# Лабораторна робота №2

## Вирішення задачі 0-1 Knapsack problem за допомогою алгоритма Tabu Search

Умова задачі. Для заданої множини предметів *N*, кожен з яких має вагу *wi* і цінність *c*i, визначити яку кількість кожного з предметів слід взяти, так, щоб сумарна вага не перевищувала задану *Cap*, а сумарна цінність *Profit* була максимальною:

Вирішенням задачі є бінарний вектор заповнення X = { x1, x2, …, xn }, та значення Profit, якого вдалось досягти.

Визначаємо цільову функцію для алгоритму TabuSearch.

Якщо P(X) – це вартість предметів для певного набору X

То цільова функція F(X) має вигляд:

При вирішенні задачі використовувався 1-flip окіл, який для певного бінарного вектора X є множина векторів {Y0, Y1, …, Yn}, у яких 1 біт буде відрізнятись. (N векторів в околі)

Реалізація алгоритма мовою С++ знаходиться за посиланням https://github.com/ospodinets/knapsackhc/tree/master/ts

Був обраний варіант з двома типами пам’яті - короткостроковою, та тривалою. Короткострокова пам’ять зберігає табу впродовж 5 ітерацій на один біт поточного стану вектора рішень. Довгострокова пам’ять накопичує кількість переходів для кожного біта, який потім з коефіцієнтом 0.7 погіршує значення цільової функції. Якщо під час обробки всіх значень з околу певної точки не знайдено найкращого значення (Перевищений ліміт рюкзаку, оптимальні точки табуйовані, чи інше), відбувається очищення обох типів пам’яті і алгоритм продовжує виконання з поточної точки.

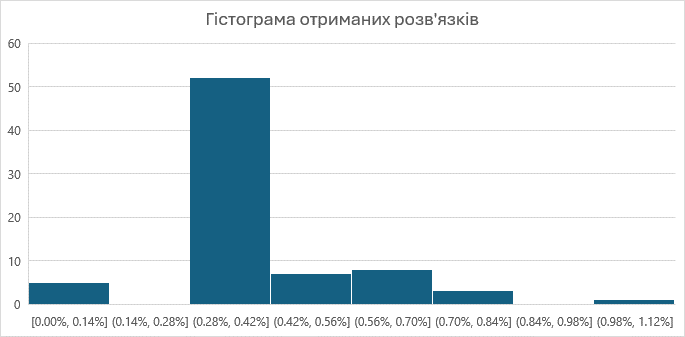
Вхідні дані та еталонний розв’язок згенеровано за допомогою відповідних утіліт, розроблених David Pisinger: http://hjemmesider.diku.dk/~pisinger/codes.html

Серед різних варіантів тестових даних було обрано набір зі 100 елементів максимальною вартістю і вагою 100 одиниць.

Було проведено 100 випробувань, дані випробувань викладено у наступних графіках.

Еталонний результат, 2671, він же найкращій, отриманий алгоритмом один раз.

Найгірший результат – 2640, на 1.16% відхиляється від еталонного



Отримані розв’язки розподілені таким чином.

Графік значень цільової функції