Politechnika Wrocławska

Wydział Informatyki i Zarządzania
kierunek studiów: Informatyka
specjalność: Inżynieria oprogramowania

Praca dyplomowa - magisterska

Wielokryterialny problem rozmieszczenia zraszaczy wodnych na zadanej powierzchni

Multicriteria water sprinklers deployment problem on a given area

Grzegorz Dziedzic

słowa kluczowe:
optymalizacja wielokryterialna,
algorytmy genetyczne,
zraszacze wodne

krótkie streszczenie:
SHORT ABSTRACT

Promotor:	dr Mariusz Fraś	.....	.....
	imię i nazwisko	*ocena*	*podpis*

*Do celów archiwalnych pracę dyplomową zakwalifikowano do:**

a) kategorii A (akta wieczyste)

b) kategorii BE 50 (po 50 latach podlegające ekspertyzie)

** niepotrzebne skreślić*

pieczątka wydziałowa

Wrocław 2017

Niniejszy dokument został złożony w systemie LaTeX.

Streszczenie

ABSTRACT PL

Abstract

ABSTRACT EN

Spis treści

Rozdział 1

Wstęp

1.1. Wprowadzenie

Odpowiednie nawodnienie ogrodu jest jedną z podstawowych czynności pielęgnacyjnych. Gdy właściciel dysponuje odpowiednim budżetem najlepszym rozwiązaniem będzie dla niego inwestycja w automatyczny system nawadniania. System taki składa się z zraszaczy wodnych, rur pomiędzy nimi oraz systemu sterowania. Takie rozwiązanie pozwala zaoszczędzić czas tracony na ręcznym podlewaniu ogrodu oraz zapewnia równomierne nawodnienie na całej ustalonej powierzchni. Jednym z głównych problemów koniecznych do rozwiązania podczas instalacji takiego systemu jest odpowiednie rozmieszczenie poszczególnych zraszaczy. Te najczęściej znajdują się pod ziemią oraz posiadają wynurzalną głowicę. Z tego powodu raz zainstalowany zraszacz najczęściej zostaje na swoim miejscu, aż do momentu wymiany całej instalacji wodnej. Biorąc to pod uwagę rozmieszczenie zraszaczy powinno być dobrze przemyślane już podczas etapu projektowania systemu nawadniania. Projektując taki system należy przyjąć jako cel nawodnienie całości wskazanego obszaru jak najmniejszym kosztem przy przestrzeganiu wskazanych przez właściciela ograniczeń.

Proces projektowania sieci zraszaczy może być żmudny oraz długotrwały, biorąc pod uwagę różnorodność sprzętu dostępnego na rynku czy chociażby nieregularność powierzchni, która ma zostać nawodniona. Z pomocą może przyjść tutaj jedna z gałęzi matematyki oraz informatyki, która obecna jest wokół nas na każdym kroku. Mowa tutaj o optymalizacji. Czym dokładnie jest optymalizacja zostanie opisane później. W tym momencie ważne natomiast jest to, że opisany powyżej problem idealnie wpasowuje się obszar problemów, których rozwiązanie obejmują procesy optymalizacyjne. Jednym z najczęściej używanych narzędzi do rozwiązywania skomplikowanych problemów optymalizacyjnych stały się algorytmy genetyczne, opierające swoje działanie na procesach zachodzących w przyrodzie.

Praca ta będzie skupiać się na rozwiązaniu omówionego problemu rozmieszczenia zraszaczy wodnych oraz poprowadzenia połączeń pomiędzy nimi poprzez opracowanie systemu wspomagania decyzji oraz implementację i porównanie wielokryterialnych algorytmów genetycznych rozwiązujących omówione zagadnienie.

1.2. Problem nawodnienia obszaru

Jednym z podstawowych problemów z jakim spotykają się właściciele ogrodów jest instalacja odpowiedniego systemu nawadniania. Mogą to zrobić sami albo zlecić zadanie firmie zajmującej się projektowaniem ogrodów i systemów nawadniania. Jest kilka szczególnie ważnych elementów, na które należy zwrócić uwagę projektując taki system:

- Odpowiedni pomiar terenu, który ma zostać nawodniony
- Wzięcie pod uwagę ukształtowania terenu
- Obliczenie potrzebnego ciśnienia wody
- Wybór i rozmieszczenie zraszaczy
- Wybór i poprowadzenie rur
- Umiejscowienie zaworów

Wszystkie wymienione powyżej elementy znacząca wpływają na cenę, jakość oraz wydajność zaprojektowanego systemu.

W tej pracy autor skupia się na dwóch z powyższych punktów: wyborze i rozmieszczeniu zraszaczy oraz poprowadzeniu rur i nie będzie brał pod uwagę reszty wymienionych punktów.

Odpowiednie rozmieszczenie zraszaczy jest ważne, ponieważ zapewnia równomierne nawodnienia.

1.3. Przegląd literatury

Do tej pory problem nawodnienia danego obszaru nie był przedstawiany w literaturze w sposób zaproponowany przez autora. Istnieją jednak publikacje poruszającego problem pokrycia terenu, obejmujący również tematykę tej pracy.

W [8] autorzy zajmują się problemem pokrycia danego obszaru w bezprzewodowych sieciach sensorów ze zmiennym zasięgiem. Sensor jest zdefiniowany jako okrąg o określonym zasięgu oraz współrzędnych środka. Na obszar mający zostać pokryty nakładana jest siatka składająca się z określonej liczby kwadratów. Sensor może zostać ulokowany w środku każdego kwadratu. Autorzy w swojej pracy określili trzy kryteria względem których znalezione rozwiązania są oceniane. Są to:

1. Stopień pokrycia terenu.
2. Koszt rozmieszczonych sensorów.
3. Koszt energetyczny rozmieszczonych sensorów.

Jak widać problem przedstawiony w publikacji jest problemem wielokryterialnym. W celu rozwiązania przedstawionego problemu autorzy zdecydowali się na wykorzystanie algorytmu genetycznego NSGA-II.

W zaprezentowanych rezultatach autorzy zaczynali symulacje losowo rozmieszczając 500 sensorów. W 800. pokoleniu algorytm znalazł rozwiązanie składające się z 9 sensorów pokrywających 100% wyznaczonego terenu oraz charakteryzujące się kosztem energetycznym na poziomie 0.576 (6-krotna poprawa względem 100. pokolenia).

Przedstawione wyniki są dowodem na odpowiednie podejście autorów do wielokryterialnego problemu pokrycia obszaru. Do optymalizacji rozmieszczenia zraszaczy wodnych wykorzystano model matematyczny opisany w przedstawionej publikacji w odpowiednio dopasowanej do problemu formie oraz wzorowano się na sposobie reprezentacji osobników w wykorzystanym algorytmie genetycznym.

Autorzy [17] skupiają się na porównaniu kilku wielokryterialnych algorytmów genetycznych. Do porównań wybrali sześć funkcji testowych. Każda z nich charakteryzowała się tym, że sprawiała pewne trudności algorytmom genetycznym. Trudności te najczęściej skupiały się wokół problemu z osiąganiem prawdziwego frontu Pareto. Ponadto autorzy zdefiniowali trzy metryki służące do oceny znalezionych rozwiązań oraz skonstruowali ranking obrazujący jakość działania każdego z wybranych algorytmów.

Kolejną pracą skupiającą się na porównaniu algorytmów genetycznych jest [7]. W tej konkretnej publikacji autorzy starają się zaprojektować budynek o zerowej konsumpcji energii. Jest to budynek, który zapotrzebowanie na energię zaspokaja korzystając ze źródeł energii odnawialnej. Autorzy do porównania wybrali siedem wielokryterialnych algorytmów genetycznych oraz zdefiniowali metryki służące do ich oceny.

Dwie powyższe pracy pomogły autorowi wybrać dwa algorytmy użyte do rozwiązania problemu rozmieszczenia zraszaczy wodnych oraz pomogły zdefiniować kryteria wykorzystane do oceny wybranych algorytmów.

W pracy [12] autor skupia się na problemie minimalizacji wielkości systemu rur wodnych miasta Agric Nsima. Jest to motywowane zwiększającą się populacją, a tym samym zapotrzebowaniem na wodę. Autor poprzez rozwiązanie przedstawionego problemu chce zredukować koszty konieczne to do funkcjonowania oraz utrzymania takiego systemu.

Autor w swojej pracy sugerują, że najlepszym rozwiązaniem okazał się algorytm Prima, służący do znajdowania minimalnego drzewa rozpinającego grafu. Do poprowadzenia rur pomiędzy rozmieszczonymi zraszaczami zdecydowano na wykorzystanie proponowanego przez autora rozwiązania.

1.4. Cel pracy

Opracowanie informatycznego systemu wspomagania podejmowania decyzji w rozmieszczeniu zraszaczy wodnych na zadanej powierzchni oraz poprowadzeniu rur pomiędzy nimi, a w szczególności zaproponowanie oraz przebadanie wielokryterialnych algorytmów genetycznych w kontekście wybranego problemu.

1.5. Opis pracy

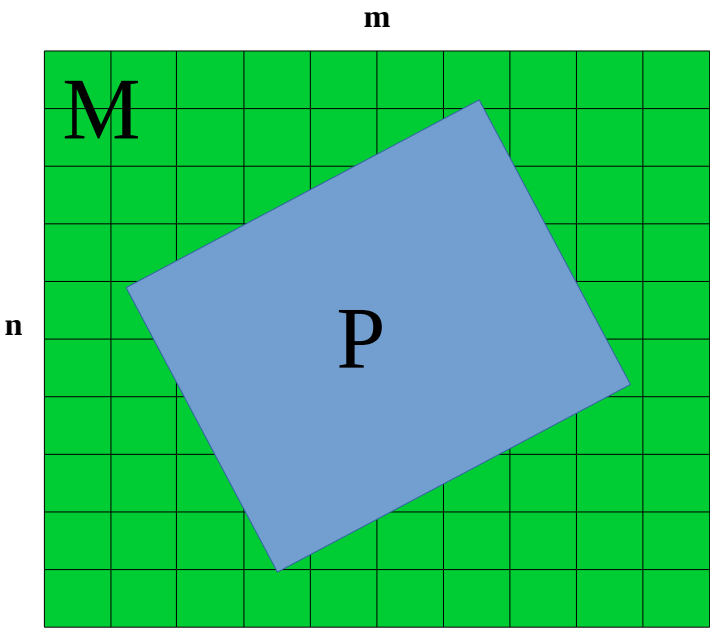
W kolejnych rozdziałach opisane będą poszczególne zagadnienia związane z realizacją celu pracy. Na samym początku sformułowany zostanie model matematyczny opisujący rozwiązywany problem. Rozdział ten zawierał będzie dokładny opis podejścia do danego zagadnienia wraz z opisem wzorów wykorzystanych do zdefiniowania funkcji celu. Następnie wyjaśnione zostaną pojęcia optymalizacji, optymalizacji jednokryterialnej oraz optymalizacji wielokryterialnej, czyli zagadnienia na których bazuje cała praca. W kolejnym rozdziale przedstawione zostaną algorytmy genetyczne, zarówno

te w wersji podstawowej jak i wielokryterialnej wraz z ich wybranymi wersjami, tak aby przybliżyć czytelnikowi w jaki sposób ona działają oraz dlaczego są używane. W następnej kolejności zostanie wyjaśnione czym jest system wspomagania decyzji, jakie założenia powinien spełniać oraz jakimi funkcjonalnościami się wyróżniać. Kolejne rozdziały będą już odzwierciedleniem wykonanej pracy i dokładnym opisem przyjętego sposobu realizacji zadania. Najpierw dokładnie opisany zostanie opracowany system wspomagania decyzji. Wymienione zostaną technologie użyte do implementacji oraz przykładowy scenariusz interakcji użytkownika z system wraz z zrzutami ekranu prezentującymi poszczególne widoki aplikacji. W kolejnym rozdziale zaprezentowana zostanie porównanie ze sobą wybranych algorytmów genetycznych. Przedstawiony zostanie plan badań oraz ich rezultaty wraz z wynikającymi z nich wnioskami. Na końcu pracy znajdzie się podsumowanie oceniające przedstawione rozwiązanie oraz propozycje dalszego rozwoju opracowanego systemu oraz badań.

Rozdział 2

Model matematyczny

Załóżmy, że P jest wielokątem odwzorowującym kształt terenu, który ma zostać nawodniony. Poprzez opisanie danego wielokąta prostokątem otrzymujemy teren roboczy A. Szerokość i wysokość tego terenu są dodatkowo powiększone o największy możliwy zasięg zraszaczy, tak aby umożliwić odpowiednie sprawdzenie stopnia naruszenia ograniczeń, o których mowa później. A jest dodatkowo podzielony na $m \times n$ kwadratów tworząc macierz M przedstawioną na rysunku (2.1). Każdy element macierzy może przyjmować wartości od 0 do ms, gdzie ms jest maksymalną liczbą zraszaczy. Wartość 0 oznacza, że dany kwadrat nie został podlany. Wartość większa od 0 mówi o tym przez ile zraszaczy dany obszar został nawodniony.



Rys. 2.1: blah

Załóżmy, że $p(x, y)$ jest funkcją mówiącą o tym czy punkt o współrzędnych (x, y)

znajduje się wewnątrz P. Jeśli $p = 0$ punkt należy do P, w przeciwnym wypadku $p = 1$. Zraszacz jest definiowany jako okrąg reprezentowany przez promień r_i, współrzędne środka (x_i, y_i) oraz konkretny model t_i. Środek każdego zraszacza musi zawierać się w P, czyli $p(x_i, y_i) = 0$.
Kwadrat o współrzędnych (x, y) jest nawodniony jeśli znajduje się w zasięgu zraszacza. Jest to wyrażone następującą funkcją:

$$is\_irr(x, y, s_i) = \begin{cases} 1, & \text{jeśli } (x - x_i)^2 + (y - y_i)^2 \leqslant r_i^2 \\ 0, & w.p.p \end{cases} \tag{2.1}$$

, gdzie x oraz y oznaczają współrzędne środka kwadratu; s_i jest konkretnym zraszaczem; x_i, y_i oznaczają współrzędne zraszacza s_i; r_i jest liczba wyrażającą zasięg zraszacza.

Załóżmy, że S jest zbiorem wszystkich rozmieszczonych zraszaczy. Wtedy dany kwadrat jest nawodniony jeśli znajduje się w zasięgu przynajmniej jednego zraszacza ze zbioru S:

$$irr(x, y, S) = \begin{cases} 1, & \text{jeśli } \sum_{i=1}^{S} is\_irr(x, y, s_i) > 0 \\ 0, & w.p.p \end{cases} \tag{2.2}$$

Generalnie poprzez stopień nawodnienia danego obszaru rozumie się różnicę pomiędzy liczbą wszystkich elementów macierzy M należących do P, a liczbą nawodnionych elementów tej macierzy należących do P:

$$F_0(S) = \sum_{m=1}^{M} \sum_{n=1}^{N} p(m, n) \cdot (1 - irr(m, n, S)) \tag{2.3}$$

Im wartość funkcji F_0 jest mniejsza tym mniej jest obszarów nienawodnionych, a więc rozwiązanie jest lepsze. $F_0 =$ jest rozwiązaniem idealnym z perspektywy kryterium nawodnienia.

Jak zostało wspomniane w poprzednich rozdziałach zaimplementowane algorytmy oceniają stopień nawodnienia obszaru na dwa sposoby w zależności od tego czy wybrana jest opcja minimalizacji nawodnienia poza wskazanym obszarem. Jeśli minimalizacja jest wyłączona do obliczeń używana jest prosta funkcja opisana powyżej (2.3). Bardziej szczegółowe rozwinięcie funkcji przewiduje wprowadzenie kary za naruszenie ograniczeń:

$$\begin{aligned} F_1(S) &= \sum_{m=1}^{M} \sum_{n=1}^{N} p(m, n) \cdot (1 - irr(m, n, S)) \\ +\alpha \cdot \Bigg[&\frac{\sum_{m=1}^{M} \sum_{n=1}^{N} (1 - p(m, n)) \cdot irr(m, n, S) \cdot 100}{\sum_{m=1}^{M} \sum_{n=1}^{N} p(m, n)} \\ &+ \frac{\sum_{m=1}^{M} \sum_{n=1}^{N} p(m, n) \cdot ovrirr(m, n, S) \cdot 100}{\sum_{m=1}^{M} \sum_{n=1}^{N} p(m, n)} \Bigg] \end{aligned} \tag{2.4}$$

, gdzie α jest zmienną określającą to czy na funkcję celu wpływa minimalizacja nawodnienia poza wskazanym obszarem. Zmienna α może przyjmować wartość 0 lub 1. Jeśli

$\alpha = 1$ minimalizacja jest brana pod uwagę; $ovrirr(m, n, S)$ jest funkcją mówiącą czy dany obszar został nadmiernie nawodniony (2.5).

$$ovrirr(x, y, S) = \begin{cases} 1, & \text{jeśli } \sum_{i=1}^{S} is\_irr(x, y, s_i) > 1 \\ 0, & w.p.p \end{cases} \tag{2.5}$$

Jak zostało wspomniane wcześniej, każdy rozmieszczony zraszacz charakteryzowany jest przez konkretny model. Załóżmy, że $C(t_i)$ jest funkcją zwracającą cenę rynkową dla danego modelu t_i. Wtedy całkowity koszt konkretnego rozwiązania wynosi:

$$F_2(S) = \sum_{i=1}^{S} C(t_i) \tag{2.6}$$

Podsumowując celem zadania jest rozwiązanie następującego problemu optymalizacyjnego:

$$\begin{aligned} min\ F_1(S) &= \sum_{m=1}^{M}\sum_{n=1}^{N} p(m,n)\cdot(1 - irr(m,n,S)) \\ &+\alpha\cdot\left[\frac{\sum_{m=1}^{M}\sum_{n=1}^{N}(1-p(m,n))\cdot irr(m,n,S)\cdot 100}{\sum_{m=1}^{M}\sum_{n=1}^{N} p(m,n)}\right. \\ &\left.+\frac{\sum_{m=1}^{M}\sum_{n=1}^{N} p(m,n)\cdot ovrirr(m,n,S)\cdot 100}{\sum_{m=1}^{M}\sum_{n=1}^{N} p(m,n)}\right] \end{aligned} \tag{2.7}$$

$$min\ F_2(S) = \sum_{i=1}^{S} C(t_i)$$

Rozdział 3

Optymalizacja

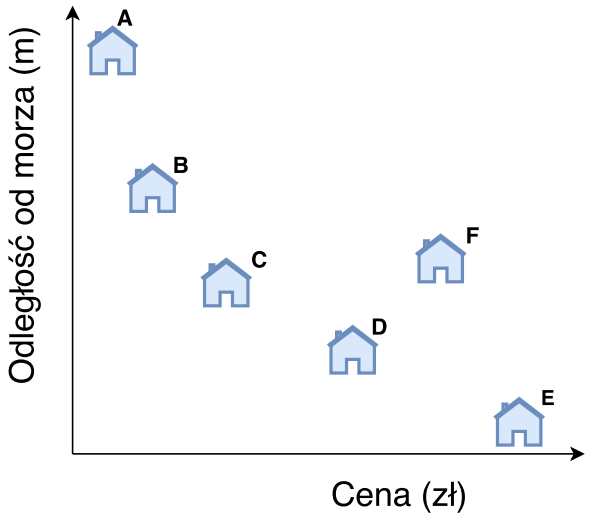
Optymalizacja jest procesem polegającym na wyznaczeniu najlepszego rozwiązania lub rozwiązań z punktu widzenia danego kryterium lub kryteriów [14]. Procesy takie w w obecnym świecie obecne są wszędzie. Zaczynając od mechaniki, poprzez ekonomie, finanse, elektrykę po geofizykę czy modelowanie molekularne. Przykładem może być konieczność minimalizacji kosztów produkcji w fabryce, maksymalizacja prawdopodobieństwa zysku z przeprowadzanych transakcji czy chociażby znalezienie najbardziej odpowiadającego danej osobie modelu samochodu [15].

W kolejnych podrozdziałach dokładnie opisane zostaną zagadanie powiązane kolejno z optymalizacją jednokryterialną oraz wielokryterialną.

3.1. Optymalizacja jednokryterialna

Optymalizacja jednokryterialna, jak wskazuje nazwa, odnosi się do problemów opisywanych za pomocą jednej funkcji celu. Przykładem może być tutaj problem spotykający ludzi wybierających się na wakacje, a więc poszukiwanie najlepszego hotelu w danym mieście. Załóżmy, że jedynym kryterium branym pod uwagę w trakcie szukania takiego hotel jest cena ze spędzoną noc. Przykładem będzie tutaj sytuacja przestawiona na rysuku (3.1). Jak łatwo zauważyć najtańszy jest hotel A. Można więc powiedzieć, że hotel A jest rozwiązaniem przedstawionego problemu optymalizacyjnego. Patrząc na to z drugiej strony, możemy stwierdzić, że cena hotelu nie ma dla nas żadnego znaczenia, a zależy nam jedynie na tym aby ten znajdował się jak najbliżej morza. Przy takiej zmianie kryterium, zmienia się również rozwiązanie problemu. Najlepszy staje się hotel E, ponieważ to właśnie on znajduje się najbliższej morza.

Z matematycznego punktu widzenia rozwiązanie jednokryterialnego problemu optymalizacyjnego sprowadza się do znalezienia ekstremum funkcji opisującej wybrane kryterium oceny [15].



Rys. 3.1: Powiązanie pomiędzy ceną hotelu, a dystansem od morza

3.2. Optymalizacja wielokryterialna

Jeśli problem optymalizacyjny zawiera w sobie kilka funkcji celu, proces znajdowania optymalnego rozwiązania bądź rozwiązań nazywa się optymalizacją wielokryterialną. Zdecydowana większość realnych problemów jest właśnie przykładem wymagającym wykorzystania optymalizacji wielokryterialnej [4]. W takich przypadkach nie można skupiać się jedynie na jednym kryterium, ponieważ pozostałe są równie ważne. Przykładem może być tutaj problem wyboru hotelu na wakacje, opisany w poprzednim podrozdziale.

Załóżmy, że tym razem oba kryteria przedstawione na rysunku (3.1) są dla nas tak samo ważne. Gdyby było inaczej i wszyscy kierowali by się jedynie ceną hotelu, wszyscy chcieliby mieszkać w hotelu A. Oczywiście w realnym świecie taka sytuacja nigdy nie będzie miała miejsca, ponieważ w większości przypadków proces decyzyjny nie jest taki prosty.

Jak możemy zauważyć do wyboru mamy 6 różnych hoteli. Każdy z nich charakteryzuje się ceną za jedną noc oraz odległością od morza wyrażoną w metrach. Do porównania wybierzmy dwa hotele: D oraz F. Jak widać hotel D jest zarówno tańszy jak i znajduje się bliżej morza w porównaniu do hotelu F. W tym przypadku decyzja o wyborze jest oczywista i jest nią hotel D. Sytuacja wygląda inaczej, gdy porównamy hotel D z hotelem C. Hotel D znajduję się bliżej plaży, natomiast jest również droższy od C. Które rozwiązanie w tej sytuacji będzie lepsze? To już zależy wyłącznie od preferencji konkretnej osoby. Nie da się bowiem stwierdzić, które z wybranych rozwiązań jest lepsze patrząc z perspektywy obu kryteriów. Z tego powodu problemy, w których występuje kilka kryteriów negatywnie na siebie wpływających, charakteryzują się posiadaniem wielu rozwiązań optymalnych [5].

Dominacja

Jednym z głównych pojęć, na których opiera się optymalizacja wielokryterialna jest pojęcie dominacji. Jednym z celów rozwiązywania problemów wielokryterialnych jest

znalezienie rozwiązań niezdominowanych, co dokładniej wyjaśnione zostanie później. Koncept dominacji polega na porównywaniu ze sobą znalezionych rozwiązań przez pryzmat wszystkich określonych kryteriów oceny. Załóżmy, że M określa liczbę kryteriów. Wtedy można powiedzieć, że rozwiązanie A dominuje rozwiązanie B jeśli:

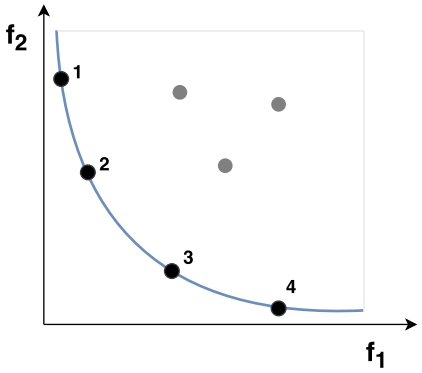
1. Dla każdego kryterium $m \in M$ rozwiązanie A nie jest gorsze od B.
2. Przynajmniej dla jednego kryterium $m \in M$ rozwiązanie A jest lepsze od B [4].

Korzystając z przykładu opisanego wcześniej przykładu hoteli, możemy zauważyć, że rozwiązanie F jest zdominowane przez rozwiązanie D. Jest tak dlatego, że rozwiązanie D jest pod względem każdego kryterium lepsze od rozwiązania F, a więc spełnione są wymienione powyżej warunki. Nie można tego samego powiedzieć, gdy do porównania użyjemy hotelu A oraz B. Rozwiązanie A nie spełnia pierwszego warunku - hotel jest położony dalej od morza.

Optimum Pareto

Rozwiązanie jest rozwiązaniem optymalnym w sensie Pareto, jeśli nie da się poprawić danego rozwiązania pod względem jednego kryterium, nie pogarszając jednocześnie oceny rozwiązania, patrząc z perspektywy pozostałych kryteriów [16]. Spójrzmy na hotel C z wykorzystywanego wcześniej przykładu oraz załóżmy, że jest to rozwiązanie optymalne w sensie Pareto. W takiej sytuacji nie da obniżyć się ceny za pobyt w hotelu bez zmiany jego lokalizacji, oddalającej go od plaży. Sytuacja wygląda identycznie w drugą stronę. Nie da się zmienić położenia hotelu na bliższe morzu, bez zwiększenia ceny za noc. Optimum Pareto jest więc kompromisem pomiędzy pomiędzy uwzględnianymi kryteriami.

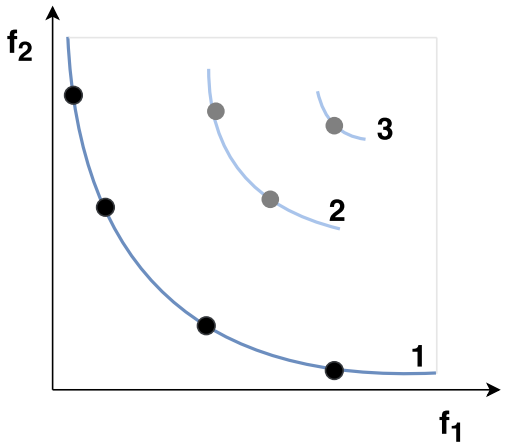
Grupa rozwiązań optymalnych w sensie Pareto tworzy tzw. front lub zbiór Pareto. Rozwiązanie optymalne w sensie Pareto jest jednocześnie rozwiązaniem niezdominowanym, tak więc front Pareto jest również zbiorem rozwiązań niezdominowanych [4]. Przykład takiego frontu, w postaci niebieskiej linii, został przestawiony na rysunku (3.2). Rozwiązania (1-4) leżące na froncie są rozwiązaniami optymalnymi w sensie Pareto, które dominują pozostałe rozwiązania znajdujące się na rysunku.



Rys. 3.2: Front Pareto

Przedstawiona powyżej definicja frontu Pareto może zostać wykorzystana w dwóch kontekstach: lokalnym oraz globalnym. Front Pareto w sensie globalnym jest zbiorem

rozwiązań niezdominowanych w kontekście całej przestrzeni możliwych rozwiązań. Taki front nazywany jest również prawdziwym frontem Pareto [4]. Lokalny front Pareto natomiast, jest grupą rozwiązań niezdominowanych ale tylko w określonym zakresie przestrzeni. Biorąc to pod uwagę dowolną grupę rozwiązań można podzielić na n frontów, gdzie n jest liczbą frontów potrzebnych do objęcia wszystkich rozwiązań z grupy. Zostało to zobrazowane na rysunku (3.3). Liczby znajdujące się przy frontach oznaczają ich miejsce w hierarchii. Najlepszy front oznaczony jest liczbą 1, kolejny liczbą 2 itd.. Warto tutaj zauważyć, że rozwiązania znajdujące się na drugim froncie nie są niezdominowane w sensie globalnym (dominują je rozwiązania z frontu pierwszego). Są natomiast optymalne w sensie lokalnym, czyli po ograniczeniu przestrzeni rozwiązań do tych nieznajdujących się na pierwszym froncie.



Rys. 3.3: Lokalne fronty Pareto

Znajdowanie rozwiązań niezdominowanych

Kolejnym ważnym zagadnieniem jest proces znajdowania rozwiązań niezdominowanych w danej grupie rozwiązań[4]. Proces ten jest wykorzystywany w większości metod służących do optymalizacji wielokryterialnej, m.in. w wielokryterialnych algorytmach genetycznych opisanych w kolejnym rozdziale.

Istnieje wiele sposobów na posortowanie grupy ze względu na kryterium dominacji. Najprostszym z nich jest podejście iteracyjne. Polega ono na porównaniu danego rozwiązania i, z każdym innym rozwiązaniem. Jeśli rozwiązanie i jest zdominowane przez jakiekolwiek inne rozwiązanie, znaczy to, że nie może zostać dodane do zbioru rozwiązań niezdominowanych. Generalnie dla danego zbioru rozwiązań S o rozmiarze N, algorytm iteracyjny może zostać opisany następującymi krokami:

1. Utwórz pusty zbiór rozwiązań niezdominowanych P'.
2. Ustaw licznik $i = 0$.
3. Dla każdego rozwiązania j, takiego że $j \neq i$, sprawdź czy rozwiązanie j dominuje rozwiązanie i. Jeśli tak przejdź do punktu 5.
4. Jeśli $j < N$, zwiększ licznik j oraz przejdź do punktu 2. Jeśli $j = N$, dodaj rozwiązanie i do zbioru rozwiązań niezdominowanych.

5. Jeśli $i < N$, zwiększ licznik i oraz przejdź do punktu 2. W innym wypadku P' jest końcowym zbiorem rozwiązań niezdominowanych.

Przedstawiony powyżej algorytm jest prosty ale też cechuje się dużą złożonością obliczeniową. Łatwo zauważyć, że złożoność punktu 3. w najgorszym przypadku wynosi $O(MN)$. Z racji tego, że punkt 4. rekursywnie wywołuje punkt 2, wymaga on złożoności obliczeniowej $O(MN^2)$. Warto zauważyć, że jest to najgorszy możliwy przypadek, ponieważ w drugim kroku nie zawsze konieczne jest wykonanie wszystkich $(N-1)$ porównań[4].

Lepszym rozwiązaniem jest zastosowanie efektywnej metody opracowanej przez Kunga wraz z innymi autorami (Kung et al.'s Efficient Method). Jak zostało wykazane metoda ta cechuje się złożonością obliczeniową $O(N(\log N)^{M-2})$ dla ponad trzech kryteriów oceny oraz $O(N \log N)$ dla dwóch i trzech kryteriów oceny [9].

Pierwszym krokiem metody jest posortowanie grupy rozwiązań na podstawie wartości pierwszej funkcji oceny, w kolejności od najlepszego do najgorszego. Następnie grupę dzielona jest na połowę, tworząc dwie podgrupy: górną (T) oraz dolną (B). Sortowanie zapewniło, że rozwiązania z grupy T są lepsze od rozwiązań z grupy B z perspektywy pierwszego kryterium. Kolejnym krokiem jest porównywanie rozwiązań z dolnej grupy z rozwiązaniami z grupy górnej w kontekście dominacji. Rozwiązanie niezdominowane są przenoszone do górnej grupy. Operacje łączenia oraz porównania zaczynają być wykonywane, gdy w wyniki rekursywnego podziału grupy na dwie podgrupy, w danej podgrupie znajduje się jedynie jedno rozwiązanie. Przedstawiony proces może zostać zobrazowany w formie kodu głównej funkcji:

Kod źródłowy 3.1: Znajdowanie rozwiązań niezdominowanych

```
def Front(P):
        if len(P) == 1:
                return P
        T = Front(P[:len(pop)/2])
        B = Front(P[len(pop)/2:])

        NotDominatedSolutions = []
        for B_Solution in B:
                NotDominated = True
                for T_Solution in T:
                        if T_Solution.Dominates(B_Solution):
                                NotDominated = False
                                break
                if NotDominated:
                        NotDominateSolutions.append(B_Solution)
        NotDominatedSolutions += T
        return NotDominatedSolutions
```

Kod został napisany w języku Python 2.7. Argumentem funkcji (P) jest posortowana lista rozwiązań.

Ze względu na najlepszą złożoność obliczeniową metoda opisana powyżej została zaimplementowana w wybranych algorytmach genetycznych.

Sortowanie rozwiązań niezdominowanych

Większość z metod służących do rozwiązywania problemów wielokryterialnych wykorzystuje koncept znajdowania zbioru rozwiązań niezdominowanych. Istnieją również algorytmy, w tym opisywany później NSGA-II, które wymagają podziału całej populacji na kolejne niezdominowane grupy (fronty)[4]. W takim przypadku populacja jest przechowywana jako zbiór frontów ułożonych w kolejności od najlepszego (front pierwszy) do najgorszego.

Istnieje prosta procedura służąca do dzielenia całej populacji na poszczególne fronty[3]. Wykorzystuje ona metody opisane w poprzedniej sekcji. Po znalezieniu pierwszego frontu, rozwiązania należącego do niego stają się tymczasowo ignorowane. Następnie procedura jest powtarzana, w celu znalezienia kolejnego frontu. Ponownie rozwiązania leżące na danym froncie są dodawane do grupy rozwiązań ignorowanych. Proces jest powtarzany, dopóki każde rozwiązanie z populacji nie jest przyporządkowane do odpowiedniego frontu.

Cele optymalizacji wielokryterialnej

Jak zostało wspomniane wcześniej rezultatem procesu optymalizacji wielokryterialnej, nie jest jedno rozwiązanie, lecz grupa rozwiązań. Grupę taką ocenia się z perspektywy dwóch kryteriów[4]. Po pierwsze grupa ta powinna znajdować się jak najbliższej prawdziwego frontu Pareto. Rozwiązanie idealne odwzorowuje sytuację kiedy każde znalezione rozwiązanie jest optymalne w sensie Pareto. Po drugie znaleziona grupa powinna cechować się różnorodnością. Jest to ważne z perspektywy zachowania takiej samej wagi dla każdego ustalonego kryterium. Warto tutaj wspomnieć, że proces optymalizacji wielokryterialnej nie jest odpowiedzialny za wybór jednego konkretnego rozwiązania, które będzie najlepsze dla konkretnego użytkownika. Wynikiem takiej optymalizacji ma być natomiast zbiór równie dobrych rozwiązań, z których użytkownik będzie mógł wybrać najbardziej dla siebie odpowiednie. Im większa różnorodność rozwiązań tym więcej opcji do wyboru ma użytkownik.

Oba wymienione kryteria są tak samo ważne, dlatego algorytmy służące do optymalizacji wielokryterialnej powinny skupiać się na zaspokojeniu obu tych kryteriów w takim samym stopniu[4].

Rozdział 4

Algorytmy genetyczne

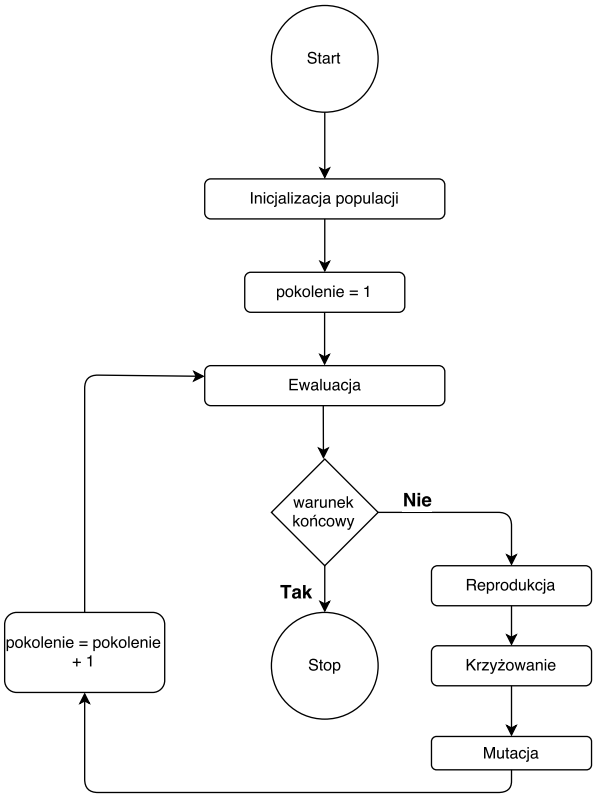
Problem rozmieszczenia zraszaczy opisany na wstępie pracy został rozwiązany przy użyciu algorytmów genetycznych. Są to algorytmy, których działanie oparte jest na tych samych zasadach na których odbywa się proces ewolucji zachodzący w naturze [11]. W tym rozdziale przedstawiony zostanie opis ogólny algorytmu genetycznego, jego powiązanie z procesami zachodzącymi w przyrodzie, schemat działania oraz dokładny opis wykorzystywanych operatorów: krzyżowania, selekcji oraz mutacji. Następnie przedstawiony zostanie koncept wielokryterialnych algorytmów genetycznych oraz opisane zostaną dwa algorytmy wykorzystane do rozwiązania przedstawionego wcześniej problemu: NSGA-II oraz SPEA. Kolejny podrozdział będzie skupiał się na procesie strojenia algorytmów. Następnie opisana zostanie konkretna implementacja zastosowana w celu rozwiązania problemu nawodnienia.

4.1. Opis ogólny

Algorytmy genetyczne są procedurami wykorzystywanymi do rozwiązywania problemów optymalizacyjnych, opartymi o genetykę oraz selekcję naturalną. Cały proces działania takiego algorytmu można porównywać do procesu ewolucji występującego w przyrodzie[11]. Obserwując taki proces można zauważyć, że najlepiej przystosowane osobniki mają największe szanse na reprodukcję, a więc przekazanie swojego DNA kolejnym pokoleniom. Dzieje się tak dlatego, że z punktu widzenia biologii najlepiej przystosowane osobniki to te najbardziej atrakcyjne dla potencjalnych partnerów oraz w przypadku zwierząt stadnych, te dominujące całą resztę, a więc mające pierwszeństwo jeśli chodzi o wybór partnerów, reprodukcję czy chociażby pożywianie się. Generalizując natura dąży do tego, aby do puli z której utworzone ma zostać kolejne pokolenie danej populacji trafiały jak najlepsze geny. Gdyby działo się inaczej żaden gatunek nie przetrwałby więcej niż kilkudziesięciu pokoleń.
Jak zostało wspomniane największe szanse na potomstwo mają osobniki najlepsze patrząc z perspektywy szansy na przetrwanie. Potomstwo par takich osobników powstaje na skutek procesu nazywanego krzyżowaniem. Krzyżowanie polega na łączeniu genów jednego osobnika z drugim, tworząc w ten sposób potomka, charakteryzującego się cechami zarówno pierwszego jak i drugiego rodzica. Oczywiście im lepiej rodzice danego

osobnika byli przystosowani, tym większa szansa, że również i on będzie stał wysoko w hierarchii.
Kolejnym istotnym elementem w procesie ewolucji jest mutacja. Mutacja jest losowym procesem polegającym na zmianie fragmentu kodu DNA. Poprzez taką zmianę osobnik staje się jeszcze lepiej lub gorzej przystosowany. W pierwszym przypadku jego szanse na reprodukcję wzrastają, więc jest większe prawdopodobieństwo, że zmiana wprowadzona przez mutację zostanie przekazana dalej. W drugim przypadku linia osobników ze zmienionym DNA prawdopodobnie wygaśnie zdecydowanie szybciej niż w przypadku pierwszym.

Podsumowując, proces ewolucji złożony jest z trzech elementów: selekcji, krzyżowania oraz mutacji[11]. Na tych samych operacjach opiera się działanie algorytmu genetycznego. W kontekście algorytmu są one nazywane operatorami genetycznymi. Na rysunku (4.1) przedstawiony został ogólny schemat działania takiego algorytmu.



Rys. 4.1: Schemat działania algorytmu genetycznego

Punktem startowym jest utworzenie losowej populacji osobników. Następnie każdy osobnik jest oceniany przez pryzmat ustalonego kryterium. Na podstawie oceny przeprowadzany jest proces selekcji, czyli wyboru najlepszych osobników, które wykorzystane zostaną do reprodukcji. Podczas procesu reprodukcji generowane są nowe osobniki tworzące kolejne pokolenie. Każdy nowy osobnik jest wynikiem krzyżowania, czyli łączenia chromosomu dwóch rodziców w jeden. Chromosom z punktu widzenia biologii jest formą organizacji materiału genetycznego. Z punktu widzenia algorytmu będzie to zestaw parametrów charakteryzujących danego osobnika, który zostanie opisany później. Następnie chromosomy nowych osobników mogą ulec mutacji, czyli losowej zmia-

nie pojedynczych genów. Gdy zostanie utworzona odpowiednia liczba nowych osobników, czyli powstanie nowe pokolenie, cały proces zaczyna się od początku, dopóki nie osiągnięte zostanie zdefiniowane kryterium końcowe.

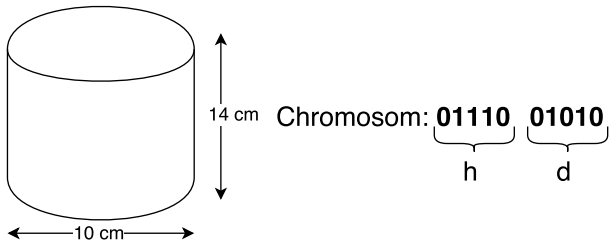
W powyższym opisie pojawiło się kilka pojęć wartych wyjaśnienia.

Populacja

Populacja jest grupą o ustalonym rozmiarze. Elementami grupy są pojedyncze osobniki. Jedną z przewag algorytmów genetycznych nad tradycyjnymi technikami optymalizacyjnymi jest właśnie operowanie nie na jednym konkretnym rozwiązaniu, lecz na całej grupie. Duża różnorodność osobników w połączeniu z odpowiednim użyciem operatorów genetycznych pozwala uniknąć problemu utknięcia na optimum lokalnym.

Chromosom

Chromosom, jak zostało już wspomniane, jest sposobem reprezentacji właściwości cechujących danego osobnika. Chromosom można zdefiniować na dwa sposoby: za pomocą ciągu binarnego lub wartości rzeczywistych. Ciąg binarny jest ciągiem wartości 0 lub 1 ułożonych w taki sposób aby odpowiednio zakodować przekazywaną informację. W ramach przykładu załóżmy, że mamy dany cylinder o średnicy $d = 10cm$ oraz wysokości $h = 14cm$ [4]. Rysunek (4.2) pokazuje przykładowy sposób reprezentacji takiego cylindra w formie ciągu binarnego.



Rys. 4.2: Przykład ciągu binarnego

Chromosom reprezentujący cylinder złożony jest z 10 bitów. Na pierwszych pięciu z nich zakodowana jest informacja o wysokości, a na kolejnych informacja o średnicy. Wartości przedstawione przez poszczególne części ciągu należy traktować jak liczby binarne i w ten sam sposób je interpretować.

W większości przypadków preferowane jest używanie ciągu binarnego jako sposobu reprezentacji osobników. Jednym z powodów jest łatwość przeprowadzania operacji krzyżowania oraz mutacji, o których mowa będzie później. Używając reprezentacji binarnej można w prosty sposób zmienić sposób kodowania osobników bez konieczności ingerowania w operatory genetyczne - operatory krzyżowania oraz mutacji nie biorą pod uwagę sposobu w jaki informacja została zapisana w chromosomie. Kolejne argumenty przemawiające za użyciem takiej reprezentacji przedstawione zostały w pracach Goldberga [6] oraz Reeves'a [13]. Kolejne etapy działania algorytmów genetycznych wyjaśnione zostaną przy założeniu, że osobniki przedstawione są za pomocą ciągów binarnych.

Ewaluacja

Po ustaleniu sposobu reprezentacji osobników należy wygenerować losową populację oraz poddać ją ocenie, czyli ewaluacji. Proces ewaluacji polega na przypisaniu każdemu osobnikowi odpowiedniej wartości pozwalającej ocenić go na tle reszty populacji. Jest to wartość dopasowania osobnika. Korzystając z wcześniejszego przykładu załóżmy, że podczas procesu ewaluacji zdefiniowanemu wcześniej cylindrowi przypisywany jest koszt wyprodukowania, wyrażony następującym wzorem:

$$f(d,h) = c\Big(\frac{\pi d^2}{2} + \pi dh\Big) \tag{4.1}$$

, gdzie c oznacza koszt materiału na cm^2 [4].
Zgodnie z podanymi wcześniej danymi oraz przy założeniu, ze $c = 0.01$ koszt wyprodukowania danego cylindra wynosi:

$$f(10,14) = 6$$

Zakładając, że celem procesu optymalizacji jest minimalizacja kosztu wyprodukowania cylindra, należy zauważyć, że osobnik z przypisaną mniejszą wartością jest lepszy w kontekście wybranego kryterium.

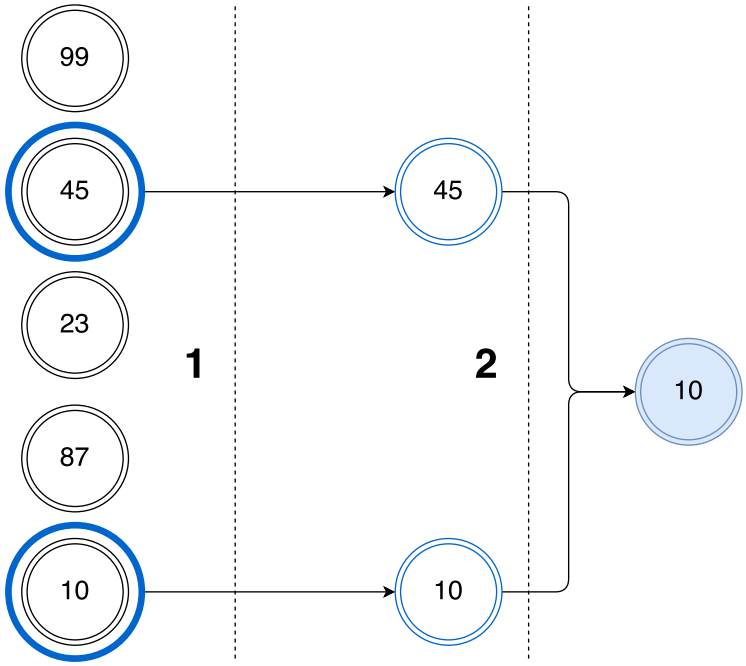
Reprodukcja

Kolejnym etapem działania algorytmu genetycznego jest reprodukcja inaczej zwana też selekcją. Jest to proces mający na celu zwiększenie liczby dobrych osobników w populacji oraz redukcję liczby tych gorszych, jednocześnie utrzymując niezmieniony rozmiar całej populacji. Jest to osiągane poprzez realizację następujących zadań:

1. Zidentyfikowanie dobrych (ocenionych powyżej średniej) osobników.
2. Utworzenie kopii dobrych osobników.
3. Eliminacja osobników gorszych, tak aby kopie dobrych mogły zostać dodane do populacji.[4]

Powyższe zadania mogą zostać osiągnięte przy użyciu kilku metod. Najpopularniejszymi z nich są: metoda turniejowa, metoda ruletki oraz metoda rankingu. Metoda turniejowa polega na rozgrywaniu turniejów pomiędzy kilkoma osobnikami wybranymi losowo z populacji [10]. Zwycięzca każdego turnieju, jako rodzic, przechodzi do etapu krzyżowania, który zostanie opisany później(4.3). Rozmiar pojedynczego turnieju znacząco wpływa na przebieg procesu selekcji. Im jest większy, tym mniejsza szansa, że słabsze osobniki przejdą do kolejnego etapu. Wbrew pozorom takie zachowanie nie jest pożądane ze względu na potrzebę zachowania różnorodności populacji. Im rozmiar jest większy tym większa szansa, że do kolejnego etapu dostanie się zbyt dużo słabszych osobników, co również nie jest dobrym rozwiązaniem. Jeśli rozmiar turnieju $S = 1$ selekcja staje się równoważna losowemu wyborowi osobnika. Najczęściej stosuje się turnieje o rozmiarze 2. Metoda turniejowa jest jedną z najczęściej wybieranych metod selekcji. Jest prosta w implementacji oraz pozwala na łatwe manipulowanie rozmiarem turnieju, co z kolei wpływa na szybkość zbliżania się znalezionych rozwiązań do rozwiązań optymalnych. Ponadto jak zostało udowodnione selekcja turniejowa zawsze zapewnia lepszą lub taką samą szybkość zbliżania się do rozwiązań optymalnych jak

inne metody selekcji oraz mniejszą lub taką samą złożoność obliczeniową [4]. Z wymienionych powyżej powodów to właśnie metoda turniejowa została zaimplementowana w wybranych przez autora algorytmach.

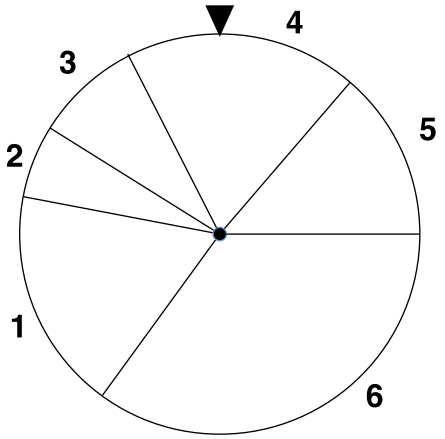


Rys. 4.3: Przykład rozegranego turnieju o rozmiarze 2.
1.Wybór losowych osobników; 2.Turniej pomiędzy wybranymi osobnikami.

Kolejną popularną metody selekcji jest metoda proporcjonalna[4]. W metodzie tej wprowadzone zostaje pojęcie puli rozrodczej. Jest to pula spośród której wybierane są osobniki wykorzystywane później do krzyżowania. Wypełnienie puli rozrodczej kopiami osobników wykonywane jest procesie selekcji. Liczba kopii danego osobnika w puli jest proporcjonalna do jego wartości dopasowania. Jeśli średnia wartość dopasowania dla całej populacji wynosi f_{avg}, dany osobnik s_i będzie posiadał f_i/f_{avg} kopii w puli rozrodczej. Metoda proporcjonalna często nazywana jest również metodą ruletki, ponieważ właśnie za pomocą koła ruletki można zobrazować sobie jej działanie. Wspomniane koło jest podzielone na N części, gdzie N jest wielkością populacji. Każdy wycinek koła odpowiada jednemu osobnikowi z populacji. Wielkość danego wycinka jest proporcjonalna do wartości dopasowania osobnika. Wynika z tego, że wycinki powiązane z lepszymi osobnikami będą zajmować większą część koła. Jak można zauważyć na rysunku (4.4) osobnik o numerze 6 jest najlepszy na tle całej populacji, natomiast osobnik o numerze 2 jest osobnikiem najgorszym. Wybór osobników do puli rozrodczej odbywa się poprzez kręcenie kołem. Czynność ta powtarzana jest N razy. Po zatrzymaniu wybierany jest osobnik na którego wycinku zatrzymał się wskaźnik. Osobniki z najlepszą wartością dopasowania mają największe szanse na trafienie do puli rozrodczej.

Wadami metody proporcjonalnej jest duża złożoność obliczeniowa oraz problemy ze skalowaniem [4]. Załóżmy, że jeden osobnik jest znacząco lepszy od reszty. Wtedy powiązany z nim wycinek będzie zajmował niemal całe koło, a co za tym idzie prawdopodobieństwo wyboru takiego osobnika będzie zbliżone do 1. W konsekwencji pula rozrodcza będzie w większości zapełniona kopiami jednego osobnika, co prowadzi do

braku zróżnicowania. Z drugiej strony jeśli wszystkie osobniki mają bardzo zbliżone wartości dopasowania, każdy z nich ma również podobne prawdopodobieństwo wyboru do puli. Prowadzi to do sytuacji, kiedy każdy osobnik ma dokładnie jedną kopię w puli rozrodczej. Taki proces selekcji nie wnosi żadnej wartości i równie dobrze mógłby zostać pominięty.



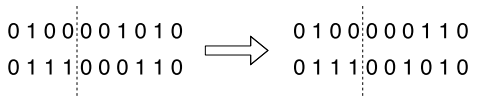
Rys. 4.4: Przykładowa wizualizacja selekcji proporcjonalnej

Opisanych powyżej problemów ze skalowaniem można uniknąć korzystając z metody rankingowej [4]. Polega ona na posortowaniu osobników z populacji ze względu na ich wartość dopasowania. W ten sposób najgorszemu osobnikowi zostaje przypisane pierwsze miejsce w tworzonym rankingu. Osobnik najlepszy znajdzie się wtedy na N-tym miejscu. Po takim przygotowaniu na posortowanej populacji zostaje użyty operator selekcji proporcjonalnej. Na tym etapie przygotowanie koła ruletki nie opiera się już na wartości dopasowania poszczególnych osobników, lecz na ich pozycji w rankingu. Pozwala to na pozbycie się opisanych wcześniej problemów.

Krzyżowanie

Po wyborze osobników mających zostać rodzicami następuje czas na skorzystanie z operatora krzyżowania. Tutaj należy zwrócić uwagę na to, że opisany wcześniej proces reprodukcji nie tworzy żadnych nowych osobników. Dzieje się to dopiero w trakcie procesu krzyżowania. Jak wskazuje nazwa, krzyżowanie polega na łączeniu dwóch osobników w celu utworzenia ich potomstwa [11]. Dzieje się to na poziomie zdefiniowanych wcześniej chromosomów. Najprostszą i najlepiej ilustrującą ideę krzyżowania metodą jest metoda jednopunktowa. Polega ona na losowym wybraniu punktu względem którego nastąpi podział chromosomu na dwie części. Potomstwo powstaje na skutek wymiany pomiędzy rodzicami części chromosomu znajdującej się po prawej stronie od wybranego punktu podziału. Proces ten został przedstawiony na rysunku (4.5). Po lewej stronie znajdują się chromosomy pary rodziców. Punkt podziału został losowo ustalony na $d = 4$, co symbolizuje pionowa przerywana linia. Po prawie stronie znajduje się potomstwo - wynik zastosowania operatora krzyżowania. Jak widać chromosom każdego nowego osobnika powstał na skutek złożenia chromosomów rodziców.

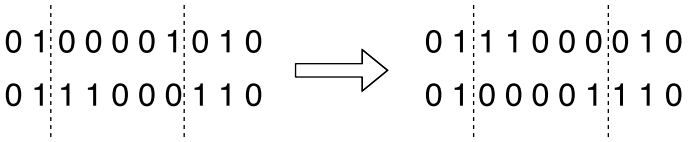
W tym miejscu należy zauważyć, że nowe osobniki utworzone w procesie krzyżowania niekoniecznie muszą być lepsze od swoich rodziców. Może zdarzyć się tak, że potomstwo będzie gorzej przystosowane. Nie należy się tym jednak martwić, ponieważ jak



Rys. 4.5: Przykład krzyżowania jednopunktowego

zostało wyjaśnione wcześniej osobniki o mniejszej wartości przystosowania mają mniejsze szanse na przetrwanie selekcji, a co za tym idzie ich geny nie zostaną przekazane kolejnym pokoleniom. Ponadto jest znacznie bardziej prawdopodobne, że operatory krzyżowania wygenerują rozwiązania lepsze niż generowanie losowe [4]. Dzieje się tak dlatego, że chromosomy wybrane do procesu nie są przypadkowe. Przeszły one selekcje co wskazuje na to, że zawierają w sobie wartościowe kombinacje bitów. Operator krzyżowania użyty na parze rodziców, może wygenerować jedynie l par potomków, gdzie l jest długością chromosomu. Biorąc pod uwagę fakt, że potomstwo jest generowane przy użyciu chromosomów dobrych osobników, istnieje duże prawdopodobieństwo, że utworzone rozwiązania również będą dobre.

Opisany powyżej proces krzyżowania może zostać przeprowadzony również przy użyciu kilku punktów podziału. Na rysunku (4.6) przedstawiona jest selekcja dwupunktowa. Chromosom dzielony jest na trzy części. Potomstwo powstaje na skutek wymiany części środkowej.



Rys. 4.6: Przykład krzyżowania dwupunktowego

Przedstawioną ideę podziału chromosomu można dowolnie rozszerzać. Ekstremum jest tzw. podział jednolity [4]. W takim przypadku potomstwo powstaje poprzez wymianę pojedynczych bitów z określonym prawdopodobieństwem p. Najczęściej przyjmuje się, że $p = 0.5$, czyli istnieję 50% szansy, że dany bit zostanie wymieniony na ten pochodzący z drugiego rodzica.

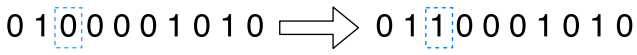
W wybranych algorytmach autor stosuje metodę krzyżowania dwupunktowego, która jest typowym wyborem przy chromosomach reprezentowanych za pomocą ciągów binarnych [8].

W celu zachowania części dobrych osobników z populacji, do procesu krzyżowania dodaje się dodatkową zmienną p_c, oznaczająca prawdopodobieństwo krzyżowania [11]. Oznacza ona jaka część obecnej populacji zostanie użyta do procesu krzyżowania. Gdy $p_c = 1$ nowa populacja w całości zostanie zapełniona potomkami osobników ze starej populacji. W innym przypadku $100(1 - p_c)\%$ osobników ze starej populacji zostanie skopiowanych do nowej. Najczęściej prawdopodobieństwo krzyżowania ustawia się na poziomie 80-90%.

Mutacja

Nowe osobniki powstałe na skutek krzyżowania mogą ulec mutacji. Mutacja jest odpowiedzialna za losową modyfikację chromosomu danego osobnika. Operator mutacji

zapewnia utrzymanie różnorodności populacji na odpowiednim poziomie [11]. Działanie tego operatora opiera się na zmiennej wyrażającej prawdopodobieństwo mutacji: p_m. Operator nie działa na poziomie całego chromosomu, lecz pojedynczych bitów, a jego zastosowanie jest równoznaczne ze zmianą wartości danego bitu. Zostało to zilustrowane na rysunku (4.7).



Rys. 4.7: Przykład mutacji

Podobnie jak w przypadku zastosowania operatora krzyżowania tak i w przypadku mutacji, osobnik będący rezultatem może okazać się gorszy od oryginału. Mimo to, warto zauważyć, że używanie operatora mutacji przy użyciu niskiego prawdopodobieństwa p_m nie jest operacją losową [4]. Powodem jest to, że rozwiązanie generowane jest na podstawie już istniejącego osobnika oraz cały proces może utworzyć jedynie kilka nowych rozwiązań, z dużym prawdopodobieństwem, również dobrych.

Podsumowując, algorytmy genetyczne używają trzech operatorów: selekcji, krzyżowania oraz mutacji. Operator selekcji odpowiedzialny jest za wybór najlepiej przystosowanych osobników. Operator krzyżowania służy do łączenia wybranych wcześniej osobników w celu utworzenia potomstwa. Operator mutacji losowo zmienia część chromosomu danego osobnika w nadziei na znalezienie jeszcze lepszego rozwiązania. Należy tutaj zwrócić uwagę, że żaden z opisanych operatorów nie jest deterministyczny. Każdy z nich oparty jest w znacznym stopniu o losowość, co powoduje, że nie zawsze będą one osiągać zamierzone rezultaty. Nie mniej jednak w kontekście działania algorytmu wymagane jest, aby gorsze osobniki były eliminowane w procesie reprodukcji, na rzecz osobników lepszych, których geny powinny być przekazywane do kolejnych pokoleń.

4.2. Algorytmy wielokryterialne

Główną różnicą pomiędzy klasycznymi metodami optymalizacji, a algorytmami genetycznymi jest fakt, że te drugie nie operują na jednym rozwiązaniu ale na całej populacji rozwiązań. Różnica ta daje ogromną przewagę algorytmom genetycznym jeśli chodzi o rozwiązywanie wielokryterialnych problemów optymalizacyjnych. Jak zostało wspomniane w rozdziale dotyczącym optymalizacji wielokryterialnej, jej celem jest znalezienie jak największej liczby rozwiązań optymalnych w sensie Pareto. Algorytmy genetyczne dają możliwość znalezienia grupy takich rozwiązań jako rezultatu uruchomienia jednej symulacji [4]. Eliminuje to konieczność wielokrotnego używania metod optymalizacji jednokryterialnej w celu znajdowania kolejnych optymalnych rozwiązań. Ponadto używając algorytmów genetycznych można pozbyć się wielu parametrów koniecznych przy używaniu metod klasycznych. Przykładem może być chociażby wektor wag, wykorzystywany do transformacji problemu wielokryterialnego w problem jednokryterialny. Jest on konieczny, ponieważ konkretne ustawienie takiego wektora jest ściśle powiązane z jednym znalezionym rozwiązaniem optymalnym. Jeśli chcemy znaleźć inne rozwiązanie należy zmienić ustawienia wektora oraz ponownie uruchomić

symulację [4]. W przypadku algorytmów genetycznych nie będzie to konieczne, ponieważ jak zostało wspomniane rezultatem zwróconym przez taki algorytm będzie nie pojedyncze rozwiązanie ale cała grupa rozwiązań.

Co więcej operatory genetyczne mogą zostać zmodyfikowane tak, aby faworyzować rozwiązania niezdominowane oraz jednocześnie utrzymywać różnorodność populacji na odpowiednim poziomie. Te dwie cechy połączone z ogólna ideą algorytmów genetycznych prowadzą do znalezienia zróżnicowanej grupy osobników optymalnych w sensie Pareto, czyli rozwiązania wielokryterialnego problemu optymalizacyjnego.

W kolejnych podrozdziałach opisane zostaną dwa algorytmy spełniające powyższe założenia: NSGA-II oraz SPEA. Wyjaśnione zostanie na jakich zasadach działają oba algorytmy oraz w jaki sposób modyfikują poszczególne operatory genetyczne.

4.2.1. NSGA-II

NSGA-II jest drugą wersją zaproponowanego w 2000 roku algorytmu NSGA (Elitist Non-Dominated Sorting Genetic Algorithm). Jego działanie opiera się na dwóch konceptach: elitaryzmie oraz strategii zachowania różnorodności w populacji [3].

Elitaryzm

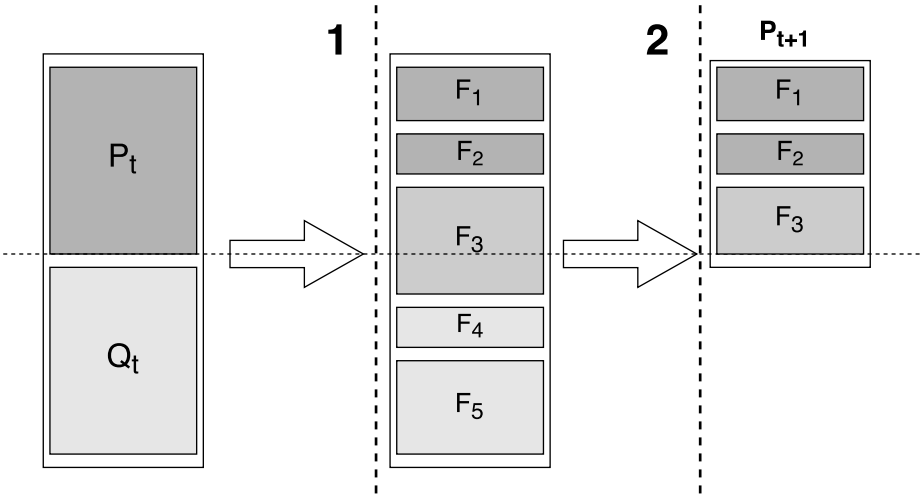
Elitaryzm jest dodatkowym operatorem używanym w algorytmach genetycznych. Jego rolą jest zwiększenie szansy, że najlepsze rozwiązania z danego pokolenia (elita) zostaną bezpośrednio przeniesione do pokolenia następnego. Istnieje wiele sposobów na zdefiniowanie operatora elitaryzmu. W przypadku problemów jednokryterialnych przykładem może być porównywanie potomków utworzonych w procesie krzyżowania z ich rodzicami oraz zachowanie dwóch najlepszych osobników. W globalnego punktu widzenia elitaryzm może być wprowadzony poprzez łączenie rodziców oraz potomków w jedną populację, na której używane są później operatory genetyczne. W ten sposób rodzice również mają szanse na rywalizację ze swoim potomstwem.

Celem elitaryzmu jest zapewnienie, że wartość przystosowania najlepszego osobnika w danym pokoleniu nie jest gorsza od jego odpowiednika z poprzedniego pokolenia. Operator elitaryzmu, że najlepsze rozwiązania znalezione na początku symulacji nie zostaną zgubione. Ponadto ich obecność w kolejnych populacjach zwiększa szanse na utworzenie równie dobrego potomstwa.

Podsumowując, odpowiednio użyty elitaryzm zapewnia lepsze działanie algorytmów genetycznych. Należy jednak zwrócić uwagę na to jaki procent najlepszych osobników można nazwać elitą. Jest to określane przez zmienną α. Jeśli procent ten jest duży, prawdopodobne jest, że w następnych pokoleniach populacja zacznie skupiać się wokół najlepszych osobników, prowadząc do zmniejszenia różnorodności. Jeśli procent będzie za mały wpływ elitaryzmu może zostać nadmiernie zredukowany nie wprowadzając żadnego pozytywnego wpływu. Wartości pomiędzy $\alpha = 1$, a $\alpha = 0.1N$, gdzie N jest wielkością populacji, są najbardziej popularne [4].

W NSGA-II początkowym krokiem jest połączenie populacji rodziców P_t z populacją potomstwa Q_t [3]. Wskutek tego powstaje grupa R_t o rozmiarze $2N$, gdzie N jest rozmiarem populacji. Jest to grupa, na której będzie pracował algorytm. Następnym krokiem jest poddanie R_t sortowaniu rozwiązań niezdominowanych, które zostało opisane w rozdziale **??**. Następnie tworzona zostaje nowa populacja P_{t+1} o rozmiarze N.

Dzieje się to poprzez wypełnianie nowej populacji osobnikami należącymi do danego frontu grupy R_t, zaczynając od frontu najlepszego, czyli pierwszego. Proces ten jest kontynuowany aż do momentu kiedy w nowej populacji nie ma już miejsca na kolejne osobniki. Warto zwrócić uwagę, że osobniki przenoszone są z grupy o rozmiarze $2N$ do grupy o rozmiarze N. Przez to nie wszystkie fronty mogą zostać przeniesione. Fronty, które nie mieszczą się w nowej populacji są po prostu usuwane. Inaczej sytuacja wygląda, gdy dany front może zmieścić się tylko częściowo. Zostało to przedstawione na rysunku (4.8). Wtedy na danym froncie stosowane jest sortowanie oparte o lokalną wartość zagęszczenia, która zdefiniowana zostanie później. Następnie do nowej populacji zostaje przeniesionych n brakujących osobników, które podczas sortowania zostały uznane za najlepsze.



Rys. 4.8: Uproszczony schemat działania algorytmu NSGA-II.
1.Sortowanie rozwiązań niezdominowanych.
2.Sortowanie w oparciu o lokalna wartość zagęszczenia.

Na nowej populacji P_{t+1} stosowany jest następnie zmodyfikowany operator selekcji [3]. Użycie takie operatora zakłada, że każdy osobnik charakteryzuje się dwoma cechami:

1. Numerem frontu, na którym się znajduje.
2. Lokalną wartością zagęszczenia, która opisana zostanie później.

Bazując na tych dwóch cechach, osobnik i jest lepszy od osobnika j, jeśli:

1. Osobnik i znajduje się na lepszym froncie niż osobnik. j
2. Osobnik i znajduje się na tym samym froncie co osobnik j ale charakteryzuje się mniejszą lokalną wartością zagęszczenia.

Pierwszy warunek jest odzwierciedleniem konceptu dominacji. Osobnik leżący na lepszym froncie dominuje osobnika znajdującego się na froncie gorszym. Drugi warunek został wprowadzony, aby rozstrzygać, który z osobników jest bardziej wartościowy jeśli oba leżą na tym samym froncie. Jak zostało wyjaśnione wcześniej rozwiązując wielokry-

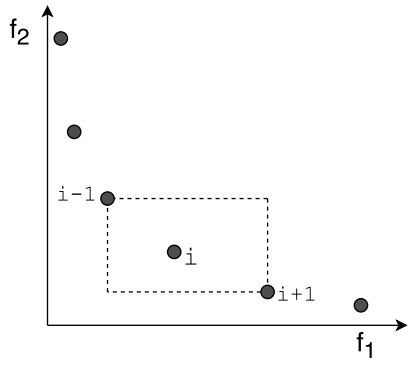
terialne problemy optymalizacyjne należy skupić się nie tylko na jakości rozwiązań ale również ich różnorodności. Różnorodność danego rozwiązania na tle całości populacji jest definiowana za pomocą metryki określającej zagęszczenie opisanej w [4].

Polega ona na obliczeniu średniego dystansu pomiędzy dwoma najbliższymi sąsiadami danego osobnika, znajdującymi się po jego przeciwnych stronach. Zostało to zilustrowane na rysunku (4.9). Takie obliczenia są przeprowadzane m razy dla każdego osobnika, gdzie m jest liczbą zdefiniowanych kryteriów. Następnie wyniki otrzymane dla każdego kryterium są agregowane do jednej wartości d_i. Generalnie algorytm obliczający zagęszczenie populacji można przedstawić następująco:

1. Dla każdego osobnika z populacji przypisz $d_i = 0$ oraz zdefiniuj l jako liczbę osobników.
2. Dla każdego kryterium m, posortuj populację względem wartości funkcji oceny danego kryterium f_m, tworząc wektor I^m
3. Pierwszemu i ostatniemu osobnikowi z wektora I^m przypisz dużą wartość zagęszczenia: $d_{I_1^m} = d_{I_l^m} = \infty$. Reszcie osobników przypisz wartość zagęszczenia korzystając ze wzoru:

$$d_{I_j^m} = d_{I_j^m} + \frac{f_m^{(I_{j+1}^m)} - f_m^{(I_{j-1}^m)}}{f_m^{max} - f_m^{min}} \tag{4.2}$$

I_j oznacza osobnika znajdującego się na j-ej pozycji w posortowanym wektorze I^m. Wynika z tego, że I_1 oraz I_l są osobnikami charakteryzującymi się kolejno najgorszą oraz najlepszą wartością funkcji oceny w porównaniu do całości populacji. Prawa strona równania (4.2) jest różnicą wartości funkcji oceny dwóch najbliższych sąsiadów osobnika I_j. Jak łatwo zauważyć różnica ta jest połową obwodu prostokąta przedstawionego na rysunku (4.9). Dodatkowo wyliczony dystans jest dzielony przez różnicę pomiędzy największą (f_m^{max}), a najmniejszą (f_m^{min}) wartością przyjmowaną przez funkcję oceny f_m.



Rys. 4.9: Reprezentacja najbliższego sąsiedztwa punktu i

Podsumowując schemat działania algorytmu NSGA-II wygląda następująco:

1. Połącz populację rodziców P_t z populacją potomków Q_t tworząc grupę R_t.
2. Na R_t zastosuj sortowanie rozwiązań niezdominowanych w celu zdefiniowania frontów rozwiązań.

3. Utwórz nową populację P_{t+1} za pomocą opisanej wcześniej procedury.
4. Utwórz nową populację potomków Q_{t+1} za pomocą zmodyfikowanego operatora selekcji oraz standardowych operatorów krzyżowania oraz mutacji.

W opisanym powyżej algorytmie NSGA-II różnorodność znalezionych rozwiązań niezdominowanych jest zapewniona poprzez zastosowanie zmodyfikowanego operatora selekcji. Osobniki rywalizują ze sobą opierając swoją wartość o front, na którym się znajdują oraz lokalną wartość zagęszczenia. Wartość zagęszczenie staje się szczególnie ważna w późniejszych momentach symulacji, kiedy większość osobników znajduje się już na najlepszym froncie. Biorąc pod uwagę fakt, że algorytm operuje na grupie o rozmiarze $2N$ istnieje duża szansa, że liczba osobników z pierwszego frontu jest większa niż N, a więc w celu wybrania najlepszych osobników wzięta zostanie pod uwagę wartość zagęszczenia. W konsekwencji rezultatem będzie zbiór rozwiązań niezdominowanych, cechujący się wysokim stopniem różnorodności [3].

4.2.2. SPEA

SPEA (Strength Pareto Evolutionary Algorithm) jest algorytmem zaproponowanym w 1998 roku [18]. Koncept elitaryzmu jest tutaj wprowadzony w postaci zewnętrznej populacji $\overline{P}$. W populacji tej przechowywane są wszystkie niezdominowane rozwiązania znalezione w trakcie działania algorytmu. SPEA oprócz przechowywania najlepszych rozwiązań wykorzystuje je także w procesie reprodukcji, co ma znaczący wpływ na jakość generowanych potomków.

Pierwszym krokiem algorytmu jest utworzenie losowej populacji P_0 oraz pustej zewnętrznej populacji $\overline{P_0}$ o ustalonym rozmiarze, nie większym niż N, gdzie N jest wielkością populacji P_0. Na początku każdej iteracji t, na populacji P_t stosowane jest sortowanie rozwiązań niezdominowanych, w celu znalezienia najlepszego frontu. Osobniki znajdujące się na tym froncie są dodawane do populacji zewnętrznej. Następnie z populacji zewnętrznej usuwane są osobniki zdominowane przez nowo dodane rozwiązania. W ten sposób w populacji zewnętrznej zawsze będą znajdować się jedynie osobniki najlepsze. Łatwo uświadomić sobie, że wraz z działaniem algorytmu wielkość zewnętrznej populacji będzie rosnąć, aż do osiągnięcia zdefiniowanego wcześniej rozmiaru. Gdy ten zostanie osiągnięty pojawia się problem co robić z elitami znajdowanymi w kolejnych pokoleniach, skoro w populacji zewnętrznej nie ma już dla nich miejsca. Rozwiązaniem jest posłużenie jest wartością zagęszczenia, opisaną w poprzednim podrozdziale. Do populacji zewnętrznej trafiają osobniki charakteryzujące się największą różnorodnością na tle populacji. Jak zostało zasugerowane w [4] wartość zagęszczenia może zostać zdefiniowana przy użyciu metryki wykorzystywanej w algorytmie NSGA-II.

Kiedy populacja zewnętrzna zostanie zaktualizowana, następuje proces ewaluacji. Proces ten zostaje najpierw wykonany na populacji zewnętrznej. Dla każdego osobnika i przypisywana zostaje wartość, zdefiniowana jako *siła* S_i. Siła danego osobnika z zewnętrznej populacji wyrażona jest wzorem:

$$S_i = \frac{n_i}{N+1} \tag{4.3}$$

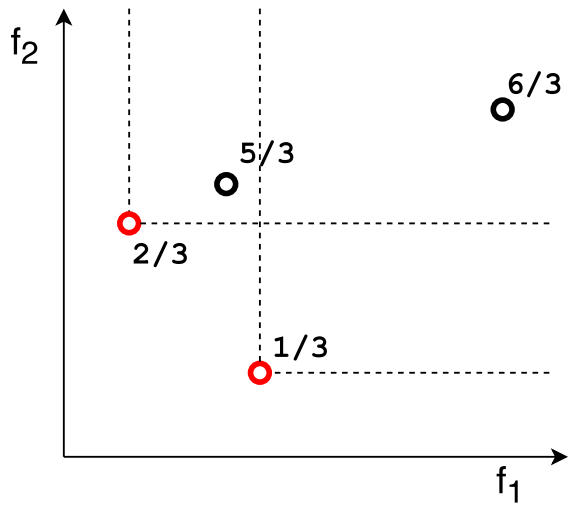
, gdzie n_i określa liczbę osobników populacji, która jest zdominowana przez osobnika i. Jak łatwo zauważyć, osobnik dominujący więcej rozwiązań będzie charakteryzował

się wyższą wartością funkcji S_i.

Osobniki z populacji P_i ocenianie są przy użyciu następującej funkcji:

$$F_j = 1 + \sum_{i \in \overline{P_t} \wedge i \preceq j} S_i \tag{4.4}$$

Liczba 1 zostaje dodana to po to, aby zapewnić, że wartość oceny każdego rozwiązania będzie większa od siły osobników znajdujących się w populacji zewnętrznej. Ponadto dodawana jest suma siły każdego osobnika, przez którego danego rozwiązanie j jest słabo zdominowane. W tym miejscu warto zauważyć, że im mniejsza jest wartość funkcji oceny F_j, tym lepszy jest dany osobnik. Przykład procesu ewaluacji został przedstawiony na rysunku (4.10). Osobniki zaznaczone na czerwone należą do populacji zewnętrznej. Jak łatwo zauważyć w procesie reprodukcji faworyzowane będą osobniki charakteryzujące się mniejszą wartością funkcji oceny, a więc te z populacji zewnętrznej.



Rys. 4.10: Przypisanie wartości funkcji oceny

Po zakończeniu procesu ewaluacji, reszta procesu przebiega według standardowego schematu. Należy jednak pamiętać, że wszystkie operatory genetyczne działają zarówno na populacji P_t jak i populacji zewnętrznej $\overline{P_t}$.

Podsumowując schemat działania algorytmu SPEA wygląda następująco:

1. Zastosuj sortowanie niezdominowanych rozwiązań na populacji P_t w celu znalezienia rozwiązań niezdominowanych.
2. Skopiuj znalezione rozwiązania do populacji zewnętrznej $\overline{P_t}$.
3. Z populacji zewnętrznej usuń rozwiązania zdominowane.
4. Jeśli rozmiar populacji zewnętrznej jest większy niż N, zastosuj sortowanie w oparciu o wartość zagęszczenia oraz zredukuj rozmiar populacji zewnętrznej do N pozbywając się najgorszych osobników.
5. Oceń wszystkie rozwiązania korzystając z opisanych wzorów (4.3, 4.4).

6. Utwórz nową populacje korzystając z operatorów selekcji, krzyżowania oraz mutacji.

Analizując przedstawiony algorytm, łatwo zauważyć, że raz znalezione rozwiązanie optymalne w sensie Pareto nigdy nie przepadnie, ponieważ będzie przechowywane w populacji zewnętrznej. Jedynym sposobem na pozbycie się takiego rozwiązania jest znalezienie kolejnego rozwiązania optymalnego w sensie Pareto, lecz charakteryzującego się lepszą wartością zagęszczenia. Takie podejście zapewnia jakość oraz różnorodność znajdowanych osobników.

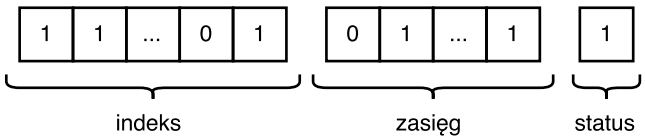
Duże znaczenie dla działania algorytmu ma wielkość zewnętrznej populacji $\overline{N}$. Jeśli ta jest duża (porównywalna do N) nacisk na selekcję rozwiązań z populacji zewnętrznej będzie większy, co może prowadzić do tego, że algorytm nie będzie potrafił wygenerować rozwiązań leżących na prawdziwym froncie Pareto. Gdy $\overline{N}$ będzie zbyt małe, wpływ elitaryzmu będzie niezauważalny. Ponadto może to prowadzić do sytuacji w której wiele rozwiązań z populacji nie będzie zdominowanej przez żadnego osobnika z populacji zewnętrznej. Będzie skutkować to tym, że wartość funkcji oceny dla tych osobników będzie tak sama. Proponowaną wielkością populacji zewnętrznej jest 1/4 wielkości populacji N [4].

4.3. Strojenie

Jak zostało opisane wcześniej działanie algorytmów genetycznych opiera się na wielu parametrach. Parametry te są ze sobą mocno powiązane oraz znacząco wpływają na jakość symulacji. Proces wyboru odpowiednich wartości dla wspomnianych parametrów, nazywa się strojeniem.

4.4. Implementacja

W obu wybranych algorytmach (NSGA-II, SPEA) pojedynczy osobnik reprezentowany jest jako zbiór zraszaczy S. Maksymalna ilość zraszaczy została ustalona na poziomie $ms = 50$. Pojedynczy zraszacz jest reprezentowany jako następujący ciąg bitów:



Rys. 4.11: Reprezentacja bitowa pojedynczego zraszacza

Ciąg binarny podzielony jest na trzy części. Pierwsza z nich odpowiada za indeks, czyli pozycję danego zraszacza. Zraszacz może znajdować się w jednym z kwadratów macierzy M, którego środek znajduje się w środku wielokąta P. Indeks danego kwadratu jest ustalany według wzoru:

$$ind_i = n_i m + m_i \tag{4.5}$$

, gdzie n_i oznacza wierz macierzy M, w którym znajduję się kwadrat i; m jest wartością wyrażającą szerokość macierzy; m_i oznacza kolumnę macierzy M.

Długość części reprezentującej położenie danego zraszacza wynosi:

$$L_p = \left\lceil \log_2 \ ind_p \right\rceil + 1 \tag{4.6}$$

, gdzie ind_p jest liczbą reprezentującą największy indeks należący do P.

Kolejną częścią reprezentacji binarnej jest część odpowiadająca za zasięg danego zraszacza. Wymaga ona zużycia L_r bitów, gdzie L_r jest definiowane jako:

$$L_r = \left\lceil \log_2 \ r_{max} \right\rceil + 1 \tag{4.7}$$

Wartość r_{max} jest wartością oznaczającą maksymalny zasięg zraszacza. Jest ona definiowana na podstawie zbudowanej bazy danych i określana poprzez wybranie największej wartości zasięgu spośród wszystkich dostępnych.

Ostatnia, trzecia część reprezentacji binarnej to jeden bit mówiący o statusie danego zraszacza. $b = 1$ oznacza, że dany zraszacza został wybrany do rozwiązania oraz będzie uwzględniany w obliczeniach. W przeciwnym wypadku $b = 0$.

Podsumowując długość chromosomu każdego osobnika z populacji wynosi:

$$L = ms(L_p + L_r + 1) \tag{4.8}$$

W tym miejscu warto zauważyć, że długość chromosomu jest zawsze stała. Nieaktywne zraszacze ($b = 0$) nie są w żaden sposób usuwane z reprezentacji. Takie podejście niweluje problemy pojawiające się przy użyciu reprezentacji chromosomów o zmiennej długości. Problemy te są najczęściej powiązane z trudnościami w implementacji operatorów krzyżowania oraz selekcji.

Oba algorytmy zaimplementowane zostały z wykorzystaniem selekcji turniejowej o rozmiarze turnieju równym 2, krzyżowania dwupunktowego (z losowym wyborem punktów podziału) oraz mutacji binarnej. Wskutek przeprowadzonego procesu strojenia ustalono, że algorytmy najlepiej pracują na następującym zestawie parametrów:

- Prawdopodobieństwo krzyżowania : 0.9.
- Prawdopodobieństwo mutacji : 0.05.
- Wielkość populacji zewnętrznej (SPEA) : 1/4N.

Rozdział 5

Systemy wspomagania decyzji

Rozdział 6

Rozwiązanie problemu

6.1. System wspomagania decyzji

System wspomagania decyzji został zaimplementowany w formie aplikacji webowej. Została ona wykonana przy użyciu następujących technologii:

- Python (PyPy) [2].
- Framework Django [1].
- HTML 5.
- JavaScript.
- CSS.

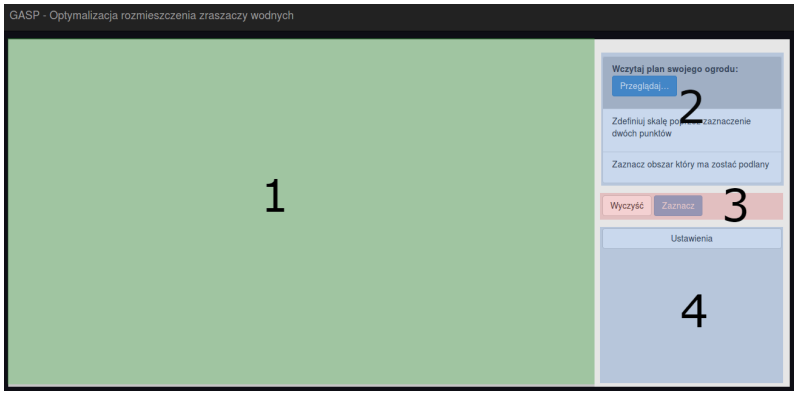
Użytkownik systemu poprzez interfejs graficzny wprowadza dane oraz wybiera opcje, która później przesyłane są na serwer. Na serwerze dane są przekształcane do formatu zrozumiałego przez zaimplementowane algorytmy genetyczne. Po zakończeniu obliczeń rezultaty zwracane są użytkownikowi w czytelnej oraz przejrzystej formie.

W następnej części podrozdziału przedstawiony zostanie przykładowy scenariusz interakcji użytkownika z system, obrazujący jego działanie.

Interakcja z użytkownikiem

Po uruchomieniu aplikacji użytkownikowi ukazuje się ekran przedstawiony na rysunku (6.1). Ekran dzieli się na cztery zasadnicze części. Zostały one oznaczone na rysunku odpowiadającymi im liczbami. Część pierwsza jest częścią główną aplikacji. To właśnie do tego obszaru ładowany jest plan ogrodu, na którym użytkownik definiuje teren mający zostać nawodniony oraz określa miejsce źródła wody. To również na tym ekranie prezentowany są wszystkie rezultaty. Część druga jest częścią informacyjną. Tutaj wyświetlane są instrukcje podpowiadające użytkownikowi co w danym momencie powinien zrobić. Trzecia część jest kontenerem na przyciski, za pomocą których użytkownik zatwierdza swoje wybory, anuluje je lub przechodzi do kolejnych etapów. Część czwarta jest przeznaczona na dodatkowe opcje mające wpływ na działanie algorytmów znajdujących się po stronie serwera.

Pierwszym krokiem w pracy z aplikacją jest wczytanie planu ogrodu. Przykładowy



Rys. 6.1: Ekran startowy aplikacji

wczytany plan został przedstawiony na rysunku (6.2). Jak można zauważyć zmienił się status panelu informacyjnego widocznego po prawej stronie.

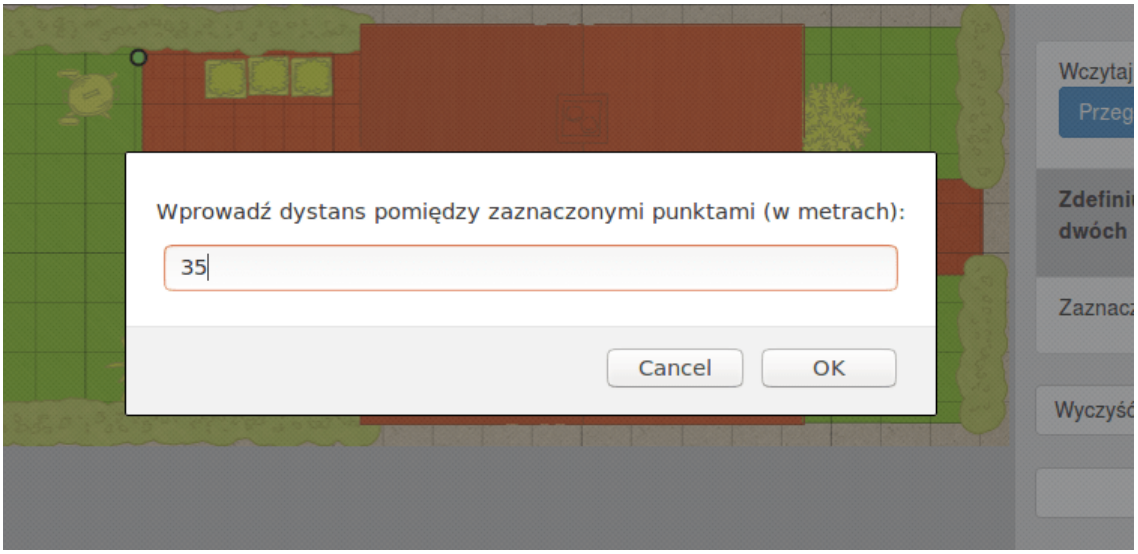
Rys. 6.2: Wczytanie planu ogrodu

Mając wczytany plan ogrodu należy zdefiniować skalę rysunku. Jest to wykonywane poprzez zaznaczenie na wczytanym planie dwóch punktów. Po wykonaniu tej czynności na ekranie pojawia się okno, w którym można zdefiniować dystans pomiędzy zaznaczonymi punktami (6.3).

Następnym krokiem jest zaznaczenie na załadowanym planie obszaru, który ma zostać nawodniony. Dzieje się to poprzez określenie, za pomocą lewego przycisku myszy, wierzchołków wielokąta mającego tworzyć wspomniany obszar (6.4). Liczba wierzchołków jest nieograniczona. Po wybraniu obszaru należy rozwinąć panel "Ustawienia". Jak widać w obecnej wersji aplikacji użytkownik może jedynie zdecydować się na to czy chce ograniczyć nawadnianie obszaru poza zaznaczonymi granicami. Aby przejść dalej należy wcisnąć przycisk "Dalej". Spowoduje to przesłanie obecnych ustawień do serwera oraz uruchomienie obliczeń.

Po zakończeniu działania algorytmu użytkownik zostanie automatycznie przeniesiony na stronę prezentującą listę wyników (6.5). Użytkownik może tutaj wybrać rozwiązanie, które go interesuje oraz podejrzeć je wciskając przycisk "Zobacz".

Po wybraniu konkretnego rozwiązania użytkownik zostanie przeniesiony na stronę prezentującą rozmieszczone zraszacze (6.6). Są one zaprezentowane w postaci przezro-



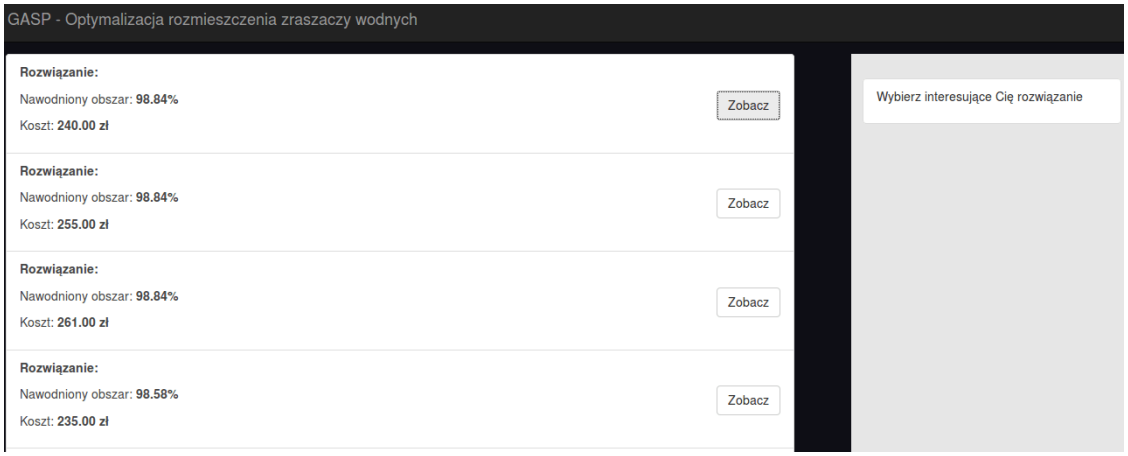
Rys. 6.3: Definiowanie skali wczytanego planu ogrodu

Rys. 6.4: Zaznaczenie obszaru mającego zostać nawodniony

czystych niebieskich okręgów. Po najechaniu na każdy z nich pojawia się informacja o modelu konkretnego zraszacza, zasięgu oraz cenie. Na tym samym widoku użytkownik może określić, w którym miejscu na planie znajduje się źródło wody. Po wciśnięciu przycisku "Zakończ" na planie ogrodu dodatkowo pojawi się schemat poprowadzenia rur (6.7)

W tym momencie użytkownik ma możliwość przejścia do widoku "Lista zakupów" (6.8). W tym miejscu warto zauważyć, że lista, znajdująca się po prawej stronie ekranu i pełniąca do tej pory rolę informacyjną, zmieniła się w panel nawigacyjny. Za jej pomocą użytkownik może w łatwy sposób przemieszać się pomiędzy widokami, będącymi podsumowaniem wybranego rozwiązania. Użytkownik ma do wyboru trzy widoki:

1. Widok rozmieszczonych zraszaczy.
2. Widok rozmieszczonych zraszaczy oraz schematu poprowadzenia rur.



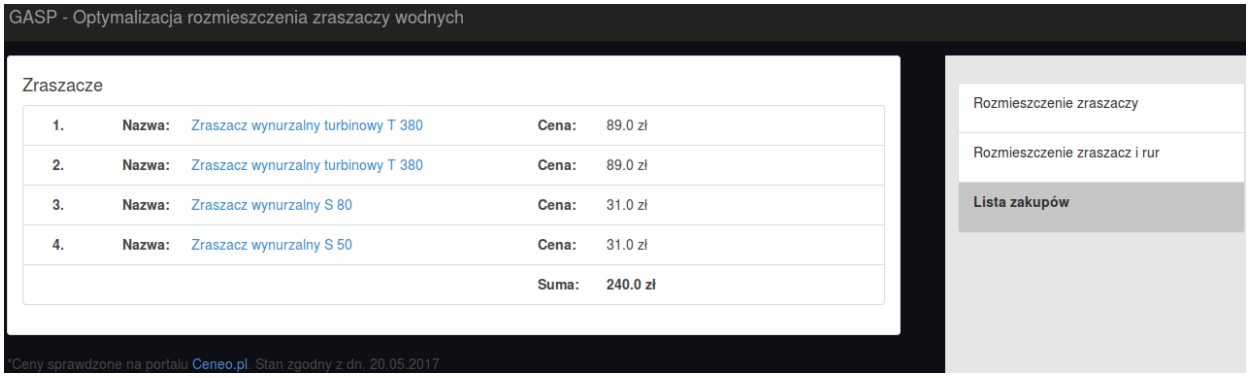
Rys. 6.5: Lista rezultatów

Rys. 6.6: Rozmieszczenie zraszaczy

Rys. 6.7: Schemat poprowadzenia rur

3. Widok listy zakupów.

Wybierając widok listy zakupów użytkownik zobaczy listę rozmieszczonych zraszaczy. Nazwa każdego zraszacza jest jednocześnie odnośnikiem do opisu konkretnego modelu znajdującego się na stronie producenta. Oprócz tego do każdej pozycji na liście przypisana jest cena. Pod listą znajduje się kwota będąca sumą całych zakupów.



Rys. 6.8: Lista zakupów

6.2. Porównanie algorytmów genetycznych

6.2.1. Plan badań

Złożoność oceny algorytmów służących do optymalizacji wielokryterialnej jest znacznie większa niż tej dla algorytmów rozwiązujących problemy obejmujące jedno kryterium [17]. Wynika to z samej definicji rozwiązania problemu wielokryterialnego, która wyjaśniona została wcześniej. Oceniając rozwiązanie takiego problemu należy wziąć pod uwagę następujące składowe:

- Dystans znalezionego niezdominowanego zbioru rozwiązań od prawdziwego zbioru Pareto.
- Równomierny rozkład znalezionych rozwiązań.
- Długość przedziału pokrytego przez znalezione rozwiązania.
- Czas wykonania algorytmu [7].

Na potrzeby analizy i oceny dwóch zaproponowanych algorytmów wybrane zostały cztery metryki [17][7]:

1. *Współczynnik błędu (ER):* wskazuje procent rozwiązań, które nie należą do prawdziwego zbioru Pareto.

$$ER = \frac{\sum_{i=1}^{n} e_i}{n} \tag{6.1}$$

   Gdzie n jest liczbą znalezionych niezdominowanych rozwiązań; e_i zmienną określającą czy dane rozwiązanie znajduje się w prawdziwym zbiorze Pareto. Jeśli $e_i = 0$ rozwiązanie znajduje się w prawdziwym zbiorze Pareto, w przeciwnym razie $e_i = 1$. $ER = 0$ oznacza idealne rozwiązanie.

2. *Równomierny rozkład (SP):* metoda mierząca wariancję dystansu sąsiadujących rozwiązań znajdujących się na znalezionym froncie Pareto.

$$SP = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\overline{d} - d_i)^2} \tag{6.2}$$

   Gdzie n jest liczbą znalezionych niezdominowanych rozwiązań; d_i oznacza dystans Euklidesowy pomiędzy danym rozwiązaniem, a jego najbliższym sąsiadem; $\overline{d}$ jest

Tabela 6.1: Przypadki testowe

Wielkość	Obszar
Mały	1
Średni	2
Duży	3

wartością oznaczającą średni dystans pomiędzy rozwiązaniami. Wartość $SP = 0$ oznacza, że znalezione rozwiązania są rozłożone równomiernie.

3. *Jakość ogólna (DG):* wskazuje jak daleko od prawdziwego zbioru Pareto znajduje się zbiór znalezionych niezdominowanych rozwiązań. Miara zdefiniowana w następujący sposób:

$$DG = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{6.3}$$

Gdzie n oznacza liczbę znalezionych niezdominowanych rozwiązań; d_i jest dystansem Euklidesowym pomiędzy danym rozwiązaniem, a najbliższym rozwiązaniem należącym do prawdziwego frontu Pareto. Wartość $DG = 0$ oznacza, że wszystkie znalezione rozwiązania leżą na prawdziwym froncie Pareto.

4. *Długość frontu (FE):* metoda wskazująca jak duży obszar jest pokryty przez znalezione niezdominowane rozwiązania.

$$FE = \sum_{k=1}^{K} \max_{i,j!=i} \sqrt{(f_i^k - f_j^k)^2} \tag{6.4}$$

Gdzie K oznacza liczbę funkcji celu; f_i^k jest wartością i-tego rozwiązania z perspektywy kryterium k.

Aby skorzystać z powyższych metryk konieczne jest znalezienie prawdziwego frontu Pareto. Na potrzeby badań został on wygenerowany poprzez wybranie niezdominowanych rozwiązań ze skumulowanej puli rozwiązań powstałej na skutek uruchomienia symulacji 10 razy dla każdego algorytmu. Rezultaty każdej symulacji były dodawane do puli po czym zredukowane do rozwiązań niezdominowanych.
Dla badań przygotowanych zostało 6 przypadków testowych. Zostały one podzielone ze względu na wielkość obszaru mającego zostać nawodniony oraz tego czy algorytm ma minimalizować nawadnianie poza zaznaczonym obszarem. Kształt wybranych obszarów został dobrany tak, aby jak najlepiej wizualizować wpływ ograniczeń na działanie algorytmów.

6.2.2. Rezultaty badań

6.2.3. Podsumowanie badań

Rozdział 7

Podsumowanie

Definicja 1
Definicja - *pierwsza*

Dodatek A

Appendix 1

Spis rysunków

Spis wzorów

Spis algorytmów

Bibliografia

[1] Deb K., Agrawal S., Pratap A., Meyarivan T. *A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II.* Springer Berlin Heidelberg, 2000.

[2] Deb K., Kalyanmoy D. *Multi-Objective Optimization Using Evolutionary Algorithms.* John Wiley & Sons, Inc., 2001.

[3] Django. Django: The web framework for perfectionists with deadlines, 2017. Dostęp: 2017-05-10.

[4] Ehrgott M. *Multicriteria Optimization.* Springer-Verlag New York, Inc., 2005.

[5] Goldberg D. E. Real-coded genetic algorithms, virtual alphabets, and blocking. *Complex Systems*, 5, 1990.

[6] Hamdy M., Nguyen A.-T., Hensen J. L. A performance comparison of multi-objective optimization algorithms for solving nearly-zero-energy-building design problems. *Energy and Buildings*, 121, 2016.

[7] Jia J., Chen J., Chang G., Wen Y., Song J. Multi-objective optimization for coverage control in wireless sensor network with adjustable sensing radius. *Computers & Mathematics with Applications*, 57(11), 2009.

[8] Kung H. T., Luccio F., Preparata F. P. On finding the maxima of a set of vectors. *J. ACM*, 22(4), 1975.

[9] Miller B. L., Miller B. L., Goldberg D. E., Goldberg D. E. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, 9, 1995.

[10] Mitchell M. *An Introduction to Genetic Algorithms.* MIT Press, Cambridge, MA, USA, 1998.

[11] Ofobi E. F. Minimum connection of pipes for water distribution network in Agric Nsima. Kwame Nkrumah University of Science and Technology, 2012.

[12] Pypy. Pypy - welcome to pypy, 2017. Dostęp: 2017-05-10.

[13] Reeves C. R. Using genetic algorithms with small populations. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, 1993.

[14] Wikipedia. Mathematical optimization — wikipedia, the free encyclopedia, 2017. Dostęp: 2017-06-02.

[15] Wikipedia. Optymalizacja — wikipedia, wolna encyklopedia, 2017. Dostęp: 2017-06-02.

[16] Wikipedia. Pareto efficiency — wikipedia, the free encyclopedia, 2017. Dostęp: 2017-06-03.

[17] Zitzler E., Deb K., Thiele L. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2), 2000.

[18] Zitzler E., Thiele L. An evolutionary algorithm for multiobjective optimization: The strength Pareto approach. Technical Report 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), 1998.