УДК 519.854.2

АЛГОРИТМИ ЛОКАЛЬНОГО ПОШУКУ ДЛЯ ОДНІЄЇ ЗАДАЧІ ПЛАНУВАННЯ ВИКОНАННЯ РОБІТ

Галкіна Г. А.

Національний технічний університет України “Київський політехнічний інститут імені Ігоря Сікорського”, Україна, Київ

Розглядається задача планування виконання робіт декількома пристроями. Характеристиками розглянутої задачі є наявність загального директивного строку для всіх пристроїв, різна непропорційна продуктивність пристроїв, наявність відношення часткового строгого впорядкування між роботами та визначена для кожної з робіт важливість. Наведено змістовну та математичну постановки задачі, що розглядається. Запропоновано два алгоритми локального пошуку для розв’язування цієї задачі: алгоритм детермінованого локального пошуку та алгоритм імітаційного відпалу. Наведено ключові аспекти реалізації для запропонованих алгоритмів, схеми алгоритмів та порівняння результатів їх роботи на 16 наборах тестових даних.

Ключові слова: оптимальне планування, розклад, різна продуктивність, директивний термін, паралельні пристрої, алгоритм детермінованого локального пошуку, алгоритм імітаційного відпалу.

Галкина Г. А. Алгоритмы локального поиска для одной задачи планирования выполнения работ / Национальный технический университет Украины «Киевский политехнический институт имени Игоря Сикорского», Украина, Киев

Рассматривается задача планирования выполнения работ несколькими устройствами. Характеристиками рассмотренной

задачи являются наличие общего директивного срока для всех устройств, разная непропорциональная продуктивность устройств, наличие отношения частичной строгой упорядоченности между работами и определённая для каждой из работ важность. Приведены смысловая и математическая постановки рассматриваемой задачи. Предложены два алгоритма локального поиска для решения этой задачи, алгоритм детерминированного локального поиска и алгоритм имитации отжига. Приведены ключевые аспекты реализации предложенных алгоритмов, схемы алгоритмов и сравнение результатов их работы на 16 различных наборах входных данных.

Ключевые слова: оптимальное планирование, расписание, разная продуктивность, директивный срок, параллельные устройства, алгоритм детерминированного локального поиска, алгоритм имитации отжига.

H. Halkina Local search algorithms for one optimal scheduling problem / National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, Ukraine, Kyiv

The problem of scheduling work for a set of machines is considered. The main characteristics of the problem include the existence of a common due date for all the machines, different unrelated productivity of the machines, the partial strict ordering set for the tasks and the value set for each of the tasks to be scheduled. The semantic and formal models of the problem are specified. Two local search algorithms are suggested: a deterministic local search algorithm and a simulated annealing algorithm. Key implementation points for each of the algorithms are covered; the algorithms’ as sequences of steps are described and a comparison of the suggested algorithms’ results when used on 16 input data sets is performed.

Keywords: optimal scheduling, schedule, different productivity, due date, parallel machines, deterministic local search, simulated annealing.

Вступ. Планування є невід'ємною частиною сучасного життя – це процес, який відбувається у майже всіх сферах людської діяльності.

У поточний час планування виконання робіт декількома паралельними пристроями є дуже розповсюдженою задачею. Існує багато різновидів цієї задачі, які визначаються її параметрами та обмеженнями. Одним із найпоширеніших параметрів є продуктивність пристроїв, в залежності від умов вона може бути однаковою, пропорційною або різною непропорційною. Також роботи можуть бути деяким чином впорядковані або ж розбиті на декілька частин, які можна виконувати на різних пристроях. Що стосується пристроїв, то для них можуть бути визначені час початку або закінчення роботи, періоди простою та інші параметри.

Кожна комбінація вищезгаданих та інших додаткових параметрів є окремою новою задачею, яку необхідно розв'язувати з урахуванням її особливостей. Саме тому розробка алгоритмів для конкретних варіацій задачі планування є корисною – практичні умови накладають різні види обмежень, і тому алгоритми, розраховані на деякі специфічні параметри, знайдуть своє застосування в тій або іншій сфері.

В цій статті розглянуто один із специфічних підвидів задач планування з кількома паралельними пристроями. В задачі, що розглядається, пристрої мають різну непропорційну продуктивність, всі роботи мають спільний директивний термін, а також вони є частково впорядкованими. Для кожної роботи визначено її важливість, і необхідно обрати такий набір робіт з усіх, що його сумарна важливість буде максимальною.

Така задача є розповсюдженою на практиці, як-то у методиці гнучкої розробки Скрам [1] та взагалі при командній роботі над проектами, наприклад, при підготовці проекту студентами під час навчання в університеті.

Аналіз останніх досліджень і публікацій. Наразі розв'язування задач планування є напрямком, в якому ведуться активні дослідження. Задачі планування з наявністю різної непропорційної продуктивності пристроїв та частковим впорядкуванням є менш розповсюдженими, ніж задачі планування з пропорційною продуктивністю пристроїв. Однією з робіт, що розглядає саме такі умови, є [2]. У [2] розглядається задача планування незалежних робіт з призначенням однакового директивного терміну на паралельних пристроях. В якості цільової функції у [2] виступає сумарні втрати, які залежать від обраного директивного терміну та від запізнення робіт, які не вкладаються в даний термін. В цій роботі було доведено, що сформульована задача є NP-складною при наявності лише двох пристроїв. Якщо ж кількість пристроїв є довільною, то задача стає NP-складною у сильному сенсі. [2] Задачі планування на паралельних машинах з різною непропорційною продуктивністю розглянуто у [3]. У [3] показано, що лише один вид цих задач є розв'язуваним за поліноміальний час шляхом зведення до лінійної задачі про призначення.

Задачі про оптимальне планування на пристроях з різною продуктивністю та відношенням часткового впорядкування було розглянуто у [4]. У публікації розглянуто два види часткового впорядкування, які названо І-впорядкування та АБО-впорядкування. Вони визначають відповідно роботи, які не можна розпочати до закінчення всіх їх попередників, та роботи, які не можна розпочати до закінчення хоча б одного з попередників. У випадку нашої задачі це

не грає ролі, оскільки у кожної з робіт може бути тільки один попередник. У [4] було показано, що задача планування робіт з відношенням часткового передування на кількох пристроях рівної продуктивності є NP-повною. Очевидно, що при розгляданні пристроїв різної продуктивності також отримаємо NP-повну задачу.

З наведеного вище можна зробити висновок, що розробка алгоритмів для такого формулювання задачі планування виконання робіт декількома паралельними пристроями має практичну цінність, оскільки через складність задачі знаходження точного розв'язку потребує великих витрат часу.

Мета та завдання статті. Метою статті є огляд запропонованих алгоритмів розв'язування задачі планування за наявності різної непропорційної продуктивності пристроїв та порівняння результатів їх роботи.

Для досягнення даної мети необхідно виконати наступні завдання:

– навести постановку та математичну модель задачі, що розв'язується;

– навести опис розроблених алгоритмів;

– провести аналіз результатів роботи розроблених алгоритмів на декількох наборах вхідних даних.

Постановка задачі. Для автоматизації процесу планування виконання робіт декількома пристроями протягом обмеженого проміжку часу необхідно формалізувати дану задачу. Наведемо її змістовну постановку.

Для зручності введемо позначення $N = \{1,..,n\}$.

Нехай є m пристроїв, цими пристроями може бути виконано n робіт. Кожна з цих робіт може виконуватися будь-яким пристроєм, але тільки одним.

Також роботи є частково упорядкованими, тобто є такі роботи $j, j \in N$, для яких задана деяка робота $k, k \in N \setminus \{j\}$, яку має бути виконано до них. Кожна робота може передувати або одній іншій роботі, або жодній.

Для кожного пристрою заданий час виконання кожної з робіт цим пристроєм $t_{ij}, i = \overline{1,m}, j = \overline{1,n}$.

Для кожної роботи визначено її важливість $v_j, j = \overline{1,n}$.

Заданий також директивний термін виконання підмножини робіт d.

Необхідно вибрати для виконання пристроями такий набір робіт, щоб сумарна важливість цього набору робіт була максимальною, та скласти розклад виконання цих робіт пристроями.

Математична модель

Задача полягає у максимізації цільової функції

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} v_j, \tag{1}$$

яка задає сумарну важливість усіх робіт, що будуть виконані, де

$$x_{ij} = \begin{cases} 1, \textit{якщо пристрій } i \textit{ виконує роботу } j, \\ 0, \textit{інакше} \end{cases}, i = \overline{1,m}, j = \overline{1,n}. \tag{2}$$

Також позначимо s_j час початку виконання роботи $j, j \in N$, причому:

$$s_j = \begin{cases} 0, \textit{якщо роботу } j \textit{ не буде виконано}, \\ > 0, \textit{якщо роботу } j \textit{ буде виконано}. \end{cases} \tag{3}$$

Обмеження задачі:

$$0 \le \sum_{i=1}^{m} x_{ij} \le 1, \tag{4}$$

Для всіх $j \in N, k \in N$ таких, що $x_{ij} = x_{ik}$ для всіх $i = \overline{1,m}$ та

$$\sum_{i=1}^{m} x_{ij} = 1, \sum_{i=1}^{m} x_{ik} = 1:$$

$$s_j < s_k \rightarrow s_j + t_{ij} < s_k. \quad (5)$$

Для всіх робіт $j, j \in N$ таких, що $s_j > 0$:

$$s_j + \sum_{i=1}^{m} x_{ij} t_{ij} \le d. \quad (6)$$

Для всіх $j, j \in N \setminus Q$, для яких $\sum_{i=1}^{m} x_{ij} = 1$ та є роботи k, що їм передують:

$$s_k < s_j,\ s_k + \sum_{i=1}^{m} x_{ik} t_{ij} < s_j. \quad (7)$$

Обмеження (4) задає можливість виконання кожної з робіт лише одним пристроєм. (5) – неперетинання запланованих робіт для одного пристрою, (6) задає виконання всіх робіт у розкладі до настання директивного терміну. Обмеження (7) задає виконання робіт відповідно до відношення часткового строгого впорядкування.

Ця задача є задачею комбінаторної оптимізації, оскільки простір припустимих розв’язків є простором розміщень робіт $i \in N$, і цей простір є обмеженим. Одним із найбільш розповсюджених видів алгоритмів для розв’язування таких задач є алгоритми локального пошуку, оскільки вони дозволяють за допомогою перегляду сусідніх розв’язків суттєво покращити деякий існуючий результат. Алгоритми детермінованого й стохастичного локального пошуку, незважаючи на відносну простоту своєї парадигми і тривалу історію, продовжують бути ефективним інструментом при створенні інформаційних технологій розв'язання багатьох практичних ЗКО як шляхом автономного застосування (особливо при розв'язанні задач підвищеної розмірності та складності), так і при використанні як

"будівельних" блоків при створенні сучасних метаевристик і гіперевристик, зокрема з розпаралелювання обчислювального процесу. [5]

Нехай деяким чином було знайдено припустимий розв'язок S'. Запропонуємо алгоритм детермінованого локального пошуку (ДЛП) та алгоритм імітаційного відпалу (АІВ) для розв'язування цієї задачі.

Алгоритм детермінованого локального пошуку.

Алгоритми детермінованого локального пошуку - це сім'я ітераційних методів, заснована на частковому перебиранні варіантів на кожній ітерації серед точок околу поточної точки, тобто серед сусідніх до неї. [5]

Ключовими аспектами реалізації алгоритмів детермінованого пошуку є:

1) визначення околів $O(x)$;

2) генерація чергової точки $y \in O(x)$;

3) критерій завершення перегляду точок у поточному околі та переходу до наступного;

4) спосіб обчислення величини зміни цільової функції при переході до нового поточного варіанта;

5) критерій завершення. [5]

Наведемо опис ключових аспектів реалізації алгоритму, що пропонується.

Визначення околів та генерація чергової точки околу

Використаємо кільцевий генератор точок околу, тобто такий, що продовжує свою роботу з попереднього місця замість перегляду всіх точок, які вже були переглянуті. Вважатимемо два розв'язки сусідніми, якщо вони розрізняються складом робіт на одну роботу, тобто розв'язок 1, який має набір робіт

$$K_1 = \{i \mid i \in N\},$$

та розв'язок 2, який має набір робіт

$$K_2 = K_1 \setminus \{r\} \cup \{j\}, j \in N \setminus K_1$$

є сусідніми.

Для процедури генерації сусіднього розв'язку введемо означення відстані зсуву вниз та відстані зсуву вверх.

Означення 1. Відстанню зсуву вниз d_{down} для деякої роботи $j, j \in N$, яку в поточному розв'язку буде виконане пристроєм $i, i \in [1,..,m]$, починаючи у момент часу $s_j > 0$ та за час t_{ij}, є проміжок часу, на який можна посунути ближче до закінчення директивного терміну всі роботи $k \in N \setminus \{j\} = K$, які у розкладі пристрою i будуть виконані після роботи j, тобто $x_{ik} = 1, s_k > s_j + t_{ij}$. Зрозуміло, що цей проміжок часу залежить від того, чи мають роботи $k \in K$ роботи, які заплановані після них, та чи мають вони роботи, які можуть бути виконані тільки після них. Позначимо роботу, яку можна виконати тільки після деякої роботи $k \in K$, як $succ(k), succ(k) \in N \setminus \{k\}$. Також позначимо роботу, яку заплановано на виконання після роботи $k \in K$, як $after_k, after_k \in N \setminus \{k\}$. Також позначимо роботу, яке заплановано на виконання до роботи $k \in K$, як $before_k, before_k \in N \setminus \{k\}$. Позначимо роботу, яку має бути виконано до деякої роботи $k \in K$, як $prev(k), prev(k) \in N \setminus \{k\}$. Проміжок часу для кожної роботи $k \in K$, окрім останньої у розкладі пристрою i, яка має $succ(k), succ(k) \in N \setminus \{k\}$, обчислюється таким чином:

$$d_{down}(k) = s_{succ(k)} - (s_k + t_{ik}).$$

Для останньої ж роботи $k_{last} \in K$, якщо вона має $succ(k), succ(k) \in N \setminus \{k\}$, цей проміжок часу обчислюється так:

$$d_{down}(k_{last}) = \min\{d - (s_{k_{last}} + t_{ik_{last}}), s_{succ(k_{last})} - (s_{k_{last}} + t_{ik_{last}})\},$$

де d – директивний термін.

Якщо ж робота $k \in K$ не має $succ(k), succ(k) \in N \setminus \{k\}$, то проміжок часу обчислюється таким чином:

$$d_{down}(k) = d - (s_k + t_{ik}).$$

Тоді загальна відстань зсуву вниз обчислюється наступним чином:

$$d_{down} = \min_{k \in K} \{d_{down}(k)\} \quad (8)$$

Звільнений для нової роботи проміжок часу буде обчислюватись таким чином: якщо є робота $after_j, after_j \in N \setminus \{j\}$ та робота $before_j, before_j \in N \setminus \{j\}$, то:

$$gap_{down} = (s_{after_j} + t_{iafter_j}) + d_{down} - s_{before_j}, \quad (9)$$

інакше, якщо є тільки робота $after_j, after_j \in N \setminus \{j\}$, то

$$gap_{down} = (s_{after_j} + t_{iafter_j}) + d_{down}, \quad (10)$$

інакше, якщо є тільки робота $before_j, before_j \in N \setminus \{j\}$, то

$$gap_{down} = d - s_{before_j}, \quad (11)$$

інакше:

$$gap_{down} = d. \quad (12)$$

Тоді час початку нової доданої роботи буде обчислюватись так: якщо є робота $before_j, before_j \in N \setminus \{j\}$ та $prev(j), prev(j) \in N \setminus \{j\}$, то:

$$s_k = \max\{s_{before_j} + t_{ibefore_j}, s_{prev(j)} + t_{iprev(j)}\}, \quad (13)$$

інакше, якщо є лише $before_j, before_j \in N \setminus \{j\}$, то

$$s_k = s_{before_j} + t_{ibefore_j}, \quad (14)$$

інакше, якщо є лише $prev(j), prev(j) \in N \setminus \{j\}$, то

$$s_k = s_{prev(j)} + t_{iprev(j)}, \quad (15)$$

інакше:

$$s_k = 0. \quad (16)$$

Означення 2. Відстанню зсуву вверх d_{up} для деякої роботи $j, j \in N$, яку в поточному розв'язку буде виконано пристроєм $i, i \in [1,..,m]$, починаючи у момент часу $s_j > 0$ та за час t_{ij}, є проміжок часу, на який можна посунути ближче до моменту початку цієї роботи всі роботи $k \in N \setminus \{j\} = K$, які у розкладі пристрою i будуть виконані після роботи j, тобто $x_{ik} = 1, s_k > s_j + t_{ij}$, якщо цю роботу $j, j \in N$ вилучити з поточного розкладу для пристрою i. Зрозуміло, що цей проміжок часу залежить від того, чи мають роботи $k \in K$ роботи, які заплановані до них, та чи мають вони роботи, які мають їм передувати. Проміжок часу для кожної роботи $k \in K$, окрім першого у розкладі пристрою i, яка має $prev(k), prev(k) \in N \setminus \{k\}$, обчислюється таким чином:

$$d_{up}(k) = \min\{s_k - (s_{before_k} + t_{ibefore_k}), s_k - (s_{prev(k)} + t_{iprev(k)})\}.$$

Для першої ж роботи $k_{first} \in K$, якщо вона має $prev(k), prev(k) \in N \setminus \{k\}$, цей проміжок часу обчислюється так:

$$d_{up}(k_{first}) = s_{k_{first}} - (s_{prev(k_{first})} + t_{iprev(k_{first})}).$$

Якщо ж робота $k \in K$ не має $prev(k), prev(k) \in N \setminus \{k\}$, то проміжок часу для всіх робіт, окрім першої, обчислюється таким чином:

$$d_{up}(k) = s_k - (s_{before_k} + t_{ibefore_k}),$$

для першої ж роботи $k_{first} \in K$ формула має такий вигляд:

$$d_{up}(k_{first}) = s_{k_{first}}.$$

Тоді загальна відстань зсуву вверх обчислюється наступним чином:

$$d_{up} = \min_{k \in K}\{d_{up_k}\}. \quad (17)$$

Тоді час початку нової доданої роботи буде обчислюватись так:

якщо є роботи $prev(j), prev(j) \in N \setminus \{j\}$ та $k_{last} \in K$, то:

$$s_k = \max\{s_{k_{last}} + t_{ik_{last}}, s_{prev(j)} + t_{iprev(j)}\}, \quad (18)$$

інакше, якщо є лише $k_{last} \in K$, то

$$s_k = s_{k_{last}} + t_{ik_{last}}, \quad (19)$$

інакше, якщо є лише $prev(j), prev(j) \in N \setminus \{j\}$, то

$$s_k = s_{prev(j)} + t_{iprev(j)}, \quad (20)$$

інакше:

$$s_k = 0. \quad (21)$$

Наведемо схему процедури генерації сусіднього розв'язку.

Процедура 1 (генерація сусіднього розв'яку).

Крок 1. Доки не знайдено припустимий новий розв'язок або не переглянуто всі роботи у поточному розв'язку:

1.1. ЯКЩО множина переглянутих робіт у поточному розкладі є порожньою,

ТО вважати першу роботу першого пристрою поточною

ІНАКШЕ вважати наступну після останньої переглянутої роботи поточною

1.2. Вважати, що ще не виконувались зсув вверх та зсув вниз;

1.3. ЯКЩО для поточної роботи j визначено роботу $succ(j), succ(j) \in N \setminus \{j\}$ та вона є у поточному розкладі, ТО перейти до 1.1.

ІНАКШЕ перейти до 1.4.

1.4. Вилучити поточну роботу j з поточного розв'язку.

1.5. ЯКЩО у поточної роботи є $succ(j), succ(j) \in N \setminus \{j\}$ та ще не виконувався зсув вверх,

ТО перейти до 1.6.

ІНАКШЕ,ЯКЩО ще не виконувався зсув вниз, ТО перейти

до1.7.

ІНАКШЕ, ЯКЩО ще не виконувався зсув вверх, ТО перейти до 1.6.

ІНАКШЕ перейти до 1.1.

1.6. Спробувати виконати зсув вверх:

1.6.1. Обчислити відстань зсуву вверх d_{up} за (17).

1.6.2. Знайти таку роботу $k \in N$ з тих, які не у поточному розв'язку, що її важливість максимальна та $t_{ik} \le d - (s_{k_{last}} + t_{ik_{last}}) + d_{up}$.

1.6.3. ЯКЩО є така робота $k \in N$, ТО додати її до поточного розв'язку до розкладу поточного пристрою з часом початку s_k, обчисленим за формулами (18) – (21);

Вважати зсув вверх виконаним;

Вважати отриманий розв'язок новим сусіднім розв'язком;

Перейти до кроку 1.

ІНАКШЕ

Вважати зсув вверх виконаним;

Перейти до 1.5.

1.7. Спробувати виконати зсув вниз:

1.7.1. Обчислити відстань зсуву вниз d_{down} за (8).

1.7.2. Знайти таку роботу $k \in N$ з тих, які не у поточному розв'язку, що її важливість максимальна та $t_{ik} \le gap_{down}$, де gap_{down} обчислюється за формулами (9) – (12).

1.7.3. ЯКЩО є така робота $k \in N$, ТО додати її до поточного розв'язку до розкладу поточного пристрою з часом початку s_k, обчисленим за формулами (13) – (16);

Вважати зсув вниз виконаним;

Вважати отриманий розв'язок новим сусіднім розв'язком;

Перейти до кроку 1.

ІНАКШЕ

Вважати зсув вниз виконаним;

Перейти до 1.5.

Критерій завершення перегляду точок у поточному околі та переходу до наступного

Перехід до наступного розв'язку здійснюється при знаходженні першого кращого за поточний розв'язок сусіднього розв'язку.

Спосіб обчислення зміни цільової функції при переході до нового поточного варіанта

Нехай з поточного розв'язку було вилучено деяку роботу $i, i \in N$ та її було замінено роботою $j, j \in N \setminus \{i\}$. Тоді зміна цільової функції обчислюється таким чином:

$$\Delta = v_j - v_i \tag{22}$$

Критерій завершення

В якості критерію завершення обрано відсутність кращих варіантів в околі поточного розв'язку або перевищення заданої максимальної кількості ітерацій.

З урахуванням наведеного вище сформулюємо схему алгоритму ДЛП для задачі планування робіт, що розглядається.

Крок 1. Вважати початковий розв'язок S' поточним.

Крок 2. Доки окіл поточного розв'язку не переглянуто повністю та не перевищено кількість проведених ітерацій,

2.1.Сформулювати наступний розв'язок з околу поточного розв'язку за процедурою 1.

2.2. ЯКЩО сформований розв'язок має більше значення цільової функції за значення поточного розв'язку,

ТО вважати сформований розв'язок поточним;

2.3. Перейти на крок 2.

Алгоритм імітаційного відпалу.

Алгоритм детермінованого локального пошуку дозволяє перехід тільки до кращих розв'язків, що обмежує його результати околом початкового розв'язку. Для того, щоб уникнути цього обмеження, можна використати алгоритми стохастичного локального пошуку, в яких за деяких умов дозволяється перехід до гіршого розв'язку.

Саме така ідея лягла в основу алгоритмів *імітаційного,* або *модельованого, відпалу* (AIB) (Simulatedannealing), які базуються на аналогії з процесом оптимізації та фізичним процесом відпалу. [5]

Наведемо ключові аспекти реалізації для алгоритмів імітаційного відпалу.

1) Поняття околу та перебір точок в околі, тобто спосіб генерації точок.

2) Імовірність переходу від поточного варіанта до нової точки.

3) Принцип рівноваги: задаються параметри $\varepsilon > 0$ та v – натуральне, а v переходів у алгоритмі називається *прогоном.* Тоді, якщо здійснюється k прогонів і маємо значення $f_1,...,f_k$ (характеристики цільових функцій на цих прогонах), то на $(k+1)$-му прогоні вважається досягнутою рівновага, якщо

$$\exists i = \overline{1,k} : \left| f_i - f_{k+1} \right| / f_i \le \varepsilon \qquad (23)$$

4) Температурний розклад (правило зміни значення параметра T). [5]

Опишемо ці аспекти для алгоритму, що пропонується.

Поняття околу та перебір точок в околі, тобто спосіб генерації точок

Для перебору точок в околі використовується процедура 1.

Імовірність переходу від поточного варіанта до нової точки

Нехай для розв'язку S знайдено деякий припустимий сусідній

розв'язок S_{new}. Тоді спочатку обчислюється різниця цільових функцій Δ за формулою (22). Після цього генерується випадкове число $r = random[0,1]$. Тоді імовірність переходу до нового розв'язку обчислюється так:

$$p = e^{\frac{-\Delta}{T}}. \tag{24}$$

Якщо $r < \min\{p,1\}$, то перехід відбувається.

Принцип рівноваги

Нехай є початковий розв'язок S_{start}, і в ньому $n_{S_{start}}$ робіт. Тоді довжина прогону визначається таким чином:

$$length = \frac{n_{S_{start}}}{m}. \tag{25}$$

А f_i обчислюється для прогону як довжиною $length$ таким чином:

$$f_i = \frac{\sum_{w=1}^{length} f_w}{length} \tag{26}$$

Температурний розклад (правило зміни значення параметра T)

Зміна температури відбувається від значення $T_{\max} = 1$ до значення $T_{\min} = 0.00001$. Наступне значення параметру T_r на ітерації під номером r обчислюється за формулою

$$T_r = T_{r-1} \cdot \alpha, \tag{27}$$

де $\alpha = 0.9$.

Крок 1. Вважати початковий розв'язок S' поточним.

Крок 2. Обчислити необхідну кількість ітерацій для одного прогону за формулою (25).

Крок 3. Доки температура не досягла мінімально припустимого значення $T_{\min}$:

3.1.Доки не досягнуто балансу за умовами (26) та (23):

3.1.1. Доки не проведено визначену кількість ітерацій r для прогону:

3.1.1.1. Згенерувати новий розв'язок з околу поточного розв'язку;

3.1.1.2. ЯКЩО новий розв'язок має більше значення цільової функції за поточний розв'язок

ТО вважати новий розв'язок поточним

Перейти до 3.1.1;

3.1.1.3. Розрахувати імовірність переходу до гіршого розв'язку $p \in [0,1]$ за формулою (24);

3.1.1.4. Згенерувати випадкове число $t \in [0,1]$.

3.1.1.5. ЯКЩО $t < p$

ТО вважати новий розв'язок поточним

Перейти до 3.1.1

ІНАКШЕ перейти до 3.1.1.1

3.1.2. Перейти до 3.1.

3.2. Обчислити нове значення температури за формулою (27).

Аналіз результатів роботи алгоритмів.

У таблиці 1 наведено результати проведених експериментів. Для AIB було обрано кращий результат з 50 запусків для кожного набору вхідних даних.

Таблиця 1

Результати роботи алгоритмів ДЛП та АІВ

№	m	N	d	$start$	$best_{ДЛП}$	$\frac{best_{ДЛП}}{start}$	$best_{AIB}$	$\frac{best_{AIB}}{start}$
1	7	116	40	1217	2069	1.7	2090	1.72
2	9	55	40	819	1106	1.35	1131	1.38
3	9	67	40	789	1137	1.44	1159	1.47
4	7	88	40	701	1132	1.61	1139	1.63
5	5	65	80	116	167	1.44	166	1.43
6	7	141	80	1119	1763	1.58	1776	1.59
7	9	89	80	627	1158	1.85	1176	1.88
8	10	65	80	921	1245	1.35	1303	1.41
9	7	133	120	1149	2222	1.93	2252	1.96
10	8	87	120	1713	2255	1.32	2284	1.33
11	8	138	120	1732	2525	1.46	2551	1.47
12	10	67	120	1721	2491	1.45	2491	1.45
13	3	92	160	222	390	1.76	379	1.7
14	5	142	160	530	1231	2.32	1247	2.35
15	6	28	160	122	139	1.14	144	1.18
16	9	60	160	713	1080	1.51	1125	1.58

З даних таблиці 1 можна зробити висновок, що в загальному алгоритми детермінованого локального пошуку та імітаційного відпалу дають схожі результати, але в більшості випадків результати останнього є трохи кращими.

Висновки. Наведено постановку задачі планування за умови наявності різної непропорційної продуктивності пристроїв та часткового впорядкування робіт. Запропоновано два алгоритми розв'язування поставленої задачі: алгоритм детермінованого локального пошуку та алгоритм імітаційного відпалу. Наведено результати розв'язування 16 екземплярів задачі обома алгоритмами. Алгоритм імітаційного відпалу в загальному дозволяє отримати кращі результати, ніж алгоритм детермінованого локального пошуку.

Література:

1. Посібник зі Скраму [Електронний ресурс] / К. Швабер, Д. Сазерленд – Режим доступу до ресурсу: https://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-UA.pdf

2. De P. Due-date assignment and early / tardy scheduling on identical parallel machines / P. De, J. B. Ghosh, C. E. Wells // Naval Research Logistics. – 1994. – №1. – P.17-32.

3. Brucker P. Scheduling Algorithms / P. Brucker. – New York: Springer Publishing, 2007. – 371 p.

4. Gillies D. W. Scheduling Tasks with AND/OR Precedence Constraints / D. W. Gillies, J. W.-S. Liu. // SIAM Journal on Computing. – 1995. – №4. – P.797-810.

5. Гуляницький Л.Ф. Прикладні методи комбінаторної оптимізації: навч. посіб. / Л. Ф. Гуляницький, О. Ю. Мулеса. – К.: Видавничо-поліграфічний центр "Київський університет", 2016. – 142 с.

References:

1. The Scrum Guide [Elektronnyj resurs] / K. Schwaber, J. Sutherland – Rezhym dostupu do resursu: https://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-UA.pdf

2. De P. Due-date assignment and early / tardy scheduling on identical parallel machines / P. De, J. B. Ghosh, C. E. Wells // Naval Research Logistics. – 1994. – №1. – P.17-32.

3. Brucker P. Scheduling Algorithms / P. Brucker. – New York: Springer Publishing, 2007. – 371 p.

4. Gillies D. W. Scheduling Tasks with AND/OR Precedence Constraints / D. W. Gillies, J.W.-S. Liu. // SIAM Journal on Computing. – 1995. – №4. – P.797-810.

5. Guljanyc'kyj L.F. Prykladni metody kombinatornoi' optymizacii': navch. posib. / L. F. Guljanyc'kyj, O. Ju. Mulesa. – K.: Vydavnycho-poligrafichnyj centr "Kyi'vs'kyj universytet", 2016. – 142 s.