Rozmieszczanie kamer bezpieczeństwa

Przemysław Kopański
Mateusz Forc

9 czerwca 2017

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).

2 Przyjęte założenia

- wszystkie kamery są takie same (mają taki sam zasięg)
  - zasięg kamery jest kołem o stałym promieniu
  - promień zasięgu kameru wynosi 2 (możliwa interpretacja - średnica kamery wynosi 4 metry)
- rzut pomieszczenia reprezentowany jest przez zbiór punktów
  - punkty mają współrzędne odpowiadające I ćwiartce wykresu

$$x \geqslant 0, y \geqslant 0$$

  - punkty podawane są jako lista, która reprezentuje zamknięty wielokąt - muszą one zostać podane we właściwej kolejności, tak aby można je było jednoznacznie połączyć (każde dwa kolejne punkty łączone są w odcinek)

3 Przestrzeń przeszukiwań

Pojedynczym elementem przestrzeni przeszukiwań jest zbiór kamer wraz z ich pozycjami.
Rozwiązanie początkowe zawiera zbiór składający się z ⌈obszar rzutu/obszar jednej kamery⌉ kamer rozmieszczonych losowo wewnątrz wielokątu. Do kolejnego stanu możemy przejść poprzez dodanie/usunięcie kamery lub przemieszczenie jednej z aktualnie umieszczonych kamer. Do zbioru kamer nie można wstawić kamery, która jest na zewnątrz obserwowanego pomieszczenia.

4 Funkcja celu

Dostępna informacja:
k_{min} - minimalna teoretyczna liczba kamer dla aktualnie rozpatrywanego rzutu ($\lceil$obszar rzutu/obszar jednej kamery$\rceil$)

Parametry zadania:
d_k - koszt użycia kamery
d_p - wartość pokrycia 1% powierzchni

Funkcja: $f(p,k) = d_p * p - \frac{100 d_k}{k_{min}} * max(0, k - k_{min})$

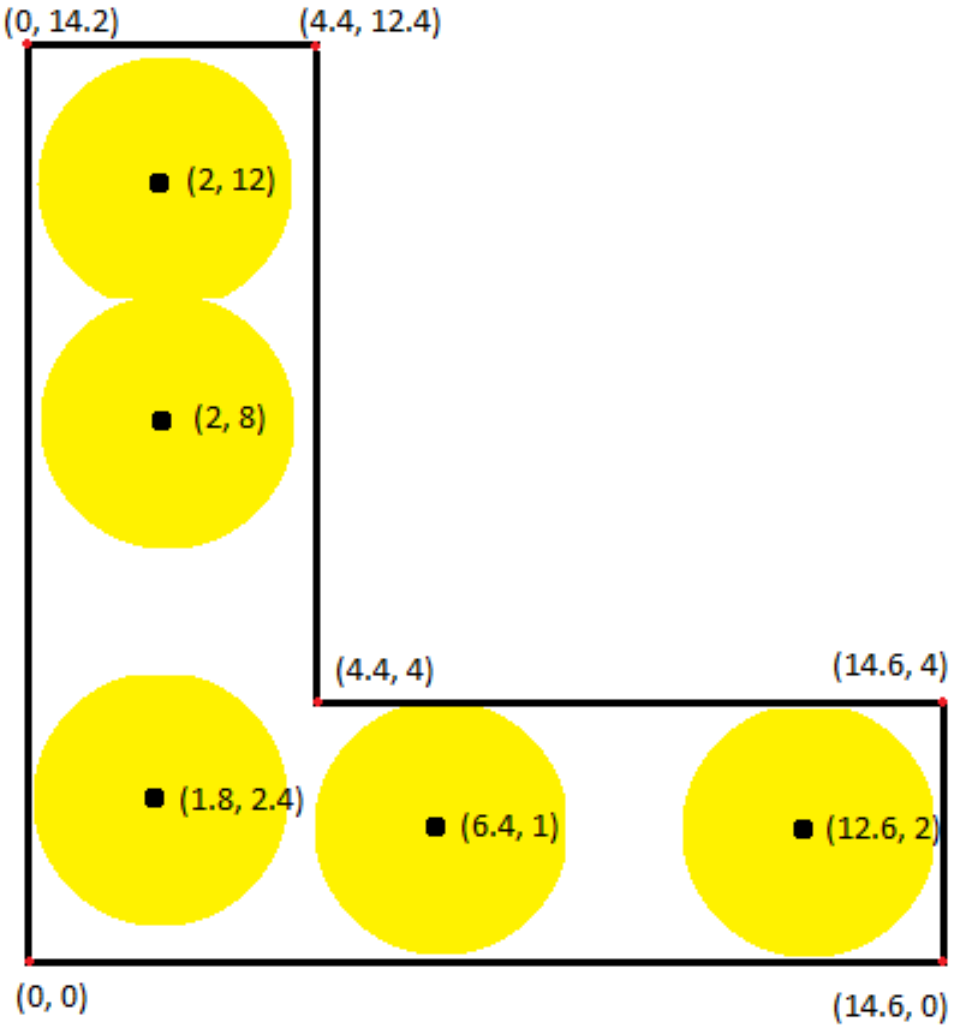
gdzie:
p - % pokrycia dla danego stanu
k - ilość użytych kamer w danym stanie

Zadanie polega na maksymalizacji funkcji f.

Przykładowo
$d_p = 1$
$d_k = 1$

Dla podanych parametrów algorytm będzie znajdował „złoty środek pomiędzy" procentem pokrycia a ilością użytych kamer. Odpowiednio przeskalowując podane parametry i trzymajac odpowiedni stosunek pomiędzy tymi wartościami, mozemy sterowac na czym bardziej nam zależy, jeżeli np $d_p = 2d_k = 1$ to będziemy w stanie zaakceptować dwukrotna ilość kamer w zamian za dwukrotnie większe pokrycie.

4.1 Przykład



Podane pomieszczenie ma pole powierzchni równe 103.38.
Zasięgi kamer są między sobą rozłączne, a ich pole wynosi 62.83.
Minimalna teoretyczna liczba kamery wynosi 9.
Pokrycie dla danego stanu wynosi 60.8%.

Dla parametrów: $d_p = 1 d_k = 1$
wartość funkcji wynosi 60.8-0 = 60.8

Podane rozwiązanie posiada zbyt małą liczbę kamer, by móc w stanie pokryć całą powierzchnię.

5 Metaheurystyka

Zastosowany zostanie stabuizowany algorytm symulowanego wyżarzania. Ze względu na to, że stworzenie funkcji heurystycznej do badanej przestrzeni jest obliczalnie trudne, użycie metody A* jest niewskazane. Algorytmy wspinaczkowe nie sprawdzą się w opisywanej przestrzeni ze względu na dużą liczbę ekstremów lokalnych. Stosując algorytm symulowanego wyżarzania zapewnione jest, że algorytm nie 'utknie' w ekstremum. Wraz z rosnącą liczbą iteracji można zmniejszać temperaturę, w celu znalezienia coraz lepszego rozwiązania. Dodatkowym mechanizmem pozwalającym uniknąć zakotwiczenia w ekstremum lokalnym jest tabuizacja.

Podana metoda wymaga strojenia ze względu na 2 parametry:

1. wielkość kolejki tabu - określa ile maksymalnie jednocześnie punktów przestrzeni może ulec tabuizacji
2. parametr funkcji temperatury - do doboru temperatury zostanie wykorzystana funkcja zależna od numeru iteracji, która udostępni parametr do strojenia

6 Przewidywane wyniki pracy

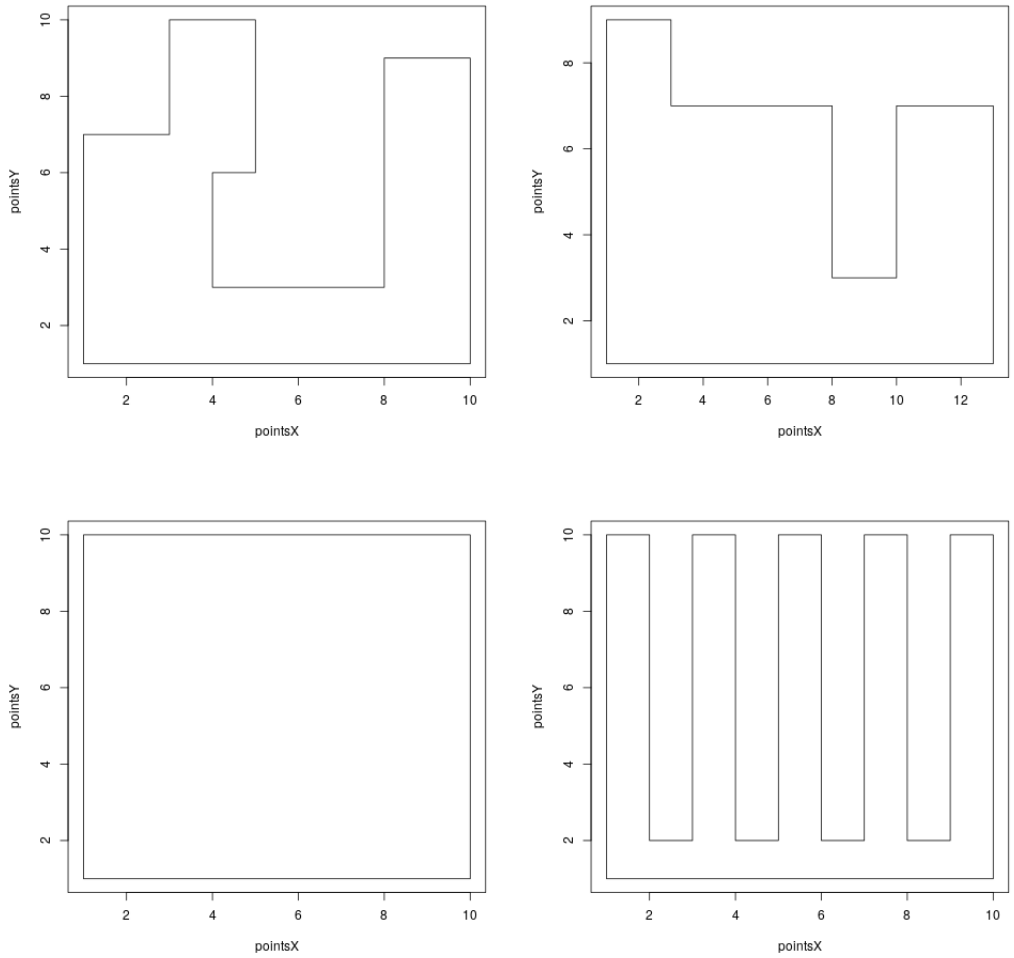
Dla kilkunastu zadanych rzutów ilustrujących różne warianty pomieszczeń np. długie, wąskie i kręte, duże otwarte zostaną przeprowadzone symulacje w celu odnalezienia i zaprezentowania parametrów funkcji celu, które pozwalają implementacji na znalezienie możliwie najlepszego rozwiązania dla danego rodzaju przypadku.

Wyniki zostaną zaprezentowane jako zestawy rzutów oraz wykresów ilustrujących pokrycie w zależności od parametrów: d_p i d_k wraz z wyróżnionym zestawem parametrów dla każdego zestawu, który ilustruje najlepsze rezultaty.

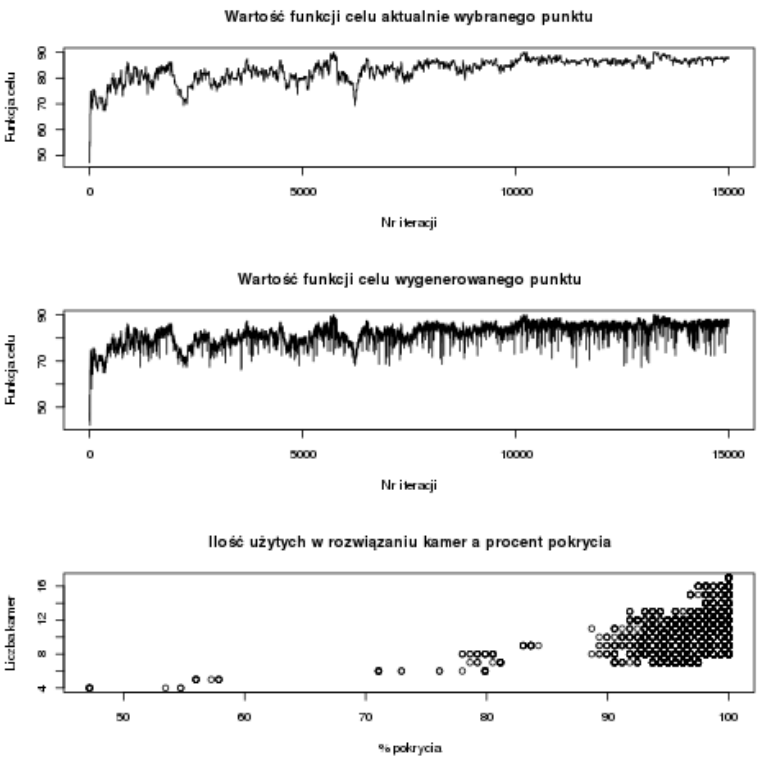
7 Podsumowanie

Dla różnych rodzajów rzutów, uruchomiony został zaimplementowany algorytm, wraz z różnymi parametrami d_p, d_k oraz wielkością kolejki tabu.

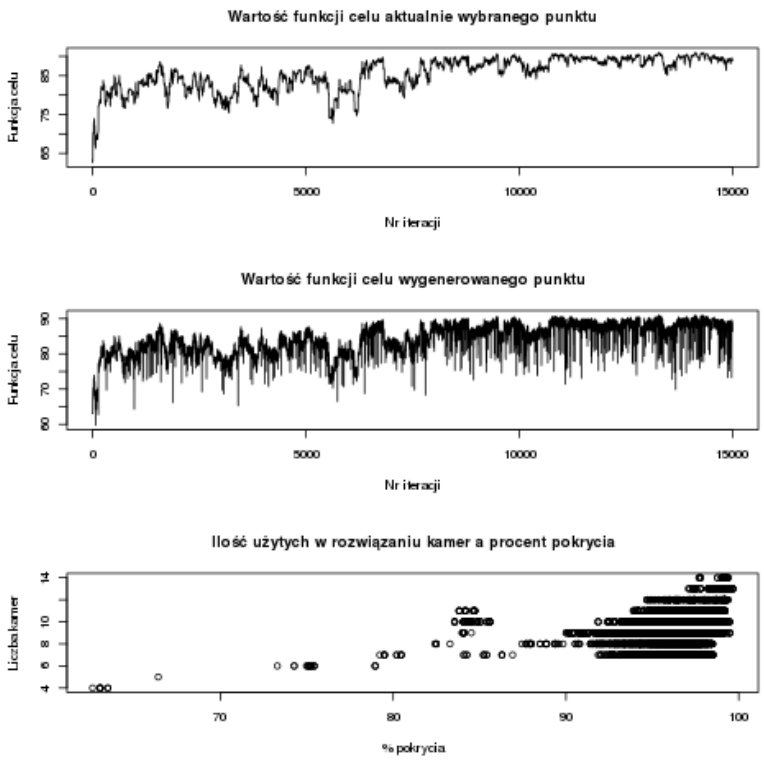
7.1 Przykłady testowe



Przetestowane zostały powyższe mapy w różnych skalach. Znajdują się tutaj dwa przypadki skrajne jeżeli chodzi o szerokość pojedyńczego korytarza na rzucie jak i o ilość korytarzy oraz dwa przypadki dosyć wyważone.
W każdym przypadku promień kamery wynosił 2 i w przypadku skalowania był odpowiednio powiększany.



Rysunek 1: MediumMap, $d_p = 1$, $d_k = 0.1$, $skala = 2$

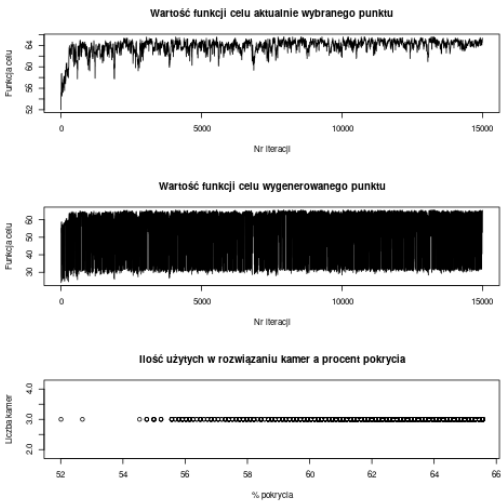


Rysunek 2: MediumMap, $d_p = 1$, $d_k = 0.1$, $skala = 5$

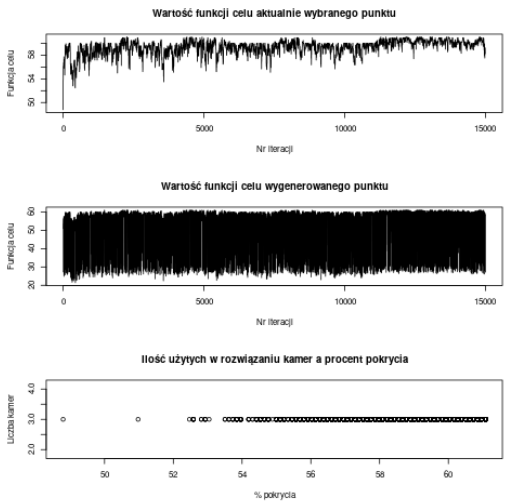
Podane wykresy prezentuja przykładowe działanie algorytmu dla podanych parametrów. Pierwszy wykres pokazuje aktualnie wybrany punkt w jednej iteracji projektu. Drugi wykres prezentuje punkt sąsiedni wygenerowany w danej iteracji algorytmu. Może on zostać wybrany przez algorytm jako następny punkt. Trzeci wykres prezentuje wszystkie wygenerowane rozwiązania przez algorytm, ze względu na ilość użytych kamer oraz procentu pokrycia powierzchni przez kamery.

7.2 Wnioski

Na podstawie przeprowadzonych symulacji nie zauważyliśmy znaczącej różnicy w pracy dla różnej wartości wielkości kolejki.

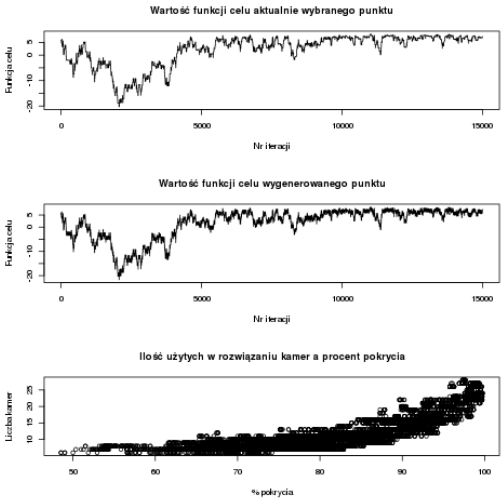


Rysunek 3: MediumMap, $d_p = 1$, $d_k = 1$, $skala = 5$, $queueSize = 10$

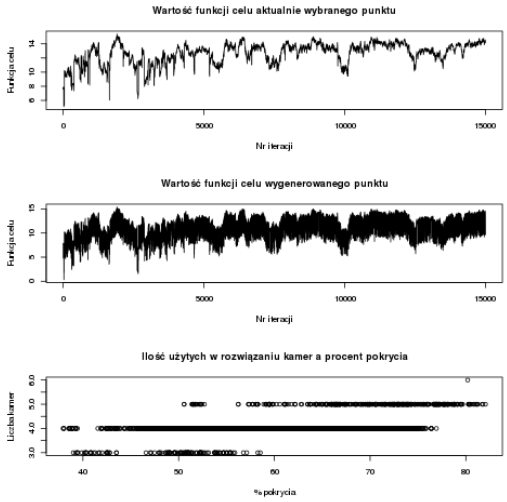


Rysunek 4: MediumMap, $d_p = 1$, $d_k = 1$, $skala = 5$, $queueSize = 100$

7.2.1 Wpływ parametrów d_p i d_k na zachowanie algorytmu



Rysunek 5: OpenSpace,
$d_p = 0.1$, $d_k = 0.1$, $skala = 5$

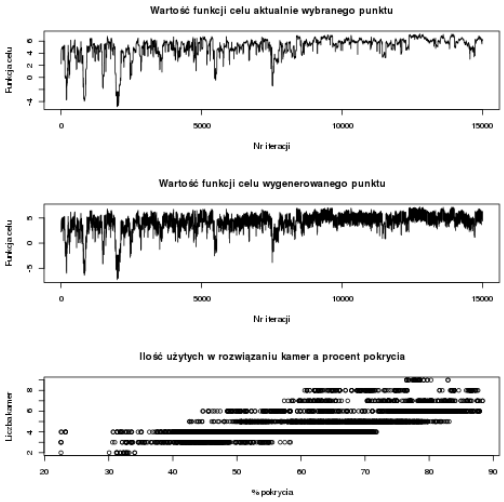


Rysunek 6: MediumMap,
$d_p = 0.2$, $d_k = 0.2$, $skala = 5$

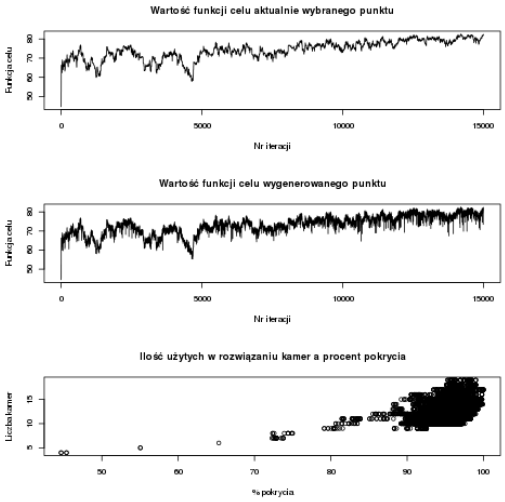
Dla bardzo małych wartości d_p i d_k odnotowaliśmy zachowanie bardzo zbliżone do błądzenia przypadkowaego. Wynika to z faktu, iż różnica pomiędzy kolejnymi rozwiązaniami jest niewielka, przez co istnieje duże prawdopodobieństwo wybrania punktu gorszego.

Prawdopodobieństwo wybrania gorszego punktu wynosi: $exp(\frac{-|q(x)-q(y)|}{t})$, gdzie $q(x), q(y)$ to wartości funkcji celu dla punktu bazowego i jego wygenerowanego sąsiada, a t - aktualna temperatura.

W przypadku niższych wartości d_p i d_k, podana funkcja dla danego t osiąga wartości bliskie jedności.

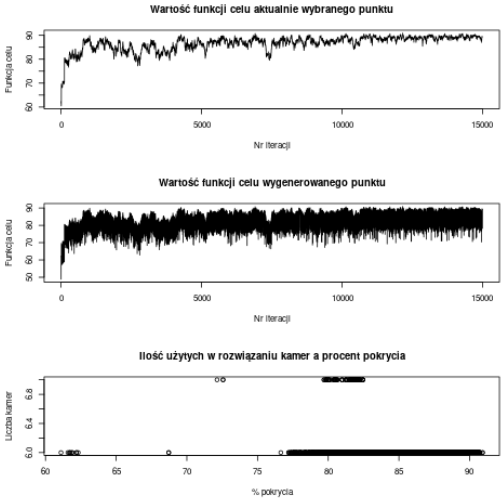


Rysunek 7: MediumMap,
$d_p = 0.9$, $d_k = 0.1$, $skala = 5$

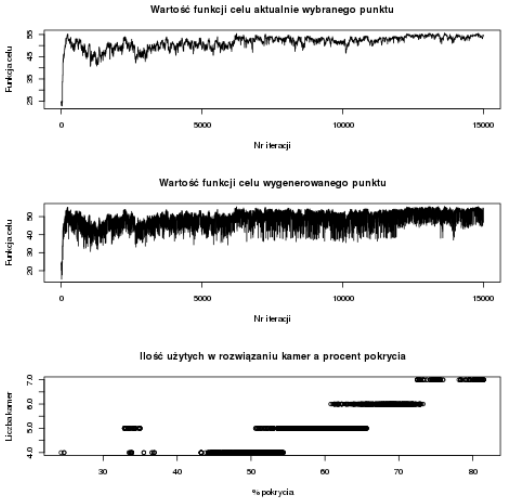


Rysunek 8: SlimMap,
$d_p = 1$, $d_k = 0.1$, $skala = 5$

Dla $d_p >> d_k$ dało sie zaobserwować częste dokładanie kamer. Wynika to z faktu, że koszt dodania kamery jest bardzo mały (róznica wartości funkcji celu) a prawdopodobieństwo dodania nowej kamery jest dosyć duże.



Rysunek 9: MediumMap2,
$d_p = 1.0$, $d_k = 0.7$, $skala = 5$

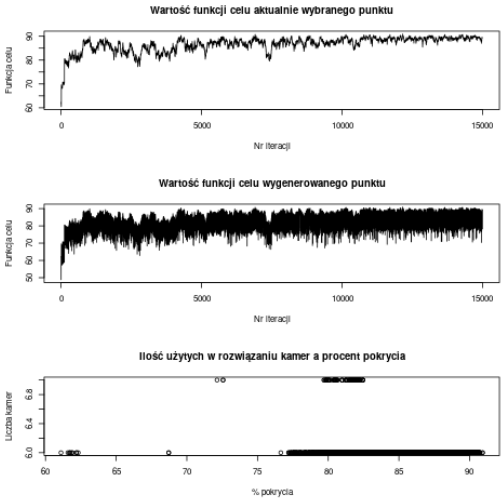


Rysunek 10: SlimMap,
$d_p = 1$, $d_k = 0.4$, $skala = 5$

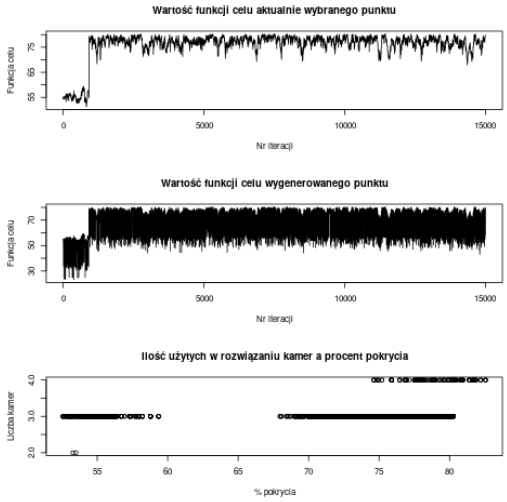
Dla nierównomiernych powierzchni idealne wartości parametrów oscylują pomiędzy $dp = 1$ $d_k = 0.7$. Dzięki temu, gdy kolejne wygenerowane rozwiązanie będzie zawierało dodatkową kamerę, istnieje większa szansa, że zostanie wybrane pomimo, że ta dodatkowa kamera mogłaby sie znajdować w nieoptymalnym dla niej miejscu w danej chwili. W przypadku mapy zawierającej większą ilość wąskich korytarzy preferewana jest niższa wartość parametru d_k.

7.2.2 Wpływ funkcji temperatury

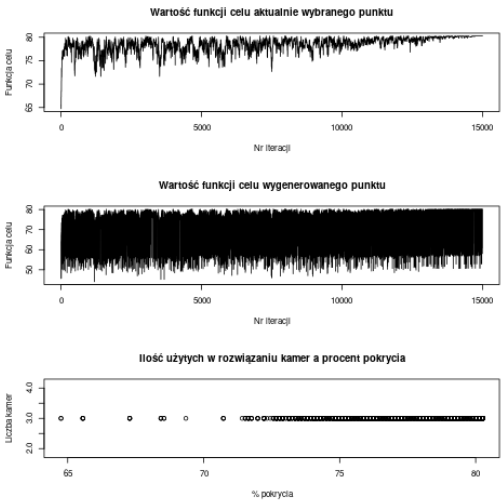
Funkcja temperatury w podanym algorytmie reguluje, jak w danym kroku algorytm ma zachowywać się - z jakim prawdopodobieństwem wybrać nowo wygenerowany punkt, który jest gorszy od obecnie rozpartywanego. Dla przypadku MediumMap2 z parametrami $d_p = 1, d_k = 0.7, skala = 5$ przeprowadziliśmy eksperymenty z innymi funkcjami. Parametr x określa numer aktualnej iteracji i jest znormalizowany (podzielony przez limit liczby iteracji).



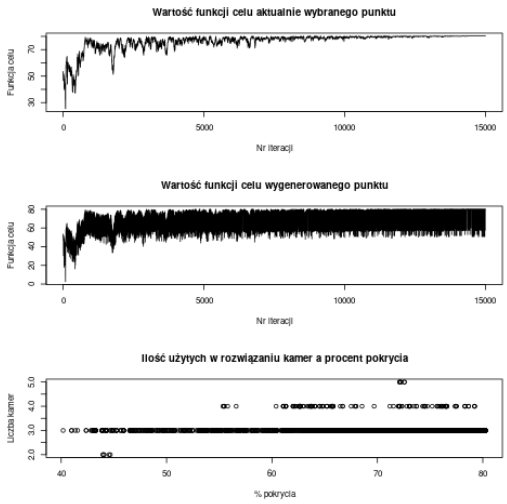
Rysunek 11: MediumMap2, $f(x) = exp(-x)$



Rysunek 12: MediumMap2, $f(x) = exp(-x/100)$



Rysunek 13: MediumMap2, $f(x) = 1 - x^2$



Rysunek 14: MediumMap2, $f(x) = -log(x)$

7.2.3 Zmiana interfejsu użytkownia

Po przebadaniu zachowania algorytmu można zauważyć, że lepszym sposobem do strojenia algorytmu jest podanie współczynnika $\frac{d_p}{d_k}$ zamiast poszczególnych wartości.
Zapewniamy przez to uniknięcie nieintuicyjnej różnicy zachowań algorytmu jak w przypadku uruchomieniu z argumentami:

$$d_p = 1, d_k = 1$$

a na przykład:

$$d_p = 0.5, d_k = 0.5$$

.

7.2.4 Zmiana reprezentacji problemu

W dalekiej fazie prac nad projektem, doszliśmy do wniosku, że jednym z czynników, które spowalniają przebieg symulacji jest nieoptymalna reprezentacja problemu.

Lepszy pomysł reprezentacji:
Do każdego pola wewnątrz rzutu, przypisujemy liczbę całkowitą i rozwiązanie reprezentujemy jako zbiór owych liczb.
Dzięki takiemu podejściu, jestesmy w stanie pominąć kosztowne sprawdzanie poprawności punktów oznaczonych jako leżące w zasięgu kamer, które staraliśmy się optymalizować w trakcie prac nad implementacją.