# Projekt z przedmiotu - Badania operacyjne i logistyka

| Kierunek studiów: Informatyka stosowana | Semestr: 6, grupa: 1, zespół: 9 |
|---|---|
| Imię i nazwisko I: Krzysztof Jarek<br>Imię i nazwisko II: Krzysztof Klimczyk | |

## 1. Zagadnienie pośrednika

### 1.1. Temat projektu:

Aplikacja desktopowa z interfejsem użytkownika i wizualizacją danych. Dodatkowo nie występuje blokowanie tras, a rozmiar problemu jest sztywno określony i wynosi dwóch dostawców i dwóch odbiorców.

### 1.2. Opis metody:

Na początku pobierane są wielkości dostawy i odbioru obu dostawców i obu odbiorów, wartości zakupów i sprzedaży, wielkości poszczególnych kosztów transportowych. Z tak pozyskanymi danymi pozyskiwana jest funkcja celu, dzięki której następnie możliwe jest dokonywanie maksymalizacji zysków ze sprzedaży. Jako wynik otrzymywane są całkowity koszt, całkowity przychód i zysk, jako różnica z nich.

### 1.3. Opis użytych narzędzi informatycznych:

Aplikacja została zbudowana w Javie. Do stworzenia interfejsu graficznego został użyty pakiet JavaFX. Do rozwiązywania problemu pośrednik został wykorzystany przebudowany przez nas algorytm stworzony przez użytkownika o nazwie *isstaif*, z serwisu GitHub.

### 1.4. Najważniejsze fragmenty kodu:

Metoda *maximumProfitRule()* jest odpowiedzialna za rozwiązanie zagadnienia pośrednika. Główna część obliczeń dokonuje się wewnątrz pętli *while()*, gdzie są dobierane wartości na kandydatów do zrealizowania maksymalizacji dochodów, dzięki odpowiedniemu liczeniu i wybieraniu wartości (głównie w wewnętrznych pętlach *for()*).

```java
public double maximumProfitRule() {
    calculateProfits();

    long start = System.nanoTime();

    double min;
    int k = 0; //feasible solutions counter

    //isSet is responsible for annotating cells that have been allocated
    boolean[][] isSet = new boolean[stockSize][requiredSize];
    for (int j = 0; j < requiredSize; j++)
        for (int i = 0; i < stockSize; i++)
            isSet[i][j] = false;

    int i = 0, j = 0;
    Variable maxCost = new Variable();

    //this will loop is responsible for candidating cells by their least cost
    while (k < (stockSize + requiredSize - 1)) {

        maxCost.setValue(-Double.MAX_VALUE);
        //picking up the least cost cell
        for (int m = 0; m < stockSize; m++)
            for (int n = 0; n < requiredSize; n++)
                if (!isSet[m][n])
                    if (unit_profit[m][n] > maxCost.getValue()) {
                        maxCost.setStock(m);
                        maxCost.setRequired(n);
                        maxCost.setValue(unit_profit[m][n]);
                    }
```

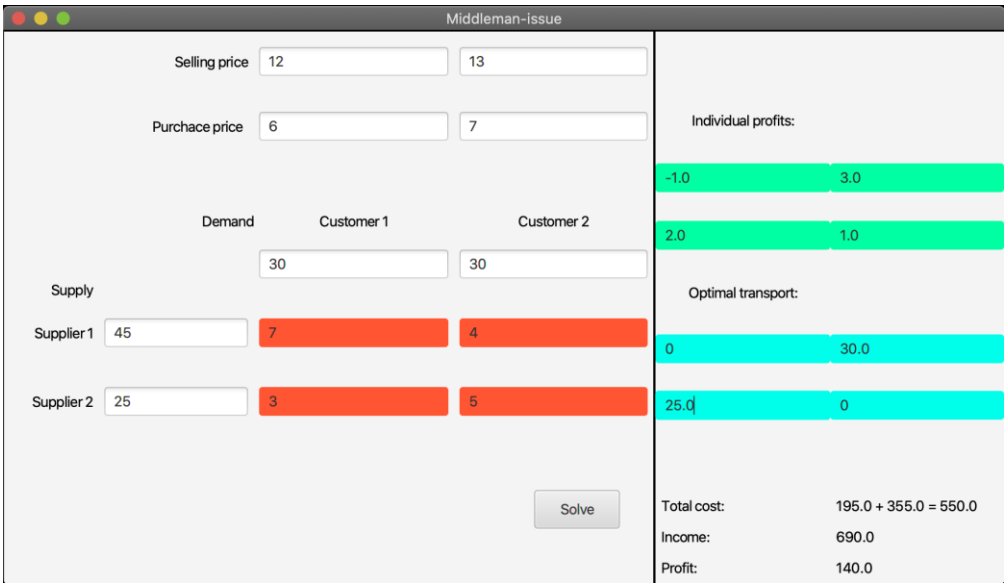
```java
        i = maxCost.getStock();
        j = maxCost.getRequired();

        //allocating stock in the proper manner
        min = Math.min(required[j], stock[i]);

        feasible.get(k).setRequired(j);
        feasible.get(k).setStock(i);
        feasible.get(k).setValue(min);
        k++;

        required[j] -= min;
        stock[i] -= min;

        //allocating null values in the removed row/column
        if (stock[i] == 0)
            for (int l = 0; l < requiredSize; l++)
                isSet[i][l] = true;
        else
            for (int l = 0; l < stockSize; l++)
                isSet[l][j] = true;

    }

    return (System.nanoTime() - start) * 1.0e-9;

}
```

### 1.5. Przykładowe rozwiązanie:

1. Pośrednik kupuje towar od dwóch dostawców i przesyła do dwóch odbiorców. Podaż dostawców (w tonach), popyt odbiorców (w tonach), jednostkowe ceny zakupu, sprzedaży i koszty transportu przedstawia tabela.

| Podaż dostawców | Popyt odbiorców | | |
|---|---|---|---|
| | 30 | 30 | Cena zakupu |
| 45 | 7 | 4 | 6 |
| 25 | 3 | 5 | 7 |
| Cena sprzedaży | 12 | 13 | |

a) Zapisz to zadanie w postaci tablicy transportowej.
b) Ustal optymalny plan dostaw.
c) Oblicz przychód, koszt zakupu, koszt transportu, koszt magazynowania oraz dochód pośrednika.




# 2. Programowanie liniowe - dobór optymalnego asortymentu produkcji

### 2.1. Temat projektu:

Aplikacja desktopowa z interfejsem użytkownika i wizualizacją danych. Rozmiar problemu jest dowolny.

### 2.2. Opis metody:

Na samym początku definiowany jest rozmiar problemu czyli liczba nakładów i środków produkcji. Później definiowana jest funkcja celu. W tym przypadku jest to

maksymalizacja przychodów ze sprzedaży. Poszukiwanymi wartościami są rozmiary produkcji poszczególnych nakładów. Następnie do określonego rozmiaru problemu tworzone są ograniczenia liniowe (dowolne, indywidualne ograniczenia nakładów i limity na środki produkcji).

## 2.3. Opis użytych narzędzi informatycznych:

Aplikacja została zbudowana w Javie. Do stworzenia interfejsu graficznego został użyty pakiet JavaFX. Za optymalizacje odpowiada algorytm simplex znajdujący się w bibliotece o nazwie SSC służącej do rozwiązywania problemów liniowych.

## 2.4. Najważniejsze fragmenty kodu:

Fragment kodu pod komentarzem ”goal function” odpowiadający za zdefiniowanie funkcji celu. Algorytm zawsze poszukuje maksymalnej wartości zysku. Kod pod komentarzem ”material constaint” odpowiada za stworzenie ograniczeń na jednostkowe nakłady.

```java
public void buttonSolve() throws Exception {
    ArrayList<String> constraints = new ArrayList<String>();

    //goal function
    String goalFunction = "max: ";
    for (Problem p : problemList) {
        double c = p.getPrice();
        goalFunction += c + p.getName() + " + ";
    }
    goalFunction = goalFunction.substring(0, goalFunction.length() - 3);
    System.out.println(goalFunction);
    constraints.add(goalFunction);

    //material constains
    for (Restriction r : restrictionList) {
        Object sign = r.getComboBoxSign().getSelectionModel().getSelectedItem();
        String res = "";
        if (r.getLimit().matches( regex: "^[0-9.]*$")) {
            res += r.getVariable() + sign + r.getLimit();
        } else {
            res += r.getVariable() + "-" + r.getLimit() + sign + "0";
        }
        System.out.println(res);
        constraints.add(res);
    }
```

Fragment kodu pod komentarzem ”stock constaint” odpowiada za ustawienie ograniczeń na środki produkcji. Pozostały kod to wizualizacja otrzymanych wyników czyli wyświetlenie wartości funkcji celu i optymalnych rozmiarów produkcji.

```
    //stock constains
    for (Stock s : stockList) {
        String sto = "";
        for (UnitOutlay u : s.getUnitOutlayList()) {
            sto += u.getValue() + u.getName() + " + ";
        }
        sto = sto.substring(0, sto.length() - 3);
        sto += " <= " + s.getMaxProduction();
        System.out.println(sto);
        constraints.add(sto);
    }

    LP lp = new LP(constraints);
    SolutionType solution_type = lp.resolve();

    if (solution_type == SolutionType.OPTIMUM) {
        Solution soluzione = lp.getSolution();
        for (Variable var : soluzione.getVariables()) {
            for (Problem p : problemList) {
                if (var.getName().equals(p.getName())) {
                    p.setAmount(var.getValue());
                    break;
                }
            }
            SscLogger.log("Variable name :" + var.getName() + " value :" + var.getValue());
        }
        income.setText(String.valueOf(soluzione.getOptimumValue()));
        SscLogger.log("Value:" + soluzione.getOptimumValue());
    }
    tableViewResult.setItems(problemList);
    tableViewResult.refresh();
}
```

## 2.5. Przykładowe rozwiązanie:

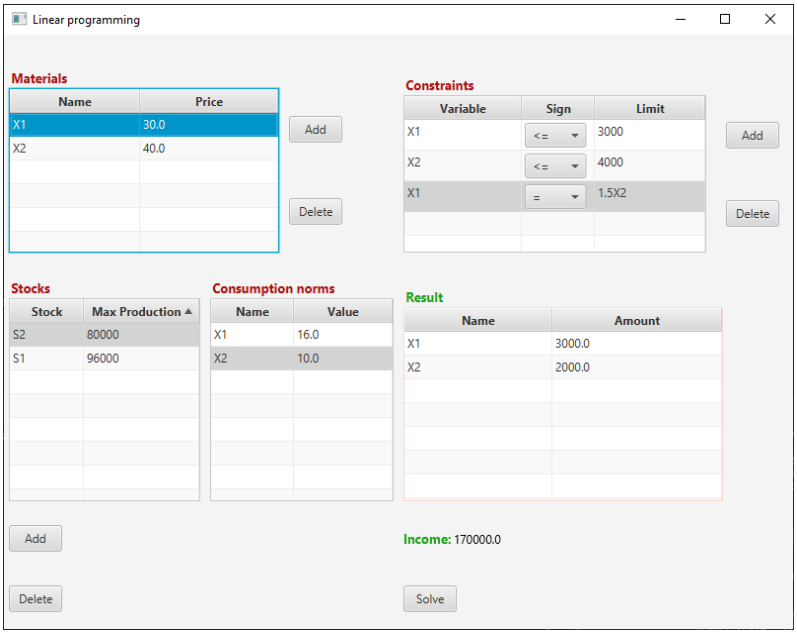
**Zadanie 1.**

Kuźnia produkuje dwa rodzaje wyrobów: *W1* i *W2*. W procesie produkcji tych wyrobów zużywa się wiele środków, spośród których dwa są limitowane. Limity te wynoszą: środek I - 96 000 jednostek, natomiast środek II - 80 000 jednostek. Nakłady limitowanych środków na jednostkę wyrobów *W1* i *W2* podano w poniższej tabeli.

| Środki produkcji | Jednostkowe nakłady | |
|---|---|---|
| | $W_1$ | $W_2$ |
| I | 16 | 24 |
| II | 16 | 10 |

Wiadomo także, że zdolności produkcyjne jednej z pras stanowią wąskie gardło procesu produkcyjnego, nie pozwalają kuc więcej niż 3000 szt. wyrobów *W1* oraz 4000 szt. wyrobów *W2*.

Ponadto, działająca w ramach kuźni komórka analizy rynku ustaliła optymalne proporcje produkcji, które kształtują się odpowiednio jak 3:2. Cena sprzedaży (w zł) jednostki wyrobu *W1* wynosi 30, a wyrobu *W2* - 40. Należy ustalić optymalne rozmiary produkcji gwarantujące maksymalizację przychodu ze sprzedaży przy istniejących ograniczeniach.


Linear programming
Materials
Name
Price
X1
30.0
X2
40.0
Add
Delete
Constraints
Variable
Sign
Limit
X1
<=
3000
X2
<=
4000
X1
=
1.5X2
Add
Delete
Stocks
Stock
Max Production
S2
80000
S1
96000
Add
Delete
Consumption norms
Name
Value
X1
16.0
X2
10.0
Result
Name
Amount
X1
3000.0
X2
2000.0
Income: 170000.0
Solve