Łazik marsjański
Raport przebiegu rozwiązania metodą wyżarzania oraz tabu-search

Majewski Maciej | Kaźmierczak Mikołaj

1. Opis problemu

Problem dotyczy marsjańskiego łazika, którego zadaniem jest zbieranie próbek znajdujących się na Marsie. Robot stacjonuje w bazie, w której składuje zbierane próbki oraz ładuje swoje baterie. Aby uniknąć przypadku gdy zabraknie mu prądu jego wyprawa musi zakończyć się w bazie.

Robot charakteryzuje się dwoma cechami: **ładownością** oraz **zasięgiem**.
Próbki mają **różne wartości** ze względu na ich rodzaj materiału. Wymagane jest **zebranie minimum jednej próbki każdego rodzaju**. Powierzchnia Marsa nie jest płaska dlatego koszt przejazdu z jednego punktu do innego może być inny dla tej samej ścieżki w drugą stronę.

Celem łazika jest maksymalizowanie wartości załadowanego towaru, wyznaczając przy tym optymalną ścieżkę od bazy przez próbki i z powrotem do bazy.

Oznaczenia:

1. **Robot**
- ❏ Ładowność - **c** [kg]
- ❏ Zasięg - **r** [m]

2. **Próbki**
- ❏ Liczba próbek - **n**
- ❏ Indeks próbki - **i,j = 1,2,…,n**
- ❏ Masa - $\mathbf{m_i}$ [kg]
- ❏ Kategoria, wartość elementu - $\mathbf{w_i}$
- ❏ Pozycja próbki - **p(a,b)**
- ❏ Koszt przemieszczenia się robota między próbką **i** a **j** - $\mathbf{c_{ij}}$
- ❏ Wektor kosztów przemieszczenia - $\mathbf{c = [c_{ij}]_{i=1,n;j=1,n}}$
- ❏ Oznaczenie czy robot przemieszcza się z punktu **i** do **j** - $\mathbf{y_{ij} = \{1,0\}}$
- ❏ Ścieżka robota - $\mathbf{y = [y_{ij}]_{i=1,n;j=1,n}}$
- ❏ Oznaczenie próbki, która została załadowana do robota - $\mathbf{x_i = \{1,0\}}$
- ❏ Wektor załadowanych próbek - $\mathbf{x = [x_i]_{i=1,n}}$
- ❏ Wektor kategorii próbek **k**, o długości **l**

Ograniczenia:

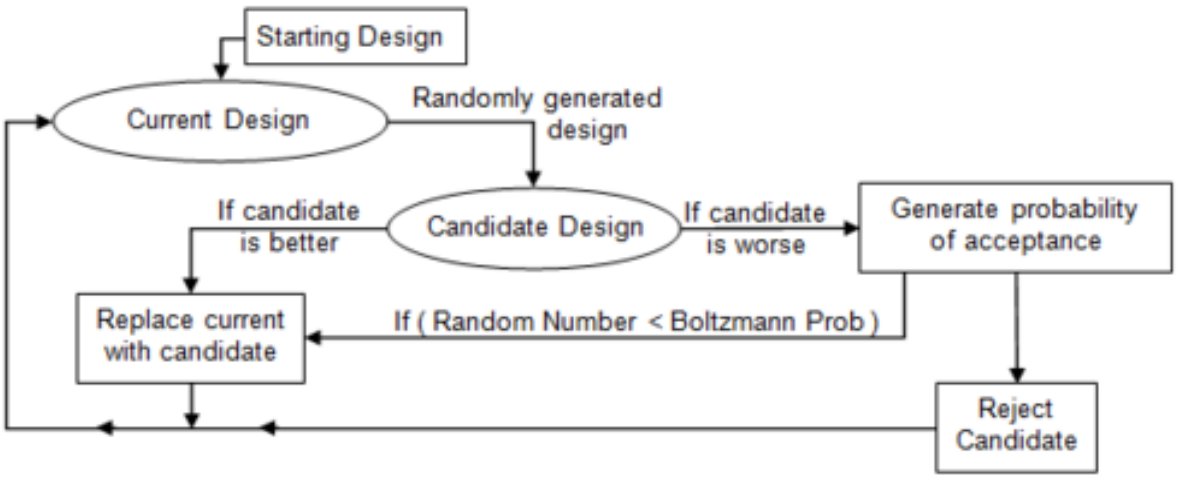
1. $\sum_{i=1}^{n} m_i x_i < C$ - ograniczenie pojemności robota
2. $x_i \in \{0,1\}, i = 1,2,...,n$
3. $\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} y_{ij} < r$ - ograniczenie zasięgu robota
4. $\sum_{j} 1(\sum_{i} x_i k_{ij})$ =1 - ograniczenie min 1 próbki w każdej kategorii
5. $\sum_{j=0} y_{0j} = 1$ oraz $\sum_{j=0} y_{j0} = 1$ - wymagania powrotu robota do bazy

Funkcja celu:

$Q(x) = \sum_{i=1}^{n} w_i x_i$ - zadanie maksymalizacji czyli $x^* = argmax_x Q(x)$

2. Podejście metodą wyżarzania

Symulowane wyżarzanie jest algorytmem heurystycznym służącym do przeszukiwania rozwiązań problemu w celu znalezienia jak najlepszego rozwiązania.
Sposób działania algorytmu przedstawia poniższy graf:

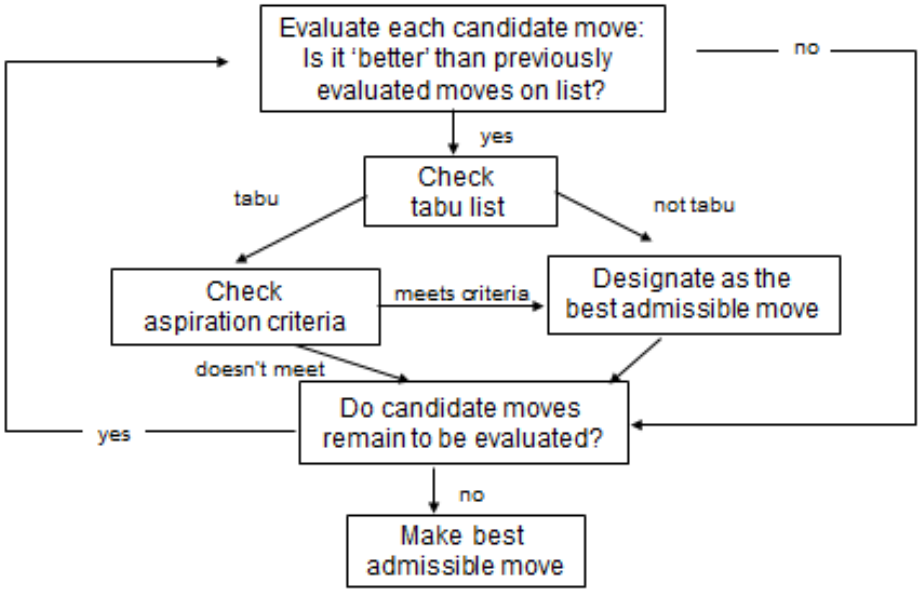


W przypadku naszego problemu początkowym rozwiązaniem jest prosta ścieżka przejścia po mapie,która zapewnia zebranie kilku wylosowanych próbek. Następnie rozpoczynamy proces wyżarzania rozpoczynając od początkowego rozwiązania. Następny kandydat jest generowane losowo rozwiązanie, które jest zależne od poprzedniego. Jeżeli rozwiązanie kandydata jest lepsze niż poprzednie, to kandydat staje się aktualnie najlepszym rozwiązaniem. Jednak kiedy okaże się ono gorsze liczone jest prawdopodobieństwo Boltzmanna uwzględniające temperaturę. Jeżeli losowa liczba z przedziału [0,1] okaże się mniejsza niż prawdopodobieństwo Boltzmanna, rozwiązanie kandydata zostaje najlepszym rozwiązaniem. Na początku temperatura programu osiąga maksimum i z każdą iteracją spada. Spadek temperatury sprawia że zmniejsza się szansa na zaakceptowanie mniej korzystnego rozwiązania. Po wykonaniu zadanej liczby iteracji algorytm zwraca rozwiązanie, którego dokładność zależy od liczby iteracji.

3. Podejście metodą tabu-search

Tabu-search jest to algorytm metaheurystyczny służący do przeszukiwania rozwiązań problemu w celu znalezienia rozwiązań optymalnych lub niewiele się od nich różniących.

Sposób działania algorytmu przedstawia poniższy graf:



Początek dla rozwiązania tabu-search jest podobny jak w przypadku podejścia wyżarzania. Potrzebne jest nam początkowe, losowe rozwiązanie i w tym wypadku jest ono tak samo generowane jak w pierwszej metodzie. Następnie generowana jest lista sąsiednich(podobnych) rozwiązań. Dla każdego sąsiada sprawdzana jest dokonana w nim zmiana względem oryginału. Jeżeli taka zmiana nie znajduje się w liście tabu to jest tam umieszczana oraz sprawdzana jest funkcja aspiracji - sprawdzane jest czy funkcja celu jest większa niż poprzednia. W przypadku spełnienia tej funkcji dany sąsiad staje się rozwiązaniem, dla którego w następnej iteracji będą szukani sąsiedzi. Po wykonaniu zadanej liczby iteracji algorytm zwraca rozwiązanie, którego dokładność zależy od liczby iteracji.

4. Porównanie i wnioski

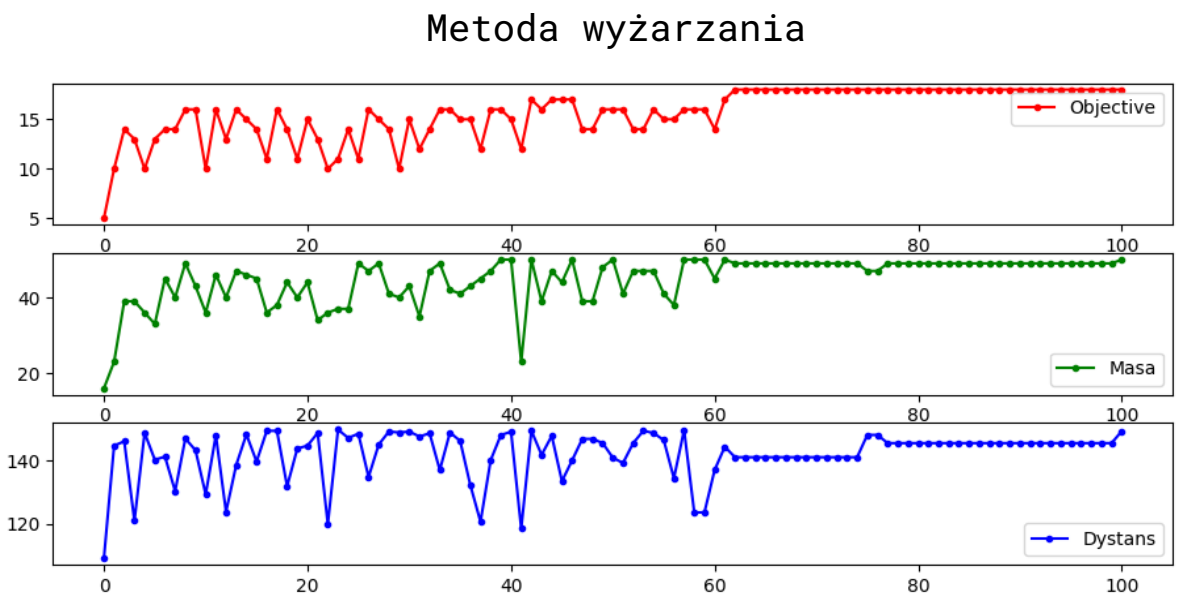
Symulacje zostały przeprowadzone dla takich samych: danych wejściowych, konfiguracji robota oraz ilości iteracji. Losowanie danych odbywało się po określeniu wspólnego ziarna do losowania.

Dane robota:

1. Ładowność - 50
2. Zasięg - 150

Dane mapy i losowanych próbek:

1. Wymiary mapy: 50x50
2. Ilość próbek: 20
3. Wartość próbek w przedziale: (1,10)
4. Masa próbek w przedziale: (1,10)



Best solution:

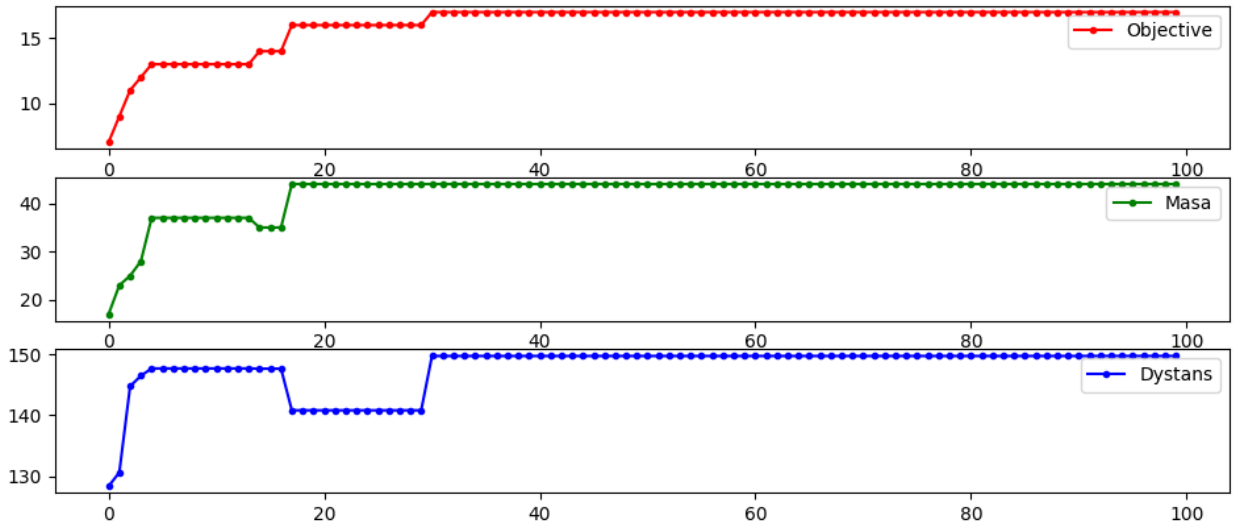
```
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Best decisions:

[0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0]

Best objective: 17

Metoda Tabu Search



Best solution:

```
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Best decisions:

[0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0]

Best objective: 18

Jak wykazały przebiegi obu symulacji, obie metody znajdują rozwiązanie optymalne. Wyżarzanie szybciej znajduje pierwsze efektywne rozwiązanie, a później zaczyna oscylować wokół rozwiązania optymalnego. Tabu search natomiast do pierwszego efektywnego rozwiązania dochodzi wolniej, natomiast później stale poprawia wyniki.

Przy wyżarzaniu więc można łatwiej dostrzec kiedy algorytm przestaje już poprawiać wynik rozwiązania, a przy tabu search potrzeba więcej przejść by móc stwierdzić ten sam fakt - że dalsze przejścia nie byłyby już sensowne ze względu na koszt czasowy.