалить [" + lstVariables.Items[lstVariables.SelectedIndex] + "]?", "Удаление
переменной", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
                {
                    data.Remove(data[lstVariables.SelectedIndex]);
                    UpdateVariablesList();
                }
            }
        }
        private void cmdMethodDelete_Click(object sender, EventArgs e)
        {
            if (lstVariables.SelectedIndex != -1 && lstMethods.SelectedIndex != -1)
            {
                if (MessageBox.Show("Удалить [" + lstMethods.Items[lstMethods.SelectedIndex] + "]?", "Удаление
способа", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
                {
                    data[lstVariables.SelectedIndex].Remove(data[lstVariables.SelectedIndex][lstMethods.SelectedIndex]);
                    UpdateMethodsList();
                }
            }
        }
        private void cmdExport_Click(object sender, EventArgs e)
        {
            dlgSave.Filter = "Data files|*.osjson";
            dlgSave.FileName = "";
            dlgSave.ShowDialog();
        }
        private void dlgSave_FileOk(object sender, CancelEventArgs e)
        {
            if (dlgSave.FileName != "" && dlgSave.FileName.EndsWith(".osjson"))
            {
                WriteToJsonFile<List<List<StepPart>>>(dlgSave.FileName, data, false);
            }
        }
        private void cmdImport_Click(object sender, EventArgs e)
        {
            dlgOpen.Filter = "Data files|*.osjson";
            dlgOpen.FileName = "";
            dlgOpen.ShowDialog();
        }
        private void dlgOpen_FileOk(object sender, CancelEventArgs e)
        {
            if (dlgOpen.FileName != "" && dlgOpen.FileName.EndsWith(".osjson"))
            {
                data = ReadFromJsonFile<List<List<StepPart>>>(dlgOpen.FileName);
                UpdateVariablesList();
            }
        }
```

```
        private static void WriteToJsonFile<T>(string filePath, T objectToWrite, bool append = false) where T : new()
        {
            TextWriter writer = null;
            try
            {
                var contentsToWriteToFile = JsonConvert.SerializeObject(objectToWrite);
                writer = new StreamWriter(filePath, append);
                writer.Write(contentsToWriteToFile);
            }
            finally
            {
                if (writer != null)
                    writer.Close();
            }
        }
        private static T ReadFromJsonFile<T>(string filePath) where T : new()
        {
            TextReader reader = null;
            try
            {
                reader = new StreamReader(filePath);
                var fileContents = reader.ReadToEnd();
                return JsonConvert.DeserializeObject<T>(fileContents);
            }
            finally
            {
                if (reader != null)
                    reader.Close();
            }
        }
        private void cmdClear_Click(object sender, EventArgs e)
        {
            if(MessageBox.Show("Очистить все данные?", "Очистка данных", MessageBoxButtons.YesNo,
MessageBoxIcon.Question) == DialogResult.Yes)
            {
                data.Clear();
                UpdateVariablesList();
            }
        }

        private void frmMain_Shown(object sender, EventArgs e)
        {
            Boolean showHelp = (Boolean) operator_selection.Properties.Settings.Default["show_help"];
            if (showHelp)
            {
                frmAbout frm = new frmAbout();
                frm.ShowDialog();
            }
        }

        private void toolStripButton1_Click(object sender, EventArgs e)
        {
            frmAbout frm = new frmAbout();
            frm.ShowDialog();
        }
        private void lstResults_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (lstResults.Items.Count > 0 && results.Count > 0 && lstResults.Items.Count == results.Count)
            {
                StepResult result = results[lstResults.SelectedIndex];
                txtResult.Text = "Результат: " + result.iteration[result.iteration.Count - 1].result;
                txtResultIds.Text = "Способы: " + result.GetResultIds();
                txtResults.Clear();
                result.GetResultIterations(txtResults);
            }
```

```
        }
    }
}
```

Файл program.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace norm_dsitrib
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new frmMain());
        }
    }
}
```

Файл resultItem.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
//using System.Threading.Tasks;

namespace operator_selection
{
    class ResultItem
    {
        public int id;
        public String dateTime;
        public float t0;
        public List<int> ids = new List<int>();
        public float multiplyMax;
        public float tMax;
        public float dispers;
        public float result;
        public float new_t0;

        public ResultItem(int id, String dateTime, float t0, List<int> ids, float multiplyMax, float tMax, float dispers,
float result)
        {
            this.id = id;
            this.dateTime = dateTime;
            this.t0 = t0;
            this.ids = ids;
            this.multiplyMax = multiplyMax;
            this.tMax = tMax;
            this.dispers = dispers;
            this.result = result;
        }

        public void AddT0(float t0)
```

```
        {
            this.new_t0 = t0;
        }

        public String GetIteration()
        {
            String res;
            res = "Итерация №" + id + " (" + dateTime + ")" +
                        "\r\n  T0: " + t0 +
                        "\r\n  способы: " + GetIds() + "" +
                        "\r\n  вероятность безошибочности: " + multiplyMax +
                        "\r\n  мат.ожидание: " + tMax +
                        "\r\n  дисперсия: " + dispers +
                        "\r\n  вероятность своевременного выполнения: " + result;

            if (new_t0 != 0)
                res += "\r\n  рубеж отхода: " + new_t0;

            return res;
        }

        private String GetIds()
        {
            String res = "";

            for (int i = 0; i < ids.Count; i++) {
                res += "" + ids[i];
                if (i < ids.Count - 1)
                    res += ", ";
            }

            return res;
        }
    }
}
```

Файл stepPart.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace norm_dsitrib
{
    class StepPart
    {
        public float multiplier;
        public float waiting;
        public float disperse;

        public StepPart()
        {
            this.multiplier = 0.99f;
            this.waiting = 10f;
            this.disperse = 1f;
```

```
        }
        public StepPart(float multiplier, float waiting, float disperse)
        {
            this.multiplier = multiplier;
            this.waiting = waiting;
            this.disperse = disperse;
        }
        public String ToDisplay()
        {
            return "B1i = " + multiplier.ToString() +
                   ", Tij = " + waiting.ToString() +
                   ", Distribution = " + disperse.ToString();
        }
    }
}
```

Файл stepResult.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
//using System.Threading.Tasks;
using System.Windows.Forms;

namespace operator_selection
{
    class StepResult
    {
        public String name;
        public List<ResultItem> iteration = new List<ResultItem>();

        public StepResult() { }

        public void GetResultIterations(TextBox txt)
        {
            for (int i = 0; i < iteration.Count; i++) {
                txt.AppendText(iteration[i].GetIteration());
                if (i < iteration.Count - 1)
                {
                    txt.AppendText("\r\n\r\n");
                }
            }
        }
        private String GetResult(List<int> result)
        {
            String res = "";
            for (int i = 0; i < result.Count; i++)
            {
                res += "" + result[i];
                if (i < result.Count - 1)
                    res += ", ";
            }
            return res;
        }
        public String GetResultIds()
```

```
        {
            return GetResult(iteration[iteration.Count - 1].ids);
        }
    }
}
```