

# Rozmieszczanie kamer bezpieczeństwa

Wiktor Franus  
Grzegorz Staniszewski

12 stycznia 2018

## Spis treści

<b>1</b>	<b>Treść zadania</b>	<b>2</b>
<b>2</b>	<b>Założenia</b>	<b>2</b>
<b>3</b>	<b>Przestrzeń przeszukiwań</b>	<b>2</b>
<b>4</b>	<b>Funkcja celu</b>	<b>3</b>
<b>5</b>	<b>Przykład</b>	<b>4</b>
<b>6</b>	<b>Metaheurystyka</b>	<b>5</b>
<b>7</b>	<b>Przewidywane wyniki pracy</b>	<b>5</b>
<b>8</b>	<b>Implementacja</b>	<b>6</b>
8.1	Plik konfiguracyjny . . . . .	7
8.2	Sposób uruchomienia . . . . .	8
<b>9</b>	<b>Badania</b>	<b>8</b>
9.1	Badanie 1 . . . . .	8
9.2	Badanie 2 . . . . .	9
9.3	Badanie 3 . . . . .	9

## 1 Treść zadania

Jak optymalnie rozmieścić kamery monitoringu w ustalonym pomieszczeniu (rzut z góry), aby minimalną liczbą kamer móc obserwować dowolne miejsce (z uwzględnieniem maksymalnej dopuszczalnej odległości od kamery). W rozwiązaniu należy uwzględnić możliwość zapewnienia parametryzowanej redundancji - tzn. wymagania, aby każde miejsce było obserwowane przez co najmniej  $n$  kamer.

## 2 Założenia

1. Pomieszczenie jest wielokątem zawierającym tylko kąty o mierze 90 lub 270 stopni. Pomieszczenie reprezentowane jest przez zbiór punktów (z I ćwiartki układu współrzędnych) podanych w formie listy. Połączenie tych punktów linią, zgodnie z ich kolejnością na liście, skutkuje otrzymaniem linii łamanej ograniczającej pomieszczenie. Punkty podawane są w kolejności zgodnej z ruchem wskazówek zegara. Pierwszy i ostatni punkt jest taki sami (należy domknąć pomieszczenie).
2. Kamery mają jednakowy zasięg reprezentowany przez kwadrat o parametryzowanej długości boku. Współrzędne kamery są jednocześnie współrzędnymi środka tego kwadratu. Kamera musi znajdować się wewnątrz pomieszczenia i nie przenika przez ściany.
3. Wnętrze pomieszczenia zdyskretyzowane jest do zbioru punktów o współrzędnych całkowitych poprzez nałożenie siatki o parametryzowanej gęstości.
4. Punkty leżące na krawędziach wielokąta opisującego pomieszczenie nie należą do jego wnętrza.

## 3 Przestrzeń przeszukiwań

- Elementem przestrzeni przeszukiwań jest wektor par liczb całkowitych oznaczających współrzędne kamer:

$$[(x_1, y_1), \dots, (x_i, y_i), \dots, (x_k, y_k)]$$

gdzie:

$x_i$  - współrzędna x i-tej kamery,

$y_i$  - współrzędna y i-tej kamery,

$k$  - liczba kamer.

- Przejście do sąsiedniego elementu możliwe jest poprzez:
  - zmianę położenia jednej z kamer na 2 sposoby (sposób ustalany jest na początku zadania):
    - \* zmiana współrzędnych  $x$  lub  $y$  jednej z kamer o 1 jednostkę,
    - \* przeniesienie jednej z kamer do innego punktu z wnętrza pomieszczenia wylosowanego zgodnie z rozkładem jednostajnym,
  - dodanie nowej kamery w losowym miejscu (rozkład jednostajny),
  - usunięcie jednej kamery.
- Przestrzeń ma strukturę grafową, w której każda krawędź odpowiada jednemu z wymienionych wyżej przejść między elementami przestrzeni.

## 4 Funkcja celu

Informacje znane dla danej instancji problemu:

$X$  - zbiór punktów reprezentujących wnętrze pomieszczenia.

$n_{kmin}$  - minimalna teoretyczna liczba kamer wymagana do pokrycia danego pomieszczenia (obliczana jako stosunek liczności zbioru  $X$  do liczby punktów pokrywanych przez 1 kamerę, zaokrąglany do jedności w górę),

Parametry funkcji celu:

$\alpha$  - zysk z pokrywania powierzchni pomieszczenia,

$\beta$  - koszt użycia nadmiarowej kamery,

$r_{min}$  - minimalna liczba kamer pokrywająca każde miejsce w pomieszczeniu.

Zadanie polega na maksymalizacji funkcji:

$$f(p, k, r) = \alpha * p - \beta * k - \frac{1}{r_{min}} * r$$

gdzie:

$p$  - stosunek liczby punktów pokrytych przez kamery do liczności zbioru  $X$

$k$  - stosunek nadwyżki liczby kamer do  $n_{kmin}$ , obliczany wg. wzoru:

$$k = \frac{\max(0, n_k - n_{kmin})}{n_{kmin}}, \text{ gdzie } n_k - \text{liczba kamer w aktualnym stanie}$$

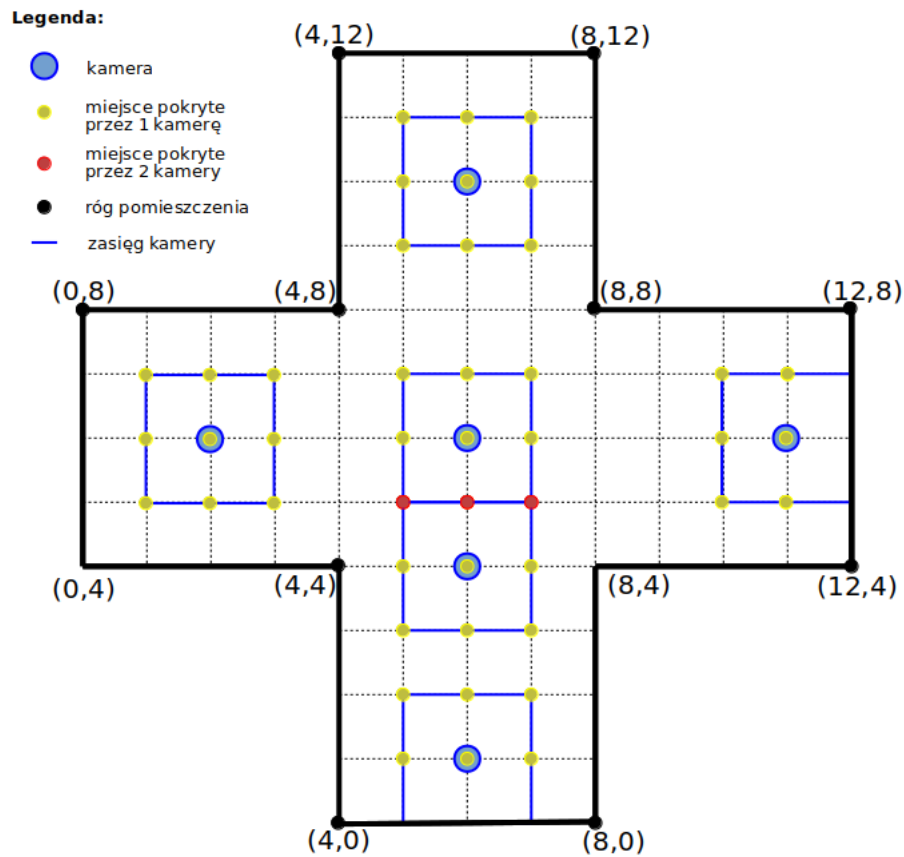
Parametr  $r$  może być obliczany na dwa sposoby (sposób ustalany jest na początku zadania):

- jako średni stopień niespełnienia warunku redundancji dla punktu z wnętrza pomieszczenia, obliczany wg. wzoru:

$$r = \frac{\sum_{x \in X} \max(0, r_{min} - r_x)}{|X|}, \text{ gdzie } r_x - \text{liczba kamer pokrywających punkt } x$$

- jako maksymalne niespełnienie warunku redundancji spośród wszystkich punktów z wnętrza pomieszczenia, wg. wzoru:  
 $r = \max(0, r_{\min} - r_x)$ , gdzie  $r_x$  - liczba kamer pokrywających punkt  $x$ , który jest najsłabiej pokrytym punktem.

## 5 Przykład



- Wartości parametrów:  
 $\alpha = 1$   
 $\beta = 1$   
 $r_{\min} = 1$
- Informacje obliczone dla powyższego pomieszczenia:  
Rozmiar boku kwadratu reprezentującego zasięg kamery: 2  
Liczba punktów pokrywanych przez jedną kamerę: 9

$$|X| = 57$$

$$n_{kmin} = \lceil \frac{57}{9} \rceil = 7$$

- Obliczenie wartości funkcji celu dla stanu z rysunku:

Liczba kamer użytych: 6

Liczba punktów pokrytych przez kamery: 45

$$p = \frac{45}{57} = 0.7895$$

$$k = \frac{\max(0, 6-7)}{7} = \frac{0}{20} = 0$$

Parametr  $r$  obliczany pierwszym sposobem:

$$r = \frac{45*0+12*1}{57} = \frac{12}{57} = 0.2105$$

$$f(p, k, r) = 1 * 0.7895 - 1 * 0 - 1 * 0.2105 = 0.5790$$

Parametr  $r$  obliczany drugim sposobem:

$$r = \max(0, 1 - 0) = 1$$

$$f(p, k, r) = 1 * 0.7895 - 1 * 0 - 1 * 1 = -0.2105$$

## 6 Metaheurystyka

Element początkowy przestrzeni przeszukiwań jest zbiorem zawierającym  $n_{kmin}$  kamer rozmieszczonych losowo wewnątrz pomieszczenia.

Do rozwiązania problemu użyjemy algorytmu symulowanego wyżarzania. Przy odpowiednio dobranych parametrach metoda ta, w porównaniu do algorytmów wspinaczkowych, daje większą szansę na znalezienie optymalnego rozwiązania, ponieważ zmniejsza ryzyko zatrzymania się w ekstremach lokalnych. W początkowej fazie przeszukiwania przestrzeni dopuszczalne jest przechodzenie do stanów gorszych (o mniejszej wartości funkcji celu). Wraz z rosnącą liczbą iteracji obszar poszukiwań jest ograniczany, a algorytm bardziej skupia się na poprawie bieżącego rozwiązania.

## 7 Przewidywane wyniki pracy

Przeprowadzona zostanie seria eksperymentów z różnymi wartościami parametrów  $\alpha$ ,  $\beta$ ,  $r_{min}$  na kilku instancjach problemu (różne pomieszczenia). Dla ustalonych parametrów funkcji celu, sterować będziemy parametrami metaheurystyki, tj. funkcją wygaszania temperatury i jej wartością początkową. Ponadto sprawdzimy dwa podejścia do zmiany położenia kamery oraz dwa sposoby obliczania parametru  $r$  funkcji celu. Dla wybranej instancji zadania sprawdzimy też wpływ gęstości siatki punktów z wnętrza pomieszczenia

na zachowanie metaheurystyki. Sporządzone zostaną wykresy przedstawiające wartość funkcji celu oraz liczbę użytych kamer w zależności od liczby wykonanych iteracji.

## 8 Implementacja

Do realizacji zadania wykorzystaliśmy gotową implementację metaheurystyki zawartą w pakiecie `simanneal` w wersji 0.4.1, której dokumentacja jest dostępna pod adresem <https://github.com/perrygeo/simanneal>. Biblioteka implementuje algorytm symulowanego wyżarzania z wykładniczą funkcją wygaszania temperatury minimalizujący zadaną funkcję celu. Z racji, że naszym zadaniem miało być maksymalizowanie opisanej w specyfikacji funkcji celu, musieliśmy zmodyfikować tę funkcję, zmieniając jej znak na przeciwny. Poprawiony wzór na funkcję celu ma postać:

$$f(p, k, r) = -(\alpha * p - \beta * k - \frac{1}{r_{min}} * r)$$

Biblioteka `simanneal` udostępnia użytkownikowi 3 parametry, którymi można sterować zachowaniem metaheurystyki:

- *steps* - liczba iteracji algorytmu, domyślnie 50000,
- *Tmax* - początkowa wartość temperatury, domyślnie 25000,
- *Tmin* - końcowa wartość temperatury, domyślnie 2.5.

Początkowo sprawdziliśmy zachowanie metaheurystyki dla domyślnych wartości udostępnionych parametrów, jednak okazały się one nietrafione. Zmniejszyliśmy zatem temperaturę początkową 100-krotnie, czyli do wartości 250. W rezultacie na wykresie wartości funkcji celu względem numeru iteracji zaczęły pojawiać się charakterystyczne dla symulowanego wyżarzania „skoki”, których nasilenie zmniejszało się z kolejnymi iteracjami. Zaobserwowaliśmy także, że zmniejszenie liczby iteracji 2-krotnie, czyli do wartości 25000, nie wpłynęło na uzyskiwane rezultaty, ale za to zgodnie z intuicją zmniejszył się czas obliczeń. Badania postanowiliśmy zatem przeprowadzać na 25000 iteracjach.

### 8.1 Plik konfiguracyjny

Przykładowy plik konfiguracyjny wraz z komentarzami znajduje się poniżej.

```

{
  "room": [
    {
      "x": 0.0,
      "y": 0.0
    },
    {
      "x": 0.0,
      "y": 4.0
    },
    {
      "x": 4.0,
      "y": 4.0
    },
    {
      "x": 4.0,
      "y": 0.0
    },
    {
      "x": 0.0,
      "y": 0.0
    }
  ],
  "t_max": 2500.0,
  "t_min": 2.5,
  "alpha": 10,
  "beta": 1,
  "r_min": 1,
  "num_iterations": 10,
  "num_updates" : 1,
  "camera_move_method": "local",
  "camera_side" : 4,
  "r_count_method": "average",
  "density" : 1
}

// Pomieszczenie opisane jako lista wierzchołków,
// zgodnie z założeniami:
// * punkty podawane są w kolejności
//   zgodnej z ruchem wskazówek zegara,
// * Pomieszczenie jest wielokątem zawierającym
//   tylko kąty o mierze 90 lub 270 stopni,

// * ostatni punkt musi domykać pomieszczenie,
//   musi pokrywać się z pierwszym na liście.

// Temperatura maksymalna
// Temperatura minimalna
// Zysk z pokrywania powierzchni pomieszczenia
// Koszt użycia nadmiarowej kamery
// Minimalna liczba kamer
//   pokrywająca każde miejsce w pomieszczeniu
// Liczba iteracji
// Liczba wyników pośrednich (nie ma wpływu na wyniki)
// Sposób przesuwania kamery:
// * local - o 1 jednostkę w dowolnym kierunku
// * random - w dowolne miejsce w pomieszczeniu
// Długość boku kwadratu reprezentującego zasięg kamery
// Sposób wyznaczania niespełnienia redundancji:
// * average - średni stopień
// * max - maksymalne
// Odległość pomiędzy punktami (gęstość siatki)

```

## 8.2 Sposób uruchomienia

Program wymaga do działania Pythona w wersji 3.x. Przed uruchomieniem programu należy zainstalować wymagane pakiety języka Python znajdujące się w pliku `pip.requirements`. Można to zrobić poleceniem:

```
pip3 install -r pip.requirements
```

Opis parametrów uruchomienia dostępny jest po wpisaniu polecenia:

```
python3 main.py --help
```

Usage: main.py [options]

Options:

```
-h, --help            show this help message and exit
-f CONFIGFILE, --config=CONFIGFILE
                        Json file with configuration
-e EXPERIMENTS, --experiments=EXPERIMENTS
                        Number of experiments
```

Domyślnie program wczytuje konfigurację z pliku o nazwie `config.json`, a liczba badań wynosi 1. Wyniki badań zapisywane są w katalogu `./out/<data>`. W wyjściowym katalogu tworzone są podkatalogi dla każdej iteracji programu (eksperymentu). Każdy podkatalog zawiera wykres przedstawiający wartości funkcji celu względem liczby iteracji. Wykres zapisany jest w pliku o nazwie `costs.png`. W tym samym podkatalogu znajdują się także pliki graficzne przedstawiające stan pomieszczenia w wybranych, równomiernie rozłożonych punktach czasowych. Liczbę tych punktów można określić zmieniając wartość parametru `num_updates` w pliku konfiguracyjnym. Dodatkowo w pliku `best_state.png` zapisany jest także wygląd stanu, w którym wartość funkcji celu była minimalna. W pliku `./out/<data>/average_costs.png` znajduje się wykres przedstawiający średnią wartość funkcji celu dla wszystkich iteracji (eksperymentów). Ponadto do katalogu wyjściowego kopiowany jest plik konfiguracyjny użyty w danym uruchomieniu programu.

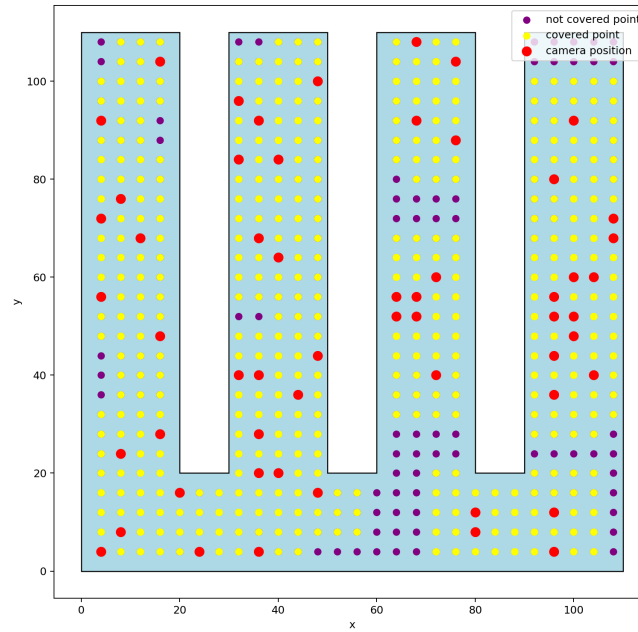
## 9 Badania

### 9.1 Badanie 1

```
"t_max": 250.0,
"t_min": 2.5,
"alpha": 10,
"beta": 1,
"r_min": 1,
"num_iterations": 10000,
"num_updates": 100,
"camera_move_method": "local",
"camera_side": 20,
```



```
"r_count_method": "average",
"density" : 4
```

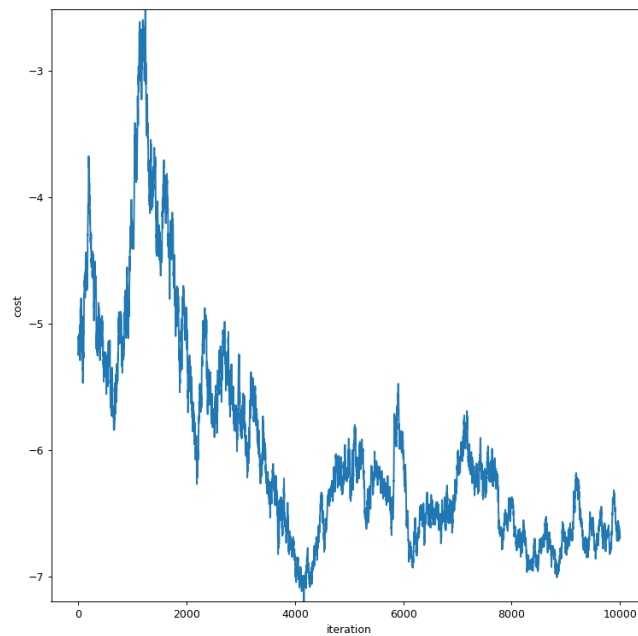


Rysunek 1: Obliczony układ kamer w pomieszczeniu przez algorytm.

zależy nam na jak największym pokryciu, nie boli nas używanie kamer, dlatego jest ich duże zagęszczenie.

## 9.2 Badanie 2

```
"t_max": 50.0,
"t_min": 2.5,
"alpha": 10,
"beta": 1,
"r_min": 1,
"num_iterations": 10000,
"num_updates" : 100,
"camera_move_method": "local",
"camera_side" : 20,
"r_count_method": "average",
"density" : 4
```

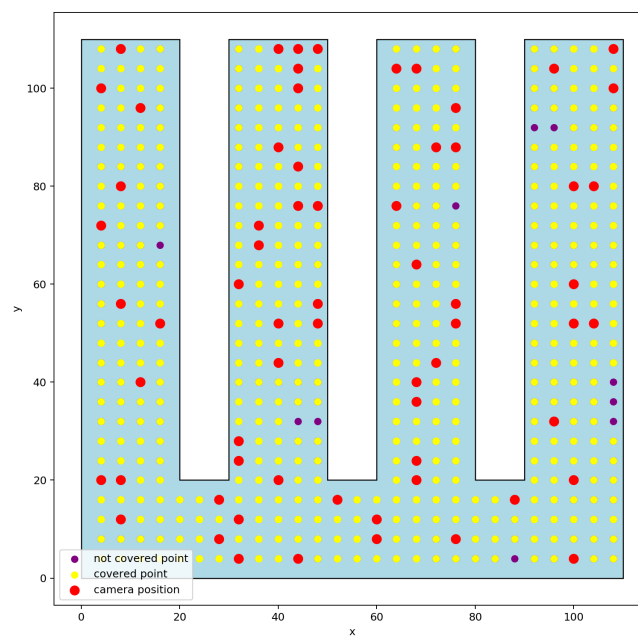


Rysunek 2: .

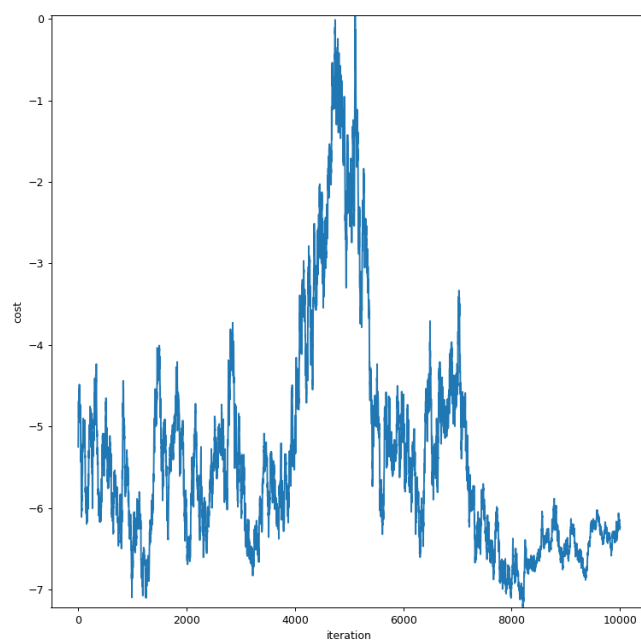
Zostało

### 9.3 Badanie 3

```
"t_max": 50.0,
"t_min": 2.5,
"alpha": 10,
"beta": 1,
"r_min": 1,
"num_iterations": 10000,
"num_updates" : 100,
"camera_move_method": "random",
"camera_side" : 20,
"r_count_method": "average",
"density" : 4
```



Rysunek 3: Obliczony układ kamer w pomieszczeniu przez algorytm.



Rysunek 4: .