УДК 519.086

Новіцька В.І., студентка 4 курсу спеціальності 122 «Комп'ютерні науки»
Штовба С.Д., д. т. н., професор, професор кафедри інформаційних технологій

АЛГОРИТМ ОПТИМІЗАЦІЇ ІНТЕРВАЛЬНОГО РОЗКЛАДУ ДЛЯ ПРОГРАМИ ПЛАНУВАЛЬНИКА ПЕРСОНАЛЬНОЇ ДІЯЛЬНОСТІ

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасному швидкоплинному та динамічному світі ефективне планування стало важливим інструментом для підприємств, організацій та окремих людей. Швидкий розвиток технологій ще більше змінив наш підхід до планування з появою відповідних застосунків. Ці потужні цифрові інструменти змінили традиційні підходи організації та планування завдань, пропонуючи користувачам спрощений та ефективний засіб керування своїм часом, ресурсами та цілями.

Планування передбачає створення стратегічного плану дій, який визначає порядок і важливість завдань. Його мета — організувати робочий процес, встановити конкретні терміни виконання завдань і ефективно розподілити ресурси (наприклад, персонал, матеріали та фінанси).

Одним з цікавих інструментів, що допоможе структурувати робочий процес, є оптимізація розкладу завдань, яка передбачає розподіл часових інтервалів для конкретних завдань, операцій або робіт таким чином, щоб отриманий розклад не мав задач, які конфліктують між собою, і відповідав якійсь меті, наприклад, виконання у найкоротший термін. Призначаючи конкретні часові інтервали для різних завдань, люди можуть визначати пріоритети своїх дій на основі їх важливості, терміновості або залежності. Такий підхід забезпечує виконання основних завдань у встановлені терміни, запобігаючи затримкам і потенційним вузьким місцям.

Ідея організації планування основана на використанні програмного забезпечення, що допомагає керувати та оптимізувати розклад завдань. Вже існує чимало програм та сервісів, які надають функціонал для планування, проте не всі з них враховують особливості інтервальної оптимізації розкладу завдань. Серед подібних сервісів оберемо Microsoft Project, Asana та Trello та розглянемо їхні особливості.

Таблиця 1 – Порівняння аналогів

Функціонал	**MS Project**	**Trello**	**Asana**	**Lotus**
Командна робота	✓	✓	✓	✗
Індивідуальне використання	✗	✓	✓	✓
Канбан дошка	✓	✓	✓	✓
Список усіх задач	✓	✗	✓	✓
Статистика	✓	✗	✗	✓
Календар	✓	✓	✓	✓
Генерація розкладу	✓	✗	✗	✓
Мінімальний тарифний план	$10	free	free	free

Розглядаючи характеристики додатків з таблиці 1, бачимо, що перші три сервіси більше спрямовані на командну співпрацю. Хоча MS Project пропонує широкий спектр функціонала для генерації та керування розкладом, він може бути надмірним для особистого використання, а мінімальний тарифний план складає $10. З іншого боку, Asana пропонує безкоштовний план, який може бути використаний для особистих потреб, але він обмежений у функціоналі і не має автоматизованої генерації розкладу. Аналогічним чином, Trello, широко використовуваний інструмент персонального планування, не має інтегрованої функції планування.

В розглянутих аналогах відсутня можливість планування інтервального розкладу. Задачу створення інтервального розкладу сформулюємо таким чином. Відомо: список з мінімум однієї задачі, яка має час початку та кінця виконання, а також пріоритет, робочі години, що обмежують розклад та інтервал перерви – 30 хвилин, який має бути після запланованої задачі. Потрібно створити такий варіант розкладу в межах призначеного робочого часу, щоб виконати усі поставлені задачі в найкоротший термін з урахуванням інтервалів на перерву. При цьому потрібно забезпечити організацію завдань таким чином, щоб уникнути конфліктів або збігів одне з одним.

Перейдемо до розгляду алгоритму генерації розкладу завдань, на якому буде побудовано планівник. Розглянемо покроково принцип дії алгоритму інтервального планування:

1. Спочатку інтервали сортуються за часом початку або закінчення, забезпечуючи порядок без спадання.
2. Вибирається перший інтервал із відсортованого списку та додається до розкладу.

3. Після цього, інтервали, що залишилися, повторюються, починаючи з другого інтервалу в сортованому списку.
4. Перевіряється сумісність між інтервалами: для кожного інтервалу порівнюється час його початку з часом закінчення раніше обраного інтервалу в розкладі. Якщо час початку поточного інтервалу дорівнює або перевищує час закінчення попереднього інтервалу, він вважається сумісним і може бути включений до розкладу.
5. Якщо інтервал виявився несумісним, тоді пошук переходить на наступний день, і перевіряє задачі уже цього дня.
6. Якщо запропонований час не належить до меж годин робочого дня, аналогічним чином пошук переходить до перевірки наступного дня.
7. Коли сумісний інтервал знайдено, він додається до розкладу, а попередньо вибраний інтервал оновлюється, щоб стати поточним інтервалом.
8. Кроки 4 - 7 повторюються: ітерація продовжується через решту інтервалів, перевіряючи сумісність і відповідно коригуючи розклад.

У результаті отримаємо розклад, що максимізує кількість інтервалів без конфліктів, забезпечуючи відсутність двох різних завдань, запланованих для одного інтервалу. Цей підхід називають «жадібною» стратегією, оскільки він робить локально оптимальний вибір на кожному кроці, не враховуючи глобальну перспективу.

На рисунку 1 зображено список задач, на основі яких буде створено розклад. Задамо задачам різний час початку та кінця, щоб продемонструвати роботу алгоритму (див. рис. 2). Основна мета полягає в тому, щоб завдання були організовані без будь-якого накладення та з інтервалами «відпочинку».

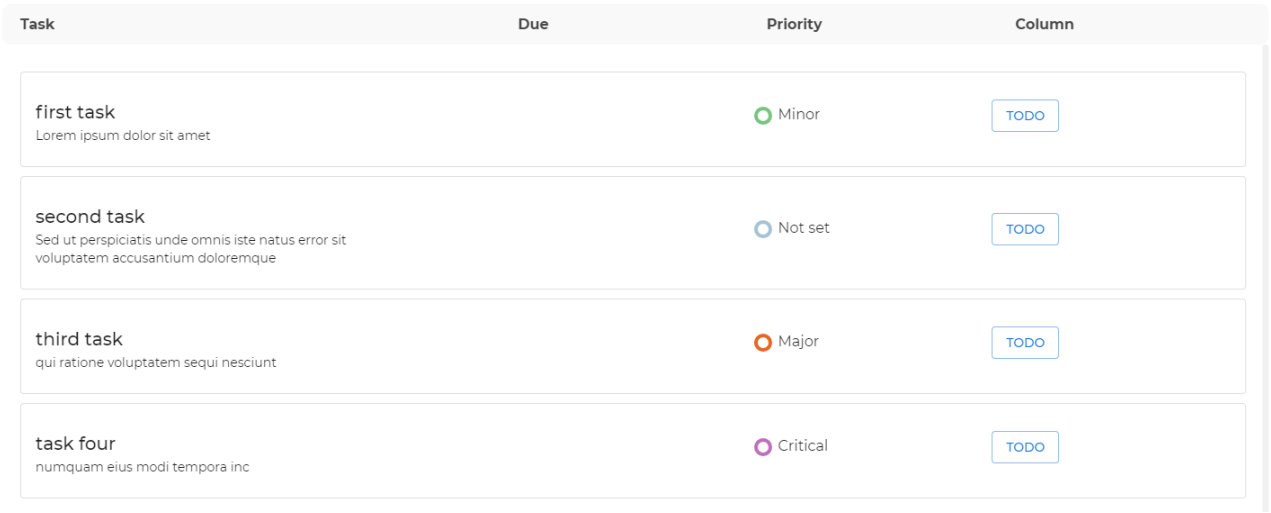
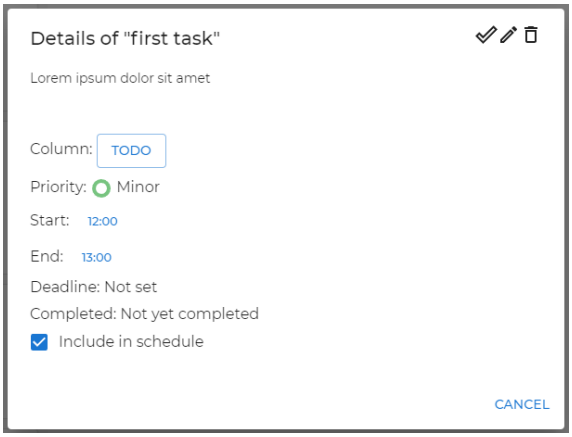
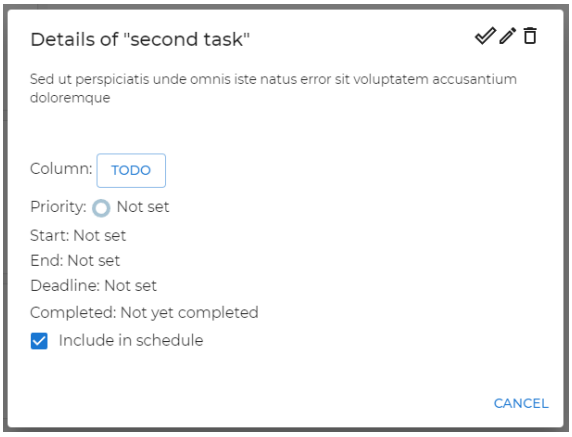


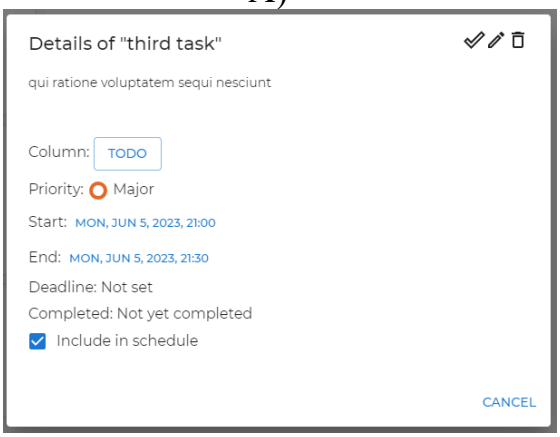
Рисунок 1 – Список з чотирьох задач



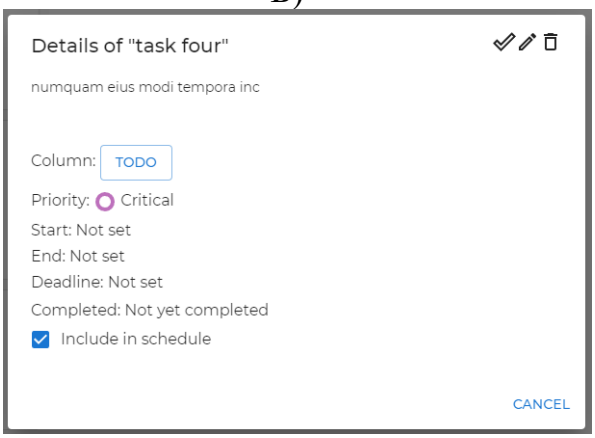
А)



Б)



В)



Г)

Рисунок 2 – Задачі з різним часом початку та кінця виконання: А) задано тільки години; Б) та Г) не задано години; В) задано і години і день виконання

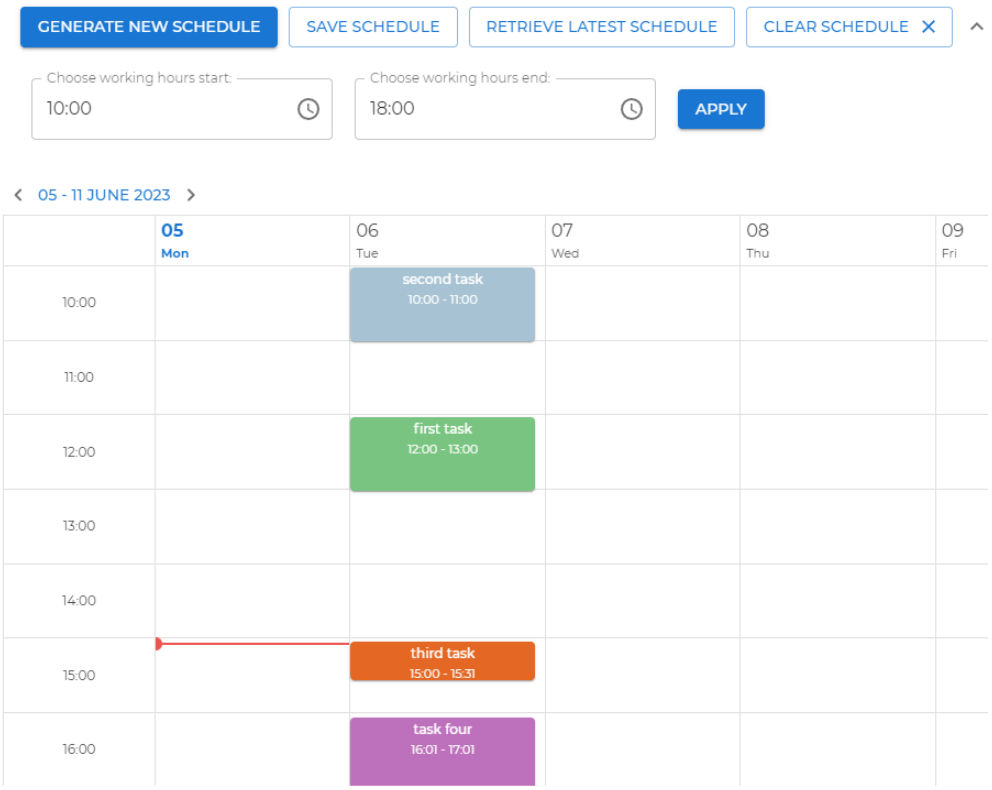


Рисунок 3 – Згенерований розклад на наступний день

На рисунку 3 можна побачити результат – усі 4 задачі були успішно заплановані на наступний день без накладень. Задача «second task» не мала зазначених годин виконання, тому була поставлена на початок дня. Задача «first task» мала інтервал з 12 по 13 годину, і, оскільки у цей день інтервал був вільним, задача була запланована на цей час без проблем. Аналогічним чином задач «third task», яка мала зазначені години та день. Задача «task four» не могла бути запланована після «first task», оскільки, за умовою, потрібно мати інтервал у 30 хв після задачі, а у такому випадку, перед «third task» не лишиться часу для цього інтервалу, тому вона була запланована після задачі «third task».

Часова складність алгоритму інтервального планування – O(n log n), де n представляє кількість інтервалів. Завдяки своїй ефективності алгоритм знаходить широке застосування в різних сценаріях планування та розподілу ресурсів. Даний алгоритм за бажанням можна доповнювати іншими керованими змінними: наявністю дедлайна, пріоритетністю задач, ресурсами.

Застосування цього підходу до планування полегшує управління завданнями, забезпечуючи відповідний розподіл ресурсів і своєчасне виконання завдань. Ці процеси додатково сприяють збалансованому плануванню та контролю виконання роботи, та мінімізації ризиків пов'язаних із затримками та невиконанням завдань. На основі запропонованого алгоритму розроблено зручний веб-застосунок для індивідуального планування, доступний за посиланням https://lotus-todo-app.web.app.

Список літературних джерел

1. *Brucker, P.: Scheduling Algorithms, 2nd edn. Springer, Heidelberg (2007)*
2. *The Ultimate Guide to Personal Productivity Methods. [Електронний ресурс]. Режим доступу – https://todoist.com/inspiration/personal-productivity-methods*
3. *Planning and Scheduling: Definitions and Differences. [Електронний ресурс]. Режим доступу – https://www.indeed.com/career-advice/career-development/planning-and-scheduling*

УДК 004.428 : 004.514

Ярош О. Л., студент 1 курсу
спеціальності 122 «Комп'ютерні науки» СО Магістр
Бабаков Р. М., д.т.н., доцент,
доцент кафедри інформаційних технологій

МЕТОДИ ОПТИМІЗАЦІЇ ПРОДУКТИВНОСТІ ВЕБ-ЗАСТОСУНКІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

Веб-інтерфейси API (Application Programming Interfaces) слугують основою сучасних веб-додатків, забезпечуючи взаємодію між різними програмними компонентами. Тому їхня продуктивність безпосередньо впливає на загальну продуктивність веб-додатків. У цій статті розглядаються основні