Rozmieszczanie kamer bezpieczeństwa

Wiktor Franus
Grzegorz Staniszewski

12 stycznia 2018

Spis treści

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:

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

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:
n_{kmin} - minimalna teoretyczna liczba kamer wymagana do pokrycia danego pomieszczenia (obliczana jako stosunek pola powierzchni pomieszczenia do pola powierzchni zasięgu jednej kamery, zaokrąglany do jedności w górę),
X - zbiór punktów reprezentujących wnętrze pomieszczenia.

Parametry funkcji celu:
α - zysk z pokrywania powierzchni pomieszczenia,
β - koszt użycia nadmiarowej kamery,
r_{min} - minimalna liczba kamer pokrywająca każde miejsce w pomieszczeniu.

Zadanie polega na maksymalizacji funkcji:

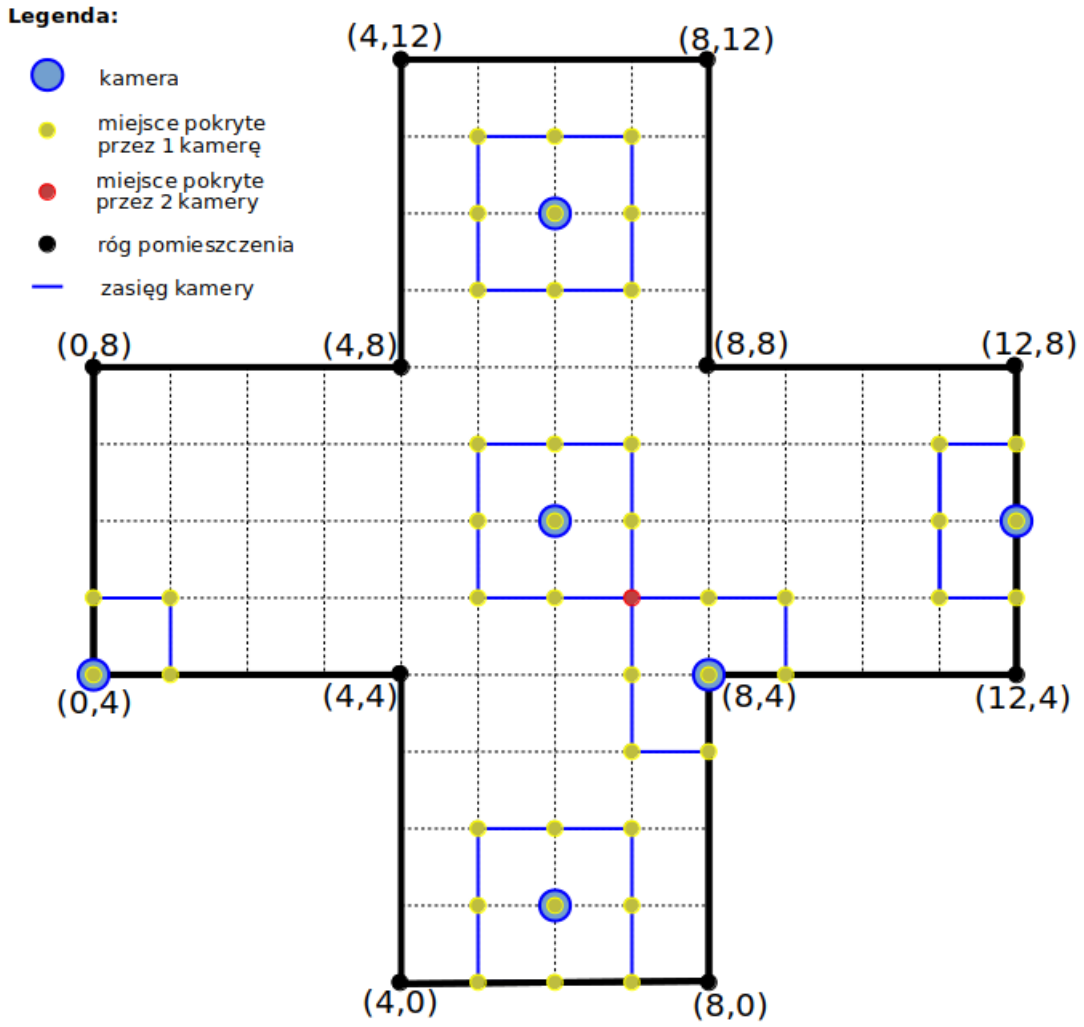
$$f(p,k,r) = \alpha * p - \beta * k - \frac{1}{r_{min}} * r$$

gdzie:
p - stosunek powierzchni pokrytej przez kamery do powierzchni pomieszczenia
k - stosunek nadwyżki liczby kamer do n_{kmin}, obliczany wg. wzoru:
$k = \frac{max(0, n_k - n_{kmin})}{n_{kmin}}$, gdzie n_k - 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:
  $r = \frac{\sum_{x \in X} max(0, r_{min} - r_x)}{|X|}$, gdzie r_x - 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 będący najsłabiej pokrytym punktem.

5 Przykład



- Wartości parametrów:
  $\alpha = 1$
  $\beta = 1$
  $r_{min} = 1$

- Informacje obliczone dla powyższego pomieszczenia:
  pole powierzchni pomieszczenia: 80
  $n_{kmin} = \frac{80}{2*2} = 20$
  $|X| = 105$

- Obliczenie wartości funkcji celu dla stanu z rysunku:
  Liczba kamer użytych: 6
  Pole powierzchni pokryte przez kamery: 18
  $p = \frac{18}{80} = 0.225$
  $k = \frac{max(0,6-20)}{20} = \frac{0}{20} = 0$
  Parametr r obliczany pierwszym sposobem: $r = \frac{44*0+61*1}{105} = \frac{61}{105} = 0.58$
  $f(p,k,r) = 1 * 0.225 - 1 * 0 - 1 * 0.58 = -0.355$

  Parametr r obliczany drugim sposobem: $r = 1 - 0 = 1$
  $f(p,k,r) = 1 * 0.225 - 1 * 0 - 1 * 1 = -0.775$

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 α, β, 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 udostepnionych 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.

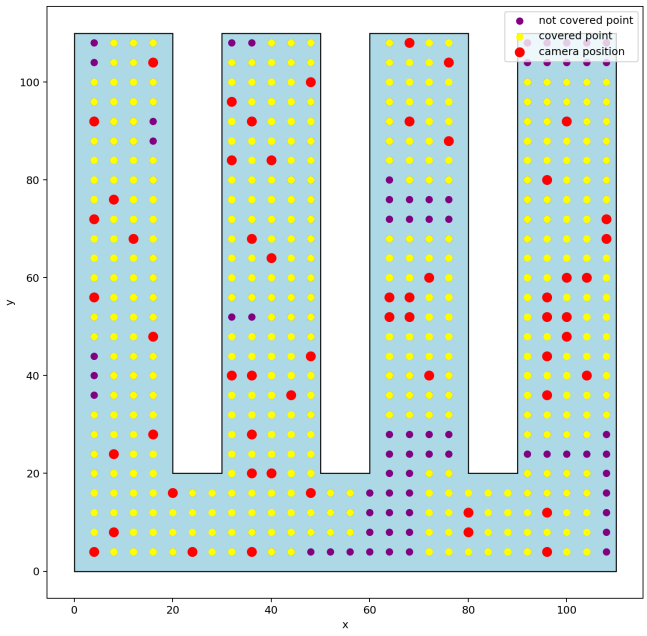
```
{
  "room": [                       // Pomieszczenie opisane jako lista wierzchołków,
    {                             // zgodnie z założeniami:
      "x": 0.0,                   // * punkty podawane są w kolejności
      "y": 0.0                    //   zgodnej z ruchem wskazówek zegara,
    },                            // * Pomieszczenie jest wielokątem zawierającym
    {                             //   tylko kąty o mierze 90 lub 270 stopni,
      "x": 0.0,
      "y": 4.0
    },
    {
      "x": 4.0,
      "y": 4.0
    },
    {
      "x": 4.0,
      "y": 0.0
    },
    {
      "x": 0.0,                   // * ostatni punkt musi domykać pomieszczenie,
      "y": 0.0                    //   musi pokrywać się z pierwszym na liście.
    }
  ],
  "t_max": 2500.0,                // Temperatura maksymalna
  "t_min": 2.5,                   // Temperatura minimalna
  "alpha": 10,                    // Zysk z pokrywania powierzchni pomieszczenia
  "beta": 1,                      // Koszt użycia nadmiarowej kamery
  "r_min": 1,                     // Minimalna liczba kamer
                                  // pokrywająca każde miejsce w pomieszczeniu
  "num_iterations": 10,           // Liczba iteracji
  "num_updates" : 1,              // Liczba wyników pośrednich (nie ma wpływu na wyniki)
  "camera_move_method": "local",  // Sposób przesuwania kamery:
                                  // * local - o 1 jednostkę w dowolnym kierunku
                                  // * random - w dowolne miejsce w pomieszczeniu
  "camera_side" : 4,              // Długość boku kwadratu reprezentującego zasięg kamery
  "r_count_method": "average",    // Sposób wyznaczania niespełnienia redundancji:
                                  // * average - średni stopień
                                  // * max - maksymalne
  "density" : 1                   // Odległość pomiędzy punktami (gęstość siatki)
}
```

8.2 Sposób uruchomienia

9 Badania

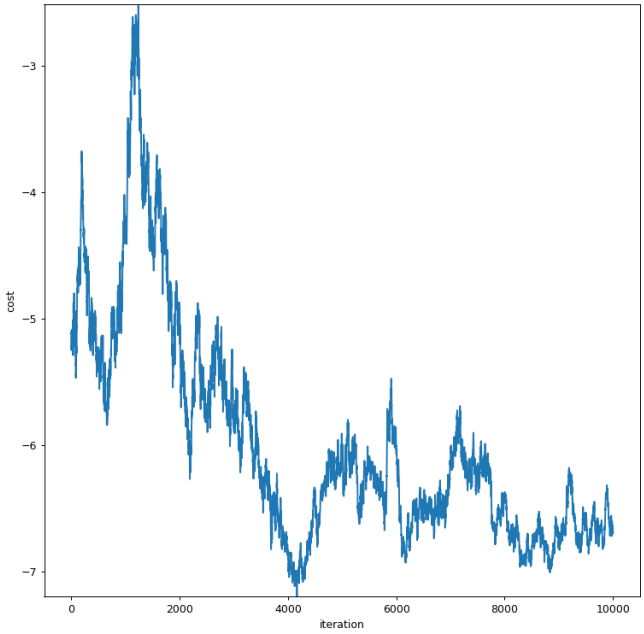
9.1 Badanie 1

```
"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.

zalezy nam na jak najwiekszym pokryciu, nie boli nas uzywanie kamer, dlatego jest ich duze zageszczenie.



Rysunek 2: .

9.2 Badanie 2

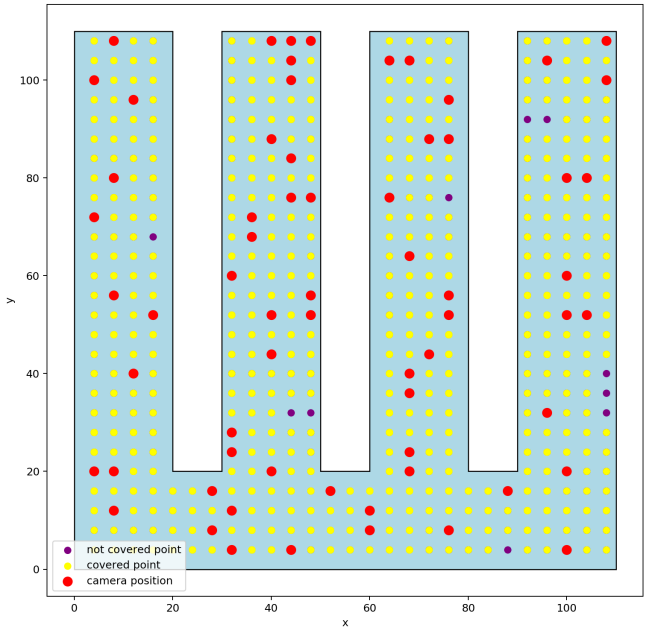
```
"t_max": 50.0,
"t_min": 2.5,
"alpha": 10,
"beta": 1,
"r_min": 1,
"num_iterations": 10000,
"num_updates" : 100,
"camera_move_method": "local",
"camera_side" : 20,
"r_count_method": "average",
"density" : 4
```

Zostało

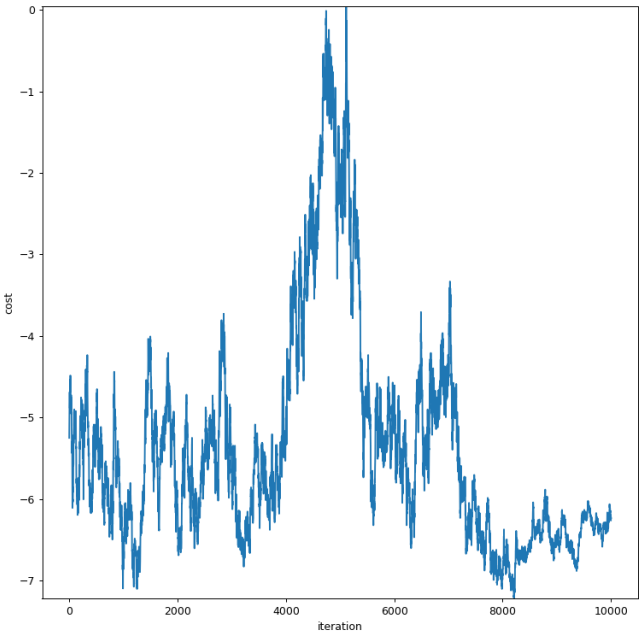
9.3 Badanie 3

```
"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: .