Autoreferat

1. Imię i nazwisko

Piotr Wnuk-Lipiński

2. Wykształcenie

2002 – 2004 European Doctoral College, Strasbourg, Francja, absolwent (dyplom ukończenia)

2001 – 2004 Université Louis Pasteur, Strasbourg, Francja, docteur d'informatique

2001 – 2004 Uniwersytet Wrocławski, doktor nauk matematycznych w zakresie informatyki (praca doktorska wyróżniona przez Radę Wydziału Matematyki i Informatyki)

(praca doktorska *Evolutionary Data-Mining Methods in Discovering Stock Market Expertise from Financial Time Series* napisana pod kierunkiem promotora prof. dr. hab. Jerzego Korczaka z Université Louis Pasteur, Strasbourg, Francja i kopromotor dr hab. Anny Bartkowiak z Uniwersytetu Wrocławskiego, w ramach umowy co-tutelle między ULP i UWr, wspierana przez stypendium Rządu Francuskiego)

1996 – 2001 Uniwersytet Wrocławski, Wydział Matematyki i Informatyki, Instytut Informatyki, magister informatyki, specjalność zastosowania informatyki

(praca magisterska *Portfolio Optimization using Evolution Strategies* napisana pod kierunkiem promotora prof. dr. hab. Jerzego Korczaka z Université Louis Pasteur, Strasbourg, Francja, wspierana przez stypendium Rządu Francuskiego)

1996 – 2001 Uniwersytet Wrocławski, Wydział Matematyki i Informatyki, Instytut Matematyki, magister matematyki, specjalność matematyka teoretyczna

(praca magisterska *Zbieżne ciągi kombinacji wypukłych elementów ciągów całkowalnych* napisana pod kierunkiem promotora prof. dr. hab. Kazimierza Musiała)

1992 – 1996 XIV Liceum Ogólnokształcące we Wrocławiu (świadectwo dojrzałości)

PL

3. Zatrudnienie

2004 – Uniwersytet Wrocławski, Wydział Matematyki i Informatyki, Instytut Informatyki

(2004 – 2005, asystent; 2005 – obecnie, adiunkt; 2014 – obecnie, kierownik Pracowni Inteligencji Obliczeniowej)

2004 – 2006 Université Louis Pasteur, UFR Mathématique Informatique, Département d'Informatique, Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection, CNRS, Strasbourg, Francja

(attaché temporaire d'enseignement et de recherche)

3.1. Staże, szkolenia, pobyty

2011 – 2012 Natural Computing Research and Applications Group, Complex and Adaptive Systems Laboratory, University College Dublin, Irlandia

(pobyt naukowy związany z badaniami nad szeregami czasowymi ultrawysokiej częstotliwości, współpraca z prof. A. Brabazonem, 3 miesiące)

2003 Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana Champaign, Urbana-Champaign, USA

(pobyt naukowy w ramach współpracy CNRS, Francja i UIUC, USA związany z badaniami nad algorytmami genetycznymi, współpraca z prof. D. Goldbergiem, 2 tygodnie)

2001 Equipe de Recherche Technologique en Informatique, CNRS, Université Louis Pasteur, Illkirch, Francja

(pobyt związany z pracą w projekcie COMMUNIGRAM prowadzonym przez ERTI i ERMITE)

2000 – 2001 Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection, CNRS, Université Louis Pasteur, Illkirch, Francja

(pobyt związany z pracą w projekcie iBE-RT prowadzonym przez LSIIT)

3.2. Nagrody, granty, stypendia

2010 – 2013 Stypendium MNiSW dla wybitnych młodych uczonych

2003 Grant badawczy European Doctoral College, Strasbourg, Francja

2001 – 2004 Stypendium Rządu Francuskiego na badania związane z pracą doktorską

2000 – 2001 Stypendium Rządu Francuskiego na badania związane z pracą magisterską

2000 – 2001 Stypendium MEN

1996 Wyróżnienie w XLVII Olimpiadzie Matematycznej

PL

4. Wskazanie osiągnięcia naukowego

Rozprawę habilitacyjną stanowi jednotematyczny cykl prac zatytułowany

algorytmy ewolucyjne dla wysokowymiarowych problemów optymalizacji i ich zastosowania

w skład którego wchodzą następujące prace:

[HAB1] Lipinski, P., ECGA vs. BOA in Discovering Stock Market Trading Experts, [in] Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO 2007, ACM, 2007, pp.531-538.

[HAB2] Lipinski, P., Knowledge Patterns in Evolutionary Decision Support Systems for Financial Time Series Analysis, [in] Applications of Evolutionary Computing, EvoWorkshops 2009, Lecture Notes in Computer Science, vol. 5484, Springer, 2009, pp.203-212.

[HAB3] Lipinski, P., Frequent Knowledge Patterns in Evolutionary Decision Support Systems for Financial Time Series Analysis, [in] Natural Computing in Computational Finance, Studies in Computational Intelligence, vol. 293, Springer, 2010, pp.131-145.

[HAB4] Lipinski, P., A Hybrid Evolutionary Algorithm to Quadratic Three-Dimensional Assignment Problem with Local Search for Many-Core Graphics Processors, [in] Intelligent Data Engineering and Automated Learning, IDEAL 2010, Lecture Notes in Computer Science, vol. 6283, Springer, 2010, pp.344–351.

[HAB5] Lipinski, P., Evolution Strategies for Objective Functions with Locally Correlated Variables, [in] Intelligent Data Engineering and Automated Learning, IDEAL 2010, Lecture Notes in Computer Science, vol. 6283, Springer, 2010, pp.352–359.

[HAB6] Lipinski, P., A Stock Market Decision Support System with a Hybrid Evolutionary Algorithm for Many-Core Graphics Processors, [in] Euro-Par 2010 Parallel Processing Workshops, Lecture Notes in Computer Science, vol. 6586, Springer, 2011, pp.455-462.

[HAB7] Lipinski, P., Parallel Evolutionary Algorithms for Stock Market Trading Rule Selection on Many-Core Graphics Processors, [in] Natural Computing in Computational Finance, Studies in Computational Intelligence, vol. 380, Springer, 2012, pp.79-92.

[HAB8] Lipinski, P., Training Complex Decision Support Systems with Differential Evolution Enhanced by Locally Linear Embedding, [in] Applications of Evolutionary Computation, EvoWorkshops 2014, Lecture Notes in Computer Science, vol. 8602, Springer, 2014, pp.125-137.

[HAB9] Lipinski, P., Optimizing Objective Functions with Non-Linearly Correlated Variables Using Evolution Strategies with Kernel-Based Dimensionality Reduction, [in] Hybrid Artificial Intelligence Systems, HAIS 2014, Lecture Notes in Computer Science, vol. 8480, Springer, 2014, pp.342-353.

[HAB10] Lipinski, P., Training Financial Decision Support Systems with Thousands of Decision Rules Using Differential Evolution with Embedded Dimensionality Reduction, [in] Applications of Evolutionary Computation, EvoWorkshops 2015, Lecture Notes in Computer Science, vol. 9028, Springer, 2015, pp.289-301.

Wymienione prace zostały opublikowane w materiałach konferencyjnych międzynarodowych konferencji naukowych poświęconych algorytmom ewolucyjnym i inteligencji obliczeniowej, takich jak

92

GECCO (najważniejsza światowa konferencja poświęcona algorytmom ewolucyjnym), EvoStar - EvoWorkshops (najważniejsza europejska konferencja poświęcona algorytmom ewolucyjnym i ich zastosowaniom, zakwalifikowana w 15% najlepszych z 361 międzynarodowych konferencji w dziedzinie sztucznej inteligencji, według Microsoft Research Conference Ranking, z 4879 cytowaniami i 1181 publikacjami, co daje współczynnik cytowań do publikacji 4.13), Euro-Par (europejska konferencja poświęcona obliczeniom równoległym, zakwalifikowana na 22 miejscu z 203 międzynarodowych konferencji w dziedzinie obliczeń rozproszonych i równoległych, według Microsoft Research Conference Ranking, z 9756 cytowaniami i 2394 publikacjami, co daje współczynnik cytowań do publikacji 4.08), IDEAL (europejska konferencja poświęcona inteligentnemu przetwarzaniu danych, zakwalifikowana na 20 miejscu z 41 międzynarodowych konferencji w dziedzinie eksploracji danych, według Microsoft Research Conference Ranking, z 1789 cytowaniami i 1032 publikacjami, co daje współczynnik cytowań do publikacji 1.73), HAIS (europejska konferencja poświęcona systemom inteligentnym i ich zastosowaniom) oraz w rozdziałach monografii Natural Computation in Computational Finance wydanymi w serii Studies in Computational Intelligence wydawnictwa Springer.

4.0. Wprowadzenie

Cykl prac poświęcony jest algorytmom ewolucyjnym dla wysokowymiarowych problemów optymalizacji i ich zastosowaniom. Wysokowymiarowe problemy optymalizacji coraz częściej występują w praktyce ze względu na coraz większy wolumen i coraz większą wymiarowość przetwarzanych danych. Standardowe algorytmy ewolucyjne często okazują się niewystarczające do rozwiązywania takich problemów, bo wysoka wymiarowość przestrzeni poszukiwań wymaga uruchamiania algorytmu z bardzo dużą populacją i bardzo dużą liczbą iteracji, prowadząc tym samym do bardzo długiego, często przekraczającego rzeczywiste możliwości techniczne, czasu obliczeń.

W cyklu prac przedstawiam trzy podejścia do rozwiązywania wysokowymiarowych problemów optymalizacji:

- Pierwsze podejście, opisane w pracach [HAB1, HAB6, HAB7], oparte jest na algorytmach estymowania rozkładu (ang. Estimation of Distribution Algorithms, EDA) [1], które starają się konstruować model probabilistyczny opisujący rozwiązanie optymalne analizowanego problemu optymalizacji. W pracy [HAB1] opisałem zastosowanie algorytmów Extended Compact Genetic Algorithm (ECGA) [2] i Bayesian Optimization Algorithm (BOA) [3] do rozwiązywania praktycznego problemu optymalizacji związanego z konstrukcją systemu wspomagania decyzji dla finansowych szeregów czasowych, a także zaproponowałem redukcję zachłannych części tych algorytmów konstruujących model probabilistyczny opisujący aktualną populację. Modyfikacje te ułatwiły konstrukcję wysokowymiarowych modeli probabilistycznych i uruchomienie algorytmów na wysokowymiarowej przestrzeni poszukiwań. W pracach [HAB6] i [HAB7] zaproponowałem połączenie prostszego algorytmu estymowania rozkładu, Population-Based Incremental Learning (PBIL) [4], z masowym przeszukiwaniem lokalnym realizowanym równolegle na procesorach graficznych. Zaproponowane podejście z masowym przeszukiwaniem lokalnym okazało się skuteczne w rozwiązywaniu wybranych problemów optymalizacji, mimo bardzo prostych (w porównaniu do ECGA i BOA) modeli probabilistycznych.

- Drugie podejście, opisane w pracach [HAB2, HAB3][1], dotyczy rozwiązywania sekwencji podobnych do siebie problemów optymalizacji, w którym przy rozwiązywaniu kolejnych problemów wy-

[1] praca [HAB3] jest rozszerzoną wersją pracy [HAB2]

korzystuje się wzorce częste powtarzające się w znalezionych wcześniej rozwiązaniach poprzednich problemów. Zaproponowane przeze mnie wykorzystanie wzorców częstych umożliwiło znaczne przyśpieszenie obliczeń poprzez redukcję wymiarowości problemu optymalizacji nie wpływając na jakość znajdowanych rozwiązań.

- Trzecie podejście, opisane w pracach [HAB5, HAB8, HAB9, HAB10], oparte jest na metodach liniowej i nieliniowej redukcji wymiarowości, które stosowane do aktualnej populacji próbują przekształcać przestrzeń poszukiwań do przestrzeni o mniejszej liczbie wymiarów i odpowiednio transformować problem optymalizacji. Prace [HAB5] i [HAB10] przedstawiają podejście oparte na metodzie składowych głównych (ang. Principal Component Analysis, PCA) [5]. Praca [HAB8] przedstawia podejście oparte na Locally Linear Embeddings (LLE) [6], zaś praca [HAB9] przedstawia podejście oparte na Kernel Principal Component Analysis (KPCA) [7]. Wszystkie podejścia okazały się bardzo skuteczne w rozwiązywaniu wysokowymiarowych problemów optymalizacji ze złożonymi zależnościami między współrzędnymi wektora rozwiązania.

Najwięcej miejsca w swoim autoreferacie poświęcam ostatniemu podejściu, bo z perspektywy czasu wydaje się ono najbardziej interesujące i najbardziej uniwersalne.

Ciekawym uzupełnieniem tych podejść jest praca [HAB4], w której proponuję algorytm ewolucyjny z masowym przeszukiwaniem lokalnym (podobnie jak w pracach [HAB6] i [HAB7]) do rozwiązywania problemu Quadratic Three-Dimensional Assignment Problem (Q3AP). W Q3AP przestrzeń poszukiwań jest zbiorem par permutacji ustalonej długości n (a zatem jej moc to $n! \times n!$), a funkcja celu jest na tyle nieregularna, że trudnością jest znalezienie rozwiązań optymalnych już dla n rzędu kilkunastu. Jest więc to przykład problemu optymalizacji, w którym wysoka wymiarowość zaczyna się już dla relatywnie niewielkich n.

Jako przykładu wysokowymiarowego problemu optymalizacji w prowadzonych badaniach używałem praktycznego problemu optymalizacji związanego z konstrukcją systemu wspomagania decyzji dla finansowych szeregów czasowych, opisanego dokładniej w rozdziale 4.1, w którym algorytmy ewolucyjne wykorzystywane były do konstrukcji ekspertów decyzyjnych złożonych z pewnej, zazwyczaj bardzo dużej, liczby ustalonych reguł decyzyjnych. Przestrzeń poszukiwań była więc wysokowymiarowa, najczęściej była to przestrzeń $\Omega = \{0,1\}^n$ lub $\Omega = \mathbb{R}^n$, gdzie n było rzędu kilkuset.

Część eksperymentów obliczeniowych dotyczyło też popularnych testowych problemów optymalizacji używanych często do testowania algorytmów ewolucyjnych i ich wysokowymiarowych rozszerzeń, opisanych dokładniej w rozdziale 4.4.

4.1. Podejście oparte na algorytmach estymowania rozkładu

W moim pierwszym podejściu do wysokowymiarowych problemów optymalizacji, opisanym w pracach [HAB1, HAB6, HAB7], rozpatruję algorytmy estymowania rozkładu (ang. Estimation of Distribution Algorithms, EDA) [1], które starają się konstruować model probabilistyczny opisujący rozwiązanie optymalne analizowanego problemu optymalizacji[2].

[2] w dalszej części autoreferatu *rozwiązaniem* nazywam każde rozwiązanie kandydujące (ang. candidate solution), czyli każdy element przestrzeni poszukiwań, *rozwiązaniem optymalnym* nazywam dokładne rozwiązanie problemu optymalizacji; jeśli przestrzeń poszukiwań jest iloczynem kartezjańskim pewnych przestrzeni, czyli rozwiązanie jest pewną krotką, to *wektorem rozwiązania* nazywam strukturę danych przechowującą rozwiązanie lub opisujący je wektor losowy. *współrzędną wektora rozwiązania* nazywam strukturę danych przechowującą pojedynczy element rozwiązania lub opisującą go zmienną losową

PL

Rozwiązując problem optymalizacji funkcji celu $F : \Omega \rightarrow \mathbb{R}$ określonej na przestrzeni poszukiwań Ω, EDA traktują rozwiązanie optymalne takiego problemu jako realizację pewnej zmiennej losowej X o wartościach w Ω i starają się iteracyjnie estymować rozkład tej zmiennej losowej na podstawie aktualnej populacji w kolejnych iteracjach algorytmu ewolucyjnego. Popularne EDA dotyczą przypadków, w których $\Omega = \{0,1\}^n$ lub $\Omega = \mathbb{R}^n$, a więc X można traktować jako wektor losowy $X = (X_1, X_2, \ldots, X_n)^T$. Proste EDA, takie jak Population-Based Incremental Learning (PBIL) [4], Compact Genetic Algorithm CGA) [8], Univariate Marginal Distribution Algorithm (UMDA) [9], traktują X jako wektor niezależnych zmiennych losowych i wyznaczają brzegowe rozkłady prawdopodobieństwa zmiennych losowych $X_1, X_2, \ldots, X_n$. Bardziej złożone EDA, takie jak Extended Compact Genetic Algorithm (ECGA) [2], Bayesian Optimization Algorithm (BOA) [3] czy Evolutionary Bayesian Network Algorithm (EBNA) [10], traktują X jako wektor zależnych zmiennych losowych i wyznaczają łączny rozkład prawdopodobieństwa wektora losowego X.

EDA są uważane za bardzo efektywne w rozwiązywaniu problemów optymalizacji, gdyż oprócz standardowych mechanizmów ewolucyjnych wykorzystują wiedzę o rozwiązywanym problemie optymalizacji, pozyskiwaną w trakcie swojego działania, reprezentowaną przez konstruowany rozkład prawdopodobieństwa. Umożliwia on losowanie rozwiązań o wysokich (dla problemów maksymalizacji) lub niskich (dla problemów minimalizacji) wartościach funkcji celu, a także wskazuje obszary przestrzeni poszukiwań warte dokładnego przeszukania. EDA są też cenione za to, że oprócz samego rozwiązania problemu optymalizacji dostarczają informacji, która może być wykorzystana do rozwiązywania podobnych problemów optymalizacji (m.in. z podobnymi zależnościami między współrzędnymi wektora rozwiązania).

Słabą stroną EDA, zwłaszcza tych używających łącznych rozkładów prawdopodobieństwa, jest bardzo duża złożoność obliczeniowa i bardzo długi czas obliczeń, szczególnie w przypadku wysokiej wymiarowości przestrzeni poszukiwań, a także konieczność przetwarzania bardzo dużych populacji, gdyż tylko takie umożliwiają skuteczną estymację wysokowymiarowych rozkładów prawdopodobieństwa.

Interesujące wydało mi się sprawdzenie działania EDA ze złożonymi modelami probabilistycznymi na wysokowymiarowych problemach optymalizacji ze złożonymi zależnościami między współrzędnymi wektora rozwiązania. Spodziewałem się, że EDA odkryją przynajmniej część tych zależności i ograniczą przeszukiwanie do stosownych obszarów przestrzeni poszukiwań. W pracy [HAB1] opisałem zastosowanie algorytmów Extended Compact Genetic Algorithm (ECGA) [2] i Bayesian Optimization Algorithm (BOA) [3] do rozwiązywania praktycznego problemu optymalizacji związanego z konstrukcją systemu wspomagania decyzji dla finansowych szeregów czasowych. Problem polegał na wyznaczeniu podzbioru ustalonego zbioru reguł decyzyjnych $\mathcal{R} = \{f_1, f_2, \ldots, f_n\}$ (czyli przypisaniu regułom decyzyjnym wag binarnych) prowadzącego do jak najwyższej skuteczności podejmowanych decyzji. Przestrzenią poszukiwań była więc przestrzeń $\Omega = \{0,1\}^n$, gdzie n oznacza liczbę dostępnych reguł decyzyjnych (w pracy [HAB1] było ich $n = 350$), zaś funkcją celu $F : \Omega \rightarrow \mathbb{R}$ była skuteczność podejmowanych decyzji obliczana w pewnego rodzaju symulacji wykonywanej na finansowych szeregach czasowych z wykorzystaniem miary oceny inwestycji finansowych wyrażonej przez współczynnik Sharpe'a [11]. Rozważany problem optymalizacji charakteryzował się wysoką wymiarowością przestrzeni poszukiwań ($n = 350$) i silnymi zależnościami między współrzędnymi wektora rozwiązania (skuteczność pojedynczej reguły decyzyjnej zależała zazwyczaj od tego jakie inne reguły decyzyjne były stosowane wraz z nią), a zatem istniała możliwość wykorzystania wiedzy o tych zależnościach przy przeszukiwaniu przestrzeni poszukiwań.

ECGA starał się wyznaczyć zbiory wzajemnie zależnych współrzędnych wektora rozwiązania, zwane

blokami budującymi (ang. building blocks), czyli znaleźć partycję zbioru $\{1, 2, \ldots, n\}$ na rozłączne podzbiory $B_1, B_2, \ldots, B_m$, gdzie liczba tych podzbiorów m była też wyznaczana przez algorytm, tak aby zmienne losowe X_i o indeksach $i \in B_k$ należących do tego samego B_k były wzajemnie zależne, dla każdego $k = 1, 2, \ldots, m$, zaś zmienne losowe X_i i X_j o indeksach $i \in B_{k_1}$ i $j \in B_{k_2}$ należące do dwóch różnych B_{k_1} i B_{k_2} były niezależne, dla każdego $k_1, k_2 = 1, 2, \ldots, m$, $k_1 \neq k_2$. Dla każdego B_k wyznaczany był brzegowy rozkład prawdopodobieństwa wektora losowego X_{B_k} złożonego ze zmiennych losowych X_i o indeksach $i \in B_k$, co w konsekwencji definiowało łączny rozkład prawdopodobieństwa wektora losowego X. Partycja zbioru współrzędnych wektora rozwiązania wyznaczana była przez optymalizację miary oceny partycji opartej na entropii, w sposób zachłanny, poczynając od początkowej partycji na zbiory jednoelementowe, przez ich iteracyjne scalanie w większe bloki budujące. Brzegowe rozkłady prawdopodobieństwa były estymowane na podstawie aktualnej populacji w kolejnych iteracjach algorytmu ewolucyjnego.

Oryginalny algorytm ECGA był zaprojektowany dla znacznie mniej wymiarowych problemów optymalizacji, więc podstawową trudnością związaną z jego zastosowaniem był długi czas działania części algorytmu wyznaczającej w sposób zachłanny partycję zbioru współrzędnych, głównie ze względu na wysoką wymiarowość wektora rozwiązania uniemożliwiającą zachłanne sprawdzenie wszystkich podzbiorów. Zastosowane usprawnienie polegające na wkomponowaniu algorytmu zachłannego w proces ewolucyjny (w każdej iteracji algorytmu ewolucyjnego nie był uruchamiany cały algorytm zachłanny, a jedynie kilka jego iteracji) znacznie przyspieszyło działanie ECGA nie wpływając na jakość znajdowanych rozwiązań.

BOA starał się konstruować sieć bayesowską modelującą zależności między współrzędnymi wektora rozwiązania. Rozkład prawdopodobieństwa był zadany przez graf sieci bayesowskiej i powiązane z każdym jego węzłem warunkowe rozkłady prawdopodobieństwa określone przez tabele prawdopodobieństw warunkowych (ang. Conditional Probability Tables, CPT). Sieć bayesowska była konstruowana iteracyjnie przez optymalizację metryki $K2$ mierzącej prawdopodobieństwo tego, że aktualna populacja została wygenerowana z danej sieci bayesowskiej (podobnie do funkcji wiarygodności). Podobnie jak w przypadku ECGA, podstawową trudnością był długi czas konstruowania sieci bayesowskiej, a duże przyspieszenie uzyskano włączając algorytm zachłanny do procesu ewolucji i w każdej iteracji algorytmu ewolucyjnego wykonując ograniczoną liczbę iteracji algorytmu aktualizującego sieć bayesowską.

Przeprowadzone eksperymenty obliczeniowe pokazały, że ECGA i BOA z zaproponowanymi usprawnieniami są skuteczne w rozwiązywaniu wybranych problemów optymalizacji i czas ich działania jest akceptowalny (w przypadku ECGA czas obliczeń dla populacji złożonej z 4000 rozwiązań ewoluowanej przez 30 iteracji zredukowano ze 115 minut do około 2 minut, a w przypadku BOA czas obliczeń zredukowano z kilku godzin do poniżej 1 minuty). Jednakże, mimo zaproponowanych usprawnień, skuteczność ECGA i BOA dla wysokowymiarowych problemów optymalizacji jest ograniczana przez konieczność stosowania bardzo dużych populacji i wykonywania bardzo dużej liczby iteracji algorytmu ewolucyjnego. Wynika to z faktu, że algorytmy te stosując bardzo złożone modele probabilistyczne mają problemy z efektywnym uczeniem dla wysokowymiarowych problemów optymalizacji, a trudności w uczeniu modeli przenoszą się na słabe możliwości redukcji wymiarowości. Uzyskane wyniki są interesujące z tego względu, że oprócz samego rozwiązania problemu optymalizacji zawierają też model probabilistyczny opisujący charakterystykę zagadnienia, która może być używana przy rozwiązywaniu podobnych problemów optymalizacji (przy pomocy algorytmów opartych na już zbudowanym modelu probabilistycznym).

Badania nad EDA ze złożonymi modelami probabilistycznymi stały się inspiracją do podejścia opartego na redukcji wymiarowości, opisywanego w rozdziale 4.3.

PL

Ze względu na długi czas działania ECGA i BOA na wysokowymiarowych problemach optymalizacji, zainteresowałem się prostszymi EDA i zwiększaniem ich efektywności nie przez stosowanie bardziej złożonych modeli probabilistycznych, a przez wykorzystanie przeszukiwania lokalnego i wydajnej implementacji równoległej na pojawiających się wówczas procesorach graficznych z technologią CUDA. Prace [HAB6] i [HAB7] dotyczą EDA zakładających niezależność zmiennych losowych $X_1, X_2, \ldots, X_n$ i używających jedynie brzegowych rozkładów prawdopodobieństwa.

W pracy [HAB6] zaproponowałem algorytm Hybrid Population-Based Incremental Learning with Local Search (HPBIL-LS), który łączy algorytm Population-Based Incremental Learning (PBIL) [4] z masowym przeszukiwaniem lokalnym realizowanym równolegle na procesorach graficznych[3]. W każdej iteracji HPBIL-LS, wybierane jest losowo 20% rozwiązań z aktualnej populacji i stosowany jest do nich operator przeszukiwania lokalnego (ang. Local Search, LS). Dla rozwiązania $\mathbf{x} \in \Omega = \{0,1\}^n$, operator LS sprawdza wszystkie wektory binarne różniące się od $\mathbf{x}$ na dokładnie jednej współrzędnej, wybiera najlepszy z nich $\hat{\mathbf{x}}$ i jeśli jest on lepszy od $\mathbf{x}$, to zastępuje $\mathbf{x}$ w aktualnej populacji. Podstawowa część algorytmu jest analogiczna do oryginalnego PBIL. Algorytm 1 przedstawia pseudokod HPBIL-LS.

Algorithm 1 Hybrid Population-Based Incremental Learning with Local Search (HPBIL-LS)

```
p = (0.5, 0.5, ..., 0.5);
t = 0;
while not Termination-Condition() do
  P = Random-Population(p);
  Population-Evaluation(P);

  PartialLocalSearch(P);

  e* = Find-Best-Solution(P);

  {updating the probability model};
  p = (1 − α) · p + α · e*;

  {mutating the probability model};
  if random(0, 1) < β then
    u = random-binary-vector();
    p = (1 − γ) · p + γ · u;
  end if

  t = t + 1;
end while
```

Proponowany algorytm oceniano na praktycznym problemie optymalizacji związanym z konstrukcją systemu wspomagania decyzji dla finansowych szeregów czasowych (tym razem złożonego z $n = 500$ reguł decyzyjnych). Algorytm HPBIL-LS znajdował rozwiązania o wysokich wartościach funkcji celu, a czas jego działania z populacją złożoną z 3600 rozwiązań optymalizowaną przez 1000 iteracji wynosił około 30 minut (na komputerze z procesorem Intel Pentium Core2Duo i kartą graficzną nVidia GeForce GTX 280 z 240 rdzeniami CUDA). Dokładne wyniki można znaleźć w pracy [HAB6].

W pracy [HAB7] rozszerzyłem przeszukiwanie lokalne do iteracyjnego przeszukiwania lokalnego (ang. Iterative Local Search, ILS) i iteracyjnego przeszukiwania lokalnego z symulowanym wyżarzaniem (ang. Iterative Local Search with Simulated Annealing, ILS-SA) i włączyłem je do algorytmów

[3] proponowane przeszukiwanie lokalne nazywam masowym, bo jest stosowane do dużej części populacji, w przeciwieństwie do wielu innych podejść, w których przeszukiwaniu lokalnemu poddawane są pojedyncze rozwiązania

Algorithm 2 Iterative Local Search (ILS)

{$\mathbf{x}$ is the original candidate solution to optimise}

$\hat{\mathbf{x}}$ = Best-Neighbour-Solution($\mathbf{x}$);
while $F(\hat{\mathbf{x}}) > F(\mathbf{x})$ **do**
  $\mathbf{x} = \hat{\mathbf{x}}$;
  $\hat{\mathbf{x}}$ = Best-Neighbour-Solution($\mathbf{x}$);
end while

opartych na Population-Based Incremental Learning (PBIL) [4] i Simple Genetic Algorithm (SGA) [12]. Operator ILS wykonuje przeszukiwanie lokalne iteracyjnie: działa podobnie do operatora LS, ale po zastąpieniu $\mathbf{x}$ przez $\hat{\mathbf{x}}$ ponawia przeszukiwanie lokalne dla zaktualizowanego $\mathbf{x}$ i powtarza to po każdej aktualizacji $\mathbf{x}$, zastępując w efekcie $\mathbf{x}$ optimum lokalnym. Algorytm 2 przedstawia pseudokod operatora ILS. Operator ILS-SA rozszerza operator ILS: do rezultatu operatora ILS wprowadza zaburzenie losowe (przez negację określonej liczby losowo wybranych współrzędnych wektora binarnego) otrzymując rozwiązanie $\check{\mathbf{x}}$, uruchamia ponownie operator ILS dla $\check{\mathbf{x}}$ otrzymując rozwiązanie $\bar{\mathbf{x}}$ i powtarza takie zaburzanie i optymalizowanie określoną liczbę iteracji. W kolejnych iteracjach zaburzane jest ostatnio znalezione przez operator ILS $\bar{\mathbf{x}}$, chyba że jest ono gorsze od najlepszego znalezionego dotychczas rozwiązania $\hat{\mathbf{x}}$, wtedy, z pewnym malejącym w kolejnych iteracjach prawdopodobieństwem, zaburzane jest ponownie $\hat{\mathbf{x}}$. Ostatecznie początkowe rozwiązanie $\mathbf{x}$ jest zastępowane przez najlepsze znalezione rozwiązanie $\hat{\mathbf{x}}$. Algorytm 3 przedstawia pseudokod operatora ILS-SA.

Algorithm 3 Iterative Local Search with Simulated Annealing (ILS-SA)

{$\mathbf{x}$ is the original candidate solution to optimise}

$\hat{\mathbf{x}}$ = Iterative-Local-Search($\mathbf{x}$);
$\check{\mathbf{x}}$ = Mutation($\hat{\mathbf{x}}$);
$\bar{\mathbf{x}}$ = Iterative-Local-Search($\check{\mathbf{x}}$);
while $t < T$ **do**
  if $F(\bar{\mathbf{x}}) > F(\hat{\mathbf{x}})$ **then**
    $\hat{\mathbf{x}} = \bar{\mathbf{x}}$;
  else
    if random() $> \exp(\frac{F(\hat{\mathbf{x}})-F(\bar{\mathbf{x}})}{t/T})$ **then**
      $\bar{\mathbf{x}} = \hat{\mathbf{x}}$;
    end if
  end if
  $\check{\mathbf{x}}$ = Mutation($\bar{\mathbf{x}}$);
  $\bar{\mathbf{x}}$ = Iterative-Local-Search($\check{\mathbf{x}}$);

  $t = t + 1$;
end while
$\mathbf{x} = \hat{\mathbf{x}}$;

W pracy [HAB7] dodatkowo rozszerzono przeszukiwanie lokalne do sprawdzania wszystkich wektorów binarnych różniących się od początkowego rozwiązania na co najwyżej δ współrzędnych (a nie na dokładnie jednej, jak było w pracy [HAB6]).

Zaproponowane operatory przeszukiwania lokalnego ILS i ILS-SA są bardzo kosztowne obliczeniowo, ale ich implementacja równoległa na procesorach graficznych z technologią CUDA umożliwiła wykonywanie tysięcy przeszukiwań lokalnych jednocześnie, co pozwoliło utrzymać akceptowalny czas

PL

obliczeń dla całego algorytmu.

W eksperymentach obliczeniowych dotyczących praktycznego problemu optymalizacji związanego z konstrukcją systemu wspomagania decyzji dla finansowych szeregów czasowych (złożonego z $n = 500$ reguł decyzyjnych) algorytm znajdował rozwiązania o wysokich wartościach funkcji celu, a czas jego działania z populacją złożoną z 3200 rozwiązań optymalizowaną przez 1000 iteracji wynosił około 2 minut dla algorytmu używającego operatora ILS i sąsiedztwa rozmiaru $\delta = 4$ oraz około 20 minut dla algorytmu używającego operatora ILS-SA, sąsiedztwa rozmiaru $\delta = 4$ i zaburzania przez negację losowo wybranych 6 współrzędnych wektora binarnego (na komputerze nowszej generacji niż w pracy [HAB6] z procesorem Intel Core i7 950 i dwoma kartami graficznymi nVidia GeForce 580 z 1024 rdzeniami CUDA łącznie). Dokładne wyniki można znaleźć w pracy [HAB7].

Przeprowadzone eksperymenty obliczeniowe pokazały, że zaproponowane proste EDA z masowym przeszukiwaniem lokalnym są skuteczne w rozwiązywaniu wybranych problemów optymalizacji, mimo bardzo prostych (w porównaniu do ECGA i BOA) modeli probabilistycznych. Osiągnięto to dzięki implementacji przeszukiwania lokalnego na procesorach graficznych z technologią CUDA, w której przeszukiwanie lokalne prowadzone jest jednocześnie dla tysięcy rozwiązań, co nie zwiększa znacząco czasu obliczeń, a poprawia wyniki optymalizacji. W implementacji sekwencyjnej proponowane przeszukiwanie lokalne byłoby albo nieefektywne (zbyt mała część populacji poddana przeszukiwaniu lokalnemu lub zbyt szybkie wyżarzanie w operatorze ILS-SA) albo niemożliwe do wykonania ze względu na nierealny czas obliczeń.

Podsumowując, do najważniejszych uzyskanych wyników można zaliczyć usprawnienie ECGA i BOA umożliwiające stosowanie ich dla wysokowymiarowych problemów optymalizacji (dzięki redukcji zachłannej części algorytmów konstrukcji złożonych modeli probabilistycznych), opracowanie HPBIL-LS łączącego PBIL z przeszukiwaniem lokalnym oraz jego dalszych rozszerzeń z operatorami iteracyjnego przeszukiwania lokalnego ILS i iteracyjnego przeszukiwania lokalnego z symulowanym wyżarzaniem ILS-SA.

Z perspektywy czasu, istotną trudnością w stosowaniu EDA do wysokowymiarowych problemów optymalizacji wydaje się fakt, że EDA konstruują zazwyczaj jednomodalne rozkłady prawdopodobieństwa. Tymczasem, w wielu praktycznych problemach optymalizacji często istnieje kilka satysfakcjonujących rozwiązań, porównywalnych ze sobą pod względem wartości funkcji celu, ale bardzo odległych od siebie pod względem metryki w przestrzeni poszukiwań. Zastosowanie wielomodalnych rozkładów prawdopodobieństwa, takich jak mieszaniny gaussowskie, mogłoby poprawić otrzymane wyniki, ale z pewnością zwiększyłoby złożoność obliczeniową i jeszcze bardziej wydłużyłoby czas obliczeń. Badania nad takimi algorytmami, akurat dla problemów optymalizacji dynamicznej, prowadzone przeze mnie we współpracy z innymi osobami, opisane są w pracy [P35] (nie wchodzącej w skład rozprawy habilitacyjnej).

4.2. Podejście oparte na wykrywaniu wzorców częstych

Moje drugie podejście do wysokowymiarowych problemów optymalizacji, opisane w pracach [HAB2, HAB3][4], dotyczy rozwiązywania sekwencji podobnych do siebie problemów optymalizacji, w którym przy rozwiązywaniu kolejnych problemów wykorzystuje się wiedzę pozyskaną z rozwiązań poprzednich problemów. Podejście skupia się na wyznaczaniu powtarzających się w znalezionych wcześniej rozwiązaniach wzorców częstych i ograniczaniu przy ich użyciu przestrzeni poszukiwań podczas rozwiązywania dalszych problemów optymalizacji.

[4] praca [HAB3] jest rozszerzoną wersją pracy [HAB2]

PL

Podejście to było inspirowane pracami dotyczącymi bloków budujących (ang. building blocks) oraz teorii schematów stosowanej w algorytmach genetycznych, m.in. przez Hollanda [13] i Goldberga [12], choć oni stosowali takie podejście do rozwiązań otrzymywanych w kolejnych iteracjach algorytmu genetycznego rozwiązującego pojedynczy problem optymalizacji.

W odróżnieniu od podejścia opartego na algorytmach estymowania rozkładu, opisanego w rozdziale 4.1, w tym podejściu rozpatrywane są nie tyle zależności między współrzędnymi wektora rozwiązania, co konkretne charakterystyczne konfiguracje wartości wybranych współrzędnych tego wektora, co znacznie upraszcza model obliczeniowy.

Badania prowadzone były w kontekście opisanego w rozdziale 4.1 systemu wspomagania decyzji dla finansowych szeregów czasowych (tym razem złożonego z $n = 250$ reguł decyzyjnych). Funkcja celu mierzyła skuteczność podejmowanych decyzji obliczaną w pewnego rodzaju symulacji wykonywanej na finansowych szeregach czasowych, a więc pojawianie się nowych danych finansowych zmieniało funkcję celu i definiowało kolejny problem optymalizacji (co odpowiadało aktualizacji systemu wspomagania decyzji po otrzymaniu nowych danych finansowych poprzez wyznaczenie nowego podzbioru zbioru reguł decyzyjnych optymalnego dla nowego stanu rynku finansowego).

W eksperymentach obliczeniowych używano finansowych szeregów czasowych o częstotliwości jednej minuty. Częstotliwość szeregów czasowych wpływa znacząco na trudność problemu: Szeregi czasowe o niskiej częstotliwości (zazwyczaj w pewien sposób uśredniane) są bardziej stabilne i charakteryzują się większą regularnością, bo ewentualne duże wahania lub odchylenia od ogólnej charakterystyki (m.in. trendu, sezonowości) są albo niwelowane przez uśrednianie, albo w ogóle nierejestrowane ze względu na niską częstotliwość danych. Szeregi czasowe o wysokiej częstotliwości są mniej stabilne i charakteryzują się mniejszą regularnością, bo zazwyczaj występują w nich duże wahania i odchylenia od ogólnej charakterystyki. Częstotliwość jednej minuty prowadziła do częstych, lecz relatywnie niewielkich, zmian funkcji celu.

Wyższa częstotliwość szeregów czasowych używanych w systemie wspomagania decyzji wymaga też szybszych algorytmów przetwarzania danych, aby opracowywane decyzje mogły być dostarczane w odpowiednim czasie.

W trakcie działania systemu wspomagania decyzji, po każdej zmianie w finansowych szeregach czasowych, wyznaczano więc nowy podzbiór zbioru reguł decyzyjnych (o ile sprawność poprzedniego na nowych danych finansowych spadała poniżej ustalonego progu). Algorytm optymalizacji wykorzystywał informacje o poprzednio wyznaczonych podzbiorach zbioru reguł decyzyjnych definiując wzorce częste, tzn. zbiory reguł decyzyjnych, które często występowały lub często nie występowały w poprzednio wyznaczanych rozwiązaniach, w następujący sposób:

Dla ustalonego zbioru reguł decyzyjnych $\mathcal{R} = \{f_1, f_2, \ldots, f_n\}$, wzorcem nazywamy ciąg $\mathbf{k} = (k_1, k_2, \ldots, k_n) \in \{0, 1, \#\}^n$, w którym każdy element k_i, dla $i = 1, 2, \ldots, n$, odpowiada i-tej regule decyzyjnej f_i w taki sposób, że $k_i = 0$ oznacza niewystępowanie, a $k_i = 1$ oznacza występowanie reguły decyzyjnej f_i w rozwiązaniach reprezentujących wzorzec $\mathbf{k}$ (jeśli $k_i = \#$, rozwiązanie reprezentujące wzorzec $\mathbf{k}$ może zawierać regułę decyzyjną f_i lub nie). Rzędem $\mathrm{ord}(\mathbf{k})$ wzorca $\mathbf{k}$ nazywamy liczbę jego elementów równych 0 lub 1.

Rozwiązanie $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \Omega = \{0, 1\}^n$ reprezentuje wzorzec $\mathbf{k}$, co oznaczamy przez $\mathbf{k} \vdash \mathbf{x}$, jeśli rozwiązanie $\mathbf{x}$ nie zawiera żadnej reguły decyzyjnej f_i, dla której $k_i = 0$, oraz zawiera wszystkie reguły decyzyjne f_i, dla których $k_i = 1$, tzn. jeśli $x_i = k_i$, dla każdego $i = 1, 2, \ldots, n$, takiego że $k_i = 0$ lub $k_i = 1$.

Dla ustalonego zbioru rozwiązań $\mathcal{E}$, nośnikiem $\mathrm{supp}(\mathbf{k}, \mathcal{E})$ wzorca $\mathbf{k}$ nazywamy podzbiór zbioru $\mathcal{E}$

PL

zawierający wszystkie rozwiązania reprezentujące wzorzec $\mathbf{k}$, tzn. $\text{supp}(\mathbf{k}, \mathcal{E}) = \{\mathbf{x} \in \mathcal{E} : \mathbf{k} \vdash \mathbf{x}\}$, zaś wzorcem częstym nazywamy wzorzec najwyższego rzędu reprezentowany przez wszystkie rozwiązania ze zbioru $\mathcal{E}$, tzn. $\arg\max\{\text{ord}(\mathbf{k}) : \text{supp}(\mathbf{k}, \mathcal{E}) = \mathcal{E}\}$.

Wyznaczając wzorzec częsty $\mathbf{k}$ dla zbioru rozwiązań poprzednich problemów optymalizacji, przyspieszano rozwiązywanie kolejnego problemu optymalizacji przez ograniczenie przeszukiwania przestrzeni poszukiwań $\Omega = \{0,1\}^n$ do podprzestrzeni $\Omega_{\mathbf{k}} = \{\mathbf{x} \in \Omega : \mathbf{k} \vdash \mathbf{x}\}$ złożonej z rozwiązań reprezentujących wyznaczony wzorzec częsty $\mathbf{k}$ (zmniejszało to wymiarowość problemu optymalizacji z n do $n - \text{ord}(\mathbf{k})$). Ze względu na tendencję do nadmiernego rozrastania się wzorców częstych wraz z upływem czasu, jeśli jakość rozwiązania wyznaczonego w podprzestrzeni $\Omega_{\mathbf{k}}$ była niższa niż ustalony próg, powtarzano rozwiązywanie problemu optymalizacji w pełnej przestrzeni poszukiwań Ω.

Trudnością w proponowanym podejściu było dobranie długości okresu, z którego rozwiązania były używane do wyznaczenia wzorca częstego. Rozpatrywanie długiego okresu 360 minut prowadziło do stabilnych wzorców, nie zmieniających się bardzo w czasie, ale o niskim rzędzie, około $0.10 \cdot n$ - $0.15 \cdot n$, czyli niewielkiej redukcji wymiarowości problemu optymalizacji. Rozpatrywanie krótszych okresów prowadziło do mniej stabilnych wzorców, ale o wyższym rzędzie, czyli większej redukcji wymiarowości problemu optymalizacji. Rozsądnym wyborem okazał się okres 15 minut prowadzący do wzorców o rzędzie około $0.40 \cdot n$.

Proponowane podejście pozwoliło znacznie przyśpieszyć obliczenia do czasu poniżej 1 minuty, redukując wymiarowość problemu optymalizacji do około $0.40 \cdot n$ (zmniejszając rozmiar przestrzeni poszukiwań z 2^{250} do 2^{150}), nie wpływając na jakość znajdowanych rozwiązań. Pokazało to, że w niektórych zagadnieniach proste wykorzystywanie wzorców częstych może konkurować z bardziej złożonymi technikami wyznaczania zależności między współrzędnymi wektora rozwiązania.

Podejście to ma inny charakter niż dwa pozostałe, bo dotyczy sekwencji podobnych do siebie problemów optymalizacji, a nie pojedynczego problemu optymalizacji. Próba definiowania i wykorzystywania analogicznych wzorców częstych w rozwiązaniach otrzymywanych podczas rozwiązywania pojedynczego problemu optymalizacji doprowadziłaby w zasadzie do tego samego co próbują robić standardowe operatory ewolucyjne, które też starają się wyznaczać i zachowywać w procesie ewolucji charakterystyczne konfiguracje współrzędnych wektora rozwiązania.

Z perspektywy czasu, warto zaznaczyć, że przedstawione zagadnienie może być rozpatrywane nie jako sekwencja standardowych problemów optymalizacji, a jako problem optymalizacji dynamicznej i rozwiązywany algorytmami optymalizacji dynamicznej, badania nad którymi opisuję w rozdziale 5.2 (badania te nie wchodzą w zakres rozprawy habilitacyjnej). Pokazały one, że bardzo istotne w optymalizacji dynamicznej jest wprowadzanie mechanizmów predykcji oraz losowych zaburzeń populacji, na przykład przy użyciu techniki losowych imigrantów (ang. random immigrants).

4.3. Podejście oparte na redukcji wymiarowości

W moim trzecim podejściu do wysokowymiarowych problemów optymalizacji, opisanym w pracach [HAB5, HAB8, HAB9, HAB10], staram się iteracyjnie przekształcać przestrzeń poszukiwań do przestrzeni o mniejszej liczbie wymiarów stosując metody redukcji wymiarowości i ewolucyjnie prowadzić poszukiwania rozwiązania optymalnego w zredukowanej przestrzeni poszukiwań. Prace [HAB5] i [HAB10] przedstawiają podejście oparte na metodzie składowych głównych (ang. Principal Component Analysis, PCA) [5]. Praca [HAB8] przedstawia podejście oparte na Locally Linear Embeddings (LLE) [6], zaś praca [HAB9] przedstawia podejście oparte na Kernel Principal Component Analysis (KPCA) [7].

Moje badania były inspirowane pracami dotyczącymi algorytmów inteligencji obliczeniowej do nieliniowej redukcji wymiarowości, zwłaszcza uczenia się rozmaitości (ang. manifold learning), zyskującymi w ostatnich latach coraz większą popularność, m.in. pracami Saula i Roweisa nad LLE [6], Scholdkopfa, Smola i Mullera nad KPCA [7], Hintona i Roweisa nad SNE [14], van der Maatena i Hintona nad t-SNE [15] oraz ich rozszerzeniami. Chociaż celem takich prac była zazwyczaj eksploracja albo wizualizacja danych wysokowymiarowych, interesujące wydało mi się zastosowanie takich technik do wyszukiwania zależności między współrzędnymi wektora rozwiązania i redukowania przestrzeni poszukiwań w algorytmach ewolucyjnych.

W proponowanych podejściach staram się wyznaczać pewną podprzestrzeń przestrzeni poszukiwań, która prawdopodobnie zawiera rozwiązanie optymalne i prowadzić poszukiwania w niej, a nie w całej przestrzeni poszukiwań. W celu wyznaczenia tej podprzestrzeni analizowana jest aktualna populacja traktowana jako próbka punktów przestrzeni poszukiwań z sąsiedztwa rozwiązania optymalnego i przy pomocy metody redukcji wymiarowości stosowanej do tej próbki punków wyznaczana jest nowa zredukowana przestrzeń poszukiwań (izomorficzna z pewną podprzestrzenią oryginalnej przestrzeni poszukiwań).

W przypadku algorytmów opartych na liniowej redukcji wymiarowości metodą składowych głównych szukana podprzestrzeń jest podprzestrzenią liniową przestrzeni poszukiwań wyznaczoną przez wektory własne macierzy kowariancji próbki punktów z aktualnej populacji (z pominięciem wektorów własnych związanych z najmniejszymi wartościami własnymi). W przypadku algorytmów opartych na nieliniowej redukcji wymiarowości metodami LLE lub KPCA szukana podprzestrzeń jest izomorficzna z pewną rozmaitością zanurzoną w przestrzeni poszukiwań, na której leży aktualna populacja (chociaż ta rozmaitość jest zanurzona w oryginalnej przestrzeni poszukiwań może być izomorficzna z przestrzenią o mniejszej liczbie wymiarów).

Proponowana redukcja wymiarowości może być wykonana dopiero po pewnej liczbie iteracji algorytmu ewolucyjnego, kiedy można już przyjąć, że aktualna populacja znajduje się w dość bliskim sąsiedztwie rozwiązania optymalnego. W przeciwnym przypadku rozwiązanie optymalne może nie należeć do wyznaczonej podprzestrzeni, a więc może być nieosiągalne dla algorytmu ewolucyjnego. Z tego też względu, dla osiągania dobrych wyników, konieczne jest okresowe powracanie do oryginalnej przestrzeni poszukiwań i wykonywanie w niej kilku iteracji algorytmu ewolucyjnego w celu sprawdzenia czy zredukowana przestrzeń poszukiwań jest na pewno dobrym wyborem i ewentualnego znalezienia lepszej przestrzeni.

Trudnością proponowanych podejść jest restaurowanie rozwiązań, czyli reprezentacja punktów zredukowanej przestrzeni poszukiwań w oryginalnej przestrzeni poszukiwań, konieczne do obliczania wartości funkcji celu dla zredukowanych rozwiązań (funkcja celu jest bowiem zadana na oryginalnej przestrzeni poszukiwań i jej określenie na zredukowanej przestrzeni poszukiwań może nie być proste, zwłaszcza dla złożonych, nieliniowych, redukcji wymiarowości).

Proponowane podejścia dają dobre rezultaty dla wysokowymiarowych problemów optymalizacji z funkcjami celu o zależnych zmiennych, w których wartości funkcji celu są podobne na pewnych całych podprzestrzeniach przestrzeni poszukiwań. Przykładami takich problemów mogą być opisane w rozdziale 4.4 testowe problemy optymalizacji z rozszerzonymi popularnymi funkcjami celu używanymi do oceny działania algorytmów ewolucyjnych (m.in. funkcje testowe De Jonga, funkcja Rastrigina, funkcja Schwefela, funkcja Griewanka) lub praktyczne problemy optymalizacji związane z konstrukcją systemu wspomagania decyzji dla finansowych szeregów czasowych.

Dalsze rozważania prowadzone są w kontekście optymalizacji funkcji celu $F : \mathbb{R}^n \to \mathbb{R}$ zadanej na

PL

przestrzeni poszukiwań $\Omega = \mathbb{R}^n$ za pomocą ewolucji populacji $\mathcal{P}$ złożonej z rozwiązań $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \in \mathbb{R}^n$.

4.3.1. Podejście oparte na PCA

Pierwsze z opracowanych podejść, oparte na analizie składowych głównych (ang. Principal Component Analysis, PCA) [5], opisane w pracach [HAB5] i [HAB10], polega na wyznaczeniu przekształcenia oryginalnej przestrzeni poszukiwań w zredukowaną przestrzeń poszukiwań o mniejszej wymiarowości w sposób liniowy.

Każdy punkt $\mathbf{x} \in \mathbb{R}^n$ przestrzeni poszukiwań $\Omega = \mathbb{R}^n$ może zostać przedstawiony, bez straty informacji, jako kombinacja liniowa dowolnego zbioru n wektorów ortonormalnych $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n \in \mathbb{R}^n$ w postaci

$$\mathbf{x} = \sum_{i=1}^{n} \alpha_i \mathbf{v}_i, \tag{1}$$

gdzie $\alpha_i = \mathbf{x}^T \mathbf{v}_i \in \mathbb{R}$ jest rzutem punktu $\mathbf{x}$ na kierunek wyznaczony przez wektor $\mathbf{v}_i$, dla $i = 1, 2, \ldots, n$. Zatem każdy punkt $\mathbf{x}$ przestrzeni poszukiwań może zostać przybliżony przez

$$\tilde{\mathbf{x}} = \sum_{i=1}^{m} \alpha_i \mathbf{v}_i + \sum_{i=m+1}^{n} \beta_i \mathbf{v}_i, \tag{2}$$

dla m oznaczającego ustaloną liczbę zredukowanych wymiarów i pewnych stałych $\beta_{m+1}, \beta_{m+2}, \ldots, \beta_n \in \mathbb{R}$ (niezależnych od punktu $\mathbf{x}$). Dla całej populacji $\mathcal{P} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^n$ całkowity średniokwadratowy błąd takiej aproksymacji (zależny oczywiście od wyboru wektorów $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ i stałych $\beta_{m+1}, \beta_{m+2}, \ldots, \beta_n$) wynosi

$$\begin{aligned} \mathrm{MSE}(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n, \beta_{m+1}, \beta_{m+2}, \ldots, \beta_n) &= \sum_{k=1}^{N} ||\mathbf{x}_k - \tilde{\mathbf{x}}_k||^2 = \\ &= \sum_{k=1}^{N} \sum_{i=m+1}^{n} ||(\mathbf{x}_k^T \mathbf{v}_i)\mathbf{v}_i - \beta_i \mathbf{v}_i||^2 = \sum_{k=1}^{N} \sum_{i=m+1}^{n} (\mathbf{x}_k^T \mathbf{v}_i - \beta_i)^2, \end{aligned} \tag{3}$$

bo wektory $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ są ortonormalne. Przyrównując pochodną cząstkową

$$\frac{\partial \mathrm{MSE}(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n, \beta_{m+1}, \beta_{m+2}, \ldots, \beta_n)}{\partial \beta_i} = -\sum_{k=1}^{N} 2(\mathbf{x}_k^T \mathbf{v}_i - \beta_i) = -2 \sum_{k=1}^{N} \mathbf{x}_k^T \mathbf{v}_i + 2N\beta_i \tag{4}$$

do zera, łatwo zauważyć, że

$$\beta_i = \frac{1}{N} \sum_{k=1}^{N} \mathbf{x}_k^T \mathbf{v}_i = \bar{\mathbf{x}}^T \mathbf{v}_i, \qquad \text{gdzie } \bar{\mathbf{x}} = \sum_{k=1}^{N} \mathbf{x}_k, \tag{5}$$

dla $i = m+1, m+2, \ldots, n$. Zatem średniokwadratowy błąd aproksymacji dla optymalnie dobranych wartości $\beta_{m+1}, \beta_{m+2}, \ldots, \beta_n$ określonych powyżej wynosi

$$\mathrm{MSE}(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n) = \sum_{k=1}^{N} \sum_{i=m+1}^{n} (\mathbf{x}_k^T \mathbf{v}_i - \bar{\mathbf{x}}^T \mathbf{v}_i)^2 = \sum_{k=1}^{N} \sum_{i=m+1}^{n} \mathbf{v}_i^T (\mathbf{x}_k - \bar{\mathbf{x}})(\mathbf{x}_k - \bar{\mathbf{x}})^T \mathbf{v}_i = \sum_{i=m+1}^{n} \mathbf{v}_i^T \Sigma \mathbf{v}_i, \tag{6}$$

gdzie Σ jest macierzą kowariancji punktów $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$. Można pokazać, [5], że minimum otrzymanego wyrażenia wynosi

$$\sum_{i=m+1}^{n} \lambda_i, \tag{7}$$

gdzie $\lambda_1, \lambda_2, \ldots, \lambda_n$ oznaczają wartości własne macierzy kowariancji $\mathbf{\Sigma}$ uporządkowane malejąco i jest ono przyjmowane dla wektorów $\mathbf{v}_i$ będących wektorami własnymi macierzy kowariancji $\mathbf{\Sigma}$ odpowiadającymi wartościom własnym λ_i, dla $i = m+1, m+2, \ldots, n$.

Pozwala to na określenie przekształcenia $\Phi : \mathbb{R}^n \to \mathbb{R}^m$ redukującego oryginalną przestrzeń poszukiwań $\mathbb{R}^n$ do zredukowanej przestrzeni poszukiwań $\mathbb{R}^m$, a także przekształcenia $\Psi : \mathbb{R}^m \to \mathbb{R}^n$ restaurującego rozwiązania ze zredukowanej przestrzeni poszukiwań $\mathbb{R}^m$ do oryginalnej przestrzeni poszukiwań $\mathbb{R}^n$ (m.in. w celu obliczenia wartości funkcji celu), jako

$$\Phi(\mathbf{x}) = \mathbf{D}^{-1/2}\mathbf{V}^T\mathbf{x} \qquad \text{i} \qquad \Psi(\mathbf{y}) = \mathbf{V}\mathbf{D}^{1/2}\mathbf{y}, \tag{8}$$

gdzie $\mathbf{V} \in \mathbb{R}^{n \times m}$ jest macierzą o kolumnach $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m$, zaś $\mathbf{D} \in \mathbb{R}^{m \times m}$ jest macierzą diagonalną z liczbami $\lambda_1, \lambda_2, \ldots, \lambda_m$ na głównej przekątnej (zakładając wcześniejsze scentrowanie populacji, tak aby $\bar{\mathbf{x}} = \mathbf{0}$).

4.3.1.1. Algorytm ESIDR

Algorytm Evolution Strategy with Internal Dimensionality Reduction (ESIDR), opisany w pracy [HAB5], jest rozszerzeniem strategii ewolucyjnych (ang. Evolution Strategies, ES) [16] o redukcję wymiarowości opartą na metodzie składowych głównych. Podstawowa część algorytmu opiera się na standardowych operatorach ewolucyjnych używanych w oryginalnych strategiach ewolucyjnych.

Algorytm 4 przedstawia schemat algorytmu ESIDR. Ewolucja rozpoczyna się w oryginalnej przestrzeni poszukiwań $\Omega = \mathbb{R}^n$. Początkowo więc, dla $t = 0$, zredukowana przestrzeń poszukiwań Ω_t to oryginalna przestrzeń poszukiwań Ω, liczba zredukowanych wymiarów n_t równa się n, a przekształcenie redukujące Φ_t i przekształcenie restaurujące Ψ_t to przekształcenia identycznościowe. W kolejnych iteracjach t, wyznaczana jest podprzestrzeń liniowa $\Omega_t = \mathbb{R}^{n_t}$ oryginalnej przestrzeni poszukiwań, przekształcenie redukujące $\Phi_t : \mathbb{R}^n \to \mathbb{R}^{n_t}$ oraz przekształcenie restaurujące $\Psi_t : \mathbb{R}^{n_t} \to \mathbb{R}^n$ za pomocą metody składowych głównych zastosowanej do aktualnej populacji $\mathcal{P}_t$ w oryginalnej przestrzeni poszukiwań Ω (czyli po ewentualnym uprzednim przywróceniu populacji $\mathcal{P}_t$ do oryginalnej przestrzeni poszukiwań za pomocą przekształcenia restaurującego Ψ_{t-1}, populacja $\mathcal{P}_t$ jest bowiem tworzona w zredukowanej przestrzeni poszukiwań Ω_{t-1}). Stosowana redukcja wymiarowości prowadzi do redukcji problemu optymalizacji funkcji celu F na przestrzeni poszukiwań $\Omega = \mathbb{R}^n$ do problemu optymalizacji funkcji celu $G_t = F(\Psi_t)$ na przestrzeni poszukiwań $\Omega_t = \mathbb{R}^{n_t}$.

Działanie algorytmu ESIDR rozpoczyna się od losowego inicjalizowania populacji początkowej $\mathcal{P}_0$ w oryginalnej przestrzeni poszukiwań $\Omega = \mathbb{R}^n$. Z każdym rozwiązaniem $\mathbf{x} \in \mathbb{R}^n$ związany jest dodatkowy wektor $\boldsymbol{\sigma} \in \mathbb{R}^n$ zawierający przypisane do rozwiązania parametry mutacji (jak w oryginalnych strategiach ewolucyjnych). Następnie, w pętli trwającej aż do spełnienia warunku zakończenia (zwykle wykonania ustalonej liczby iteracji lub osiągnięcia zbieżności populacji), rozpoczyna się ewolucja populacji. Na początku każdej iteracji t, tworzona jest populacja potomna $\mathcal{O}_t$ przez zastosowanie standardowych operatorów ewolucyjnych używanych w oryginalnych strategiach ewolucyjnych. Z sumy mnogościowej populacji $\mathcal{P}_t$ i $\mathcal{O}_t$ wybierana jest nowa populacja $\mathcal{P}_{t+1}$ i na jej podstawie przeprowadzana jest redukcja wymiarowości, czyli wyznaczane są n_{t+1}, Ψ_{t+1} i G_{t+1}. Liczba zredukowanych wymiarów n_{t+1} jest dobierana w taki sposób, aby średniokwadratowy błąd aproksymacji nie przekraczał ustalonego progu (zwykle 5%).

Trudnością w algorytmie ESIDR jest transformacja wektorów $\boldsymbol{\sigma}$ przy wykonywaniu redukcji przestrzeni poszukiwań. Powinny one zostać przystosowane do nowej przestrzeni poszukiwań, w taki sposób, aby zapewnić ten sam zasięg mutacji, co w oryginalnej przestrzeni poszukiwań.

PL

Algorithm 4 Evolution Strategy with Internal Dimensionality Reduction (ESIDR)

$n_0 = n$; $\Psi_0 = \mathbf{1}$; $G_0 = F$;
$\mathcal{P}_0$ = Random-Population(N);
Population-Evaluation($\mathcal{P}_0, G_0$);
$t = 0$;
while not Termination-Condition($\mathcal{P}_t$) **do**
  $\mathcal{O}_t = \emptyset$;
  while $|\mathcal{O}_t| < M$ **do**
    $([\mathbf{x}_1, \boldsymbol{\sigma}_1], [\mathbf{x}_2, \boldsymbol{\sigma}_2])$ = Parent-Selection($\mathcal{P}_t$);
    $[\tilde{\mathbf{x}}_1, \tilde{\boldsymbol{\sigma}}_1]$ = Global-Intermediary-Recombination($[\mathbf{x}_1, \boldsymbol{\sigma}_1], [\mathbf{x}_2, \boldsymbol{\sigma}_2]$);
    $[\tilde{\mathbf{x}}_2, \tilde{\boldsymbol{\sigma}}_2]$ = Uniform-Crossover($[\mathbf{x}_1, \boldsymbol{\sigma}_1], [\mathbf{x}_2, \boldsymbol{\sigma}_2]$);
    Mutation($[\tilde{\mathbf{x}}_1, \tilde{\boldsymbol{\sigma}}_1]$);
    Mutation($[\tilde{\mathbf{x}}_2, \tilde{\boldsymbol{\sigma}}_2]$);
    $\mathcal{O}_t = \mathcal{O}_t \cup \{[\tilde{\mathbf{x}}_1, \tilde{\boldsymbol{\sigma}}_1], [\tilde{\mathbf{x}}_2, \tilde{\boldsymbol{\sigma}}_2]\}$;
  end while
  $\mathcal{P}_{t+1}$ = Population-Selection($\mathcal{P}_t, \mathcal{O}_t$);
  Restoring-Mutation($\mathcal{P}_{t+1}$);
  $t = t + 1$;
  (n_t, Ψ_t, G_t) = Dependency-Mining($\mathcal{P}_t$);
  $\mathcal{P}_t$ = Population-Transformation($\mathcal{P}_t, n_t, \Psi_t, G_t$);
  Population-Evaluation($\mathcal{P}_t, G_t$);
end while

Działanie algorytmu ESIDR było oceniane na wybranych testowych problemach optymalizacji, opisanych dokładnie w rozdziale 4.4, o wymiarowości od 20 do 500 z funkcjami celu o zależnych zmiennych. Algorytm ESIDR okazał się bardziej wydajny niż oryginalne strategie ewolucyjne: dzięki stosowaniu redukcji wymiarowości potrzebował mniej iteracji na znalezienie rozwiązania optymalnego analizowanego problemu optymalizacji. Dokładne wyniki można znaleźć w pracy [HAB5].

4.3.1.2. Algorytm DEEDR

Algorytm Differential Evolution with Embedded Dimensionality Reduction (DEEDR), opisany w pracy [HAB10], jest rozszerzeniem algorytmu Differential Evolution (DE) [17] o redukcję wymiarowości opartą na metodzie składowych głównych. Podstawowa część algorytmu opiera się na standardowych operatorach ewolucyjnych używanych w oryginalnym Differential Evolution.

Algorytm 5 przedstawia schemat algorytmu DEEDR. W stosunku do algorytmu ESIDR, oprócz zmiany podstawowego algorytmu ewolucyjnego ze strategii ewolucyjnych na Differential Evolution, występuje techniczna różnica w stosowaniu redukcji wymiarowości. Algorytm DEEDR składa się z dwóch części: ewolucji głównej (prowadzonej w oryginalnej przestrzeni poszukiwań) i subewolucji (prowadzonej w zredukowanej przestrzeni poszukiwań). Na początku wykonywana jest zadana liczba iteracji ewolucji głównej w oryginalnej przestrzeni poszukiwań. Następnie przeprowadzana jest redukcja wymiarowości i wykonywana jest zadana liczba iteracji subewolucji w zredukowanej przestrzeni poszukiwań. Czynności te są naprzemiennie powtarzane aż do spełnienia warunku zakończenia.

Algorytm DEEDR był oceniany na praktycznym problemie optymalizacji wag reguł decyzyjnych w systemie wspomagania decyzji dla finansowych szeregów czasowych (złożonym z $n = 5000$ reguł decyzyjnych, podobnym do opisywanego w rozdziale 4.1, ale zamiast wag binarnych rozpatrywane były wagi rzeczywistoliczbowe). Wprowadzona redukcja wymiarowości ułatwiała wyznaczenie rozwiązań o wysokich wartościach funkcji celu. Algorytm DEEDR okazał się bardziej wydajny niż oryginalny Differential Evolution we wszystkich przypadkach biorąc pod uwagę wyniki uzyskiwane po 1000 iteracji i w

PL

Algorithm 5 Differential Evolution with Embedded Dimensionality Reduction (DEEDR) - Main Evolution

```
P_0 = Random-Population(N)
Population-Evaluation(P_0, F)
t = 0
while not Termination-Condition(P_t) do
  for all x ∈ P_t do
    pick randomly distinct x_1, x_2, x_3 from P_t \ {x}
    v = x_1 + α · (x_2 − x_3)
    u = Binomial-Recombination(v, x)
    if F(x) ⩽ F(u) then
      u replaces x in P_{t+1}
    end if
  end for
  Population-Evaluation(P_{t+1}, F)
  t = t + 1
  if Subevolution-Starting-Condition() then
    Subevolution(P_t)
  end if
end while
```

większości przypadków biorąc pod uwagę wyniki uzyskiwane po 2500 iteracji. Dokładne wyniki można znaleźć w pracy [HAB10].

4.3.2. Podejście oparte na LLE

Drugie opracowane podejście, oparte na Locally Linear Embeddings (LLE) [6], opisane w pracy [HAB8], polega na wyznaczeniu przekształcenia oryginalnej przestrzeni poszukiwań w zredukowaną przestrzeń poszukiwań o mniejszej wymiarowości w sposób nieliniowy, ale z zachowaniem liniowości lokalnej.

Przekształcenie takie jest konstruowane w trzech krokach:
1. Dla każdego rozwiązania $\mathbf{x}_i$ z populacji $\mathcal{P} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^n$ wyznaczanych jest jego K najbliższych sąsiadów w populacji $\mathcal{P}$, czyli określany jest ciąg indeksów $n_1^{(i)}, n_2^{(i)}, \ldots, n_K^{(i)} \in \{1, 2, \ldots, N\} \setminus \{i\}$, taki że

$$||\mathbf{x}_i - \mathbf{x}_{n_1^{(i)}}|| \leqslant ||\mathbf{x}_i - \mathbf{x}_{n_2^{(i)}}|| \leqslant \ldots \leqslant ||\mathbf{x}_i - \mathbf{x}_{n_K^{(i)}}|| \leqslant ||\mathbf{x}_i - \mathbf{x}_j|| \tag{9}$$

dla każdego $j \in \{1, 2, \ldots, N\} \setminus \{i, n_1^{(i)}, n_2^{(i)}, \ldots, n_K^{(i)}\}$.
2. Każde rozwiązanie $\mathbf{x}_i \in \mathcal{P}$ aproksymowane jest przez kombinację liniową swoich K najbliższych sąsiadów z populacji $\mathcal{P}$, czyli wyznaczane są współczynniki liniowe $w_1^{(i)}, w_2^{(i)}, \ldots, w_K^{(i)} \in \mathbb{R}$ minimalizujące funkcję błędu kwadratowego

$$\mathrm{Err}(w_1^{(i)}, w_2^{(i)}, \ldots, w_K^{(i)}) = ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2, \qquad \text{gdzie } \hat{\mathbf{x}}_i = \sum_{k=1}^{K} w_k^{(i)} \mathbf{x}_{n_k^{(i)}}, \tag{10}$$

z ograniczeniem $\sum_{k=1}^{K} w_k^{(i)} = 1$.
3. Dla każdego rozwiązania $\mathbf{x}_i \in \mathcal{P}$ wyznaczane jest jego zanurzenie $\mathbf{y}_i \in \mathbb{R}^m$ w zredukowaną przestrzeń poszukiwań $\mathbb{R}^m$ (liczba zredukowanych wymiarów m jest ustalonym parametrem metody LLE), w taki sposób, aby było ono jak najlepiej aproksymowane przez kombinację liniową zanurzeń wyznaczonych K najbliższych sąsiadów rozwiązania $\mathbf{x}_i$ z populacji $\mathcal{P}$ z takimi samymi współczynnikami liniowymi

PL

Algorithm 6 Differential Evolution with Embedded Dimensionality Reduction (DEEDR) - Subevolution

```
Search-Space-Reduction()
R_0 = Population-Reduction(P_t)
s = 0;
while not Subevolution-Termination-Condition(R_s) do
  for all x ∈ R_s do
    pick randomly distinct x_1, x_2, x_3 from R_s \ {x}
    v = x_1 + α · (x_2 − x_3)
    u = Binomial-Recombination(v, x)
    if F(x) ⩽ F(u) then
      u replaces x in R_{s+1}
    end if
  end for
  Reduced-Population-Evaluation(R_{s+1}, F)
  s = s + 1
end while
Search-Space-Restoring()
P_t = Population-Restoring(R_{s−1})
```

co w oryginalnej przestrzeni poszukiwań. Dokładniej, wyznaczana jest zredukowana populacja $\mathcal{R} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\} \subset \mathbb{R}^m$, gdzie każde zredukowane rozwiązanie $\mathbf{y}_i \in \mathbb{R}^m$ jest zanurzeniem oryginalnego rozwiązania $\mathbf{x}_i \in \mathbb{R}^n$ w zredukowaną przestrzeń poszukiwań $\mathbb{R}^m$, minimalizująca funkcję błędu

$$\mathrm{Err}(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N) = \sum_{i=1}^{N} ||\mathbf{y}_i - \hat{\mathbf{y}}_i||^2, \qquad \text{gdzie } \hat{\mathbf{y}}_i = \sum_{k=1}^{K} w_k^{(i)} \mathbf{y}_{n_k^{(i)}}, \tag{11}$$

z ograniczeniem $\mathbf{Y}^T\mathbf{Y} = n\mathbf{I}$, gdzie $\mathbf{Y} \in \mathbb{R}^{m \times N}$ jest macierzą o kolumnach $\mathbf{y}_i$ oraz $\mathbf{I}$ jest macierzą identycznościową. Dokładny sposób wyznaczania zredukowanej populacji można znaleźć w pracy [HAB8].

Z punktu widzenia opracowanego podejścia, istotną wadą metody LLE jest brak określonego przekształcenia odwrotnego dla dowolnego punktu zredukowanej przestrzeni poszukiwań (metoda LLE wyznacza zanurzenia określonych punktów oryginalnej przestrzeni poszukiwań w zredukowaną przestrzeń poszukiwań, ale nie określa explicite żadnego przekształcenia z oryginalnej przestrzeni poszukiwań w zredukowaną przestrzeń poszukiwań, a tym bardziej przekształcenia odwrotnego), przez co nie można w naturalny sposób liczyć wartości funkcji celu dla zredukowanych rozwiązań.

W swoich badaniach, określoną przez LLE metodę zanurzania punktów oryginalnej przestrzeni poszukiwań $\mathbb{R}^n$ w zredukowaną przestrzeń poszukiwań $\mathbb{R}^m$ rozwinąłem do przekształcenia redukującego $\Phi : \mathbb{R}^n \to \mathbb{R}^m$ w następujący sposób: dla dowolnego punktu $\mathbf{x} \in \mathbb{R}^n$ przestrzeni poszukiwań $\mathbb{R}^n$ wyznaczane jest najbliższe mu rozwiązanie $\mathbf{x}_i$ w populacji $\mathcal{P}$; następnie punkt $\mathbf{x}$ aproksymowany jest przez kombinację liniową K najbliższych sąsiadów wyznaczonego rozwiązania $\mathbf{x}_i$ z populacji $\mathcal{P}$, czyli wyznaczane są odpowiednie współczynniki liniowe $w_1, w_2, \ldots, w_K \in \mathbb{R}$ (analogicznie jak w punkcie 2): zanurzenie $\Phi(\mathbf{x})$ punktu $\mathbf{x}$ w zredukowaną przestrzeń poszukiwań $\mathbb{R}^m$ określane jest jako kombinacja liniowa zanurzeń wyznaczonych K najbliższych sąsiadów rozwiązania $\mathbf{x}_i$ z populacji $\mathcal{P}$ z takimi samymi współczynnikami liniowymi co w oryginalnej przestrzeni poszukiwań, czyli

$$\Phi(\mathbf{x}) = \sum_{k=1}^{K} w_k \mathbf{y}_{n_k^{(i)}}. \tag{12}$$

Podobnie można zdefiniować przekształcenie restaurujące $\Psi : \mathbb{R}^m \to \mathbb{R}^n$ ze zredukowanej przestrzeni poszukiwań $\mathbb{R}^m$ w oryginalną przestrzeń poszukiwań $\mathbb{R}^n$.

PL

4.3.2.1. Algorytm DEELLE

Algorytm Differential Evolution Enhanced by Locally Linear Embeddings (DEELLE), opisany w pracy [HAB8], jest rozszerzeniem algorytmu Differential Evolution (DE) [17] o redukcję wymiarowości opartą na metodzie LLE. Podstawowa część algorytmu opiera się na standardowych operatorach ewolucyjnych używanych w oryginalnym Differential Evolution.

Algorytm 7 przedstawia schemat algorytmu DEELLE. Podobnie jak algorytm DEEDR, składa się on z dwóch części: ewolucji głównej (prowadzonej w oryginalnej przestrzeni poszukiwań) i subewolucji (prowadzonej w zredukowanej przestrzeni poszukiwań). Na początku wykonywana jest zadana liczba iteracji ewolucji głównej w oryginalnej przestrzeni poszukiwań. Następnie przeprowadzana jest redukcja wymiarowości i wykonywana jest zadana liczba iteracji subewolucji w zredukowanej przestrzeni poszukiwań. Czynności te są naprzemiennie powtarzane aż do spełnienia warunku zakończenia.

Algorithm 7 Differential Evolution Enhanced by Locally Linear Embeddings (DEELLE)

$\mathcal{P}_0 =$ Random-Population(N)
Population-Evaluation($\mathcal{P}_0, F$)
$t = 0$
while not Termination-Condition($\mathcal{P}_t$) **do**
  for all $\mathbf{x} \in \mathcal{P}_t$ **do**
    pick randomly distinct $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ from $\mathcal{P}_t \setminus \{\mathbf{x}\}$
    $\mathbf{v} = \mathbf{x}_1 + \alpha \cdot (\mathbf{x}_2 - \mathbf{x}_3)$
    $\mathbf{u} =$ Binomial-Recombination($\mathbf{v}, \mathbf{x}$)
    if $F(\mathbf{x}) \leqslant F(\mathbf{u})$ **then**
      $\mathbf{u}$ replaces $\mathbf{x}$ in $\mathcal{P}_{t+1}$
    end if
  end for
  Population-Evaluation($\mathcal{P}_{t+1}, F$)
  $t = t + 1$
  if Subevolution-Starting-Condition() **then**
    Search-Space-Reduction()
    $\mathcal{R}_0 =$ Population-Reduction($\mathcal{P}_t$)
    $s = 0$;
    while not Subevolution-Termination-Condition($\mathcal{R}_s$) **do**
      for all $\mathbf{x} \in \mathcal{R}_s$ **do**
        pick randomly distinct $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ from $\mathcal{R}_s \setminus \{\mathbf{x}\}$
        $\mathbf{v} = \mathbf{x}_1 + \alpha \cdot (\mathbf{x}_2 - \mathbf{x}_3)$
        $\mathbf{u} =$ Binomial-Recombination($\mathbf{v}, \mathbf{x}$)
        if $F(\mathbf{x}) \leqslant F(\mathbf{u})$ **then**
          $\mathbf{u}$ replaces $\mathbf{x}$ in $\mathcal{R}_{s+1}$
        end if
      end for
      Reduced-Population-Evaluation($\mathcal{R}_{s+1}, F$)
      $s = s + 1$
    end while
    Search-Space-Restoring()
    $\mathcal{P}_t =$ Population-Restoring($\mathcal{R}_{s-1}$)
  end if
end while

Działanie algorytmu DEELLE było oceniane na wybranych testowych problemach optymalizacji, opisanych dokładnie w rozdziale 4.4, o wymiarowości 50, 100 i 250 z funkcjami celu o zależnych zmiennych i praktycznym problemie optymalizacji wag reguł decyzyjnych w systemie wspomagania decyzji

dla finansowych szeregów czasowych (jak w przypadku algorytmu DEEDR, ale złożonym z $n = 500$ reguł decyzyjnych). Algorytm DEELLE okazał się bardziej wydajny niż oryginalny Differential Evolution, zarówno na problemach testowych jak i na problemie praktycznym. Dokładne wyniki można znaleźć w pracy [HAB8].

4.3.3. Podejście oparte na KPCA

Trzecie z opracowanych podejść, oparte na Kernel Principal Component Analysis (KPCA) [7], opisane w pracy [HAB9], polega na przekształceniu aktualnej populacji najpierw do pewnej przestrzeni wysokowymiarowej, gdzie rozwiązania mogą być łatwiej separowane (geometria rozmaitości określanej przez aktualną populacje jest prostsza), a następnie zredukowaniu wysokowymiarowego zanurzenia aktualnej populacji do pewnej przestrzeni niskowymiarowej za pomocą metody składowych głównych użytej w przestrzeni wysokowymiarowej.

Niech $\Upsilon : \Omega \to \Pi$ będzie przekształceniem, niekoniecznie liniowym, oryginalnej przestrzeni poszukiwań $\Omega = \mathbb{R}^n$ w pewną wysokowymiarową przestrzeń liniową Π. Załóżmy wstępnie, że zanurzenie populacji $\mathcal{P} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^n$ w przestrzeni Π określone przez przekształcenie Υ jest scentrowane, czyli

$$\sum_{k=1}^{N} \Upsilon(\mathbf{x}_k) = \mathbf{0}, \tag{13}$$

gdzie $\Upsilon(\mathbf{x}_k)$ jest zanurzeniem rozwiązania $\mathbf{x}_k$ w wysokowymiarową przestrzeń Π (założenie to później będzie można ominąć).

W wysokowymiarowej przestrzeni Π dokonujemy redukcji wymiarowości zanurzenia populacji $\mathcal{P}$, czyli punktów $\Upsilon(\mathbf{x}_1), \Upsilon(\mathbf{x}_2), \ldots, \Upsilon(\mathbf{x}_N)$, metodą składowych głównych wyznaczając wartości i wektory własne macierzy kowariancji

$$\mathbf{\Sigma} = \frac{1}{N} \sum_{l=1}^{N} \Upsilon(\mathbf{x}_l)\Upsilon(\mathbf{x}_l)^T. \tag{14}$$

Łatwo zauważyć, że wszystkie wektory własne macierzy kowariancji muszą być kombinacjami liniowymi zanurzeń $\Upsilon(\mathbf{x}_1)$, $\Upsilon(\mathbf{x}_2)$, ..., $\Upsilon(\mathbf{x}_N)$ i leżeć w przestrzeni liniowej generowanej przez zanurzenia $\Upsilon(\mathbf{x}_1)$, $\Upsilon(\mathbf{x}_2)$, ..., $\Upsilon(\mathbf{x}_N)$, a więc równania własne

$$\lambda_i \mathbf{v}_i = \mathbf{\Sigma} \mathbf{v}_i, \tag{15}$$

dla $i = 1, 2, \ldots, m$ (liczba zredukowanych wymiarów m jest ustalonym parametrem metody KPCA), gdzie $\lambda_1, \lambda_2, \ldots, \lambda_m$ są największymi wartościami własnymi macierzy kowariancji $\mathbf{\Sigma}$ uporządkowanymi malejąco, a $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m$ odpowiadającymi im wektorami własnymi, są równoważne układom równań liniowych

$$\Upsilon(\mathbf{x}_k)^T(\lambda_i \mathbf{v}_i) = \Upsilon(\mathbf{x}_k)^T(\mathbf{\Sigma} \mathbf{v}_i), \qquad \text{dla każdego } k = 1, 2, \ldots, N. \tag{16}$$

Dla każdego $i = 1, 2, \ldots, m$, wektor własny $\mathbf{v}_i$ jest kombinacją liniową $\Upsilon(\mathbf{x}_1)$, $\Upsilon(\mathbf{x}_2)$, ..., $\Upsilon(\mathbf{x}_N)$, więc może być przedstawiony jako

$$\mathbf{v}_i = \sum_{j=1}^{N} \alpha_{ij} \Upsilon(\mathbf{x}_j), \tag{17}$$

gdzie $\boldsymbol{\alpha}_i = (\alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{iN})^T \in \mathbb{R}^N$ jest wektorem odpowiednich współczynników liniowych. Wówczas układ równań liniowych (16) sprowadza się do

$$\lambda_i \sum_{j=1}^{N} \alpha_{ij} \Upsilon(\mathbf{x}_k)^T \Upsilon(\mathbf{x}_j) = \sum_{j=1}^{N} \alpha_{ij} \Upsilon(\mathbf{x}_k)^T \mathbf{\Sigma} \Upsilon(\mathbf{x}_j), \qquad \text{dla każdego } k = 1, 2, \ldots, N, \tag{18}$$

i korzystając z (14) do

$$\lambda_i \sum_{j=1}^{N} \alpha_{ij} \Upsilon(\mathbf{x}_k)^T \Upsilon(\mathbf{x}_j) = \frac{1}{N} \sum_{j=1}^{N} \sum_{l=1}^{N} \alpha_{ij} \Upsilon(\mathbf{x}_k)^T \Upsilon(\mathbf{x}_l) \Upsilon(\mathbf{x}_l)^T \Upsilon(\mathbf{x}_j), \tag{19}$$

zatem po wprowadzeniu macierzy $\mathbf{K} \in \mathbb{R}^{N \times N}$ o elementach $k_{ij} = \Upsilon(\mathbf{x}_i)^T \Upsilon(\mathbf{x}_j)$, dla $i, j = 1, 2, \ldots, N$, układ równań liniowych (16) jest równoważny

$$N\lambda_i \mathbf{K} \boldsymbol{\alpha}_i = \mathbf{K}^2 \boldsymbol{\alpha}_i. \tag{20}$$

Można pokazać, [7], że rozwiązania układu równań liniowych (16) są określane przez wartości własne $N\lambda_i$ i odpowiadające im wektory własne $\boldsymbol{\alpha}_i$ macierzy $\mathbf{K}$, a zanurzenia $\Upsilon(\mathbf{x}_k)$ rozwiązań $\mathbf{x}_k$ mogą zostać przekształcone do zredukowanych rozwiązań $\mathbf{y}_k = (y_{k1}, y_{k2}, \ldots, y_{km}) \in \mathbb{R}^m$ przez rzutowanie $\Upsilon(\mathbf{x}_k)$ na m wektorów własnych $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_m$, więc

$$y_{ki} = \mathbf{v}_i^T \Upsilon(\mathbf{x}_k) = \sum_{j=1}^{N} \alpha_{ij} \Upsilon(\mathbf{x}_j)^T \Upsilon(\mathbf{x}_k) = \sum_{j=1}^{N} \alpha_{ij} k_{jk}, \tag{21}$$

dla $i = 1, 2, \ldots, m$. Oznacza to, że podczas redukcji wymiarowości nie ma potrzeby bezpośredniego wyliczania wartości przekształcenia Υ, a wystarczające jest tylko wyliczenie macierzy $\mathbf{K}$ (jest to tak zwany kernel trick, [7]).

Założenie (13), że obraz populacji jest scentrowany, można ominąć zastępując wybraną macierz $\mathbf{K}$ macierzą

$$\mathbf{K} = \mathbf{K} - \mathbf{1K} - \mathbf{K1} + \mathbf{1K1}, \tag{22}$$

gdzie $\mathbf{1}$ jest macierzą jedynek podzielonych przez N tego samego rozmiaru co macierz $\mathbf{K}$ [7].

W literaturze spotyka się wiele funkcji kernelowych, takich jak funkcje wielomianowe, funkcje gaussowskie, funkcje radialne, funkcje sigmoidalne oraz funkcje konstruowane z określonych przekształceń. W opisywanym podejściu stosowane są funkcje gaussowskie $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ określone jako:

$$k(\mathbf{x}, \mathbf{y}) = \exp(-\frac{||\mathbf{x} - \mathbf{y}||^2}{2\sigma^2}), \tag{23}$$

gdzie σ jest parametrem, a macierz $\mathbf{K}$ składa się z elementów $k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, dla $i, j = 1, 2, \ldots, N$.

Podobnie jak w przypadku LLE, wadą metody KPCA jest brak określonego przekształcenia odwrotnego dla dowolnego punktu zredukowanej przestrzeni poszukiwań, przez co nie można w naturalny sposób liczyć wartości funkcji celu dla zredukowanych rozwiązań. W swoich badaniach przyjąłem podobny sposób definiowania przekształcenia redukującego $\Phi : \mathbb{R}^n \to \mathbb{R}^m$ i przekształcenie restaurującego $\Psi : \mathbb{R}^m \to \mathbb{R}^n$ jak w przypadku LLE.

4.3.3.1. Algorytm ESwKPCDR

Algorytm Evolution Strategy with Kernel Principal Components Dimensionality Reduction (ESwKPCDR), opisany w pracy [HAB9], jest rozszerzeniem strategii ewolucyjnych (ang. Evolution Strategies, ES) [16] o redukcję wymiarowości opartą na metodzie KPCA. Podstawowa część algorytmu opiera się na standardowych operatorach ewolucyjnych używanych w oryginalnych strategiach ewolucyjnych.

Algorytm 8 przedstawia schemat algorytmu ESwKPCDR. Podobnie jak poprzednie algorytmy, składa się on z dwóch części: ewolucji głównej (prowadzonej w oryginalnej przestrzeni poszukiwań) i subewolucji (prowadzonej w zredukowanej przestrzeni poszukiwań). Na początku wykonywana jest zadana liczba iteracji ewolucji głównej w oryginalnej przestrzeni poszukiwań. Następnie przeprowadzana

Algorithm 8 Evolution Strategy with Kernel Principal Components Dimensionality Reduction (ESwKPCDR)

$\mathcal{P}_0$ = Random-Population(N, M)
Population-Evaluation$(\mathcal{P}_0, F)$
$t = 0$
while not Termination-Condition$(\mathcal{P}_t)$ **do**
  $\mathcal{P}_t^{(P)}$ = Parent-Selection$(\mathcal{P}_t, M)$
  $\mathcal{P}_t^{(O)}$ = Recombination$(\mathcal{P}_t^{(P)}, M)$
  $\mathcal{P}_t^{(O)}$ = Mutation$(\mathcal{P}_t^{(O)}, M)$
  Population-Evaluation$(\mathcal{P}_t^{(O)}, F)$
  $\mathcal{P}_{t+1}$ = New-Population-Selection$(\mathcal{P}_t, \mathcal{P}_t^{(O)})$
  $t = t + 1$
  if Subevolution-Starting-Condition() **then**
    Search-Space-Reduction()
    $\mathcal{R}_0$ = Population-Reduction$(\mathcal{P}_t)$
    $s = 0$;
    while not Subevolution-Termination-Condition$(\mathcal{R}_s)$ **do**
      $\mathcal{R}_s^{(P)}$ = Parent-Selection$(\mathcal{R}_s, M)$
      $\mathcal{R}_s^{(O)}$ = Recombination$(\mathcal{R}_s^{(P)}, M)$
      $\mathcal{R}_s^{(O)}$ = Mutation$(\mathcal{R}_s^{(O)}, M)$
      Reduced-Population-Evaluation$(\mathcal{R}_s^{(O)}, F)$
      $\mathcal{R}_{s+1}$ = New-Population-Selection$(\mathcal{R}_s, \mathcal{R}_s^{(O)})$
      $s = s + 1$
    end while
    Search-Space-Restoring()
    $\mathcal{P}_t$ = Population-Restoring$(\mathcal{R}_{s-1})$
  end if
end while

jest redukcja wymiarowości i wykonywana jest zadana liczba iteracji subewolucji w zredukowanej przestrzeni poszukiwań. Czynności te są naprzemiennie powtarzane aż do spełnienia warunku zakończenia.

Algorytm ESwKPCDR był oceniany na wybranych testowych problemach optymalizacji, opisanych dokładnie w rozdziale 4.4, o wymiarowości 50, 100 i 250 z funkcjami celu o zależnych zmiennych. Algorytm ESwKPCDR okazał się bardziej wydajny niż oryginalne strategie ewolucyjne. Dokładne wyniki można znaleźć w pracy [HAB9].

4.4. Wysokowymiarowe testowe problemy optymalizacji

W eksperymentach obliczeniowych oceniających działanie proponowanych algorytmów były wykorzystywane dwa rodzaje problemów optymalizacji: praktyczne problemy optymalizacji związane z konstrukcją systemu wspomagania decyzji dla finansowych szeregów czasowych oraz adaptacje popularnych testowych problemów optymalizacji używanych często do testowania algorytmów ewolucyjnych. Problemy pierwszego rodzaju, opisane już w rozdziale 4.1, są o tyle znaczące, że dotyczą praktycznych zastosowań, zawierają nieregularne zależności między współrzędnymi wektora rozwiązania i mogą mieć różny stopień trudności w zależności od finansowych szeregów czasowych używanych w symulacji wykonywanej podczas obliczania wartości funkcja celu. Problemy drugiego rodzaju są bardziej abstrakcyjne, ich charakter i trudność nie zawsze odpowiadają charakterom i trudnościom problemów spotykanych w praktyce, ale umożliwiają łatwą ocenę działania algorytmu ewolucyjnego na trudnym problemie

PL

optymalizacji. Należą do nich m.in. funkcje testowe De Jonga, funkcja Rastrigina, funkcja Schwefela, funkcja Griewanka, czy zestawy funkcji testowych ogłaszane na konferencji IEEE Congress of Evolutionary Computation [18].

Jako że proponowane podejścia dotyczyły wysokowymiarowych problemów optymalizacji z funkcjami celu o zależnych zmiennych, a oryginalne funkcje testowe mają zazwyczaj zmienne niezależne, zaproponowałem modyfikacje oryginalnych testowych problemów optymalizacji w sposób podobny do tworzenia funkcji testowych *deceptive one max* i *k-deceptive one max* dla algorytmu ECGA zaproponowanego w pracy [2].

Każda oryginalna funkcja testowa $f : \mathbb{R}^n \to \mathbb{R}$ została rozszerzona przez określone przekształcenie $\Phi : \mathbb{R}^m \to \mathbb{R}^n$, dla $m > n$, w taki sposób, że docelowa funkcja celu $F : \mathbb{R}^m \to \mathbb{R}$ (zwana dalej złudną funkcją celu) była złożeniem przekształcenia Φ i oryginalnej funkcji celu f, tzn. $F(\mathbf{x}) = f(\Phi(\mathbf{x}))$ dla $\mathbf{x} \in \mathbb{R}^m$. Tak uzyskana złudna funkcja celu była formalnie funkcją m zmiennych, chociaż jej zmienne były zależne i w istocie mogłaby być optymalizowana jako funkcja $n < m$ zmiennych. W eksperymentach obliczeniowych stosowano różne wartości n i m oraz różne przekształcenia Φ, co pozwalało na różnicowanie stopnia trudności problemu.

Ponadto, każda złudna funkcja celu została rozszerzona do k-złudnej funkcji celu w następujący sposób: argument k-złudnej funkcji celu, czyli $\mathbf{x} \in \mathbb{R}^m$, był dzielony na bloki k kolejnych współrzędnych (dla uproszczenia zakładamy, że m jest wielokrotnością k), dla każdego bloku była obliczana wartość złudnej funkcji celu, a następnie wyniki były uśredniane i otrzymana wartość średnia była wartością k-złudnej funkcji celu.

Uzyskane wysokowymiarowe rozszerzenia testowych problemów optymalizacji okazały się trudne. Większość sprawdzanych algorytmów ewolucyjnych rozwiązywała takie problemy ze skutecznością podobną do rozwiązywania problemów z m-wymiarową przestrzenią poszukiwań, a więc nie wykorzystywała zależności między współrzędnymi wektora rozwiązania. Proponowane podejście oparte na redukcji wymiarowości w większości przypadków znajdowało przekształcenie redukujące przestrzeń poszukiwań (nie zawsze do n wymiarów i nie zawsze zgodne z przekształceniem używanym do konstrukcji funkcji testowej), które umożliwiało redukcję problemu optymalizacji i przyspieszenie obliczeń.

4.5. Quadratic Three-Dimensional Assignment Problem

Uzupełnieniem opisanych trzech podejść do wysokowymiarowych problemów optymalizacji jest praca [HAB4] dotycząca problemu optymalizacji zupełnie innej natury – problemu Quadratic Three-Dimensional Assignment Problem (Q3AP) [19] – gdzie wysoka wymiarowość zaczyna się już od kilkunastu wymiarów. Q3AP jest problemem NP-trudnym, w którym przestrzeń poszukiwań Ω jest zbiorem par permutacji ustalonej długości n (a zatem jej moc to $n! \times n!$), a funkcja celu $F : \Omega \to \mathbb{R}$ jest na tyle nieregularna, że dużym wyzwaniem jest znalezienie rozwiązań optymalnych już dla n rzędu kilkunastu.

Problem Q3AP, oryginalnie sformułowany przez Pierskalla w 1967 roku [19] i opublikowany w Operational Research w 1968 roku [20], zaczął cieszyć się większym zainteresowaniem wraz z rozwojem technologii telefonii komórkowej, w związku z jego wykorzystaniem w protokole Hybrid Automatic Repeat Request (Hybrid-ARQ, HARQ) w systemach komunikacji bezprzewodowej, m.in. w protokołach transmisji danych HSDPA, HSUPA i LTE [21]. Był tematem prac w grancie NFS z dziedziny badań operacyjnych realizowanym na University of Pennsylvania [21] oraz badań naukowych prowadzonych w INRIA [22], a mnie zainteresował jako testowy problem optymalizacji dla algorytmów ewolucyjnych z masowym przeszukiwaniem lokalnym.

12

W pracy [HAB4] zaproponowałem algorytm ewolucyjny HEA-Q3AP łączący algorytm genetyczny z masowym przeszukiwaniem lokalnym realizowanym równolegle na procesorach graficznych z technologią CUDA, podobnie jak w algorytmach opisywanych w rozdziale 4.1. HEA-Q3AP był dodatkowo wyposażony w mechanizm autoadaptacji służący do dynamicznego wyboru operatorów krzyżowania i mutacji. W algorytmie wykorzystywane było 8 popularnych operatorów krzyżowania dla permutacji, PMX, OX, CX, PBX, OBX, PX, LCSX, LOX (pierwsze i drugie permutacje krzyżowane były niezależnie) oraz 4 operatory mutacji dla par permutacji (losowa transpozycja pierwszej permutacji, losowa transpozycja drugiej permutacji, zamiana permutacji pierwszej z drugą, zastąpienie pierwszej permutacji permutacją losową). Do każdego operatora było przypisane prawdopodobieństwo jego użycia, początkowo takie same dla każdego operatora, ale zwiększane w czasie działania algorytmu, kiedy operator doprowadził do poprawienia rozwiązania.

Eksperymenty obliczeniowe dotyczyły testowych problemów optymalizacji uzyskanych przez rozszerzenie popularnych problemów testowych NUG8, NUG10, NUG12, NUG13, NUG14 i NUG15 [23] dla jednowymiarowego problemu Quadratic Assignment Problem (QAP) w sposób stosowany w pracach [24] i [22]. Dla wszystkich tych problemów HEA-Q3AP znalazł rozwiązanie optymalne (wyznaczone algorytmami dokładnymi i znane z literatury), ale w znacznie krótszym czasie niż algorytmy proponowane w pracach [21], [24], [22] oparte głównie na prostych algorytmach metaheurystycznych, m.in. symulowanym wyżarzaniu i przeszukiwaniem z tabu. Udało się także wyznaczyć rozwiązania o wysokich wartościach funkcji celu (ze względu na brak opublikowanych wyników algorytmów dokładnych nie można stwierdzić czy i o ile różnią się one od rozwiązań optymalnych) dla problemów NUG20 i NUG30, dla których nie były wcześniej publikowane nawet rozwiązania przybliżone (prawdopodobnie ze względu na zbyt długi czas obliczeń). Naturalnie porównywanie czasu obliczeń algorytmu HEA-Q3AP z innymi nie jest obiektywne, bo trudno porównywać obliczenia sekwencyjne z obliczeniami masowo równoległymi na procesorach graficznych, trudno też porównywać obliczenia na różnych platformach sprzętowych, ale mimo wszystko warto zauważyć, że proponowane podejście umożliwiło rozwiązanie problemów, które wcześniej nie mogły być rozwiązane ze względu na czas obliczeń przekraczający praktyczne możliwości.

5. Omówienie pozostałych osiągnięć naukowo-badawczych

Oprócz prac naukowo-badawczych opisanych w cyklu artykułów stanowiącym rozprawę habilitacyjną zajmowałem się także innymi zagadnieniami.

5.1. Pozyskiwanie wiedzy z finansowych szeregów czasowych ultra-wysokiej częstotliwości

Badania nad pozyskiwaniem wiedzy z szeregów czasowych ultra-wysokiej częstotliwości rozpocząłem podczas trzymiesięcznego pobytu w kierowanej przez prof. A. Brabazona grupie Natural Computing Research and Applications Group, Complex and Adaptive Systems Laboratory, University College Dublin w Irlandii, a później kontynuowałem w kierowanej przeze mnie Pracowni Inteligencji Obliczeniowej w Instytucie Informatyki Uniwersytetu Wrocławskiego.

Przedmiotem badań był zbiór danych London Stock Exchange Rebuild Order Book (LSE ROB) zawierający informacje o zleceniach i transakcjach rejestrowanych na giełdzie papierów wartościowych w Londynie, a ich celem było pozyskanie wiedzy o rynku finansowym na podstawie charakterystyki kolejek zleceń (zlecenia kupna i sprzedaży papierów wartościowych przesyłane na giełdę papierów

PC

Algorithm 9 Hybrid Evolutionary Algorithm to Q3AP (HEA-Q3AP)

$\mathbf{p}_C = \mathbf{1}/n_C$; $\mathbf{p}_M = \mathbf{1}/n_M$;
$\mathcal{P}_0$ = Random-Population(N);
$t = 0$;
while not Termination-Condition($\mathcal{P}_t$) **do**
  $\mathcal{O}_t = \emptyset$;
  while $|\mathcal{O}_t| < M$ **do**
    $([\mathbf{p}_1, \mathbf{q}_1], [\mathbf{p}_2, \mathbf{q}_2])$ = Parent-Selection($\mathcal{P}_t$);
    if Random-Value$(0, 1) < p_C$ **then**
      Crossover-Operator = Crossover-Operator-Selection($\mathbf{p}_C$);
      $([\tilde{\mathbf{p}}_1, \tilde{\mathbf{q}}_1], [\tilde{\mathbf{p}}_2, \tilde{\mathbf{q}}_2])$ = Crossover-Operator$([\mathbf{p}_1, \mathbf{q}_1], [\mathbf{p}_2, \mathbf{q}_2])$;
    else
      $([\tilde{\mathbf{p}}_1, \tilde{\mathbf{q}}_1], [\tilde{\mathbf{p}}_2, \tilde{\mathbf{q}}_2]) = ([\mathbf{p}_1, \mathbf{q}_1], [\mathbf{p}_2, \mathbf{q}_2])$;
    end if
    if Random-Value$(0, 1) < p_M$ **then**
      Mutation-Operator = Mutation-Operator-Selection($\mathbf{p}_M$);
      Mutation-Operator$([\tilde{\mathbf{p}}_1, \tilde{\mathbf{q}}_1])$;
    end if
    if Random-Value$(0, 1) < p_M$ **then**
      Mutation-Operator = Mutation-Operator-Selection($\mathbf{p}_M$);
      Mutation-Operator$([\tilde{\mathbf{p}}_1, \tilde{\mathbf{q}}_1])$;
    end if
    $\mathcal{O}_t = \mathcal{O}_t \cup \{[\tilde{\mathbf{p}}_1, \tilde{\mathbf{q}}_1], [\tilde{\mathbf{p}}_2, \tilde{\mathbf{q}}_2]\}$
  end while
  Single-Local-Search($\mathcal{O}_t$);
  $\mathcal{P}_{t+1}$ = Population-Selection($\mathcal{P}_t$, $\mathcal{O}_t$);
  Crossover-Operator-Assessment($\mathcal{P}_t, \mathcal{O}_t, \mathbf{p}_C$);
  Mutation-Operator-Assessment($\mathcal{P}_t, \mathcal{O}_t, \mathbf{p}_M$);
  Multiple-Local-Search($\mathcal{P}_{t+1}$);
  $t = t + 1$;
end while
return $\mathcal{P}_t$;

wartościowych są, jeśli to możliwe, łączone ze sobą, co prowadzi do zawarcia transakcji, lub rejestrowane w kolejkach zleceń i oczekują na napłynięcie zleceń, z którymi mogłyby zostać połączone). Inspiracją tych badań były prace ekonomistów (m.in. laureata nagrody Nobla z ekonomii w 2003 roku R. Engle'a [25]), którzy twierdzą, że kolejki zleceń zawierają informacje pozwalające na określenie stanu rynku finansowego [25], [26], [27], co tworzy potrzebę pozyskiwania i wykorzystywania takiej wiedzy m.in. w systemach wspomagania decyzji.

Dużym problemem w przetwarzaniu zbioru danych LSE ROB jest bardzo duża ilość danych wymagająca bardzo efektywnych algorytmów przetwarzania, a także bardzo duże wahania i zaburzenia danych, które utrudniają konstrukcję stabilnych modeli (kolejki zleceń zmieniają się bardzo często, w niektórych przypadkach co kilka sekund i liczą do kilkuset pozycji).

W pracach [P30] i [P43] zastosowano samoorganizujące się sieci Kohonena (ang. Self Organizinig Maps, SOM) [28] do grupowania obrazów kolejki zleceń z kolejnych pięciosekundowych okresów sesji giełdowej, na którego podstawie wyznaczono wzorce charakterystyczne kolejki zleceń, a następnie wyznaczono macierz prawdopodobieństw przejść między stanami rynku finansowego odpowiadającymi tym wzorcom i zauważono, że przejścia między niektórymi stanami są bardziej prawdopodobne niż między innymi, co mogłoby zostać wykorzystane w systemach wspomagania decyzji. W pracy [P40]

PL

wprowadzono reprezentację obrazów kolejki zleceń przez mieszaninę rozkładów gaussowskich, a następnie zaproponowano algorytm ewolucyjny do optymalizacji tej reprezentacji w celu zwiększania efektywności klasyfikacji stanów rynku finansowego reprezentowanych przez obrazy kolejki zleceń za pomocą Support Vector Machines (SVM) [29]. W pracy [P41] zaproponowano wielokryterialny algorytm ewolucyjny do optymalizacji wzorców częstych znalezionych w obrazach kolejki zleceń, tak aby z jednej strony zwiększyć stabilność wzorców (wzorzec powinien definiować stan rynku finansowego, w którym rynek znajduje się przez dostatecznie długi czas), a z drugiej strony zwiększyć uniwersalność wzorców (wzorzec powinien definiować stan rynku finansowego, do którego rynek wraca wielokrotnie).

Mój wkład w tych pracach uważam za znaczący i szacuję go na 65% w pracy [P30], 65% w pracy [P40], 20% w pracy [P41] i 40% w pracy [P43].

Opisywane publikacje

[P30] Piotr Lipinski i Anthony Brabazon. "Pattern Mining in Ultra-High Frequency Order Books with Self-Organizing Maps". W: *Applications of Evolutionary Computation, EvoWorkshops 2014*. T. 8602. Lecture Notes in Computer Science. Springer, 2014, s. 288–298

[P40] Piotr Lipinski, Krzysztof Michalak i Lancucki Adrian. "Improving Classification of Patterns in Ultra-High Frequency Time Series with Evolutionary Algorithms". W: *Proceedings of the Companion Publication of the 2016 Annual Conference on Genetic and Evolutionary Computation, GECCO 2016*. ACM, 2016, s. 127–128

[P41] Krzysztof Michalak, Adrian Lancucki i Piotr Lipinski. "Multiobjective Optimization of Frequent Pattern Models in Ultra-High Frequency Time Series: Stability versus Universality". W: *Proceedings of IEEE Congress of Evolutionary Computation, CEC 2016*. IEEE, 2016, s. 3491 3498

[P43] Anthony Brabazon, Piotr Lipinski i Phillip Hamill. "Characterising Order Book Evolution Using Self-Organising Maps". W: *Evolutionary Intelligence*. T. 9. 4. Springer, 2016, s. 167–179

5.2. Algorytmy ewolucyjne dla problemów optymalizacji dynamicznej

Badania nad algorytmami ewolucyjnymi dla problemów optymalizacji dynamicznej rozpocząłem wspólnie z moim doktorantem Patrykiem Filipiakiem, którego rozprawa doktorska dotycząca tej właśnie tematyki została obroniona na Wydziale Matematyki i Informatyki Uniwersytetu Wrocławskiego w grudniu 2016.

Problem optymalizacji dynamicznej to problem optymalizacji, w którym funkcja celu i funkcje ograniczeń zmieniają się w czasie. Jeśli zmiany nie są częste, a czas obliczeń algorytmów optymalizacji jest wystarczająco krótki, to problem taki można traktować jako sekwencję statycznych problemów optymalizacji i rozwiązywać je kolejno standardowymi algorytmami optymalizacji. Jeśli zmiany są częste lub rozwiązywanie statycznych problemów optymalizacji zajmuje dużo czasu, to warto traktować taki problem w całości i rozwiązywać go algorytmem optymalizacji dynamicznej, który w każdej iteracji stara się znaleźć rozwiązanie optymalizujące aktualną funkcję celu i spełniające aktualne ograniczenia.

Najważniejsze uzyskane wyniki obejmują opracowanie algorytmów optymalizacji dynamicznej wykorzystujących mechanizm predykcji, zarówno predykcji przyszłych wartości funkcji celu i funkcji ograniczeń, jak i predykcji przyszłego położenia rozwiązań optymalnych w przestrzeni poszukiwań, [P20], [P31], [P34], [P37]. Algorytmy optymalizacji dynamicznej zostały zastosowane do problemów kinematyki odwrotnej (ang. inverse kinematics problem), [P22], [P33], [P39], problemów uczenia systemów wspomagania decyzji, [P25], dynamicznego problemu komiwojażera, [P23], i problemu dynamicznej optymalizacji portfela papierów wartościowych, [P44] (praca przyjęta na konferencję EvoStar 2017).

PL

Uzyskane wyniki wchodzą w skład rozprawy doktorskiej Patryka Filipiaka, który w większości prac był głównym autorem, a mój udział szacuję na 15% w pracach [P20], [P22] i [P39], 10% w pracy [P25], 20% w pracy [P33] oraz 30% w pracach [P23], [P31], [P34], [P37] i [P44].

Opisywane publikacje

[P20] Patryk Filipiak, Krzysztof Michalak i Piotr Lipinski. "Infeasibility Driven Evolutionary Algorithm with ARIMA based Prediction Mechanism". W: *Intelligent Data Engineering and Automated Learning, IDEAL 2011*. T. 6936. Lecture Notes in Computer Science. Springer, 2011, s. 345–352

[P22] Patryk Filipiak, Krzysztof Michalak i Piotr Lipinski. "A Predictive Evolutionary Algorithm for Dynamic Constrained Inverse Kinematics Problems". W: *Hybrid Artificial Intelligence Systems, HAIS 2012*. T. 7208. Lecture Notes in Computer Science. Springer, 2012, s. 610-621

[P23] Patryk Filipiak i Piotr Lipinski. "Parallel CHC Algorithm for Solving Dynamic Traveling Salesman Problem Using Many-Core GPU". W: *Artificial Intelligence: Methodology, Systems and Applications, AIMSA 2012*. T. 7557. Lecture Notes in Computer Science. Springer, 2012, s. 305 314

[P25] Krzysztof Michalak, Patryk Filipiak i Piotr Lipinski. "Usage Patterns of Trading Rules in Stock Market Trading Strategies Optimized with Evolutionary Methods". W: *Applications of Evolutionary Computation, EvoWorkshops 2013*. T. 7835. Lecture Notes in Computer Science. Springer, 2013, s. 234–243

[P31] Patryk Filipiak i Piotr Lipinski. "Infeasibility Driven Evolutionary Algorithm with Feed-Forward Prediction Strategy for Dynamic Constrained Optimization Problems". W: *Applications of Evolutionary Computation, EvoWorkshops 2014*. T. 8602. Lecture Notes in Computer Science. Springer, 2014, s. 817–828

[P33] Krzysztof Michalak, Patryk Filipiak i Piotr Lipinski. "Multiobjective Dynamic Constrained Evolutionary Algorithm for Control of a Multi-segment Articulated Manipulator". W: *Intelligent Data Engineering and Automated Learning, IDEAL 2014*. T. 8669. Lecture Notes in Computer Science. Springer, 2014, s. 199–206

[P34] Patryk Filipiak i Piotr Lipinski. "Univariate Marginal Distribution Algorithm with Markov Chain Predictor in Continuous Dynamic Environments". W: *Intelligent Data Engineering and Automated Learning, IDEAL 2014*. T. 8669. Lecture Notes in Computer Science. Springer, 2014, s. 404–411

[P37] Patryk Filipiak i Piotr Lipinski. "Making IDEA-ARIMA Efficient in Dynamic Constrained Optimization Problems". W: *Applications of Evolutionary Computation, EvoWorkshops 2015*. T. 9028. Lecture Notes in Computer Science. Springer, 2015, s. 882–893

[P39] Patryk Filipiak, Krzysztof Michalak i Piotr Lipinski. "Infeasibility Driven Evolutionary Algorithm with the Anticipation Mechanism for the Reaching Goal in Dynamic Constrained Inverse Kinematics". W: *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO 2015*. ACM, 2015, s. 1389–1390

[P44] Patryk Filipiak i Piotr Lipinski. "Dynamic Portfolio Optimization in Ultra-High Frequency Environment". W: *Applications of Evolutionary Computation*. T. 10199. Lecture Notes in Computer Science. Springer, 2017

5.3. Zagadnienia bioinformatyczne

Interesowałem się również zagadnieniami bioinformatycznymi. Początkowo były to badania naukowe związane z analizą ewolucji genomów bakterii, prowadzone w Zakładzie Genomiki Uniwersytetu Wrocławskiego, w których uczestniczyłem. Najważniejsze uzyskane wyniki, przedstawione w formie plakatu na II Polskim Kongresie Genetyki w Warszawie w 2007 [A] i World Congress in Probability and Statistics w Singapurze w 2008 [B], dotyczą zastosowania samoorganizujących się sieci Kohonena (ang. Self Organizing Maps, SOM) [28] i algorytmów ewolucyjnych do grupowania genomów wybranych bakterii. Mój udział w tych pracach dotyczył zaproponowania modelu obliczeniowego opartego na SOM oraz przeprowadzeniu części eksperymentów obliczeniowych. Następnie, prace bioinformatyczne

PL

rozpoczęły się w kierowanej przeze mnie Pracowni Inteligencji Obliczeniowej w Instytucie Informatyki Uniwersytetu Wrocławskiego, zapoczątkowane przez dra Indrajita Sahę realizującego staż podoktorski, i obecnie kontynuowane we współpracy z nim. Najważniejsze uzyskane wyniki, opublikowane w pracach [P38] i [P42], dotyczą opracowania techniki selekcji atrybutów umożliwiających efektywną klasyfikację zgromadzonych sekwencji białek. Mój udział w tych pracach dotyczył koncepcji modelu obliczeniowego, analizy działania algorytmów i interpretacji ich wyników i szacuję go na 15% w pracy [P38] i 10% w pracy [P42].

Opisywane publikacje

[A] Sobczynski, M., Mackiewicz, P., Lipinski, P., Cebrat, S., Zastosowanie sieci Kohonena i algorytmów ewolucyjnych do różnicowania proteomów prokariotycznych, II Polski Kongres Genetyki, Warszawa, 2007 (plakat)

[B] Sobczynski, M., Mackiewicz, P., Lipinski, P., Cebrat, S., The Application of Self Organizing Maps and Evolutionary Algorithms to Distinguish Bacterial Proteomes, World Congress in Probability and Statistics, Singapore, 2008 (plakat)

[P38] Adrian Lancucki, Indrajit Saha i Piotr Lipinski. "A New Evolutionary Gene Selection Technique". W: *Proceedings of IEEE Congress of Evolutionary Computation, CEC 2015*. IEEE, 2015, s. 1612–1619

[P42] Adrian Lancucki, Indrajit Saha, S Bhowmick, U Maulik i Piotr Lipinski. "A New Evolutionary MicroRNA Marker Selection using Next-Generation Sequencing Data". W: *Proceedings of IEEE Congress of Evolutionary Computation, CEC 2016*. IEEE, 2016, s. 2752–2759

5.4. Zagadnienia górnicze

Wspólnie z dr. hab. Radosławem Zimrozem z Wydziału Górniczego Politechniki Wrocławskiej i dr. hab. Edytą Brzychczy z Wydziału Górnictwa i Geoinżynierii AGH zajmowałem się zagadnieniami klasyfikacji stanów urządzeń, zwłaszcza przekładni planetarnych, na podstawie rejestrowanych drgań. Pierwsza część badań dotyczyła wykorzystania drzew klasyfikujących do klasyfikacji stanu urządzeń i jej wyniki zostały przedstawione na konferencji 6th International Conference Mechatronic Systems and Materials w Opolu [P15]. Druga część badań dotyczyła klasyfikacji opartej na sztucznych systemach immunologicznych (ang. Artificial Immune Systems, AIS) i jej wyniki zostały przedstawione w pracy [P28]. Oba podejścia umożliwiły klasyfikację danych z bardzo dobrą skutecznością, sięgającą 100% na niektórych zestawach danych testowych. Mój udział w tych pracach dotyczył zaproponowania modelu obliczeniowego oraz przeprowadzeniu części eksperymentów obliczeniowych i szacuję go na 40% w pracy [P15] oraz 30% w pracy [P28].

Opisywane publikacje

[P15] Piotr Lipinski, Edyta Brzychczy i Radoslaw Zimroz. "Planetary Gearboxes Condition Monitoring under Non-stationary Operations by Adaptive Decision Trees based Classification of Vibration Data in Multidimensional Symptom Space". W: *Proceedings of 6th International Conference Mechatronic Systems and Materials, Opole, Poland*. 2010

[P28] Edyta Brzychczy, Piotr Lipinski, Radoslaw Zimroz i Patryk Filipiak. "Artificial Immune Systems for Data Classification in Planetary Gearboxes Condition Monitoring". W: *Advances in Condition Monitoring of Machinery in Non-Stationary Operations*. Lecture Notes in Mechanical Engineering. Springer, 2014, s. 235–247

PL

5.5. Pozostałe prace naukowo-badawcze

Część moich prac badawczych dotyczyła także perceptronowych sieci neuronowych i ich zastosowań w systemach wspomagania decyzji, sieci samoorganizujących się Kohonena i samoorganizujących się sieci mieszanin (ang. Self-Organizing Mixture Networks, SOMN) [30], a także systemów inteligentnych agentów.

Brałem udział, jako ekspert zewnętrzny, w projekcie europejskim LOGICAL, którego celem było m.in. opracowanie platformy programowej opartej na koncepcji systemów wieloagentowych, realizowanej w chmurze obliczeniowej, służącej do optymalizacji procesów logistycznych związanych z transportem lądowym, powietrznym i wodnym. Moje prace dotyczyły z jednej strony zastosowania algorytmów ewolucyjnych do optymalizacji problemów transportowych (m.in. optymalizacja kosztów łańcucha dostaw w logistyce wielomodalnej), a z drugiej strony opracowania koncepcji architektury rozwiązania w chmurze obliczeniowej, w technologii VMware, umożliwiającej optymalizację procesów logistycznych.

Brałem udział, jako wykonawca części zadań, w projekcie AUDIOSCOPE dotyczącym przetwarzania języka naturalnego i rozpoznawania mowy w nagraniach dźwiękowych. Moje prace dotyczyły agregacji wyników ekspertów decyzyjnych w systemie rozpoznawania mowy przez rozwiązywanie problemu optymalizacji za pomocą algorytmów ewolucyjnych.

6. Omówienie osiągnięć dydaktycznych i organizacyjnych

Od 2004 roku pracuję w **Instytucie Informatyki Uniwersytetu Wrocławskiego** i zajmuję się działalnością dydaktyczną związaną głównie z moimi zainteresowaniami naukowym. Od 2012 roku jestem **kierownikiem Pracowni Inteligencji Obliczeniowej**, która zajmuje się pracami dydaktycznymi i naukowo-badawczymi w tematyce algorytmów ewolucyjnych, eksploracji danych, sieci neuronowych, uczenia maszynowego i przetwarzania języka naturalnego.

6.1. Opracowane i prowadzone zajęcia dydaktyczne

Od 2004 roku prowadzę autorski **wykład z algorytmów ewolucyjnych**. Początkowo wykład opierał się w dużej mierze na książce J. Arabasa *Wykłady z algorytmów ewolucyjnych* rozszerzonej o kilka publikacji naukowych dotyczących m.in. prostych algorytmów estymowania rozkładu (CGA, PBIL, UMDA). W kolejnych latach do wykładu wprowadzałem zmiany dyktowane przez rozwój algorytmów ewolucyjnych rozszerzając go o bardziej złożone algorytmy estymowania rozkładu (BOA, ECGA), strategie ewolucyjne z adaptacją macierzy kowariancji (CMA-ES), optymalizację wielomodalną, optymalizację wielokryterialną (NSGA-II, MOEA/D) i optymalizację dynamiczną.

Od 2012 roku prowadzę autorski **wykład z eksploracji danych**. Początkowo wykład zawierał wybrany subiektywnie przeze mnie zakres tematów dotyczących grupowania danych (algorytm k-means), klasyfikacji danych (drzewa klasyfikujące, CART, ID3), reguł asocjacyjnych, redukcji wymiarowości (PCA), systemów rekomendujących i zastosowań wymienionych algorytmów. W kolejnych latach wykład został uporządkowany i znacznie rozszerzony, m.in. o inne algorytmy grupowania danych (grupowanie hierarchiczne, BIRCH, DBScan, EM, PAM, CLARA, CLARANS), inne algorytmy klasyfikacji danych (KNN, C4.5), reguły asocjacyjne dla sekwencji (algorytm PrefixSpan), nieliniową redukcję wymiarowości (KPCA, LLE, t-SNE), statystykę obliczeniową i wizualizację danych oraz zastosowania m.in. w klasyfikacji obrazów. Wykład nie porusza tematyki sieci neuronowych, bo im jest poświęcony odrębny wykład w Instytucie Informatyki Uniwersytetu Wrocławskiego.

PC

W latach 2004 – 2015 opracowałem i prowadziłem też wykłady z **systemów inteligentnych agentów** (częściowo na podstawie książki M. Wooldridge'a *An Introduction to Multi Agent Systems*), **hurtowni danych i data-mining** (na podstawie wybranych materiałów naukowych i technicznych, w tym związanych z Oracle Database i Oracle Advanced Analytics oraz Microsoft SQL Server Business Intelligence) oraz **Estimation of Distribution Algorithms** (na podstawie wybranych prac naukowych), a także seminaria dotyczące podobnych zagadnień, ale ze względu na ograniczenia czasowe i popularność zajęć z algorytmów ewolucyjnych i eksploracji danych, wykłady te są prowadzone nieregularnie, zazwyczaj co kilka lat.

6.2. Prowadzone prace magisterskie

Byłem **promotorem około 20 prac magisterskich**, dotyczących głównie moich zainteresowań naukowych, w Instytucie Informatyki Uniwersytetu Wrocławskiego, m.in.

- Ciągi częste i sekwencyjne reguły asocjacyjne w analizie ruchu internautów, Jakub Petrykowski, 2006,
- Evolutionary Algorithms in Image Retrieval, Marcin Brodziak, 2006,
- Ewolucyjny system wieloagentowy, Izabela Adamczyk, 2007,
- Content-based Text Messages Clustering, Anna Molęda, 2007,
- Portal Informacji Finansowej stworzony w srodowisku Java EE, Piotr Baraniak, 2008,
- Optymalizacja programów sygnalizacji świetlnej, Krzysztof Sroka, 2009,
- System analizy rozgrywki pokerowej i modelu przeciwnika, Jakub Straszewski, 2009,
- Algorytmy ewolucyjne w prognozowaniu giełdy energii elektrycznej, Tomasz Kozakiewicz, 2009,
- Prognozowanie natężenia ruchu drogowego, Ilona Komor, 2010,
- Optymalizacja połączeń drogowych, Rafał Hładyszowski, 2010,
- Efektywna implementacja modelu Penny, Jakub Mateusz Kowalski, (współpromotor: prof. Stanisław Cebrat), 2011,
- Outliers detection in practice: Case study of internal company data mining, Michał Faleński, 2011,
- System rekomendacji dokumentów tekstowych oparty o ideę wielu zainteresowań, Zbigniew Czapran, 2012,
- Wykorzystanie danych z sieci społecznościowych w algorytmach wyszukiwania informacji, Donata Małecka, 2012,
- Programowanie genetyczne w predykcji cen energii elektrycznej, Marcin Kolarczyk, 2013,
- GGP with Advanced Reasoning and Board Knowledge Discovery, Adrian Łańcucki, 2013,
- Optymalizacja wielokryterialna w klasyfikacji wielospektralnych zdjęć satelitarnych, Łukasz Szota, 2016,
- Zagadnienie Zero-Shot Learning na przykładzie klasyfikacji danych Animals with Attributes, Piotr Rybak, 2016.

6.3. Prowadzone prace doktorskie

Byłem lub jestem **opiekunem naukowym** uczestników studiów doktoranckich z informatyki w Instytucie Informatyki Uniwersytetu Wrocławskiego:

- mgr. Marcina Brodziaka, w latach 2006 – 2007 (zrezygnował ze studiów doktoranckich ze względu na pracę zawodową),
- mgr. Błażeja Chęcińskiego, w latach 2011 – 2013 (zrezygnował ze studiów doktoranckich ze względu na pracę zawodową),
- mgr. Patryka Filipiaka, w latach 2010 – 2014 (obronił pracę doktorską w 2016 roku),
- mgr. Adriana Łańcuckiego, od 2012 roku do dzisiaj (otworzył przewód doktorski w 2016 roku).

Byłem lub jestem **promotorem pomocniczym** w dwóch przewodach doktorskich prowadzonych na Wydziale Matematyki i Informatyki Uniwersytetu Wrocławskiego:

- mgr. Patryka Filipiaka (praca zatytułowana *Proactive Evolutionary Algorithms for Dynamic Optimization Problems* została obroniona w grudniu 2016),
- mgr. Adriana Łańcuckiego (praca roboczo zatytułowana *Neighbor Embedding in Feature Selection and Multipoint Extensions* jest jeszcze w trakcie redakcji, powinna zostać złożona w 2017 roku).

6.4. Recenzje prac doktorskich

W 2016 zostałem poproszony przez Uniwersytet w Kalkucie o przygotowanie recenzji pracy doktorskiej Sri Udit Kumar Chakraborty zatutułowanej *Intelligent Textual Learner-Computer Interaction for e-Learning Systems* (przewód doktorski jest w toku).

6.5. Recenzje artykułów czasopismowych i konferencyjnych

Byłem lub jestem członkiem komitetów programowych konferencji poświęconych algorytmom ewolucyjnym, m.in.

- EvoStar: EvoFIN i EvoBAFIN w latach 2008–2017,
- CEC (Congress of Evolutionary Computation) w latach 2009–2011,
- GECCO (Genetic and Evolutionary Computation Conference) w latach 2010–2015,
- ICEC (International Conference on Evolutionary Computation) w 2010 roku,
- ECTA (International Conference on Evolutionary Computation Theory and Applications) w latach 2011–2016.

Recenzowałem artykuły dla czasopism naukowych, m.in.

- Information Technology and Management, Springer, 2010, recenzja 1 artykułu,
- Neural Network World, 2013, recenzja 1 artykułu,
- Computational Methods in Science and Technology, 2013, recenzja 1 artykułu,

PL

- Applied Soft Computing, Elsevier, 2013-2015, recenzje 3 artykułów,
- IEEE Transactions on Systems, Man and Cybernetics: Systems, IEEE, 2015, recenzja 1 artykułu,
- IEEE Access, IEEE, 2016, recenzja 1 artykułu,
- Transactions on Computational Collective Intelligence, Springer, 2016, recenzja 1 artykułu,
- Evolutionary Intelligence, Springer, 2016, recenzja 1 artykułu,
- Genetic Programming and Evolvable Machines, Springer, 2016, recenzja 1 artykułu.

6.6. Inne

W 2010 otrzymałem trzyletnie **stypendium Ministra Nauki i Szkolnictwa Wyższego dla wybitnych młodych uczonych.**

W 2013 roku byłem opiekunem doktoranta Uniwersytetu Wileńskiego, pana Andiusa Paukste, podczas jego pobytu w Instytucie Informatyki Uniwersytetu Wrocławskiego.

W latach 2014–2015 byłem opiekunem dr. Indrajit Saha realizującego swój staż podoktorski w kierowanej przeze mnie Pracowni Inteligencji Obliczeniowe Instytutu Informatyki Uniwersytetu Wrocławskiego (stypendysta ERCIM).

W latach 2012 i 2013 brałem udział w organizowaniu **Dni Małych Firm** i **Dni Innowacyjnych Firm** w Instytucie Informatyki Uniwersytetu Wrocławskiego.

W latach 2014, 2015 i 2016 byłem organizatorem (wspólnie z pracownikami i doktorantami kierowanej przeze mnie Pracowni Inteligencji Obliczeniowej Instytutu Informatyki Uniwersytetu Wrocławskiego) pokazów i wykładów na **Dolnośląskim Festiwalu Nauki.**

W 2009 roku recenzowałem projekt w POIG dla Ministerstwa Nauki i Szkolnictwa Wyższego.

W latach 2010–2011 recenzowałem kilka projektów w programie *Przedsiębiorczy doktorant - inwestycja w innowacyjny rozwój regionu* w ramach POKL dla Urzędu Marszałkowskiego Województwa Dolnośląskiego.

W 2013 roku recenzowałem projekt Sonata dla Narodowego Centrum Nauki.

W latach 2014-2015 brałem udział w realizowanym na Uniwersytecie Wrocławskim projekcie *Międzynarodowy program studiów informatyki i matematyki stosowanej dla biznesu na Uniwersytecie Wrocławskim* w ramach POKL. Brałem udział w przygotowywaniu propozycji projektu, a później uczestniczyłem w **tworzeniu specjalności *Inteligentne przetwarzanie danych* (IPD)** na stacjonarnych studiach drugiego stopnia na kierunku Informatyka na Wydziale Matematyki i Informatyki Uniwersytetu Wrocławskiego oraz w **tworzeniu *Pracowni eksploracji danych i systemów wspomagania decyzji*** dla Interdyscyplinarnych Środowiskowych Studiów Doktoranckich (IŚSD) na Wydziale Matematyki i Informatyki Uniwersytetu Wrocławskiego.

PL

7. Perspektywy

Obecnie kontynuuję badania nad algorytmami ewolucyjnymi dla wysokowymiarowych problemów optymalizacji opartymi na redukcji wymiarowości i manifold learning, opisane w rozdziale 4.3. Pracuję nad połączeniem algorytmów ewolucyjnych z podejściem zaproponowanym w t-distributed Stochastic Neighborhood Embedding (t-SNE) [15] i jego rozszerzeniach. Próbuję opracować metodę przeniesienia aktualnej populacji do zredukowanej przestrzeni poszukiwań w taki sposób, aby zachować zależności statystyczne między rozwiązaniami, m.in. zgodność warunkowych rozkładów prawdopodobieństwa opisujących przynależność punktu danych do sąsiedztw najbliższych sąsiadów w oryginalnej i zredukowanej przestrzeni przeszukiwań (podobnie jak w t-SNE).

Prowadzę też dalsze badania nad pozyskiwaniem wiedzy z finansowych szeregów czasowych ultra-wysokiej częstotliwości, opisane w rozdziale 5.1. Wspólnie z A. Brabazonem i P. Hamillem prowadzimy dalsze prace dotyczące charakteryzowania stanów rynku finansowego przez obrazy kolejki zleceń. Istotne wydaje mi się opracowanie dobrej reprezentacji obrazów kolejki zleceń, na przykład przez rozwinięcie podejścia z reprezentowaniem ich przez mieszaniny rozkładów gaussowskich, a następnie dalsze pracowanie nad grupowaniem tak reprezentowanych danych i wyznaczaniem wzorców charakterystycznych. Artykuły opisujące część uzyskanych wyników zostały przyjęte na konferencję CEC 2017 i GECCO 2017.

Wspólnie z byłym doktorantem, dr. Patrykiem Filipiakiem, pracuję także nad zastosowaniem ewolucyjnych algorytmów optymalizacji dynamicznej do problemów optymalizacji portfela instrumentów finansowych z aktualizacją ultra-wysokiej częstotliwości. Wymaga to z jednej strony redefiniowania klasycznego problemu optymalizacji portfela przez wprowadzenie stosownych dla danych tak wysokiej częstotliwości miar ryzyka, a z drugiej strony opracowania wielokryterialnych algorytmów optymalizacji dynamicznej. Artykuł opisujący pierwsze uzyskane rezultaty został przyjęty na konferencję EvoStar 2017.

Literatura

[1] P. Larranaga i J. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation.* Kluwer Academic Publishers, 2001.

[2] G. Harik. *Linkage Learning via Probabilistic Modeling in the ECGA.* University of Illinois at Urbana-Champaign, 1999.

[3] M. Pelikan, D. Goldberg i E. Cantu-Paz. "BOA: The Bayesian Optimization Algorithm". W: *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation.* 1999, s. 525–532.

[4] S. Baluja. *Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning.* Carnegie Mellon University, 1994.

[5] I. Jolliffe. *Principal Component Analysis.* Springer, 1986.

[6] S. Roweis i L. Saul. "Nonlinear Dimensionality Reduction by Locally Linear Embedding". W: *Science* 290.5500 (2000), s. 2323–2326.

[7] B. Scholkopf, A. Smola i K. Muller. "Nonlinear Component Analysis As a Kernel Eigenvalue Problem". W: *Neural Computation* 10.5 (1998), s. 1299–1319.

[8] G. Harik, F. Lobo i D. Goldberg. "The Compact Genetic Algorithm". W: *IEEE Transactions on Evolutionary Computation* 3.4 (1999), s. 287–297.

PC

[9] H. Muhlenbein. "The Equation for Response to Selection and Its Use for Prediction". W: *Evolutionary Compututation* 5.3 (1997), s. 303–346.

[10] H. Handa i O. Katai. "Estimation of Bayesian network algorithm with GA searching for better network structure". W: *Proceedings of the International Conference on Neural Networks and Signal Processing*. 2003, s. 436–439.

[11] W. Sharpe. "Capital Asset Prices: A Theory of Market Equilibrium under Conditions of Risk". W: *The Journal of Finance* 19.3 (1964), s. 425–442.

[12] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[13] J. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, 1992.

[14] G. Hinton i S. Roweis. "Stochastic Neighbor Embedding". W: *Advances in Neural Information Processing Systems*. 2003, s. 857–864.

[15] L. van der Maaten i G. Hinton. "Visualizing Data using t-SNE". W: *Journal of Machine Learning Research* 9 (2008), s. 2579–2605.

[16] H.-P. Schwefel. *Evolution and Optimum Seeking: The Sixth Generation*. John Wiley & Sons, 1993.

[17] S. Das i P. Suganthan. "Differential Evolution: A Survey of the State-of-the-Art". W: *IEEE Transactions on Evolutionary Computation* 15.1 (2011), s. 4–31.

[18] D. Whitley, S. Rana, J. Dzubera i K. Mathias. "Evaluating evolutionary algorithms". W: *Artificial Intelligence* 85.1 (1996), s. 245–276.

[19] W. Pierskalla. *The multi-dimensional Assigment and Quadratic Assignment Problems*. Technical Memorandum No. 93. Operations Research Department, CASE Institute of Technology, 1967.

[20] W. Pierskalla. "The Multidimensional Assignment Problem". W: *Operations Research* 16.2 (1968), s. 422–431.

[21] M. Guignard, P. Hahn, Z. Ding, B.-J. Kim, H. Samra, T. Stutzle i S. Kanthak. "Hybrid ARQ Symbol Mapping in Digital Wireless Communication Systems Based on the Quadratic 3-dimensional Assignment Problem (Q3AP)". W: *Proceedings of 2005 NSF DMII Grantees Conference*. 2005.

[22] L. Loukil, M. Mehdi, N. Melab, E. Talbi i P. Bouvry. "A Parallel Hybrid Genetic Algorithm-simulated Annealing for Solving Q3AP on Computational Grid". W: *Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing*. 2009, s. 1–8.

[23] C. Nugent, T. Vollmann i J. Ruml. "An Experimental Comparison of Techniques for the Assignment of Facilities to Locations". W: *Operations Research* 16.1 (1968), s. 150–173.

[24] P. Hahn, B.-J. Kim, T. Stutzle, S. Kanthak, W. Hightower, H. Samra, Z. Ding i M. Guignard. "The quadratic three-dimensional assignment problem: Exact and approximate solution methods". W: *European Journal of Operational Research* 184.2 (2008), s. 416 428.

[25] R. Engle, M. Fleming, E. Ghysels i G. Nguyen. *Liquidity and Volatility in the US Treasury Market: Evidence From A New Class of Dynamic Order Book Models*. Federal Reserve Bank of New York Working Paper. 2011.

PL

[26] M. O'Hara. *Market Microstructure Theory.* John Wiley & Sons, 1995.

[27] C. Goodhart i M. O'Hara. "High Frequency Data in Financial Markets: Issues and Applications". W: *Journal of Empirical Finance* 4.2 (1997), s. 73–114.

[28] T. Kohonen. *Self-Organizing Maps.* Springer, 2001.

[29] M. Hearst, S. Dumais, E. Osuna, J. Platt i B. Scholkopf. "Support Vector Machines". W: *IEEE Intelligent Systems and their Applications* 13.4 (1998), s. 18–28.

[30] H. Yin i N. Allinson. "Self-organizing Mixture Networks for Probability Density Estimation". W: *IEEE Transactions on Neural Networks* 12.2 (2001), s. 405–411.

8. Lista publikacji

[P1] Piotr Lipinski. "Dependency Mining in Large Sets of Stock Market Trading Rules". W: *Enhanced Methods in Computer Security, Biometric and Artificial Intelligence Systems.* Kluwer Academic Publishers, 2005, s. 329–336.

[P2] Piotr Lipinski i Jerzy Korczak. "Early Warning in On-line Stock Trading System". W: *Proceedings of the 5th International Conference on Intelligent Systems Design and Applications, ISDA 2005.* IEEE, 2005, s. 538–543.

[P3] Anna Bartkowiak i Piotr Lipinski. "Remarks on Evaluation of Correlation Dimension for 5 French Stock Data". W: *Proceedings of the 7th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2005.* IEEE, 2005.

[P4] Piotr Lipinski. "Clustering of Large Number of Stock Market Trading Rules". W: *Neural Network World.* T. 15. 4. 2005, s. 351–357.

[P5] Krzysztof Michalak i Piotr Lipinski. "Prediction of High Increases in Stock Prices using Neural Networks". W: *Neural Network World.* T. 15. 4. 2005, s. 359–366.

[P6] Piotr Lipinski, Katarzyna Winczura i Joanna Wojcik. "Building Risk-optimal Portfolio using Evolutionary Strategies". W: *Applications of Evolutionary Computation, EvoWorkshops 2007.* T. 4448. Lecture Notes in Computer Science. Springer, 2007, s. 208–217.

[P7] Piotr Lipinski. "Discovering Stock Market Trading Rules using Multi-Layer Perceptrons". W: *Computational and Ambient Intelligence, IWANN 2007.* T. 4507. Lecture Notes in Computer Science. Springer, 2007, s. 1114–1121.

[P8] Piotr Lipinski. "ECGA vs. BOA in Discovering Stock Market Trading Experts". W: *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO 2007.* ACM, 2007, s. 531–538.

[P9] Piotr Lipinski. "Evolutionary Strategies for Building Risk-Optimal Portfolios". W: *Natural Computing in Computational Finance.* T. 100. Studies in Computational Intelligence. Springer, 2008, s. 53–65.

[P10] Jerzy Korczak i Piotr Lipinski. "Systemy agentowe we wspomaganiu decyzji na rynku papierów wartościowych". W: *Rozwój informatycznych systemów wieloagentowych w środowiskach społeczno-gospodarczych.* Wydawnictwo PLACET, 2008, s. 289–301.

[P11] Piotr Lipinski. "Evolutionary Decision Support System for Stock Market Trading". W: *Artificial Intelligence: Methodology, Systems, and Applications, AIMSA 2008.* T. 5253. Lecture Notes in Computer Science. Springer, 2008, s. 405–409.

[P12] Piotr Lipinski. "Neuro-evolutionary Decision Support System for Financial Time Series Analysis". W: *Hybrid Artificial Intelligence Systems, HAIS 2008.* T. 5271. Lecture Notes in Computer Science. Springer, 2008, s. 180–187.

[P13] Piotr Lipinski. "Knowledge Patterns in Evolutionary Decision Support Systems for Financial Time Series Analysis". W: *Applications of Evolutionary Computing, EvoWorkshops 2009.* T. 5484. Lecture Notes in Computer Science. Springer, 2009, s. 203–212.

[P14] Piotr Lipinski. "Frequent Knowledge Patterns in Evolutionary Decision Support Systems for Financial Time Series Analysis". W: *Natural Computing in Computational Finance.* T. 293. Studies in Computational Intelligence. Springer, 2010, s. 131–145.

[P15] Piotr Lipinski, Edyta Brzychczy i Radoslaw Zimroz. "Planetary Gearboxes Condition Monitoring under Non-stationary Operations by Adaptive Decision Trees based Classification of Vibration Data in Multidimensional Symptom Space". W: *Proceedings of 6th International Conference Mechatronic Systems and Materials, Opole, Poland.* 2010.

[P16] Piotr Lipinski. "A Hybrid Evolutionary Algorithm to Quadratic Three-Dimensional Assignment Problem with Local Search for Many-Core Graphics Processors". W: *Intelligent Data Engineering and Automated Learning, IDEAL 2010.* T. 6283. Lecture Notes in Computer Science. Springer, 2010, s. 344–351.

[P17] Piotr Lipinski. "Evolution Strategies for Objective Functions with Locally Correlated Variables". W: *Intelligent Data Engineering and Automated Learning, IDEAL 2010.* T. 6283. Lecture Notes in Computer Science. Springer, 2010, s. 352–359.

[P18] Piotr Lipinski. "Signal Classification with Self-Organizing Mixture Networks". W: *Artificial Intelligence: Methodology, Systems, and Applications, AIMSA 2010.* T. 6304. Lecture Notes in Computer Science. Springer, 2010, s. 275–276.

[P19] Piotr Lipinski. "A Stock Market Decision Support System with a Hybrid Evolutionary Algorithm for Many-Core Graphics Processors". W: *Euro-Par 2010 Parallel Processing Workshops.* T. 6586. Lecture Notes in Computer Science. Springer, 2011, s. 455–462.

[P20] Patryk Filipiak, Krzysztof Michalak i Piotr Lipinski. "Infeasibility Driven Evolutionary Algorithm with ARIMA based Prediction Mechanism". W: *Intelligent Data Engineering and Automated Learning, IDEAL 2011.* T. 6936. Lecture Notes in Computer Science. Springer, 2011, s. 345–352.

[P21] Piotr Lipinski. "Parallel Evolutionary Algorithms for Stock Market Trading Rule Selection on Many-Core Graphics Processors". W: *Natural Computing in Computational Finance.* T. 380. Studies in Computational Intelligence. Springer, 2012, s. 79–92.

[P22] Patryk Filipiak, Krzysztof Michalak i Piotr Lipinski. "A Predictive Evolutionary Algorithm for Dynamic Constrained Inverse Kinematics Problems". W: *Hybrid Artificial Intelligence Systems, HAIS 2012.* T. 7208. Lecture Notes in Computer Science. Springer, 2012, s. 610 621.

[P23] Patryk Filipiak i Piotr Lipinski. "Parallel CHC Algorithm for Solving Dynamic Traveling Salesman Problem Using Many-Core GPU". W: *Artificial Intelligence: Methodology, Systems and Applications, AIMSA 2012.* T. 7557. Lecture Notes in Computer Science. Springer, 2012, s. 305–314.

PL

[P24] Krzysztof Michalak, Patryk Filipiak i Piotr Lipinski. "Evolutionary Approach to Multiobjective Optimization of Portfolios that Reflect the Behaviour of Investment Funds". W: *Artificial Intelligence: Methodology, Systems and Applications, AIMSA 2012*. T. 7557. Lecture Notes in Computer Science. Springer, 2012, s. 202–211.

[P25] Krzysztof Michalak, Patryk Filipiak i Piotr Lipinski. "Usage Patterns of Trading Rules in Stock Market Trading Strategies Optimized with Evolutionary Methods". W: *Applications of Evolutionary Computation, EvoWorkshops 2013*. T. 7835. Lecture Notes in Computer Science. Springer, 2013, s. 234–243.

[P26] Edyta Brzychczy i Piotr Lipinski. "Knowledge-based Modeling and Multi-objective Optimization of Production in Underground Coal Mines". W: *AGH Journal of Mining and Geoengineering*. T. 37. 1. 2013, s. 13–26.

[P27] Jerzy Korczak i Piotr Lipinski. "Cloud architecture for logistic services". W: *Polish Journal of Management Studies*. T. 8. 2013, s. 132–140.

[P28] Edyta Brzychczy, Piotr Lipinski, Radoslaw Zimroz i Patryk Filipiak. "Artificial Immune Systems for Data Classification in Planetary Gearboxes Condition Monitoring". W: *Advances in Condition Monitoring of Machinery in Non-Stationary Operations*. Lecture Notes in Mechanical Engineering. Springer, 2014, s. 235–247.

[P29] Piotr Lipinski. "Training Complex Decision Support Systems with Differential Evolution Enhanced by Locally Linear Embedding". W: *Applications of Evolutionary Computation, EvoWorkshops 2014*. T. 8602. Lecture Notes in Computer Science. Springer, 2014, s. 125–137.

[P30] Piotr Lipinski i Anthony Brabazon. "Pattern Mining in Ultra-High Frequency Order Books with Self-Organizing Maps". W: *Applications of Evolutionary Computation, EvoWorkshops 2014*. T. 8602. Lecture Notes in Computer Science. Springer, 2014, s. 288–298.

[P31] Patryk Filipiak i Piotr Lipinski. "Infeasibility Driven Evolutionary Algorithm with Feed-Forward Prediction Strategy for Dynamic Constrained Optimization Problems". W: *Applications of Evolutionary Computation, EvoWorkshops 2014*. T. 8602. Lecture Notes in Computer Science. Springer, 2014, s. 817–828.

[P32] Piotr Lipinski. "Optimizing Objective Functions with Non-Linearly Correlated Variables Using Evolution Strategies with Kernel-Based Dimensionality Reduction". W: *Hybrid Artificial Intelligence Systems, HAIS 2014*. T. 8480. Lecture Notes in Computer Science. Springer, 2014, s. 342–353.

[P33] Krzysztof Michalak, Patryk Filipiak i Piotr Lipinski. "Multiobjective Dynamic Constrained Evolutionary Algorithm for Control of a Multi-segment Articulated Manipulator". W: *Intelligent Data Engineering and Automated Learning, IDEAL 2014*. T. 8669. Lecture Notes in Computer Science. Springer, 2014, s. 199–206.

[P34] Patryk Filipiak i Piotr Lipinski. "Univariate Marginal Distribution Algorithm with Markov Chain Predictor in Continuous Dynamic Environments". W: *Intelligent Data Engineering and Automated Learning, IDEAL 2014*. T. 8669. Lecture Notes in Computer Science. Springer, 2014, s. 404–411.

PL

[P35] Adrian Lancucki, Jan Chorowski, Krzysztof Michalak, Patryk Filipiak i Piotr Lipinski. "Continuous Population-Based Incremental Learning with Mixture Probability Modeling for Dynamic Optimization Problems". W: *Intelligent Data Engineering and Automated Learning, IDEAL 2014*. T. 8669. Lecture Notes in Computer Science. Springer, 2014, s. 457–464.

[P36] Piotr Lipinski. "Training Financial Decision Support Systems with Thousands of Decision Rules Using Differential Evolution with Embedded Dimensionality Reduction". W: *Applications of Evolutionary Computation, EvoWorkshops 2015*. T. 9028. Lecture Notes in Computer Science. Springer, 2015, s. 289–301.

[P37] Patryk Filipiak i Piotr Lipinski. "Making IDEA-ARIMA Efficient in Dynamic Constrained Optimization Problems". W: *Applications of Evolutionary Computation, EvoWorkshops 2015*. T. 9028. Lecture Notes in Computer Science. Springer, 2015, s. 882–893.

[P38] Adrian Lancucki, Indrajit Saha i Piotr Lipinski. "A New Evolutionary Gene Selection Technique". W: *Proceedings of IEEE Congress of Evolutionary Computation, CEC 2015*. IEEE, 2015, s. 1612–1619.

[P39] Patryk Filipiak, Krzysztof Michalak i Piotr Lipinski. "Infeasibility Driven Evolutionary Algorithm with the Anticipation Mechanism for the Reaching Goal in Dynamic Constrained Inverse Kinematics". W: *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO 2015*. ACM, 2015, s. 1389–1390.

[P40] Piotr Lipinski, Krzysztof Michalak i Lancucki Adrian. "Improving Classification of Patterns in Ultra-High Frequency Time Series with Evolutionary Algorithms". W: *Proceedings of the Companion Publication of the 2016 Annual Conference on Genetic and Evolutionary Computation, GECCO 2016*. ACM, 2016, s. 127–128.

[P41] Krzysztof Michalak, Adrian Lancucki i Piotr Lipinski. "Multiobjective Optimization of Frequent Pattern Models in Ultra-High Frequency Time Series: Stability versus Universality". W: *Proceedings of IEEE Congress of Evolutionary Computation, CEC 2016*. IEEE, 2016, s. 3491–3498.

[P42] Adrian Lancucki, Indrajit Saha, S Bhowmick, U Maulik i Piotr Lipinski. "A New Evolutionary MicroRNA Marker Selection using Next-Generation Sequencing Data". W: *Proceedings of IEEE Congress of Evolutionary Computation, CEC 2016*. IEEE, 2016, s. 2752–2759.

[P43] Anthony Brabazon, Piotr Lipinski i Phillip Hamill. "Characterising Order Book Evolution Using Self-Organising Maps". W: *Evolutionary Intelligence*. T. 9. 4. Springer, 2016, s. 167–179.

[P44] Patryk Filipiak i Piotr Lipinski. "Dynamic Portfolio Optimization in Ultra-High Frequency Environment". W: *Applications of Evolutionary Computation*. T. 10199. Lecture Notes in Computer Science. Springer, 2017.