Metaopymalizacja algorytmów wielopopulacyjnych - Raport

Filip Katulski

I gratefully acknowledge Poland's high-performance computing infrastructure PLGrid (HPC Centers: ACK Cyfronet AGH) for providing computer facilities and support within computational grant no. PLG/2023/016123

Etap 1: State of the Art - obecny stan wiedzy

Poniższe artykuły i prace stanowiły źródło informacji dla tego projektu i używane w nich definicje i pojęcia służą za źródło informacji.

1. *What Can We Learn from Multi-Objective Meta-Optimization of Evolutionary Algorithms in Continuous Domains?* - Roberto Ugolotti, Laura Sani and Stefano Cagnoni

Wnioski:

- Algorytmy Ewolucyjne (EA) w swoich podstawowych ustawieniach mogą znajdować optima lokalne, jednak dopiero tuning parametrów może przybliżyć je do optimum globalnego.
- W artykule poruszono zagadnienia odnośnie dobierania parametrów Algorytmów Ewolucyjnych takie jak: natura problemu, jego ograniczenia, czas obliczeniowy.
- Użyty framework: EMOPaT - *Evolutionary Multi-Objective Parameter Tuning.*
- W pracy przedstawiono kilka wersji problemu badawczego w następujących wariantach:
    - Multi-Objective Single-Function Optimization Under Different Constraints - optymalizacja pojedynczej funkcji w ramach różnych warunków ograniczających

    - Multi-Function Optimization - wszystkie funkcje opisane w problemie badawczym są optymalizowane razem.

2. *Meta-optimization of multi-objective population-based algorithms using multi-objective performance metrics* - Krystian Łapa

Wnioski:

- W niniejszym artykule wybrano algorytm genetyczny (GA) do optymalizacji parametrów rozważanego algorytmu wielokryterialnego (metaoptymalizacja)
- Algorytmy populacyjne (population-based algorithms) to algorytmy oparte o populację rozwiązań, zwanych indywiduami (individuals). Mogą (lecz nie muszą) być oparte o działanie populacji prawdziwych, występujących w naturze i środowisku ludzkim.
- Artykuł opisuje nową metodę optymalizacji parametrów algorytmów populacyjnych. W zaproponowanej metodzie parametry algorytmu są wyznaczane przez funkcje zmieniające się od iteracji (iteration-dependent function). Funkcje te zwracają różne wartości parametrów w zależności od iteracji algorytmu i mogą mieć inny kształt zdefiniowany przez ich parametry.
- Optymalizować należy nie tylko parametry całkowite (np. liczba osobników w populacji), ale także parametry wartości rzeczywistej (np. parametry operatorów).

3. *Meta-optimization of massively concurrent optimization algorithms using Elixir technology* - Grzegorz Wcisło, praca dyplomowa

Wnioski:

- Metaoptymalizacja to proces wykorzystywania algorytmu optymalizacji do dostrojenia parametrów innego algorytmu optymalizacji.
- Istniejące metody metaoptymalizacji, np.: CALIBRA*, CMA-ES, BOBYQA, MADS, ParamILS, Irace.

- MEOW jako workbench do uruchamiania algorytmów genetycznych, napisany w Elixirze, LEAP oraz DEAP napisane w Pythonie oraz EASEA napisany w C++ mogą posłużyć do uruchamiania algorytmów genetycznych, których parametry mają być docelowo zoptymalizowane.

4. *A Platform for Testing Multi-Population Evolutionary Algorithms using The BEAM Virtual Machine* - Jonatan Kłosko, Mateusz Benecki

Wnioski:

- Większość frameworków łączy to, że operacje ewolucyjne są stosowane oddzielnie do każdej osoby lub małej grupy osób. Takie podejście ma istotne ograniczenie - uniemożliwia zastosowanie bardziej wydajnej reprezentacji populacji i zastosowanie operacji na populacji w jednym kroku.
- Meow jest deklaratywny - algorytm można opisać w formie serii operacji jaką przechodzi populacja - pipeline.
- Meow może być uruchamiany przy pomocy CPU, GPU, Klastrze

Etap 2 - Analiza i wybór narzędzi

1. Frameworki ewolucyjne, wielopopulacyjne:

LEAP - Najbardziej popularny framework ewolucyjny, ogólne przeznaczenie.
DEAP - W przeciwieństwie do LEAP, algorytmy nie są predefiniowane, należy zaprojektować samemu, wykorzystując framework.

2. Frameworki optymalizacyjne:

Optuna - Framework optymalizacyjny ogólnego zastosowania, wykorzystywany głownie do ML, jednak ma implementację “Nondominated Sorting Genetic Algorythm II” jako algorytm próbkujący (czyli optymalizujący). Dodatkowo ma wbudowany moduł wizualizacji.

pymoo - wieloobiektowy framework optymalizacyjny, zawiera różnorodne algorytmy ewolucyjne (tzw. constrained single-, multi-, and many-objective optimization algorithms).

3. Źródła informacji:

DEAP:
- https://deap.readthedocs.io/en/master/overview.html

LEAP:
- https://leap-gmu.readthedocs.io/en/latest/index.html

Optuna:
- https://optuna.readthedocs.io/en/stable/index.html
- https://www.analyticsvidhya.com/blog/2021/09/optimize-your-optimizations-using-optuna/

pymoo:
- https://www.researchgate.net/publication/340976217_Pymoo_Multi-Objective_Optimization_in_Python
- https://pymoo.org/index.html

4. Dodatkowe frameworki warte uwagi:
- PyGMO: https://esa.github.io/pygmo2
- jMetalPy: https://jmetal.github.io/jMetalPy/index.html
- Platypus: https://platypus.readthedocs.io/en/latest/experimenter.html

Etap 3 - określone cele i zakres prac, uruchomione narzędzia

1. Cel pracy - Wykorzystanie metod metaoptymalizacji algorytmów wielopopulacyjnych w celu uzyskania lepszych wyników w Problemach ciągłych oraz dyskretnych
2. Zakres prac:
    - Budowa modeli wybranych problemów dyskretnych i ciągłych.
    - Uruchomienie rozwiązania - algorytmu wielopopulacyjnego (np. wyspowego) dla wybranych problemów.
    - Uruchomienie iRace w celu optymalizacji parametrów rozwiązania.

3. Uruchomione narzędzia:
    - DEAP - instalacja, uruchomienie problemu plecaka (ang. “Knapsack”) jako przykładowy model problemu dyskretnego.
    - LEAP - instalacja, uruchomienie problemu Schwefela jako przykładowy problem model problemu ciągłego. Porównanie wyników z wynikami Grzegorza Wcisły.
    - Instalacja oraz uruchomienie iRace, analiza pracy dyplomowej Grzegorza Wcisły i jego metod inicjalizacji parametrów iRace.

Etap 4: prototyp, działające pierwsze elementy

1. Uruchomiony prototyp za pomocą LEAP_EC
2. Pierwsza interpretacja problemu Shekela za pomocą skryptu w języku Python
3. Uruchomienie skryptu lokalne na dwurdzeniowym procesorze

Etap 5: prototyp: działa większość funkcji, pierwsze wyniki

W celu prezentacji metaoptymalizacji algorytmów wybrany został problem Shekela, zwany również problemem lisich dziur (ang. foxholes).
Osiągnięte cele:

- za pomocą LEAP uruchomiłem problem Shekela dla kilku różnych konfiguracji wysp i kilku innych wybranych parametrów algorytmu ewolucyjnego.
- Uruchomienie iRace w celu metaoptymalizacji algorytmów wielopopulacyjnych, poszukiwanie najbardziej optymalnych parametrów algorytmu oraz układu wysp

Parametry algorytmu:

Parameter	Zakres/rodzaj	Wartości	Opis
population_size	int	(5,25)	Rozmiar populacji na pojedynczej wyspie
selection	categorical	tournament, sus, roulette	wybór sposobu selekcji osobników w procesie ewolucyjnym
migration_selection	categorical	tournament, sus, roulette	wybór sposobu selekcji populacji do migracji
populations	int	(100, 200)	rozmiar całkowitej populacji dla wszystkich wysp
topology	categorical	fully_connected, ring, mesh, star	wybór topologii wysp

Konfiguracja testu w programie iRace:

- maksymalna ilość eksperymentów: 1500
- dwa równoległe wątki z włączonym Load Balancerem
- pojedyncza runda zajmuje około 4 godziny

iRace mierzy wybrany element jako koszt i próbuje zbliżyć go do możliwie najmniejszego wyniku. Kosztem jest średnia wartość funkcji fitness, im mniejsza tym bardizej dopasowane rozwiązanie.

Cztery najlepsze wyniki:

population_size	populations	selection	migration_selection	topology
13	186	tournament	tournament	fully_connected
13	190	tournament	sus	fully_connected
14	180	tournament	tournament	ring
9	137	tournament	tournament	mesh

Etap 6: RC: osiągnięte cele, wyniki testów, analiza wyników

1. Sposób przeprowadzania eksperymentu

W ramach zmodyfikowanego planu działania zastosowano następujące ustawienia parametrów algorytmu rozwiązującego problem Shekela:

Parametr	**Zakres/rodzaj**	**Wartości**	**Opis**
archipelago_size	ordinal (integer)	(2,3,5,10,20,40,60,80, 100)	Rozmiar archipelagu, czyli ilości populacji
populations	integer	(200 : 500)	Łączna ilość osobników na archipelagu
selection	categorical	tournament, sus, roulette	wybór sposobu selekcji osobników w procesie ewolucyjnym
migration_selection	categorical	tournament, sus, roulette	wybór sposobu selekcji populacji do migracji
topology	categorical	fully_connected, ring, mesh, star	wybór topologii wysp

W ramach tego eksperymentu wykorzystano zasoby HPC należące do ACK Cyfronet ARES. Algorytm został uruchomiony za pomocą narzędzia SLURM na 20 węzłach obliczeniowych po 4 rdzenie każdy.

W celu zapewnienia stałej ilości generacji, ich liczba obliczana jest jako ułamek:

int(populations / archipelago_size)

W ten sposób liczba ewolucyjnych zmian w całkowitej populacji jest stała dla danej całkowitej ilości osobników. Dzięki temu większe zbiory nie są premiowane za swój rozmiar.

W ramach metaoptymalizacji algorytmu użyto programu irace, z następującymi ustawieniami scenariusza optymalizacji (niewymienione parametry mają wartość domyślną):

Parametr	**Wartość**	**Opis**
targetRunnerRetries	2	Ilość ponownych prób uruchomienia pojedynczego wywołania skryptu
parallel	4	ilość równoległych uruchomień skryptu
maxExperiments	15000	maksymalna liczba uruchomień skryptu
loadBalancing	“1” / True	Uruchamia Load Balancer dla równoległych uruchomień skryptu

2. Wyniki testów

W ramach testów każdy z 4 najlepszych wyników dla wszystkich 20 iteracji testu dla parametru *selection* wskazywał *tournament* jako najlepszy, pomijam go w tabeli.

Wyniki testów przedstawiają się następująco:

archipelago_size	populations	topology	migration_selection
2	495	mesh	sus
2	434	star	tournament
2	399	mesh	tournament
2	314	mesh	tournament
2	272	star	roulette
2	236	ring	sus
3	464	fully_connected	tournament
3	394	mesh	tournament
3	438	star	tournament
3	365	mesh	tournament
3	322	star	roulette
3	293	ring	sus
3	352	star	sus
5	316	fully_connected	sus
5	463	fully_connected	tournament

5	398	ring	sus
5	312	ring	tournament
5	285	mesh	roulette
5	268	fully_connected	sus
5	242	fully_connected	sus
5	317	ring	roulette
5	437	ring	roulette
5	439	ring	sus
5	394	fully_connected	sus
5	477	star	tournament
10	442	mesh	sus
10	234	ring	sus
10	282	fully_connected	sus
10	440	fully_connected	sus
10	459	fully_connected	tournament
10	468	ring	sus
10	408	star	roulette
10	471	ring	roulette
10	397	star	roulette
10	388	star	sus
10	387	fully_connected	sus
10	206	fully_connected	tournament
10	410	mesh	sus
20	314	ring	tournament
20	500	star	sus
20	416	ring	sus
20	485	ring	sus
20	452	star	roulette
20	447	fully_connected	sus
20	393	star	roulette
20	355	star	roulette
20	349	star	sus
20	256	fully_connected	tournament
20	216	ring	roulette
20	259	ring	roulette
20	465	star	roulette
20	289	ring	roulette

40	316	fully_connected	sus
40	310	fully_connected	sus
40	478	star	sus
40	290	ring	tournament
40	481	fully_connected	tournament
40	499	fully_connected	tournament
40	223	star	sus
60	373	ring	sus
60	431	star	roulette
60	363	fully_connected	sus
60	395	star	tournament
60	266	ring	roulette
60	359	star	sus
80	351	fully_connected	tournament
80	435	star	sus
80	429	star	tournament
80	336	star	tournament
80	459	star	tournament
80	249	ring	roulette
100	250	star	tournament
100	283	star	roulette
100	276	star	roulette
100	354	ring	sus
100	364	ring	sus
100	334	star	tournament
100	392	fully_connected	tournament
100	389	ring	roulette
100	489	ring	tournament

3. Analiza wyników

- W każdej z wykonanych iteracji testowych liczba wysp dla najlepszych wyników nie skłaniała się ku żadnej konkretnej wartości.
- Jako wartość dla *selection* irace wskazał *tournament* jako najbardziej optymalny.
- Topologia *star* jest najbardziej powszechna, 28 na 80 przypadków testowych.
- dla *migration_selection* parametr *sus* jest najczęściej wskazywany

4. Osiągnięte cele
    - Uruchomienie Algorytmu dla problemu Foxhole wraz z programem irace na zasobach Cyfronetu.
    - Zebranie wyników z 20 iteracji testowych, poszerzonych o zwiększoną liczbę wykonań eksperymentu.
    - Wstępna analiza wyników, odrzucenie mniej istotnych parametrów.
5. Propozycje rozwoju
    - Dalsze próby metaoptymalizacji przy dostosowanych parametrach
    - Rozszerzona analiza
    - Uruchomienie najlepszych kombinacji dla zwiększonej ilości generacji i sprawdzenie wyników