ELECTRIC FISH OPTIMIZATION: A NEW HEURISTIC ALGORITHM BASED ON ELECTROLOCATION

ELEKTRİK BALIĞI OPTİMİZASYONU: ELEKTROLOKASYON TABANLI YENİ BİR SEZGİSEL ALGORİTMA

SELİM YILMAZ

ASSOC. PROF. DR. SEVİL ŞEN AKAGÜNDÜZ

Supervisor

Submitted to

Graduate School of Science and Engineering of Hacettepe University

as a Partial Fulfillment to the Requirements

for the Award of the Degree of Doctor of Philosophy

in Computer Engineering

January 2020

To my beloved wife, sweet daughter, and family…

ETHICS

In this thesis study, prepared in accordance with the spelling rules of Institute of Graduate Studies in Science of Hacettepe University,

I declare that

- all the information and documents have been obtained in the base of the academic rules.
- all audio-visual and written information and results have been presented according to the rules of scientific ethics
- in case of using others works, related studies have been cited in accordance with the scientific standards
- all cited studies have been fully referenced
- I did not do any distortion in the data set
- and any part of this thesis has not been presented as another thesis study at this or any other university.

20/01/2020

SELİM YILMAZ

YAYINLANMA FİKRİ MÜLKİYET HAKLARI BEYANI

Enstitü tarafından onaylanan lisansüstü tezimin/raporumun tamamını veya herhangi bir kısmını, basılı (kağıt) ve elektronik formatta arşivleme ve aşağıda verilen koşullarla kullanıma açma iznini Hacettepe üniversitesine verdiğimi bildiririm. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet haklarım bende kalacak, tezimin tamamının ya da bir bölümünün gelecekteki çalışmalarda (makale, kitap, lisans ve patent vb.) kullanım hakları bana ait olacaktır.

Tezin kendi orijinal çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanması zorunlu metinlerin yazılı izin alarak kullandığımı ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederim.

Yükseköğretim Kurulu tarafından yayınlanan ***"Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge"*** kapsamında tezim aşağıda belirtilen koşullar haricince YÖK Ulusal Tez Merkezi / H. Ü. Kütüphaneleri Açık Erişim Sisteminde erişime açılır.

- ☐ Enstitü / Fakülte yönetim kurulu kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren 2 yıl ertelenmiştir.
- ☐ Enstitü / Fakülte yönetim kurulu gerekçeli kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren .... ay ertelenmiştir.
- ☐ Tezim ile ilgili gizlilik kararı verilmiştir.

20/01/2020

SELİM YILMAZ

ABSTRACT

ELECTRIC FISH OPTIMIZATION: A NEW HEURISTIC ALGORITHM BASED ON ELECTROLOCATION

Selim YILMAZ

Doctor of Philosophy, **Computer Engineering Department**

Supervisor: Assoc. Prof. Dr. Sevil ŞEN AKAGÜNDÜZ

January 2020, 238 pages

Swarm behaviors in nature have inspired the emergence of many heuristic optimization algorithms. These algorithms have attracted much attention, particularly for solving complex problems, owing to their characteristics of supporting high dimensional data, non-differentiate functions, and the like. In this thesis, a new heuristic algorithm inspired by communication of electric fish and their way of finding their prey location is proposed.

Nocturnal electric fish have very poor eyesight and live in muddy, murky water, where visual senses is very limited. Therefore, they rely on their species-specific ability called *electrolocation* to perceive their environment. The active and passive electrolocation capability of such fish is believed to be a good candidate for balancing local and global search, and hence it is modeled in this study.

A new heuristic algorithm called *Electric Fish Optimization* (EFO) is introduced for solving single-objective optimization problems and compared with both six well-known heuristics (Simulated Annealing, Vortex Search, Genetic Algorithm, Differential Evolution, Particle Swarm Optimization, and Artificial Bee Colony) and popular methods in the literature. In the

experiments, a well-known selection of bound-constrained 50 basic and 30 complex mathematical functions, unconstrained 13 clustering and classification problems, and bound- and design-constrained 5 real-world problems have been used as the benchmark sets. The simulation results indicate that EFO is better than or very competitive with its competitors.

Furthermore, the single-objective EFO algorithm has been extended in order to handle multi-objective optimization problems. While the search behaviour of the single-objective EFO is preserved, few modifications are applied in order to adapt the algorithm for solving multi-objective optimization problems. Three types of multi-objective EFO algorithms are proposed, namely Pareto-Dominance based Electric Fish Optimization (PD-EFO), Non dominated Sorting-based Electric Fish Optimization (NS-EFO), and Cellular-based Electric Fish Optimization (CB-EFO). All these multi-objective algorithms have been tested on 30 multi-objective benchmark problems that are commonly used in the literature. The obtained optimization performances have been evaluated through the most popular convergence metrics. Findings from these metrics reveal that CB-EFO algorithm has shown the best performance compared to the other multi-objective EFO algorithms and that NS-EFO has shown a performance that is the most competitive to CB-EFO. CB-EFO algorithm has then been compared to the well-known multi-objective optimization algorithms in the literature using the same problem set. Experimental findings have shown that CB-EFO performs a performance superior than the competitors. To sum up, in this thesis, a new swarm-based algorithm which is shown to be effective for both single- and multi-objective optimization problems, is introduced to the community.

Keywords: Nature-inspired algorithm, Heuristic algorithms, Population-based algorithms, Swarm intelligence, Electric Fish, Electrolocation, Electrocommunication, Real parameter optimization, Single-objective optimization, Multi-objective optimization.

ÖZET

ELEKTRİK BALIĞI OPTİMİZASYONU: ELEKTROLOKASYON TABANLI YENİ BİR SEZGİSEL ALGORİTMA

Selim YILMAZ

Doktora, Bilgisayar Mühendisliği

Danışman: Doç. Dr. Sevil ŞEN AKAGÜNDÜZ

Ocak 2020, 238 sayfa

Doğada gözlemlenen sürü davranışları, birçok sezgisel optimizasyon algoritmasının ortaya çıkmasına neden olmuştur. Yüksek boyutlu veriyi, türevlenemez fonksiyonları desteklemesi ve benzeri karakteristikleri ile bu algoritmalar özellikle karmaşık problemlerin çözümü için oldukça dikkat çekmektedirler. Bu tez çalışmasında elektrikli balıkların iletişiminden ve av yerlerini bulma davranışlarından ilham alan yeni bir sezgisel algoritma önerilmiştir.

Geceleri oldukça hareketli yaşayan elektrik balıkları çok zayıf görme yeteneğine sahiptir ve görsel duyuların çok sınırlı olduğu çamurlu, bulanık suda yaşamaktadırlar. Bu nedenle, çevrelerini algılamak için *elektrolokasyon* adı verilen türlerine özgü yeteneklerine güvenmektedirler. Bu türdeki balıkların sergilediği aktif ve pasif elektrolokasyon yeteneğinin, yerel ve küresel aramanın dengelenmesi için iyi bir davranış olduğuna inanılmaktadır ve balıkların bu yeteneği bu çalışmada modellenmiştir.

Bu tez çalışmasında, *Elektrik Balığı Optimizasyonu* (EFO) adı verilen, tek-amaçlı optimizasyon problemlerini çözmek için yeni bir sezgisel yöntem tanıtılmış ve literatürdeki iyi bilinen altı sezgisel algoritma (Benzetilmiş Tavlama, Girdap Arama, Genetik Algoritma, Diferansiyel Evrim, Parçacık Sürüsü Optimizasyonu, ve Yapay Arı Kolonisi) ve mevcut birçok

yöntem ile karşılaştırılmıştır. Kıyaslama kümesi olarak, parametre uzayı sınırlandırılmış 50 temel ve 30 karmaşık matematiksel fonksiyon, parametre uzayı sınırlandırılmamış 13 adet kümeleme ve sınıflandırma problemi, parametre uzayı ve tasarımı sınırlandırılmış 5 adet gerçek dünya problemini içeren bir küme deneylerde kullanılmıştır. Deney sonuçları, EFO'nun rakiplerine göre daha iyi ya da oldukça rekabetçi bir algoritma olduğunu göstermiştir.

Çok-amaçlı problemleri de ele almak amacıyla önerilen tek-amaçlı EFO algoritması bu çalışmada genişletilmiştir. Tek-amaçlı EFO algoritmasındaki arama davranışı genel olarak korunurken, algoritmaya bazı modifikasyonlar yapılarak algoritma çok-amaçlı problemlerin çözümü için uyarlanmıştır. Üç farklı çok amaçlı optimizasyon algoritması önerilmiştir: Pareto-baskın Tabanlı EFO (PD-EFO), Bastırılamayan-sıralı Tabanlı EFO (NS-EFO) ve Hucresel Tabanlı EFO (CB-EFO). Bu algorithmalar, literatürde sıklıkla kullanılan 30 adet çok-amaçlı optimizasyon problemi üzerinde test edilmiştir. Elde edilen optimizasyon performansları bazı yakınsama metriklerine göre değerlendirilmiştir. Bu metriklerden elde edilen bulgular; CB-EFO algoritmasının diğerlerine göre en iyi başarımı sergilediğini, bu başarıma en çok NS-EFO algoritmasının yaklaştığını göstermiştir. CB-EFO algoritması yine aynı problem kümesi üzerinde literatürde iyi bilinen çok-amaçlı optimizasyon algoritmaları ile karşılaştırılmış ve deneysel bulgular, CB-EFO'nun rakip yöntemlere göre çok daha iyi bir performans sergilediğini göstermiştir. Sonuç olarak, bu tez çalışmasında, hem tek-amaçlı hem de çok-amaçlı optimizasyon problemleri etkin bir şekilde çözen yeni bir sürü tabanlı optimizasyon algorithması literatüre katılmıştır.

Anahtar Kelimeler: Doğadan esinlenen algoritmalar, Sezgisel algoritmalar, Popülasyon-tabanlı algoritmalar, Sürü zekası, Elektrikli balıklar, Elektrolokasyon, Elektroiletişim, Gerçek parametre optimizasyonu, Tek-amaçlı optimizasyon, Çok-amaçlı optimizasyon.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank to my supervisor Assoc. Prof. Dr. Sevil ŞEN for her valuable advice, guidance, and encouragement. She patiently supported me with her knowledge and experiences throughout the preparation of this thesis.

Besides, I would like to thank to my thesis committee members, Prof. Dr. Bahriye AKAY, Prof. Dr. Ecir Uğur KÜÇÜKSİLLE, Asst. Prof. Dr. Burcu CAN BUĞLALILAR, and Asst. Prof. Dr. Ceren TUNCER ŞAKAR for reviewing this thesis and giving their insightful comments. I wish to show my gratitude to the current and former administrative board as well as the academic personnel at the Department of Computer Engineering at Muğla Sıtkı Koçman University for letting me to study at Hacettepe University.

I would like to thank my friends Emre AYDOĞAN, Dr. Levent KARACAN, Dr. Cemil ZALLUHOĞLU, all the members of *Wireless Networks and Intelligent Secure Systems Laboratory* (WISE Lab.) of Hacettepe University, and all the research assistants I worked with at the Department of Computer Engineering at Hacettepe University.

I wish to express my huge gratitude to my mother Mukadder and my deceased father Avni for their self devotion, love, and giving me opportunities that have made me who I am; my brother Fatih Mehmet and my sister Halise Merve for their love and support that I feel throughout my life.

I would like to pay my special regards to my beloved wife Duygu for continually encouraging me during my Ph.D. and for her supports that I feel every time I need during my academic life. I would also like to express my thank to my cute little daughter Simay who always cheers me up. I am very lucky to be with both of you.

Finally, I would like to thank Scientific and Technological Research Council of Turkey (or TUBITAK) for supporting this thesis by the National M.Sc and Ph.D Scholarship Programme for Senior Undergraduate Students (2228).

GENİŞLETİLMİŞ ÖZET

Optimizasyon, belirli sayıda kısıtlamaya sahip bir sorun için sayısız çözüm seti arasında mümkün olan en iyi veya en azından kabul edilebilir çözümleri bulma sürecidir [1]. Çok kiplilik, yüksek boyutluluk, türevlenemezlik, doğrusal olmama gibi karmaşık sorunların özellikleri, bu problemlerin klasik algoritmalar kullanılarak çözülmesini imkansız hale getirmektedir. NFL [2] teoremine göre tüm sorunlara en iyi çözümü sağlayan tek bir yöntem bulunmamaktadır. Dolayısıyla, bir soruna en uygun çözümü garanti etmeyen ancak kabul edilebilir bir süre içinde kabul edilebilir çözümler bulan sezgisel algoritmalara olan ilgi artmaktadır. Bu algoritmalar literatürde görüntü işleme [3, 4], veri madenciliği [5], rotalama [6] ve konuşma tanıma [7] gibi birçok mühendislik problemine uygulanmıştır.

Sezgisel yöntemlerde iki önemli arama bileşeni vardır: *Keşif* ve *sömürü*. Keşifsel veya küresel arama, arama ajanlarını mümkün olduğunca çeşitli tutarak arama alanının bilinmeyen bölgelerinde arama yapmaya yönlendiren bir sezgisel tarama yeteneğidir. Diğer taraftan sömürü, en iyi ajanların bilgilerini kullanarak uzayda arama yapmaya odaklanan bir yetenektir. Sezgisel yöntemlerin başarısı büyük ölçüde bu bileşenlerin ne kadar iyi dengelendiğine bağlıdır [1, 8]. Yerel optimumda takılı kalmamak için ilk iterasyonlarda keşif kullanmak ve iterasyonlar ilerledikçe keşifsel aramayı azaltıp algoritmanın yerel arama yapmasına izin vermek kabul edilen genel bir kuraldır [9, 10].

Üstsezgilerin büyük çoğunluğu doğadaki biyolojik veya fiziksel sistemlerin davranışlarından esinlenerek türetilmiştir. Literatürde, doğal olayları modelleyen, doğadan esinlenen algoritmalar olarak adlandırılan, birçok popüler sezgisel algoritma vardır. Örneğin, karıncalar tarafından bırakılan feromon, Karınca Kolonisi Optimizasyonu algoritmasının temelidir, benzer şekilde arıların kuyruk dansı davranışı Yapay Arı Kolonisi Algoritmasının temelini oluşturmaktadır. Bu çalışmada, gerçek parametreli optimizasyon problemlerini çözmek için elektrikli balıkların *elektrolokasyon* ve *elektro-iletişim* özelliklerinden esinlenilen Elektrik Balığı Optimizasyonu adlı yeni bir popülasyon tabanlı sezgisel algoritma önerilmiştir. Elektrolokasyon, elektrikli balıkların etrafındaki nesneleri/yiyecekleri algılamalarını sağlayan türe özgü bir yetenektir. Elektrikli balıkların elektrolokasyon kabiliyeti iki ana dalda incelenir:

aktif elektrolokasyon ve *pasif elektrolokasyon*. Bu yetenekler, önerilen EFO algoritmasında yerel ve küresel arama yeteneklerini yerine getirmektedir. Elektro-iletişim, elektrik balıklarının elektrik sinyalleri yoluyla kurduğu türe özgü iletişim şeklidir. Bu çalışmada, önerilen algoritmanın diğer buluşsal yöntemlere göre performansını araştırmak için dört farklı deney gerçekleştirdik. İlk deneyde, 50 temel ve 30 karmaşık matematiksel problemi içeren (*CEC'17 Ölçüt Kümesi* [11] olarak da bilinir) bir kıyaslama seti kullanılmıştır. Önerilen EFO algoritması ile rakip buluşsal yöntemler arasındaki istatistiksel farklılıkları değerlendirmek için Wilcoxon İmzalı Sıralama Testi, VIKOR ve TOPSIS gibi çok kriterli karar verme yöntemleri uygulanmıştır. İkinci deneyde, önerilen EFO algoritmasının performansı, önemli makine öğrenmesi işlemlerinden olan kümeleme ve sınıflandırma problemleri üzerinde analiz edilmiştir. Kümeleme yöntemleri görüntü işleme, jeofizik, tarıma, güvenlik ve suç tespitine kadar birçok araştırma alanında yaygın olarak kullanılmaktadır. Bu deneyde, gerçek verilerden elde edilen 13 kümeleme veri kümesi test amacıyla kullanılmıştır. Bunun yanısıra bu deneyde; literatürde son zamanlarda oldukça önem kazanmış *bot ağ akışlarının tespiti* üzerinde EFO algoritması bir sınıflandırma algoritması olarak incelenmiş ve literatürdeki mevcut yöntemler ile kıyaslanarak algoritmanın performansı ölçülmüştür. Son deney, bilinen beş adet gerçek-hayat tasarım probleminden oluşmaktadır. Bunlar; *kaynaklı kiriş*, *yay tasarımı*, *basınçlı kap*, *üç çubuklu kafes tasarımı* ve *hız düşürücü* problemleri olarak adlandırılmaktadır. Üçüncü deneyin asıl amacı, EFO'nun kısıtlanmış problemler üzerindeki performansını gözlemlemektir. Önerilen algoritma, deneylerde iyi bilinen sezgisel yöntemlerle karşılaştırılmıştır. Tüm deneylerden elde edilen sonuçlar; EFO'nun küresel ve yerel aramalar arasında ince bir denge sağladığını, "önce keşfet, sonra sömür" yaklaşımını iyi bir biçimde sürdürdüğünü ve EFO'nun özellikle karmaşık tasarım problemlerinde rakiplerinden çok daha iyi veya rekabetçi performans gösterdiğini göstermektedir ki bu da çoğu gerçek dünya problemi için hayatidir. Performansı, farklı türdeki problemler üzerinde derinlemesine incelenerek, doğadan ilham alan yeni bir tek-amaçlı optimizasyon algoritması bu tez çalışmasında önerilmiştir.

Buna ek olarak, tek-amaçlı problemler üzerindeki başarısından ilham alınarak, EFO algoritması, çok-amaçlı optimizasyon problemlerini ele almak amacıyla genişletilmiş ve çok-amaçlı EFO algoritmasının üç farklı uygulaması önerilmiştir. Bu da tez çalışmasının ikinci aşamasını oluşturmaktadır. Önerilen çok-amaçlı EFO algoritmaları, tez çalışmasının birinci aşamasında önerilen tek-amaçlı EFO algoritmasına dayanmaktadır. Dolayısıyla tek-amaçlı EFO algoritmasının arama davranışının temeli çok değiştirilmeden çok-amaçlı problemlere uyarlanmış, algoritmanın arama performansını bu türdeki karmaşık problemlerde iyileştirecek bazı değişiklikler yapılmıştır. Böylece; Pareto-baskın Tabanlı, Bastırılamayan-sıralı Tabanlı, ve Hücresel Tabanlı Elektrik Balığı Optimizasyon algoritmaları çok-amaçlı optimizasyon algoritmaları olarak literatüre önerilmiştir.

Önerilen algoritmalar, literatürde sıklıkla kullanılan DTLZ [12], LZ09 [13], WFG [14], ve ZDT [15] ailelerinden 30 adet ölçüt problem üzerinde test edilmiş ve istatistiksel başarıları çok-amaçlı problemlerde algoritmaların yakınsama performansını ölçen *hiper-hacim*, *yayılma*, *epsilon* ve *ters nesilsel mesafe* ölçütlerine göre araştırılmıştır. Elde edilen bulgular, CB-EFO algoritmasının ortalamada diğerlerine göre açık bir şekilde daha iyi performans sergilediğini gözler önüne sermiştir. Literatüdeki başarısını tespit etmek için, CB-EFO algoritması aynı problem kümesi üzerinde ve aynı ölçüt kriterleri kullanılarak literatürde çok bilinen; Hızlı-bastılamayan Sıralama Genetik Algoritması (NSGA-II) [16], Kuvvet Pareto Evrimsel Algoritması (SPEA2) [17], İndikatör Tabanlı Evrimsel Algoritma (IBEA) [18], Ayrıştırma Tabanlı Çok-amaçlı Evrimsel Algoritma (MOEAD) [19] ve Çok-amaçlı Hücresel Genetik Algoritma (MOCell) [20] ile karşılaştırılmıştır. Bulgular CB-EFO'nun, *hiper-hacim*, *epsilon* ve *ters nesilsel mesafe* ölçütlerine göre rakiplerinden daha iyi, *yayılma* metriğine göre olduçka rekabetçi bir optimizasyon performansı sergilediğini kanıtlamıştır. Böylece çok-amaçlı CB-EFO algoritması, var olan metotlara güçlü bir alternatif algoritma olarak literatüre önerilmiştir.

Önerilen çok amaçlı EFO algoritmalarından; PD-EFO, tek amaçlı EFO algoritmasında önerilen arama operatörleri üzerinde bir değişiklik önermemektedir, ancak çok amaçlı optimizasyon problemlerine uygulanabilir hale getirmek için PD-EFO, çözümlerin amaç uzayında ne kadar yakınsama sağladığını gözler önüne seren *Pareto baskınlık* bilgisini kullanmaktadır. PD-EFO için yapılan modifikasyonlar; bireylerin arama rollerinin belirlenmesini,

yeni aday bireyin seçilmesini ve baskın olmayan çözümlerin farklı bir kümede depolanmasını kapsamaktadır. PD-EFO'da olduğu gibi, NS-EFO da tek amaçlı EFO algoritmasının arama operatörleri üzerinde bir değişiklik yapmamaktadır. Ayrıca NS-EFO, baskın olmayan arama bireylerini saklamak için PD-EFO'da kullanılan Pareto-baskınlık stratejisini kullanmaktadır. Bireylerin arama rollerini belirlemek ve popülasyona dahil edilecek yeni aday bireyleri seçmek için NS-EFO'da kullanılan strateji PD-EFO'dan farklıdır. NS-EFO, bu görevleri yerine getirmek için [16, 21]'de önerilen *Pareto front* ve *kalabalık mesafesi* kavramlarını kullanmaktadır. PD-EFO'nun aksine, NS-EFO'da baskın olmayan çözümleri başka bir kümede saklamak için bir strateji kullanılmamıştır. Bununla birlikte, CB-EFO, çok amaçlı optimizasyon problemlerinde optimizasyon performansını artırmak amacıyla tek amaçlı EFO algoritmasında önerilen arama operatörlerinde birkaç değişiklik yapmaktadır. Buna ek olarak, EFO'daki zaman karmaşıklığını azaltmak için CB-EFO'daki komşuluk yapısı [22, 23]'de önerilen bir kafes topolojisi kullanılarak soyut bir şekilde temsil edilmektedir. NS-EFO'da olduğu gibi CB-EFO da bireylerin arama rolünü belirlemek için *Pareto front* ve *kalabalık mesafesi* kavramlarını kullanmaktadır. Buna ek olarak, CB-EFO, baskın olmayan çözümleri de depolamak için PD-EFO'da kullanılan stratejiye benzer bir strateji kullanmaktadır, ancak burada depolanan baskın çözümlerin arama sürecine katkıda bulunmasına izin verilirken PD-EFO'daki çözümlere izin verilmemektedir. Önerilen çok amaçlı EFO algoritmalarının her adımı ayrıntılı bir biçimde bu tezde açıklanmıştır.

Bildiğimiz kadarıyla, bu çalışmada önerilen algoritma, karmaşık sorunları çözmek için elektrikli balıkların aktif/pasif elektrolokasyon ve elektro-iletişim yeteneklerinden esinlenen ilk algoritmadır. Yakın zamanda önerilen bir algoritma, *Balık Elektrolokasyon Optimizasyonu (FEO)* [24], *kapasitans tespiti* olarak adlandırılan, elektrikli balıkların canlı nesneleri cansız nesnelerden kapasitif ve direnç özelliklerini analiz ederek ayırt etme yeteneğini modelleyen bir algoritmadır. FEO'daki bireyler, aktif elektrolokasyon ile ölçülen nesnenin kapasitansı, eğimi ve genişliği gibi niteliklerini göz önünde bulundurmaktadırlar. FEO'nun oluşturulması için kullanılan metafor ilginç olmasına rağmen, algoritmanın tasarımında bazı kritik eksiklikler tespit edilmiştir. En dikkat çekici olanı, algoritmanın probleme bağlı olan birçok parametreye (15'ten fazla) sahip olmasıdır. Bu nedenle, parametre setindeki küçük bir değişiklik,

algoritmanın performansında büyük bir değişiklik sağlayabilmektedir. FEO'nun performans araştırması da oldukça tartışmalı bulunmuştur. Çünkü algoritma, düşük boyutlu sınırlı sayıdaki problemler üzerinde fakat yüksek boyutlu sadece bir problem üzerinde karşılaştırmalı olarak test edilmiştir. Bu da yeterli bir karşılaştırmanın yapılmadığına işaret etmektedir. Buna karşın önerilen EFO algoritması, elektrolokasyon yaparken balık ve hedef nesne arasındaki mesafeye göre değişen elektrik deşarj aktivitesi davranışını modellemektedir. Ayrıca, elektrik sinyalleri yoluyla elektrikli balıklar arasındaki iletişim, sürü tabanlı algoritmalar için oldukça önemli olan bir davranış, EFO'da da modellenmiştir. FEO'nun aksine, önerilen EFO algoritması, daha az parametre ve daha basit arama bileşenleriyle FEO'dan çok daha iyi performans göstermiştir. Ayrıca EFO, yüksek performans göstermesinin yanında, uygulaması kolay bir algoritmadır.

Literatürde, elektrik balıkların aktif elektrolokasyon yeteneğini robotik duyu sistemleri tasarlamada kullanan çalışmalar da mevcuttur. Lebastard ve ark. [25] tarafından bir sualtı robotunun bulunduğu çevrede navigasyon işlevi gerçekleştirmesi için zayıf elektrik balıklarının aktif elektrolokasyon davranışını taklit eden bir model geliştirilmiştir. Model, biri sinyal yayan ve devamı alıcılardan oluşan bir dizi küresel elektrot içermektedir. Robot çevresindeki nesne tarafından oluşturulan dağılım, modelin nesnenin konumunu ve yarıçapını tahmin etmesine yardımcı olmaktadır. MacIver ve ark. [26], zayıf bir elektrikli balık olan *hayalet balığı*'nı daha iyi gelecek nesil Otonom Sualtı Araçları (OSA) tasarlamak amacıyla bir prototip geliştirmek için incelemişlerdir. Prototip sadece aktif elektrolokasyondan değil, aynı zamanda avı algıladıktan sonra elektrikli balıkların manevra davranışından ilham almıştır. Bir diğer çalışmada, Maciver ve Nelson [27], zayıf elektrikli balıkların davranışlarından esinlenilen bir sinyal işleme algoritması geliştirmiş ve bir denizaltı robotuna elektrosensörel yetenekler sağlayan yapay bir sensör dizisi tasarlamıştır. Solberg ve ark. [28], elektrikli balıklardaki aktif elektrolokasyon kabiliyetini bir *algılama robotu* oluşturmak üzere modellemiştir. Bu robot, bir elektrik alanının konumunu kontrol etmek ve su altı nesnelerini tespit etmek için elektro-sensör verilerini sürekli olarak işlemektedir. Bu tarz tasarım çalışmaların ayrıntıları, Neveln ve ark. [29] tarafından detaylı bir şekilde incelenmiştir. Daha önce de belirtildiği gibi bu çalışmalar robotik sensör teknolojisi alanındadır ve sadece elektrikli

balıkların aktif elektrolokasyon yeteneğinden ilham almışlardır. Ayrıca, önerilen tasarım çözümleri tek bir algısal robot ajana dayandığından, bu çalışmalarda sürü zekâsına yer verilmemiştir [25–28]. Öte yandan, önerilen EFO algoritması, elektrikli balıkların hem aktif hem de pasif elektrolokasyon davranışlarından esinlenmiştir ve elektrikli balıklar arasındaki iletişimi kullanarak sürü zekası sağlamaktadır.

Önerilen EFO algoritması, yerel ve küresel arama mekanizmalarını sağlamak için elektrikli balıkların hem aktif hem de pasif elektrolokasyon davranışlarından esinlenen ilk sürü tabanlı sezgisel algoritmadır. Ek olarak, elektrikli organ deşarj aktiviteleri (frekans ve genlik), sürüye dayalı sezgisel bir algoritmada ilk kez modellenmiştir. Önerilen EFO algoritmasının arkasındaki metafor, dengeli bir yerel ve küresel aramanın yanı sıra "önce keşfet, sonra sömür" yaklaşımını desteklemektedir. Önerilen EFO algoritması, FEO'nun aksine ayarlanması gereken sadece iki parametreye sahiptir. Ek olarak, deneylerde EFO parametrelerinin, algoritmanın performansı üzerinde dramatik bir etkisi olmadığı gösterilmiştir, bu da EFO'nun parametrelerinden bağımsız sağlam bir algoritma olduğunu göstermektedir.

Bu tez çalışmasında, önerilen EFO algoritmasının performansı çeşitli problemler üzerinde kapsamlı bir şekilde değerlendirilmiştir. EFO'nun, farklı kısıtlı, kümeleme ve gerçek dünya tasarım problemleri üzerinde, literatürdeki neredeyse tüm mevcut yaklaşımlardan önemli ölçüde daha iyi performans gösterdiği gösterilmiştir.

CONTENTS

Page

TABLES

FIGURES

ABBREVIATIONS

AAA	**A**rtificial **A**lgae **A**lgorithm
ABC	**A**rtificial **B**ee **C**olony
ACO	**A**nt **C**olony **O**ptimization
AI	**A**rtificial **I**ntelligence
A-MOABC/NS	**A**synchronous **M**ulti-**O**bjective **A**rtificial **B**ee **C**olony/**N**on-dominated **S**orting
A-MOABCPD	**A**synchronous **M**ulti-**O**bjective **A**rtificial **B**ee **C**olony/**P**areto **D**ominance
ANN	**A**rtificial **N**eural **N**etwork
ANN-BP	**A**rtificial **N**eural **N**etwork with **B**ack **P**ropagation
ANN-EFO	**A**rtificial **N**eural **N**etwork with **E**lectric **F**ish **O**ptimization
BA	**B**at **A**lgorithm
BFO	**B**acterial **F**oraging **O**ptimization
BHA	**B**lack **H**ole **A**lgorithm
BNC	**B**ayesian **N**etwork **C**lassifiers
BP	**B**ack **P**ropagation
BSA	**B**acktracking **S**earch **A**lgorithm
CB-EFO	**C**ellular-**B**ased **E**lectric **F**ish **O**ptimization
CEP	**C**lassification **E**rror **P**ercentage
CNN	**C**onvolutional **N**eural **N**etwork
CS	**C**uckoo **S**earch
CSA	**C**row **S**earch **A**lgorithm
CSGA	**C**uckoo **S**earch-**G**enetic **A**lgorithm
DA	**D**ragonfly **A**lgorithm
DE	**D**ifferential **E**volution
DMCMOABC	**D**ynamic **M**ulti **C**olony **M**ulti-**O**bjective **A**rtificial **B**ee **C**olony
dMOABC	**D**ivision-based **M**ulti-**O**bjective **A**rtificial **B**ee **C**olony
DN-PSO	**D**ynamic **N**eighborhood **P**article **S**warm **O**ptimization

DoE	**D**esign **o**f **E**xperiment
DOE	**DO**lphin **E**cholocation
DSA	**D**ifferential **S**earch **A**lgorithm
DSS-MDE	**D**ynamic **S**tochastic **S**election **M**ultimember **D**ifferential **E**volution
EFO	**E**lectric **F**ish **O**ptimization
eMOABC	**E**litism-based **M**ulti-**O**bjective **A**rtificial **B**ee **C**olony
EOD	**E**lectric **O**rgan **D**ischarge
EP	**E**volutionary **P**rogramming
ES	**E**volutionary **S**trategy
FA	**F**irefly **A**lgorithm
FEN	**F**unction **E**valuation **N**umber
FEO	**F**ish **E**lectrolocation **O**ptimization
FOA	**F**ruit-fly **O**ptimization **A**lgorithm
FPA	**F**lower **P**ollination **A**lgorithm
FPR	**F**uzzy **P**attern **R**ecognition
GA	**G**enetic **A**lgorithm
GA-AIS	**G**enetic **A**lgorithm-**A**rtificial **I**mmune **S**ystem
GE	**G**rammatical **E**volution
GP	**G**enetic **P**rogramming
GSA	**G**ravitational **S**earch **A**lgorithm
GWO	**G**rey **W**olf **O**ptimization
HV	**H**yper**V**olume
IBEA	**I**ndicator-**B**ased **E**volutionary **A**lgorithm
IGD	**I**nverted **G**enerational **D**istance
ILS	**I**terated **L**ocal **S**earch
ISA	**I**nterior **S**earch **A**lgorithm
ISOT	**I**nformation **S**ecurity and **O**bject **T**echnology
IWO	**I**nvasive **W**eed **O**ptimization
KHA	**K**rill **H**erd **A**lgorithm

LSA	**L**ightning **S**earch **A**lgorithm
MAPSO	**M**ulti-**A**gent **P**article **S**warm **O**ptimization
MCDM	**M**ulti **C**riteria **D**ecision-**M**aking
MOABC	**M**ulti-**O**bjective **A**rtificial **B**ee **C**olony
MOBA	**M**ulti-**O**bjective **B**at **A**lgorithm
MOCell	**M**ulti-**O**bjective **Cell**ular-based Genetic Algorithm
MOCS	**M**ulti-**O**bjective **C**uckoo **S**earch
MOEAD	**M**ulti-**O**bjective **E**volutionary **A**lgorithm based on **D**ecomposition
MOFA	**M**ulti-**O**bjective **F**irefly **A**lgorithm
MOGSA	**M**ulti-**O**bjective **G**ravitational **S**earch **A**lgorithm
MOGWO	**M**ulti-**O**bjective **G**rey **W**olf **O**ptimizer
MOOP	**M**ulti-**O**bjective **O**ptimization **P**roblem
MOPSO	**M**ultiple **O**bjective **P**article **S**warm **O**ptimization
MOTLBO	**M**ulti-**O**bjective **T**eaching-**L**earning-**B**ased **O**ptimization
MPSO/D	**M**ultiple **O**bjective **P**article **S**warm **O**ptimization based on **D**ecomposition
MSE	**M**ean **S**quared **E**rror
NB	**N**aïve **B**ayes
NDE	**N**ovel **D**ifferential **E**volution
NFL	**N**o **F**ree **L**unch
NSABC	**N**on-dominated **S**orting-based **A**rtificial **B**ee **C**olony
NS-EFO	**N**on-dominated **S**orting-based **E**lectric **F**ish **O**ptimization
NSGA-II	**N**on-dominated **S**orting **G**enetic **A**lgorithm-**II**
PD-EFO	**P**areto-**D**ominance **E**lectric **F**ish **O**ptimization
PS	**P**attern **S**earch
PSO	**P**article **S**warm **O**ptimization
PSO-DE	**P**article **S**warm **O**ptimization-**D**ifferential **E**volution
QPSO	**Q**uantum-behaved **P**article **S**warm **O**ptimization
RFC	**R**andom **F**orest **C**lassifier
RS	**R**andom **S**earch

SA	**S**imulated **A**nnealing
SADE	**S**elf **A**daptive **D**ifferential **E**volution
SMO	**S**pider **M**onkey **O**ptimization
S-MOABC/NS	**S**ynchronous **M**ulti-**O**bjective **A**rtificial **B**ee **C**olony/**N**on-dominated **S**orting
SMPSO	**S**peed-constrained **M**ulti-objective **P**article **S**warm **O**ptimization
SOOP	**S**ingle-**O**bjective **O**ptimization **P**roblem
SPEA2	**S**trength **P**areto **E**volutionary **A**lgorithm **2**
S-PSO	**S**igma-based **P**article **S**warm **O**ptimization
STA	**S**tate **T**ransition **A**lgorithm
SVM	**S**upport **V**ector **M**achine
TCA	**T**-**C**ell **A**gorithm
TLBO	**T**eaching-**L**earning-**B**ased **O**ptimization
TOPSIS	**T**echnique for **O**rder **P**reference by **S**imilarity to an **I**deal **S**olution
TS	**T**abu **S**earch
TV-MOPSO	**T**ime-**V**ariant **M**ultiple **O**bjective **P**article **S**warm **O**ptimization
VEABC	**V**ector **E**valuated **A**rtificial **B**ee **C**olony
VEGA	**V**ector **E**valuated **G**enetic **A**lgorithm
VEPSO	**V**ector **E**valuated **P**article **S**warm **O**ptimization
VIKOR	**VI**se **K**ri-terijumska **O**ptimizacija I Kompromisno **R**esenje
VNS	**V**ariable **N**eighborhood **S**earch
VS	**V**ortex **S**earch
WOA	**W**hale **O**ptimization **A**lgorithm

1. INTRODUCTION

Optimization is the process of finding the best possible or at least admissible solutions among numerous solution sets for an optimization problem with a certain number of constraints [1]. An optimization problem[1] with a certain number of equality and inequality constraints can often be stated as follows:

$$\underset{X}{\textit{minimize}} \quad f(X)$$

$$\textit{subject to}$$

$$g_i(X) \leq 0 \; i = 1, 2, \ldots, p$$
$$h_j(X) = 0 \; j = 1, 2, \ldots, q$$

where X is an n-dimensional vector ($X = [x_1, x_2, \ldots, x_n]^T$), called *design vector*. x_k, $\{k = 1, 2, ..., n\}$ is kth parameter of the vector to be optimized. $g_i(X)$ and $h_j(X)$ indicate p inequality and q equality constraints, respectively. Among infinite number candidate solution set ($\mathcal{S}$) of the search space (Ω), optimal solution vector (X^*) is found ($X^* \subseteq \mathcal{S} \subseteq \Omega$) by an optimization algorithm.

The characteristics of complex optimization problems such as multimodality, high dimensionality, nondifferentiability, nonlinearity make it impossible to solve these problems using classical optimization algorithms. Owing to the NFL [2] theorem, which states that there is no sole method that provides the best solution to all problems, there has been growing interest in heuristic algorithms that do not guarantee an optimal solution to the problem, but find admissible solutions within an acceptable time. Such algorithms have been applied to numerous engineering problems in the literature, such as image processing [3, 4], data mining [5], routing [6], and speech recognition [7].

There are two major search components in heuristics: *Exploration* and *exploitation*. Exploration, or global search, is the ability of a heuristic that directs it to search unknown regions

[1] *minimize* should be changed to *maximize* when the goal is maximization.

of the search space by keeping the search agents as diverse as possible, whereas exploitation is the ability that focuses on searching by exploiting information of the best agents in the space. The success of a heuristic is strongly dependent on how well these components are balanced [1, 8]. It is a rule of thumb to use exploration at the beginning of the iterations to avoid trapping in *local optimum* and to fade it out by a lapse of iterations to allow an algorithm to search its locality, which hence enables the algorithm to better converge to the *global optimum* [9, 10]. In the following, the definitions of local and global optimum have been given while they have been demonstrated on a two-dimensional search space in Figure 1.1..

- **Local optimum:** A local optimum of an optimization problem is either local minimum or local maximum.
  - **Local minimum:** A local minimum $\hat{x}_l \mid \hat{x}_l \in \Omega$ of an optimization problem ($f : \Omega \to \mathbb{R}$) is a vector of points where $f(\hat{x}_l) \leq f(x)$ such that $x \in \Omega$ for all x neighboring of $\hat{x}_l$.
  - **Local maximum:** A local maximum $\hat{x}_l \mid \hat{x}_l \in \Omega$ of an optimization problem ($f : \Omega \to \mathbb{R}$) is a vector of points where $f(\hat{x}_l) \geq f(x)$ such that $x \in \Omega$ for all x neighboring of $\hat{x}_l$.
- **Global optimum:** A global optimum of an optimization problem is either global minimum or global maximum.
  - **Global minimum:** A global minimum $\hat{x} \mid \hat{x} \in \Omega$ of an optimization problem ($f : \Omega \to \mathbb{R}$) is a vector of points where $f(\hat{x}) < f(x)$ such that $\forall x \in \Omega, \hat{x} \neq x$.
  - **Global maximum:** A global maximum $\hat{x} \mid \hat{x} \in \Omega$ of an optimization problem ($f : \Omega \to \mathbb{R}$) is a vector of points where $f(\hat{x}) > f(x)$ such that $\forall x \in \Omega, \hat{x} \neq x$.

The vast majority of metaheuristics have been derived from the behavior of biologic or physical systems in nature. There have been very popular heuristic algorithms in the literature, called *Nature-Inspired Algorithms*, that model natural phenomena. For example, pheromone,

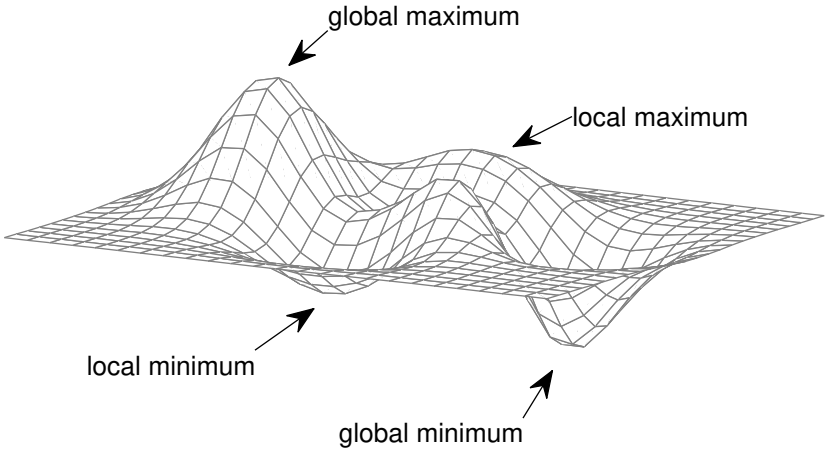

Figure 1.1. Local and global optimum demonstration on a two-dimensional search space.

deposited by ants, is the basis of ACO algorithm, and the waggle dance is the basis of ABC algorithm. In this study, a new population-based heuristic algorithm is proposed, named Electric Fish Optimization, which is inspired by the *electrolocation* and *electrocommunication* capabilities of electric fish for solving real parameter optimization problems. Electrolocation is the ability of electric fish that enables them to perceive objects/prey around them. The electrolocation capability of electric fish is examined in two major branches: *active electrolocation* and *passive electrolocation*. These capabilities inherently fulfill the exploitation and exploration characteristic of EFO. Electrocommunication is species-specific communication behavior of electric fish through electric signals. To investigate the performance of the proposed algorithm with respect to other heuristics, we have conducted three experiments. In the first experiment, 50 basic and 30 complex mathematical functions (also known as the *CEC'17 Benchmark Suite* [11]) are employed as a benchmark set. The Wilcoxon Signed Rank Test and multicriteria decision-making methods, namely VIKOR and TOPSIS, are applied to evaluate the statistical differences between the proposed EFO algorithm and the competitor heuristics. In the second experiment, the performance of the proposed EFO algorithm is analyzed as a clustering method, one of the important approaches used in data mining. Clustering methods are widely used in many research areas from image processing,

geophysics, agriculture to security and crime detection [30]. In this experiment, 13 clustering datasets, which are obtained from actual data, are used as the test bed. The final experiment comprises five well-known real-world design problems: *welded-beam*, *spring design*, *pressure vessel*, *three-bar truss design*, and *speed reducer* problems. The main purpose of the third experiment is to observe the performance of EFO on constrained problems. The proposed algorithm is compared with well-known heuristics in the experiments. The results obtained from all the experiments show that EFO ensures a fine balance between global and local searches, maintaining an 'explore first, exploit later' approach, and that EFO performs much better than or at least competitive with its competitors, especially on complex design problems employed here, which is essential for most real-world problems. Hence EFO is introduced to the community as a new nature-inspired single-objective algorithm, which constitutes the first and the main part of this thesis.

In the second part of the thesis, motivating from its performance on SOOPs, the EFO algorithm has been extended to handle MOOPs and three different multi-objective EFO algorithms have been proposed. The proposed multi-objective EFO algorithms have been built on the single-objective EFO algorithm proposed in the first part of this thesis. Therefore, the search behavior of the single-objective EFO algorithm has been kept almost unchanged. However, some schemes adopted in the multi-objective algorithms have been applied to the multi-objective EFO algorithms in order to adapt EFO for solving MOOPs and to improve its search performance on such complex problems. Hence; Pareto-dominant based Electric Fish Optimization, Non-dominated Sorting based Electric Fish Optimization, and Cellular based Electric Fish Optimization algorithms have been introduced as multi-objective optimization algorithms to the literature.

Among the proposed multi-objective EFO algorithms; PD-EFO does not change the search operators proposed in single-objective EFO algorithm but, to make it applicable to MOOPs, PD-EFO exploits *Pareto dominance* information, an indicator in multi-objective concept used to measure how well a solution could converge to the expected region in the search space, obtained from the solutions. The modifications made for PD-EFO include the determination the search roles of the individuals, the selection of new candidate individual,

and the storage of non-dominated solutions in a distinct set. like PD-EFO, NS-EFO does not change the search operators of single-objective EFO algorithm either and it employs the same strategy that is used in PD-EFO to store non-dominated search individuals. NS-EFO differs from PD-EFO in the strategy that is used to determine the search roles of the individuals and to select new candidate individuals to be included in the population. NS-EFO makes use of *Pareto front* and *crowding distance* concepts proposed in [16, 21] to perform these tasks. Contrary to PD-EFO, a strategy to store the non-dominated solutions in a set has not been introduced in NS-EFO. CB-EFO, however, makes a few modification on the search operators proposed in single-objective EFO algorithm to enhance the optimization performance on MOOPs. In addition, the concept of neighborhood in EFO has been represented in an abstract way through introduced a mesh topology which is proposed in [22, 23] to reduce the time complexity. Like NS-EFO, CB-EFO exploits the *Pareto front* and *crowding distance* concepts for determination of search role of individuals. In addition, similar to PD-EFO, CB-EFO employs a strategy to store the non-dominated solutions as well, however, the stored-solutions are allowed to contribute the search process while those in PD-EFO are not allowed. Each step of the proposed multi-objective EFO algorithms has been explained in detail in this thesis.

The proposed multi-objective optimization algorithms have been tested on 30 benchmark problems in DTLZ [12], LZ09 [13], WFG [14], and ZDT [15] families, which are frequently used in the literature. The statistical performance has been measured through *HV*, *SPREAD*, *EPSILON*, *Inverted Generational Distance – IGD*, the most popular performance metrics that are able to reveal the convergence performance of algorithms proposed for solving MOOPs. The results show that the CB-EFO algorithm performs significantly better on average than other multi-objective proposals of of EFO. Then, CB-EFO has also been compared with the well-known multi-objective algorithms in the literature, namely NSGA-II [16], SPEA2 [17], IBEA [18], MOEAD [19], MOCell [20]. The results show that CB-EFO performs better than its competitors in terms of *HV*, *EPSILON*, and *IGD* and it has shown a competitive performance with MOEAD in terms of the *SPREAD* metric. Thus, CB-EFO has been introduced as an alternative approach to the existing multi-objective optimization algorithms in

the literature.

To the best of our knowledge, the proposed algorithm in this study is the first model inspired by the active/passive electrolocation and electrocommunication capabilities of electric fish in order to solve complex problems. A recently proposed FEO algorithm [24] models *capacitance detection*, the ability of electric fish to discriminate animate objects from inanimate objects by analyzing their capacitive and resistive properties. Individuals in FEO take into account the attributes of objects such as capacitance, slope, and width, which are measured through active electrolocation. Although the metaphor utilized for the creation of FEO is interesting, there are some critical deficits in the algorithm's design choices. The most noticeable is that it possesses many parameters (more than 15), which are problem-dependent. Thus, a small change in the parameter set may yield a high variation in its performance. The performance validation is also quite questionable. Because it is comparatively tested on a limited number of low-dimensional problems and only on one high-dimensional problem, a fair comparison is missing. The proposed EFO, on the contrary, models the electric discharge activity that changes as a response to the distance between the fish and the target object while electrolocating. Moreover, the communication between electric fish through electric signals is modeled in EFO as well, which is crucial for swarm-based algorithms. The proposed EFO algorithm shows a superior performance than FEO with fewer parameters and simpler search components, which makes EFO an easy-to-implement algorithm.

There are also a few studies that exploit the active electrolocation capability of electric fish to design robotic sensory systems. A model mimicking the active electrolocation behavior of weakly electric fish was developed in [25] for the navigation around a sphere of an underwater robot. The model comprises a number of spherical electrodes, one of which emits a signal and the remaining ones are the receivers. Perturbation created by the surrounding object helps the model to estimate the position and radius of the object. In [26], the authors examined *ghost kniefish*, a weakly electric fish, to develop a prototype for the purpose of designing better next-generation autonomous underwater vehicles (AUV). The prototype was inspired not only by the active electrolocation, but also by the maneuvering behavior of electric fish once they detect prey. Another study [27] developed a signal processing algorithm

inspired by the behavior of weakly electric fish and designed an artificial sensor array that provides electrosensory capabilities to a submarine robot. In [28], the active electrolocation capability of electric fish was modeled to create a sensory robot. This robot continuously processes the electrosensory data to control the position of an electric field and to localize underwater objects. Interested readers are referred to [29] for the details of these studies. On one hand, as stated earlier, these studies are in the field of robotic sensor technology and were only inspired by the active electrolocation capability of electric fish. Moreover, because the proposed design solutions rely on a single sensory agent, swarm intelligence is not introduced in these studies [25–28]. On the other hand, the proposed EFO algorithm is inspired by both the active and passive electrolocation of electric fish, and it provides swarm intelligence by using communication among electric fish.

1.1. Contributions of the Thesis

The contributions of this thesis are summarized as follows:

- The proposed single-objective EFO algorithm is the first swarm-based heuristic algorithm inspired by both active and passive electrolocation behaviors of electric fish to ensure local and global search mechanisms.
- Electric organ discharge activities (frequency and amplitude) are modeled for the first time in a swarm-based heuristic algorithm. The metaphor behind the proposed EFO algorithm supports a well-balanced local and global search as well as an 'explore first, exploit later' approach.
- The proposed single-objective EFO algorithm has been extended in order to solve MOOPs which could have various conflicting objectives. Hence a multi-objective EFO algorithm has been introduced in this thesis. To the best of our knowledge, this is the first study that proposes an multi-objective optimization algorithm based on electrich fish behaviours.
- The performance of single- and multi-objective EFO algorithm have been extensively explored on various problems in this study. Single-objective EFO is shown to perform considerably better than the well-known approaches in the literature, which are used for comparison in this study, on a variety of problems from bound-constrained, unconstrained clustering and classification problems to bound and design-constrained real-world problems. Multi-objective EFO has also shown a superior optimization performance than the most popular multi-objective optimization algorithms in the literature.
- The proposed EFO algorithm has only two parameters that need to be tuned, as opposed to FEO, which makes the proposed algorithm much easier to use. In addition, it is shown in the experiments that the parameters of EFO do not have a dramatic effect on the performance of the algorithm, which makes EFO a robust algorithm. The proposed algorithm is shared with the community in [31].

1.2. Organization of the Thesis

The organization of this thesis is as follows:

Chapter 2. reviews single-objection and multi-objective optimization algorithms in the literature. Especially, the ones used for comparison in the experiments are given in details in this section. For the sake of a better readability, these algorithms have been independently explained under a clear-cut taxonomy. Moreover, the 'popularity' of the reviewed single- and multi-objective algorithms has been examined in this section to reveal how well they have been adopted in the literature.

Chapter 3. comprehensively introduces electric fish and their behaviors and characteristics in nature. The metaphors used in the formation of the EFO algorithm have also been discussed in this section. Then, the single-objective EFO algorithm has also been described in details this section. The time complexity of single-objective EFO algorithm with its best and worst case performance has been mathematically given in this section to reveal its efficacy. The search strategy of single-objective EFO algorithm has also been demonstrated in this section on a two-dimensional exemplar problem to reveal the effect of each search component of single-objective EFO algorithm. In addition, a comparison of the search strategies that are employed by single-objective EFO algorithm and the well-known heuristic algorithms has been given in this section.

Chapter 4. first presents the competitor heuristic algorithms that have been employed in the experiment. It then explains parameter tuning strategy used to obtain optimal parameter values of both proposed single-objective EFO algorithms and the competitor heuristic algorithms. The performance evaluation of single-objective EFO algorithm has been comparatively evaluated on a variety of problems: *i*) bound-constrained mathematical benchmark problems, *ii*) unconstrained clustering and classification problems, and *iii*) bound- and design-constrained problems.

Chapter 5. first introduces the concept of MOOPs, gives some definitions that are used by this problem domain, explains a categorization of MOOPs. The adaptations of the single-objective EFO algorithm for solving MOOPs have also been explained in this section. The search procedures used in the proposed multi-objective PD-EFO, NS-EFO, and CB-EFO algorithm have been explained in detail in this section.

Chapter 6. introduces the performance metrics and the characteristics of the problems that are used in performance assessment of proposed multi-objective algorithms. The experimental results of the multi-objective PD-EFO, NS-EFO, and CB-EFO algorithms have comparatively been given in this section. In addition, the best-performing multi-objective EFO algorithm (CB-EFO) has also evaluated through a comparison with the well-known approaches and the results obtained from this evaluation have also been discussed in this section.

Chapter 7. concludes the thesis. The benefits and downsides of the proposed EFO algorithms have been listed in this section.

2. RELATED WORK

Today, the increasing complexity of computation for solving real life problems, the challenges in processing large amount of data, and the dynamicity of problems having multiple nonlinear constraints make traditional optimization algorithms insufficient. That's why, a great deal of nature-inspired algorithms have emerged in order to solve complex optimization problems in science and engineering. The success of these algorithms in solving complex problems within an acceptable period of time has made them very attractive for researchers in recent years. Therefore, nature-inspired algorithms mimicking the physical, chemical, or biological laws in the nature or inspiring from the intelligent behavior of groups of animals such as birds, insects, and the like have been attracking researchers for a long time. Hence a great deal of nature-inspired heuristic algorithms have been proposed.

Nature-inspired algorithms have mainly been proposed in order to solve SOOPs and often been expanded for solving complex MOOPs. Single-objective heuristic algorithms can be classified in many ways in the literature, but it is the number of solutions employed that makes heuristics fundamentally distinctive in the literature [32, 33]. Based on this criterion, they are classified as *single-solution-based* and *population-based* heuristics. Population-based heuristics are further classified into evolutionary-based and swarm-based heuristics based on the mechanisms they are inspired from nature. Similarly, multi-objective heuristic algorithms are often categorised as evolutionary- and swarm-based multi-objective algorithms in the literature. In this thesis study, the most popular single- and multi-objective nature-inspired heuristic algorithms as well as their recent advancements have been reviewed according to this categorization as demonstrated in Figure 2.1..

In addition to the development of a new heuristic algorithm, there are many studies in the literature that apply such algorithms in order to solve different types of real-life problems and, that aim to enhance the performance of these algorithms. In order to reveal the impact of these studies in the literature, the number of scientific studies on nature-inspired algorithms and their applications, that are published as an article paper, a conference paper, or a

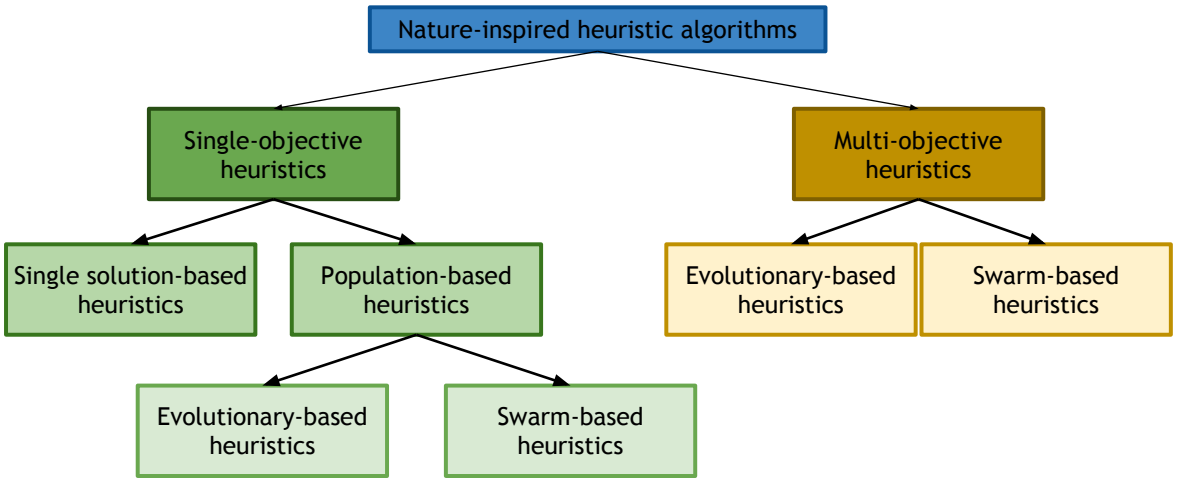

Figure 2.1. Classification of nature-inspired heuristic algorithms.

book chapter is explored. In order to achieve that, research studies on this research area are retrieved from Scopus, which is one of the most popular abstract and citation databases.

2.1. Single-objective Optimization Methods

2.1.1. Single solution-based heuristic algorithms

Single-solution-based heuristics search with a single agent and continue searching in a sequential manner. RS [34], PS [35], TS [36], ILS [37], VNS [38], SA [39], and VS [40] are well-known single solution based heuristic algorithms in the literature. These algorithms are explained briefly in this section.

RS [34] is a straightforward heuristic algorithm that relies on a simple random search strategy. The solution vector in RS is created in a random manner at the initialization phase. Then, a new position of the solution vector is generated within the given radius at every iteration and is accepted in case a better position in the search space is found. Therefore, the algorithm could get trapped in a local optimum, which is the main drawback of this approach.

PS [35] algorithm starts with an initial solution vector. However, differently from RS, the PS algorithm relies on a strategy called *exploratory move* in order to escape from a local optimum by using a pattern matrix defining the range of the search at each iteration. The

pattern matrix uses a scalar parameter that determines the step size. At each iteration, the solution vector is moved through a step size by an exploratory move and the better state is chosen; otherwise, the step size is decreased for the next iterations. PS continues to search until the step size is small enough.

TS [36] algorithm uses a different approach to prevent premature convergence problem or not to miss the global optimum. It employs a short-term memory in order to store recently visited part of search space. Hence the algorithm promotes to explore new regions in next iterations by using this memory. TS generates a number of neighbor solution vectors from an initial vector by using local or neighborhood search operator to move a solution to another solution, then keeps only the best neighbor to further exploit its neighborhood in next iterations until the termination criterion is met.

ILS [37] uses first *perturbation* on the solution vector to avoid from getting stuck in local optimum and to discover new regions in the search space, then employs *local search* to exploit the very vicinity of the perturbed vector. This approach ensures ILS to have a well-balanced exploration and exploitation capability. However, the degree of perturbation should be fine tuned. Because, the algorithm behaves like a random search as the magnitude of perturbation increases and, otherwise it behaves like a local search algorithm, which could trap in a local minimum.

VNS [38] algorithm employs a *nested neighborhood generation* approach in order to search the very distant parts of the search space, which allows the algorithm to perform a better exploration. So the kth neighbor of the solution vector is recursively derived from the $k-1$th neighbour. Then, a local search operator is applied on each generated neighbor in order to perform exploitation.

SA [39] algorithm is inspired by the annealing process of glass or metal materials. Contrary to other single solution-based approaches, SA occasionally could accept a worse solution than the current solution based on the idea that the current solution could be a local optimum. Hence by selecting a worse solution than the current one, it allows to further explore the search space. To do that, it employs a scalar parameter called *temperature* which is initially

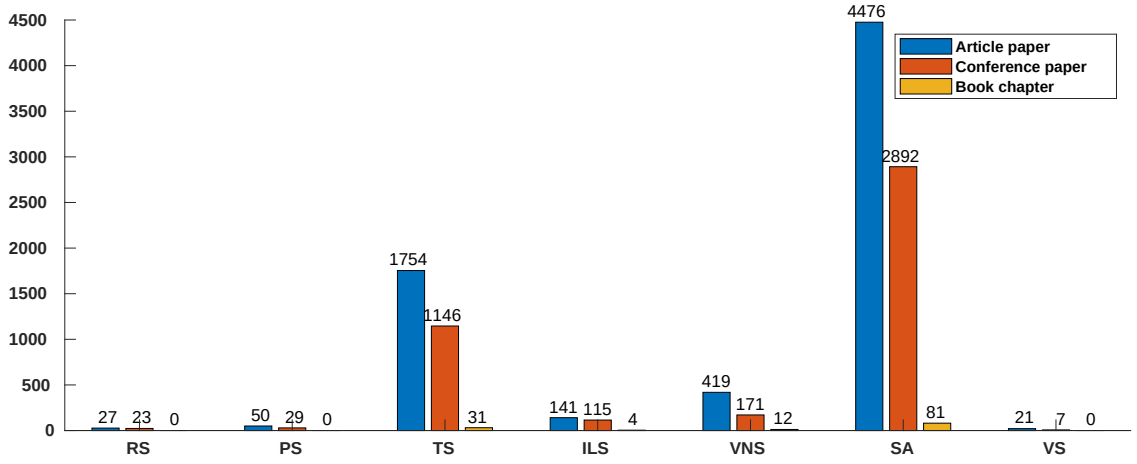

Figure 2.2. Total number of scientific publications on single solution-based heuristics.

set to a higher value and lowered through a *cooling* function throughout iterations. The higher the temperature is, the more likely SA to accept the recently generated solution even it is a worse solution than the current solution. So, the proposed scheme in SA ensures 'explore first, exploit later' approach.

VS [40] is a recently proposed single solution-based algorithm. It is inspired by the vortex pattern in nature. VS starts with a solution vector that is randomly created and generates a number of candidate solution vectors around the initial solution within the search space. Among them the best neighbor is then chosen to be exploited in the next iterations. The radius of the search space is shrunk through a *incomplete gamma function* as the iteration proceeds. The incomplete gamma function is a type of special functions mostly used in probability theory particularly [41]. The main drawback of VS is that it could prematurely converge in case a chosen solution is somewhere in the vicinity of a local minimum.

The total number of scientific studies based on single solution-based approaches is given in Figure 2.2.. The number of recent studies which are published in the last six years is given as a heat map chart in Figure 2.3.. These figures show that SA and TS are the most employed algorithms, whereas RS, PS, and VS have been employed in a limited way so far. However, it is worth mentioning that VS is the most recent approach and, albeit slowly, it's popularity has been increasing year by year.

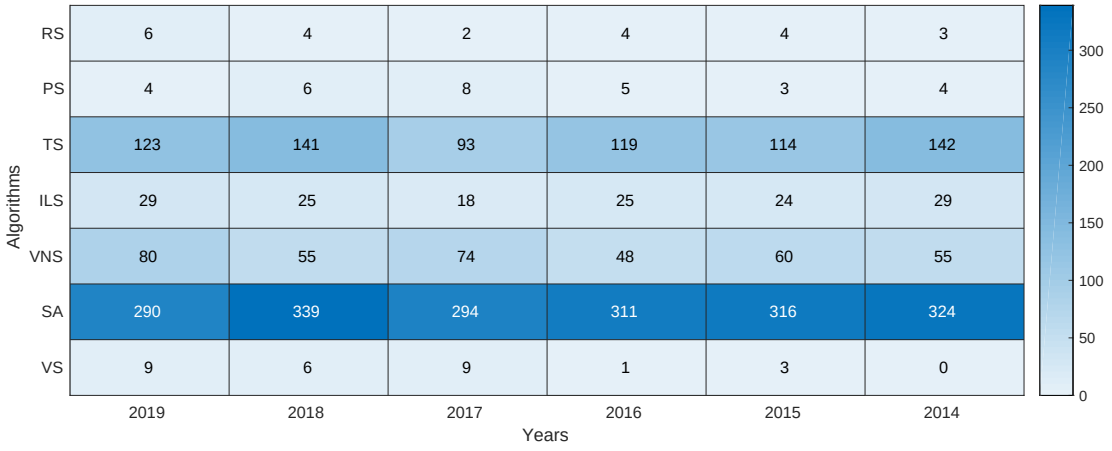

Figure 2.3. Analysis of recent scientific publications on single solution-based heuristics.

2.1.2. Population-based heuristic algorithms

Population-based heuristics employ multiple agents to explore the search space. They are basically studied under two major groups: *Evolutionary-based algorithms* and *swarm intelligence-based*, or *swarm-based*, algorithms.

2.1.2.1. Evolutionary-based heuristic algorithms

Evolutionary-based algorithms are inspired by the natural selection phenomenon in natural evolution. GA [42], GP [43], ES [44], DE [45], EP [46], and BSA [47] are examples of the most popular evolutionary-based algorithms inspired by the notion of *survival of the fittest*. They are mainly differentiated from each other in the representation of individuals in a population, which are candidate solutions evolved for solving a problem. Even if they rely on similar generic operators which are basically *crossover*, *mutation*, and *reproduction* during a search, they also differ from each other in their way to apply the generic operators to individuals.

GA [42] and GP [43] are the most popular evolutionary-based algorithms. In GA, individuals of the population called chromosomes are represented using a string (usually string of bits). On the other hand, individuals are represented as trees in GP. GA/GP evolves a set of offspring from parent individuals selected from the current population. These parent

individuals are selected based on their fitness values through different selection strategies such as roulette-wheel, rank-based, tournament selection. Fitness value of an individual indicates how well this individual solves the problem at hand. Then, genetic operators such as crossover, mutation, reproduction are applied to these parent individuals in order to generate new individuals. Crossover operator is often applied by exchanging sub-parts of two strings (or the subtrees of individuals in GP). Mutation operator generally exchanges a part of the randomly selected individual with a randomly generated substring (GA) or subtree (GP), hence provides the diversity in a population. Since the population size is usually constant, so the individuals who will survive in the next generation need to be selected based on replacement mechanisms such as steady-state or simple approaches. The same process is repeated until the termination criterion is satistified.

ES [44] is another evolutionary-based algorithm. There have been several variants of ES in the literature, which show differences usually in the number of selected parents and evolved offspring. The ES algorithm is names with the notation of $(\mu/\rho + \lambda)$-ES, which states that $\rho \mid \rho \leq \mu$ parent individuals are selected from the μ-sized population and λ offspring individuals are evolved. While $(\mu/1 + 1)$-ES is the base strategy of ES where only one parent individual is chosen to evolve only one offspring; $(\mu/\rho + 1)$-ES is the most popular multi-membered evolution variant of ES where ρ parent individuals are chosen to evolve one offspring.

DE [45] also employs generic operators in order to evolve new candidate individuals. However, unlike GA and GP, it firstly applies mutation operator in order to generate new solutions. In other words, mutation is the main search operator in DE. In mutation, three different individuals are randomly chosen and a new solution vector is constructed through a scaling factor. The crossover operator is applied to those individuals. Since all individuals in the population have the equal chance of being selected as a parent individual in DE, it often performs a better global search than other evolutionary-based algorithms.

Similar to DE, EP [46] also applies mutation operator to all individuals. As a result of the application of this operator, the population size is doubled. Again, replacement strategies

are applied to preserve the population size at each generation. Here, EP propose a novel replacement strategy that is completely different from other evolutionary-based algorithms. In this strategy, a rank value of every individual is calculated by comparing that individual with a randomly chosen individual from the population, instead of using all individuals. Then, the best individuals who have the highest rank values are preserved in the new population. Hence, depending on the randomly chosen individuals, EP does not always ensure all the fittest individuals to survive in the next generations, which increases the diversity in the population.

BSA [47] is a recently-proposed algorithm that highly differs from the other evolutionary-based algorithms. Apart from the parent population, BSA also relies on another population called *historical* population. Historical population in BSA is an auxiliary population used to calculate the search direction at each cycle. It behaves like a memory in BSA and stores the current population as long as a condition governed by a stochastic way is satisfied. As in DE, BSA first applies mutation operator in order to generate offspring individuals. Here, not only the parent population but the historical population is also taken into account. The mutated population is then exposed to the crossover operator and the fittest individuals are taken for the next generations through greedy selection.

The total number of scientific studies based on evolutionary-based approaches is given in Figure 2.4.. The number of recent studies which are published in the last six years could be further analyzed by using the heat map chart given in Figure 2.5.. These figures show that GA is by far the most popular evolutionary-based method. Then, GP and DE come after GA in terms of the number of publications. Considering the limited number of publications, BSA can be regarded as the least popular evolutionary-based approach in the literature. However it is not surprising, since BSA is the most recently proposed evolutionary-based algorithm (2013).

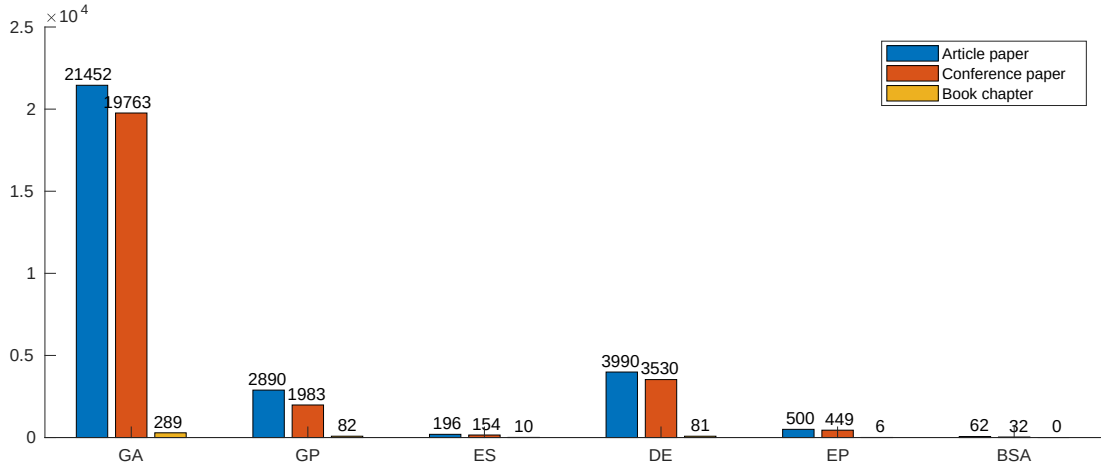

Figure 2.4. Total number of scientific publications on evolutionary-based heuristics.

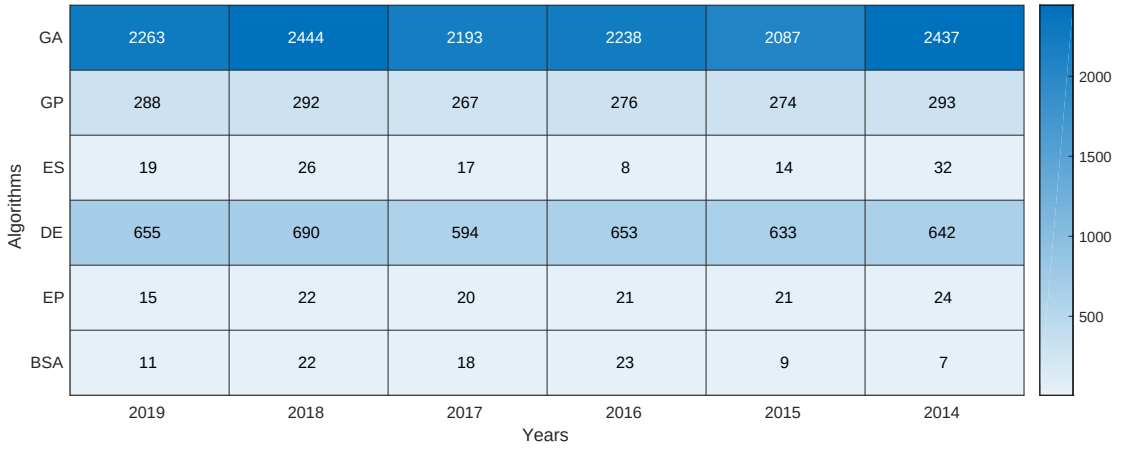

Figure 2.5. Analysis of recent scientific publications on evolutionary-based heuristics.

2.1.2.2. Swarm-based heuristic algorithms

These algorithms are inspired from collaborative intelligent behavior of swarm agents. ACO [48], PSO [49] and ABC [50, 51] algorithms are the most popular swarm-based heuristics. There are also other swarm-based heuristics which especially target to solve the real-valued problems. All these algorithms are explained in this section.

With their ability of discovering the shortest paths between their nests and the food sources and their indirect communication behavior, ants are the inspiration of the ACO algorithm [48]. It was originally proposed to find the optimal paths of a given directed graph. An ant deposits

a substance known as *pheromone*, which evaporates over time, on the path where it passes through. The other ants tend to follow a path where the amount of pheromone is higher. The other ants tend to follow paths which have higher amount of pheromone. Therefore, in time, more pheromone is deposited on the shorter paths and remained longer, since more ants will complete such paths in a shorter time than other paths. As the pheromone deposited on a path remains longer, the probability of this shortest path being selected by the colony increases. This is the basic principle on which the ACO algorithm is based.

The collective behavior of a school of fish or a flock of birds inspired the emergence of the PSO algorithm [49]. The algorithm keeps the historical best position of every individual called particle and the global best position of the population called swarm. Based on these positions, the velocity of each particle is changed and then their new positions are determined. The best positions of all particles are mainly used in PSO to contribute a better the local search. That having a straightforward structure and using only a single search operator make PSO a very popular approach in the literature. However, the lack of a proper global search mechanism is the main disadvantage of PSO.

ABC [50, 51] is inspired by the collective foraging behavior of honey bees. It comprises three search phases performed by employed, onlooker, and scout bees. Employed bees are in charge of exploiting the existing solutions called as food source in the algorithm. Then, onlooker bees further develop the high quality ones among these solutions. The poor food sources are abandoned and new sources are discovered by scout bees. Thus, a balanced search is provided in ABC by employed and onlooker bees contributing to local search and scout bees mainly contributing to global search.

The flashing light of fireflies to communicate and to attract potential prey inspired the emergence of FA [52]. The algorithm makes use of the 'attractiveness' of the fireflies which changes as a function of brightness of the light and the distance. Among two individuals (fireflies in the algorithm), an individual having a higher brightness will attract the other individual. This mechanism provides local search in FA. If there is no 'brighter' individual, the algorithm perform a random search. This contributes to global search in FA. FA is an

easy-to-implement algorithm; however the distance calculation in FA in order to update the brightness of every individual in the population increases the running time of the algorithm considerably. That, therefore, deteriorates the computation efficacy of the algorithm.

Similar to FA, GSA [9], which is inspired from the law of gravity and mass interactions, also employs the 'attractiveness' of the search agents. The quality of every search agent in GSA is represented by its mass which determines the gravity force of the agent. So, the agents with the heavier-masses, corresponding the good solutions, attract the other agents. The poor agents moving faster in the search space contribute to exploration; whereas the good agents contribute to exploitation in GSA. As in FA, GSA uses Euclidean distance calculation in order to update the gravitational forces of individuals, which increases the time complexity of the algorithm.

BHA [30] is also inspired from the gravitational force. The irresistible gravitational force of the black hole was modelled in BHA. The stars represent the candidate solutions in the algorithm and the fittest star among them represents the black hole, a collapsed form of a massive size star. The quality of each solution is measured by its mass, as in GSA. All stars are attracted by the black hole, and hence the positions of the stars are updated accordingly. This ensures the exploitation in the algorithm. Any star crossing the event horizon (a poor candidate solution), a sphere-shaped boundary of the black hole, is replaced with a new star randomly generated in the space, which ensures the exploration in BHA.

The aggressive breeding behavior of cuckoos has inspired the CS [53] algorithm. The cuckoo lays their eggs in other's nest and throws the other eggs from that nest in order to increase the hatching probability of of its eggs. The host parent either removes the alien eggs or abandon its nest. In the algorithm, the cuckoos represent the search agents and the eggs represent the solutions. The Lévy flight is modelled in CS for the movement of the cuckoos, and hence for the discovery of nest. The local search is not enhanced as much as the global search, which is highly provided by both the Lévy flight and a random generation of new eggs.

The *echolocation* behavior of microbats, an ability to locate objects by listening the echoes returning from the emitted very loud sounds, was modelled in the development of BA [54].

The insectivorous microbats use a type of sonar called echolocation to detect prey or to avoid obstacles. The echolocation is characterised by the frequency (pulse emission rate) and the amplitude of microbats. The pulse emission rate and amplitude change as a function of the distance to the target hence the former speeds up sharply as a microbat approaches to the target and the latter reaches up the loudest level while it is searching for food. In the algorithm, the microbats represent the search agents, which store food sources (candidate solutions). The quality of each source is assessed through its fitness. In the case the emission rate is high, the search agent is enforced to perform a local search in the vicinity of the best food source, whose range is determined by the amplitude of the agent. Otherwise it performs global search by flying randomly in the search space.

Another echolocation inspired algorithm is DOE [55] algorithm which mimics the strategy used by dolphins for navigation and hunting. It was originally proposed to solve discrete optimization problems and later extended in order to solve continuous problems [56]. The dolphins (search agents) initially performs exploration in the search space, then it decreases the degree of global search through a convergence factor in each iteration, which ensures the local search during the last iterations. Therefore, the DOE algorithm ensures 'explore first, exploit later' approach; however, it contains a number of parameters such as power, loop number, radius that should be properly tuned.

IWO [57] mimics the aggressive reproduction behavior of vigorous weeds. Weeds in nature are undesired and pose several thread for agriculture. The robustness and adaptability to the changing environment enable weeds to occupy a territory easily in spite of the countermeasures like herbicides application. The IWO algorithm starts with an initialization of a population of seeds, which represents search agents in the algorithm. Depending on its robustness/fitness, every seed produces a number of seeds. The spread of the produced seeds decreases as the iteration proceeds. Hence, the distant part of every seed is explored at the initial iterations; while their vicinity is exploited at the final iterations. Hence, IWO first performs global search and then it performs local search.

FPA [58] is inspired from the pollination of flowering plant species. Pollination is a process

to transfer the pollen of a flower through pollinators such as birds, insects. Pollination can be examined under two forms: *abiotic* and *biotic* pollination. Where pollen needs pollinators to be transferred in abiotic pollination, it does not in biotic pollination. In addition, pollination can be achieved through both self-pollination, the pollination through the pollen of the same flower of the same plant, and cross-pollination, the pollination through the pollen of the different flowers. Through pollinators, biotic and cross-pollination may occur at long distance; that's why it is called as global pollination. Abiotic and self-pollination, however, occur within the short distance. In FPA where flowers represent search agents search is performed through two processes: global and local pollination, which corresponds in the algorithm to the exploration and exploitation, respectively. The search mode selection in an iteration is completely probabilistic in FPA. The global pollination is done over a long range and the search is performed between the flower itself and the best flower through pollinators. Here, Lévy flight behavior of pollinators is used, which further contributes to the exploration. The local pollination, however, is done in the neighborhood of a flower through a random search between the flower itself and randomly selected one neighbor.

AAA [59] is inspired from the reproduction and adaptation behavior of algae. Algae grow in a liquid with sufficient nutrition and light for photosynthesis. The lack or insufficient amount of these resources causes algae to die or to adapt changing environment. Therefore, algae are in a continuous search for the optimal environment, which shows that they often change their locations. The algal colony represents search agents in the algorithm and their locations represents candidate solutions for the problem at hand. Hence, the objective of AAA is to find an environment with optimal condition with respect to the light and nutrition. *Helical movement*, *evolutionary process*, and *adaptation* are the main phases proposed in AAA. With its flagella, an algal colony swims in a helical manner until its energy finishes, which is modelled in helical movement phase. The movement speed of an algal colony is reversely proportional to its growth rate. The larger size a colony gets, the slower it moves since its friction to the surface intensifies, and vice versa. This strategy enables the algorithm to have exploitation and exploration capabilities that are performed through, respectively, the large- (cost efficient agents) and thin-sized (not-good agents) colonies. In the evolutionary

and the adaptation phases of AAA, an algae colony having the poor condition is adapted itself to another algal colony having one of the most appropriate conditions by replacing one of the randomly selected genes.

Similar to AAA, the movement behavior of *superorganism* such as honeybees, whales, and etc. is the basis of the swarm-based DSA [60]. DSA is inspired by the seasonal migration behavior of living beings to find fruitful areas throughout the year. The superorganism discovers a new area, called *stopover area* in DSA, and settle there for a while if it is suitable in terms of natural source and then they continue their migration from this area on. The superorganism in DSA represents the population in which artificial-organisms represent the search agents. The so-called *stopover area* of an artificial-organism, however, represents the solution in the algorithm. The discovery of new stopover area in DSA relies on three factors: i) current position of superorganism, ii) shuffled current position of superorganism, and iii) a complete stochastic design phase. The former two factors evaluate the next position of each artificial-organism while the latter one determines which of the evaluated new positions are considered for the following generation. This stochastic design in DSA contributes mainly to the exploration of the algorithm. Hence, exploitation is not supported as much as it is needed in the algorithm.

The hunting and migration behavior of dragonfly is modelled in DA [61]. Dragonflies rarely perform a swarm behavior, unless they are hunting or migrating. They build two types of swarm: local and global swarm. In local swarming, a small group of dragonflies hunt in a too restricted area where the target prey live. In global swarming, however, a large number of dragonflies gathers to migrate over long distances. This swarming types form the basis of DA's search strategy. While the former corresponds to exploitation, the latter corresponds to the exploration. Dragonflies represent search agents in DA, whereas their locations represent the candidate solutions. Similar to the velocity calculation in PSO, a step size vector is used in order to calculate the next positions of agents. The step size calculation, however, relies on weighted five different factors that are modelled from the swarming principle of dragonflies. These are *separation*, *alignment*, *cohesion*, *attraction*, and *distraction*. The former three ones come from the flying pattern of dragonflies, whereas the latter two ones come from

the natural behaviors that change as a response to the target (prey and enemy). Through the modelled swarming behavior, DA ensures a well-balanced global and local search. However, there are five parameters that determine the weight of every factor in step size vector, which should be properly adjusted.

The natural phenomenon of lightning inspired the emergence of LSA [62]. The lightning is a natural event that occurs when the difference between the negative charge within the thunderstorm and the positive charge at the base of the storm becomes too large. When this happen, downward move of projectiles such as stepped leaders space leaders and the like starts. This movement from cloud to ground progresses through regular and discrete steps within different channels, which was modelled in LSA. The movement velocity of every projectile through a channel depends on its mass. The smaller a mass is, the higher potential for a projectile to exploit the nearby space, and vice versa. Therefore, the exploitation and exploration in LSA is controlled through the masses of the projectiles which represents the population in the algorithm.

KHA [63] was inspired by the herding behavior of krill individuals. The krill have an ability to form a large herd and often live within it. However, average krill density decreases when predators like seabirds attack the krill. This also leads the krill swarm to be distant from the locations of food. The krill are search agents while their positions represent candidate solutions in the algorithm. The location of each krill is updated based on the Lagranian model of krill herding. This model includes three motions: *movement induced by other krill individuals*, *foraging activity*, and *random physical diffusion*. In addition to that, KHA also applies generic operators such as crossover and mutation to further improve the search. Adopted motions driven by Lagranian model allows the algorithm to perform a well-balanced local and global search.

GWO algorithm [64] was inspired from the social leadership hierarchy in grey wolves and their hunting strategies. Living in small packs, grey wolves are grouped into four different social dominant levels (alpha, beta, gamma, and omega). The leader wolves at the most dominant level (alpha) make almost all the decision in the pack, wolves at the second level

(beta) are the best candidate to be leader and they assist the current leader wolves in making decisions. Delta wolves are the employed individuals responsible for necessary tasks such as scouting, hunting. Omega wolves are the least important individuals but they ensure the order in a pack. The wolves represent search agents in the algorithm. Among them; the fittest, the second fittest, and the third fittest agents are, respectively, represented by alpha, beta, and gamma wolves, while the rest of population are represented by omega wolves. The hunting (corresponds to a search in GWO) is guided by the fittest wolves and performed by omega wolves. The attacking –approaching towards the target prey– behavior of grey wolves is modelled to fulfill the exploitation, while searching –moving away from the current target– for a better target prey is modelled to fulfill the exploration in the algorithm.

Similar to GWO, the functioning of social leadership in animals inspired the emergence of the SMO algorithm [65]. Spider monkeys are animals that follow a very strict leadership. They build fission-fusion swarm meaning that they split or merge their group depending on the scarcity or abundance of the food, respectively. There are two types of leader in the group: global and local leader. Global leader is the leader of all the groups that make the decision to divide the groups into sub-groups. However, a local leader is only responsible for his own sub-group and make plans for foraging. The search in SMO is initiated by a number of spider monkeys, corresponding to the search agents in the algorithm. Then, the locations of search agents are updated with respect to the positions of the local leader (exploration) and the global leader (exploitation). In case where a local leader of a group is unable to find a better food source for a long while, the positions of all the individuals in that group is updated, which avoids premature convergence in the algorithm. Similarly, if the global leader is unable to find a better food source, then the group is divided to further intensify the search in parallel manner, which avoids the stagnation of the global best agent.

WOA [66] was inspired by the foraging behavior of humpback whales. The gigantic humpback whales live alone or in groups throughout their lives. Krill and small fish are their primary prey. These whales use an fascinating strategy to hunt the prey: bubble-net feeding. In this strategy, whales consecutively create distinctive bubbles along a circle to make prey encircled within very restricted area near to the surface. The whales are the search agents and

their positions are the solutions in the WOA algorithm. Hence the objective of the algorithm is to find the fittest prey. The search strategy adopted in WOA is very similar to the GWO algorithm with three sequential steps: *encircling prey*, *attacking to the prey*, and *searching for new prey*. The bubble-net foraging is mimicked in the first step (*encircling prey*). Here, the positions of the whales is updated such that they encircle the best prey found so far. The *attacking to the prey* step allows the algorithm to further exploit by approaching the best prey. However, the last step, allows the algorithm to explore a distinct part of the space by moving away from the prey as proposed in GWO.

ISA [67] is a novel and an interesting heuristic method inspired by an interior design procedure. The primary goal of an interior design is to obtain a beautiful and a decorative view satisfying the needs and resources of the clients. The interior design, according to the authors, usually starts from the walls to the center. Designers changes the composition of elements to have a more beautiful environment. Mirrors are one of the important elements in interior design process. They should be placed very close to the most beautiful elements to emphasize their beauty. The elements used in interior designing represent the search individuals and their places represent the candidate solutions. Hence, the fitness of a search individual represent how beautiful a decorative view is obtained. ISA has only one parameter which balances the exploration and exploitation. Depending on the value of this parameter, ISA randomly divides the population into the two groups (excluding the fittest element): *composition* and *mirror*. The individuals in the composition group updates their places in a random manner within the space bounded by the current places of all elements. This adoption provides global search ability to the algorithm. On the other hand, the individiuals in the mirror group are placed between any element and the fittest element and so they perform a search near to the fittest element, which ensures the local search ability in the algorithm. The fittest element in the population performs a random walk around of itself, which increases exploitation ability as well.

TCA [68] is another heuristic algorithm with multi-group population. TCA was inspired by a central role of T-cells in the cell-mediated immunity. T-Cells belong to the group of

white blood cells playing a central role in the cell-mediated therapy. Based on their development stages, T-Cells are divided into three groups: *virgin-cell*, *effector-cell*, and *memory-cell*. While virgin-cells do not suffer from pollination or differentiation, effector-cells are the cells that are activated by differentiation. They go through activation by antigens, which hence replicates the cells themselves. Memory-cells also respond through differentiation like effector-cells but they persist into the host even after the infection is overtaken so that they are stimulated by the same antigens. There are three populations corresponding to these group of T-Cells which represent the search agents in the algorithm. TCA starts with only virgin-cell population and it becomes the effector-cell population. The agents in effector-cell population then undergoes mutation operation. The best agents in effector-cell population are inserted into the memory-cell population where another mutation operation is also applied. While the effector-cell population enables the algorithm to ensure diversity, the memory-cell population ensures exploitation by enabling the algorithm to search neighborhood of best agents.

CSA [69] was inspired by the intelligent foraging behavior of crows. The form of the flock, the memorization of the position of their hiding places, the thievery, and the protection of their caches are the most intelligent behaviors of crows that are modeled in the algorithm. The crows are the search agents and their locations are the candidate solutions in CSA. Algorithm starts with a generation of a number of crows. As similar to PSO, each crow holds its location and the position of the hiding place, which represents the best location that the crow has obtained in CSA. The position of a crow is based on the fooling behavior of them in nature and is updated based on two scenarios: i) it approaches to another crow's hiding place or ii) it performs a search in another location. While the former model increases the exploitation, the latter model increases the exploration in CSA. CSA is easy to implement with only two search operators. However, there are two control parameters that should be fine tuned, since they directly affect the balance between exploration and exploitation.

BFO algorithm [70, 71] was inspired by the foraging behavior of bacteria (E. Coli). Their ability to move to regions with high nutrient levels and to communicate with each other form the basis of the algorithm. Bacteria (search individuals in the algorithm) are initially

positioned in the search space in a random manner. In order to update the position of every bacteria, BFO uses three nested processes that are applied separately at every iteration: *elimination-dispersal*, *reproduction*, and *chemotaxis*. Even if these search processes enable BFO to successfully converge, they highly increase the computational cost of the algorithm. In addition to that, BFO involves a sorting algorithm in order to sort the population at each generation.

TLBO algorithm [72] was inspired by a mathematical model on presenting the effect of teachers on learning of students. The effect of teachers who are regarded as well-educated people is often measured by the grades of students. The main goal of teachers to increase the knowledge level of whole class, which causes higher student grades eventually. While learners represent the population where their grades are measured through the fitness function, teacher represents the best solution in TLBO. There are two successive steps in each iteration of the algorithm: *teacher phase* and *learner phase*. While the teacher phase refers to learning from the teacher, the learner phase refers to learning as a result of interaction between learners. In the teacher phase, the search is performed between a learner and a teacher, while in the learner phase, it is performed between two randomly chosen learners. So, the exploitation of the best solution and the exploration of the search space are ensured in the teacher and learner phases, respectively.

STA [73] is another swarm-based algorithm which was inspired by the concept of state transition and state space representation in control theory. In STA, every solution is regarded as a state and the update of the current solution is treated as a state transition. Only the best solution in the algorithm is subjected to three consecutive transformation operators at every iteration: *rotation*, *expansion*, and *axesion*. The rotation operator enables a solution in STA to perform a local search through generating a new solution within its hypersphere. The expansion operator enables to perform a global search by expanding the search range. The axesion operator, however, allows STA to perform a single-dimensional search, which strengthens both global and local searches.

The total number of scientific researches proposed on the swarm-based approaches has been

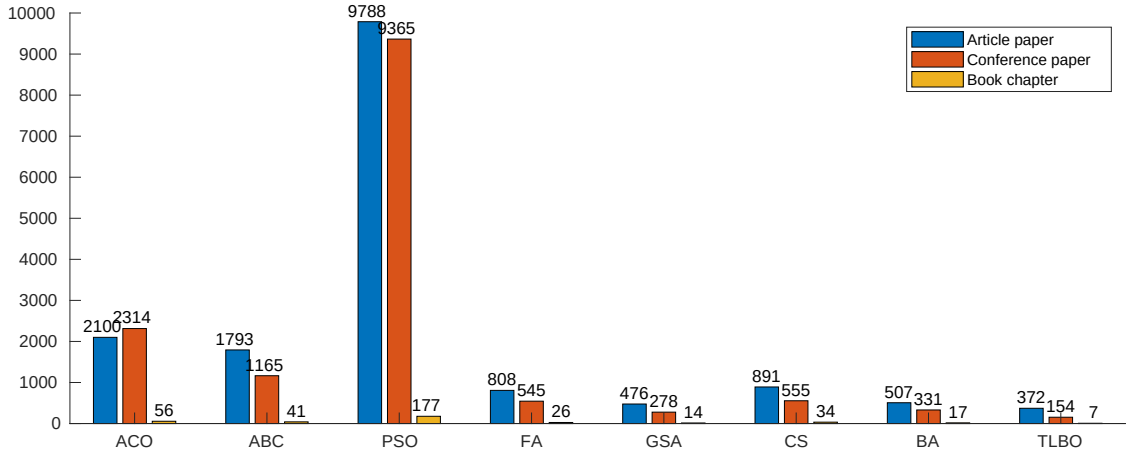

Figure 2.6. Total number of scientific publications on swarm-based heuristics.

given in Figure 2.6.. The distribution of publications in this area outcomed within the last six years is demonstrated in Figure 2.7.. It is worth mentioning here that only swarm-based approaches having the highest publication counts have been included into the figures for the sake of better readability. The findings in these figures show that most of the publications in the literature use PSO algorithm, then followed by ACO and ABC algorithms in order to develop or to apply to solve a problem. Among them, ABC is a recently proposed algorithm, which indicates that it has been well adopted in the literature. Their easy-to-implement structure and gaining a satisfying performance on solving problems are the main aspects that make these algorithms very popular in the literature. In addition, there are several publications made on FA and GSA that rely on a high-computational distance measurement, which shows that *i*) computationally expensive algorithms have much more been studied to make them efficient or that *ii*) researchers can often sacrifice the efficiency of the employed algorithm on solving a problem.

The distribution of application areas of single solution-, evolution- and swarm-based algorithms are given in separate pie charts in Figure 2.8.. These charts clearly show that *computer science*, *engineering*, and *mathematics* are the dominant subject areas comprising a very large portion of all areas for all types of heuristic algorithms.

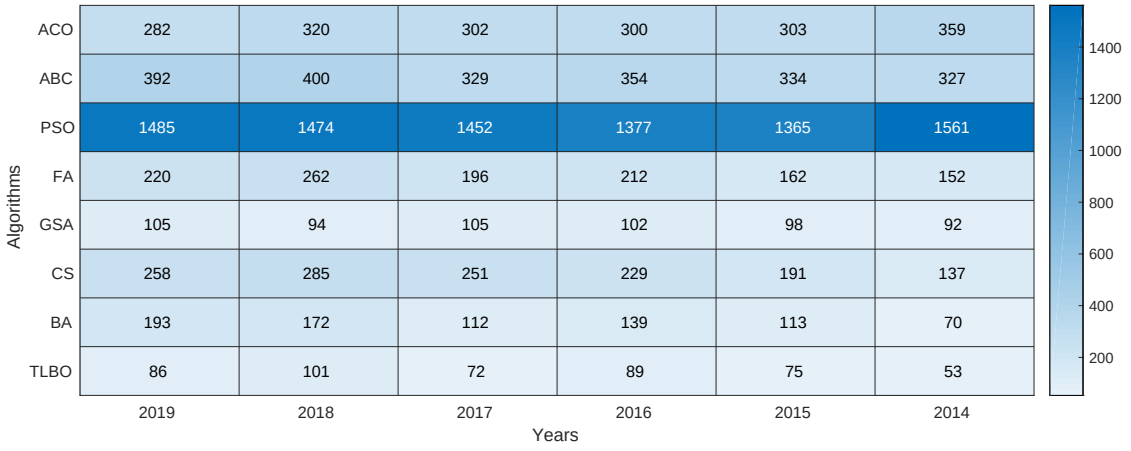

Figure 2.7. Analysis of recent scientific publications on swarm-based heuristics.

In addition to the base population-based optimization algorithms, modification of these algorithms and combining different algorithms in hybrid methods have also been proposed in the literature in an attempt to further improve the performance of population-based heuristics. Although there have been numerous advancements on the several base algorithms, only few of these studies that were built on the most popular population-based GA, DE, ES, PSO, and ABC algorithms has been explained in this thesis.

The stagnation and slow convergence problems arisen in PSO were handled by integrating it with DE in PSO-DE [74]. The particles in PSO prematurely terminate exploration and stagnate in the rest of the search process especially when they gather to the same local minimum. In order to avoid that, PSO-DE enables PSO to evolve new particles using only half of the population involving the poor particles and then employs DE to generate three offsprings from the historical bests of these particles. Another modification that improves the search performance of PSO and that prevents the premature convergence problem of PSO is proposed in QPSO algorithm [75] algorithm by employing mutation operator with Gaussian probability distribution.

A binary-coded PSO is proposed in [76] for dealing directly with the problems that have real, integer, and discrete variables. The only modification is carried out in the velocity and position update mechanisms of the standard real-coded PSO. While continuous variables are still handled by the standard real-coded PSO, discrete and integer variables are updated by

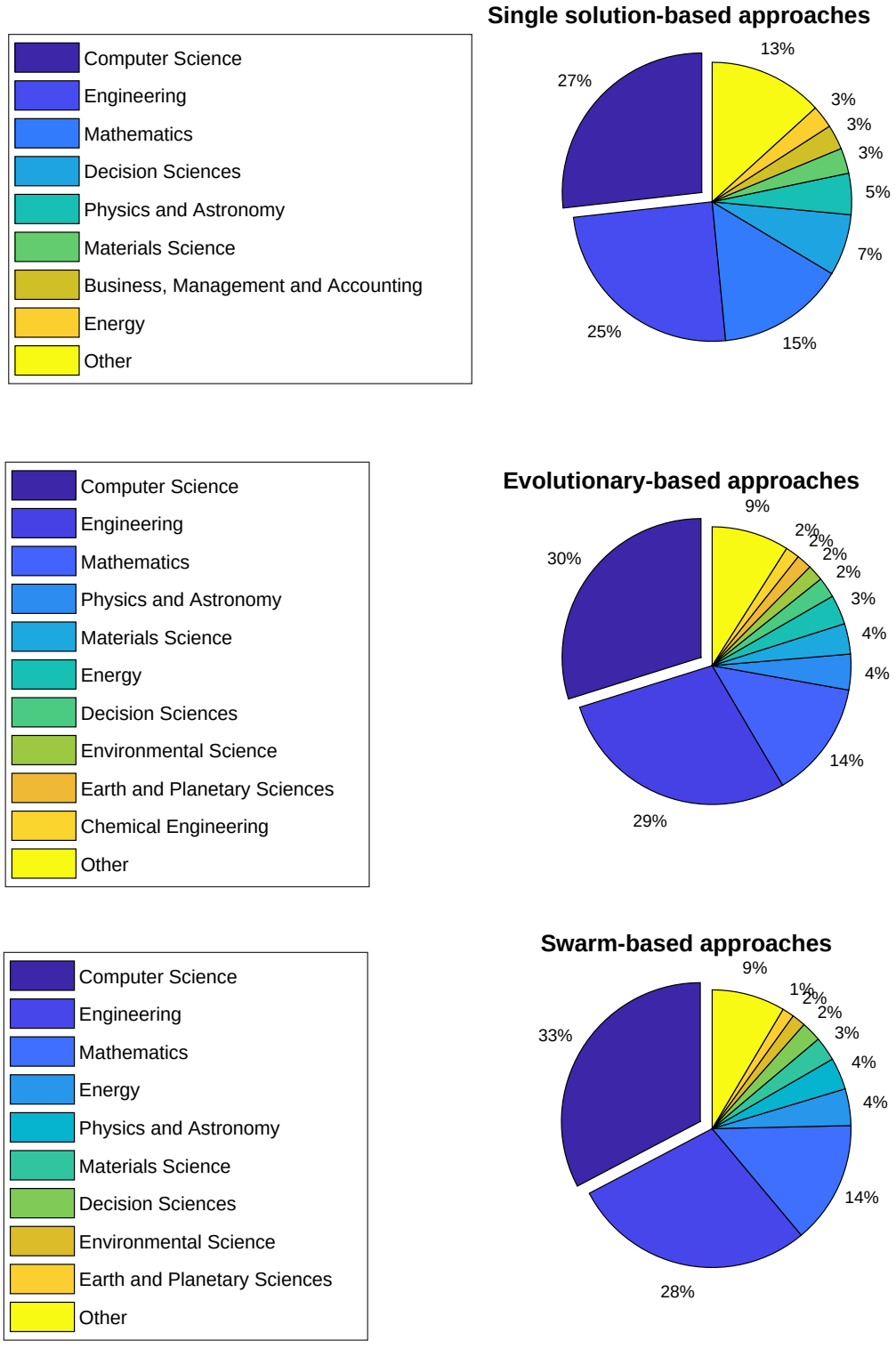

Figure 2.8. Distribution of application research areas of single-objective heuristics.

the operators proposed by binary-coded PSO. Therefore, while real variables are represented by a single dimension, every integer and discrete variable are represented by an array of bits that are calculated from the upper limit, which incredibly increases the total number of parameters to be optimized and hence the complexity of the problem.

DE has also been modified to achieve a better optimization performance in various studies. In [77], it has been subjected to a simple modification to generate more than one offspring per parent so that the probability of obtaining better solution is increased; however it leads the proposed algorithm to have an additional parameter that needs to be prior set. The exploration and exploitation capability and the convergence rate of DE have been enhanced in NDE [78] through a new mutation scheme, called triangular mutation rule. In contrast to the standard DE, where only random or best vectors are included into the mutation operator, the proposed scheme relies on a combination of a random vector and a difference vector between the best, better, and worst vectors.

Another modification to DE has been introduced for constrained optimization problems in DSS-MDE [79]. DSS-MDE adopts a dynamic stochastic ranking scheme in order to keep promising infeasible solutions that are often discarded during the selection procedures. Although the proposed scheme ensures a fine search balance between feasible and infeasible regions, it brings about an additional computational complexity to the algorithm.

The time consuming and problem-dependant schemes in DE for determining the search parameters such as scale factor, crossover rate were made adaptive in Self-adaptive Differential Evolution (jDE) [80]. Differently from to the basic DE algorithm, every individual in jDE carries a different scale factor and crossover rate that are calculated depending on how well the individual has converged. However, the proposed dynamic scheme in jDE has extra two parameters used to update the search parameters between generations, even if the empirical findings from a limited number of problems shows that these search parameters are less-sensitive to different problems [80].

Adaptive Differential Evolution (JADE) [81] has also addressed the premature convergence problem in DE by increasing the diversity of the population. Since only the best individuals

are chosen before the application of the mutation operator to evolve new offspring individuals, the premature convergence problem occurs in DE. In order to avoid that the use of an external archive that comprises of the only inferior individuals is proposed in JADE. During the evolution of a new individual in JADE, a fittest parent individual as well as an individual randomly chosen from the archive is taken into account. In addition to that, crossover rate and scaling factor is also updated among generations in order to further ensure an adaptive convergence speed and a better optimization performance.

In SADE [82], a well-balanced search strategy was applied to ensure a good diversity and convergence in DE. It makes use of two search operators; while the first operator is applied on the selection of best individuals, the second is applied on the random selection of individuals. Every individual in SADE probabilistically chooses one of the search operators then evolves an offspring accordingly. Scaling factor and crossover rate were also enforced to be updated as in [80, 81].

The performance of ES has been improved in [83] by introducing three simple selection mechanisms in order to maintain a higher diversity. Even though the proposed approach does not introduce new parameters and yields low computational cost, its weak local search pointed out by the authors needs to be further improved. In [84], an evolutionary-based algorithm (RAER) with a novel roulette inversion operator has been proposed in order to maintain a diverse population. The main goal of the roulette inversion operator is to enable RAER to explore every search dimension evenly.

In [85, 86], GA has been hybridized with artificial immune system to help the population move into the feasible region. Chromosomes are classified as *antigens* and *antibodies* depending on the feasibility. Afterwards, they are subjected to a series of operators such as tournament selection, mutation, crossover, and the like. The higher computational complexity of the proposed hybrid method is stressed as the obvious drawback of the approach.

In [87], CS algorithm [53] was hybridized with GA through its the generic operators in order to improve poor balance between global and local search capability of CS. Crossover in GA and the Lévy flight in CS contribute the algorithm by performing a further diversified global

search in the space and reducing the premature convergence problem in CS. Mutation allows a better local search through a moderate change on individuals.

The improved version of FOA [88], which relies on the excellent behavior of olfaction and vision of the fruit fly while foraging, has been proposed in order to address the problems of premature convergence and poor quality of solutions in the standard FOA algorithm [89]. In the proposed algorithm, linear diminishing and logistic chaotic mapping are employed in order to avoid becoming trapped in a local optima and to increase stability of FOA, respectively.

To sum up, numerous algorithms and their improvements have been introduced in the literature. Nevertheless, the research area is still open for further improvements. Some of the following remarks may identify the potential areas in the future:

- *Parallelization of heuristics:* While some of the popular heuristic algorithms like GA, ACO, ABC, PSO are already implemented to solve the problems in parallel and distributed to reduce the computational time remarkably or to store all needed data in the memory efficiently, it is yet remains as an open research topic for recently proposed heuristic algorithms such as KHA, AAA and the like.

- *Focusing on the nascent heuristics:* As seen from the Figure 2.2., Figure 2.4., and Figure 2.6.; the distribution of the studies on the heuristic algorithms is too skewed. To be concrete, more than 90% of the studies are based on SA, TS for single-solution approaches, more than 97% of the studies have been conducted on GA, DE, and GP for evolutionary-based algorithm, and more than 80% of the studies are the extensions of ACO, ABC, and PSO. However, there are numerous heuristic algorithms other than the most popular ones such as TCA, AAA, DOE and the like that have been proposed in the literature and each of them has proven to show a very promising performance on different problem types. The so-called algorithms are also waiting for the researchers attention to be applied on different combinatorial problems or to be further strengthened for a better search performance.

- *More intelligent, self-adaptive heuristic:* It is easy to conclude from the reviewed heuristics here that parameter tuning is one of the drawbacks for a heuristic algorithm. One reason is that algorithm-related parameters, e.g. the limit parameter in ABC, may lead the algorithms to work in a biased manner once they are not well-tuned. Another reason is that, the parameter set of the algorithms changes according to the problem's characteristic. Therefore, it is crucial for an algorithm to find its optimal parameter values for the problems. Considering the time-consuming process of parameter tuning, implementing the heuristic algorithms to be smart enough to tune their parameters has become a necessity for the future works.

- *Adoption to deep learning research:* With the increase in the amount of the data and easy ability to access fast and efficient computing terminals has enabled the AI community to train very large neural networks. Therefore, the concept of 'deep learning' has recently welcomed in the literature as a "superpower" that builds different AI systems. Being a powerful deep learning technique, CNN is often applied for computer vision applications in deep learning research. Although there are some recent works built on heuristic algorithms used to obtain optimal CNN architecture [90], or to optimize the hyper-parameters (like kernel size, pooling window size) [91], the application of heuristic in the research area of deep learning is still immature. The promising performance of heuristic algorithms on traditional machine learning techniques may be motivation for AI research community to apply them to the unconventional deep learning techniques.

2.2. Multi-objective Optimization Methods

The complex nature of search space of some objectives in MOOPs such as non-differentiability, discontinuity makes traditional mathematical-based techniques impossible or at least ineffective to apply. That's why, researchers have proposed many population-based heuristic optimization methods, which are not dependent on the complexity of the search space. Such methods target to find as many non-dominated solutions as possible. The following section summarizes the well-known studies in this research area.

2.2.1. Evolutionary-based multi-objective heuristic algorithms

The first use of heuristic algorithms for solving MOOPs dates back to the introduction of the VEGA [92] into the literature. VEGA decomposes a problem with n objectives into n sub-problems and splits population into n sub-population. Hence, each decomposed sub-problem is addressed only by the corresponding sub-population. The most drawback of VEGA is that it leads the entire population to converge to only one optimum due to using a selection mechanism that often relies on the best solutions. The lack of proper selection scheme in VEGA led the emergence of the Pareto concept [93].

MOEAD [19] is another method that decomposes the problem into smaller sub-problems. The main goal is to minimize the maximum margin between each objective and its corresponding reference value that is equal to the worst objective. Contrary to VEGA, it uses Pareto optimal solutions in order to ensure diversity of solutions and maintains them in a memory. The main drawback, however, is that its performance is directly proportional to the decomposition method employed.

Contrary to the approaches above, IBEA [18] handles a MOOP as a whole. However, it does not consider the objectives, but a particular performance metric obtained from the objectives such as HV, EPSILON, and aims to increase the quality of solutions with respect to the chosen metric. While parent and offspring individuals build the next population, the poor solutions with regard to this metric is eliminated to maintain the size of the population. However, the use of a single metric could lead to a poor performance with respect to other metrics.

Similar to IBEA, SPEA-2 [17], efficiency-aware extension of SPEA [94], does not make directly use of the approximated Pareto front set for fitness evaluation either, but uses a *fitness* value calculated from the Pareto domination count of a solution and its distance to other solutions. One of the main concerns about SPEA-2 is that these two criteria used for fitness evaluation are treated equally, which enables solutions that are completely dominated

but farthest away from others to be kept in the population. The other concern is about the increased complexity that is risen from the distance calculation of solutions.

NSGA-II [16] is the most salient Pareto-based algorithm in the literature. It is based on the Pareto dominance and distances of solutions in the objective space. Depending on the Pareto domination, NSGA-II splits the population into several fronts in which distance, called crowding distance, is calculated. Differently from SPEA-2, non-dominated solutions that are in better fronts are allowed in NSGA-II to survive. If the size of the next front exceeds the size of the population, the distance plays a key role in determining individuals to remain in the population.

MOCell [20] is similar to NSGA-II in the selection of solutions to survive for the next generations. The basic generic operators of GA are applied in MOCell to breed offspring considering only the individuals in the close neighborhood which is determined through a cellular-like topology.

The total number of scientific studies on the multi-objective evolutionary-based approaches is demonstrated in Figure 2.9.. The distribution of recent studies on the area is also given as a heat map chart in Figure 2.10.. The figures show that NSGA-II is the most adapted algorithm in the literature by far. The main reason behind this popularity is *crowding distance*, the measurement strategy of NSGA-II, which is highly efficient with respect to time complexity. MOEA/D is another multi-objective heuristic algorithm that has gained a higher popularity although it is proposed recently compared to the other algorithms.

2.2.2. Swarm-based multi-objective heuristic algorithms

In addition to the well-known evolutionary-based algorithms, swarm-based heuristics have also been proposed in order to solve MOOPs. Such heuristics are generally proposed for solving SOOPs, then their extensions are introduced for MOOPs. As stated earlier, PSO [95] and ABC [96] are the most popular algorithms. Therefore they mostly have been adapted for solving MOOPs.

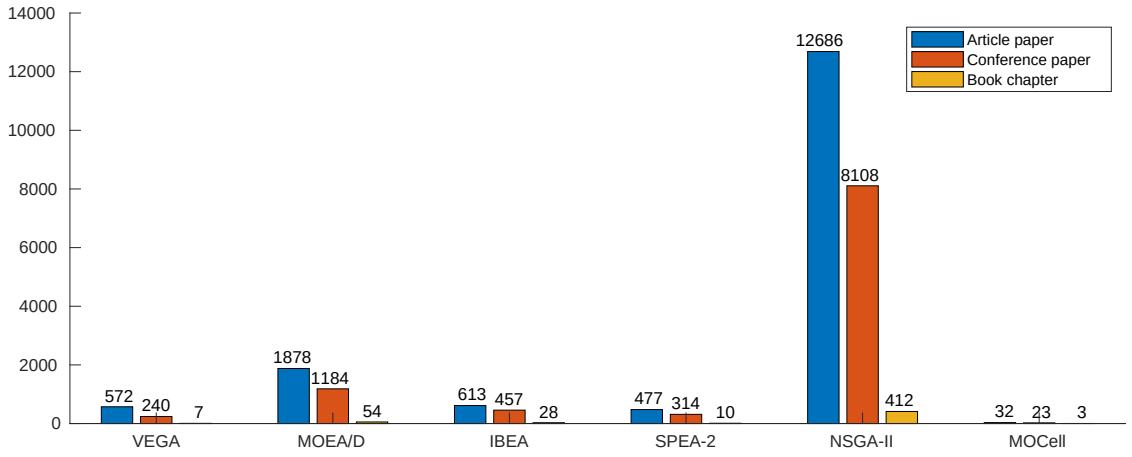

Figure 2.9. Total number of scientific publications on multi-objective evolutionary-based methods.

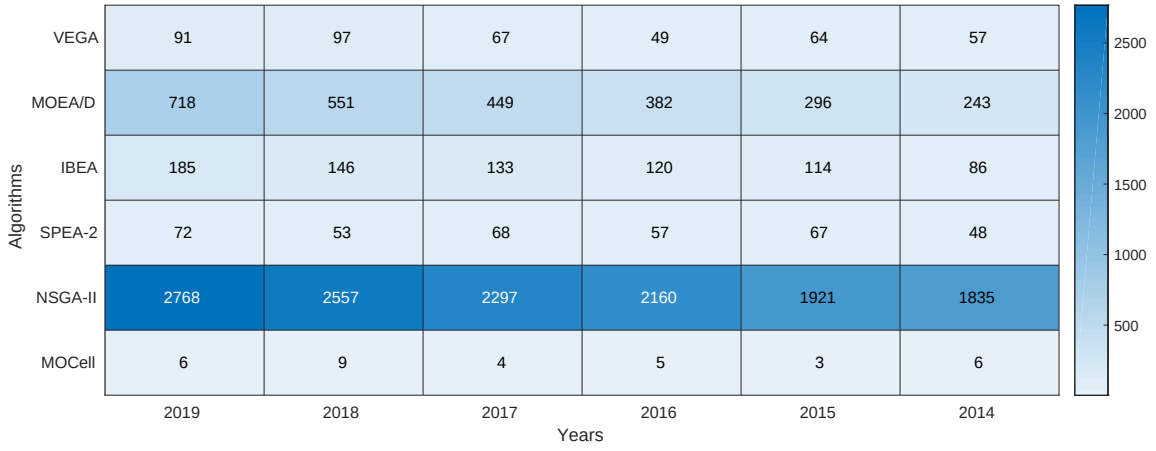

Figure 2.10. Analysis of recent scientific publications on multi-objective evolutionary-methods.

The initial attempt made on PSO to deal with MOOPs is MOPSO [97]. MOPSO is proposed as a Pareto-based approach and it makes a slight modification on the search operators in the single-objective PSO algorithm. It uses an external archive (called *repository*) to store non-dominated solutions. These solutions are used for calculation of particles' velocity. MOPSO is extended in (E-MOPSO) [98] to also handle with the constrained MOOPs and to further improve the exploration capability of original MOPSO. The introduced constraint handling mechanism in the E-MOPSO works as follows: among the two solutions; *i*) the dominated one is simply selected if two solutions are feasible, *ii*) the feasible one is selected if only the other is infeasible, *iii*) the solution having the least constraint violation is selected without

considering their dominance if both solutions are infeasible. The mutation operator in E-MOPSO, however, is very straightforward and it randomly chooses one parameter to mutate then allows a particle to search in a single dimension by shrinking the search range through an iteration-dependant formula.

Another PSO-based multi-objective optimization algorithm is Pareto-based OMOPSO [99]. OMOPSO considers the Pareto-dominance and crowding factor of particles in order to select them as leaders in an introduced external archive, which avoids a quick increase in the size of leader set. In addition, similar to E-MOPSO, OMOPSO employs mutation to increase the ability of global search in the algorithm. There are two mutation types introduced in OMOPSO: uniform and non-uniform representing, respectively, that keeps the variability range of a constant and decreases it over time. OMOPSO divides the population into three sub-populations of the same size to implement the mutation. While a sub-population is not exposed to mutation, the remaining two sub-populations are exposed to uniform and non-uniform mutations.

TV-MOPSO was introduced in [100]. Similar to OMOPSO, TV-MOPSO employs a strategy that relies on a crowding distance measurement made on the particles in the external archive in order to select the best particle found so far. In addition to that, the inertia weight and coefficient factors proposed in the standard PSO algorithm were modified to better balance global and local search. To achieve that, a time-dependant scheme was applied on these parameters. Finally, a new mutation strategy was also adopted in TV-MOPSO.

SMPSO [101] is an extension of OMOPSO. While the Pareto-based SMPSO has the same framework as OMOPSO, it modified the velocity calculation formula introduced in single-objective PSO. The motivation here is to avoid particles to have extreme velocities, since higher/extreme velocities leads particles to position at the limits of the sarch space, which does not contribute to the search. To do that, SMPSO adopts *constriction coefficient*, originally proposed in [102], and *velocity constriction*. The effect of restriction in the velocity is validated through the experiments.

Another PSO-based approach that relies also on constriction coefficient is MAPSO [103]. Using the standard search operations in single-objective PSO, MAPSO integrates the concept of Multiagent Systems (MAS), which represents a system where the agents interact and work together to achieve goals, and the PSO algorithm. MAPSO first maps the search space to a lattice- or cellular-like structure. Every agent in MAPSO, which represents a particle in PSO, is located fixed on a lattice point and competes with its neighbors, the agents at the north, south, west, and east lattice points. After competition, an agent can survive if it Pareto-dominates its neighbors; otherwise it is replaced with a new agent created with respect to the winner neighbor.

MPSO/D [104] is a multi-objective optimization algorithm which first decomposes the space into sub-regions as done similarly in DN-PSO. MPSO/D initially assigns each particle to a sub-region through the following proposed strategy: *i*) a particle closest to the center of a sub-region is selected if there is no particle in this sub-region, *ii*) a non-dominated particle is selected in case where two or more particles are in a sub-region (the closeness to the center of the region is considered in case where two or more non-dominated particles exist). After assignment of the particles to the sub-regions, MPSO/D employs a crossover operator to update the positions of the particles.

VEPSO [105] is inspired by the VEGA algorithm. Therefore, it performs the search with multiple swarm populations. As in VEGA, each swarm population in VEPSO is responsible for solving a different objective of the problem and evaluated by a single objective. The search operators introduced in single-objective PSO are separately applied for every swarm population. Interaction between swarm populations was also incorporated in VEPSO through exchanging the best experiences so that a generic model is generated for all the objectives.

DN-PSO [106] is another PSO-based multi objective optimization algorithm. Similar to VEPSO, DN-PSO also deals with objectives separately. However, the swarm population is not split into sub-populations in DN-PSO; instead, the whole population is responsible for handling the MOOPs in a way that every objective is dealt with sequentially. In addition to

that, a new dynamic neighborhood strategy was proposed in DN-PSO. Calculating the historical best positions of particles, this strategy attempts to find Pareto-dominant solutions as much as possible. DN-PSO was then extended in [107] to reduce the computational complexity of DN-PSO and to save the non-dominated solutions in a memory by introducing the external archive to the algorithm.

S-PSO was proposed in [108]. It introduced a generalized *sigma* method for any archive-based multi-objective PSO. Main purpose of S-PSO is to efficiently discover a global best particle (as a guide for local search) from the archive separately for each of the current particles. To do that, by considering its positions in the objective space, a sigma value is calculated and assigned to every particle both in the current population and in the archive. Each particle in the population then selects its 'leader' particle depending on the closeness of the sigma value which represent how close a particle to the selected archive member in the objective space. The efficient closeness calculation through the sigma method ensures an admissible computational cost. So, the proposed sigma method can also be adapted to an algorithm that relies on the neighborhood measurement such as DN-PSO for the sake of better computational cost.

The first extension effort on the ABC algorithm for solving MOOPs was made in VEABC [109], which is inspired by the VEGA [92] and VEPSO [105] algorithms. VEABC also splits the whole population into sub-populations which are responsible to handle a different objective of the problem. The update of the position of artificial bees in VEABC depends on the the previous positions of them and their communication between the sub-populations, which ensures an achievement of a generalized solution for all the objectives of the problem.

MOABC [110, 111] is another ABC-based multi-objective optimization algorithm. The search framework of single-objective ABC was generally preserved in MOABC except the

fitness proportionate selection scheme that is used by onlooker bees. So, the single-objective-based selection that is used in basic ABC algorithm was replaced with a Pareto-dominance-based selection approach in MOABC. In addition to that, an external archive was also proposed in MOABC to save the Pareto-dominant solutions in the memory. ϵ-dominance method proposed in [112] is used in maintenance of solutions in archive. MOABC decomposes objective space into the ϵ-sized boxes. Second, the dominated boxes, and hence dominated solution(s) in those boxes, are eliminated. Then, boxes are re-formed such that every box has no more than one non-dominated solution. In the case that two or more solutions are in the same box, a solution closer to the left corner of the box remains in the box while others are removed from the box. This method ensures that the remaining solutions are non-dominated.

A-MOABC/PD, A-MOABC/NS, and S-MOABC/NS were proposed as three different implementations of single-objective ABC in [113] to extend the standard algorithm to handle MOOPs. As in MOABC, the search framework employed in all variants here is the same as the standard ABC algorithm.However, they differs from both each other and the standard ABC algorithm in some search steps. In A-MOABC/PD, Pareto-dominance based fitness assignment strategy which includes the Pareto rank, crowding distance, and the like was employed.

Every time after a new source is found by an artificial bee, the Pareto-dominance-based greedy selection is applied. That's why this variant is regarded as 'asynchronous'. In A-MOABC/NS, however, applies the standard fitness assignment strategy which requires a straightforward objective evaluation. As opposed to A-MOABC/PD, A-MOABC/NS does not apply greedy selection; but a 'non-dominated sorting method' that was proposed in NSGA [16]. Finally, S-MOABC/NS uses the same procedure as A-MOABC/NS with an only difference of the time where the non-dominated sorting method is applied. While this method is applied every time a new solution is found in A-MOABC/NS, it is applied here after the search phases of employed and onlooker bees. That's why, it is regarded as 'synchronous'.

NSABC, another non-dominated sorting based approach, was proposed in [114]. As common in most of the multi-objective approaches, NSABC also introduced an external archive for keeping non-dominated solutions. In addition to that, it modified the search behaviors of employed and onlooker bees in order to avoid slow and premature convergence problems. Therefore, a 'dynamic mutation operator' is included in the employed bee phase in NSABC and it is applied when the generated new solution does not dominate. Moreover, the fitness assignment strategy also used in [113] is used in the onlooker bee phase in NSABC. The maintenance of archive is made through the non-dominated sorting method of NSGA-II.

eMOABC algorithm was proposed in [115]. eMOABC employs an external archive and applies a slight modification to the main search operators governed by employed and onlooker bees in the basic ABC algorithm by including the effect of *elite* individuals. An elite individual in eMOABC is a solution in the external archive that is chosen with respect to the crowding distance measurement method of NSGA-II. The crowding distance measurement method was also used in eMOABC for the maintenance of archive.

dMOABC was presented in [116]. dMOABC divides the search space into two sub-spaces and constructs three colonies. While two colonies (called as basic colonies in dMOABC) perform search independently on the sub-spaces, the last population (called as synthetic colony) performs a search throughout the search space. This adoption ensures a good diversity in the algorithm. An external archive was also employed in dMOABC to store non-dominated solutions. The communications between colonies is indirect and is done through the synthetic colony.

Another ABC-based approach that uses a multi-colony model similar to dMOABC is DMCMOABC [117]. Differently from dMOABC, more than three colonies can be generated in DMCMOABC (it was set to 10 in [117]) such that each colony performs the search task throughout the whole space independently, which is also another difference from dMOABC. Every colony in DMCMOABC applies search operators as in the basic ABC algorithm simultaneously. Non-dominated solutions from each colony are saved in the archive. Information

exchange between the colonies was performed through the archive in DMCMOABC by replacing the elite individual, which is chosen with respect to the crowding distance method, with the worst individual from a randomly selected colony.

As stressed earlier, PSO and ABC have become a very popular, may be the most, algorithms that have been used to deal with the MOOPs in the literature mostly due to their easy-to-implement structures and satisfying performances on different types of problems including the discrete/combinatorial problems. PSO- and ABC-based multi-objective optimization algorithms reviewed here are among the most popular ones. Refer, respectively, to [118, 119] for a detailed review of PSO- and ABC-based multi-objective optimization algorithms. Apart from them, there are also multi-objective optimization algorithms that were built on other swarm-based single-objective optimization algorithms to be applicable for MOOPs. Some of them have also been outlined in the following.

A generic ACO-based multi-objective optimization algorithm was proposed in [120]. The proposed algorithm (m-ACO) is characterized with two parameters m-ACO($\#col$,τ): $\#col$ defines the number of ant colonies while τ defines the number of pheromone structures. Four different variants were examined in [120]: m-ACO$_1$($m+1$,m), m-ACO$_2$($m+1$,m), m-ACO$_3$(1,1), and m-ACO$_4$(1,m). In m-ACO$_1$ and m-ACO$_2$; the first m ant colonies are responsible to solve each single-objective independently of the problem having m objectives while the $m+1$th colony aims at optimizing all the objectives. The communications between the m colonies and $m+1$th colony is performed through the pheromones. The way to use of pheromone by $m+1$th colony differs m-ACO$_1$ from m-ACO$_2$. In m-ACO$_3$, only a single colony with a single pheromone structure is employed. The agents in this colony optimize all the objective simultaneously. As in the single-objective ACO algorithm, the pheromone level of every agent is also updated in m-ACO$_3$ but in a different way by considering the dominance of the agent. The final version of generic multi-objective ACO algorithm, m-ACO$_4$, employs a single colony and m pheromone structures. At every cycle, a randomly selected objective is optimized. Pheromone level of the agents in the colony is updated for every objective separately.

MOBA [121] scalarizes the multiple objectives of a problem by weighting each objective such that the sum of the weights is equal to 1. Instead of prioritizing some objectives, every objective was considered evenly in MOBA. Therefore, a randomly generated vector was used in MOBA to determine weights. The search framework of MOBA was kept same as the single-objective BA. However, instead of greedy selection used in BA, MOBA accepts a new solution only if it Pareto-dominates the previous solution.

Firefly algorithm was also extended to solve multi-objective optimization problems as (MOFA) in [122]. Pareto-based MOFA algorithm employs the same search framework as the single-objective FA algorithm. It occasionally scalarizes the multiple objectives as done in the MOBA algorithm then performs a local search around the best individual when non-dominated solution can be found before reaching the cycle. Only the non-dominated solutions found at a cycle are passed to the next cycles in MOFA.

MOGSA was proposed to solve MOOPs in [123]. To adapt single-objective GSA algorithm to the multi-objective concept, MOGSA incorporates *uniform mutation operator* and *an external archive*. Similar to the MOABC algorithm, objective space is divided into a number of equal divisions in MOGSA to update the solutions in archive.

MOGWO was proposed in [124]. As explained earlier, social hierarchy and hunting strategy are the behaviors of grey wolves that were modeled in the single-objective GWO algorithm. These behaviors were also used in MOGWO so the search framework is similar to the GWO algorithm. However, the leader selection strategy is slightly different from the single-objective GWO algorithm. The proposed leader selection strategy in MOGWO is governed by non-dominated (Pareto-optimal) solutions saved in an external archive, which was introduced in MOGWO.

MOCS [125] is an extension of the single-objective CS algorithm. The search behavior of MOCS is similar to single-objective CS algorithm. In order to adapt CS algorithm to the multi-objective concept only Pareto-dominance-based greedy selection was introduced in MOCS to accept developed solutions. As opposed to the most the multi-objective algorithms,

MOCS does not maintain an external archive; rather it ranks the solutions with respect to their Pareto-dominance and returns only the non-dominated solutions.

MOTLBO, which extends the single-objective TLBO algorithm, was proposed in [126]. MOTLBO differs from TLBO in only a way to select teacher which represents the fittest solution in TLBO. While the fittest solution is obtained by considering only a single-objective in the TLBO algorithm, it is non-dominance of a solution which determines whether it is the fittest solution or not in MOTLBO. The fittest solutions are kept in an external archive in MOTLBO; the teacher is chosen from the archive considering the crowding distances, which is originally proposed in NSGA-II algorithm, of solutions. The solution having the highest crowding distance is chosen as the teacher in MOTLBO algorithm.

The total number of scientific researches based on multi-objective swarm-based approaches has been given in Figure 2.11.. The total number of publications on this research area within the last six years has been given in Figure 2.12.. It is worth noting here that only the multi-objective swarm-based approaches having the highest impact on the literature have been included into the figures for the sake of better readability. These figures show that the PSO-based multi-objective algorithms have been mostly adapted in the literature. Among them, MOPSO has become the well-studied approach and its impact is increasing year-by-year (refer to Figure 2.12.).

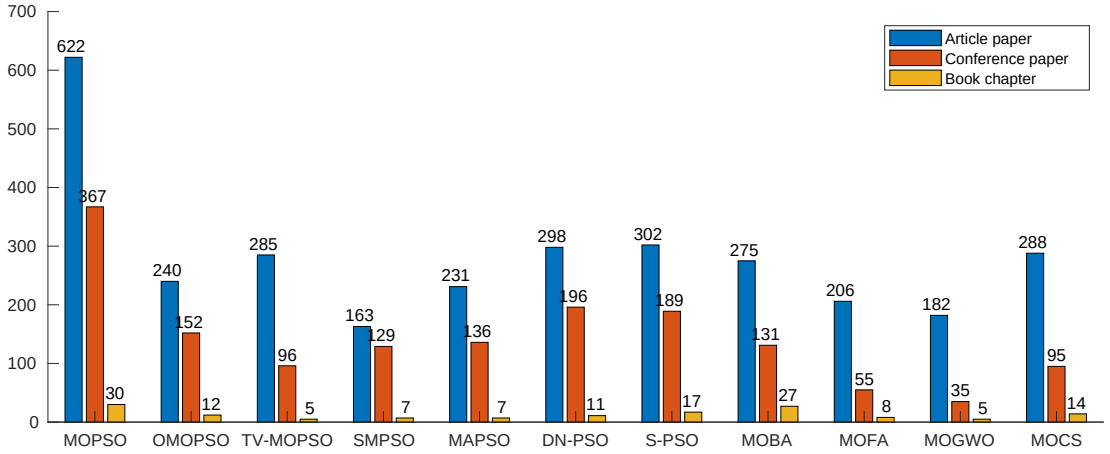

Figure 2.11. Total number of scientific publications on multi-objective swarm-based methods.

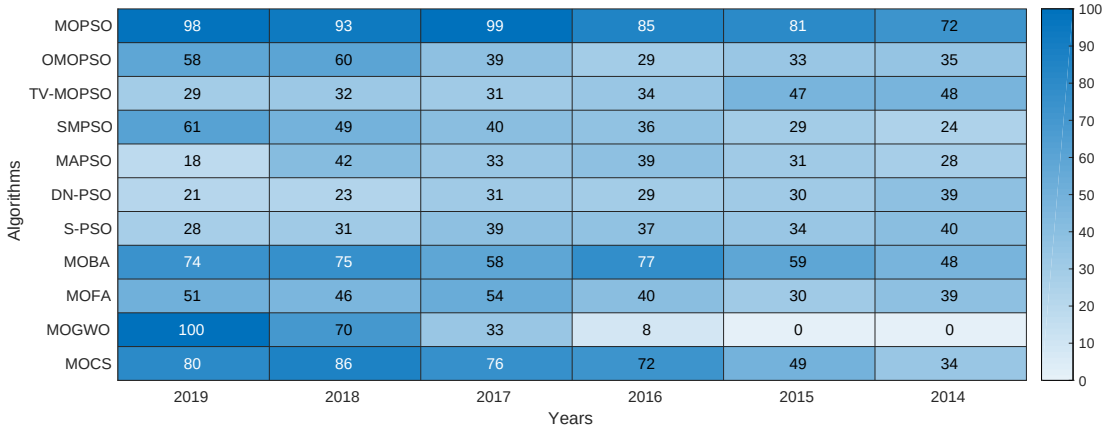

Figure 2.12. Analysis of recent scientific publications on multi-objective swarm-methods.

The distribution of application areas of multi-objective evolutionary-, and swarm-based algorithm have been given in Figure 2.13.. From the figure, it can be concluded that *computer science*, *engineering*, and *mathematics* are the dominant subject areas comprising a very large portion of all areas for all types of heuristic algorithms, which is the case same as single-objective algorithms (refer to Figure 2.8.).

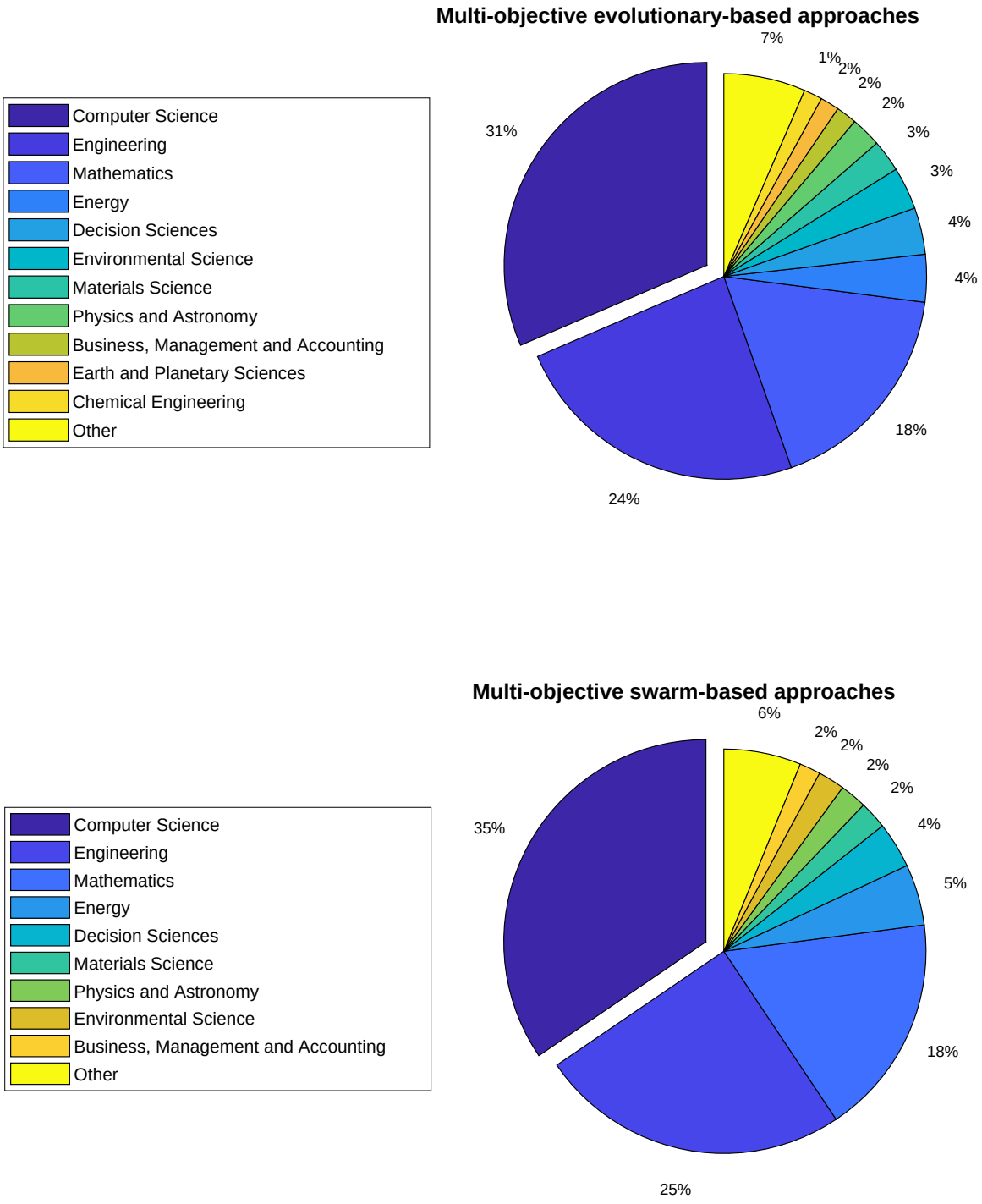

Figure 2.13. Distribution of application research areas of multi-objective heuristics.

3. ELECTRIC FISH OPTIMIZATION ALGORITHM

3.1. Electric Fish

Electric fish constitute only a fraction of all fish species ($\sim$1.2%, corresponding to 350 electric fish species out of more than 30,000 total fish species). They live in muddy water where their visual sense are very restricted and most of them are nocturnal. However, they possess a species-specific capability, *electrolocation*, which is the distinguishing sense capability that electric fish rely upon to locate objects such as prey and obstacles. Electric fish are categorized as strongly and weakly electric fish, depending on the strength of the electric field they generate. Strongly electric fish mostly use the electrolocation capability for offensive purposes and their field intensity ranges from 10 to 600 V, which is sufficient to stun their prey. In contrast, weakly electric fish only generate electric fields of intensity between a few hundred millivolts and a few volts, which is utilized to navigate, communicate, detect objects, etc. [127].

Electric fish have an electric organ containing special electric disc-like cells (called *electrocytes*), which are used to generate an electric field. This organ is generally located at the tail of the body. Each of the electrocytes synchronously receives the command signals from the brain through electromotor neurons (EMN) to 'fire'. The electrocytes are lined up in a sequence and each can produce up to 0.15 V (see Figure 3.1.).

The simultaneous excitation of *electrocytes* is called as EOD in the literature. EOD is characterized by its *frequency* and *amplitude*. The EOD frequency is inversely proportional to the interval between two consecutive electric signals. It is not random or irregular but highly affected by the presence of a novelty (sharp increase) (see Figure 3.2.A.) and the distance to the stimulus (regular increase) (see Figure 3.2.B.). The EOD amplitude, however, is proportional to the fish's size, owing to the electrocytes that are added during growth, and it decays with the inverse cube of the distance from the body surface [127, 128].

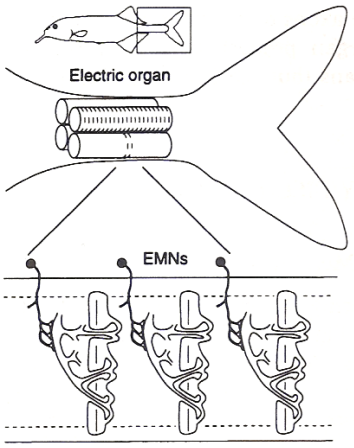

Figure 3.1. Electrocytes in electric organ at the tail of fish. Modified after Moller [127].

Depending on the EOD activity, electrolocation is examined under two categories: *active* and *passive*. In active electrolocation, electric fish generate EOD and sense their near surroundings through the modulations in the electric field. The effective range of active electrolocation is very limited, but very helpful to fish living in dark and complex environments for finding and identifying their prey. In passive electrolocation, electric fish can perceive and react to electric signals of external sources through their electroreceptors. Contrary to active electrolocation, the effective range of passive electrolocation is much wider, which allows them to locate objects at distances exceeding the active electrolocation range and to communicate with other fish.

Electric fish are both electrogenic and electroreceptive, which means they have capability to perceive electric currents emanated either from their conspecifics' organs or from organisms living in the same environment, as well as to generate an electric field, which enables them to communicate with each other through electrical signals. By using this capability, called *electrocommunication*, electric fish can exchange information about *i)* their identity, gender, status in the hierarchy; *ii)* their physiological or motivational state; and *iii)* aspects about their environment, presence of food, etc. [127].

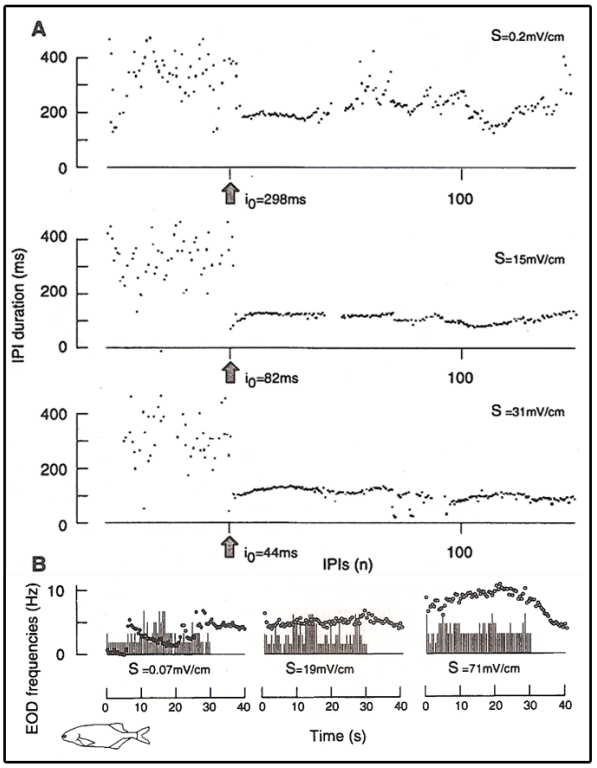

Figure 3.2. (A) Inter pulse interval (IPI) duration of *Brienomyrus niger* based on the presence of novelty within three different distances (0.2 mV/cm, 15 mV/cm, and 31mV/cm) and (B) EOD frequencies of *Marcusenius* (dots represent the mean frequency) based on different stimulus intensities. The stimulus are located at *i)* outer limit of the fish active space (0.07 mV/cm) *ii)* somewhere in the middle of space (19 mV/cm) *iii)* close within a few centimeters (71 mV/cm). Modified after Moller [127].

3.2. Swarm Intelligence in Electric Fish

The term swarm is particularly used for aggregated animals such as fish schools, bird flocks, and colonies of insects such as ants, bees, and termites. Swarm intelligence, a form of AI, is the emergent collective intelligence of groups of simple swarm agents [129]. It stems from the interaction of agents with their neighbors. In swarm intelligence, there is no centralized (global level) control structure that dictates how an agent should behave, but the rules that specify the interactions among the swarm agents are executed at a purely local level. The first use of the expression 'swarm intelligence' dates back to the 1980s in the context of cellular robotic systems [129]. Since then, researchers in the field of optimization have developed several swarm-based algorithms. In the 1990s, the success of ACO and PSO in solving

different types of problems has attracted researchers to the development of new swarm-based algorithms.

Self-organization is a key feature of a swarm that results in global level (macroscopic level) response by means of low-level interactions (microscopic level) [129, 130]. For example, the emerging structure of foraging (macroscopic level) in ants includes organized networks of pheromone trails (microscopic level). The following four characteristics are the basic ingredients upon which self-organization relies [129]: Positive feedback, negative feedback, fluctuation, and multiple interactions. Each characteristic is introduced and explained from the perspective of electric fish as follows:

1. *Positive feedback:* This promotes the creation of convenient structures. Examples of positive feedback include recruitment of ant or bee species to a food source depending on the trail laying or waggle dance, respectively.

   An electric fish that is close to a stimulus or prey source generates an electric field with higher EOD frequency, which means that it will search in its locality through its active electrolocation capability. As the fish begins to produce electric more often, it attracts other fish.

2. *Negative feedback:* This counterbalances the positive feedback, and it is required to avoid saturation, exhaustion, and competition.

   An electric fish far away from a stimulus or prey source decreases its EOD frequency, switches to silent mode, and thus no longer attracts its conspecific.

3. *Fluctuations:* This enables the discovery of new solutions, and in a swarm it is crucial for creativity and innovation. For example, foragers in an ant swarm follow trails with a certain error and may get lost in the colony; the lost foragers can find new and unexplored new food sources.

   A fish perceives electrical stimuli not only from its conspecific's electric organ, but also from other aquatic organisms.

4. *Multiple interactions:* These enable some agents in the swarm to receive information originating from other agents for information to spread throughout the swarm.

   Electric fish communicate with each other through active and passive electrolocation and broadcast messages to each other about their surroundings: *electrocommunication*.

3.3. EFO Algorithm

All heuristics use a common framework. However, the usage of search operators depends on the model, which is what differentiates heuristics in the literature [9, 131]. EFO is no exception and uses very simple search operators. The details of the algorithm will be presented in the subsequent sections. At first, the assumptions that transform the behavior of electric fish into the heuristic algorithm, in accordance with their true nature, are introduced. The first assumption is that there is an infinite food source in the search space, where one food source is recognized as the best source. Electric fish, corresponding to the individuals in the algorithm, are located somewhere in the space and carry their location information. The quality of each individual is determined by its distance to the location of the best source, and hence the problem at hand becomes a problem of finding the best quality food source. Another assumption, which is inspired by natural evolution, is that fish possessing high-quality resources over a long period of time grow faster and thus are able to produce higher-amplitude electric signals than the others [132].

The intelligent behaviors of electric fish are modeled as follows:

- *Active electrolocation:* Electric fish produce more frequent electric signals by means of their electric organs as nearer they get to the best food source (Figure 3.3.-A and Figure 3.3.-C)).

  Active electrolocation has a very limited range, and therefore it has been used to ensure the local search through individuals with better fitness values, such that these individuals could be better able to search their vicinity, as in electric fish.

- *Passive electrolocation:* Other fish do not generate an electric field, but rely on electric signals emanated either from their conspecific's electric organ or from animate organisms (Figure 3.3.-B).

  Because passive electrolocation has a wider range than active electrolocation, in the algorithm, it has been used to balance active electrolocation and to ensure the global search by enabling individuals, particularly those of poor fitness, to explore distant spaces.

- *EOD frequency:* In nature, the frequency of electric field generation depends on the closeness to the source. The fish closest to the best source are expected to produce electric field more frequently than others.

  EOD frequency has been used in EFO as a key to determine the role of each individual at time t, because it is an indicator used by electric fish to find out which individuals are in the vicinity of a better food source. As in nature, individuals with higher frequency employ active electrolocation, and others use passive electrolocation.

- *EOD amplitude:* The amplitude of electric field depends on the growth (and thus the size) of the fish, and it determines the effective range of the electrical stimuli.

  In the EFO algorithm, owing to its ability to determine electric fields' domination, EOD amplitude has been used to determine the effective range in local search and the probability of individuals' being sensed in global search.

3.3.1. Population initialization

Initially, the electric fish population (N) (henceforth called individuals) are randomly spread through the search space considering the boundaries.

$$x_{ij} = x_{\min j} + \phi(x_{\max j} - x_{\min j}) \tag{1}$$

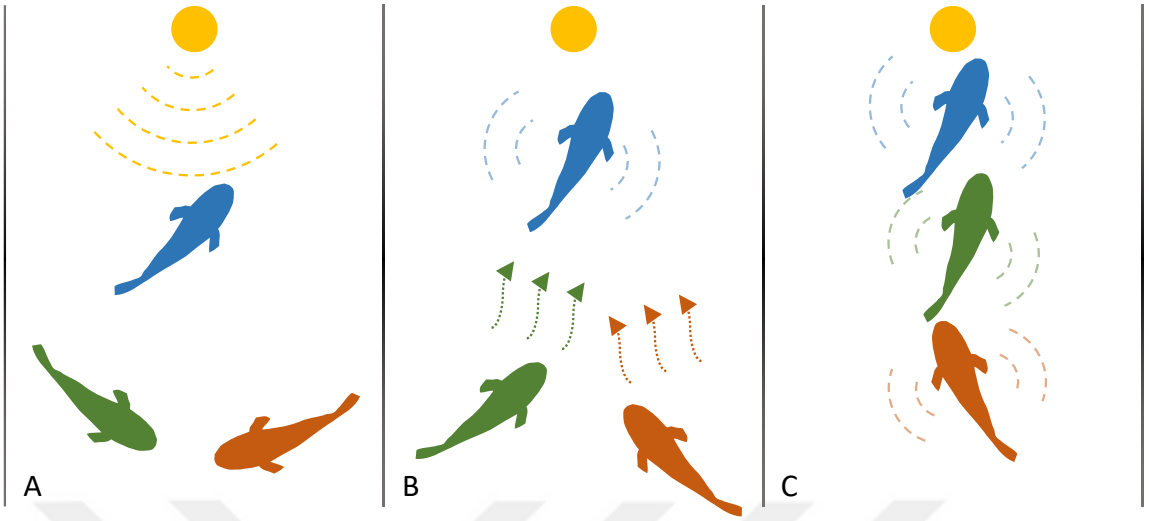

Figure 3.3. Prey detection and communication in a fish swarm. A) Fish (colored blue) perceives the signals from the source. B) It switches into active mode, generating signals with a higher frequency, and broadcasts its surroundings. C) Other fish (colored green and brown) in passive mode sense the conspecific's signal and move toward the source.

where x_{ij} represents the position of the ith individual in the population of size $|N|$ ($i = 1, 2, ..., |N|$) in the d-dimensional search space. $x_{\min j}$ and $x_{\max j}$ are the lower and upper boundaries for dimension $j \mid j \in 1, 2, ..., d$, respectively. $\phi \in [0, 1]$ is a random value drawn from a uniform distribution.

Individuals in the population move around the search space through their active or passive electrolocation capability, just after the initialization phase. The frequency plays a key role in the EFO algorithm to balance exploration and exploitation, and is used to determine whether an individual will perform active or passive electrolocation. It enforces better individuals (in active mode), which are most likely to be vicinity of promising regions, to exploit their neighborhood, and it leads other individuals (in passive mode) to explore the search space so that they discover new regions, which is essential for multimodal functions.

In the EFO algorithm, as in nature, individuals with a higher frequency employ active electrolocation, and others use passive electrolocation. The frequency value of an individual ranges from a minimum value f_{min} to a maximum value f_{max}. Because the frequency value of an electric fish at time t is strictly related to its closeness to the food source, an individual's frequency value (f_i^t) is derived from its fitness value:

$$f_i^t = f_{min} + \left(\frac{fit_{worst}^t - {fit_i}^t}{fit_{worst}^t - fit_{best}^t} \right) (f_{max} - f_{min}) \tag{2}$$

where fit_{worst}^t and fit_{best}^t are respectively the worst and best fitness values obtained from individuals in the current population at iteration t, whereas fit_i^t is the fitness value of the ith individual at iteration t. In this study, as the frequency value is used for a probability calculation, f_{min} and f_{max} are set to 0 and 1, respectively.

Apart from the frequency, electric fish have also amplitude information. This determines the active range of a fish while actively electrolocating, and the probability of being perceived by other passively electrolocating fish, as the strength of electric field decays with the inverse cube of distance.

The amplitude of an individual depends on the weight of the individual's previous amplitudes (α in Equation 3), and therefore it may not change sharply. The amplitude value of the ith individual (A_i) is calculated as follows:

$$A_i^t = \alpha A_i^{t-1} + (1 - \alpha) f_i^t \tag{3}$$

where $\alpha \mid \alpha \in [0, 1]$ is a constant value that determines the magnitude of the previous amplitude value. In EFO, the initial amplitude value of the ith individual is set to its own initial frequency value f_i.

The frequency and amplitude parameter values of a fish (or individual) are updated based on the proximity of the fish to the best prey source. In every iteration of the algorithm, the population is divided into two groups based on the frequency value of each individual: individuals performing active electrolocation (N_A) and those performing passive electrolocation (N_P) (i.e., $N_A \cup N_P = N$). Because the frequency of an individual is compared with a uniformly distributed random value, the higher the frequency value an individual has, the more likely it is to perform active electrolocation. The search is then performed by individuals in N_A and N_P in a parallel manner.

3.3.2. Active electrolocation

The biologically effective range of active electrolocation is limited to approximately half of the fish size [127], and the fish is unable to perceive prey outside this range. In other words, this capability allows fish to locate any food source in their immediate vicinity. The local search or *exploitation* capability of EFO is based on these characteristics of active electrolocation.

In EFO, all individuals in active mode (performing active electrolocation) move around the search space by modifying their traits. However, only one parameter that is randomly chosen is allowed to be modified, lest the individuals move too far away from the promising region.

The movement of the ith individual may vary depending on the existence of neighbors inside its active range. If no neighbor exists (Figure 3.4.-A), it performs a *random walk* throughout its range, otherwise; it chooses one neighbor randomly and changes its location depending on this neighbor (Figure 3.4.-B). The optional search contributes an 'explore first, exploit later' approach in EFO. As the individuals are initially far away from each other, they are less likely to belong to each other's active range. Therefore, it is very likely for active mode individuals to initially search randomly in their neighborhood and then start to exploit one of the closest neighbors as the iteration proceeds.

The active range of the ith individual (r_i) is determined by its own amplitude value (A_i), as in nature. The active range calculation in EFO is given in Equation 4.

$$r_i = \left(x_{\max j} - x_{\min j}\right) A_i \tag{4}$$

To find neighboring individuals ($S \mid S \subset N$) in the sensing/active range, one needs to measure the distance between the ith individual and the rest of the population (i.e., $N \setminus \{i\}$). The distance between individuals i and k is determined by using the Cartesian distance calculation:

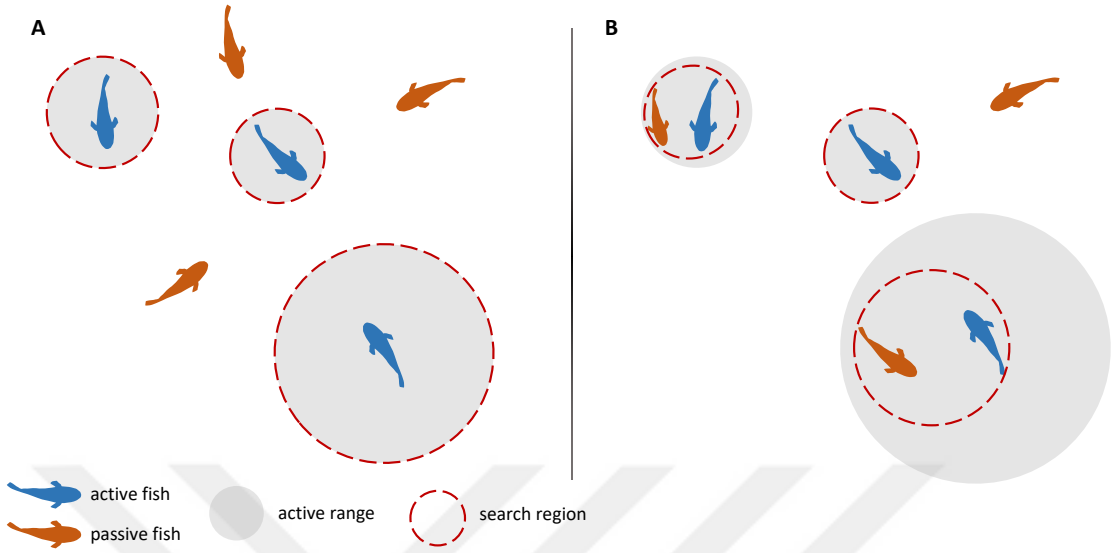

Figure 3.4. Local search demonstration of EFO. Individual in active mode either moves somewhere inside its active range (A) or search region (B) depending on the existence of neighbors in its vicinity.

$$d_{ik} = \|x_i - x_k\| = \sqrt{\sum_{j=1}^{d} \left(x_{ij} - x_{kj}\right)^2} \tag{5}$$

In the case where at least one neighbor exists in the active sensing area, EFO uses Equation 6; otherwise (i.e., $S = \emptyset$), Equation 7 is employed.

$$x_{ij}^{cand} = x_{ij} + \varphi \left(x_{kj} - x_{ij}\right) \tag{6}$$

where k represents a randomly chosen individual from the neighbor set of the ith individual (i.e., $k \in S$ and $d_{ik} \leq r_i$)

$$x_{ij}^{cand} = x_{ij} + \varphi r_i \tag{7}$$

where $\varphi \in [-1, 1]$ in Equation 6 and 7 is a random number generated from a uniform distribution and x_{ij}^{cand} represents the candidate location of the ith individual.

3.3.3. Passive electrolocation

As opposed to active electrolocation, the sensing distance does not depend on the ith individual and exceeds beyond the range of active electrolocation, as in nature. This is why the passive electrolocation capability fulfills the requirements of global search or *exploration* mechanism of the proposed EFO algorithm.

As mentioned before, the perceiving probability of a signal is directly proportional to its own amplitude value and the distance to a target individual. Individuals in the passive mode choose other individuals in active mode that propagate electrical signals depending on a probability, and then change their locations (see Figure 3.5.-A).

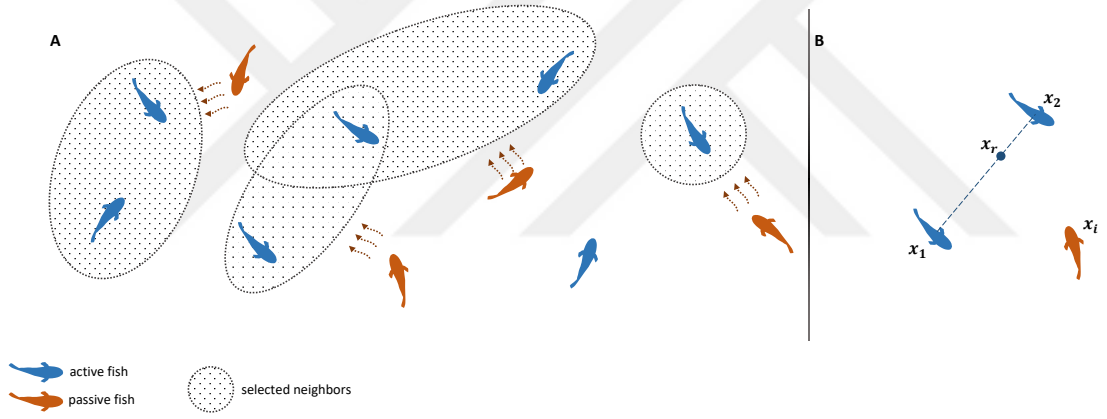

Figure 3.5. Global search demonstration of EFO.

The probability of the kth individual in active mode (i.e., $k \in N_A$) being perceived by the ith individual in passive mode (i.e., $i \in N_P$) is calculated using Equation 8.

$$p_k = \frac{A_k / d_{ik}}{\sum_{j \in N_A} A_j / d_{ij}} \tag{8}$$

As seen from the equation, beside amplitude-based selection, passive electrolocation employs a distance-based selection mechanism as in Gravitational Search Algorithm [9] and Firefly Algorithm [52]. It is important to note here that the probabilistic neighbor selection forces passive mode individuals to perform exploration before the exploitation. Therefore,

passive mode individuals choose neighbor individuals at the initial iterations, because the distance is the dominant factor, which causes the selection of poor individuals and which contributes to the discovery of new regions of the search space. Amplitude becomes the dominant factor as the iteration proceeds (because the distance between EFO individuals approaches zero), which causes the selection and local search of the best individuals.

Using various strategies, such as roulette wheel selection as employed here, K individuals are chosen from N_A based on Equation 8, and then a reference location (x_{rj}) is determined based on Equation 9 (see Figure 3.5.-B). The new location is then generated through Equation 10.

$$x_{rj} = \frac{\sum_{k=1}^{K} A_k x_{kj}}{\sum_{k=1}^{K} A_k} \tag{9}$$

$$x_{ij}^{new} = x_{ij} + \varphi \left(x_{rj} - x_{ij} \right) \tag{10}$$

Contrary to the search in active electrolocation, more than one parameter can be modified, such that individuals explore the search space much faster. However, albeit rarely, there might be a case in which an individual with a higher frequency performs passive electrolocation. In such a case, that individual would lose its location information completely. This situation is not expected or preferred due to the individual of being located in a promising region. For avoiding such cases, EFO applies Equation 11 in order to determine which parameters will be modified. With this equation, the probability for such an individual to modify its whole trait is significantly lowered.

$$x_{ij}^{cand} = \begin{cases} x_{ij}^{new} & rand_j(0,1) > f_i \\ x_{ij} & else \end{cases} \tag{11}$$

where $rand_j(0,1)$ is a uniform random number generated for the jth parameter.

The final step of passive electrolocation is to modify one parameter of the ith individual using Equation 12 to increase the probability of a trait being changed.

$$x_{ij}^{cand} = x_{\min j} + \varphi(x_{\max j} - x_{\min j}) \quad rand(0,1) \leq rand(0,1) \tag{12}$$

where $rand(0,1)$ is a random number generated from a uniform distribution.

If the jth parameter value of the ith individual exceeds the boundaries of the search space, it is relocated to the boundary of the space that it exceeds:

$$x_{ij}^{cand} = \begin{cases} x_{\min j} & x_{ij}^{cand} < x_{\min j} \\ x_{ij}^{cand} & x_{\max j} > x_{ij}^{cand} > x_{\min j} \\ x_{\max j} & x_{ij}^{cand} > x_{\max j} \end{cases} \tag{13}$$

The simplified pseudocode of the EFO algorithm and its detailed form are given in Algorithms 1 and 2, respectively. The active electrolocation phase of EFO is presented in lines 7-16, and passive electrolocation is presented in lines 18-23 in Algorithm 2. These phases determine the complexity of EFO. The complexity of active and passive phases is mainly governed by the distance calculation of the individuals (lines 9 and 18). Hence, the time complexities of the active and passive phases are $O(|N_A| \times |N|)$ and $O(|N_P| \times |N_A|)$, respectively. In addition, the algorithm has a complexity that is proportional to $O(|N|)$ for the fitness evaluation of the population (line 26). In short, for one iteration, the time complexity of EFO is $O(|N|)$ and $O(|N|^2)$ for the best (when $|N_A| = 1$) and worst cases (when $|N_A| = |N| - 1$), respectively.

3.4. Search Strategy

A fine trade-off between exploration and exploitation characteristics is very important for the overall efficiency and performance of an algorithm. It is less likely, or maybe impossible, for an algorithm to converge to the optimal solution in the case of too much exploration, which leads to performance degradation. Algorithms that mostly favor exploitation, however, may

Algorithm 1: Simplified pseudocode of Electric Fish Optimization algorithm.

```
Generate initial population;
Evaluate the quality of the individuals;
repeat
    Split the population into two subpopulations (N_A, N_P) depending on the frequency
      values of each individual (f);
    Perform active and passive electrolocation for the individuals belonging to N_A and N_P,
      respectively;
    Update frequency f and amplitude A values for each individual;
until termination criterion is met;
```

get stuck in a local optimum, even if they have fast convergence [133, 134]. In addition, an 'explore first, exploit later' approach should also be taken into consideration to further improve the optimization performance [10].

The proposed EFO enables well-balanced exploration and exploitation, owing to a metaphor that relies upon the very nature of electric fish. The search strategy of EFO fundamentally relies on three key factors: *frequency*, *amplitude*, and the *distance* among EFO individuals. A trajectory plot that captures the positions of EFO individuals (10 individuals in total) during 1000 iterations on two-dimensional *Michalewicz* function (F24 in Table B.1.) has been used to better explain the search strategy of EFO, as shown in Figure 3.6..

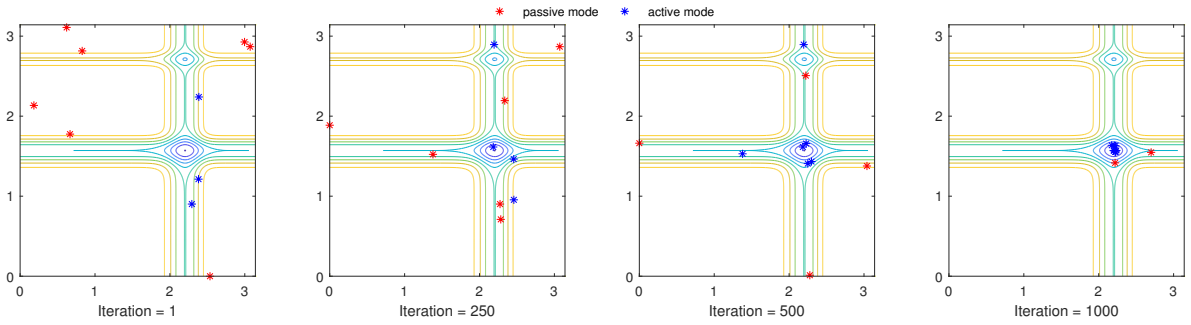

Figure 3.6. Trajectory plot tracing the positions of EFO individuals on two dimensional *Michalewicz* function.

The frequency-based mode determination in EFO individuals forces better individuals to perform exploitation and other individuals to perform exploration. As seen from the figure, individuals in active mode are closer to the promising regions (local and global optima)

Algorithm 2: Detailed pseudocode of Electric Fish Optimization algorithm.

```
Generate initial population N by Eq. 1;
Evaluate fitness value fit of the individual;
Calculate frequency f and amplitude A values of every individual by Eq. 2 and 3;
repeat
    foreach i ∈ N do
        if f_i ≥ rand then                          // active electrolocation phase
            Randomly choose one parameter j to be modified;
            Calculate active range (r_i) of ith individual;
            Calculate distance of ith individual to other individuals;
            Examine neighbor individuals (S) in the sensing area ;
            if S ≠ ∅ then
                Randomly choose one individual k in the active space;
                Modify jth parameter by Eq. 6;
            else
                Modify jth parameter by Eq. 7;
            end
        else                                        // passive electrolocation phase
            Considering p values (Eq. 8), probabilistically choose K individuals from
              actively electrolocating population N_A;
            Modify all parameters of ith individual by Eq. 9, 10 and accept only parameters
              to be modified considering Eq. 11;
            if rand(0,1) ≤ rand(0,1) then
                Determine one more parameter j to be modified;
                Modify jth parameter by Eq. 12;
            end
        end
        Check for boundaries and apply Eq. 13 for those exceeding search space;
        Evaluate quality of new source and accept if found better;
    end
    Update frequency and amplitude values of the population N;
until termination criterion is met;
```

throughout the search space, which enables them to behave more exploitatively, and passive mode individuals to discover new regions as they are too far away from the these regions.

Amplitude and distance, however, have a cooperative role in maintaining the 'explore first, exploit later' approach. As seen in the figure, the individuals are initially far away from each other and then get closer as the iteration proceeds. Therefore, individuals in active mode are more likely to perform random search in their vicinity at the beginning of the

search, as they are less likely to belong to each other's active range. They start to exploit one of the closest neighbors as the iteration proceeds. Individuals in passive mode, however, choose non-elite individuals at the beginning of the search because of the distance that plays a key role initially, which leads EFO to discover unseen regions. As the search reaches to the final iterations, the elite individuals are more likely to be chosen since distances among individuals approach to zero, which makes the amplitude dominant in the determination of neighbor individuals, which increases the exploitation capability of EFO.

The main differences between EFO and the competitor algorithms in terms of the search strategy are outlined as follows:

- EFO is a population-based algorithm, unlike SA and VS, and it makes use of multiple individuals to find an optimal solution in the search space, which makes EFO a better algorithm for solving complex problems.
- Evolutionary-based GA and DE rely on generic operators such as selection, crossover, and mutation, whereas EFO uses frequency, amplitude, and distance to control diversity throughout the search.
- PSO makes use of the particle's and population's best in the previous iterations; EFO, however, takes only the current population into consideration. Moreover, it does not consider only *elite individuals* to ensure better exploration capability.
- Onlooker individuals in ABC take only one neighbor individual at every iteration, whereas in EFO, passive mode individuals rely on more than one neighbors to reduce the likelihood of being trapped in local optima.
- Unlike to ABC and PSO, EFO initially has a much lower tendency to choose individuals with better fitness values, owing to the inclusion of distance in the algorithm to discover the search space efficiently, which also maintains the 'explore first, exploit later' approach.

There are also considerable differences between EFO and competitor algorithms in terms of the optimization performance. The superior search capability of EFO with respect to the competitor algorithms has been proven through the experimental findings, which are presented in the following section.

4. EXPERIMENTS ON SINGLE-OBJECTIVE OPTIMIZATION PROBLEMS

To evaluate optimization capability of EFO, three experiments have been conducted and EFO has been compared with a selection of well-known algorithms with a wide range of characteristics (including single-solution-based, population-based, evolutionary-based, and swarm intelligence-based). The first experiment comprises basic and complex mathematical benchmark function sets and the problems in these sets are bound-constrained. The second experiment comprises clustering and classification problem sets, in which the problems do not possess any type of constraints, including bound constraints. The third experiment, however, comprises well-known constrained real-world design problems, which have both bound and design constraints. The problem sets and competitor algorithms are introduced in detail in the following subsections.

4.1. Competitor Heuristics and Parameter Settings

4.1.1. Competitor heuristics

In addition to the benchmark set used to evaluate an optimization algorithm, the choice of competitor algorithms is also very important. Several heuristics have been proposed in the literature. To verify the effectiveness of our proposed contribution, EFO has been compared with well-known heuristics from the literature.

In the first two experiments, EFO has been compared with two single-solution-based (SA and VS) and four population-based heuristics (GA, DE, PSO, and ABC); the general characteristics of these heuristics are outlined in Table 4.1.. The common aspect of these heuristics is that they have gained so much popularity in the literature (refer to Figure 2.2., Figure 2.4., and Figure 2.6.). That's why they have been particularly chosen for performance comparison. The pseudocode of these algorithms is also provided in [31]. It is important to note

here that we have intentionally excluded FEO as a competitor heuristic due to its low performance. Speaking concretely, EFO has outperformed FEO in all of the 12 benchmark functions employed in [24].

Table 4.1. Outlines of heuristics used in comparison.

	Class		Solution Type	Motivation	Parameter
SA	Single solution-based	Swarm-based	Real-valued	Thermal equilibrium	Initial temp., cooling factor
VS				Vortex pattern	N.A.
GA	Population-based	Evolution-based	Binary-coded	Survival of the fittest	Crossover and mutation rates
DE			Real-valued		Crossover and scaling factor
PSO		Swarm-based		Cognitive and social experiences	c_1, c_2, and ω
ABC				Scout, employer, and onlooker bees	$limit$
EFO				Active and passive electrolocation	α and K

The final experiment considers only the studies that were recently proposed for solving constrained design problems: *i)* Novel approaches such as ABC [135], ISA [67], CSA [69], STA [73], BFO [71], BSA [136], and TLBO [72]; *ii)* modified versions of some heuristics such as TCA [68], PSO [75, 76], FOA [89], DE [77–79], EA [83, 84]; and *iii)* hybrid approaches such as GA-AIS [85, 86], PSO-DE [74], and CSGA [87].

4.1.2. Design of experiments

There are several factors that directly affect the performance of an optimization algorithm, such as the initial condition, problem type, and complexity. The parameter setting of an algorithm is also a crucial factor, and determining the optimal or approximate values of parameters introduces a separate problem (known as a *parameter tuning problem* in the literature) [137]. The parameter settings of each algorithm should optimally be tuned to achieve a fair comparison, which is necessary to ensure the reliability of the experiments. Because finding the optimal parameter values requires highly costly large-scale experimentation [138], the design of experiments methodology is applied to tune the parameter values,

including the population size of each algorithm in this study. The DoE methodology uses a second-order linear model in order to identify approximations to the optimal parameters.

$$y = \beta_0 + \sum_i \beta_i x_i + \sum_i \sum_{j>i} \beta_{ij} x_i x_j \sum_i \beta_{ii} x_i^2 + \epsilon \tag{14}$$

where y is a response variable representing the fitness value obtained from the parameter setting x. After the coefficient β is estimated, approximations to the optimal parameter values are obtained by applying quadratic programming.

Because it would be exhaustive to obtain the optimal values for each possible parameter setting, three-level full factorial design is employed, where each parameter is set to three levels (referred to as high, intermediate, and low). Function evaluation number (FEN) is employed as the termination criterion of a single run. Every algorithm is run twice with 500,000 and 20,000 FENs for the first and second experiments at each parameter setting, respectively. Therefore, ABC is executed $3^1 \times 2$ times, SA is executed $3^2 \times 2$ times, GA, DE, and EFO are executed $3^3 \times 2$ times, and PSO is executed $3^4 \times 2$ times on each problem. As VS is a single-solution-based and parameter-free algorithm it has been discarded from the parameter tuning. The parameters with their range as well as their optimal values obtained by using the DoE methodology for the first and second experiments are outlined in Table 4.2.. While FEN is set to 500,000 for the first experiment, it is set to 20,000 for the second experiment to avoid the algorithms to have an over-fitting problem to the train dataset employed in this experiment. The parameter values of EFO are set according to each problem in the third experiment and they are given in Section 4.4.2.. It is important to note here that while the DoE methodology tries to find the optimal values of parameters, it also explores their interactions with other parameters. However such interaction effects have not been explicitly investigated in both the proposed EFO algorithm and its competitor algorithms in this study. For a heuristic algorithm to be regarded as robust, its performance should be less sensitive to differences in the problem characteristic and tuning parameters [139]. The empirical findings from 50 basic benchmark functions reveal that EFO is not dependent on parameter settings and shows

similar performance on problems with different characteristics. These results prove the robustness of the proposed EFO algorithm. Therefore, even though the DoE methodology was applied for finding the optimal parameters in this study, it is believed that any parameter settings (other than the optimal parameters) could be employed to yield sufficiently feasible solutions for the problem of interest.

Table 4.2. Approximate optimal parameter values of EFO and competitor algorithms.

Algorithms & Parameters		**Value Range**	**Optimal Parameter Value**	
			Experiment 1	**Experiment 2**
SA				
	Initial temperature	[10, 100]	79.575	73.601
	Cooling factor	[0.01, 0.99]	0.476	0.235
GA				
	Crossover rate	[0.01, 0.99]	0.772	0.842
	Mutation rate	[0.01, 0.20]	0.059	0.011
	Population size	[20, 100]	64	44
DE				
	Crossover rate	[0.01, 0.99]	0.318	0.839
	Scaling factor	[0.01, 0.99]	0.705	0.613
	Population size	[20, 100]	61	43
PSO				
	Magnitude for cognitive best (c_1)	[0.01, 2.00]	1.594	1.291
	Magnitude for social best (c_2)	[0.01, 2.00]	1.526	1.307
	Inertia for old velocity (ω)	[0.01, 0.99]	0.502	0.923
	Population size	[20, 100]	65	67
ABC				
	Population size	[20, 100]	62	40
EFO				
	Magnitude for old amplitude (α)	[0.01, 0.99]	0.416	0.501
	Max. number of neighbors (K)	[1, $\lceil \lvert N \rvert / 2 \rceil$]	20	8
	Population size	[20, 100]	65	20

4.2. Performance Analysis on Bound Constrained Problems

4.2.1. Problem definition

In the first experiment, EFO was tested on two benchmark sets, including a well-known selection of 50 basic and 30 complex mathematical functions, known as the *CEC'17 Benchmark Suite* [11], which were obtained from the studies of Karaboga and Akay [140] and Awad et al. [11], respectively. The details of these problems are provided in Table B.1. and B.2..

The mathematical benchmark functions are divided into six classes depending on their characteristics: unimodal, multimodal, separable, nonseparable, hybrid, and composition. A unimodal function (F1–F17 in Table B.1. and F1–F3 in Table B.2.) is a function that contains only one optimum in its search space and it is used to measure how effectively an algorithm performs local search; a multimodal function (F18–F50 in Table B.1. and F4–F10 in Table B.2.) has multiple local or global optima and in order for an algorithm to achieve better performance, it must exhibit a better exploration capability. Design parameters are independent of each other in a separable function (F1–F5, F18–F26 in Table B.1.), which means the cost of a separable function is the sum of the costs independently calculated from every parameter. A nonseparable function (F6–F17, F27–F50 in Table B.1.) has design parameters that are interrelated to each other, which makes them much more complex compared to separable functions. A hybrid function (F11–F20 in Table B.2.) comprises a number of subcomponents that represent the basic function. As for a composition function (F21–F30 in Table B.2.), it merges the characteristics of subfunctions to maintain continuity around global/local optima. Heuristics are very successful in solving problems with few parameters. However, they face another problem known as the *curse of dimensionality*, as the parameter size increases and data points in the search space become very sparse [141]. The number of design parameters to be optimized in the basic mathematical benchmark function set varies from 2 to 30, whereas it is 10 for all the complex mathematical benchmark functions. For the list of constant parameters of the functions that have such constant parameters, refer to [140] and [11].

4.2.2. Results

In this experiment, each algorithm has been run 30 times and the best fitness value obtained from each run has been considered when calculating the mean, deviation, and best values. The *mean* values provide insight into the overall performance of the algorithm. Owing to their stochastic nature, heuristics can produce different results each time, and thus the *standard deviation* is a good indicator to reveal the extend to which an algorithm can produce similar results. The *best* value is obtained from the mean values of several runs, indicating the best performance obtainable. All algorithms have been run in parallel on the same machine (24 core Intel® Xeon CPU, 48 GB RAM).

Some functions, such as Easom (F7 in Table B.1.), have a valley-type global minimum which is very small compared to their entire search space. Heuristics seek the search space considering the fitness quality of the population. However, such functions do not provide fitness information, because the population is initially far away from the narrow global minimum. This case has also been observed in our experiments: the standard deviation of the fitness values of the population is very close to 0. In order to avoid that, the frequency of each candidate individual for these problems is set to a random value (i.e., $f = rand(0, 1)$ when $\sigma(fit(N)) < \varepsilon$). The ε value is set to 10^{-5} in this study.

The mean, standard deviation, and best values obtained from both the basic and complex mathematical benchmark functions are provided in detail[1]. Although these results shed light on the performance of the algorithms, a pairwise statistical test has also been employed to further investigate any significant differences between the algorithms. Pairwise tests are divided into two categories, *problem-based* and *multiproblem-based*. Problem-based testing considers the best results obtained from several runs of an algorithm against a particular problem, whereas multiproblem-based testing considers the average of the best results obtained from several runs. Problem-based pairwise comparisons are widely employed in performance comparison between given two approaches [142, 143]. The problem-based Wilcoxon Signed Rank Test at a statistical significance level of 95% is employed in this experiment.

The problem-based statistical results for the basic and complex mathematical benchmark functions are provided in Table C.1. and C.2., respectively.

In this study, the researcher's null hypothesis is that there is no statistically significant difference between the median of the results produced by EFO and its competitor algorithm (C); i.e., median (EFO) = median (C). To speculate that EFO performs much better on a given problem, the null hypothesis should be rejected or an alternative hypothesis should be accepted. T+ and T- values in Table C.1. and C.2. are the sum of the ranks for the problem in which the competitor algorithm outperforms EFO and is inferior to EFO, respectively. These values are examined to reject or to accept [143]. The (+/-/=) marks in column W are used to show statistical findings. A '+' mark indicates a case where the null hypothesis is rejected and EFO exhibited superior performance, whereas a '-' mark indicates a case where the null hypothesis is accepted and EFO exhibited inferior performance; the '=' mark indicates no statistical finding to reject or to accept the null hypothesis and EFO performs similar performance with its competitor. The last row of the table shows the total count (+/-/=) of the cases.

Table 4.3. shows problem-type-classified statistical results of the first experiment. As in most modern development tools, values below 10^{-16} are accepted as 0 during the statistical pairwise comparison in this study, because an arithmetic precision that is higher than necessary makes it difficult to compare the local search abilities of the algorithms [142]. From the problem-type statistical results for the basic benchmark functions in Table 4.3., it is seen that EFO outperforms all its competitors without exception, which is also the case for the complex benchmark functions, with the only exception that DE outperforms EFO on the eight problems overall. It is also seen that EFO obtains considerably better results on *multimodal*, *hybrid* (except DE), and *composition* (except ABC) functions, and it shows similar or better performance on *unimodal* functions. The experiment results support that EFO does not face a *curse of dimensionality* problem due to its better performance on problems with high dimensions (15 out of 50 basic mathematical problems) such as *Schwefel*, *Rastrigin*, and *Griewank*. Moreover, EFO has shown a better or comparative performance on hard problems such as

Perm and *Power Sum* problems, where the global optimal solutions lie at the boundaries of the search space.

Table 4.3. Problem-type-based statistical comparison of EFO on mathematical benchmark functions.

Basic benchmark functions

Problem Type	EFO / SA	EFO / VS	EFO / GA	EFO / DE	EFO / PSO	EFO / ABC
US	2/2/1	2/2/1	4/1/0	1/4/0	0/4/1	3/2/0
UN	9/2/1	4/6/2	8/4/0	4/7/1	2/7/3	6/4/2
MS	4/1/3	4/4/0	4/4/0	2/6/0	4/4/0	0/8/0
MN	21/2/0	3/15/5	12/9/2	2/17/4	11/9/3	8/14/1
Total (+/=/-)	38/7/5	13/29/8	29/19/2	9/36/5	17/26/7	17/30/3

Complex benchmark functions

Problem Type	EFO / SA	EFO / VS	EFO / GA	EFO / DE	EFO / PSO	EFO / ABC
U	1/1/1	1/1/1	2/1/0	0/1/2	1/1/1	2/1/0
M	6/1/0	5/2/0	7/0/0	2/4/1	6/1/0	3/4/0
H	9/1/0	8/1/1	7/1/2	0/1/9	6/3/1	3/5/2
C	9/1/0	4/4/2	8/2/0	6/0/4	9/1/0	0/7/3
Total (+/=/-)	25/4/1	18/8/4	24/4/2	8/6/16	22/6/2	8/17/5

In addition to the Wilcoxon's sign test, the VIKOR (Vise Kri-terijumska Optimizacija I Kompromisno Resenje) [144] and TOPSIS (Technique for Order Preference by Similarity to an Ideal Solution) [145] ranking methods are applied for the performance evaluation of EFO and its competitor heuristics on basic and complex mathematical benchmark problems. VIKOR and TOPSIS are MCDM methods considering the ranking of alternatives. They make use of a decision matrix, weight vector, and real number (v) to calculate rankings. In this study, a $7\times TPS$ decision matrix is given, in which rows represent the heuristic methods (alternatives) and columns represent the mean performance obtained from a heuristic on the problem (criteria). In addition, a $1\times TPS$ weight vector is given. Because each problem is given equal weight, each value in this vector is equal to $1/TPS$. Finally, v is set to 0.5 in the ranking calculation. Here, TPS represents total problem size, which is equal to 50 and 30 for the

Table 4.4. Ranking scores evaluated using VIKOR and TOPSIS methods for basic and complex mathematical benchmark functions

Methods	Basic benchmark functions				Complex benchmark functions			
	VIKOR		TOPSIS		VIKOR		TOPSIS	
	Q_j	Rank	R_j	Rank	Q_j	Rank	R_j	Rank
SA	0.5000	6	0.4946	7	0.9355	7	0.4187	7
VS	0.0781	4	0.8455	3	0.3760	2	0.8163	3
GA	0.3468	5	0.5364	6	0.5399	5	0.6604	6
DE	0.0351	3	0.8327	4	0.4694	4	0.9749	1
PSO	0.6598	7	0.7680	5	0.6308	6	0.6920	5
ABC	0.0154	2	0.9338	2	0.4521	3	0.7328	4
EFO	0.0000	1	0.9879	1	0.0000	1	0.9113	2

basic and complex problem sets, respectively. Table 4.4. shows the rankings of EFO and its competitor heuristics evaluated separately on the basic and complex benchmark functions. The VIKOR and TOPSIS ranking values that are evaluated on the basic benchmark set and the VIKOR ranking values that are evaluated on the complex benchmark sets support the superior performance of EFO over its competitors as well. However, according to the TOPSIS ranking values, EFO showed a worse performance than DE but much better than the other competitors on the complex benchmark function set.

The convergence rate indicates how long it takes for an algorithm to find a solution. The problem known as *premature convergence* occurs when an algorithm converges to a local optimum and is no longer able to explore other parts of the search space [146]. To reveal the comparative convergence capability of EFO, the change in fitness value obtained from basic and complex mathematical benchmark functions with respect to the iteration is comparatively shown in Figure C.1. through C.3.. Because SA is not an iteration-based heuristic, it is excluded from all these figures. The convergence behaviors demonstrate that EFO showed better or at least highly competitive performance on the problem set with respect to the convergence speed and also that VS, which showed a much more compelling performance than most of the other competitors, exhibited much slower convergence behavior on almost all of the problems.

4.3. Performance Analysis on Machine Learning Tasks

As a subfield of AI, machine learning provides an ability that automatically learns and improves itself from the past experience without a human intervention. The abundance of data in the recent decades has made too hard to extract the useful knowledge from such big data, or to recommend the most relevant content by eliminating trivial, noisy information. That's why machine learning algorithms, which can easily overcome the information overload, has gained so much popularity today. Machine learning algorithms have been categorized based on what they target to achieve with the data they process, which is referred as *task* in the literature. *Clustering*, *classification*, and *regression* are examples to popular machine learning task. In this part of the thesis, EFO algorithm has been evaluated as a machine learning algorithm on two most popular machine learning tasks: *clustering* and *classification*. The definitions of these tasks have been explained in the following sections.

4.3.1. Clustering

Clustering is a process that aims to determine the optimal cluster centers to gather objects with great similarities into the same cluster or, in other words, to separate different objects from the same cluster as much as possible. Here, the distance measurement generally determines the similarities among objects. The formal definition of the clustering problem, also known as unsupervised learning in the literature, is as follows: given *O* objects, assign every object into one of *C* clusters such that it minimizes the Euclidean distance between every object and the center of the cluster to which it belongs:

$$J(w,z) = \sum_{i=1}^{O}\sum_{j=1}^{C} w_{ij}||x_i - z_j||^2 \tag{15}$$

where x_i and z_j are the locations of the object i and cluster j, respectively. w_{ij} is a weight parameter that takes either 1 (if i belongs to the jth cluster) or 0 (otherwise).

The goal of an optimization algorithm is to find the optimal *C* cluster centers from an infinite number of candidate cluster centers by minimizing the fitness function. Every individual in EFO represents a candidate solution to the clustering problem, which is actually a vector containing *C* clusters. The cost function for determining the quality of the ith individual in EFO is adopted as follows:

$$f_i = \frac{1}{D_{Train}} \sum_{j=1}^{D_{Train}} d(x_j, p_i^{CL_{known}(x_j)}) \tag{16}$$

where D_{Train} is the number of instances used for training purpose and $p_i^{CL_{known}(x_j)}$ is the label of the class to which instance x_j belongs to.

The performance of EFO as a clustering technique is investigated in this experiment. In order to achieve that, 13 datasets from a well-known database repository in literature, the UCI repository [147], are used. Each dataset contains actual data collected from different domains. These datasets and their features are listed in alphabetical order in Table 4.5.. In this table, the total number of objects (*O*) and clusters (*C*); number of features (*F*); percentages of continuous, integer, Boolean values (*%num*), and nominal values (*%symb*); flag (*miss*) indicating whether a dataset contains missing attribute values; and percentage of classes (*%major*) to which the majority of instances belongs are listed.

Table 4.5. Properties of datasets used for clustering.

	O	*C*	*F*	*%num*	*%symb*	*miss*	*%major*
Balance	625	3	4	100	0	No	46
Cancer	569	2	30	100	0	No	63
Cancer-Int	699	2	9	100	0	Yes	66
Credit	690	2	51	40	60	Yes	56
Dermatology	366	6	34	97	3	Yes	31
Diabetes	768	2	8	100	0	No	65
E-Coli	327	5	6	88	12	No	43
Glass	214	6	9	100	0	No	36
Heart	303	2	35	100	0	Yes	54
Horse	364	3	58	32	68	Yes	61
Iris	150	3	4	100	0	No	33
Thyroid	215	3	5	100	0	No	70
Wine	178	3	13	100	0	No	40

4.3.1.1. Clustering results

Contrary to the mathematical (see Section 4.2.) benchmark problems, where the formulation of the problem directly measures the performance of an algorithm, this experiment comprises two phases: *training* and *testing*. We have used the first 75% of every dataset for the training purpose and the last 25% for the testing purpose.

The first phase of this experiment, training, is crucially important for evaluating the learning capability of the algorithm. It should be noted that, as in [148] and [149], the purpose of the clustering in this study is to learn the optimal clustering centers with a priori knowledge of the number of clusters. However, there are also studies [150, 151] in the literature that aim to optimize also the number of clusters. Optimal cluster centers are learned through the training phase according to Eq. 16.

Testing reveals how well an algorithm classifies instances taken as a subset of the whole dataset. In the testing phase, every instance in the test dataset is assigned to a class depending on the Euclidean distances to the cluster centers learned in the training phase. For a performance assessment of EFO; CEP, the percentage of misplaced instances in the whole test dataset, is used as the performance metric:

$$CEP = \frac{\#\ of\ misplaced\ instances}{total\ number\ of\ test\ instances} \tag{17}$$

The comparative CEP and ranking values obtained from the testing phase are shown in Table 4.6.. The results indicate that EFO performs best on 11 out of 13 problems. The comparative findings of the first two experiments strongly support the NFL [2] theorem. GA that performs worse in the first experiment shows very compelling performance on the clustering problems, whereas VS and PSO that are very competitive in the first experiment are inferior in this problem. To reveal the overall performance of the algorithms, the average CEP values calculated from all the problems and general ranking based on the average values are provided in Table 4.7.. Upon close inspection of the average CEP values and the rankings, it can be seen that EFO shows the best performance overall.

Table 4.6. Classification error percentages and rankings (within parenthesis) of the algorithms on clustering problems.

Problem	SA	VS	GA	DE	PSO	ABC	EFO
Balance	25.64 (2)	57.05 (3)	25 (1)	25 (1)	25.64 (2)	25 (1)	25 (1)
Cancer	6.34 (3)	31.69 (5)	4.93 (1)	5.63 (2)	7.04 (4)	4.93 (1)	4.93 (1)
Cancer-int	2.86 (1)	6.86 (2)	2.86 (1)	2.86 (1)	2.86 (1)	2.86 (1)	2.86 (1)
Credit	21.51 (1)	49.42 (5)	25 (2)	26.16 (3)	26.16 (3)	40.7 (4)	26.16 (3)
Dermatology	6.52 (2)	65.22 (7)	23.91 (6)	7.61 (3)	8.7 (4)	18.48 (5)	5.43 (1)
Diabetes	22.92 (2)	63.02 (3)	22.92 (2)	22.4 (1)	22.92 (2)	22.4 (1)	22.4 (1)
E-coli	14.63 (1)	93.9 (2)	14.63 (1)	14.63 (1)	15.85 (1)	14.63 (1)	14.63 (1)
Glass	39.62 (2)	88.68 (5)	41.51 (3)	37.74 (1)	52.83 (4)	52.83 (4)	37.74 (1)
Heart	17.11 (1)	38.16 (3)	18.42 (2)	17.11 (1)	18.42 (2)	17.11 (1)	17.11 (1)
Horse	43.96 (5)	46.15 (6)	40.66 (3)	42.86 (4)	32.97 (1)	36.26 (2)	42.86 (4)
Iris	2.63 (1)	84.21 (2)	2.63 (1)	2.63 (1)	2.63 (1)	2.63 (1)	2.63 (1)
Thyroid	5.66 (1)	26.42 (2)	5.66 (1)	5.66 (1)	5.66 (1)	5.66 (1)	5.66 (1)
Wine	4.44 (1)	51.11 (3)	4.44 (1)	4.44 (1)	6.67 (2)	4.44 (1)	4.44 (1)
# of 1st ranking	7	0	7	9	5	9	11

Table 4.7. Average classification error percentages and ranking of the algorithms on clustering problems.

	SA	VS	GA	DE	PSO	ABC	EFO
Average CEP	16.45	53.99	17.89	16.52	17.57	19.07	16.30
Rank	2	7	5	3	4	6	1

4.3.2. Classification

As in clustering, classification also categorizes the objects into one or more classes depending on the problem itself. Classification primarily differs from clustering in that predefined label vector assigned to each instance of the data is available in classification. That's why classification is a supervised learning approach, whereas clustering is an unsupervised learning approach. Depending on the problem characteristic, classification can be viewed as *binary*

classification and *multi-class (or multi-nominal) classification*. It is called as binary classification when the objective is to classify instance one of two classes; similarly it is called as multi-class classification when the objective is to classify instance one of more than two classes.

In this experiment, EFO algorithm has been run as a supervised learning algorithm for the classification task on both binary and multi-class classification problems. The proposed standard EFO algorithm, as seen from Section 3., is a real-parameter optimization algorithm targeting constrained and unconstrained mathematical benchmark problems. That's why it cannot be applied on classification problems directly. ANN has been utilized in this experiment in order to adapt EFO for such problem types.

ANNs are special computing computing systems employed in machine learning to perform various machine learning tasks. Simulating the biological nervous system, ANNs are regarded as a simplified model of a brain. An ANN has a number of layers in which tens, or even hundreds of computing units, called as *neurons*, take part. Each neuron at a layer is interconnected to neurons at other layers. ANN layers are categorized as *input*, *hidden*, and *output*. Input layer is the first layer that brings the data to the system to be processed through intermediate layers (hidden layer). The output layer is the final layer that produces the outcome of the system. The connections between the layers have weights and activation functions which determine the magnitude of a signal/message outgoing from the source neuron. An example of 3 layer-, or 1 hidden layer-, ANN (with 4 neurons at the hidden layer) is given in Figure 4.1. ANN calculates an output value with respect to an input of a given system and magnitude values of each neurons, then evaluates an error (E) value.

The main purpose of ANN is to find optimum magnitude values of each neuron so that it produces a higher accuracy estimation. In other words, ANN optimizes the magnitude values between neurons to obtain desired output of system of interest. Therefore, it relies on different learning algorithms to produce a knowledge from the data provided. The output of ith neuron at kth layer is given below:

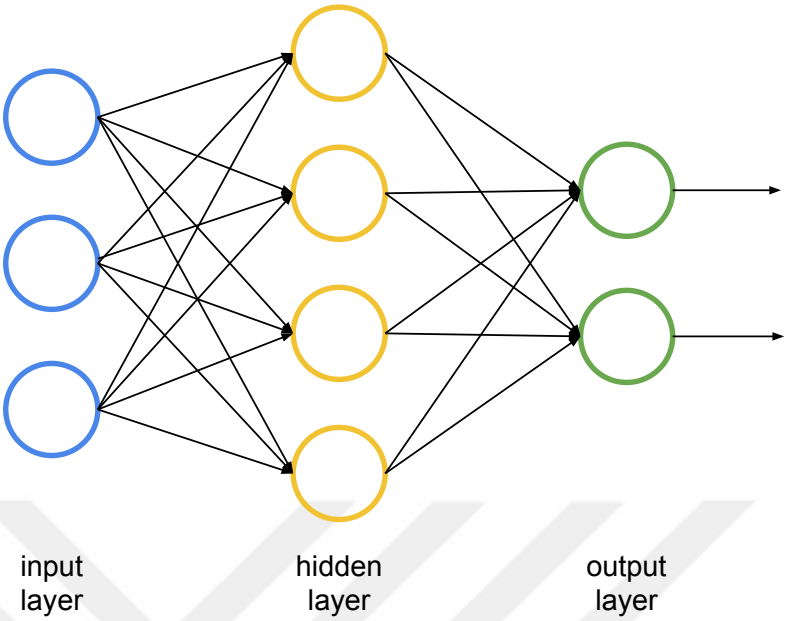

Figure 4.1. An illustration of a feed-forward neural network with three layers.

$$o_i^k = f_i^k \left(\sum_{j=1}^{n} o_j^{k-1} w_{ij}^{k-1} + b_i^k \right) \tag{18}$$

where o_i^k is output value, f_i^k is activation function and b_i^k is bias weight of the *i*th unit in the *k*th layer. w_{ij}^{k-1} is connection weight between neurons *j* and *i* at layers *k-1* and *k*, respectively. *n* is neuron size (see Figure 4.2.).

MSE is often used as error measurement in ANN, therefore the error (*E*) value calculated by ANN is given as below:

$$E = \frac{1}{P} \sum_{p=1}^{P} \sum_{k=1}^{S} \left(d_k^p - o_k^p \right)^2 \tag{19}$$

where *P* and *S* represent total numbers of instances and the neurons at output layer, respectively. d_k^p and o_k^p are desired and actual output values of pair *p* respectively.

The success of estimation driven by ANN is strongly dependant both on the data (that is provided at the input layer) and on the weight vectors that is optimized by the learning algorithm. EFO has been employed here in order to find the approximation of the global

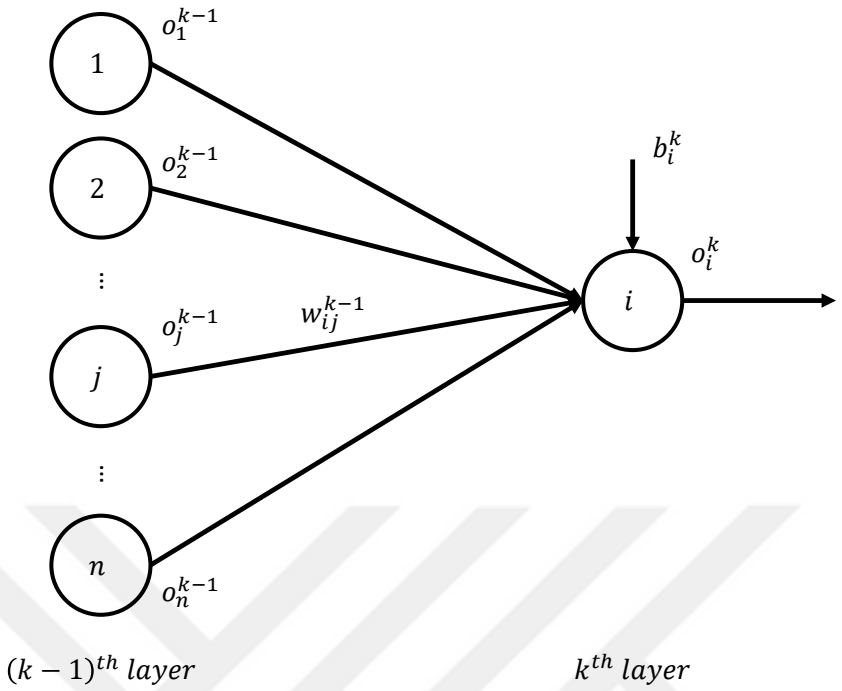

Figure 4.2. A demonstration of output calculation for neuron i at layer k.

optimum weight vector of the problem of interest. In short, EFO algorithm has been adapted to solve classification problems by optimizing the weight vector of ANNs. The framework employed in this experiment is shown in Figure 4.3.

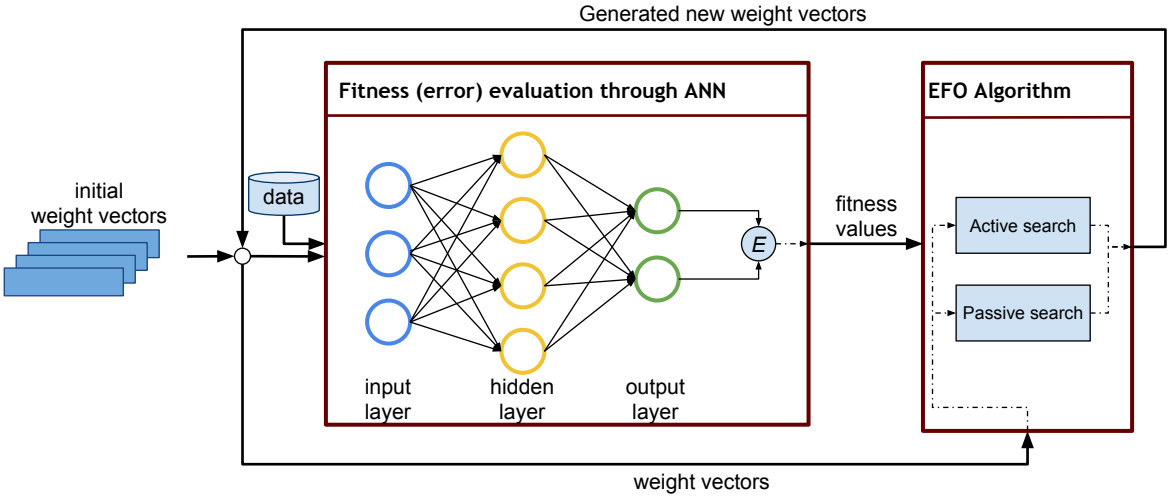

Figure 4.3. A framework of EFO classifier algorithm.

The proposed classifier framework works as follows:

1. **Generation of initial weight vectors:** It corresponds to the initialization of population phase in EFO. So, vector size is equal to the population size while vector length is equal to the problem size. Not only the problem itself determines the problem size here, but also the number of hidden layers, the number of neurons at every layer, and the number of biases connected to neurons. Here, in the experiments, 1 hidden layer-ANN is constructed. Hence problem dimension is $(PS \times HS + BH) + (HS \times OS + BO)$. Here PS represents the problem (attribute) size; HS and OS are, respectively, the number of neurons at the hidden and output layers; BH and BO are the number of bias units connected to neurons at the hidden and output layers, respectively. Every unit value at each weight vector is initially takes a random value between -1 and +1, which corresponds to lower and upper bounds of the experiments, respectively.

2. **Error measurement through ANN:** It corresponds to the evaluation of weight vector. So E is calculated separately on every weight vector. Apart from the weight vectors, the data with its label vector is provided to ANN. *Sigmoid* function $(= 1/(1 + e^{-x}))$ has been used as an activation function between layers. Therefore, the output value of each neuron is restricted to be between 0 and 1. While the neuron size at the input layer is equal to the number of attributes, the neuron size at the output layer is equal to number of classes to be predicted. Here, the output layer is encoded according to the number of classes for the multi-class problems (note that binary-class problems can be represented by only one neurons producing either 0 or 1). Therefore, for the multi-class problems, the number of neurons at the output layer is set to the number of classes of the problem of interest. An example output layer for the problem Iris, where *Setosa*, *Versicolor*, and *Virginia* are the classes of the problem, is illustrated in Figure 4.4..

3. **Generation of new weight vectors:** It is the phase where the standard EFO algorithm (Section 3.) is applied. In this phase, new candidate vectors are generated through active and passive searches in order to find the least error compared to the previous iteration. The generated solutions (weight vectors) are again evaluated through ANN (Step 2) and the evolution continues until the termination criteria is satisfied.

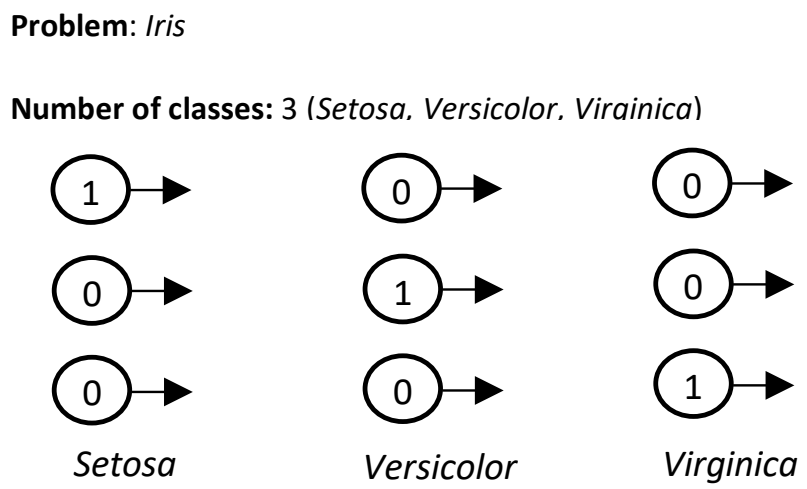

Figure 4.4. An output layer representation in ANN for *Iris* problem with 3 classes.

Table 4.8. Parameter List Used for Classification

Parameter	**Problems & Values**	
	Benchmark problems	**Botnet detection**
Number of neurons in hidden layer	25	5
Run time	10	10
Maximum cycle	1000	2000
Population size	40	40

To show the performance of EFO as supervised learning approach, two different test sets are employed. The first set comprises of 13 benchmark problems that are also used in the clustering experiment (see Section 4.3.). The main objective here is to investigate how better EFO algorithm can categorise the objects into the classes than the competitor methods can. The second test set, however, comprises of a real-world problem known as *botnet detection*. The parameter settings for the two experiments have been given in Table 4.8.. The parameter values of the algorithms given in Table 4.2. have also been used in this experiment. The experimental results as well as the detailed problem definition of botnet detection have been given in the following sections.

4.3.2.1. Classification results

The comparative algorithms employed in clustering problem have also been used in this experiment. Each of the competitor methods, including EFO, has been run as a supervised

learning method to optimize weight vector of ANN in the framework (see Figure 4.3.). To ensure a fair comparison they have been run under the same conditions. As in the clustering experiment, the first 75% of the individual dataset has been used as training whereas the remaining part (25%) has been used for testing in the experiments as well.

MSE (see Equation 19) has been used in error measurement for training, while CEP (see Equation 17) has been used for performance evaluation in testing. The convergence of every algorithm (except SA as it is not an iteration-based algorithm) has been given in Figure 4.5.. The problem-based CEP performances of EFO and the competitor methods as well as their rankings are given in Table 4.9. and the overall CEP values are given in Table 4.10.

Table 4.9. Classification error percentages and rankings (within parenthesis) of the classification algorithms.

Problem	**SA**	**VS**	**GA**	**DE**	**PSO**	**ABC**	**EFO**
Balance	15.51 (3)	15.45 (2)	18.40 (7)	**15.13** (1)	17.05 (5)	17.50 (6)	15.96 (4)
Cancer	8.80 (6)	**2.39** (1)	3.73 (4)	3.10 (3)	4.86 (5)	2.89 (2)	**2.39** (1)
Cancer-int	8.51 (7)	2.51 (4)	2.34 (2)	2.91 (6)	2.69 (5)	2.40 (3)	**1.89** (1)
Credit	20.12 (5)	18.55 (3)	20.23 (6)	17.91 (2)	20.12 (5)	18.60 (4)	**16.51** (1)
Dermatology	15.98 (4)	21.41 (5)	29.13 (6)	15.54 (3)	35.54 (7)	12.28 (2)	**10.00** (1)
Diabetes	27.29 (7)	22.97 (5)	22.34 (3)	**22.03** (1)	23.33 (6)	22.19 (2)	22.40 (4)
E-coli	31.46 (7)	21.71 (4)	29.76 (6)	19.39 (3)	27.56 (5)	13.29 (2)	**10.49** (1)
Glass	56.79 (7)	**34.34** (1)	55.85 (6)	35.66 (2)	50.19 (5)	42.45 (4)	35.85 (3)
Heart	19.87 (6)	17.24 (4)	16.84 (3)	16.05 (2)	17.76 (5)	16.05 (2)	**15.53** (1)
Horse	38.90 (5)	**35.16** (1)	36.26 (3)	38.35 (6)	37.47 (4)	35.38 (2)	39.67 (7)
Iris	18.95 (7)	6.84 (4)	13.68 (6)	5.26 (3)	9.74 (5)	4.47 (2)	**3.42** (1)
Thyroid	16.04 (6)	4.72 (3)	22.45 (7)	**3.40** (1)	12.08 (5)	5.09 (4)	4.53 (2)
Wine	10.00 (5)	3.33 (2)	5.78 (4)	4.89 (3)	14.89 (6)	3.33 (2)	**2.22** (1)
# of 1st ranking	0	3	0	3	0	0	8

Table 4.10. Average classification error percentages and ranking of the classification algorithms.

	SA	**VS**	**GA**	**DE**	**PSO**	**ABC**	**EFO**
Average CEP	22.17	15.89	21.29	15.36	21.02	15.07	**13.91**

Table 4.10. Average classification error percentages and ranking of the classification algorithms.

	SA	**VS**	**GA**	**DE**	**PSO**	**ABC**	**EFO**
Rank	7	4	6	3	5	2	1

The CEP results reveal that EFO has shown the best performance by far than the competitors. It has become the best classifier on 8 problems out of 13 (these problems have been given in bold in the table). EFO only shows the worst performance for the *horse* problem. ABC is the second superior algorithm considering its overall CEP performance; however it has been unable to find the least CEP value on any problem. DE and VS have shown equal performance and, considering their outstanding convergence performance during training, it can be concluded that they face over-fitting problem. It is worth mentioning that while DE has shown good performance in clustering; it is not the case in in classification. Similarly, SA and GA, which perform reasonably well performance in clustering have shown the worst performances in classification. These results supports the NFL theorem.

4.3.2.2. Classification results for botnet detection

Botnet is a network of compromised systems called as *bots or zombies* which are controlled through a special Command and Control (C&C) channel by an intruder node known as *botmaster*. By taking advantage of computational resources of bots, botnets are used for performing several distributed illegal activities such as spamming, distributed denial-of-service attacks. The increasing size of botnets have now reached to an unprecedented level such that more than 80% of the Internet traffic is propagating through botnets [152]. Distributing such bots are cheap, however they could have drastic effects on the economy. It is reported that malware distribution has caused a damage of 13.2 billion to 67.2 billion USD to the global market within only two years in the past [152]. Hence, developing robust and effective detection systems towards different forms of botnets has become a must.

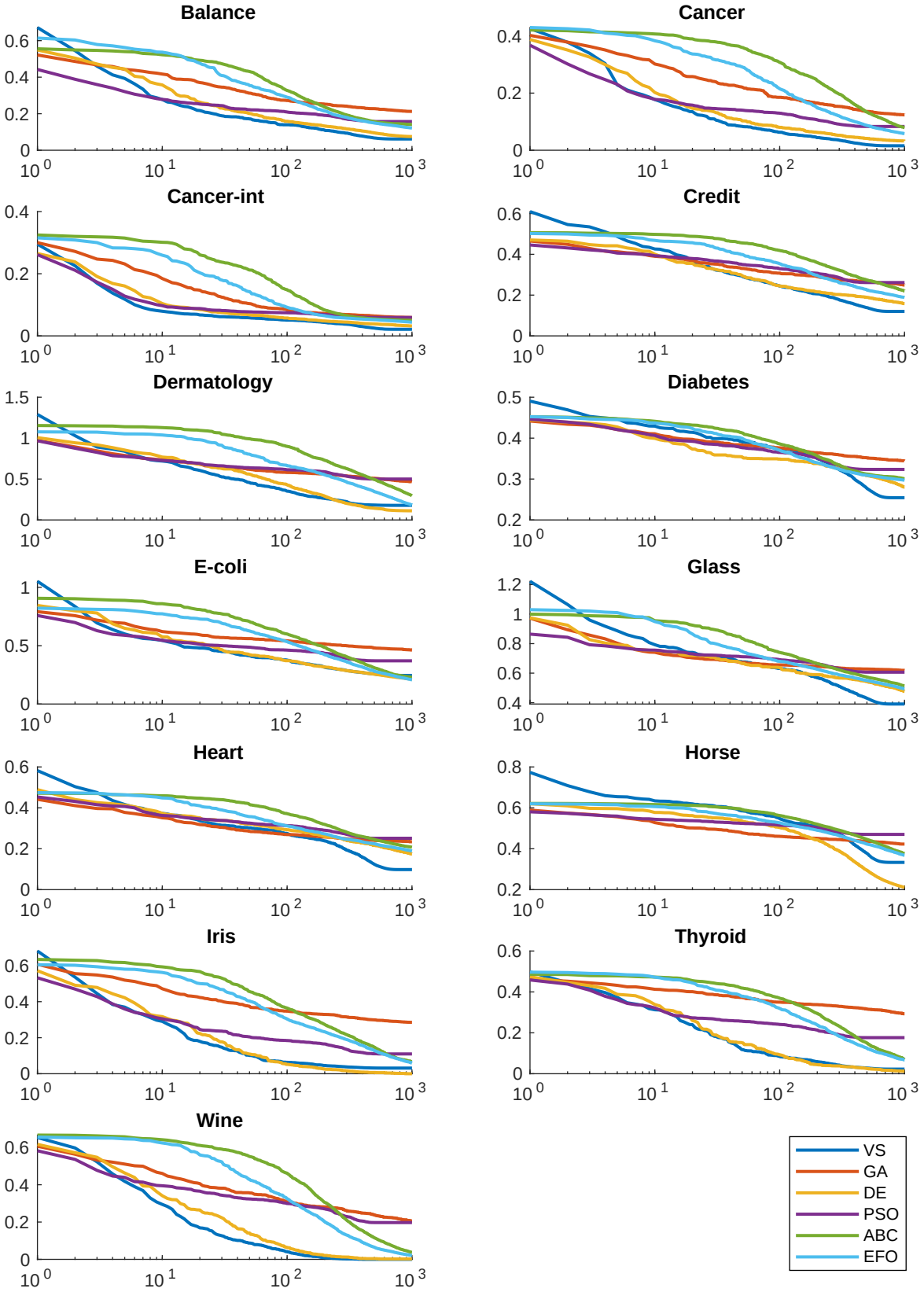

Figure 4.5. Convergence of the classification algorithms

Experiment and Findings

In this experiment, the classification performance of EFO on botnet detection is explored. As explained earlier, every individual in EFO represents the weight vector of ANN that it has trained. The convergence of that weight vector to the optimum weights ensures an emergence of a fine botnet detection that truly discriminates bot flows the benign flows. Instead of the MSE, Equation 20 has been used as error/loss function in this experiment.

$$E = \frac{\big(1 - (TP - FP)\big)}{2} \tag{20}$$

Here TP and FP are the true and false positive rates, respectively. The detailed definitions of these metrics are given in the following sections. Loss error can range from 0 (for the base case when TP = 1, FP = 0) to 1 (for the worst case when TP = 0, FP = 1). The reason behind using this equation instead of the MSE is the imbalanced botnet dataset which contains 2,390,624 benign flows and only 79,428 botnet flows). If MSE was used as the loss function, the detection system would yield a low error rate (~0.0321), even when it detected every flow including malicious flows as benign, which would let the learning algorithm to over-stimulate only the benign flows neglecting the bot flows.

In order to avoid the over-fitting problem and also to reduce the computational complexity, neurons at the hidden layer has been pruned in the experiments. It is empirically observed that a network having five neurons at the the hidden layer is good enough for a high performance. The following sections explain *i*) the existing botnet detection approaches in the literature, *ii*) the dataset, *iii*) the flow generation , *iv*) test metrics, and *v*) experimental findings.

Competitor approaches on botnet detection:

Machine learning-based techniques have already been proposed to detect different types of botnets (i.e., IRC, HTTP, and P2P) within recent years. However, most of these approaches employ deep packet inspection, which cannot analyze encrypted traffic. Given that the majority of today's network traffic is encrypted, researchers have moved their focus on developing flow-based detection systems as complementary to such systems recently. In the experiments,

some of these flow-based detection approaches are used for comparison. These approaches are explained in the following paragraphs.

Huseynov et al. [153] proposed an approach built on similarity of communication patterns between botmasters and bots. It relies on a semi supervised method where only 10% of all traffic was labeled. Firstly, similar activities are clustered by using ant colony optimization and then, clusters are identified based on their similarities with the labeled data. Narang et al. [154] applied different feature selection methods for identification of botnet activities. The authors concluded that the build time of the algorithm increases considerably when the full feature set is employed.

Kirubavathi et al. [155] proposed a method for detecting HTTP-based botnets. Therefore, the inputs to the proposed neural network consist of only TCP related features. They extended their study in [156] to study other types of communication schemes (IRC & P2P) and to investigate other classification methods (i.e., J48, NB, SVM). In another study based on neural networks, Nogueira et al. [157] proposed a framework called botnet security system (BoNeSSy) where the historical profiling provided by each application were used to train ANN through BP algorithm [158].

A behavioral-based P2P botnet identification system was proposed by Saad et al [159]. They regarded network traffic as data stream where bots tend to behave differently over time. The authors also studied the ability of five different learning algorithms for online botnet detection. Although the performance of these algorithms was promising (the best detection accuracy was achieved by SVM), they were not enough to satisfy all the requirements of an online detection system. Another behavioral-based botnet detection method proposed by Wang et al. [160] identifies malicious domain names and IP addresses by using a FPR technique.

Livadas et al. [161] proposed an IRC-based botnet detection system of two stages. The first stage filters out non-chat flows. In the second stage, real chat flows are distinguished from botnet flows using J48, NB, and BNC. Fedynyshyn et al. [162] proposed a host-based approach to detect botnet traffic by also employing RFC and J48 classifier.

Botnet datasets:

Two well-known botnet datasets are employed in the experiments. The first dataset called ISOT dataset [163] is the combination of publicly available different datasets comprising of bot traffics from the French Chapter of Honeynet Project [164] and benign traffics from both the Traffic Lab at Ericsson Research in Hungary [165] and the Lawrence Berkeley National Lab (LBNL) [166]. Bot traffics are belonging to the P2P Storm and Waledac botnets captured from 2007 to 2009. Benign traffics, on the other hand, were recorded from everyday traffics such as HTTP web browsing, gaming packets, torrent packets like Azureus. Even though the ISOT dataset is big enough to train a classifier, an additional dataset called the CICIDS2017 dataset [167] which contains more up-to-date bot activities is used in this study. It was generated from 5-days of traffic captures between July 3 and July 7 in 2017 and different attacks were implemented and captured for each day. In the experiments, only the traffic captured on July 7, 2017 is used, since it contains both malicious and benign data. Refer to [163] and [167] for the detailed description of the ISOT and CICIDS2017 datasets, respectively.

Flow generation:

CICFlowMeter (or ISCXFlowMeter) [168], a Java-based bidirectional flow generator and analyzer tool integrating jNetPcap [169] library, has been used to generate traffic flow comprising of IPv4 packets and to extract all traffic-related features. It supports 78 traffic-related statistical features calculated separately for the forward and backward directions as well as other attributes identifying the flow such as flow ID, source and destination IPs and ports, and protocols. The identifying attributes of traffic flows such as IP addresses have been removed from the feature set as they might not be good indicators especially when the training and testing data come from different network domains. The tool terminates a flow depending on the flow timeout and the FIN flag. In this study, the flow timeout value has been set 30 mins. It labels a flow *bot* or *benign* considering the IPv4 addresses of the sender or the receiver. The feature set generated through the CICFlowMeter has been provided in Table D.1. and refer to [168] for the detailed description of these features. A normalization process is carried out and the values of features are mapped to a range from 0 to 1 using the *Weka* machine learning

tool [170]. In the final step, the first 70% of the individual flows has been used for training and the remaining 30% of the individuals flows has been used in testing in this experiment.

Performance metrics:

In order to evaluate the performance of EFO on botnet detection, the following performance evaluations metrics are employed: *accuracy*, *precision*, *recall*, and *F1 Score*. The calculation of *accuracy* is given in Eq. 21.

$$accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{21}$$

where TP, TN, FP, and FN are the well-known metrics indicating *true positive*, *true negative*, *false positive*, and *false negative* rates, respectively. For the problem addressed here, TP represents the number of correctly detected bot traffics. FP represents the amount of benign traffics misclassified as bot traffics. TN (the amount of correctly classified benign traffics) and FN (the amount of bot traffics misclassified as benign traffic) are the other metrics complementing the FP and TP, respectively.

Apart from the accuracy; *precision*, *recall* and *F1 score* are other performance evaluation metrics that have been used to interpret the performance of the evolved best model in this study. *Precision* (Eq. 22) represents the ratio of the correctly labelled bot flows to all flows labelled as bots by the model.

$$precision = \frac{TP}{(TP + FP)} \tag{22}$$

Recall (Eq. 23), or sensitivity, gives how good the model detects the flows belonging to botnet traffic.

$$recall = \frac{TP}{(TP + FN)} \tag{23}$$

F1 score (Eq. 24), or F-Measure, is the weighted average of precision and recall and it takes both false positive and false negative rates into account. F1 score may be a more useful than other metrics for uneven class distribution.

$$F1\ Score = \frac{2 \times (recall \times precision)}{(recall + precision)} \tag{24}$$

Results:

The proposed approach has been run 10 times under the same conditions on both datasets, namely ISOT & CICIDS2017 datasets. The obtained test results reveal that EFO-based botnet detection system is able to detect bot flows with the accuracy ranging from 83.03 to 90.99. The *accuracy*, *recall*, *precision*, *F1 score* values calculated of the best model is provided in Table 4.11.. The corresponding convergence plot captured on the training is given in Figure 4.6..

Table 4.11. Performance of EFO on botnet detection.

Accuracy	**Recall**	**Precision**	**F1 Score**
9.098781e-01	9.379968e-01	8.880549e-01	9.123429e-01

The best performance has also been compared with some of the existing botnet detection approaches . It is worth mentioning that, it is hardly to conduct a fair comparison because of the different network environments and hence datasets employed by different approaches [171].

The findings reveal that the proposed ANN-EFO has outperformed the approaches proposed by Huseynov et al. [153], Nogueira et al. [157], and Saad et al. [159]; while it has shown competitive to the approaches proposed by Livadas et al. [161], Narang et al. [154], Fedynyshyn et al. [162], Yilmaz and Sen [173]; but become inferior to the other approaches proposed by Wang et al. [160], Kirubavathi and Anitha [155], and Kirubavathi and Anitha [156].

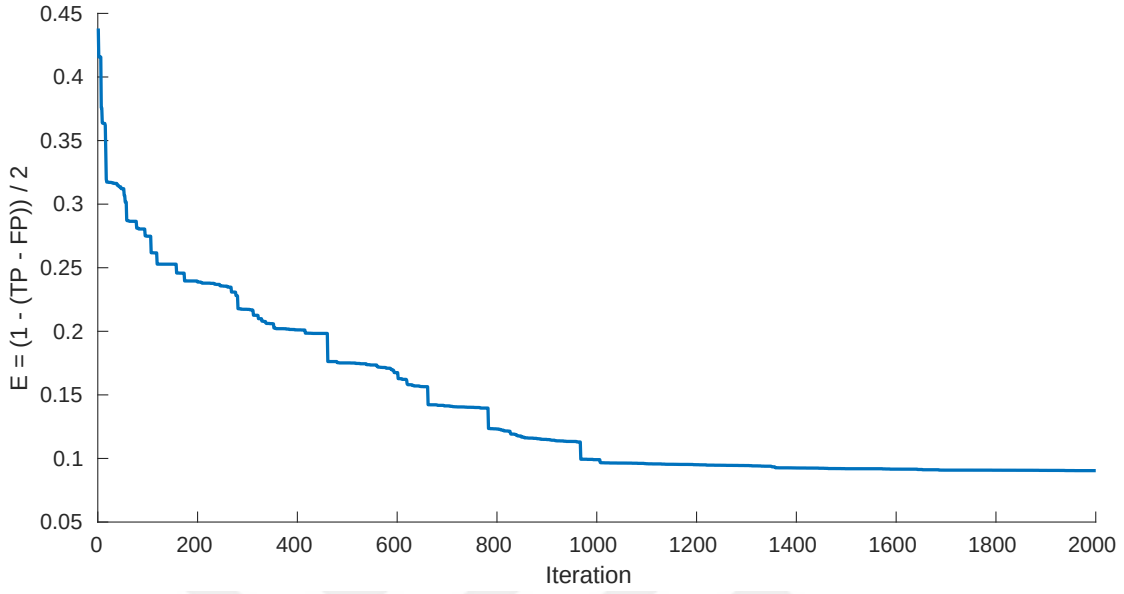

Figure 4.6. Convergence of EFO on botnet detection.

Table 4.12. Comparison of EFO with other approaches on botnet detection

Study	Method	Dataset	C&C structures	Accuracy (%)
Huseynov et al. [153]	ACO	ISOT	P2P	72.15
Nogueira et al. [157]	ANN-BP	Generated traces	IRC, P2P, HTTP	87.56
Saad et al. [159]	SVM	ISOT	P2P	89.00
Huseynov et al. [153]	ACO	ISOT	P2P	89.85
Livadas et al. [161]	BNC	*Dartmouth* trace [172]	IRC	92.00
Narang et al. [154]	BNC	ISOT & Generated trace	P2P	92.55
Fedynyshyn et al. [162]	RFC	Generated traces	IRC, P2P, HTTP	92.90
Yilmaz and Sen [173]	GE	ISOT & CICIDS2017	P2P	92.92
Wang et al. [160]	FPR	Generated traces	IRC, P2P, HTTP	95.00
Yilmaz and Sen [173]	GE	CICIDS2017	P2P	95.09
Yilmaz and Sen [173]	GE	ISOT	P2P	96.24
Kirubavathi and Anitha [155]	ANN-BP	Generated traces	HTTP	99.02
Kirubavathi and Anitha [156]	NB	ISOT & Others*	IRC, P2P, HTTP	99.14
Present study	ANN-EFO	ISOT & CICIDS2017	P2P	**90.99**

* In this dataset, 11 datasets including various types of bots like Skynet, Rbot and etc. are combined.

4.4. Performance Analysis on Constrained Design Problems

4.4.1. Problem definition

The main purpose of this experiment is to validate the performance of EFO on real-world problems, as most of these problems involve equality and/or inequality constraints. Different approaches have been proposed in the literature to handle design constraints. Because

of its simplicity and ease of implementation, the common approach to handle constrained problems is the *penalty method*. The penalty method maps the constrained search space to the unconstrained search space through a penalty function that introduces all the equality and inequality constraints into the fitness function of the problem, such that it penalizes all the constraint violations (see Eq. 25). In this experiment, the penalty method has been employed as a constraint handling method.

$$\min \Pi(X) = f(X) + \sum_{j=1}^{n} \mu_j g_j^2(X) + \sum_{k=1}^{p} \upsilon_k q_k^2(X) \tag{25}$$

In this equation, n and p stand for total number of inequality and equality constraints that are violated by a solution vector (X), respectively; g_j and q_k represent the costs of jth inequality and ith equality constraints achieved by X, respectively; and $\mu(> 0)$ and $\upsilon(> 0)$ are the magnitudes employed for the inclusion of constraints in the fitness function. These values should be fine-tuned, as an excessively large value considerably increases the complexity of the search space and feasible solutions may not be found throughout the run; an excessively small value, however, leads to a constraint violation problem as it does not yield a considerable cost to the problem fitness. It is empirically observed that EFO is able to yield feasible solutions when μ and υ are taken as 10^4. Therefore, they are set to 10^4 in the experiments. The penalty function is equal to the fitness function, i.e., $\Pi(X) = f(X)$, when a solution found is feasible.

In this experiment, the five most popular real-world engineering benchmark problems have been chosen: *welded-beam*, *spring design*, *pressure vessel*, *three-bar truss design*, and *speed reducer*. These problems are introduced in the following paragraphs and their definitions are given in Appendix E.

4.4.1.1. Welded-beam design problem:

The main goal is to manufacture a beam that is made of low-carbon steel (C-1010) and welded to a rigid member within a minimum fabrication cost. This problem was introduced

to the literature as a benchmark structural engineering problem by Rao [174]. Figure 4.7. presents the beam and the member.

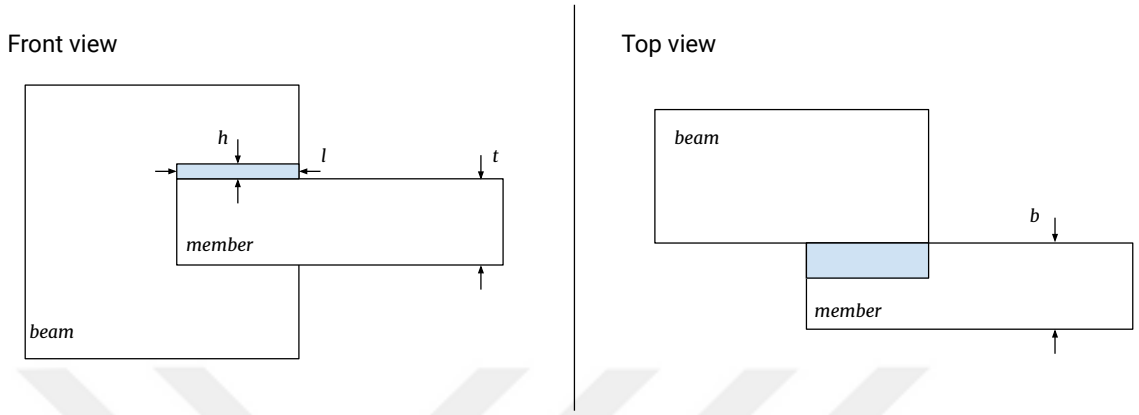

Figure 4.7. Welded-beam design.

The problem has four design parameters and five inequality constraints. The parameters to be optimized are h (x_1), the thickness of the weld; l (x_2), the length of the welded joint; t (x_3), the width of the member; and b (x_4), the thickness of the member. The first two variables are discrete and are multiples of 0.0065. EFO is adopted by rounding the values of these parameters to the nearest integer. The bound constraints of these parameter variables are: $0.125 \leq x_1 \leq 5$ and $0.1 \leq x_2, x_3, x_4 \leq 10$.

4.4.1.2. Spring design problem:

The purpose is to obtain a minimum-weight spring to achieve its optimal parameter values [175]. The problem with the variables to be optimized is demonstrated in Figure 4.8..

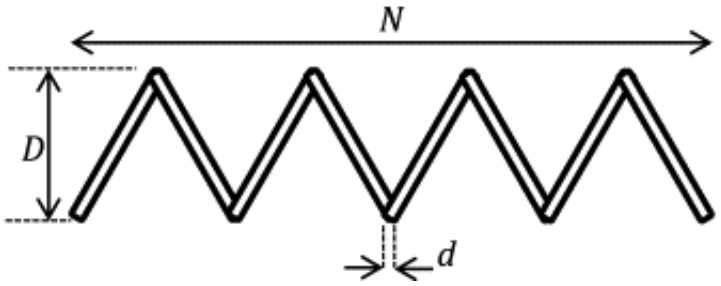

Figure 4.8. Spring design problem.

There are three design variables and four inequality constraints in this problem. The wire diameter *d* (x_1), the mean diameter *D* (x_2), and the number of active coils *N* (x_3) are the design variables of this problem. The bound constraints of these parameters are 0.05 $\leq$ x_1 $\leq$1, 0.25 $\leq$ x_2 $\leq$ 1.3, and 2 $\leq$ x_3 $\leq$ 15.

4.4.1.3. Pressure vessel problem:

The pressure vessel problem, where the objective is to minimize the cost value including welding, material, and forming costs, has four design variables and four inequality constraints. The problem was proposed by Sandgren [176]. Figure 4.9. depicts the schematic of the problem with its design parameters. The thickness of the shell T_s (x_1), the thickness of the head T_h (x_2), the inner radius *R* (x_3), and the length of the cylindrical section of the vessel *L* (x_4) are the design variables.

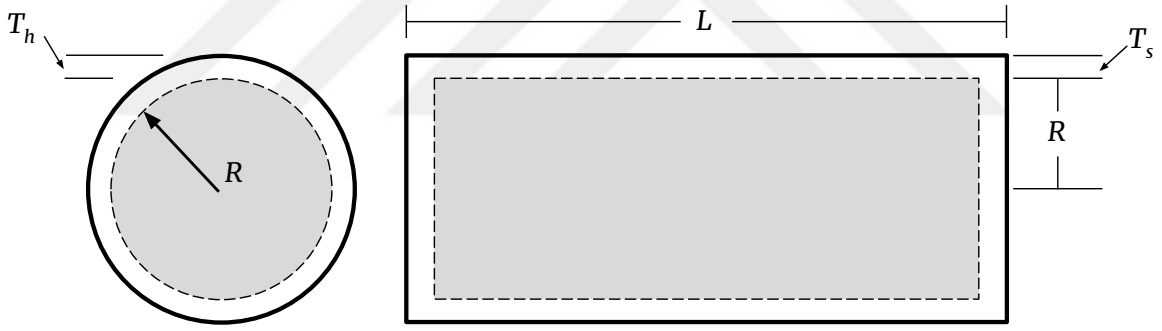

Figure 4.9. Pressure vessel design problem.

The bound constraints of these design variables are: 1$\times$0.0625$\leq$ x_1, x_2 $\leq$99$\times$0.0625 and 10 $\leq$ x_3, x_4 $\leq$ 200. The first two variables (the thicknesses) are discrete and multiples of 0.0625. As in the welded problem, EFO is adopted here by rounding the values of these parameters to the nearest integer to handle discrete variables.

4.4.1.4. Three-bar truss design:

The three-bar truss design problem is one of the well-known constrained mechanical design problem in the literature, where the objective is to design a three-bar truss through minimizing the volume. The schematic is given in Figure 4.10..

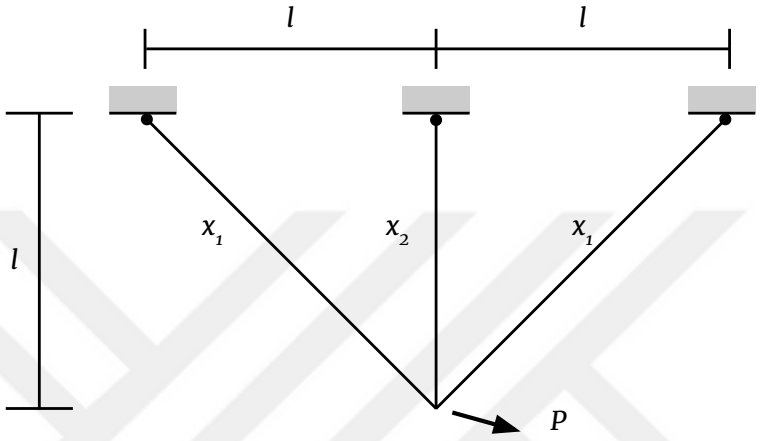

Figure 4.10. Three-bar truss design problem.

As shown in the schematic (Figure 4.10.), there are two parameters to be optimized in this problem, which represent the cross-sectional areas (x_1 and x_2). The bound constraints of these parameters are $0 \leq x_1, x_2 \leq 1$. The problem also has three inequality design constraints.

4.4.1.5. Speed reducer:

The speed reducer problem, proposed by Golisnki [177], represents the design of a gearbox. The objective of this problem is to minimize the total weight of the speed reducer. The schematic of the speed reducer problem is given in Figure 4.11..

As seen from Figure 4.11., there are seven design variables (three of these, from x_2 to x_4, lie at the boundaries of the feasible search space) and eleven design constraints, four of which are linear and seven are nonlinear, which makes this problem a much more challenging benchmark [178]. The face width d (x_1), module of teeth m (x_2), number of teeth in the pinion z (x_3), length of the first shaft between bearings l_1 (x_4), length of the second shaft

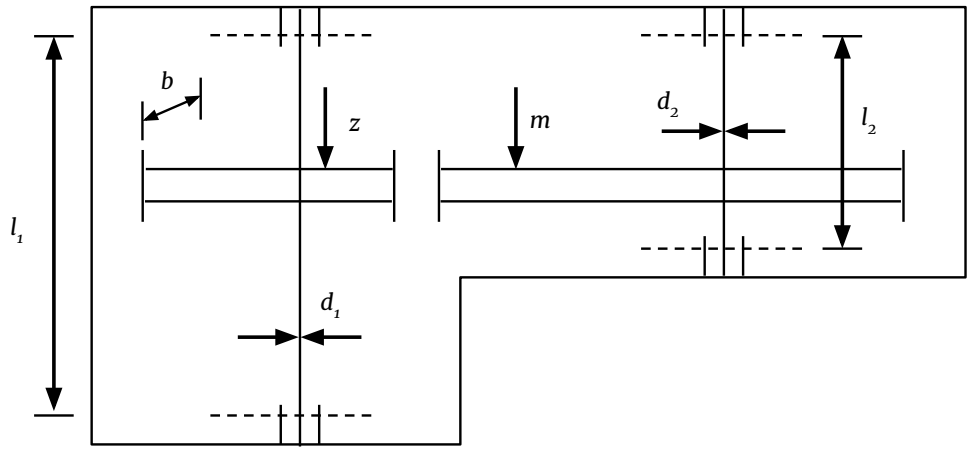

Figure 4.11. Speed reducer design problem.

between bearings l_2 (x_5), diameter of the first shaft d_1 (x_6), and diameter of the second shaft d_2 (x_7) are the design parameters to be optimized. The bound constraints are: $2.6 \leq x_1 \leq 3.6$, $0.7 \leq x_2 \leq 0.8$, $17 \leq x_3 \leq 28$, $7.3 \leq x_4 \leq 8.3$, $7.8 \leq x_5 \leq 8.3$, $2.9 \leq x_6 \leq 3.9$, and $5 \leq x_7 \leq 5.5$.

4.4.2. Results

The performance of EFO has been tested through 30 separate runs for every problem introduced in Section 4.4.1., as in the first experiment. A cost function in which fitness function of the problem as well as all its constraints are included through the penalty method has been targeted for optimization in this experiment. The obtained best, average, and worst performances as well as the average search time, population size, and function evaluation numbers are given in Table 4.13.

Table 4.13. General performance of EFO on the design problems.

						Optimal parameter values			
Problem	**Best**	**Mean**	**Worst**	**Std. Dev.**	**Avg. Time (s)**	α	K	Pop. Size	**FEN**
welded-beam	1.8555	2.2719	2.4984	0.1626	17.714	0.664	9	15	30,000
spring design	0.012650	0.012703	0.012789	4.23e-05	10.508	0.990	12	15	30,000
pressure vessel	6,059.60	6,102.87	6,138.60	20.998	12.038	0.989	13	25	50,000
truss design	263.895409	263.8954	263.8955	3.62e-05	5.0707	0.588	5	10	30,000
speed reducer	2,996.243	2,996.243	2,996.243	5.19e-08	1.923	0.989	6	10	10,000

From the statistical findings in Table 4.13., it can be concluded that EFO showed a consistent performance on the spring design, three-bar truss design, and speed reducer problems, whereas it produced results with much higher deviation on the pressure vessel problem. In spite of the much higher evaluation number used in the pressure vessel problem than that used in the other problems, it is seen that the average time taken for EFO to complete the search process is less than that in the welded-beam problem. This is due to the computational complexity of the constraints of the welded-beam problem.

Detailed comparison of the best performance of EFO with the already proposed approaches in the literature are presented in Tables 4.15. to 4.19. for the welded-beam, spring design, pressure vessel, three-bar truss, and speed reducer problems. The minimum cost obtained, optimal design parameter values to ensure a minimum cost, as well as the FENs to find this cost are provided in these tables. The constraint values of the design problems obtained by the EFO algorithm are given in Table 4.14.

Table 4.14. Constraint values of the best solution obtained by EFO for design problems.

	welded-beam	**spring design**	**pressure vessel**	**speed reducer**	**truss design**
g(1)	-22.3465	-0.0003	0.0000	0.0000	-0.0739
g(2)	-959.2553	0.0000	-0.0359	-1.4642	-0.1980
g(3)	-0.0023	-4.0382	-1.5256	-0.5358	-0.4992
g(4)	-3.3271	-0.7326	-63.3700	-	-0.9015
g(5)	-0.1090	-	-	-	0.0000
g(6)	-0.2352	-	-	-	0.0000
g(7)	-2770.9099	-	-	-	-0.7025
g(8)	-	-	-	-	0.0000
g(9)	-	-	-	-	-0.5833
g(10)	-	-	-	-	-0.0513
g(11)	-	-	-	-	-0.0109

Table 4.15. Comparison of the best EFO performance with other approaches on solving the *welded-beam design* problem.

Researchers	Method	FEN	$x_1(h)$	$x_2(l)$	$x_3(t)$	$x_4(b)$	Cost
Akay and Karaboga [135]	ABC	30,000	0.2057	3.4704	9.0366	0.2057	1.7249
Aragon et al. [68]	TCA	320,000	0.2444	6.2186	8.2915	0.2444	2.3811
Bernardino et al. [86]	GA-AIS	320,000	0.2444	6.2183	8.2915	0.2444	2.3812
Bernardino et al. [85]	GA-AIS	320,000	0.2443	6.2202	8.2912	0.2444	2.3812
Datta and Figueira [76]	PSO	N.A.	0.1875	1.7821	8.2500	0.2500	1.9553
Gandomi et al. [67]	ISA	30,000	0.2443	6.2199	8.2915	0.2443	2.3812
Gandomi et al. [67]	CSGA	30,000	0.2443	6.2199	8.2915	0.2443	2.3812
Han et al. [73]	STA	60,000	0.2053	3.2603	9.0366	0.2057	1.6956
Kanagaraj et al. [87]	CSGA	25,000	0.2443	6.2175	8.2915	0.2444	2.3809
Montes and Ocana [71]	BFO	48,000	0.2057	3.4711	9.0367	0.2057	2.3868
Wang et al. [136]	BSA	60,000	0.2057	3.4704	9.0366	0.2057	1.7249
Zhang et al. [79]	DE	24,000	0.2444	6.2175	8.2915	0.2444	2.3810
Zhang et al. [84]	EA	28,897	0.2443	6.2201	8.2940	0.2444	2.3816
Present Study	EFO	30,000	0.2340	3.1135	8.5707	0.2362	1.8555

From the comparative cost values in Table 4.15., it can be seen that EFO showed better performance than TCA [68], GA-AIS [85, 86], PSO [76], ISA [67], and BFO [71], with a much lower FEN. DE [79], EA [84], and CSGA [87] are also inferior to EFO, but with a lower FEN than EFO. ABC [135], STA [73], BSA [136] are better methods for this problem than EFO; however, they all considered the first two parameters as continuous, when in fact they should be discrete and integer multiples of 0.0065 [178].

The results obtained from the spring design problem (see Table 4.16.) reveal that EFO showed superior performance on the spring design problem in comparison to its all competitors. It can also be seen that Q-PSO [75] and ISA [67] yield their results with the lowest FENs, which are 2,000 and 8,000, respectively. NDE [78] and DE [79] used fewer function evaluations than EFO to complete their search processes.

From the best cost values obtained from the pressure vessel design problem, as shown in Table 4.17., it can be concluded that EFO was able to produce the lowest welding, material, and forming costs compared to the other methods, except for FOA [89] and STA [73]. However, the both methods considered the first two parameters as continuous. ISA [67] and Q-PSO [75] are the two methods that complete the search with the lowest FENs to produce their costs for this problem.

Table 4.16. Comparison of the best EFO performance with other approaches on solving the *spring design* problem.

Researchers	Method	FEN	$x_1(d)$	$x_2(D)$	$x_3(N)$	Cost
Akay and Karaboga [135]	ABC	30,000	0.051749	0.358179	11.203763	0.012665
Aragon et al. [68]	TCA	36,000	0.051622	0.355105	11.384534	0.012665
Askarzadeh [69]	CSA	50,000	0.0516890284	0.3567169544	11.2890117993	0.0126652328
Bernardino et al. [86]	GA-AIS	36,000	0.051660806	0.35603234	11.329555	0.012666
Bernardino et al. [85]	GA-AIS	36,000	0.0514305	0.3505298	11.6611924	0.012666
Dos Santos Coelho [75]	Q-PSO	2,000	0.051515	0.352529	11.538862	0.012665
Du et al. [89]	FOA	25,000	0.05206590	0.36570924	10.78621813	0.01267607
Gandomi et al. [67]	ISA	8,000	N.A.	N.A.	N.A.	0.012665
Han et al. [73]	STA	60,000	0.0516800	0.3565001	11.3018335	0.01266534
Mohammed [78]	NDE	24,000	0.051689058	0.35671768	11.28896875	0.01266523
Montes and Ocana [71]	BFO	48,000	0.051825	0.359935	11.107103	0.012671
Wang et al. [136]	BSA	60,000	0.051743	0.358017	11.213187	0.012665
Zhang et al. [79]	DE	24,000	0.05169	0.35672	11.289	0.012665233
Present Study	EFO	30,000	0.051496	0.35248	11.523	0.012650

Table 4.17. Comparison of the best EFO performance with other approaches on solving the *pressure vessel design* problem.

Researchers	Method	FEN	$x_1(T_s)$	$x_2(T_h)$	$x_3(R)$	$x_4(L)$	Cost
Akay and Karaboga [135]	ABC	30,000	0.8125	0.4375	42.098446	176.636596	6,059.7147
Aragon et al. [68]	TCA	80,000	0.8125	0.4375	42.098429	190.787695	6,390.554
Askarzadeh [69]	CSA	250,000	0.8125	0.4375	42.09844539	176.63659855	6,059.7144
Bernardino et al. [86]	GA-AIS	80,000	0.8125	0.4375	42.0973	176.6509	6,059.8546
Bernardino et al. [85]	GA-AIS	36,000	0.8125	0.4375	42.094967	176.67972	6,060.138
Dos Santos Coelho [75]	Q-PSO	8,000	0.8125	0.4375	42.0984	176.6372	6,059.7208
Du et al. [89]	FOA	25,000	0.7804	0.3849	40.3888	199.1172	5,894.5981
Gandomi [67]	ISA	5,000	0.8125	0.4375	42.09845	176.6366	6,059.714
Han et al. [73]	STA	60,000	0.7785	0.3848	40.3389	199.7753	5,886.45436
Mohammed [78]	NDE	20,000	0.8125	0.4375	42.0984	176.63659	6,059.71433

Table 4.17. (continued).

Researchers	Method	FEN	$x_1(T_s)$	$x_2(T_h)$	$x_3(R)$	$x_4(L)$	Cost
Montes and Ocana [71]	BFO	48,000	0.8125	0.4375	42.096394	176.683231	6,060.460
Wang et al. [136]	BSA	40,000	0.8125	0.4375	42.098497	176.635967	6,059.708274
Present Study	EFO	50,000	0.8125	0.4375	42.099	176.6300	6,059.60

Table 4.18. Comparison of the best EFO performance with other approaches on solving the *three-bar truss design* problem.

Researchers	Method	FEN	x_1	x_2	Cost
Askarzadeh [69]	CSA	25,000	0.7886751284	0.4082483080	263.895843
Liu et al. [74]	PSO-DE	17,600	0.7886751347	0.4082482900	263.895843
Mohammed [78]	NDE	4,000	0.7886753196	0.4082477671	263.895843
Wang et al. [136]	BSA	7,500	0.788675	0.408248	263.895843
Zhang et al. [79]	DE	15,000	0.7886751359	0.4082482868	263.895843
Present Study	EFO	30,000	0.7886970484	0.4081771662	263.895409

The results in Table 4.18. obtained from the three-bar truss design problem suggest that EFO showed better optimization performance than all the competitor methods, but with more function evaluations. Among the competitor methods, BSA [78] showed the fewest function evaluations.

As seen from the results in Table 4.19., EFO showed overwhelmingly better performance than ABC [135], GA-AIS [86], CSGA [87], PSO-DE [74], EA [83], DE [77], and TLBO [72] on the *speed reducer* problem with fewer function evaluations. NDE [78], GA-AIS [85], and BSA [136], however, were shown to be better methods to solve this problem. It is worth mentioning here that although EFO has become inferior to BSA [136] it showed better performance overall as the mean value yielded by BSA is much higher than that of EFO, at 2,998.0101. Given that the *speed reducer* problem differs from the other design problems in the best feasible solution where two of the parameters lie at the boundaries, it can be

concluded that EFO can still competitively handle the problems with such a complex search space.

Table 4.19. Comparison of the best EFO performance on solving the *speed reducer design* problem.

Researchers	Method	FEN	$x_1(d)$	$x_2(m)$	$x_3(z)$	$x_4(l_1)$	$x_5(l_2)$	$x_6(d_1)$	$x_7(d_2)$	Cost
Akay and Karaboga [135]	ABC	30,000	3.49	0.7	17	7.3	7.8	3.3502	5.2878	2,997.0584
Bernardino et al. [86]	GA-AIS	36,000	3.5	0.7	17	7.3	7.8	3.3502	5.2866	2,996.3483
Bernardino et al. [85]	GA-AIS	36,000	3.5	0.7	17	7.3	7.7153	3.3502	5.2866	2,994.4712
Kanagaraj et al. [87]	CSGA	25,000	3.5	0.7	17	7.6050	7.8181	3.35	5.2687	2,996.3482
Liu et al. [74]	PSO-DE	54,350	3.5	0.7	17	7.3	7.8	3.3502	5.2866	2,996.3481
Mohammed [78]	NDE	18,000	3.5	0.7	17	7.3	7.7153	3.3502	5.2866	2,994.4710
Montes et al. [83]	EA	36,000	3.5061	0.7	17	7.4601	7.9621	3.3629	5.3089	2,996.3566
Montes et al. [77]	DE	24,000	3.5	0.7	17	7.3	7.8	3.3502	5.2866	2,996.3566
Rao and Vakharia. [72]	TLBO	10,000	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	2,996.3481
Wang et al. [136]	BSA	7,500	3.49	0.7	17	7.3	7.7153	3.3502	5.2866	2,994.4683
Present Study	EFO	10,000	3.5	0.7	17	7.3	7.8	3.3502	5.2865	2,996.2430

5. MULTI-OBJECTIVE ELECTRIC FISH OPTIMIZATION ALGORITHM

5.1. Multi-objective Optimization

Many real-life problems have two or more conflicting objectives to optimize. For example, many solutions take into account not only the accuracy of the solution, but also its efficiency. Multi-objective optimization aims to optimize these objectives simultaneously. A general MOOP is expressed in the following form:

Definition 1. (Multi-Objective Optimization Problem) A type of problem where the objective is to find a vector $\overrightarrow{\boldsymbol{x}^*} = [\boldsymbol{x}_1^*, \boldsymbol{x}_2^*, \ldots, \boldsymbol{x}_k^*]$ that minimizes the function vector $f(\overrightarrow{\boldsymbol{x}^*})$ satisfying the p inequality constraints $g(\overrightarrow{\boldsymbol{x}^*})$ and q equality constraints $h(\overrightarrow{\boldsymbol{x}^*})$. Considering $\vec{\boldsymbol{x}}$ is a decision variable vector, a function and constraint vectors can be written as follows[2]:

$$\underset{X}{\textit{minimize}}\ f(\vec{\boldsymbol{x}}) = [f_1(\vec{\boldsymbol{x}}), f_2(\vec{\boldsymbol{x}}), \ldots, f_n(\vec{\boldsymbol{x}})],$$

$$\begin{aligned} &subject\ to \\ &\quad g(\vec{\boldsymbol{x}}) = [g_1(\vec{\boldsymbol{x}}), g_2(\vec{\boldsymbol{x}}), \ldots, g_p(\vec{\boldsymbol{x}})] \leq 0, \\ &\quad h(\vec{\boldsymbol{x}}) = [h_1(\vec{\boldsymbol{x}}), h_2(\vec{\boldsymbol{x}}), \ldots, h_q(\vec{\boldsymbol{x}})] = 0 \end{aligned}$$

The region constrained by g and h is *feasible region* (Ω) and any solution vector ($\vec{\boldsymbol{x}}$) within this region (i.e., $\vec{\boldsymbol{x}} \in \Omega$) is called as *feasible solution*.

As can be concluded from the definition, the fundamental distinction between MOOPs and SOOPs is the number of objectives to be optimized. The objectives in the MOOPs are represented within the objective space, an n-dimensional space completely different from the search (or decision) space. While there exists only a single solution that represents the global

[2]In case when the goal is maximization, *minimize* should be changed to *maximize*.

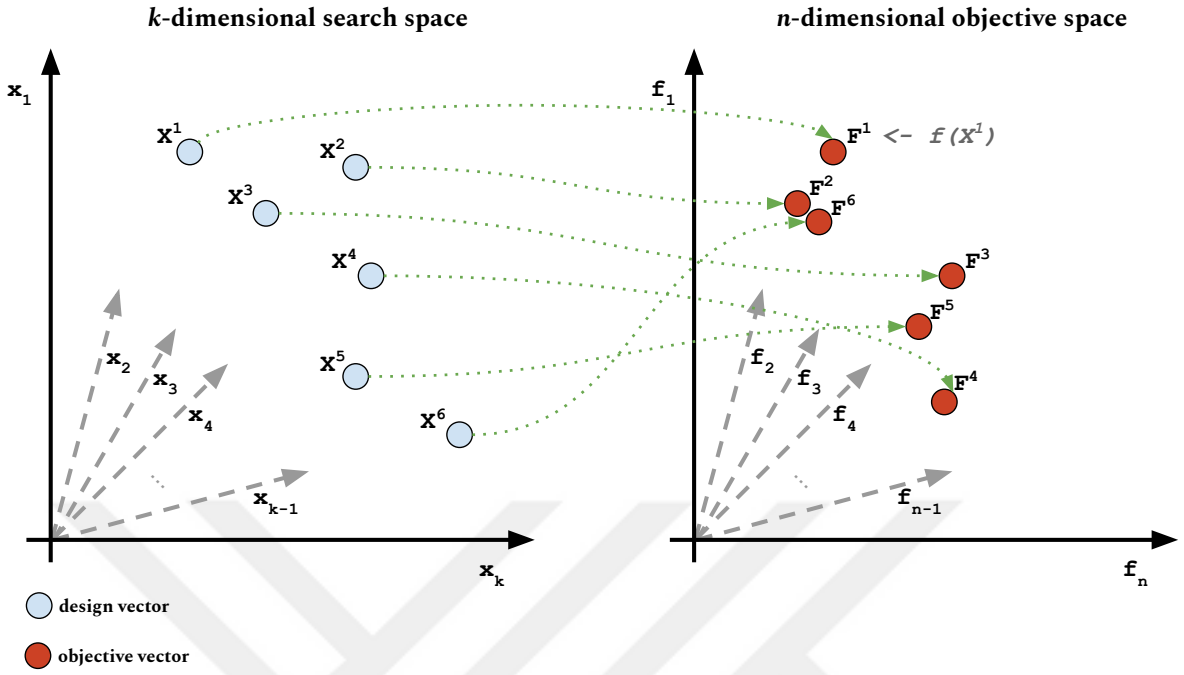

Figure 5.1. Illustration of the search space and the objective space.

optimum in SOOPs, MOOPs can have more than one solution that represent optimum solutions. Each of the optimum solution found from the search space of MOOP can be shown in the objective space. Figure 5.1. reveals an illustration of k- and n-dimensional search and objective space, respectively. Here, X^1 through X^6 represent the design vectors; whereas F^1 through F^6 represent the corresponding the objective vectors.

There are many approaches proposed in literature in order to discover solutions for MOOPs. Such approaches are categorised with the four classes: *aggregating*, *lexicographic*, *subpopulation*, and *Pareto-based* [118].

Aggregating approach is the simplest approach to handle multiple objectives. The objectives are scalarized in this approach through the existing methods like weighting sum approach, ϵ-constraint approach, and the like. Hence, MOOP can be mapped into a SOOP. The performance of aggregating approach is highly dependant on the *mapper* approach.

Lexicographic approach ranks the objectives with respect to the importance given by the system designer then optimizes the objective functions separately in the order of their ranks.

While this approach is useful only when few objectives are used, it may tend to favor more particular objectives as the number of them increases, which is the main weakness of this approach [179].

Subpopulation approach decomposes the population into the subpopulations. So every subpopulation is assigned for each objective of the problem and behaves like a separate single-objective optimizer. As in the lexicographic approach, the objectives are handled separately here as well. Subpopulations somehow exchange information among them, then they are recombined to provide a trade-off among different solutions.

Pareto-based approach is the most popular approach, hence the majority of the proposed approaches in the literature falls in this category [110]. In this approach, the solution to MOOPs is generally not unique, but a set of optimal, non-dominated solutions called Pareto optimal. A decision vector $\vec{\boldsymbol{x}}$ is said to dominate another decision vector $\vec{\boldsymbol{y}}$, if no criterion of $f(\vec{\boldsymbol{x}})$ is worse than the corresponding component of $f(\vec{\boldsymbol{y}})$ and at least one criterion of $f(\vec{\boldsymbol{x}})$ is better. So, the final set comprises solutions that are not dominated. Hence, the Pareto front represents different trade-offs among conflicting objectives that decision-makers could choose the most appropriate solution according to their needs.

The definition of the basic concepts of Pareto-based multi-objective optimization approaches have been given in the following. It is important to note that the following Pareto definitions are made with respect to the notations in MOOP definition (refer to Definition 1).

Definition 2. (Pareto Dominance) A k-dimensional solution $\vec{\boldsymbol{u}} = [u_1, \ldots, u_k]$ is said to dominate another k-dimensional solution $\vec{\boldsymbol{v}} = [v_1, \ldots, v_k]$ (denoted as $\vec{\boldsymbol{u}} \prec \vec{\boldsymbol{v}}$); if $f(\vec{\boldsymbol{u}})$ is partially less than $f(\vec{\boldsymbol{v}})$, i.e., $\forall i : f_i(\vec{\boldsymbol{u}}) \leq f_i(\vec{\boldsymbol{v}}) \wedge \exists i : f_i(\vec{\boldsymbol{u}}) < f_i(\vec{\boldsymbol{v}}) \mid i \in \{1, \ldots, n\}$; $\vec{\boldsymbol{u}}, \vec{\boldsymbol{v}} \in \Omega$.

Definition 3. (Pareto Optimal) A solution $\vec{\boldsymbol{u}}$ is Pareto optimal if there is no other solution in Ω that dominates $\vec{\boldsymbol{u}}$, i.e., $\neg\exists\vec{\hat{\boldsymbol{u}}} \in \Omega : \vec{\hat{\boldsymbol{u}}} \prec \vec{\boldsymbol{u}}$.

Definition 4. (Pareto Optimal Set) A set $\mathcal{P}^*$ that contains all the Pareto optimal solutions, i.e., $\mathcal{P}^* = \{\vec{\boldsymbol{u}} \in \Omega \mid \neg\exists\vec{\hat{\boldsymbol{u}}} \in \Omega : \vec{\hat{\boldsymbol{u}}} \prec \vec{\boldsymbol{u}}\}$.

Definition 5. (Pareto Front) A set $\mathcal{PF}^*$ that contains all the objective vectors of the Pareto optimal set ($\mathcal{P}^*$), i.e., $\mathcal{PF}^* = \{f(\vec{\boldsymbol{u}}) \mid \vec{\boldsymbol{u}} \in \mathcal{P}^*\}$.

In this thesis, three different implementations of EFO have been introduced by adopting some concepts used for MOOPs. The main motivation of here is to enhance the promising EFO algorithm for solving MOOPs. Some known schemes like non-dominated sorting proposed in NSGA-II [16] have been included and a moderate number of modifications have been made to the EFO algorithm to adapt EFO for solving MOOPs, to enhance the search performance on this problem type, and to reduce the computation complexity of EFO. As a result of all these modifications, PD-EFO, NS-EFO, and CB-EFO algorithms have been introduced and shared with the community [31]. The proposed implementations of EFO have been applied to solve well-known MOOP families and have been compared with each other with respect to the most popular metrics used for performance evaluations: HV, SPREAD, EPSILON, IGD. Then, the best-performing implementation (CB-EFO) has been reevaluated to reveal its optimization performance in comparison to the well-known approaches in the literature, namely NSGA-II [16], SPEA2 [17], IBEA [18], MOEAD [19], and MOCell [20]. The experimental results show that proposed CB-EFO algorithm outperforms other algorithms in evaluations that are carried out by using all metrics but SPREAD. It only takes the second place on evaluations that are made by using the SPREAD metric.

The design details of the proposed algorithms (PD-EFO, NS-EFO, and CB-EFO) are explained in details int the following sections.

5.2. Pareto Dominance-based EFO Algorithm

This is the base multi-objective optimization implementation of EFO where the least changes have been applied. The main goal of this implementation is to measure the search capability of the proposed EFO algorithm on the MOOPs. Therefore, the general search framework of EFO given in Section 3. has been preserved. However, small modifications have been applied so that proposed single-objective EFO can be applicable to MOOPs. These modifications have been outlined in the followings and detailed thereafter:

1. Pareto dominance-based frequency update,

2. Pareto dominance-based acceptance,

3. Keeping *elite* individuals through a Pareto dominance set.

5.2.1. Frequency update

The frequency update in EFO is strongly dependant on only a single objective. This approach is not applicable for solving MOOPs where at least two conflicting objectives take part. Therefore, this procedure should be made adaptable for such problems.

The total number of solutions that are dominated by every solution could be a good indicator in revealing how good a solution found is because the quality of a solution for MOOPs is mainly evaluated through *Pareto dominance* approach. As better individuals are more promoted to perform local search in EFO, they are expected to have higher frequency when compared to other, particularly poor, individuals. In order to ensure the same functionally in PD-EFO, the frequency calculation of ith individual is made based on its *Pareto dominance* and *crowding distance*.

Crowding distance is a metric proposed in NSGA-II [16] to reveal how two solutions are far away from each other. Hence, it is a good indicator to find out how diverse the solutions are in the objective space. Algorithm 3 calculates and returns crowding distance measurement of a given solution set.

An illustration of the crowding distance measurement is given in Figure 5.2.. In this figure, the crowding distances of solution a and b are set to *Infinity* so that they are selected in the next generations in NSGA-II, which increases the divergence of the solutions in the objective space. Considering the algorithm and the figure, the crowding distance measurement for the ith individual is calculated as given in the Equation 26.

$$CrowDist_i = \left(\frac{f_1^{i+1} - f_1^{i-1}}{f_1^b - f_1^a} \right) + \left(\frac{f_2^{i-1} - f_2^{i+1}}{f_2^a - f_2^b} \right) \tag{26}$$

Algorithm 3: Crowding distance calculation proposed in NSGA-II [16].

def *crowding_distance_assignment(S: Solutions, M: Objectives)*:
  $l = |S|$; // number of solutions in S
  foreach $i \in \{1, 2, \ldots l\}$ **do** // initialize distance
    $S[i]_{distance} \leftarrow 0$;
  end
  foreach $m \in M$ **do**
    $S \leftarrow sort(S, m)$; // sort using each objective value
    $S[1]_{distance} \leftarrow S[l]_{distance} \leftarrow \infty$; // extreme points are set to Inf.
    for *i=2; i < (l-1); i++* **do** // for all other points
      $S[i]_{distance} \leftarrow S[i]_{distance} + \left(\frac{f_m^{i+1} - f_m^{i-1}}{f_m^{max} - f_m^{min}}\right)$
    end
  end
  return S

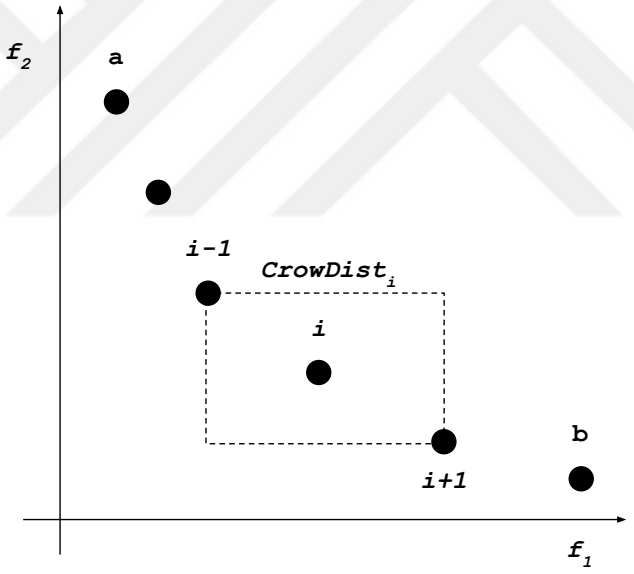

Figure 5.2. An example of the crowding distance calculation for the ith individual.

In PD-EFO, the frequency update mechanism of EFO has been replaced with the Algorithm 4. This algorithm simply states that, the more an individual dominates other individuals, or the higher the distance an individual has; the higher frequency it would have. Hence, the frequency of every individual can range from 0 (for the worst individual) to 1 (for the best individual). Contrary to the frequency, the way to update amplitude of every individual is reserved here as in the single-objective EFO algorithm.

Algorithm 4: Frequency update mechanism in PD-EFO.

```
def frequency_update(S: Population Size):
    n_i, n_j = 0;
    for (i = 0; i < S; i++) do
        for (j = 0; j < S; j++) do
            if N_i ≺ N_j then                                   // if i dominates j
                n_i ← n_i + 1;
            else if N_j ≺ N_i then                              // if j dominates i
                n_j ← n_j + 1;
            else                          // if both do not dominates each other
                CrowDist_i, CrowDist_j ← get crowding distances of i and j;
                if CrowDist_i > CrowDist_j then  // if i has higher distance
                 than j
                    n_i ← n_i + 1;
                else                                    // if j has higher distance
                    n_j ← n_j + 1;
                end
            end
        end
    end
    f_min ← min(n);
    f_max ← max(n);
    for (u = 0; u < S; u++) do
        f_u ← (n_u − f_min)(f_max − f_min);
    end
    return f
```

5.2.2. Acceptance of new individuals

In single-objective EFO, *greedy selection* can replace an individual with a candidate individual (x_{cand}). However, greedy selection takes only the single objective into consideration in order to accept or to reject the candidate solution (Equation 27 –for *minimization* problem).

$$x_i = \begin{cases} x_{cand} & f(x_{cand}) \leq f(x_i) \\ x_i & else \end{cases} \tag{27}$$

As MOOPs possess at least two conflicting objectives, instead of greedy selection, a Pareto dominance-based selection mechanism has been integrated here for PD-EFO:

$$x_i = \begin{cases} x_{cand} & x_{cand} \prec x_i \\ x_i & else \end{cases} \tag{28}$$

5.2.3. Archive for elite individuals

Single-objective heuristic yields elite/best individual that ensures the optimal solution toward the end of the search process, which is expected to be global minimum that the algorithm is able to converge. However, the concept does not fit for MOOPs and it is expected from multi-objective algorithms to offer more than one solution for designer's choice. The output of the algorithm should be non-dominated solutions in order to represent the Pareto front. Therefore, in PD-EFO, an external archive, which is a typical 'set-like' data structure that stores all the non-dominated solutions throughout the run, is used to avoid the promising solution to be lost throughout the search. It is worth noting here that non-dominated solutions in the archive are not allowed to contribute the search in PD-EFO.

The pseudo-code of this mechanism is given in Algorithm 5. From the algorithm, it can be seen that *i*) a solution p is inserted into the archive Arc if and only if it is not dominated by any solution in the archive, *ii*) any solution is removed from the archive if it is dominated by p, and *iii*) a solution having the least crowding distance is removed from Arc when the number of non-dominated solutions in Arc exceeds the allowed capacity C.

An example of the solution update mechanism in the archive is demonstrated in Figure 5.3. on a minimization problem. In the figure, the archive initially comprises of four solutions (solution *1* through *4*) that do not dominate each other. In the case where a solution *5* is attempted to be added into the archive, the solution *3* will be removed from the archive because the solution *5* dominates the other solution *3* with respect to all objectives (Figure 5.3.a.).

Algorithm 5: An archive mechanism that keeps the non-dominated solutions.

```
def update_archive(N: Population, M: Objectives, Arc: Archive, C: Archive Capacity):
    foreach p ∈ N do
        n_p ← 0;
        foreach q ∈ Arc do
            if p ≺ q then                                          // if p dominates q
                Arc ← Arc \ {q};                          // remove the solution q from Arc
            else if !(p ≺ q) ∧ !(q ≺ p) then                // if p and q do not dominate
                n_p ← n_p + 1;                  // increase the domination counter for p
            end
        end
        if n_p = |Arc| then
            Arc ← Arc ∪ {p};                                // add the solution p into Arc
            if |Arc| > C then                                              // if Arc is full
                S ← crowding_distance_assignment(Arc, M);                  // Algorithm 3
                k ← CrowDist_k = min(S);        // get solution having min. distance
                Arc ← Arc \ {k};                                  // remove k from Arc
            end
        end
    end
    return Arc;
```

Similarly, all the solutions will be removed from the archive when the solution *5* is added to archive because it Pareto dominates all the solutions in the archive (Figure 5.3.b.).

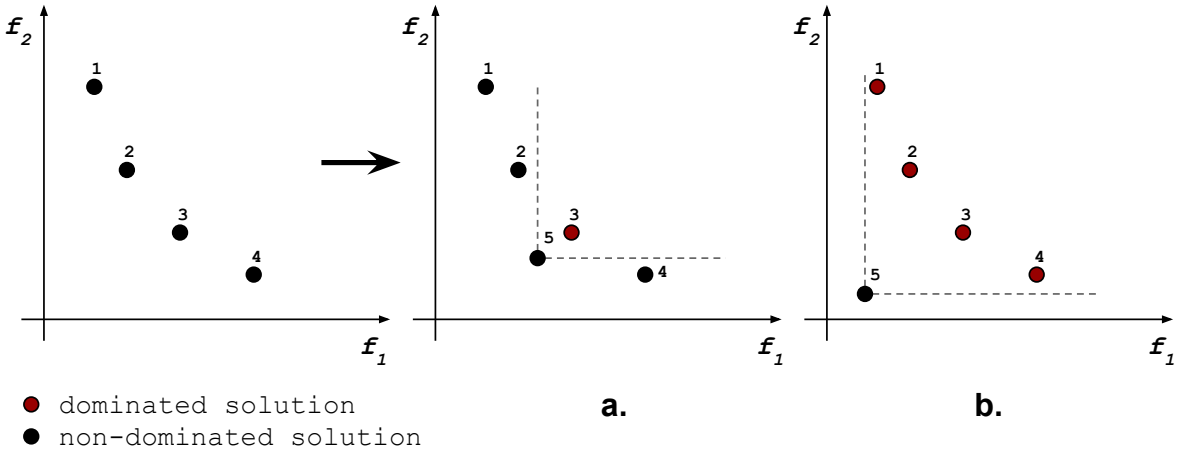

Figure 5.3. An example of the solution update in the archive.

The pseudo-code of PD-EFO algorithm is given in Algorithm 6.

Algorithm 6: Pseudocode of PD-EFO algorithm

Input: population size $|N|$, objective size $|M|$
Output: archive

```
population ← Initialize population;
Update frequency through Algorithm 4;
archive ← Update archive through Algorithm 5;
repeat
    for i = 1 to |N| do
        individual ← population[i];
        offspring ← {};
        if individual.frequency > rand then
            offspring ← active search(individual);
        else
            offspring ← passive search(individual);
        end
        if offspring ≺ individual then
            individual ← offspring;
    end
    Update frequency through Algorithm 4;
    archive ← Update archive through Algorithm 5;
until termination criterion is met;
```

5.3. Non-dominated Sorting-based EFO Algorithm

In NS-EFO, two different adaptation mechanisms are employed to the single-objective EFO. The first adaptation mechanism covers the frequency update of solutions, which mainly relies on the *Pareto fronts* and the *crowding distances* of solutions. The second adaptation, however, is made on the mechanism that the new candidate solutions are included to the population. These adaptations have been explained in the following two sections.

5.3.1. Frequency update

As stressed earlier, single-objective dependant frequency adjustment in standard EFO cannot be applied for MOOPs. Rather, an approach that achieves the same functionality as done in PD-EFO should be adapted to EFO.

In NS-EFO, a modification scheme based on Pareto ranking and the crowding distance of every individual is proposed for the frequency update mechanims. While f_{min} and f_{max} represent the worst and the best respectively in EFO (refer to Equation 2), they are evenly split into the fronts that represent the total Pareto ranking of the population here.

Individuals in the first front are placed into the first interval, those in the second front are placed in the second interval, and so on. Then the crowding distance is applied separately to every sub-populations in the intervals such that the higher distance an individual has, the closer the individual to the upper limit of that interval it has been placed. The calculation of the frequency of ith individual (f_i) in the population is given in Eq. 29.

$$f_i = \left(1 - \left(\frac{front_i}{front_{max}}\right) - dist_i\right) \tag{29}$$

where $front_i$ and $front_{max}$ represent the front at which the ith individual belongs to and the total number of fronts, respectively; while $dist_i$ is a crowding distance measurement for the ith individual ($CrowDist_i$) normalized for $front_i$, which is calculated as follows:

$$dist_i = \left(\left(1 - \frac{CrowDist_i - CrowDist_{min}}{CrowDist_{max} - CrowDist_{min}}\right) / front_{max}\right) \tag{30}$$

where $CrowDist_{min}$, and $CrowDist_{max}$ represents, respectively, the value of minimum and maximum crowding distance measurements obtained from $front_i$.

As in EFO, this updating mechanism ensures better individuals to have higher frequency and thus them to perform local search , and vice versa. The update procedure of individuals' amplitudes in EFO is not modified here, and hence it is calculated by using Eq. 3 and 29.

Figure 5.4. demonstrates the frequency update approach in NS-EFO. The figure is based on a scenario where the population is represented with four Pareto fronts in total such that each front comprises of different number of individuals. The proposed frequency update approach assigns the maximum and minimum frequency values to every front. Hence, individuals in the first front have a frequency value between 1.00 and 0.75, individuals in the

second front have a frequency value between 0.75 and 0.5, and so on. The distance of every individual is then calculated separately on every front with respect to the crowding distance measurement (refer to Equation 30) and then normalized. Here, the individuals having *Infinity* crowding distances represent the solutions where their objective vectors are located in the extreme points of the front, which are known to be solutions that increase the diversity of the population. In NS-EFO these individuals are excluded in distance calculation and they are assigned to the maximum frequency value of the front where they belong to. Hence, these individuals are enabled to perform local search and their probability of being selected by other individuals are increased. The assignment of distance and and frequency values for the first front is also given in the figure. The procedure to update frequency in NS-EFO is given in Algorithm 7.

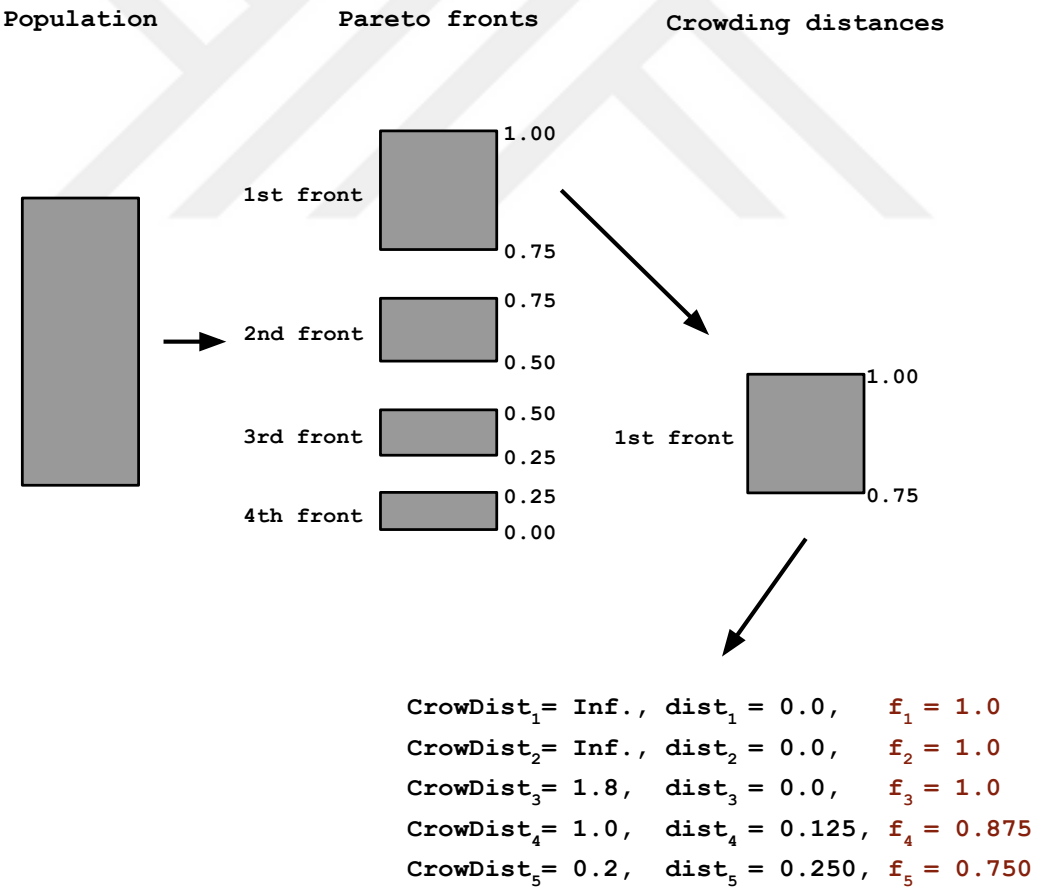

Figure 5.4. An example of the frequency update mechanism in NS-EFO

Algorithm 7: Frequency update phase of NS-EFO algorithm

```
Function update_frequency(population)
    fronts ← split population into the fronts;
    foreach front ∈ fronts do
        CrowDist ← calculate crowding distance of front;
        foreach individual ∈ front do
            individual.frequency ← calculate frequency through Equation 29;
        end
    end
end
```

5.3.2. Acceptance of new individuals

The *greedy selection* used both in single-solution EFO and PD-EFO allows only parent or offspring individual to survive for the next generation through a direct comparison on the objective(s). In NS-EFO, the greedy selection is replaced and the survival mechanism based on non-dominated sorting procedure proposed in NSGA-II [16] is adapted.

In NS-EFO, the parent and offspring individuals are merged and form a population with a size of $(2 \times N)$ and they are sorted according to the nondomination. The individuals in the first, second, and following fronts are taken for the next generation in order. The crowding distance of the individuals is taken into consideration only when all individuals in the front at turn exceed the capacity of (N). In such a case the individuals with higher crowding distance values are selected for the next generation. The remaining individuals (parent or offspring) are no more accepted and die out at that generation.

The survival procedure of NSGA-II is illustrated in Figure 5.5.. Here, both parent and offspring individuals can survive in the next generations as long as they belong to the promising fronts. NS-EFO does not employ a memory like *archive* in PD-EFO in order to store non-dominated solutions because solutions at the first front are always non-dominated solutions. Therefore, NS-EFO returns the first front as the Pareto front at the end of the search. The pseudo-code of NS-EFO algorithm is given in Algorithm 8.

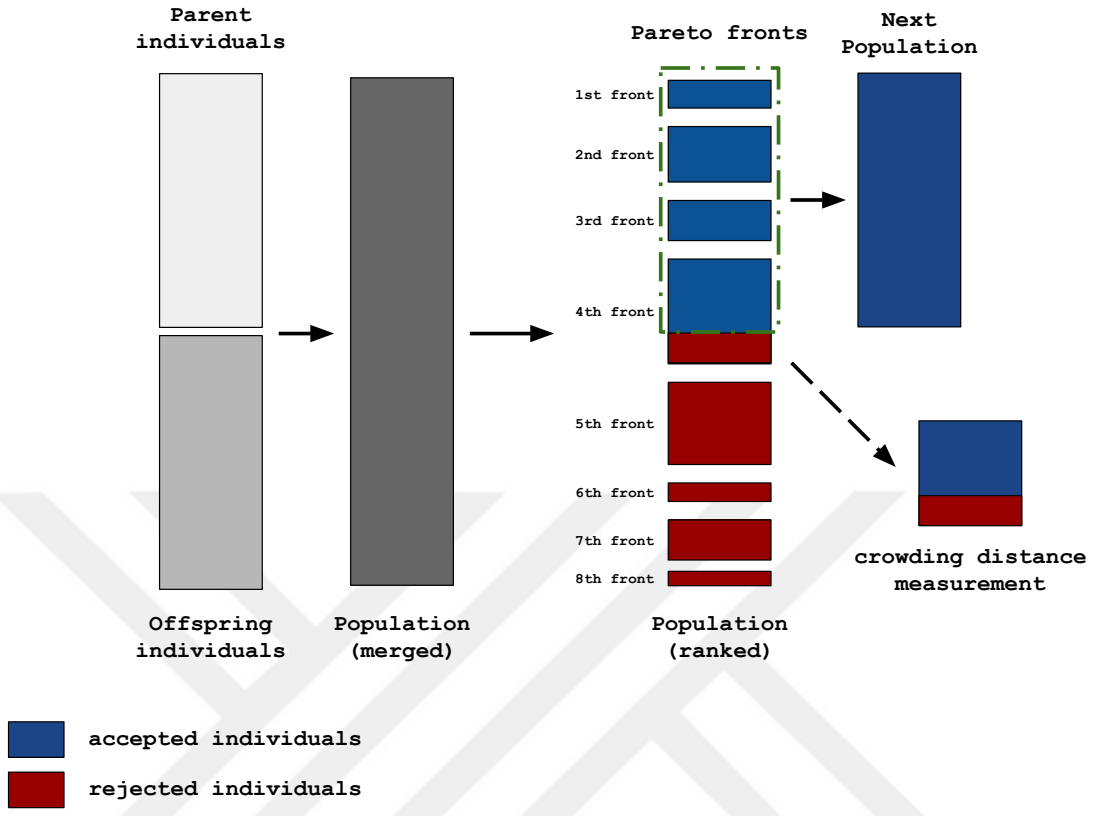

Figure 5.5. An example of non-dominated sorting in NSGA-II.

5.4. Cellular-based EFO Algorithm

While the general framework based on active and passive electrolocation is kept as in EFO, some modifications have been made to the base algorithm in order to adapt it for solving MOOPs (1-2). Furthermore, by taking into account the more complex search space of such problems, some changes are applied to strengthen the global and local search capabilities (3-4), and to reduce complexity of the algorithm (5). The modifications to the EFO algorithm is summarized in the followings:

1. The frequency updating of individuals (Eq. 2) has been modified. Hence the single fitness proportionate-based approach is replaced with a ranking- and crowding distance-based approach.

Algorithm 8: Pseudocode of NS-EFO algorithm

Input: population size $|N|$, objective size $|M|$
Output: Pareto optimal solutions

```
population ← Initialize population;
offspring ← {};
Update frequency through Algorithm 7;
repeat
    for i = 1 to |N| do
        individual ← population[i];
        if individual.frequency > rand then
            offspring[i] ← active search(individual);
        else
            offspring[i] ← passive search(individual);
        end
    end
    union ← Union(population, offspring)// merges two populations
    population ← Apply non-dominated sorting on union;
    Update frequency through Algorithm 7;
until termination criterion is met;
return Inidividuals at first front
```

2. The greedy selection has been replaced with a dominance-based selection and an external archive for storing Pareto optimal solutions is introduced.

3. The conditional acceptance in passive search has been improved in order to support promising solutions to perform their search in the vicinity of a non-dominated solution.

4. Binary tournament selection has been employed in active and passive search to ensure a fine-tuned local and global search.

5. The distance calculation in EFO has been excluded here for the sake of efficiency of the proposed algorithm. Instead, a *cellular-based* approach has been adopted.

5.4.1. Population Initialization

Here, the individuals are randomly generated as in EFO. So, Eq. 1 has been applied for the formation of initial population. In EFO, an individual needs to measure its distance to

the rest of the population for finding neighboring individuals in the sensing/active range. However, this distance calculation at each iteration increases the complexity of the algorithm. Therefore it is excluded in CB-EFO; instead the cellular-based approach is employed. Neighborhood in this approach has been organized in a random manner.

In the area of optimization, cellular-based approach was emerged from parallelization of genetic algorithm [22, 23]. This approach relies on the placement of every node in a toroidal mesh topology in which nodes correspond to individuals, and edges directly connected to nodes represents one-hop neighbors of that individual. Figure 5.6.a shows an exemplar mesh topology with 16 nodes. As a result of this topology, every individual intensively uses its nearby neighbors. Even if the neighborhood in this approach has no relation to the geographical position of neighbour individuals, it still well balances the *exploration* and *exploitation*. While exploration is ensured by a slow diffusion of solution through the population; exploitation is ensured by application of search operators within the neighborhood [20]. Every individual in CB-EFO has been directly connected with eights neighbors (see Figure 5.6.b).

After the population is generated through the adopted cellular-based approach, their frequency values are calculated and updated in every iteration. Here, the frequency update procedure proposed for NS-EFO is also adopted. Hence Algorithm 7 is applied to update the frequency of the population in CB-EFO. In addition to that, an *archive* that is proposed in PD-EFO has been employed in CB-EFO as well in order to store the Pareto-optimal solutions.

5.4.2. Active and Passive Electrolocation

As opposed to the other multi-objective EFO implementations, few modifications has been made on the search performed by active and passive individuals in CB-EFO, which is mostly limited to the selection scheme employed in active and passive electrolocation phases.

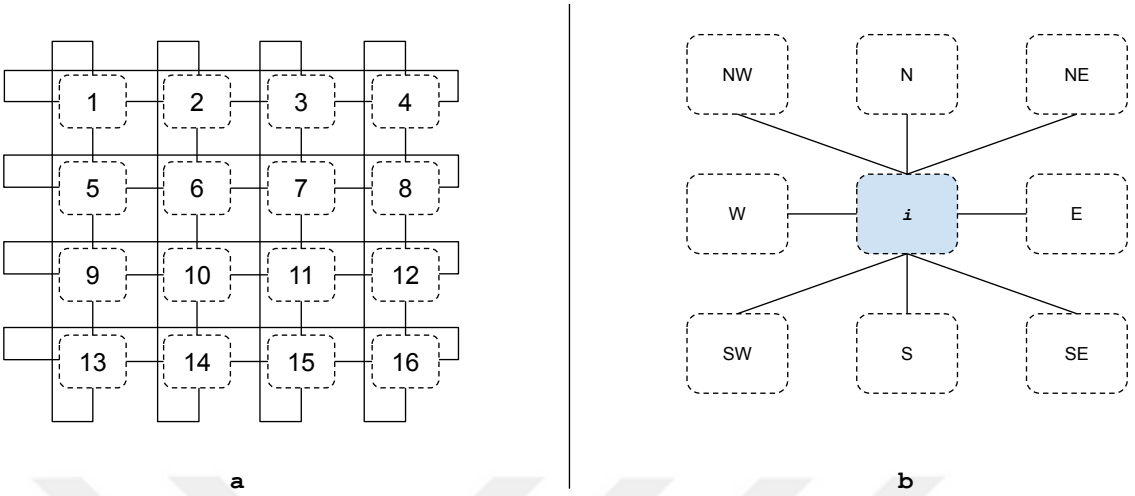

Figure 5.6. **a)** Toroidal mesh topology. **b)** Neighborhood of the ith individual. (N: North, S: South, W: West, E: East)

The main role of active mode individuals in EFO is to perform local search since they take into account only individuals in their very close neighborhood during the search. However, depending on the existence of neighbors that are evaluated by using Euclidean distance formula, individuals could perform random selection or random walk that could occasionally lead the algorithm to contribute to global and local search during the initial and final iterations, respectively. However CB-EFO does not rely on Euclidean distance between individuals, but instead it makes use of one-hop neighbors that are assigned at the initialization (see Section 5.4.1.). The proposed scheme contributes also to global search at the initial iterations, because the neighborhood is randomly established. Then, it contributes to local search as the neighbors continuously converge to the desired global minima. In order to further ensure a well-balanced search, *binary tournament selection* procedure has been used in neighbor selection because it first picks out two candidate solutions in a random manner then returns the better one among them. In CB-EFO, the crowding distance operator has been used to evaluate distances. This operator considers only the front and the distance of the solutions. It prefers the ith solution, if it belongs to a lower front than the jth solution (i.e., $front_i < front_j$). If both solutions belong to the same front, it chooses the solution with

lesser crowded region (i.e., $front_i = front_j \ \wedge \ CrowDist_i > CrowDist_j$). If both solutions belong to the same front and have the same distances, it returns a random solution (see Eq. 31). After this selection mechanism, the ith active individual applies the same formula as in EFO (see Eq. 6) in order to evolve a new candidate solution.

$$return \begin{cases} i & (front_i < front_j) \vee ((front_i = front_j) \wedge (CrowDist_i > CrowDist_j)) \\ j & (front_j < front_i) \vee ((front_j = front_i) \wedge (CrowDist_j > CrowDist_i)) \\ i \ or \ j & otherwise \end{cases} \tag{31}$$

In passive search, EFO takes into account all active individuals in the population in order to perform global search. This mechanism is also adopted in CB-EFO. However, the selection approach applied to passive individuals has been modified, since the distance-based search is excluded in CB-EFO. As in EFO, K active individuals are probabilistically chosen from the population in CB-EFO by applying K times *binary tournament selection*. The selection procedure here is based on the amplitude values that are implicitly determined by the crowding distance measurement (see Sect. 5.3.1.). The reason behind using amplitudes in the selection phase instead of the crowding distance metrics as in active search is that amplitude values of individuals do not change instantaneously. This adoption enables currently poor but formerly promising individuals being selected, which further promotes exploration capability of CB-EFO.

The other modification has been made on the conditional acceptance condition given in Eq. 11 by replacing it with Eq. 32. This modification simply states that the probability of a promising solution with higher frequency being evolved through its neighbors is lowered but that through a Pareto-optimal solution (a) from *archive* is considerably increased.

$$x_{ij}^{cand} = \begin{cases} x_{ij}^{new} & rand_j(0,1) > f_i \\ x_{ij} + \varphi \left(x_{aj} - x_{ij} \right) & else \end{cases} \tag{32}$$

Algorithm 9: Pseudocode of CB-EFO algorithm

Input: population size $|N|$, objective size $|M|$
Output: archive

```
population ← Initialize population;
Update frequency through Algorithm 7;
archive ← Update archive through Algorithm 5;
repeat
    for i = 1 to |N| do
        individual ← population[i];
        offspring ← {};
        if individual.frequency > rand then
            offspring ← active search(individual);
        else
            offspring ← passive search(individual);
        end
        if offspring ≺ individual then
            individual ← offspring;
        else if offspring ≺ individual.neighbor[worst] then
            individual.neighbor[worst] ← offspring;
        end
    end
    Update frequency through Algorithm 7;
    archive ← Update archive through Algorithm 5;
until termination criterion is met;
```

The final step in one iteration is on the replacement strategy of existing solutions with evolved candidate solutions. In EFO, *greedy selection* relying on only a single objective has been used. However it is not suitable for MOOPs. Therefore, a dominance-based selection, which prefers a non-dominated solution over a dominated solution, is used in this study. So, a candidate solution can be replaced with its parent solution only when it dominates, otherwise it can be replaced with its worst neighbor or it is discarded when it is worse than its worst neighbor from the point of domination view. The pseudocode of the proposed CB-EFO algorithm has been provided in Algorithm 9.

6. EXPERIMENTS ON MULTI-OBJECTIVE OPTIMIZATION PROBLEMS

Firstly, the performance of three multi-objective implementations of EFO algorithm (PD-EFO, NS-EFO, and CB-EFO) on MOOPs has been compared with each other. Then the best performing algorithm (CB-EFO) has been taken and reevaluated through some metrics that are intrinsic for such problem types. The following sections introduce benchmark problems, performance metrics used in the experiments, and discuss the experimental results.

6.1. Performance Metrics

The performance metric applied on MOOPs differs from SOOPs. Without the loss of generality, an algorithm is regarded as the best performing algorithm on a SOOP when its final solution is closest to the global optimum in comparison to others. However, for MOOPs, it depends on how well $\mathcal{PF}^*$ computed by non-dominated solutions of an algorithm approximates to the uniformly distributed true Pareto Front ($\mathcal{TPF}^*$) of the problem. The following metrics reveal the obvious convergence performance of an algorithm on MOOPs:

6.1.1. HV:

It measures the union of the volume between every member in $\mathcal{PF}^*$ and a reference point r which represents a vector comprising the worst values of each objective. Therefore, the higher HV implies that $\mathcal{PF}^*$ well represents $\mathcal{TPF}^*$. HV is calculated by the following equation:

$$HV = volume\left(\bigcup_{i=1}^{|\mathcal{PF}^*|} v_i\right) \tag{33}$$

6.1.2. SPREAD:

It reveals the degree of spread of $\mathcal{PF}^*$ throughout the objective space through the following equation:

$$\Delta = \frac{\sum_{i=1}^{m} d(e_i, \mathcal{PF}^*) + \sum_{x \in \mathcal{PF}^*} \left| d(x, \mathcal{PF}^*) - \hat{d} \right|}{\sum_{i=1}^{m} d(e_i, \mathcal{PF}^*) + |\mathcal{PF}^*| \cdot \hat{d}} \tag{34}$$

where e_i stands for ith extreme point in mth objective in $\mathcal{TPF}^*$. $d(e_i, \mathcal{PF}^*)$ refers the minimum distance between the extreme points in the set $\mathcal{PF}^*$ and $\mathcal{TPF}^*$, whereas $\hat{d}$ is the mean of the distances between every point in $\mathcal{PF}^*$ and in $\mathcal{TPF}^*$.

6.1.3. EPSILON:

It calculates the minimum factor (ϵ) that is necessary for at least one solution in $\mathcal{PF}^*$ to dominate all the vectors in $\mathcal{TPF}^*$. EPSILON is calculated according to the given equation:

$$E(\mathcal{PF}^*, \mathcal{TPF}^*) = \inf_{\epsilon \in \mathbb{R}^+} \{\forall \vec{p} \in \mathcal{TPF}^*, \exists \vec{v} \in \mathcal{PF}^* : \vec{v} \prec_\epsilon \vec{p}\} \tag{35}$$

here m refers to the number of objective functions whereas ϵ represents a very small value used for a tolerance.

6.1.4. IGD:

It is a good indicator that reveals how diverse the solutions in objective space is and how well the solutions can converge. The small value of IGD suggests that the solution set obtained from the objective space is very close to $\mathcal{TPF}^*$. IGD is calculated as follows:

$$IGD(\mathcal{TPF}^*, \mathcal{PF}^*) = \frac{\sum_{v \in \mathcal{TPF}^*} d(v, \mathcal{PF}^*)}{|\mathcal{TPF}^*|} \tag{36}$$

here $d(v, \mathcal{PF}^*)$ gives the minimum Euclidean distance between the solution v and every point from the set $\mathcal{PF}^*$.

6.2. Benchmark Problems

In the experiments, 30 well-known unconstrained MOOPs from DTLZ (Deb-Thiele-Laumanns-Zitzler) [12], LZ09 (Li-Zhang) [13], WFG (Walking Fish Group) [14], and ZDT (Zitzler-Deb-Thieler) [15] families are used. These problems have become the standard problems used in literature to conduct a fair comparison. They differ from each other in search space, number of parameters to optimize (6 through 30), and number of objectives (2 and 3). Table 6.1. gives the basic characteristics of these problems. The mathematical definitions of employed benchmark families, except the WFG problem family, have been given in Appendix F. The benchmark problems in WFG family are subjected to a series of transformation processes such as linear, polynomial, and etc. That's why, they cannot be expressed in a mathematical form.

Table 6.1. Multi-objective benchmark problems used in the experiments.

Family	Problem	O	V	Range
DTLZ	1	3	7	$0 \leq x_1, x_2, \cdots, x_V \leq 1$
	2-6		12	
	7		22	
LZ09	1,7-8	2	10	$0 \leq x_1, x_2, \cdots, x_V \leq 1$
	2-5,9	2	30	
	6	3	10	
WFG	1-9	2	6	$0 \leq x_k \leq (2 \times k)$, $k \in \{1, 2, \cdots, V\}$
ZDT	1-3	2	30	$0 \leq x_1, x_2, \cdots, x_V \leq 1$
	4		10	$0 \leq x_1 \leq 1$, $-5 \leq x_2, \cdots, x_V \leq 5$
	6		10	$0 \leq x_1, x_2, \cdots, x_V \leq 1$

O: Number of objectives, V: Number of variables.

6.2.1. Results

In this study, two separate experiments in which the 30 benchmark problems and the four performance metrics have been employed is conducted and analyzed. While the main goal of the first experiment is to explore the search capabilities of different implementations of EFO algorithm on complex MOOPs, the main goal of the second experiment is to reveal

the comparative search performance of CB-EFO, the best-performing EFO implementation, against the well-known evolutionary-based multi-objective optimization algorithms. These algorithms are NSGA-II [16], SPEA2 [17], IBEA [18], MOEAD [19], and MOCell [20].

jMetal (Metaheuristic Algorithms in Java) [180, 181], a Java-based framework for multi-objective optimization with metaheuristics, is employed in the experiments. jMetal is a framework that provides some facilities such as state-of-the algorithms, most-popular benchmark problems, and the like. Refer to [180] for a detailed description of the framework. In both experiments, every algorithm is run 50 times with a population and a generation size of 100 and 1,000 (corresponding to 100,000 function evaluations), respectively. The archive size is set same as the population size. α and K values of CB-EFO are empirically found and set to 0.9 and 20 respectively. The parameters of the competitor algorithms are set to the default parameters in jMetal. The CB-EFO implementation in jMetal framework that is ready-to-replicate the same experimentation is shared with the community [31].

The comparative convergence and diversification performance of the multi-objective EFO algorithms and the well-known multi-objective optimization algorithms on the benchmark problems have been provided in Section 6.2.1.1. and 6.2.1.2.. For both experiments, the detailed statistical analysis of 50 runs for each problem is also given in Appendix G.

6.2.1.1. Comparison of Multi-objective EFO Algorithms

Figure 6.1. presents the number of problems where PD-EFO, NS-EFO, and CB-EFO have performed satisfying performance (i.e. becoming the first or the second algorithm in ranking). The plot clearly shows that CB-EFO outperforms other algorithms by performing the best/2nd best performance on problems with respect to HV, SPREAD, and EPSILON metrics. The results of CB-EFO (the number of problems solved as the best algorithm, the number of problems solved as the second best algorithm) on HV, SPREAD, EPSILON, and IGD metrics respectively are (14/9), (20/6), (15/9), and (15/5). The results reveal also that NS-EFO has become the second best performing algorithm after CB-EFO and showing (7/20), (3/23), (7/21), and (7/22) ranking performance with respect to the HV, SPREAD, EPSILON,

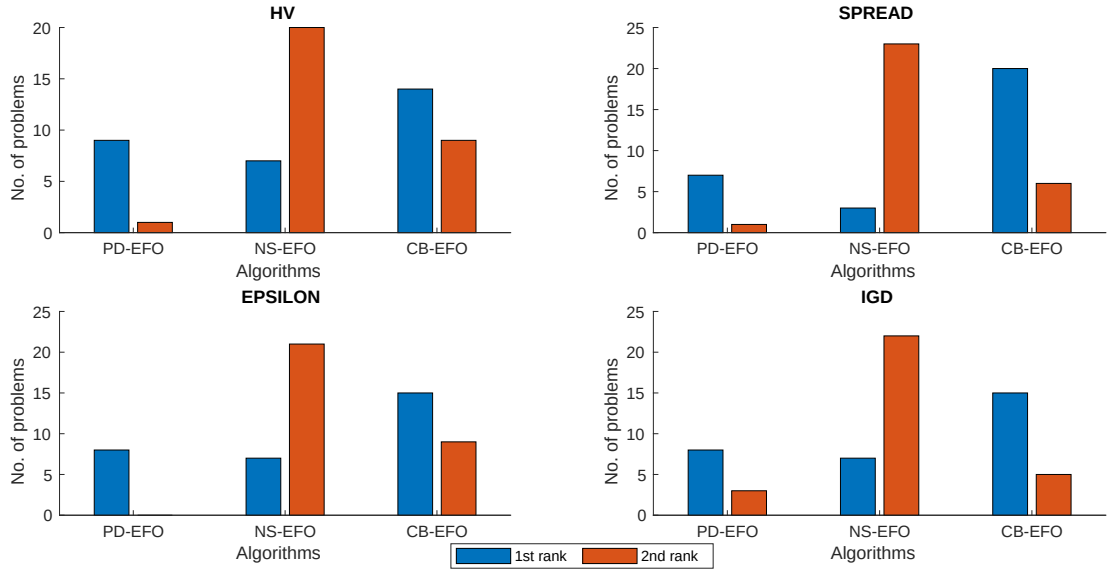

Figure 6.1. Ranking of the multi-objective EFO algorithms.

and IGD metrics, respectively. PD-EFO algorithm, however, has shown a poor optimization performance with the rankings of (9/1), (7/1), (8/0), and (8/3) with respect to the HV, SPREAD, EPSILON, and IGD metrics, respectively. It often shows better performance on problems in the LZ09 family, which shows that its performance depends on the problem under test. Table G.1. through G.4. gives the detailed statistical mean and deviation results of PD-EFO, NS-EFO, and CB-EFO with respect to the performance metrics.

Friedman's ranking test is applied to further examine the overall performance of every algorithm. The results have been provided in Table 6.2.. It is worth stressing here that the algorithm with higher mean HV values performs better than others. However, for other metrics, the performance of the algorithm increases as the metric becomes smaller. The best and 2nd best performing algorithms have been implied with dark and light gray cell colors, respectively.

It can be concluded from the table that CB-EFO has outperformed its competitors on HV, SPREAD, and EPSILON metrics. Only when the IGD metric is taken into consideration, CB-EFO becomes the second best algorithm and comes after NS-EFO.

Table 6.2. Friedman's ranking test results for the multi-objective EFO algorithms.

Algorithms	Ranking			
	HV	**SPREAD**	**EPSILON**	**IGD**
PD-EFO	1.633	2.500	2.467	2.367
NS-EFO	2.133	2.033	1.833	1.800
CB-EFO	2.233	1.467	1.700	1.833

6.2.1.2. Comparison of CB-EFO with Other Algorithms in the Literature

In this section, the performance results of CB-EFO is compared with the well-known multi-objective optimization algorithms in the literature: NSGA-II, SPEA2, IBEA, MOEAD, and MOCell. Their ranks according to each metric is demonstrated in Figure 6.2..

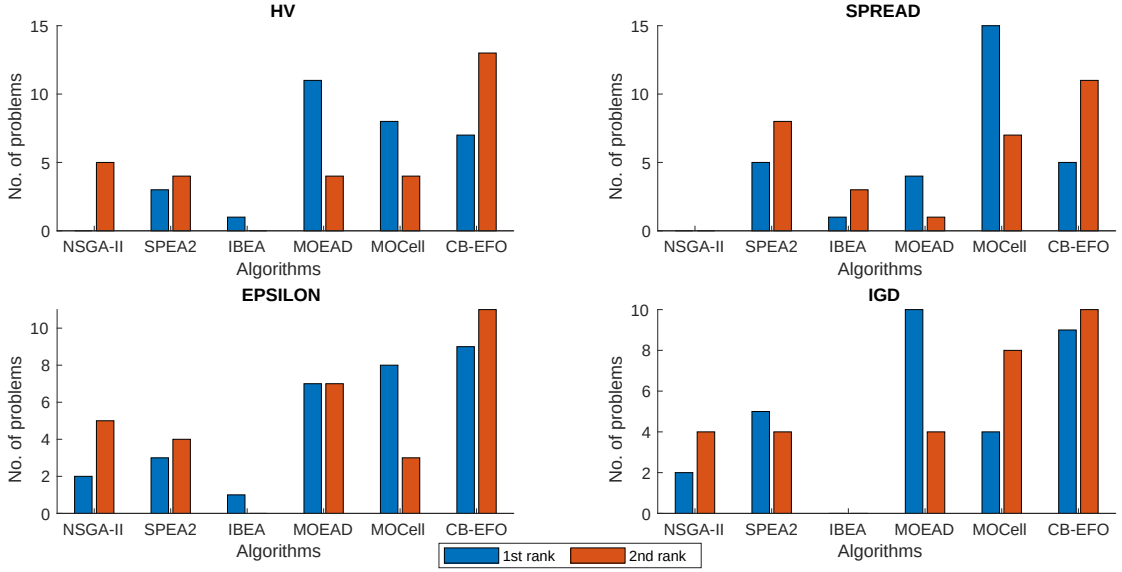

Figure 6.2. Comparison of the multi-objective algorithms on each metric.

As it is seen clearly in the figure, CB-EFO is superior than other algorithms by performing the best/2nd best performance on the problems with respect to HV, EPSILON, and IGD metrics. The results (the number of problems solved as the best algorithm, the number of problems solved as the second best algorithm) on HV, EPSILON, and IGD metrics respectively are (7/13), (9/11), and (9/10). MOCell is the most competitive algorithm against CB-EFO and it has shown the best performance on 15 problems and, the second best performance on 7 problems with respect to SPREAD metrics. It outperforms CB-EFO on the SPREAD metric

in evaluations. MOEAD is another competitive algorithm; however, like PD-EFO, it often shows better performance on problems in the LZ09 family.

As done in the previous experiment, Friedman's ranking test is also applied here. The results have been provided in Table 6.3.. It can be concluded from the table that CB-EFO has outperformed its competitors on HV, EPSILON, and IGD metrics. Only on evaluations by using the SPREAD metric, it becomes the second best algorithm and comes after MOCell.

Table 6.3. Friedman's ranking values for multi-objective algorithms on each metric.

Algorithms	**Ranking**			
	HV	**SPREAD**	**EPSILON**	**IGD**
NSGA-II	2.933	4.467	3.800	3.667
SPEA2	3.833	2.733	3.367	2.833
IBEA	2.233	5.167	4.867	5.500
MOEAD	4.033	4.000	2.867	2.833
MOCell	3.567	2.000	3.633	3.567
CB-EFO	4.400	2.633	2.467	2.600

The true Pareto front of problems and the Pareto front of algorithms have been given in Figure G.1. through G.6. to reveal comparative convergence behavior of every approach on objective space.

7. CONCLUSION

In this thesis study, active and passive electrolocation and electrocommunication capabilities of electric fish have been explored for proposing a nature-inspired heuristic algorithm. While active electrolocation represents a behavior where electric fish generate EOD and sense their surroundings through the modulations in the electric field, passive electrolocation represents a behavior of electric fish where they perceive external signal from their electroreceptors. The emanated EOD is characterized by frequency and amplitude. The EOD frequency changes with respect to the closeness of an electric fish to the prey and it increases as electric fish gets closer to the prey, and vice versa. The EOD amplitude, however, is proportional to the fish's size. These capabilities and characteristics of electric fish have been used for the creation of a novel swarm-based heuristic algorithm called EFO in order to solve bound-constrained real parameter SOOPs. To the best knowledge of the authors, this is the first study to propose a heuristic inspired by the electrolocation and electrocommunication capabilities as well as the EOD frequency and amplitude characteristics of electric fish.

The proposed EFO is quite simple, with its easy-to-understand structure, few parameters, and two main search frameworks (active and passive search phases). Active search phase contributes to the local search in EFO, because individuals in active mode are regarded to be in the vicinity of the best food sources, which is the case in nature. Passive search phase, however, contributes to the global search in EFO, because individuals in passive mode are regarded as not-so-good individuals and they are far away from finding promising solutions. EOD frequency is employed in the algorithm to determine the search role of individuals (active or passive) as it is a good indicator to find out which individuals are in the vicinity of the promising food sources in nature. EOD amplitude determines the search range in local search and communication probability in global search.

It is believed that the optimization and convergence performance of EFO is largely due to the modeled active and passive electrolocation capabilities, which fulfill the "explore first, exploit later" approach that is strongly suggested for heuristics in the literature [9, 10, 142].

Individuals in EFO are more prone to perform global search in the premier iterations and local search as the iterations lapse, owing to the distance among the population that gradually decreases through the iterations. In active electrolocation, individuals in active mode perform a random search inside their active search range at the beginning iterations, as it is less likely to belong to another's active area, and start to search around their neighborhood toward the end of the iterations. In passive electrolocation, however, the selection of neighbor individuals is highly dependent on the distance between them (see Eq. 8). At the initial phase of the iterations, individuals in passive mode select their neighbors considering the distance rather than their amplitudes. The distance-based selection results in a better global search owing to the higher selection probability of the "poor" individuals. As the iteration proceeds, the amplitude becomes the dominant factor that determines the individuals being selected, which contributes to the local search of EFO.

In order to measure the performance of single-objective EFO, it is compared with well-known single-solution-based (SA and VS) and population-based (GA, DE, PSO, and ABC) heuristics as well as approaches that were recently proposed for constrained and unconstrained problems in the experiments with different characteristics. From the overall optimization performance of EFO, it performs much better than almost all of the other heuristics, with a highly compelling convergence capability. The optimization performance measured on different problem set domains (i.e., bound-constrained mathematical benchmark problems, unconstrained clustering and classification problems, and bound- and design-constrained real-world design problems) has proven the robustness of the proposed single-objective EFO algorithm. In addition, it can be understood from the results that EFO has better global search capability and it outperforms almost all the competitor algorithms on highly complex problems such as *Schwefel*, *Rastrigin*, and *Griewank* problems. It makes EFO a very promising approach, given the complexity of most real-world problems.

In addition to that, the single-objective EFO algorithm has been extended in order to solve unconstrained MOOPs. In order to adapt EFO for solving MOOPs, the single-objective EFO algorithm has been subjected to a number of modifications while mostly maintaining the search characteristics of the base algorithm. Hence; PD-EFO, NS-EFO, and CB-EFO

algorithms have been introduced. Among them, PD-EFO and NS-EFO do not change the search operators proposed by the single-objective EFO algorithm while a moderate change is made by CB-EFO to enhance the search performance. To adapt the single-objective EFO algorithm to the MOOPs, the Pareto dominance concept has been introduced in PD-EFO; while *Pareto front* and *crowding distance* proposed by NSGA-II have been introduced in NS-EFO and CB-EFO. In addition, external archive has been used in PD-EFO and CB-EFO to store non-dominated solutions throughout the search.

The proposed multi-objective EFO algorithms have been evaluated on 30 well-known bound-constrained MOOPs, in which the number of objectives is either 2 or 3, and the number of parameters to be optimized ranges from 6 to 30. The problem families employed in this thesis are DTLZ [12], LZ09 [13], WFG [14], and ZDT [15]. To reveal the convergence and diversification performance of Pareto-optimal solutions found by PD-EFO, NS-EFO, and CB-EFO; four well-known metrics that are HV, SPREAD, EPSILON, and IGD have been considered. The empirical results show that CB-EFO is overwhelmingly better than PD-EFO and NS-EFO. Therefore, it has been reevaluated through a comparison with the well-known multi-objective approaches to determine its success in the literature. From the comparative results, CB-EFO ensures a much better optimization performance on such problems in comparison to these well-known approaches, namely NSGA-II [16], SPEA2 [17], IBEA [18], MOEAD [19], MOCell [20]. Therefore, in this study, CB-EFO has been introduced as an alternative to the most popular multi-objective optimization algorithms.

There are few limitations with the proposed algorithm, which can be considered as the potential areas for future work. They have been outlined as follows:

- While the distance-based selection leads EFO to have superior performance on multi-modal and complex problems, the complexity of the algorithm is also mainly governed by this distance calculation of individuals.
- Two parameters (the magnitude for old amplitude α, the maximum number of neighbor individuals K) of the EFO algorithm are need to be set, whereas some algorithms do

not even have a single parameter such as VS. Fortunately, the experiments show that EFO is less sensitive to the difference in parameter settings.

- EFO may require some additional efforts to further enhance its search capability for better performance, particularly on the unimodal, hybrid, and composed functions in the first experiment where the single-objective EFO is evaluated.

To conclude, in this thesis, a new swarm-based algorithm called EFO, which is shown to be effective for both SOOPs and MOOPs, is introduced to the community. We believe that the active and passive electrolocation capabilities of electric fish fulfill the “explore first, exploit later" approach that is strongly suggested for heuristics. We encourage the community to further improve the algorithm and to explore the use of EFO on different application areas.

REFERENCES

[1] Xin-She Yang. *Nature-Inspired Metaheuristic Algorithms: Second Edition*. Luniver Press, **2010**. ISBN 1905986289, 9781905986286.

[2] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, **1997**.

[3] Hossein Nezamabadi-pour, Saeid Saryazdi, and Esmat Rashedi. Edge detection using ant algorithms. *Soft Computing*, 10(7):623–628, **2006**. ISSN 1433-7479. doi:10.1007/s00500-005-0511-y.

[4] I. De Falco, A. Della Cioppa, D. Maisto, and E. Tarantino. Differential evolution as a viable tool for satellite image registration. *Applied Soft Computing*, 8(4):1453 – 1462, **2008**. ISSN 1568-4946. doi:https://doi.org/10.1016/j.asoc.2007.10.013. Soft Computing for Dynamic Data Mining.

[5] Tiago Sousa, Arlindo Silva, and Ana Neves. Particle swarm based data mining algorithms for classification tasks. *Parallel Computing*, 30(5–6):767 – 783, **2004**. ISSN 0167-8191. doi:https://doi.org/10.1016/j.parco.2003.12.015. Parallel and nature-inspired computational paradigms and applications.

[6] Gianni Di Caro, Frederick Ducatelle, and Luca Maria Gambardella. Anthocnet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *European Transactions on Telecommunications*, 16(5):443–455, **2005**. ISSN 1541-8251. doi:10.1002/ett.1062.

[7] Fang Sun and Guangrui Hu. Speech recognition based on genetic algorithm for training HMM. *Electronics Letters*, 34(16):1563, **1998**. doi:10.1049/el:19980096.

[8] X.S. Yang, A.H. Gandomi, S. Talatahari, and A.H. Alavi. *Metaheuristics in Water, Geotechnical and Transport Engineering*. Elsevier insights. Elsevier Science, **2012**. ISBN 9780123983176.

[9] Esmat Rashedi, Hossein Nezamabadi-pour, and Saeid Saryazdi. GSA: A gravitational search algorithm. *Information Sciences*, 179(13):2232–2248, **2009**. doi:10.1016/j.ins.2009.03.004.

[10] K.C. Tan, S.C. Chiam, A.A. Mamun, and C.K. Goh. Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization. *European Journal of Operational Research*, 197(2):701–713, **2009**. doi:10.1016/j.ejor.2008.07.025.

[11] NH Awad, MZ Ali, JJ Liang, BY Qu, and PN Suganthan. Problem definitions and evaluation criteria for the cec 2017 special session and competition on single objective real-parameter numerical optimization. *Tech. Rep.*, **2016**.

[12] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable test problems for evolutionary multiobjective optimization. In *Evolutionary multiobjective optimization*, pages 105–145. Springer, **2005**.

[13] H. Li and Q. Zhang. Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. *IEEE Transactions on Evolutionary Computation*, 13(2):284–302, **2009**.

[14] Simon Huband, Philip Hingston, Luigi Barone, and Lyndon While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, **2006**.

[15] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195, **2000**.

[16] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, **2002**.

[17] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. *TIK-report*, 103, **2001**.

[18] Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. In Xin Yao, Edmund K. Burke, José A. Lozano, Jim Smith, Juan Julián Merelo-Guervós, John A. Bullinaria, Jonathan E. Rowe, Peter Tiňo, Ata Kabán, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VIII*, pages 832–842. Springer Berlin Heidelberg, Berlin, Heidelberg, **2004**. ISBN 978-3-540-30217-9.

[19] Q. Zhang and H. Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, **2007**. ISSN 1941-0026. doi:10.1109/TEVC.2007.892759.

[20] Antonio J Nebro, Juan J Durillo, Francisco Luna, Bernabé Dorronsoro, and Enrique Alba. Design issues in a multiobjective cellular genetic algorithm. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 126–140. Springer, **2007**.

[21] N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, **1994**. ISSN 1063-6560. doi:10.1162/evco.1994.2.3.221.

[22] L. Darrell Whitley. Cellular genetic algorithms. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 658–. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, **1993**. ISBN 1-55860-299-2.

[23] H. Mühlenbein, M. Schomisch, and J. Born. The parallel genetic algorithm as function optimizer. *Parallel Computing*, 17(6):619 – 632, **1991**. ISSN 0167-8191. doi:https://doi.org/10.1016/S0167-8191(05)80052-3.

[24] Vivekananda Haldar and Niladri Chakraborty. A novel evolutionary technique based on electrolocation principle of elephant nose fish and shark: fish electrolocation optimization. *Soft Computing*, pages 1–22, **2016**. ISSN 1433-7479. doi:10.1007/s00500-016-2033-1.

[25] V Lebastard, C Chevallereau, A Amrouche, B Jawad, A Girin, F Boyer, and P B Gossiaux. Underwater robot navigation around a sphere using electrolocation sense and kalman filter. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, **2010**. doi:10.1109/iros.2010.5648929.

[26] M.A. MacIver, E. Fontaine, and J.W. Burdick. Designing future underwater vehicles: Principles and mechanisms of the weakly electric fish. *IEEE Journal of Oceanic Engineering*, 29(3):651–659, **2004**. doi:10.1109/joe.2004.833210.

[27] Malcolm A. Maciver and Mark E. Nelson. Towards a biorobotic electrosensory systemtowards a biorobotic electrosensory system. *Autonomous Robots*, 11(3):263–266, **2001**. doi:10.1023/a:1012443124333.

[28] J. R. Solberg, K. M. Lynch, and M. A. MacIver. Robotic electrolocation: Active underwater target localization with electric fields. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4879–4886. **2007**. ISSN 1050-4729. doi:10.1109/ROBOT.2007.364231.

[29] I. D. Neveln, Y. Bai, J. B. Snyder, J. R. Solberg, O. M. Curet, K. M. Lynch, and M. A. MacIver. Biomimetic and bio-inspired robotics in electric fish research. *Journal of Experimental Biology*, 216(13):2501–2514, **2013**. doi:10.1242/jeb.082743.

[30] Abdolreza Hatamlou. Black hole: A new heuristic optimization approach for data clustering. *Information Sciences*, 222(Supplement C):175 – 184, **2013**. ISSN 0020-0255. doi:https://doi.org/10.1016/j.ins.2012.08.023. Including Special Section on New Trends in Ambient Intelligence and Bio-inspired Systems.

[31] WISE LAB. Efo source code and experimental results. https://wise.cs.hacettepe.edu.tr/, **2020**. [Online; accessed 20-Jan-2020].

[32] Ilhem Boussaïd, Julien Lepagnot, and Patrick Siarry. A survey on optimization metaheuristics. *Information Sciences*, 237:82 – 117, **2013**. ISSN 0020-0255.

doi:https://doi.org/10.1016/j.ins.2013.02.041. Prediction, Control and Diagnosis using Advanced Neural Computations.

[33] El-Ghazali Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, **2009**.

[34] LA Rastrigin. Convergence of random search method in extremal control of many-parameter system. *Automation and Remote Control*, 24(11):1337, **1964**.

[35] Robert Hooke and T. A. Jeeves. Direct search solution of numerical and statistical problems. *Journal of the ACM*, 8(2):212–229, **1961**. doi:10.1145/321062.321069.

[36] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549, **1986**. doi:10.1016/0305-0548(86)90048-1.

[37] Helena R. Lourenço, Olivier C. Martin, and Thomas Stützle. *Iterated Local Search: Framework and Applications*, pages 363–397. Springer US, Boston, MA, **2010**. ISBN 978-1-4419-1665-5. doi:10.1007/978-1-4419-1665-5_12.

[38] Pierre Hansen, Nenad Mladenović, and José A. Moreno Pérez. Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175(1):367–407, **2010**. ISSN 1572-9338. doi:10.1007/s10479-009-0657-6.

[39] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, **1983**. ISSN 0036-8075. doi:10.1126/science.220.4598.671.

[40] Berat Dogan and Tamer Ölmez. A new metaheuristic for numerical function optimization: Vortex search algorithm. *Information Sciences*, 293:125–145, **2015**. doi:10.1016/j.ins.2014.08.053.

[41] Larry C Andrews and Larry C Andrews. *Special functions of mathematics for engineers*. McGraw-Hill New York, **1992**.

[42] John H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, MA, USA, **1992**. ISBN 0-262-58111-6.

[43] John R. Koza. Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems, **1990**.

[44] H.P. Schwefel. *Kybernetische evolution als strategie der experimentellen forschung in der stromungstechnik*. Master's thesis, Technical University of Berlin, Germany, **1965**.

[45] David Corne, Marco Dorigo, Fred Glover, Dipankar Dasgupta, Pablo Moscato, Riccardo Poli, and Kenneth V. Price, editors. *New Ideas in Optimization*. McGraw-Hill Ltd., UK, Maidenhead, UK, England, **1999**. ISBN 0-07-709506-5.

[46] Lawrence J. Fogel. *Intelligence Through Simulated Evolution: Forty Years of Evolutionary Programming*. John Wiley & Sons, Inc., New York, NY, USA, **1999**. ISBN 0-471-33250-X.

[47] Pinar Civicioglu. Backtracking search optimization algorithm for numerical optimization problems. *Applied Mathematics and Computation*, 219(15):8121 – 8144, **2013**. ISSN 0096-3003. doi:https://doi.org/10.1016/j.amc.2013.02.017.

[48] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39, **2006**. doi:10.1109/mci.2006.329691.

[49] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4. **1995**.

[50] Dervis Karaboga. An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, **2005**.

[51] Dervis Karaboga and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of Global Optimization*, 39(3):459–471, **2007**. ISSN 1573-2916. doi:10.1007/s10898-007-9149-x.

[52] Xin-She Yang. Firefly algorithms for multimodal optimization. In Osamu Watanabe and Thomas Zeugmann, editors, *Stochastic Algorithms: Foundations and Applications*, pages 169–178. Springer Berlin Heidelberg, Berlin, Heidelberg, **2009**.

[53] X. Yang and Suash Deb. Cuckoo search via lévy flights. In *2009 World Congress on Nature Biologically Inspired Computing (NaBIC)*, pages 210–214. **2009**. doi: 10.1109/NABIC.2009.5393690.

[54] Xin-She Yang. *A New Metaheuristic Bat-Inspired Algorithm*, pages 65–74. Springer Berlin Heidelberg, Berlin, Heidelberg, **2010**. ISBN 978-3-642-12538-6. doi:10.1007/978-3-642-12538-6_6.

[55] A. Kaveh and N. Farhoudi. A new optimization method: Dolphin echolocation. *Advances in Engineering Software*, 59:53 – 70, **2013**. ISSN 0965-9978. doi: https://doi.org/10.1016/j.advengsoft.2013.03.004.

[56] A Kaveh and N Farhoudi. Dolphin echolocation optimization: continuous search space. *Advances in Computational Design*, 1(2):175–194, **2016**.

[57] A.R. Mehrabian and C. Lucas. A novel numerical optimization algorithm inspired from weed colonization. *Ecological Informatics*, 1(4):355 – 366, **2006**. ISSN 1574-9541. doi:https://doi.org/10.1016/j.ecoinf.2006.07.003.

[58] Xin-She Yang. Flower pollination algorithm for global optimization. In Jérôme Durand-Lose and Nataša Jonoska, editors, *Unconventional Computation and Natural Computation*, pages 240–249. Springer Berlin Heidelberg, Berlin, Heidelberg, **2012**. ISBN 978-3-642-32894-7.

[59] Sait Ali Uymaz, Gulay Tezel, and Esra Yel. Artificial algae algorithm (aaa) for nonlinear global optimization. *Applied Soft Computing*, 31:153 – 171, **2015**. ISSN 1568-4946. doi:https://doi.org/10.1016/j.asoc.2015.03.003.

[60] Pinar Civicioglu. Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm. *Computers & Geosciences*, 46:229 – 247, **2012**. ISSN 0098-3004. doi:https://doi.org/10.1016/j.cageo.2011.12.011.

[61] Seyedali Mirjalili. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications*, 27(4):1053–1073, **2016**.

[62] Hussain Shareef, Ahmad Asrul Ibrahim, and Ammar Hussein Mutlag. Lightning search algorithm. *Applied Soft Computing*, 36:315 – 333, **2015**. ISSN 1568-4946. doi:https://doi.org/10.1016/j.asoc.2015.07.028.

[63] Amir Hossein Gandomi and Amir Hossein Alavi. Krill herd: A new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation*, 17(12):4831 – 4845, **2012**. ISSN 1007-5704. doi:https://doi.org/10.1016/j.cnsns.2012.05.010.

[64] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. Grey wolf optimizer. *Advances in Engineering Software*, 69:46 – 61, **2014**. ISSN 0965-9978. doi:https://doi.org/10.1016/j.advengsoft.2013.12.007.

[65] Jagdish Chand Bansal, Harish Sharma, Shimpi Singh Jadon, and Maurice Clerc. Spider monkey optimization algorithm for numerical optimization. *Memetic Computing*, 6(1):31–47, **2014**. doi:10.1007/s12293-013-0128-0.

[66] Seyedali Mirjalili and Andrew Lewis. The whale optimization algorithm. *Advances in Engineering Software*, 95:51 – 67, **2016**. ISSN 0965-9978. doi:https://doi.org/10.1016/j.advengsoft.2016.01.008.

[67] Amir H. Gandomi. Interior search algorithm (isa): A novel approach for global optimization. *ISA Transactions*, 53(4):1168 – 1183, **2014**. ISSN 0019-0578. doi: https://doi.org/10.1016/j.isatra.2014.03.018. Disturbance Estimation and Mitigation.

[68] Victoria Aragon, Esquivel Susana C., and Coello Carlos A. Coello. A modified version of a t cell algorithm for constrained optimization problems. *International Journal for Numerical Methods in Engineering*, 84(3):351–378, **2010**. doi:10.1002/nme.2904.

[69] Alireza Askarzadeh. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. *Computers & Structures*, 169:1 – 12, **2016**. ISSN 0045-7949. doi:https://doi.org/10.1016/j.compstruc.2016.03.001.

[70] K. M. Passino. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, 22(3):52–67, **2002**. ISSN 1066-033X. doi:10.1109/MCS.2002.1004010.

[71] Efren Mezura-Montes and Betania Hernandez-Ocana. Bacterial foraging for engineering design problems: preliminary results. In *Memorias del 4o Congreso Nacional de Computacion Evolutiva (COMCEV 2008)*. **2008**.

[72] R.V. Rao, V.J. Savsani, and D.P. Vakharia. Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, 43(3):303 – 315, **2011**. ISSN 0010-4485. doi: https://doi.org/10.1016/j.cad.2010.12.015.

[73] Jie Han, Chunhua Yang, Xiaojun Zhou, and Weihua Gui. A two-stage state transition algorithm for constrained engineering optimization problems. *International Journal of Control, Automation and Systems*, 16(2):522–534, **2018**. ISSN 2005-4092. doi:10.1007/s12555-016-0338-6.

[74] Hui Liu, Zixing Cai, and Yong Wang. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Applied Soft Computing*, 10(2):629 – 640, **2010**. ISSN 1568-4946. doi:https://doi.org/10.1016/j.asoc.2009.08.031.

[75] Leandro dos Santos Coelho. Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. *Expert Systems with Applications*, 37(2):1676 – 1683, **2010**. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2009.06.044.

[76] Dilip Datta and José Rui Figueira. A real-integer-discrete-coded particle swarm optimization for design problems. *Applied Soft Computing*, 11(4):3625 – 3633, **2011**. ISSN 1568-4946. doi:https://doi.org/10.1016/j.asoc.2011.01.034.

[77] Efrén Mezura-Montes, CA Coello Coello, and Jesús Velázquez-Reyes. Increasing successful offspring and diversity in differential evolution for engineering design. In *Proceedings of the seventh international conference on adaptive computing in design and manufacture (ACDM 2006)*, pages 131–139. **2006**.

[78] Ali Wagdy Mohamed. A novel differential evolution algorithm for solving constrained engineering optimization problems. *Journal of Intelligent Manufacturing*, 29(3):659–692, **2018**. ISSN 1572-8145. doi:10.1007/s10845-017-1294-6.

[79] Min Zhang, Wenjian Luo, and Xufa Wang. Differential evolution with dynamic stochastic selection for constrained optimization. *Information Sciences*, 178(15):3043 – 3074, **2008**. ISSN 0020-0255. doi:https://doi.org/10.1016/j.ins.2008.02.014. Nature Inspired Problem-Solving.

[80] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, 10(6):646–657, **2006**. ISSN 1941-0026. doi:10.1109/TEVC.2006.872133.

[81] J. Zhang and A. C. Sanderson. Jade: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958, **2009**. ISSN 1941-0026. doi:10.1109/TEVC.2009.2014613.

[82] A. K. Qin and P. N. Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. In *2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1785–1791 Vol. 2. **2005**. ISSN 1941-0026. doi:10.1109/CEC.2005.1554904.

[83] Efrén Mezura-Montes, CA Coello Coello, and Ricardo Landa-Becerra. Engineering optimization using simple evolutionary algorithm. In *Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence*, pages 149–156. IEEE, **2003**.

[84] Junling Zhang, Changyong Liang, Yongqing Huang, Jian Wu, and Shanlin Yang. An effective multiagent evolutionary algorithm integrating a novel roulette inversion operator for engineering optimization. *Applied Mathematics and Computation*, 211(2):392 – 416, **2009**. ISSN 0096-3003. doi:https://doi.org/10.1016/j.amc.2009.01.048.

[85] H. S. Bernardino, H. J. C. Barbosa, and A. C. C. Lemonge. A hybrid genetic algorithm for constrained optimization problems in mechanical engineering. In *2007 IEEE Congress on Evolutionary Computation*, pages 646–653. **2007**. ISSN 1089-778X. doi:10.1109/CEC.2007.4424532.

[86] H. S. Bernardino, H. J. C. Barbosa, A. C. C. Lemonge, and L. G. Fonseca. A new hybrid ais-ga for constrained optimization problems in mechanical engineering. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 1455–1462. **2008**. ISSN 1089-778X. doi:10.1109/CEC.2008.4630985.

[87] G. Kanagaraj, S.G. Ponnambalam, N. Jawahar, and J. Mukund Nilakantan. An effective hybrid cuckoo search and genetic algorithm for constrained engineering design optimization. *Engineering Optimization*, 46(10):1331–1351, **2014**. doi:10.1080/0305215X.2013.836640.

[88] Wen-Tsao Pan. A new fruit fly optimization algorithm: Taking the financial distress model as an example. *Knowledge-Based Systems*, 26:69 – 74, **2012**. ISSN 0950-7051. doi:https://doi.org/10.1016/j.knosys.2011.07.001.

[89] Ting-Song Du, Xian-Ting Ke, Jia-Gen Liao, and Yan-Jun Shen. Dslc-foa : Improved fruit fly optimization algorithm for application to structural engineering design optimization problems. *Applied Mathematical Modelling*, 55:314 – 339, **2018**. ISSN 0307-904X. doi:https://doi.org/10.1016/j.apm.2017.08.013.

[90] Francisco Erivaldo Fernandes Junior and Gary G. Yen. Particle swarm optimization of deep neural networks architectures for image classification. *Swarm and Evolutionary Computation*, 49:62 – 74, **2019**. ISSN 2210-6502. doi: https://doi.org/10.1016/j.swevo.2019.05.010.

[91] Hyejung Chung and Kyung-shik Shin. Genetic algorithm-optimized multi-channel convolutional neural network for stock market prediction. *Neural Computing and Applications*, pages 1–18.

[92] James David Schaffer. *Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms (Artificial Intelligence, Optimization, Adaptation, Pattern Recognition)*. Ph.D. thesis, Vanderbilt University, Nashville, TN, USA, **1984**. AAI8522492.

[93] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, **1989**. ISBN 0201157675.

[94] Eckart Zitzler. Evolutionary algorithms for multiobjective optimization: Methods and applications, **1999**.

[95] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4. **1995**.

[96] Dervis Karaboga and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of Global Optimization*, 39(3):459–471, **2007**. ISSN 1573-2916. doi:10.1007/s10898-007-9149-x.

[97] C. A. Coello Coello and M. S. Lechuga. Mopso: a proposal for multiple objective particle swarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, volume 2, pages 1051–1056 vol.2. **2002**. ISSN null. doi:10.1109/CEC.2002.1004388.

[98] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga. Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279, **2004**. ISSN 1941-0026. doi:10.1109/TEVC.2004.826067.

[99] Margarita Reyes Sierra and Carlos A. Coello Coello. Improving pso-based multi-objective optimization using crowding, mutation and ϵ-dominance. In Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, editors, *Evolutionary Multi-Criterion Optimization*, pages 505–519. Springer Berlin Heidelberg, Berlin, Heidelberg, **2005**.

[100] Praveen Kumar Tripathi, Sanghamitra Bandyopadhyay, and Sankar Kumar Pal. Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients. *Information Sciences*, 177(22):5033 – 5049, **2007**. ISSN 0020-0255. doi:https://doi.org/10.1016/j.ins.2007.06.018.

[101] Antonio J Nebro, Juan José Durillo, Jose Garcia-Nieto, CA Coello Coello, Francisco Luna, and Enrique Alba. Smpso: A new pso-based metaheuristic for multi-objective optimization. In *2009 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM)*, pages 66–73. IEEE, **2009**.

[102] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, **2002**. ISSN 1941-0026. doi:10.1109/4235.985692.

[103] B. Zhao, C. X. Guo, and Y. J. Cao. A multiagent-based particle swarm optimization approach for optimal reactive power dispatch. *IEEE Transactions on Power Systems*, 20(2):1070–1078, **2005**. ISSN 1558-0679. doi: 10.1109/TPWRS.2005.846064.

[104] Cai Dai, Yuping Wang, and Miao Ye. A new multi-objective particle swarm optimization algorithm based on decomposition. *Information Sciences*, 325:541 – 557, **2015**. ISSN 0020-0255. doi:https://doi.org/10.1016/j.ins.2015.07.018.

[105] K. E. Parsopoulos, D. K. Tasoulis, M. N. Vrahatis, and Key Words. Multiobjective optimization using parallel vector evaluated particle swarm optimization. In *In Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2004*, pages 823–828. ACTA Press, **2004**.

[106] Xiaohui Hu and R. Eberhart. Multiobjective optimization using dynamic neighborhood particle swarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, volume 2, pages 1677–1681 vol.2. **2002**. ISSN null. doi:10.1109/CEC.2002.1004494.

[107] Xiaohui Hu, R. C. Eberhart, and Yuhui Shi. Particle swarm with extended memory for multiobjective optimization. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No.03EX706)*, pages 193–197. **2003**. ISSN null. doi:10.1109/SIS.2003.1202267.

[108] S. Mostaghim and J. Teich. Strategies for finding good local guides in multi-objective particle swarm optimization (mopso). In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No.03EX706)*, pages 26–33. **2003**. ISSN null. doi:10.1109/SIS.2003.1202243.

[109] S.N. Omkar, J. Senthilnath, Rahul Khandelwal, G. Narayana Naik, and S. Gopalakrishnan. Artificial bee colony (abc) for multi-objective design optimization of composite structures. *Applied Soft Computing*, 11(1):489 – 499, **2011**. ISSN 1568-4946. doi:https://doi.org/10.1016/j.asoc.2009.12.008.

[110] Reza Akbari, Ramin Hedayatzadeh, Koorush Ziarati, and Bahareh Hassanizadeh. A multi-objective artificial bee colony algorithm. *Swarm and Evolutionary Computation*, 2:39 – 52, **2012**. ISSN 2210-6502. doi:https://doi.org/10.1016/j.swevo.2011.08.001.

[111] R. Hedayatzadeh, B. Hasanizadeh, R. Akbari, and K. Ziarati. A multi-objective artificial bee colony for optimizing multi-objective problems. In *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, volume 5, pages V5–277–V5–281. **2010**. ISSN 2154-7491. doi:10.1109/ICACTE.2010.5579761.

[112] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation*, 10(3):263–282, **2002**. ISSN 1063-6560. doi:10.1162/106365602760234108.

[113] Bahriye Akay. Synchronous and asynchronous pareto-based multi-objective artificial bee colony algorithms. *Journal of Global Optimization*, 57(2):415–445, **2013**. ISSN 1573-2916. doi:10.1007/s10898-012-9993-1.

[114] Avadh Kishor, Pramod Kumar Singh, and Jay Prakash. Nsabc: Non-dominated sorting based multi-objective artificial bee colony algorithm and its application in data clustering. *Neurocomputing*, 216:514 – 533, **2016**. ISSN 0925-2312. doi:https://doi.org/10.1016/j.neucom.2016.08.003.

[115] Yi Xiang, Yuren Zhou, and Hailin Liu. An elitism based multi-objective artificial bee colony algorithm. *European Journal of Operational Research*, 245(1):168 – 193, **2015**. ISSN 0377-2217. doi:https://doi.org/10.1016/j.ejor.2015.03.005.

[116] Yu bin Zhong, Yi Xiang, and Hai-Lin Liu. A multi-objective artificial bee colony algorithm based on division of the searching space. *Applied Intelligence*, 41:987–1011, **2014**.

[117] Yi Xiang and Yuren Zhou. A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization. *Applied Soft Computing*, 35:766 – 785, **2015**. ISSN 1568-4946. doi:https://doi.org/10.1016/j.asoc.2015.06.033.

[118] Margarita Reyes-Sierra, CA Coello Coello, et al. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International journal of computational intelligence research*, 2(3):287–308, **2006**.

[119] Y. Liu, L. Ma, and G. Yang. A survey of artificial bee colony algorithm. In *2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 1510–1515. **2017**. ISSN null. doi:10.1109/CYBER.2017.8446301.

[120] I. Alaya, C. Solnon, and K. Ghedira. Ant colony optimization for multi-objective optimization problems. In *19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007)*, volume 1, pages 450–457. **2007**. ISSN 2375-0197. doi:10.1109/ICTAI.2007.108.

[121] Xin-She Yang. Bat algorithm for multi-objective optimisation. *arXiv preprint arXiv:1203.6571*, **2012**.

[122] Xin-She Yang. Multiobjective firefly algorithm for continuous optimization. *Engineering with Computers*, 29(2):175–184, **2013**.

[123] H. R. Hassanzadeh and M. Rouhani. A multi-objective gravitational search algorithm. In *2010 2nd International Conference on Computational Intelligence, Communication Systems and Networks*, pages 7–12. **2010**. ISSN null. doi: 10.1109/CICSyN.2010.32.

[124] Seyedali Mirjalili, Shahrzad Saremi, Seyed Mohammad Mirjalili, and Leandro dos S. Coelho. Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization. *Expert Systems with Applications*, 47:106 – 119, **2016**. ISSN 0957-4174. doi:https://doi.org/10.1016/j.eswa.2015.10.039.

[125] Xin-She Yang and Suash Deb. Multiobjective cuckoo search for design optimization. *Computers & Operations Research*, 40(6):1616 – 1624, **2013**. ISSN 0305-0548. doi:https://doi.org/10.1016/j.cor.2011.09.026. Emergent Nature Inspired Algorithms for Multi-Objective Optimization.

[126] Feng Zou, Lei Wang, Xinhong Hei, Debao Chen, and Bin Wang. Multi-objective optimization using teaching-learning-based optimization algorithm. *Engineering Applications of Artificial Intelligence*, 26(4):1291 – 1300, **2013**. ISSN 0952-1976. doi:https://doi.org/10.1016/j.engappai.2012.11.006.

[127] P. Moller. *Electric fishes: history and behavior*. Chapman and Hall fish and fisheries series. Chapman & Hall, **1995**. ISBN 9780412373800.

[128] Bernd Kramer. *Electroreception and communication in fishes*, volume 42. Gustav Fischer, **1996**.

[129] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Inc., New York, NY, USA, **1999**. ISBN 0-19-513159-2.

[130] Dervis Karaboga and Bahriye Akay. A survey: algorithms simulating bee swarm intelligence. *Artificial Intelligence Review*, 31(1):61, **2009**. ISSN 1573-7462. doi:10.1007/s10462-009-9127-4.

[131] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer Berlin Heidelberg, **2003**. doi:10.1007/978-3-662-05094-1.

[132] Payam Zahadat and Thomas Schmickl. Wolfpack-inspired evolutionary algorithm and a reaction-diffusion-based controller are used for pattern formation.

In *Proceedings of the 2014 conference on Genetic and evolutionary computation - GECCO 14*. ACM Press, **2014**. doi:10.1145/2576768.2598262.

[133] Thomas Weise. *Global Optimization Algorithms – Theory and Application*. Thomas Weise, 2008-1-4 edition, **2008**.

[134] Xin-She Yang. 1 - optimization and metaheuristic algorithms in engineering. In Xin-She Yang, Amir Hossein Gandomi, Siamak Talatahari, and Amir Hossein Alavi, editors, *Metaheuristics in Water, Geotechnical and Transport Engineering*, pages 1 – 23. Elsevier, Oxford, **2013**. ISBN 978-0-12-398296-4.

[135] Bahriye Akay and Dervis Karaboga. Artificial bee colony algorithm for large-scale problems and engineering design optimization. *Journal of Intelligent Manufacturing*, 23(4):1001–1014, **2012**. ISSN 1572-8145. doi: 10.1007/s10845-010-0393-4.

[136] Hailong Wang, Zhongbo Hu, Yuqiu Sun, Qinghua Su, and Xuewen Xia. A novel modified bsa inspired by species evolution rule and simulated annealing principle for constrained engineering optimization problems. *Neural Computing and Applications*, **2018**. ISSN 1433-3058. doi:10.1007/s00521-017-3329-5.

[137] Enda Ridge and Daniel Kudenko. Screening the parameters affecting heuristic performance. In *In Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, **2007**.

[138] Sevil Sen. *Evolutionary Computation Techniques for Intrusion Detection in Mobile Ad Hoc Networks*. Ph.D. thesis, University of York, **2010**.

[139] Richard S. Barr, Bruce L. Golden, James P. Kelly, Mauricio G. C. Resende, and William R. Stewart. Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, 1(1):9–32, **1995**. ISSN 1572-9397. doi:10.1007/BF02430363.

[140] Dervis Karaboga and Bahriye Akay. A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, 214(1):108 – 132, **2009**. ISSN 0096-3003. doi:https://doi.org/10.1016/j.amc.2009.03.090.

[141] Xin-She Yang, Zhihua Cui, Renbin Xiao, Amir Hossein Gandomi, and Mehmet Karamanoglu. *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*. Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, 1st edition, **2013**. ISBN 0124051634, 9780124051638.

[142] Pinar Civicioglu. Backtracking search optimization algorithm for numerical optimization problems. *Applied Mathematics and Computation*, 219(15):8121 – 8144, **2013**. ISSN 0096-3003. doi:https://doi.org/10.1016/j.amc.2013.02.017.

[143] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3 – 18, **2011**. ISSN 2210-6502. doi: https://doi.org/10.1016/j.swevo.2011.02.002.

[144] Serafim Opricovic. Multicriteria optimization of civil engineering systems. *Faculty of Civil Engineering, Belgrade*, 2(1):5–21, **1998**.

[145] Ching-Lai Hwang and Kwangsun Yoon. *Multiple attribute decision making: methods and applications a state-of-the-art survey*, volume 186. Springer Science & Business Media, **2012**.

[146] Thomas Weise. *Global Optimization Algorithms – Theory and Application*. Thomas Weise, 2008-1-4 edition, **2008**. Online available at http://www.it-weise.de/.

[147] CL Blake and CJ Merz. University of california at irvine repository of machine learning databases. *Department of Information and Computer Science*, **1998**.

[148] I. De Falco, A. Della Cioppa, and E. Tarantino. Facing classification problems with particle swarm optimization. *Applied Soft Computing*, 7(3):652 – 658, **2007**. ISSN 1568-4946.

[149] Dervis Karaboga and Celal Ozturk. A novel clustering approach: Artificial bee colony (abc) algorithm. *Applied Soft Computing*, 11(1):652 – 657, **2011**. ISSN 1568-4946.

[150] Sanghamitra Bandyopadhyay and Ujjwal Maulik. Genetic clustering for automatic evolution of clusters and application to image classification. *Pattern Recognition*, 35(6):1197 – 1208, **2002**. ISSN 0031-3203.

[151] Shuyan Fan, Shifei Ding, and Yu Xue. Self-adaptive kernel k-means algorithm based on the shuffled frog leaping algorithm. *Soft Computing*, 22(3):861–872, **2018**. ISSN 1433-7479. doi:10.1007/s00500-016-2389-2.

[152] Ahmad Karim, Rosli Bin Salleh, Muhammad Shiraz, Syed Adeel Ali Shah, Irfan Awan, and Nor Badrul Anuar. Botnet detection techniques: review, future trends, and issues. *Journal of Zhejiang University SCIENCE C*, 15(11):943–983, **2014**. ISSN 1869-196X.

[153] Khalid Huseynov, Kwangjo Kim, and Paul D Yoo. Semi-supervised botnet detection using ant colony clustering. In *Proc. Symp. Cryptogr. Inf. Secur.(SCIS)*, pages 1–7. **2014**.

[154] Pratik Narang, Jagan Mohan Reddy, and Chittaranjan Hota. Feature selection for detection of peer-to-peer botnet traffic. In *Proceedings of the 6th ACM India Computing Convention*, Compute '13, pages 16:1–16:9. ACM, New York, NY, USA, **2013**. ISBN 978-1-4503-2545-5.

[155] G. Kirubavathi Venkatesh and R. Anitha Nadarajan. Http botnet detection using adaptive learning rate multilayer feed-forward neural network. In Ioannis Askoxylakis, Henrich C. Pohls, and Joachim Posegga, editors, *Information Security Theory and Practice. Security, Privacy and Trust in Computing Systems*

and Ambient Intelligent Ecosystems, pages 38–48. Springer Berlin Heidelberg, Berlin, Heidelberg, **2012**.

[156] G. Kirubavathi and R. Anitha. Botnet detection via mining of traffic flow characteristics. *Computers & Electrical Engineering*, 50:91 – 101, **2016**. ISSN 0045-7906.

[157] A. Nogueira, P. Salvador, and F. Blessa. A botnet detection system based on neural networks. In *2010 Fifth International Conference on Digital Telecommunications*, pages 57–62. **2010**.

[158] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, **1986**.

[159] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, and P. Hakimian. Detecting p2p botnets through network behavior analysis and machine learning. In *2011 Ninth Annual International Conference on Privacy, Security and Trust*, pages 174–180. **2011**.

[160] Kuochen Wang, Chun-Ying Huang, Shang-Jyh Lin, and Ying-Dar Lin. A fuzzy pattern-based filtering algorithm for botnet detection. *Computer Networks*, 55(15):3275 – 3286, **2011**. ISSN 1389-1286.

[161] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer. Using machine learning techniques to identify botnet traffic. In *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, pages 967–974. **2006**. ISSN 0742-1303.

[162] Gregory Fedynyshyn, Mooi Choo Chuah, and Gang Tan. Detection and classification of different botnet c&c channels. In Jose M. Alcaraz Calero, Laurence T. Yang, Félix Gómez Mármol, Luis Javier García Villalba, Andy Xiaolin Li, and Yan Wang, editors, *Autonomic and Trusted Computing*, pages 228–242. Springer Berlin Heidelberg, Berlin, Heidelberg, **2011**.

[163] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, and P. Hakimian. Detecting p2p botnets through network behavior analysis and machine learning. In *2011 Ninth Annual International Conference on Privacy, Security and Trust*, pages 174–180. **2011**.

[164] French chapter of honeynet. `http://www.honeynet.org/chapters/france`. Accessed: 2018-07-01.

[165] Geza Szabo, Daniel Orincsay, Szabolcs Malomsoky, and Istvan Szabo. On the validation of traffic classification algorithms. In Mark Claypool and Steve Uhlig, editors, *Passive and Active Network Measurement*, pages 72–81. Springer Berlin Heidelberg, Berlin, Heidelberg, **2008**.

[166] Lbnl enterprise trace repository. `http://www.icir.org/enterprise-tracing/`. Accessed: 2018-07-01.

[167] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy*. SCITEPRESS - Science and Technology Publications, **2018**.

[168] Arash Habibi Lashkari, Gerard Draper Gil, Mohammad Saiful Islam Mamun, and Ali A. Ghorbani. Characterization of tor traffic using time based features. In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,*, pages 253–262. INSTICC, SciTePress, **2017**. ISBN 978-989-758-209-7.

[169] jnetpcap. `http://jnetpcap.com`. Accessed: 2018-07-01.

[170] Ian H. Witten, Eibe Frank, Leonard E. Trigg, Mark A. Hall, Geoffrey Holmes, and Sally Jo Cunningham. Weka: Practical machine learning tools and techniques with java implementations. Technical report, **1999**. 99/11.

[171] Wei Lu, Goaletsa Rammidi, and Ali A. Ghorbani. Clustering botnet communication traffic based on n-gram feature selection. *Computer Communications*, 34(3):502 – 514, **2011**. ISSN 0140-3664. Special Issue of Computer Communications on Information and Future Communication Security.

[172] Tristan Henderson, David Kotz, and Ilya Abyzov. The changing usage of a mature campus-wide wireless network. In *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, MobiCom '04, pages 187–201. ACM, New York, NY, USA, **2004**. ISBN 1-58113-868-7.

[173] Selim Yilmaz and Sevil Sen. Early detection of botnet activities using grammatical evolution. In Paul Kaufmann and Pedro A. Castillo, editors, *Applications of Evolutionary Computation*, pages 395–404. Springer International Publishing, Cham, **2019**.

[174] Singiresu S. Rao. *Engineering Optimization: Theory and Practice, 3rd Edition*. Wiley-Interscience, **1996**. ISBN 9780471550341.

[175] Jasbir S Arora. Introduction to optimum design, 1989. *McGraw-Mill Book Company*, **1967**.

[176] E. Sandgren. Nonlinear integer and discrete programming in mechanical design optimization. *Journal of Mechanical Design*, 112(2):223, **1990**. doi:10.1115/1.2912596.

[177] Jan Golinski. Optimal synthesis problems solved by means of nonlinear programming and random methods. *Journal of Mechanisms*, 5(3):287 – 309, **1970**. ISSN 0022-2569. doi:https://doi.org/10.1016/0022-2569(70)90064-9.

[178] Amir Hossein Gandomi and Xin-She Yang. *Benchmark Problems in Structural Optimization*, pages 259–281. Springer Berlin Heidelberg, Berlin, Heidelberg, **2011**. ISBN 978-3-642-20859-1.

[179] Carlos A Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information systems*, 1(3):269–308, **1999**.

[180] J. J. Durillo and A. J. Nebro. jMetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760–771, **2011**. ISSN 0965-9978. doi:DOI:10.1016/j.advengsoft.2011.05.014.

[181] J.J. Durillo, A.J. Nebro, and E. Alba. The jMetal framework for multi-objective optimization: Design and architecture. In *CEC 2010*, pages 4138–4325. Barcelona, Spain, **2010**.

APPENDICES

APPENDIX A – Source Code (Matlab) of Electric Fish Optimization Algorithm

```
function EFO
%%   Electric Fish Optimization Algorithm, @version 1.0 (01/2020)
%
%   Electric Fish Optimization (or EFO) algorithm, inspired from the
%   object detection (electrolocation) and communication
%   (electrocommunication) bahaviors of electric fish, is the most recently
%   swarm-intelligence based algorithm proposed by Selim Yilmaz and Sevil
%   Sen.
%
%   Contact author:
%   Selim Yilmaz {selimy[at]cs.hacettepe.edu.tr}


clear all
clc

%   Control parameters of EFO
populationSize = ;                      %   Population size, (M)
maxCycle       = ;                      %   Maximum number of iterations
x              = ;                      %   Magnitudes old amplitude
activeIndSize  = ;                      %   Subpopulation size, (K)
iterCounter    = ;                      %   Iteration counter
problemIndex   = ;                      %   Problem index
thres          = ;                      %   A small constant value

%   Problem specific parameters
problem        = ;                      %   A struct that gets problem information
lowerBound     = problem.lowerBound;    %   Lower limit of the problem, (x_maxj)
upperBound     = problem.upperBound;    %   Upper limit of the problem, (x_minj)
parameterSize  = problem.parameterSize; %   Parameter size to be optimized, (d)
name           = problem.name;          %   Problem name
fMin           = problem.fMin;          %   Global minimum of the problem


population     = struct;                %   Population struct, (N)
```

```
%   Initially all individuals are first initialized in the range of
%   [x_maxj, x_minj] then their fitness values are calculated
for p = 1 : populationSize
  population(p).solution = lowerBound + (upperBound - lowerBound) * rand(1, parameterSize);
  population(p).fitness  = ;   %  Evaluates fitness
  population(p).isActive = true;
end


%   The best individual information is stored
[bestCost, index]  = min( [population(:).fitness] );
bestIndividual     = population(index).solution;

%   The worst individual information is also stored for frequency calculation
[worstCost, index] = max( [population(:).fitness] );
worstIndividual    = population(index).solution;

globalBest         = bestCost;
globalBestSol      = bestIndividual;


%   Frequency (f) and amplitude (A) values are also initialized
for p = 1 : populationSize
  if std([population(:).fitness]) < thres,
    population(p).frequency = rand;
  else
    population(p).frequency = (worstCost - population(p).fitness) / (worstCost - bestCost);
  end
    population(p).amplitude = population(p).frequency;
end


%   Beginning of the optimization
while (iterCounter < maxCycle && globalBest ~= fMin)

    for p = 1 : populationSize  %   For every individual do,
        newIndividual      = population(p);  %   ith individual, (i)
        neighborIndividual = population;    neighborIndividual(p) = []; %   Neighbors of i

        if (newIndividual.frequency > rand) %   Active electrolocation phase
          population(p).isActive = true;  %   ith individual will be in active mode
          newIndividual.solution = activeSearch(newIndividual, neighborIndividual,
                                    lowerBound, upperBound);   %   Performs search


```

```
86          else                                  %   Passive electrolocation phase
87            population(p).isActive = false; %   ith individual will be in passive mode
88            activePopulation       = neighborIndividual( [neighborIndividual(:).isActive] );
89            if ~isempty(activePopulation)
90              newIndividual.solution = passiveSearch(newIndividual, activePopulation,
91                                       activeIndSize);  %   Performs search
92            end
93            if rand < rand, %   Mutates one more parameter
94              I = randi(parameterSize);
95              newIndividual.solution(I) = lowerBound + (upperBound - lowerBound) * rand;
96            end
97          end
98
99          newIndividual.solution = boundaryCheck(newIndividual.solution, lowerBound,
100                                  upperBound); %   Checks the constraint violations
101
102         newIndividual.fitness = ;   %   Evaluates fitness
103         if  newIndividual.fitness < population(p).fitness   %   Accepts if better
104           population(p) = newIndividual;
105         end
106       end
107
108       %   The best individual information is stored
109       [bestCost, index] = min( [population(:).fitness] );
110       bestIndividual    = population(index).solution;
111
112       %   The worst individual information is also stored for frequency calculation
113       [worstCost, index] = max( [population(:).fitness] );
114       worstIndividual    = population(index).solution;
115
116       %   Frequency (f) and amplitude (A) values are updated
117       for p = 1: populationSize
118         if std([population(:).fitness]) < thres,
119           population(p).frequency = rand;
120         else
121           population(p).frequency = (worstCost - population(p).fitness)
122                                   / (worstCost - bestCost);
123         end
124           population(p).amplitude = population(p).amplitude * (x)
125                                   + population(p).frequency * (1 - x);
126       end
127
128
129
130
```

```
131     if bestCost < globalBest,   %   The best cost and solution found so far are stored
132         globalBest = bestCost;
133         globalBestSol = bestIndividual;
134     end
135
136     iterCounter = iterCounter + 1;
137
138     disp(['EFO  Problem:', num2str(problemIndex), '-',name,' D:',num2str(problem
139          .parameterSize), ' iter:', num2str(iterCounter), ' Fmin:', num2str(globalBest)])
140
141 %   End of the iteration
142 end
143
144 end
145
146 %%  Active Electrolocation Phase
147 function newSolution = activeSearch(newIndividual, neighborIndividual,
148                         lowerBound, upperBound)
149
150    activeRange = (upperBound - lowerBound) *
151                         newIndividual.amplitude;    %   Calculates active search range
152
153    parameter    = randi(size(newIndividual.solution, 2)); %   Determines jth parameter
154
155    dist         = getDistance(newIndividual.solution,
156                         [neighborIndividual(:).solution]);   %   Obtains cartesian distance
157
158    index        = find(dist < activeRange);   %   is a neighbor in the active search?
159
160    if isempty(index)    %   If no neighbor exists in the search range
161        newIndividual.solution(parameter) = newIndividual.solution(parameter)
162    else    %   If at least one neighbor exists in the search range
163        selectedNeighbor = index(randi(size(index, 1))); %   Randomly selects one neighbor
164        newIndividual.solution(parameter) = newIndividual.solution(parameter) +
165                                            (neighborIndividual(selectedNeighbor).solution
166                                            (parameter) - newIndividual.solution(parameter))
167                                            * (2 * rand - 1);
168    end
169    newSolution = newIndividual.solution;
170
171 end
172
173
174
175
```

```
%%  Passive Electrolocation Phase
function newSolution = passiveSearch(newIndividual, activePopulation, activeIndSize)

   e = 1e-6;   %   A small constant used in distance calculation
   D = size(newIndividual.solution, 2);    %   Parameter size of the problem

   if size(activePopulation, 2) < activeIndSize,   %   If the number of active individuals
    is less than that of predetermined neighbor, (K)
      activeIndSize = size(activePopulation, 2);  %   Set K as the number of active
    individuals
   end

   dist = getDistance(newIndividual.solution, [activePopulation(:).solution]); %   Obtains
    cartesian distance between ith individual and those in active modes
   p    = cumsum([activePopulation(:).amplitude] ./ (dist' + e));   %   The probability
    values of active mode individuals
   selectedNeighbor = zeros(1, activeIndSize);

   for i = 1 : activeIndSize,  %   K individuals are selected
      I = find (rand*p(end) < p);
      selectedNeighbor(i) = I(1);
   end

   neighbor    = activePopulation(selectedNeighbor);
   solutionSet = reshape([neighbor.solution], D, activeIndSize)';
   xReference  = sum(repmat([neighbor.amplitude]', 1, D) .* solutionSet)
                 ./ sum(repmat([neighbor.amplitude]', 1, D));    %   A reference point
   newSolution = newIndividual.solution + (xReference - newIndividual.solution)
                 .* (2 * rand(1, D) - 1);    %   Generates new location
   I           = find(rand(1, D) > newIndividual.frequency); %   Determines parameters
   newIndividual.solution(I) = newSolution(I);
   newSolution = newIndividual.solution;

end

%   Returns the cartesian distance between individual 'newIndividualSol' and
%   its neigbors 'solutionSet'
function dist = getDistance(newIndividualSol, solutionSet)
   parameterSize = size(newIndividualSol, 2);
   popSize       = size(solutionSet, 2) / parameterSize;
   solutionSet   = reshape(solutionSet, parameterSize, popSize)';
   dist          = sqrt(sum((solutionSet - repmat(newIndividualSol, popSize, 1)) .^2, 2));
end
```

```
217 %   Controls if a parameter value exceeds the space limits, if so it is
218 %   relocated into the bound where it exceeds
219 function newIndividualSol = boundaryCheck(newIndividualSol, lowerBound, upperBound)
220    newIndividualSol = min(newIndividualSol, upperBound);
221    newIndividualSol = max(newIndividualSol, lowerBound);
222 end
```

APPENDIX B – Properties of Basic and Complex Benchmark Problem Sets

Table B.1. Mathematical benchmark function set used in the first experiment. (D: Parameter size, T: Type, U: Unimodal, M: Multimodal, S: Separable, N: Nonseparable.)

No	Function	D	T	Min	Formulation
F1	Stepint	5	US	0	$f(x) = 25 + \sum_{i=1}^{5} \lfloor x_i \rfloor$
F2	Step	30	US	0	$f(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$
F3	Sphere	30	US	0	$f(x) = \sum_{i=1}^{n} x_i^2$
F4	Sum Squares	30	US	0	$f(x) = \sum_{i=1}^{n} i x_i^2$
F5	Quartic	30	US	0	$f(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$
F6	Beale	5	UN	0	$f(x) = (1.5 - x_1 + x_1 x_2)^2 + \left(2.25 - x_1 + x_1 x_2^2\right)^2 + \left(2.625 - x_1 + x_1 x_2^3\right)^2$
F7	Easom	2	UN	-1	$f(x) = -\cos(x_1)\cos(x_2)\exp\left(-(x_1-\pi)^2 - (x_2-\pi)^2\right)$
F8	Matyas	2	UN	0	$f(x) = 0.26\left(x_1^2 + x_2^2\right) - 0.48 x_1 x_2$
F9	Colville	4	UN	0	$f(x) = 100\left(x_1^2 - x_2\right)^2 + (x_1-1)^2 + (x_3-1)^2 + 90\left(x_3^2 - x_4\right)^2 + 10.1\left((x_2-1)^2 + (x_4-1)^2\right) + 19.8(x_2-1)(x_4-1)$
F10	Trid 6	6	UN	-50	$f(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$
F11	Trid 10	10	UN	-210	$f(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$
F12	Zakharov	10	UN	0	$f(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^4$
F13	Powell	24	UN	0	$f(x) = \sum_{i=1}^{n/k} \left(x_{4i-3} + 10x_{4i-2}\right)^2 + 5\left(x_{4i-1} + x_{4i}\right)^2 + \left(x_{4i-2} + x_{4i-1}\right)^4 + 10\left(x_{4i-3} + 10x_{4i}\right)^4$
F14	Schwefel 2.22	30	UN	0	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
F15	Schwefel1.2	30	UN	0	$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$
F16	Rosenbrock	30	UN	0	$f(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i\right)^2 + (x_i - 1)^2\right]$
F17	Dixon-Price	30	UN	0	$f(x) = (x_i - 1)^2 + \sum_{i=1}^{n} i\left(2x_i^2 - x_{i-1}\right)^2$
F18	Fox-holes	2	MS	0.998	$f(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}\left(x_i - a_{ij}\right)^6}\right]^{-1}$
F19	Branin	2	MS	0.398	$f(x) = \left(x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$
F20	Bohachevsky 1	2	MS	0	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$
F21	Booth	2	MS	0	$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$
F22	Rastrigin	30	MS	0	$f(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$
F23	Schwefel	30	MS	-12569.5	$f(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{x_i}\right)$
F24	Michalewicz 2	2	MS	-1.8013	$f(x) = -\sum_{i=1}^{n} \sin(x_i)\left(\sin\left(\frac{i x_i^2}{\pi}\right)\right)^{2m}$
F25	Michalewicz 5	5	MS	-4.6877	$f(x) = -\sum_{i=1}^{n} \sin(x_i)\left(\sin\left(\frac{i x_i^2}{\pi}\right)\right)^{2m}$
F26	Michalewicz10	10	MS	-9.6602	$f(x) = -\sum_{i=1}^{n} \sin(x_i)\left(\sin\left(\frac{i x_i^2}{\pi}\right)\right)^{2m}$
F27	Schaffer	2	MN	0	$f(x) = 0.5 + \frac{\sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - 0.5}{\left(1 + 0.001\left(x_1^2 + x_2^2\right)\right)^2}$
F28	SHCB	2	MN	-1.03163	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$
F29	Bohachevsky2	2	MN	0	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)(4\pi x_2) + 0.3$
F30	Bohachevsky 3	2	MN	0	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3$
F31	Shubert	2	MN	-186.73	$f(x) = \left(\sum_{i=1}^{5} i\cos\left((i+1)x_1 + i\right)\right)\left(\sum_{i=1}^{5} i\cos\left((i+1)x_2 + i\right)\right)$

(continued on next page)

Table B.1. *(continued).*

No	Function	D	T	Min	Formulation
F32	Goldstein-Price	2	MN	3	$f(x) = \left[1 + (x_1 + x_2 + 1)^2 X_1\right]\left[30 + (2x_1 - 3x_2)^2 X_2\right]$ $X_1 = \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)$ $X_2 = \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)$
F33	Kowalik	4	MN	0.00031	$f(x) = \sum_{i=1}^{11}\left(a_i - \frac{x_1\left(b_i^2 + b_i x_2\right)^2}{b_i^2 + b_i x_3 + x_4}\right)$
F34	Shekel 5	4	MN	-10.15	$f(x) = -\sum_{i=1}^{5}\left[(x - a_i)(x - a_i)^T + c_i\right]^{-1}$
F35	Shekel7	4	MN	-10.4	$f(x) = -\sum_{i=1}^{7}\left[(x - a_i)(x - a_i)^T + c_i\right]^{-1}$
F36	Shekel 10	4	MN	-10.53	$f(x) = -\sum_{i=1}^{10}\left[(x - a_i)(x - a_i)^T + c_i\right]^{-1}$
F37	Perm	4	MN	0	$f(x) = \sum_{k=1}^{n}\left[\sum_{i=1}^{n}\left(i^k + \beta\right)\left(\left(\frac{x_i}{i}\right)^k - 1\right)\right]^2$
F38	Power Sum	4	MN	0	$f(x) = \sum_{k=1}^{n}\left[\left(\sum_{i=1}^{n} x_i^k\right) - b_k\right]^2$
F39	Hartman3	3	MN	-3.86	$f(x) = -\sum_{i=1}^{4} c_i exp\left[-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2\right]$
F40	Hartman 6	6	MN	-3.32	$f(x) = -\sum_{i=1}^{4} c_i exp\left[-\sum_{j=1}^{6} a_{ij}\left(x_j - p_{ij}\right)^2\right]$
F41	Griewank	30	MN	0	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
F42	Ackley	30	MN	0	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right) + 20 + e$
F43	Penalized1	30	MN	0	$f(x) = \frac{\pi}{n}\left\{\begin{array}{c} 10\sin^2\pi y_1 + \\ \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10\sin^2\left(\pi y_{i+1}\right)\right] + \\ (y_n - 1)^2 \end{array}\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(x_i, a, k, m) = \left\{\begin{array}{ll} k(x_i - a)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k(-x_i - a)^m & x_i < -a \end{array}\right.$
F44	Penalized 2	30	MN	0	$f(x) = 0.1\left\{\begin{array}{c} \sin^2(\pi x_1) + \\ \sum_{i=1}^{n-1}(x_i - 1)^2\left[11 + \sin^2\left(3\pi x_{i+1}\right)\right] + \\ (x_n - 1)^2\left[1 + \sin^2\left(2\pi x_n\right)\right] \end{array}\right\} + \sum_{i=1}^{n} u(5, 100, 4)$
F45	Langerman 2	2	MN	-1.08	$f(x) = -\sum_{i=1}^{m} c_i A_i B_i$ $A_i = \left(\exp\left(-\frac{1}{\pi}\sum_{j=1}^{n}\left(x_j - a_{ij}\right)^2\right)\right)$ $B_i = \cos\left(\pi\sum_{j=1}^{n}\left(x_j - a_{ij}\right)^2\right)$
F46	Langerman 5	5	MN	-1.5	$f(x) = -\sum_{i=1}^{m} c_i A_i B_i$ $A_i = \left(\exp\left(-\frac{1}{\pi}\sum_{j=1}^{n}\left(x_j - a_{ij}\right)^2\right)\right)$ $B_i = \cos\left(\pi\sum_{j=1}^{n}\left(x_j - a_{ij}\right)^2\right)$
F47	Langerman 10	10	MN	N.A.	$f(x) = -\sum_{i=1}^{m} c_i A_i B_i$ $A_i = \left(\exp\left(-\frac{1}{\pi}\sum_{j=1}^{n}\left(x_j - a_{ij}\right)^2\right)\right)$ $B_i = \cos\left(\pi\sum_{j=1}^{n}\left(x_j - a_{ij}\right)^2\right)$
F48	Fletcher Powell 2	2	MN	0	$f(x) = \sum_{i=1}^{n}(A_i - B_i)^2$ $A_i = \sum_{j=1}^{n}\left(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j\right)$ $B_i = \sum_{j=1}^{n}\left(a_{ij}\sin x_j + b_{ij}\cos x_j\right)$
F49	Fletcher Powell 5	5	MN	0	$f(x) = \sum_{i=1}^{n}(A_i - B_i)^2$ $A_i = \sum_{j=1}^{n}\left(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j\right)$ $B_i = \sum_{j=1}^{n}\left(a_{ij}\sin x_j + b_{ij}\cos x_j\right)$
F50	Fletcher Powell 10	10	MN	0	$f(x) = \sum_{i=1}^{n}(A_i - B_i)^2$ $A_i = \sum_{j=1}^{n}\left(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j\right)$ $B_i = \sum_{j=1}^{n}\left(a_{ij}\sin x_j + b_{ij}\cos x_j\right)$

Table B.2. Properties of complex mathematical benchmark set. D: Parameter size, T: Type, Low and Up: Lower and upper limit, U: Unimodal, M: Multimodal, H: Hybrid, C: Composition

No	Function	D	T	Low	Up
F1	Shifted and Rotated *Bent Cigar*	10	U	-100	100
F2	Shifted and Rotated *Sum of Different Power*	10	U	-100	100
F3	Shifted and Rotated *Zakharov*	10	U	-100	100
F4	Shifted and Rotated *Rosenbrock*	10	M	-100	100
F5	Shifted and Rotated *Rastrigin*	10	M	-100	100
F6	Shifted and Rotated *Expanded Scaffer*	10	M	-100	100
F7	Shifted and Rotated *Lunacek Bi-Rastrigin*	10	M	-100	100
F8	Shifted and Rotated *Non-Continuous Rastrigin*	10	M	-100	100
F9	Shifted and Rotated *Levy*	10	M	-100	100
F10	Shifted and Rotated *Schwefel*	10	M	-100	100
F11	Hybrid of *Zakharov*, *Rosenbrock*, and *Rastrigin*	10	H	-100	100
F12	Hybrid of *High Conditioned Elliptic*, *Modified Schwefel*, and *Bent Cigar*	10	H	-100	100
F13	Hybrid of *Bent Cigar*, *Rosenbrock*, and *Lunacek Bi-Rastrigin*	10	H	-100	100
F14	Hybrid of *High Conditioned Elliptic*, *Ackley*, *Schaffer*, and *Rastrigin*	10	H	-100	100
F15	Hybrid of *Bent Cigar*, *HGBat*, *Rastrigin*, and *Rosenbrock*	10	H	-100	100
F16	Hybrid of *Expanded Schaffer*, *HGBat*, *Rosenbrock*, and *Modified Schwefel*	10	H	-100	100
F17	Hybrid of *Katsuura*, *Ackley*, *Expanded Griewank & Rosenbrock*, *Modified Schwefel*, and *Rastrigin*	10	H	-100	100
F18	Hybrid of *High Conditioned Elliptic*, *Ackley*, *Rastrigin*, *HGBat*, and *Discus*	10	H	-100	100
F19	Hybrid of *Bent Cigar*, *Rastrigin*, *Expanded Griewank & Rosenbrock*, *Weierstrass*, and *Expanded Schaffer*	10	H	-100	100
F20	Hybrid of *HappyCat*, *Katsuura*, *Ackley*, *Rastrigin*, *Modified Schwefel*, and *Schaffer*	10	H	-100	100
F21	Composition of *Rosenbrock*, *High Conditioned Elliptic*, and *Rastrigin*	10	C	-100	100
F22	Composition of *Rastrigin*, *Griewank*, and *Modified Schwefel*	10	C	-100	100
F23	Composition of *Rosenbrock*, *Ackley*, *Modified Schwefel*, and *Rastrigin*	10	C	-100	100
F24	Composition of *Ackley*, *High Conditioned Elliptic*, *Girewank*, and *Rastrigin*	10	C	-100	100
F25	Composition of *Rastrigin*, *HappyCat*, *Ackley*, *Discus*, and *Rosenbrock*	10	C	-100	100
F26	Composition of *Expanded Scaffer*, *Modified Schwefel*, *Griewank*, *Rosenbrock*, and *Rastrigin*	10	C	-100	100

(continued on next page)

Table B.2. *(continued)*.

No	Function	D	T	Low	Up
F27	Composition of *HGBat*, *Rastrigin*, *Modified Schwefel*, *Bent-Cigar*, *High Conditioned Elliptic*, and *Expanded Scaffer*	10	C	-100	100
F28	Composition of *Ackeley*, *Griewank*, *Discus*, *Rosenbrock*, *HappyCat*, and *Expanded Scaffer*	10	C	-100	100
F29	Composition of *three hybrid functions* (F15, F16, and F17)	10	C	-100	100
F30	Composition Function 10 (*with three functions*)	10	C	-100	100

APPENDIX C – Experimental Results on Basic and Complex Benchmark Problems

Table C.1. Wilcoxon Sign Test comparison of the algorithms for basic mathematical benchmark functions.

No	EFO vs. SA				EFO vs. VS				EFO vs. GA				EFO vs. DE				EFO vs. PSO				EFO vs. ABC			
	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W
F1	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=
F2	0	0	0	=	9.330e-06	0	325	+	1.729e-06	0	465	+	0	0	0	=	0	0	0	=	0	0	0	=
F3	1.734e-06	0	465	+	0	0	0	=	1.734e-06	0	465	+	0	0	0	=	0	0	0	=	1.734e-06	0	465	+
F4	1.734e-06	0	465	+	0.0003	0	153	+	1.734e-06	0	465	+	0	0	0	=	0	0	0	=	1.734e-06	0	465	+
F5	1.238e-05	445	20	-	0.0006	400	65	-	1.734e-06	0	465	+	2.127e-06	2	463	+	1.734e-06	465	0	-	1.734e-06	0	465	+
F6	1.734e-06	0	465	+	0	0	0	=	0	0	0	=	0	0	0	=	0.0833	0	6	=	1.734e-06	0	465	+
F7	2.517e-06	0	276	+	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=
F8	1.734e-06	0	465	+	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0.0015	0	91	+
F9	1.734e-06	0	465	+	1.734e-06	465	0	-	2.353e-06	3	462	+	1.734e-06	465	0	-	1.734e-06	465	0	-	1.734e-06	0	465	+
F10	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=
F11	0	0	0	=	0	0	0	=	1.734e-06	0	465	+	0	0	0	=	0	0	0	=	0	0	0	=
F12	1.734e-06	0	465	+	0	0	0	=	1.734e-06	0	465	+	0	0	0	=	0	0	0	=	1.734e-06	0	465	+
F13	1.734e-06	0	465	+	1.734e-06	0	465	+	1.734e-06	0	465	+	1.734e-06	0	465	+	1.734e-06	465	0	-	0.0117	355	110	-
F14	1.734e-06	0	465	+	1.734e-06	0	465	+	1.734e-06	0	465	+	0	0	0	=	0	0	0	=	1.734e-06	0	465	+
F15	1.734e-06	465	0	-	1.734e-06	465	0	-	1.734e-06	0	465	+	1.734e-06	0	465	+	1.734e-06	465	0	-	2.353e-06	3	462	+
F16	3.182e-06	6	459	+	1.734e-06	0	465	+	1.734e-06	0	465	+	1.734e-06	0	465	+	1.921e-06	1	464	+	0.0656	143	322	=
F17	3.515e-06	7	458	+	1.734e-06	0	465	+	1.734e-06	0	465	+	1.734e-06	0	465	+	1.734e-06	0	465	+	1.734e-06	465	0	-
F18	4.320e-08	0	465	+	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0.3173	0	1	=
F19	0.3173	7	2	=	0.6547	9	6	=	0.7055	12	16	=	1.0000	10	10	=	0.3173	7	2	=	1.0000	5	5	=
F20	1.734e-06	0	465	+	0	0	0	=	4.320e-08	0	465	+	0	0	0	=	0	0	0	=	0	0	0	=

Table C.1. (Continued)

No	EFO vs. SA				EFO vs. VS				EFO vs. GA				EFO vs. DE				EFO vs. PSO				EFO vs. ABC			
	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W
F21	1.734e-06	0	465	+	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=
F22	1.734e-06	0	465	+	1.728e-06	0	465	+	1.734e-06	0	465	+	1.734e-06	0	465	+	1.717e-06	0	465	+	0	0	0	=
F23	1.734e-06	0	465	+	1.725e-06	0	465	+	1.734e-06	0	465	+	0.0003	0	153	+	1.733e-06	0	465	+	0	0	0	=
F24	1.526e-05	300	0	-	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=
F25	3.773e-06	406	0	-	5.664e-05	0	231	+	0	0	0	=	0.3173	0	1	=	1.543e-05	0	300	+	0	0	0	=
F26	1.733e-06	465	0	-	1.734e-06	0	465	+	1.734e-06	0	465	+	0.1797	0	3	=	1.734e-06	0	465	+	0	0	0	=
F27	1.656e-06	0	465	+	0	0	0	=	1.013e-07	0	465	+	0.1573	0	3	=	0.1573	0	3	=	0	0	0	=
F28	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=
F29	1.207e-06	0	465	+	0	0	0	=	4.320e-08	0	465	+	0	0	0	=	0	0	0	=	0	0	0	=
F30	1.734e-06	0	465	+	0	0	0	=	7.752e-07	0	465	+	0	0	0	=	0	0	0	=	1.335e-05	0	300	+
F31	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=
F32	0.0005	0	120	+	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=
F33	1.734e-06	0	465	+	1.734e-06	465	0	-	0.0495	328	137	-	3.112e-05	435	30	-	3.112e-05	435	30	-	0.0017	80	385	+
F34	2.399e-06	0	435	+	0.5930	45	60	=	0.0021	5	100	+	0.1967	40	80	=	2.006e-05	1	298	+	0.2482	26	52	=
F35	4.671e-06	6	429	+	0.3458	66	104	=	0.0904	63	147	=	0.7815	42	49	=	0.0002	20	280	+	0.4795	22	13	=
F36	2.503e-06	0	435	+	0.7963	64	56	=	0.1177	20	58	=	0.0339	31	4	-	0.0005	9	181	+	0.5271	33	22	=
F37	1.734e-06	0	465	+	0.0039	373	92	-	0.1156	156	309	=	1.127e-05	446	19	-	0.3185	281	184	=	4.286e-06	9	456	+
F38	1.921e-06	1	464	+	1.734e-06	465	0	-	0.0034	375	90	-	0.1414	304	161	=	2.879e-06	460	5	-	9.316e-06	17	448	+
F39	1.652e-05	1	230	+	1.0000	10	10	=	0.3173	7	2	=	0.3173	7	2	=	0.6547	9	6	=	0.6547	9	6	=
F40	3.258e-07	0	465	+	0.0041	6	85	+	0.0019	12	141	+	0.2488	25	53	=	0.0004	9	181	+	0.2568	20	8	=

Table C.1. (Continued)

No	EFO vs. SA				EFO vs. VS				EFO vs. GA				EFO vs. DE				EFO vs. PSO				EFO vs. ABC			
	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W
F41	1.734e-06	0	465	+	1.884e-06	1	464	+	1.734e-06	0	465	+	0.1088	6	0	=	0.0018	5	115	+	0.0396	25	95	+
F42	1.734e-06	0	465	+	3.962e-05	432	33	-	1.734e-06	0	465	+	1.734e-06	0	465	+	2.308e-06	434	1	-	6.306e-06	451	14	-
F43	1.734e-06	0	465	+	5.896e-05	0	231	+	1.734e-06	0	465	+	0	0	0	=	0.0231	0	21	+	1.734e-06	0	465	+
F44	1.734e-06	0	465	+	0	0	0	=	1.734e-06	0	465	+	0	0	0	=	0	0	0	=	1.734e-06	0	465	+
F45	0.0009	0	105	+	0.5637	2	4	=	0.4142	7	14	=	1.0000	5	5	=	0.0005	8	163	+	0.1797	3	12	=
F46	9.712e-07	0	435	+	0.3173	0	1	=	0.3173	0	1	=	0	0	0	=	0.0001	0	190	+	0	0	0	=
F47	4.175e-07	0	465	+	0.1609	101	199	=	2.443e-06	0	435	+	0.0455	220	80	-	0.0009	34	266	+	0.3738	15	30	=
F48	1.734e-06	0	465	+	0	0	0	=	6.907e-07	0	465	+	0	0	0	=	0	0	0	=	0	0	0	=
F49	1.734e-06	0	465	+	0.0023	381	84	-	1.921e-06	1	464	+	0.1650	300	165	=	0.0036	91	374	+	0.0008	70	395	+
F50	1.360e-05	21	444	+	0.2712	286	179	=	1.921e-06	1	464	+	4.729e-06	10	455	+	6.339e-06	13	452	+	0.5577	204	261	=
+/=/-	38/7/5				13/29/8				29/19/2				9/36/5				17/26/7				17/30/3			

Table C.2. Wilcoxon Sign Test comparison of the algorithms for complex mathematical benchmark functions.

No	EFO vs. SA				EFO vs. VS				EFO vs. GA				EFO vs. DE				EFO vs. PSO				EFO vs. ABC			
	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W
F1	1.921e-06	1	464	+	1.921e-06	1	464	+	1.734e-06	0	465	+	1.732e-06	465	0	-	2.353e-06	3	462	+	3.112e-05	30	435	+
F2	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0.3173	0	1	=	0	0	0	=
F3	0.0041	55	0	-	0.0041	55	0	-	1.733e-06	0	465	+	0.0041	55	0	-	0.0041	55	0	-	1.734e-06	0	465	+
F4	0.0502	137	327	=	0.1413	150	285	=	2.561e-06	0	435	+	1.909e-06	1	464	+	0.1324	137	269	=	0.9038	184	194	=
F5	1.733e-06	0	465	+	5.276e-05	36	429	+	8.457e-06	16	449	+	0.1589	301	164	=	1.733e-06	0	465	+	0.0086	96	339	+
F6	1.734e-06	0	465	+	0.0006	0	120	+	1.711e-06	0	465	+	0	0	0	=	8.298e-06	0	351	+	0	0	0	=
F7	1.734e-06	0	465	+	2.603e-06	4	461	+	3.182e-06	6	459	+	4.860e-05	35	430	+	1.734e-06	0	465	+	0.0719	145	320	=
F8	1.729e-06	0	465	+	1.722e-06	0	465	+	2.351e-06	3	462	+	0.0786	147	318	=	1.720e-06	0	465	+	1.668e-05	18	416	+
F9	1.734e-06	0	465	+	0	0	0	=	1.702e-06	0	465	+	0	0	0	=	0.0033	0	66	+	0.0339	0	15	+
F10	1.734e-06	0	465	+	1.127e-05	19	446	+	0.0117	110	355	+	4.860e-05	430	35	-	1.734e-06	0	465	+	0.7655	218	247	=
F11	1.734e-06	0	465	+	2.124e-06	2	463	+	2.592e-06	4	461	+	2.529e-06	435	0	-	1.920e-06	1	464	+	0.2384	163	272	=
F12	0.8451	223	242	=	0.0495	137	328	+	0.1306	159	306	=	1.734e-06	465	0	-	0.3389	279	186	=	0.0041	93	372	+
F13	1.734e-06	0	465	+	1.734e-06	0	465	+	2.843e-05	29	436	+	1.734e-06	465	0	-	1.127e-05	19	446	+	0.0207	345	120	-
F14	1.734e-06	0	465	+	0.0028	378	87	-	5.787e-05	428	37	-	1.734e-06	465	0	-	0.2369	290	175	=	0.3600	188	277	=
F15	1.734e-06	0	465	+	2.369e-05	27	438	+	0.0285	126	339	+	1.731e-06	465	0	-	0.1650	165	300	=	0.4165	193	272	=
F16	1.734e-06	0	465	+	1.731e-06	0	465	+	3.670e-06	7	457	+	2.035e-06	463	2	-	1.733e-06	0	465	+	0.0286	107	299	+
F17	1.734e-06	0	465	+	1.732e-06	0	465	+	1.921e-06	1	464	+	2.266e-05	365	13	-	1.734e-06	0	465	+	0.0197	92	286	+
F18	1.734e-06	0	465	+	1.734e-06	0	465	+	2.879e-06	5	460	+	1.734e-06	465	0	-	0.0196	119	346	+	0.1414	161	304	=
F19	2.353e-06	3	462	+	0.1650	300	165	=	0.0015	387	78	-	1.734e-06	465	0	-	0.0012	390	74	-	0.0256	341	124	-
F20	1.734e-06	0	465	+	1.732e-06	0	465	+	2.152e-06	2	462	+	0.8907	7	8	=	1.734e-06	0	465	+	0.8539	4	5	=

Table C.2. (Continued)

No	EFO vs. SA				EFO vs. VS				EFO vs. GA				EFO vs. DE				EFO vs. PSO				EFO vs. ABC			
	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W
F21	2.127e-06	2	463	+	2.863e-06	460	5	-	0.0001	46	418	+	3.880e-06	8	457	+	1.972e-05	25	440	+	0.0817	137	298	=
F22	1.734e-06	0	465	+	2.127e-06	463	2	-	1.733e-06	0	465	+	7.230e-06	10	425	+	2.163e-05	26	439	+	0.1846	297	168	=
F23	1.734e-06	0	465	+	0.0350	130	335	+	0.0002	54	411	+	0.0243	342	123	-	1.734e-06	0	465	+	0.1414	161	304	=
F24	7.033e-06	2	376	+	0.8658	13	15	=	1.734e-06	0	465	+	1.730e-06	0	465	+	1.390e-05	1	324	+	0.0844	61	17	=
F25	0.0002	54	411	+	0.0002	34	344	+	2.554e-06	0	435	+	0.0022	61	316	+	4.256e-06	9	456	+	4.281e-06	456	9	-
F26	1.734e-06	0	465	+	1.649e-06	0	465	+	1.733e-06	0	465	+	1.671e-06	0	465	+	6.161e-06	1	377	+	0.2584	120	204	=
F27	1.921e-06	1	464	+	0.4528	269	196	=	0.1731	154	280	=	1.729e-06	465	0	-	1.732e-06	0	465	+	0.5038	265	200	=
F28	3.878e-06	0	378	+	0.0117	0	36	+	1.731e-06	0	465	+	3.631e-06	0	406	+	5.723e-05	0	231	+	0.9176	66	70	=
F29	1.734e-06	0	465	+	0.8612	224	241	=	0.0207	120	345	+	0.0017	385	80	-	1.921e-06	1	464	+	0.0175	348	117	-
F30	0.8612	241	224	=	0.1359	305	160	=	0.0545	139	326	=	2.370e-05	438	27	-	0.1470	303	162	=	0.0018	384	81	-
+/=/-	25/4/1				18/8/4				24/4/2				8/6/16				22/6/2				8/17/5			

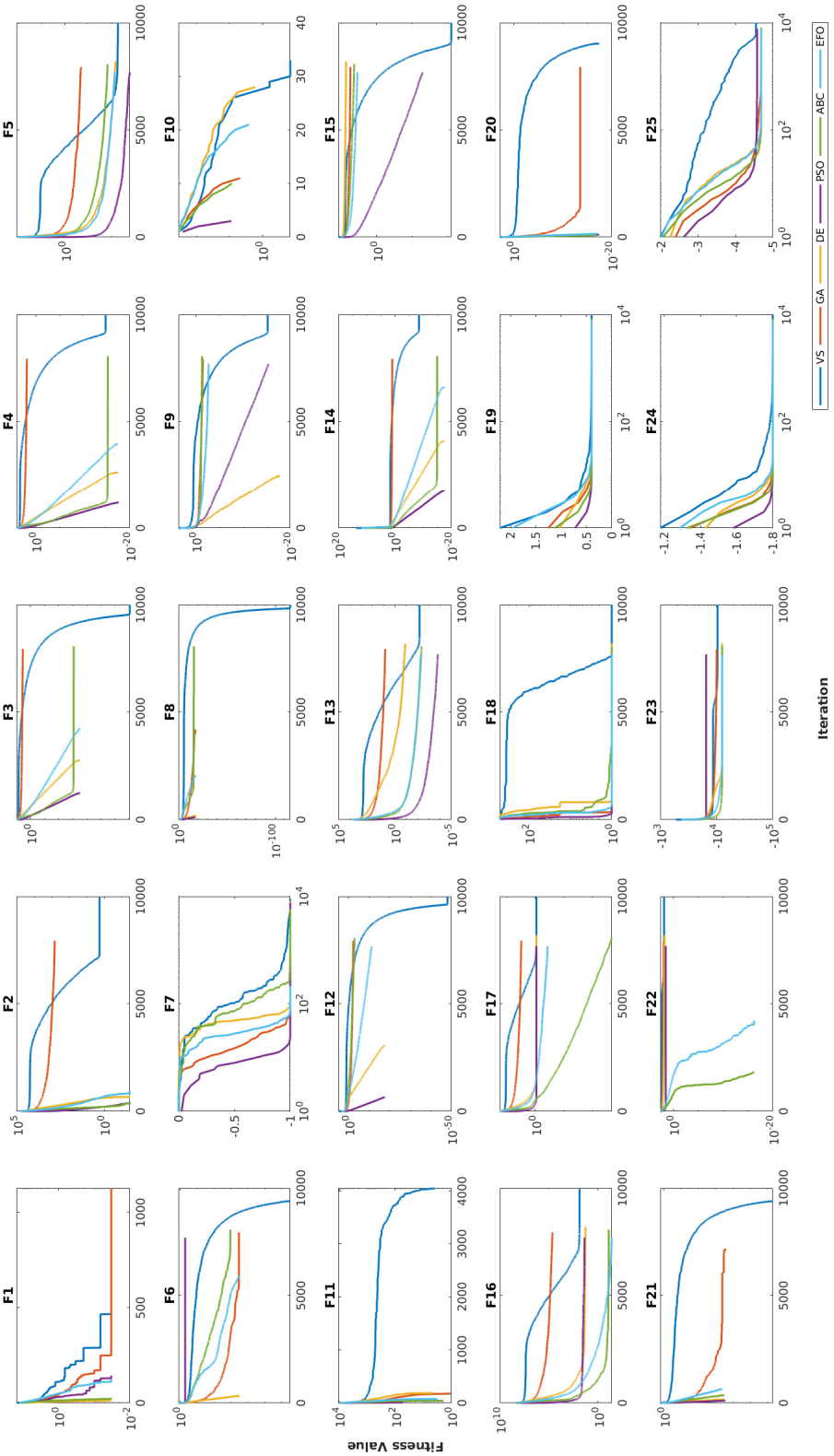

Figure C.1. Comparative convergence characteristics of EFO for basic mathematical benchmark functions 1 to 25.

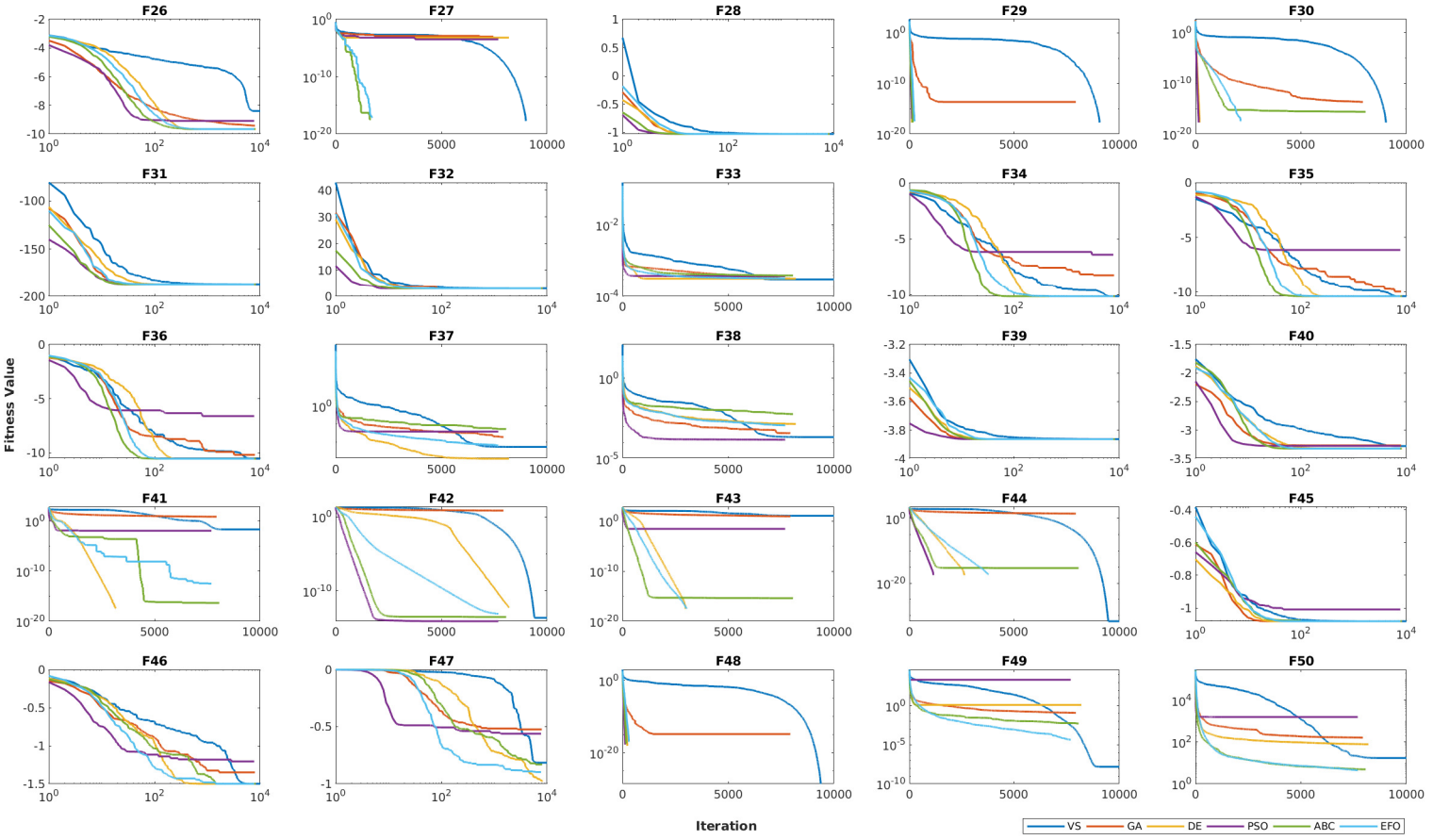

Figure C.2. Comparative convergence characteristics of EFO for basic mathematical benchmark functions 26 to 50.

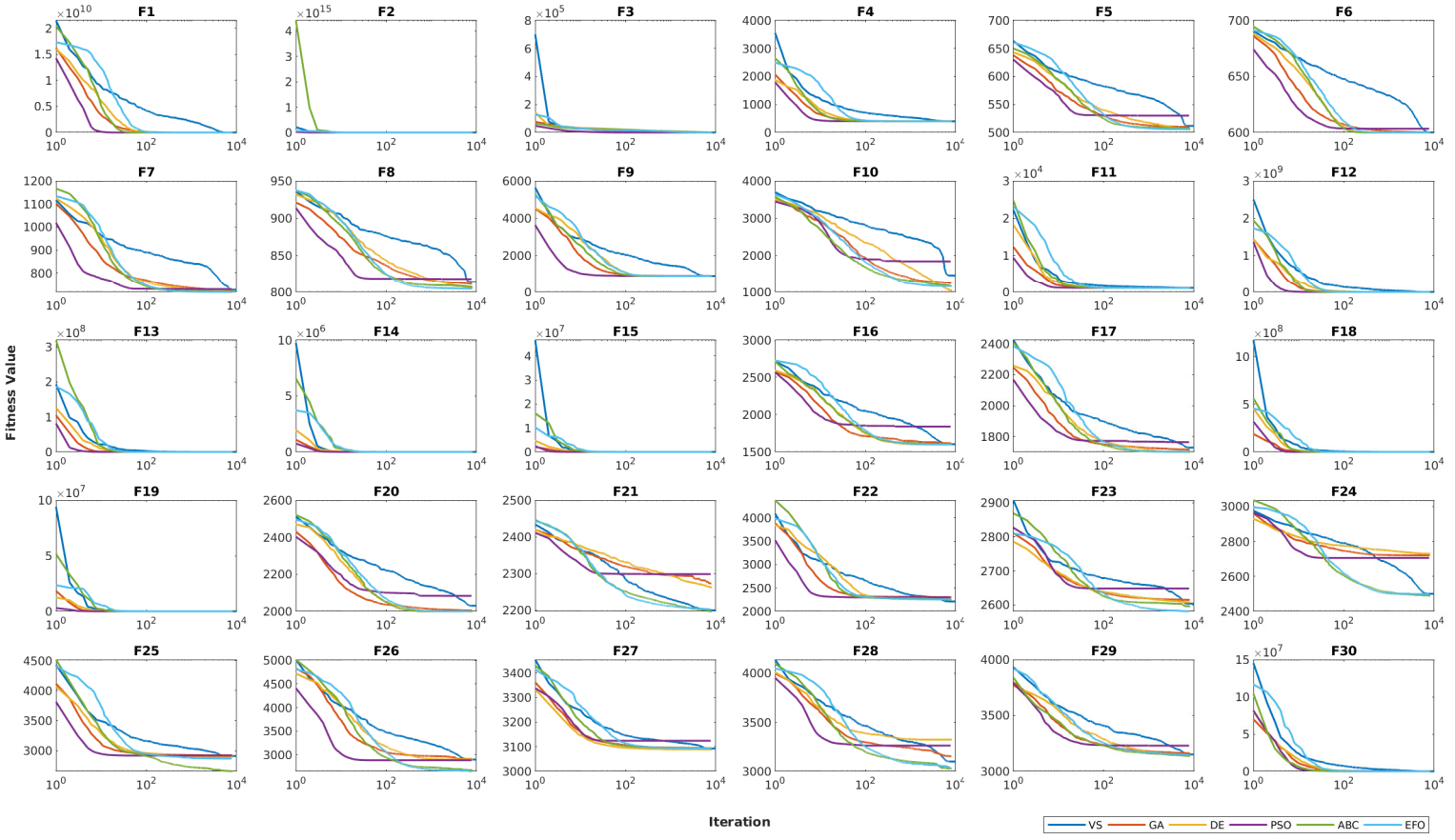

Figure C.3. Comparative convergence characteristics of EFO for complex mathematical benchmark functions.

APPENDIX D – Feature Set Used in Botnet Detection

Table D.1. Feature set used for botnet detection.

#	Feature	Description
1	**Flow_Duration**	*Flow duration*
2	**Tot_Fwd_Pkts**	*Total # of forward packets*
3	**Tot_Bwd_Pkts**	*Total # of backwark packets*
4	**TotLen_Fwd_Pkts**	*Total length of forward packets*
5	**TotLen_Bwd_Pkts**	*Total length of backward packets*
6	**Fwd_Pkt_Len_Max**	*Max. of the lengths of forward packets*
7	**Fwd_Pkt_Len_Min**	*Min. of the lengths of forward packets*
8	**Fwd_Pkt_Len_Mean**	*Mean of the lengths of forward packets*
9	**Fwd_Pkt_Len_Std**	*Std. Dev. of the lengths of forward packets*
10	**Bwd_Pkt_Len_Max**	*Max. of the lengths of backward packets*
11	**Bwd_Pkt_Len_Min**	*Min. of the lengths of backward packets*
12	**Bwd_Pkt_Len_Mean**	*Mean of the lengths of backward packets*
13	**Bwd_Pkt_Len_Std**	*Std. Dev. of the lengths of backward packets*
14	**Flow_Byts_s**	*Size of the packets (in byte) in the flow per second*
15	**Flow_Pkts_s**	*# of packets in the flow per second*
16	**Flow_IAT_Mean**	*Mean of the intervals among the packets*
17	**Flow_IAT_Std**	*Std. Dev. Of the intervals among the packets*
18	**Flow_IAT_Max**	*Max. of the intervals among the packets*
19	**Flow_IAT_Min**	*Min. of the intervals among the packets*
20	**Fwd_IAT_Tot**	*Sum of the intervals among the forward packets*
21	**Fwd_IAT_Mean**	*Mean. of the intervals among the forward packets*
22	**Fwd_IAT_Std**	*Std. Dev. of the intervals among the forward packets*
23	**Fwd_IAT_Max**	*Max. of the intervals among the forward packets*
24	**Fwd_IAT_Min**	*Min. of the intervals among the forward packets*
25	**Bwd_IAT_Tot**	*Sum of the intervals among the backward packets*
26	**Bwd_IAT_Mean**	*Mean. of the intervals among the backward packets*
27	**Bwd_IAT_Std**	*Std. Dev. of the intervals among the backward packets*
28	**Bwd_IAT_Max**	*Max. of the intervals among the backward packets*
29	**Bwd_IAT_Min**	*Min. of the intervals among the backward packets*

Table D.1. (continued).

#	Feature	Description
30	**Fwd_PSH_Flags**	*# of PSH flags in forward packets*
31	**Bwd_PSH_Flags**	*# of PSH flags in backward packets*
32	**Fwd_URG_Flags**	*# of URG flags in forward packets*
33	**Bwd_URG_Flags**	*# of URG flags in backward packets*
34	**Fwd_Header_Len**	*Sum of the header len. of the forward packets*
35	**Bwd_Header_Len**	*Sum of the header len. of the backward packets*
36	**Fwd_Pkts_s**	*# of forward packets per second*
37	**Bwd_Pkts_s**	*# of backward packets per second*
38	**Pkt_Len_Min**	*Min. of the packet lengths in the flow*
39	**Pkt_Len_Max**	*Max. of the packet lengths in the flow*
40	**Pkt_Len_Mean**	*Mean of the packet lengths in the flow*
41	**Pkt_Len_Std**	*Std. Dev. Of the packet lengths in the flow*
42	**Pkt_Len_Var**	*Variance of the packet lengths in the flow*
43	**FIN_Flag_Cnt**	*# of FIN flags in the flow*
44	**SYN_Flag_Cnt**	*# of SYN flags in the flow*
45	**RST_Flag_Cnt**	*# of RST flags in the flow*
46	**PSH_Flag_Cnt**	*# of PSH flags in the flow*
47	**ACK_Flag_Cnt**	*# of ACK flags in the flow*
48	**URG_Flag_Cnt**	*# of URG flags in the flow*
49	**CWE_Flag_Count**	*# of CWE flags in the flow*
50	**ECE_Flag_Cnt**	*# of ECE flags in the flow*
51	**Down_Up_Ratio**	*Ratio of the # of the fwd. pcks to the # bwd.pcks*
52	**Pkt_Size_Avg**	*Average size of the packets in the flow*
53	**Fwd_Seg_Size_Avg**	*Average size of the forward packets*
54	**Bwd_Seg_Size_Avg**	*Average size of the backward packets*
55	**Fwd_Byts_b_Avg**	*Average size per bulk of the forward packets*
56	**Fwd_Pkts_b_Avg**	*Average of total forward packets per bulk*
57	**Fwd_Blk_Rate_Avg**	*Average bulk rate of the forward packets*
58	**Bwd_Byts_b_Avg**	*Average size per bulk of the backward packets*
59	**Bwd_Pkts_b_Avg**	*Average of total backward packets per bulk*

Table D.1. (continued).

#	Feature	Description
60	**Bwd_Blk_Rate_Avg**	*Average bulk rate of the backward packets*
61	**Subflow_Fwd_Pkts**	*# of subflows in forward direction*
62	**Subflow_Fwd_Byts**	*Size of the subflows in forward direction*
63	**Subflow_Bwd_Pkts**	*# of subflows in backward direction*
64	**Subflow_Bwd_Byts**	*Size of the subflows in backward direction*
65	**Init_Fwd_Win_Byts**	*Size of initial TCP window of forward packet*
66	**Init_Bwd_Win_Byts**	*Size of initial TCP window of backward packet*
67	**Fwd_Act_Data_Pkts**	*Size of the ACK'ed forward packets*
68	**Fwd_Seg_Size_Min**	*Min. segment size of forward packets*
69	**Active_Mean**	*Mean active times of the flow*
70	**Active_Std**	*Std. Dev. Of Active times of the flow*
71	**Active_Max**	*Max. of active times of the flow*
72	**Active_Min**	*Min. of active times of the flow*
73	**Idle_Mean**	*Mean idle times of the flow*
74	**Idle_Std**	*Std. Dev. Of idle times of the flow*
75	**Idle_Max**	*Max. of idle times of the flow*
76	**Idle_Min**	*Min. of idle times of the flow*
77	**Fwd_1st_Payload**	*Size of the first forward packet*
78	**Bwd_1st_Payload**	*Size of the first backward packet*

APPENDIX E – Definitions of Design Problems

- **Welded-beam Problem:**

$$
\begin{aligned}
&minimize \quad f(\boldsymbol{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)\\
&\\
&subject\ to \quad g_1(\boldsymbol{x}) = \tau(\boldsymbol{x}) - \tau_{max} \leq 0\\
&\qquad\qquad g_2(\boldsymbol{x}) = \sigma(\boldsymbol{x}) - \sigma_{max} \leq 0\\
&\qquad\qquad g_3(\boldsymbol{x}) = x_1 - x_4 \leq 0\\
&\qquad\qquad g_4(\boldsymbol{x}) = 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0\\
&\qquad\qquad g_5(\boldsymbol{x}) = 0.125 - x_1 \leq 0\\
&\qquad\qquad g_6(\boldsymbol{x}) = \delta(\boldsymbol{x}) - \delta max \leq 0\\
&\qquad\qquad g_7(\boldsymbol{x}) = P - P_c(\boldsymbol{x}) \leq 0\\
&\\
&where \quad \tau(\boldsymbol{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}\\
&\qquad\qquad \tau' = \frac{P}{\sqrt{2}x_1x_2}\\
&\qquad\qquad \tau'' = \frac{MR}{J}\\
&\qquad\qquad M = P\left(L + \frac{x_2}{2}\right)\\
&\qquad\qquad R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2}\\
&\qquad\qquad J = 2\{\frac{x_1x_2}{\sqrt{2}}\left[\frac{x_2^2}{12} + \left(\frac{x_1+x_3}{2}\right)^2\right]\}\\
&\qquad\qquad \sigma(\boldsymbol{x}) = \frac{6PL}{x_4x_3^2}\\
&\qquad\qquad \delta(\boldsymbol{x}) = \frac{4PL^3}{Ex_3^3x_4}\\
&\qquad\qquad P_c(\boldsymbol{x}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)\\
&\qquad\qquad P = 6000lb,\ L = 14in,\ \delta max = 0.25in, E = 30 \times 10^6psi\\
&\qquad\qquad G = 12 \times 10^6psi, \tau_{max} = 13600psi,\ \sigma_{max} = 30000psi.
\end{aligned}
$$

- **Spring Design Problem:**

$$
\begin{aligned}
&minimize \quad f(\boldsymbol{x}) = (x_3 + 2)x_2x_1^2 \\
\\
&subject\ to \quad g_1(\boldsymbol{x}) = 1 - \tfrac{x_2^3 x_3}{71785x_1^4} \leq 0 \\
&\qquad\qquad g_2(\boldsymbol{x}) = \tfrac{4x_2^2 - x_1x_2}{12566(d_2d_1^3 - x_1^4)} + \tfrac{1}{5108x_1^2} - 1 \leq 0 \\
&\qquad\qquad g_3(\boldsymbol{x}) = 1 - \tfrac{140.45x_1}{x_2^2x_3} \leq 0 \\
&\qquad\qquad g_4(\boldsymbol{x}) = \tfrac{x_2 + x_1}{1.5} - 1 \leq 0
\end{aligned}
$$

- **Pressure Vessel Problem:**

$$
\begin{aligned}
&minimize \quad f(\boldsymbol{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \\
\\
&subject\ to \quad g_1(\boldsymbol{x}) = -x_1 + 0.0193x_3 \leq 0 \\
&\qquad\qquad g_2(\boldsymbol{x}) = -x_2 + 0.00954 \leq 0 \\
&\qquad\qquad g_3(\boldsymbol{x}) = -\pi x_3^2x_4 - \tfrac{4}{3}\pi x_3^3 + 1296000 \leq 0 \\
&\qquad\qquad g_4(\boldsymbol{x}) = x_4 - 240 \leq 0
\end{aligned}
$$

- **Three-bar Truss Design:**

$$
\begin{aligned}
&minimize \quad f(\boldsymbol{x}) = \left(2\sqrt{2}x_1 + x_2\right) L \\
\\
&subject\ to \quad g_1(\boldsymbol{x}) = \tfrac{\sqrt{2}x_1 + x_1}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leq 0 \\
&\qquad\qquad g_2(\boldsymbol{x}) = \tfrac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leq 0 \\
&\qquad\qquad g_3(\boldsymbol{x}) = \tfrac{1}{x_1 + \sqrt{2}x_2} P - \sigma \leq 0 \\
\\
&where \qquad L = 100cm,\ P = 2KN/cm^2,\ \sigma = 2KN/cm^2
\end{aligned}
$$

- **Speed Reducer:**

$$
\begin{aligned}
\textit{minimize} \quad & f(\boldsymbol{x}) = 0.7854 x_1 x_2^2 (3.3333 x_3^2 + 14.9334 x_3 - 43.0934) \\
& \qquad -1.508 x_1 (x_6^2 + x_7^2) + 7.4777 (x_6^3 + x_7^3)
\end{aligned}
$$

$$
\begin{aligned}
\textit{subject to} \quad & g_1(\boldsymbol{x}) = \frac{27}{x_1 x_2^2 x_3} - 1 \leq 0 \\
& g_2(\boldsymbol{x}) = \frac{397.5}{x_1 x_2^2} - 1 \leq 0 \\
& g_3(\boldsymbol{x}) = \frac{1.93 x_4^3}{x_2 x_3 x_6^4} - 1 \leq 0 \\
& g_4(\boldsymbol{x}) = \frac{1.93 x_5^3}{x_2 x_3 x_7^4} - 1 \leq 0 \\
& g_5(\boldsymbol{x}) = \frac{\left(\left(\frac{745 x_4}{x_2 x_3} \right)^2 + 16.9 \times 10^6 \right)^{1/2}}{110.0 x_6^3} - 1 \leq 0 \\
& g_6(\boldsymbol{x}) = \frac{\left(\left(\frac{745 x_4}{x_2 x_3} \right)^2 + 157.5 \times 10^6 \right)^{1/2}}{85.0 x_7^3} - 1 \leq 0 \\
& g_7(\boldsymbol{x}) = \frac{x_2 x_3}{40} - 1 \leq 0 \\
& g_8(\boldsymbol{x}) = \frac{5 x_2}{x_1} - 1 \leq 0 \\
& g_9(\boldsymbol{x}) = \frac{x_1}{12 x_2} - 1 \leq 0 \\
& g_{10}(\boldsymbol{x}) = \frac{1.5 x_6 + 1.9}{x_4} - 1 \leq 0 \\
& g_{11}(\boldsymbol{x}) = \frac{1.1 x_7 + 1.9}{x_5} - 1 \leq 0
\end{aligned}
$$

APPENDIX F – Definitions of Multi-objective Problems

- **DTLZ1:**

$$
\begin{aligned}
&minimize \quad f_1(\boldsymbol{x}) = \tfrac{1}{2} x_1 x_2 \cdots x_{M-1} \left(1 + g(\boldsymbol{x_M})\right), \\
&minimize \quad f_2(\boldsymbol{x}) = \tfrac{1}{2} x_1 x_2 \cdots (1 - x_{M-1}) \left(1 + g(\boldsymbol{x_M})\right), \\
&\qquad\vdots \\
&minimize \quad f_{M-1}(\boldsymbol{x}) = \tfrac{1}{2} x_1 (1 - x_2) \left(1 + g(\boldsymbol{x_M})\right), \\
&minimize \quad f_M(\boldsymbol{x}) = \tfrac{1}{2} (1 - x_1) \left(1 + g(\boldsymbol{x_M})\right), \\
&subject\ to \quad 0 \leq x_i \leq 1, \ \ i = \{1, 2, \cdots, n\}, \ n = M + |\boldsymbol{x_M}| - 1, \\
&\qquad\qquad g(\boldsymbol{x_M}) = 100 \left(|\boldsymbol{x_M}| + \sum_{x_i \in x_M} (x_i - 0.5)^2 - \cos\left(20\pi (x_i - 0.5)\right) \right)
\end{aligned}
$$

- **DTLZ2:**

$$
\begin{aligned}
&minimize \quad f_1(\boldsymbol{x}) = \left(1 + g(\boldsymbol{x_M})\right) \cos\left(x_1 \pi/2\right) \cdots \cos\left(x_{M-1} \pi/2\right), \\
&minimize \quad f_2(\boldsymbol{x}) = \left(1 + g(\boldsymbol{x_M})\right) \cos\left(x_1 \pi/2\right) \cdots \sin\left(x_{M-1} \pi/2\right), \\
&\qquad\vdots \\
&minimize \quad f_M(\boldsymbol{x}) = \left(1 + g(\boldsymbol{x_M})\right) \sin\left(x_1 \pi/2\right), \\
&subject\ to \quad 0 \leq x_i \leq 1, \ \ i = \{1, 2, \cdots, n\}, \ n = M + |\boldsymbol{x_M}| - 1, \\
&\qquad\qquad g(\boldsymbol{x_M}) = \textstyle\sum_{x_i \in X_M} (x_i - 0.5)^2
\end{aligned}
$$

- **DTLZ3:**

$$
\begin{aligned}
&minimize \quad f_1(\boldsymbol{x}) = \left(1 + g(\boldsymbol{x_M})\right) \cos\left(x_1 \pi/2\right) \cdots \cos\left(x_{M-1} \pi/2\right), \\
&minimize \quad f_2(\boldsymbol{x}) = \left(1 + g(\boldsymbol{x_M})\right) \cos\left(x_1 \pi/2\right) \cdots \sin\left(x_{M-1} \pi/2\right), \\
&\qquad\vdots \\
&minimize \quad f_M(\boldsymbol{x}) = \left(1 + g(\boldsymbol{x_M})\right) \sin\left(x_1 \pi/2\right), \\
&subject\ to \quad 0 \leq x_i \leq 1, \ \ i = \{1, 2, \cdots, n\}, \ n = M + |\boldsymbol{x_M}| - 1, \\
&\qquad\qquad g(\boldsymbol{x_M}) = 100 \left(|\boldsymbol{x_M}| + \sum_{x_i \in x_M} (x_i - 0.5)^2 - \cos\left(20\pi (x_i - 0.5)\right) \right)
\end{aligned}
$$

- **DTLZ4:**

$$
\begin{aligned}
&minimize \quad f_1(\boldsymbol{x}) = \left(1 + g(\boldsymbol{x_M})\right) \cos\left(x_1^{\alpha}\pi/2\right) \cdots \cos\left(x_{M-1}^{\alpha}\pi/2\right), \\
&minimize \quad f_2(\boldsymbol{x}) = \left(1 + g(\boldsymbol{x_M})\right) \cos\left(x_1^{\alpha}\pi/2\right) \cdots \sin\left(x_{M-1}^{\alpha}\pi/2\right), \\
&\qquad\qquad \vdots \\
&minimize \quad f_M(\boldsymbol{x}) = \left(1 + g(\boldsymbol{x_M})\right) \sin\left(x_1^{\alpha}\pi/2\right), \\
&subject\ to \quad 0 \leq x_i \leq 1,\ \ i = \{1, 2, \cdots, n\},\ n = M + |\boldsymbol{x_M}| - 1,\ \alpha = 100, \\
&\qquad\qquad g(\boldsymbol{x_M}) = \textstyle\sum_{x_i \in X_M} (x_i - 0.5)^2
\end{aligned}
$$

- **DTLZ5:**

$$
\begin{aligned}
&minimize \quad f_1(\boldsymbol{x}) = \left(1 + g(\boldsymbol{x_M})\right) \cos\left(\theta_1\pi/2\right) \cdots \cos\left(\theta_{M-1}\pi/2\right), \\
&minimize \quad f_2(\boldsymbol{x}) = \left(1 + g(\boldsymbol{x_M})\right) \cos\left(\theta_1\pi/2\right) \cdots \sin\left(\theta_{M-1}\pi/2\right), \\
&\qquad\qquad \vdots \\
&minimize \quad f_M(\boldsymbol{x}) = \left(1 + g(\boldsymbol{x_M})\right) \sin\left(\theta_1\pi/2\right), \\
&subject\ to \quad 0 \leq x_i \leq 1,\ \ i = \{1, 2, \cdots, n\},\ n = M + |\boldsymbol{x_M}| - 1, \\
&\qquad\qquad g(\boldsymbol{x_M}) = \textstyle\sum_{x_i \in X_M} x_i^{0.1}, \\
&\qquad\qquad \theta_i = \frac{\pi}{4\left(1+g(r)\right)} \left(1 + 2g(r)x_i\right),\ \ i = \{2, 3, \cdots, M-1\}
\end{aligned}
$$

- **DTLZ6:**

$$
\begin{aligned}
&minimize \quad f_1(\boldsymbol{x_1}) = x_1, \\
&minimize \quad f_2(\boldsymbol{x_2}) = x_2, \\
&\qquad\qquad \vdots \\
&minimize \quad f_{M-1}\left(\boldsymbol{x_{M-1}}\right) = x_{M-1}, \\
&minimize \quad f_M\left(\boldsymbol{x_{M-1}}\right) = \left(1 + g\left(\boldsymbol{x_{M-1}}\right)\right) h\left(f_1, f_2, \cdots, f_{M-1}, g\right), \\
&subject\ to \quad 0 \leq x_i \leq 1,\ \ i = \{1, 2, \cdots, n\},\ n = M + |\boldsymbol{x_M}| - 1, \\
&\qquad\qquad g(\boldsymbol{x_M}) = 1 + \tfrac{9}{|\boldsymbol{x_M}|} \textstyle\sum_{x_i \in X_M} x_i, \\
&\qquad\qquad h = M - \textstyle\sum_{i=1}^{M-1} \left[\frac{f_i}{1+g}\left(1 + \sin\left(3\pi f_i\right)\right)\right],
\end{aligned}
$$

- **DTLZ7:**

$$
\begin{aligned}
&minimize \quad f_j(\boldsymbol{x}) = \frac{1}{\left\lceil \frac{n}{M} \right\rceil} \sum_{i=\left\lfloor (j-1)\frac{n}{M} \right\rfloor}^{\left\lfloor j\frac{n}{M} \right\rfloor} x_i, \;\; j = 1, \cdots M, \\
&subject\ to \quad g_j(\boldsymbol{x}) = f_M(\boldsymbol{x}) + 4f_j(\boldsymbol{x}) - 1 \geq 0, \;\; j = \left\{1, \cdots, (M-1)\right\}, \\
&\qquad\qquad g_M(\boldsymbol{x}) = 2f_M(\boldsymbol{x}) + \min_{i \neq j, i, j=1}^{M-1} \left[f_i(\boldsymbol{x}) + f_j(\boldsymbol{x}) \right] - 1 \geq 0, \\
&\qquad\qquad 0 \leq x_i \leq 1, \;\; i = \{1, \cdots, n\}, \; n = 10M
\end{aligned}
$$

- **LZ09$_{\mathbf{F1}}$:**

$$
\begin{aligned}
&minimize \quad f_1 = x_1 + \frac{2}{|J_1|} \sum_{j \in J_1} \left(x_j - x_1^{0.5\left(1.0 + \frac{3(j-2)}{n-2}\right)} \right)^2 \\
&minimize \quad f_2 = 1 - \sqrt{x_1} + \frac{2}{|J_2|} \sum_{j \in J_2} \left(x_j - x_1^{0.5\left(1.0 + \frac{3(j-2)}{n-2}\right)} \right)^2 \\
&\qquad\qquad J_1 = \left\{ j \mid j \; is \; odd, \; 2 \leq j \leq n \right\} \\
&\qquad\qquad J_2 = \left\{ j \mid j \; is \; even, \; 2 \leq j \leq n \right\}
\end{aligned}
$$

- **LZ09$_{\mathbf{F2}}$:**

$$
\begin{aligned}
&minimize \quad f_1 = x_1 + \frac{2}{|J_1|} \sum_{j \in J_1} \left(x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right) \right)^2 \\
&minimize \quad f_2 = 1 - \sqrt{x_1} + \frac{2}{|J_2|} \sum_{j \in J_2} \left(x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right) \right)^2 \\
&\qquad\qquad J_1 = \left\{ j \mid j \; is \; odd, \; 2 \leq j \leq n \right\} \\
&\qquad\qquad J_2 = \left\{ j \mid j \; is \; even, \; 2 \leq j \leq n \right\}
\end{aligned}
$$

- **LZ09$_{\mathbf{F3}}$:**

$$
\begin{aligned}
&minimize \quad f_1 = x_1 + \frac{2}{|J_1|} \sum_{j \in J_1} \left(x_j - 0.8x_1 \cos\left(6\pi x_1 + \frac{j\pi}{n}\right) \right)^2 \\
&minimize \quad f_2 = 1 - \sqrt{x_1} + \frac{2}{|J_2|} \sum_{j \in J_2} \left(x_j - 0.8x_1 \sin\left(6\pi x_1 + \frac{j\pi}{n}\right) \right)^2 \\
&\qquad\qquad J_1 = \left\{ j \mid j \; is \; odd, \; 2 \leq j \leq n \right\} \\
&\qquad\qquad J_2 = \left\{ j \mid j \; is \; even, \; 2 \leq j \leq n \right\}
\end{aligned}
$$

- **LZ09$_{\mathbf{F4}}$:**

$$
\begin{aligned}
minimize \quad & f_1 = x_1 + \tfrac{2}{|J_1|}\sum_{j\in J_1}\left(x_j - 0.8x_1\cos\left(\tfrac{6\pi x_1+\frac{j\pi}{n}}{3}\right)\right)^2 \\
minimize \quad & f_2 = 1-\sqrt{x_1} + \tfrac{2}{|J_2|}\sum_{j\in J_2}\left(x_j - 0.8x_1\sin\left(6\pi x_1+\tfrac{j\pi}{n}\right)\right)^2 \\
& J_1 = \left\{j \mid j\ is\ odd,\ 2\le j\le n\right\} \\
& J_2 = \left\{j \mid j\ is\ even,\ 2\le j\le n\right\}
\end{aligned}
$$

- **LZ09$_{\mathbf{F5}}$:**

$$
\begin{aligned}
minimize \quad & f_1 = x_1 + \tfrac{2}{|J_1|}\sum_{j\in J_1}\left\{x_j - \left[0.3x_1^2\cos\left(24\pi x_1+\tfrac{4j\pi}{n}\right)+0.6x_1\right]\cos\left(6\pi x_1+\tfrac{j\pi}{n}\right)\right\}^2 \\
minimize \quad & f_2 = 1-\sqrt{x_1} + \tfrac{2}{|J_2|}\sum_{j\in J_2}\left\{x_j - \left[0.3x_1^2\cos\left(24\pi x_1+\tfrac{4j\pi}{n}\right)+0.6x_1\right]\sin\left(6\pi x_1+\tfrac{j\pi}{n}\right)\right\}^2 \\
& J_1 = \left\{j \mid j\ is\ odd,\ 2\le j\le n\right\} \\
& J_2 = \left\{j \mid j\ is\ even,\ 2\le j\le n\right\}
\end{aligned}
$$

- **LZ09$_{\mathbf{F6}}$:**

$$
\begin{aligned}
minimize \quad & f_1 = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \tfrac{2}{|J_1|}\sum_{j\in J_1}\left(x_j - 2x_2\sin\left(2\pi x_1+\tfrac{j\pi}{n}\right)\right)^2 \\
minimize \quad & f_2 = \cos(0.5x_1\pi)\sin(0.5x_2\pi) + \tfrac{2}{|J_2|}\sum_{j\in J_2}\left(x_j - 2x_2\sin\left(2\pi x_1+\tfrac{j\pi}{n}\right)\right)^2 \\
minimize \quad & f_3 = \sin(0.5x_1\pi) + \tfrac{2}{|J_3|}\sum_{j\in J_3}\left(x_j - 2x_2\sin\left(2\pi x_1+\tfrac{j\pi}{n}\right)\right)^2 \\
& J_1 = \left\{j \mid 3\le j\le n,\ j-1\ is\ a\ multiplication\ of\ 3\right\} \\
& J_2 = \left\{j \mid 3\le j\le n,\ j-2\ is\ a\ multiplication\ of\ 3\right\} \\
& J_2 = \left\{j \mid 3\le j\le n, j\ is\ a\ multiplication\ of\ 3\right\}
\end{aligned}
$$

- **LZ09$_{F7}$:**

$$
\begin{aligned}
minimize \quad & f_1 = x_1 + \tfrac{2}{|J_1|} \textstyle\sum_{j \in J_1} \left(4y_j^2 - \cos\left(8y_j\pi\right) + 1.0\right) \\
minimize \quad & f_2 = 1 - \sqrt{x_1} + \tfrac{2}{|J_2|} \textstyle\sum_{j \in J_2} \left(4y_j^2 - \cos\left(8y_j\pi\right) + 1.0\right) \\
& J_1 = \left\{j \mid j \; is \; odd, \; 2 \leq j \leq n\right\} \\
& J_2 = \left\{j \mid j \; is \; even, \; 2 \leq j \leq n\right\} \\
& y_j = x_j - x_1^{0.5\left(1.0 + \frac{3(j-2)}{n-2}\right)}, \; j = 2, \ldots, n
\end{aligned}
$$

- **LZ09$_{F8}$:**

$$
\begin{aligned}
\min \quad & f_1 = x_1 + \tfrac{2}{|J_1|} \left(4 \textstyle\sum_{j \in J_1} y_j^2 - 2 \prod_{j \in J_1} \cos\left(\tfrac{20y_j\pi}{\sqrt{j}}\right) + 2\right) \\
& f_2 = 1 - \sqrt{x_1} + \tfrac{2}{|J_2|} \left(4 \textstyle\sum_{j \in J_2} y_j^2 - 2 \prod_{j \in J_2} \cos\left(\tfrac{20y_j\pi}{\sqrt{j}}\right) + 2\right) \\
& J_1 = \left\{j \mid j \; is \; odd, \; 2 \leq j \leq n\right\} \\
& J_2 = \left\{j \mid j \; is \; even, \; 2 \leq j \leq n\right\} \\
& y_j = x_j - x_1^{0.5\left(1.0 + \frac{3(j-2)}{n-2}\right)}, \; j = 2, \ldots, n
\end{aligned}
$$

- **LZ09$_{F9}$:**

$$
\begin{aligned}
minimize \quad & f_1 = x_1 + \tfrac{2}{|J_1|} \textstyle\sum_{j \in J_1} \left(x_j - \sin\left(6\pi x_1 + \tfrac{j\pi}{n}\right)\right)^2 \\
minimize \quad & f_2 = 1 - x_1^2 + \tfrac{2}{|J_2|} \textstyle\sum_{j \in J_2} \left(x_j - \sin\left(6\pi x_1 + \tfrac{j\pi}{n}\right)\right)^2 \\
& J_1 = \left\{j \mid j \; is \; odd, \; 2 \leq j \leq n\right\} \\
& J_2 = \left\{j \mid j \; is \; even, \; 2 \leq j \leq n\right\}
\end{aligned}
$$

- **ZDT1:**

$$
\begin{aligned}
minimize \quad & f_1\left(x_1\right) = x_1, \\
minimize \quad & f_2\left(\boldsymbol{x}\right) = g\left(x_2, \ldots, x_m\right) h\left(f_1\left(x_1\right), g\left(x_2, \ldots, x_m\right)\right), \\
& g\left(x_2, \ldots, x_m\right) = 1 + 9 \textstyle\sum_{i=2}^{m} \frac{x_i}{(m-1)}, \\
& h\left(f_1, g\right) = 1 - \sqrt{f_1/g}, \\
& \boldsymbol{x} = \left(x_1, \ldots, x_m\right)
\end{aligned}
$$

- **ZDT2:**

$$
\begin{aligned}
&minimize \quad f_1(x_1) = x_1, \\
&minimize \quad f_2(\boldsymbol{x}) = g(x_2, \ldots, x_m)\, h\left(f_1(x_1), g(x_2, \ldots, x_m)\right), \\
&\qquad\quad g(x_2, \ldots, x_m) = 1 + 9 \textstyle\sum_{i=2}^{m} \frac{x_i}{(m-1)}, \\
&\qquad\quad h(f_1, g) = 1 - \left(\frac{f_1}{g}\right)^2, \\
&\qquad\quad \boldsymbol{x} = (x_1, \ldots, x_m)
\end{aligned}
$$

- **ZDT3:**

$$
\begin{aligned}
&minimize \quad f_1(x_1) = x_1, \\
&minimize \quad f_2(\boldsymbol{x}) = g(x_2, \ldots, x_m)\, h\left(f_1(x_1), g(x_2, \ldots, x_m)\right), \\
&\qquad\quad g(x_2, \ldots, x_m) = 1 + 9 \textstyle\sum_{i=2}^{m} \frac{x_i}{(m-1)}, \\
&\qquad\quad h(f_1, g) = 1 - \sqrt{f_1/g} - \left(\frac{f_1}{g}\right) \sin(10\pi f_1), \\
&\qquad\quad \boldsymbol{x} = (x_1, \ldots, x_m)
\end{aligned}
$$

- **ZDT4:**

$$
\begin{aligned}
&minimize \quad f_1(x_1) = x_1, \\
&minimize \quad f_2(\boldsymbol{x}) = g(x_2, \ldots, x_m)\, h\left(f_1(x_1), g(x_2, \ldots, x_m)\right), \\
&\qquad\quad g(x_2, \ldots, x_m) = 1 + 10(m-1) + \textstyle\sum_{i=2}^{m} \left(x_i^2 - 10\cos(4\pi x_i)\right), \\
&\qquad\quad h(f_1, g) = 1 - \sqrt{f_1/g}, \\
&\qquad\quad \boldsymbol{x} = (x_1, \ldots, x_m)
\end{aligned}
$$

- **ZDT6:**

$$
\begin{aligned}
&minimize \quad f_1(x_1) = 1 - exp(-4x_1) \sin^6(6\pi x_1), \\
&minimize \quad f_2(\boldsymbol{x}) = g(x_2, \ldots, x_m)\, h\left(f_1(x_1), g(x_2, \ldots, x_m)\right), \\
&\qquad\quad g(x_2, \ldots, x_m) = 1 + 9\left(\left(\textstyle\sum_{i=2}^{m} x_i\right)/(m-1)\right)^{0.25}, \\
&\qquad\quad h(f_1, g) = 1 - \left(\frac{f_1}{g}\right)^2, \\
&\qquad\quad \boldsymbol{x} = (x_1, \ldots, x_m)
\end{aligned}
$$

APPENDIX G – Experimental Results on Multi-objective Optimization Problems

Table G.1. Mean and standard deviation values of HV metric obtained by PD-EFO, NS-EFO, and CB-EFO.

	Algorithms					
	PD-EFO		NS-EFO		CB-EFO	
Problem	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**
DTLZ1	4.23e-01	8.8e-02	7.27e-01	1.6e-01	4.35e-01	3.5e-01
DTLZ2	3.90e-01	5.7e-03	3.78e-01	5.4e-03	3.78e-01	4.7e-03
DTLZ3	0.00e+00	0.0e+00	7.96e-02	1.4e-01	2.58e-04	1.8e-03
DTLZ4	1.03e-01	5.1e-02	3.76e-01	6.5e-03	3.80e-01	4.9e-03
DTLZ5	8.86e-02	1.7e-03	9.30e-02	1.5e-04	9.40e-02	2.9e-05
DTLZ6	7.99e-02	5.5e-03	9.37e-02	2.1e-04	9.50e-02	2.7e-05
DTLZ7	2.81e-01	6.6e-03	2.85e-01	4.3e-03	2.91e-01	3.4e-03
LZ09$_{F1}$	6.46e-01	4.2e-03	6.60e-01	2.8e-04	6.59e-01	2.7e-04
LZ09$_{F2}$	6.40e-01	3.9e-03	5.95e-01	2.8e-02	5.68e-01	2.9e-02
LZ09$_{F3}$	6.46e-01	3.1e-03	6.23e-01	7.8e-03	6.26e-01	8.0e-03
LZ09$_{F4}$	6.40e-01	3.4e-03	6.33e-01	4.9e-03	6.28e-01	5.5e-03
LZ09$_{F5}$	6.48e-01	2.3e-03	6.31e-01	6.5e-03	6.36e-01	5.7e-03
LZ09$_{F6}$	3.72e-01	1.0e-02	2.53e-01	2.3e-02	2.33e-01	5.9e-02
LZ09$_{F7}$	5.86e-01	1.6e-02	4.52e-01	5.4e-02	3.86e-01	7.4e-02
LZ09$_{F8}$	5.13e-01	2.1e-02	4.24e-01	5.3e-02	3.72e-01	6.5e-02
LZ09$_{F9}$	3.06e-01	3.5e-03	2.69e-01	2.2e-02	2.45e-01	2.6e-02
WFG1	1.12e-01	1.7e-02	6.28e-01	2.5e-02	6.31e-01	1.2e-03
WFG2	5.27e-01	8.7e-03	5.64e-01	1.5e-04	5.65e-01	6.4e-05
WFG3	3.95e-01	9.3e-03	4.42e-01	1.4e-04	4.42e-01	2.9e-05
WFG4	2.10e-01	2.8e-03	2.17e-01	4.2e-04	2.19e-01	3.3e-04
WFG5	1.93e-01	8.1e-04	2.02e-01	6.1e-03	1.97e-01	1.2e-03
WFG6	1.67e-01	4.3e-03	2.03e-01	7.7e-03	2.03e-01	9.6e-03
WFG7	2.00e-01	2.1e-03	2.10e-01	2.5e-04	2.11e-01	2.4e-05

Table G.1. (continued.)

	Algorithms					
	PD-EFO		NS-EFO		CB-EFO	
Problem	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**
WFG8	1.59e-01	1.0e-02	1.61e-01	2.1e-02	1.56e-01	1.7e-02
WFG9	2.05e-01	5.4e-03	2.38e-01	1.5e-03	2.39e-01	1.5e-03
ZDT1	6.55e-01	1.4e-03	6.61e-01	2.7e-04	6.62e-01	4.7e-05
ZDT2	1.83e-01	5.4e-02	3.27e-01	2.1e-04	3.29e-01	3.1e-05
ZDT3	5.12e-01	8.5e-04	5.15e-01	8.1e-05	5.16e-01	3.3e-05
ZDT4	3.43e-01	8.2e-02	6.56e-01	8.5e-03	5.20e-01	1.4e-01
ZDT6	3.72e-01	9.3e-03	3.99e-01	6.4e-04	4.01e-01	3.9e-05

Table G.2. Mean and standard deviation values of SPREAD metric obtained by PD-EFO, NS-EFO, and CB-EFO.

	Algorithms					
	PD-EFO		NS-EFO		CB-EFO	
Problem	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**
DTLZ1	7.76e-01	2.7e-01	7.95e-01	4.1e-02	7.51e-01	4.8e-02
DTLZ2	7.16e-01	5.7e-02	6.45e-01	3.3e-02	6.25e-01	3.7e-02
DTLZ3	8.98e-01	1.6e-01	6.63e-01	7.8e-02	6.87e-01	7.6e-02
DTLZ4	7.12e-01	2.3e-01	6.56e-01	4.3e-02	6.28e-01	3.5e-02
DTLZ5	7.42e-01	5.5e-02	4.74e-01	4.1e-02	1.60e-01	4.9e-02
DTLZ6	8.69e-01	6.9e-02	6.31e-01	4.2e-02	1.03e-01	1.4e-02
DTLZ7	9.55e-01	5.4e-02	7.51e-01	4.2e-02	6.86e-01	3.4e-02
$LZ09_{F1}$	8.89e-01	5.2e-02	2.62e-01	2.7e-02	1.75e-01	1.4e-02
$LZ09_{F2}$	7.25e-01	8.0e-02	1.36e+00	5.6e-02	1.46e+00	7.8e-02
$LZ09_{F3}$	7.17e-01	6.3e-02	7.91e-01	7.1e-02	8.40e-01	6.1e-02
$LZ09_{F4}$	7.45e-01	5.3e-02	5.91e-01	1.5e-01	5.96e-01	9.0e-02
$LZ09_{F5}$	7.00e-01	6.2e-02	7.20e-01	7.6e-02	7.72e-01	5.6e-02

Table G.2. (continued.)

	Algorithms					
	PD-EFO		NS-EFO		CB-EFO	
Problem	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**
LZ09$_{F6}$	6.88e-01	5.9e-02	8.92e-01	1.2e-01	7.15e-01	1.0e-01
LZ09$_{F7}$	8.00e-01	1.0e-01	1.43e+00	1.7e-01	1.33e+00	2.9e-01
LZ09$_{F8}$	7.05e-01	9.5e-02	1.47e+00	1.0e-01	1.33e+00	2.4e-01
LZ09$_{F9}$	6.76e-01	8.3e-02	1.40e+00	6.6e-02	1.60e+00	1.5e-01
WFG1	1.00e+00	8.9e-03	6.39e-01	4.0e-02	2.66e-01	1.7e-02
WFG2	8.88e-01	9.9e-02	7.77e-01	9.2e-03	7.58e-01	9.6e-04
WFG3	7.27e-01	4.1e-02	5.80e-01	2.6e-02	3.60e-01	5.5e-03
WFG4	7.83e-01	4.4e-02	4.42e-01	3.9e-02	1.32e-01	1.0e-02
WFG5	8.04e-01	4.7e-02	4.97e-01	4.2e-02	1.40e-01	1.8e-02
WFG6	6.13e-01	6.3e-02	3.79e-01	3.1e-02	1.14e-01	1.8e-02
WFG7	7.64e-01	6.7e-02	4.05e-01	2.5e-02	1.17e-01	1.5e-02
WFG8	9.67e-01	6.3e-02	5.92e-01	5.6e-02	6.27e-01	4.2e-02
WFG9	6.31e-01	7.5e-02	4.29e-01	3.0e-02	1.57e-01	2.0e-02
ZDT1	7.79e-01	5.5e-02	4.06e-01	2.7e-02	7.62e-02	1.0e-02
ZDT2	7.03e-01	2.1e-01	3.97e-01	3.0e-02	7.59e-02	1.3e-02
ZDT3	1.09e+00	5.3e-02	7.45e-01	1.3e-02	7.04e-01	1.9e-03
ZDT4	6.48e-01	1.5e-01	4.08e-01	1.1e-01	3.56e-01	2.6e-01
ZDT6	1.34e+00	8.5e-02	6.55e-01	1.2e-01	1.51e-01	3.2e-01

Table G.3. Mean and standard deviation values of EPSILON metric obtained by PD-EFO, NS-EFO, and CB-EFO.

	Algorithms					
	PD-EFO		NS-EFO		CB-EFO	
Problem	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**
DTLZ1	2.10e-01	4.9e-02	6.22e-02	4.9e-02	1.64e-01	1.3e-01
DTLZ2	1.36e-01	4.0e-02	1.26e-01	1.5e-02	1.24e-01	1.5e-02
DTLZ3	3.37e+00	8.5e-01	1.11e+00	8.7e-01	2.69e+00	1.1e+00
DTLZ4	7.07e-01	1.6e-01	1.11e-01	2.0e-02	1.12e-01	1.7e-02
DTLZ5	2.50e-02	6.1e-03	9.86e-03	1.4e-03	4.81e-03	5.6e-04
DTLZ6	4.36e-02	1.5e-02	1.05e-02	2.6e-03	4.11e-03	1.5e-04
DTLZ7	2.41e-01	1.5e-01	1.60e-01	4.6e-02	1.32e-01	3.9e-02
LZ09$_{F1}$	8.97e-02	2.9e-02	1.07e-02	8.8e-04	1.07e-02	1.1e-03
LZ09$_{F2}$	5.20e-02	1.2e-02	1.13e-01	3.7e-02	1.45e-01	3.8e-02
LZ09$_{F3}$	4.79e-02	1.3e-02	1.05e-01	1.8e-02	9.89e-02	2.4e-02
LZ09$_{F4}$	9.47e-02	2.7e-02	1.41e-01	3.2e-02	1.67e-01	2.8e-02
LZ09$_{F5}$	4.07e-02	9.9e-03	8.31e-02	2.2e-02	7.40e-02	1.8e-02
LZ09$_{F6}$	2.39e-01	8.7e-02	2.67e-01	2.1e-02	3.50e-01	1.8e-01
LZ09$_{F7}$	1.80e-01	5.9e-02	3.51e-01	1.0e-01	4.40e-01	1.1e-01
LZ09$_{F8}$	2.61e-01	7.1e-02	3.50e-01	1.1e-01	4.38e-01	1.1e-01
LZ09$_{F9}$	5.56e-02	1.4e-02	1.36e-01	7.0e-02	1.52e-01	4.7e-02
WFG1	1.77e+00	3.6e-02	4.33e-02	3.7e-02	5.98e-02	1.3e-02
WFG2	7.95e-01	1.2e-01	1.33e-02	2.2e-03	9.31e-03	3.6e-04
WFG3	2.43e+00	1.5e-01	2.00e+00	1.5e-04	2.00e+00	5.6e-05
WFG4	1.03e+00	3.3e-01	3.33e-02	5.4e-03	1.44e-02	5.2e-04
WFG5	3.06e-01	1.4e-01	7.56e-02	1.3e-02	6.31e-02	1.7e-03
WFG6	3.54e-01	1.4e-01	3.79e-02	1.3e-02	2.54e-02	1.5e-02
WFG7	4.18e-01	2.5e-01	3.54e-02	8.8e-03	1.40e-02	4.5e-04
WFG8	6.13e-01	2.2e-01	4.09e-01	1.2e-01	4.20e-01	1.1e-01

Table G.3. (continued.)

	Algorithms					
	PD-EFO		NS-EFO		CB-EFO	
Problem	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**
WFG9	1.58e-01	4.4e-02	3.64e-02	7.0e-03	1.88e-02	2.4e-03
ZDT1	3.05e-02	6.4e-03	1.27e-02	2.5e-03	5.55e-03	1.9e-04
ZDT2	3.58e-01	1.8e-01	1.29e-02	1.8e-03	5.42e-03	1.7e-04
ZDT3	2.03e-02	1.5e-02	8.21e-03	1.9e-03	5.14e-03	3.2e-04
ZDT4	4.13e-01	1.3e-01	1.42e-02	5.5e-03	1.25e-01	1.4e-01
ZDT6	7.03e-02	3.5e-02	1.20e-02	2.1e-03	4.51e-03	1.5e-04

Table G.4. Mean and standard deviation values of IGD metric obtained by PD-EFO, NS-EFO, and CB-EFO.

	Algorithms					
	PD-EFO		NS-EFO		CB-EFO	
Problem	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**
DTLZ1	3.03e-03	7.8e-04	8.75e-04	1.3e-03	3.58e-03	3.5e-03
DTLZ2	9.10e-04	5.8e-05	7.36e-04	3.4e-05	7.04e-04	2.6e-05
DTLZ3	5.07e-02	1.4e-02	2.17e-02	1.9e-02	5.72e-02	2.6e-02
DTLZ4	5.51e-03	1.8e-03	1.21e-03	1.4e-04	1.21e-03	1.1e-04
DTLZ5	5.09e-05	1.3e-05	1.97e-05	7.0e-07	1.44e-05	3.6e-07
DTLZ6	3.44e-04	1.3e-04	5.58e-05	3.9e-06	3.38e-05	1.1e-06
DTLZ7	4.17e-03	7.1e-04	2.74e-03	3.4e-04	2.31e-03	1.4e-04
LZ09$_{F1}$	2.26e-03	9.0e-04	2.43e-04	8.8e-06	2.43e-04	8.3e-06
LZ09$_{F2}$	1.06e-03	2.2e-04	2.93e-03	1.1e-03	4.27e-03	1.4e-03
LZ09$_{F3}$	9.99e-04	2.6e-04	2.61e-03	6.2e-04	2.62e-03	8.1e-04
LZ09$_{F4}$	2.20e-03	8.4e-04	3.62e-03	1.3e-03	4.62e-03	1.1e-03
LZ09$_{F5}$	8.68e-04	2.0e-04	1.96e-03	5.8e-04	1.76e-03	5.3e-04
LZ09$_{F6}$	3.57e-03	5.3e-04	5.20e-03	7.0e-04	6.56e-03	2.9e-03

Table G.4. (continued.)

	Algorithms					
	PD-EFO		NS-EFO		CB-EFO	
Problem	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**
LZ09$_{F7}$	4.57e-03	1.6e-03	1.12e-02	4.2e-03	1.52e-02	4.2e-03
LZ09$_{F8}$	7.03e-03	2.5e-03	1.00e-02	4.1e-03	1.47e-02	4.7e-03
LZ09$_{F9}$	1.05e-03	1.7e-04	2.91e-03	1.1e-03	4.05e-03	1.3e-03
WFG1	2.65e-02	7.3e-04	2.93e-04	5.0e-04	3.59e-04	4.1e-05
WFG2	7.93e-03	1.1e-03	3.54e-04	2.5e-05	2.95e-04	8.7e-06
WFG3	3.68e-03	3.4e-04	3.01e-03	3.1e-07	3.01e-03	7.9e-08
WFG4	6.45e-04	2.2e-04	1.77e-04	1.1e-05	1.24e-04	2.0e-06
WFG5	1.11e-03	2.0e-04	7.58e-04	1.6e-04	9.32e-04	3.5e-05
WFG6	2.31e-03	3.9e-04	4.34e-04	2.2e-04	4.24e-04	3.0e-04
WFG7	3.55e-04	5.6e-05	1.23e-04	1.7e-05	8.46e-05	1.1e-06
WFG8	3.11e-03	5.0e-04	2.97e-03	1.1e-03	3.19e-03	8.9e-04
WFG9	6.75e-04	1.3e-04	1.34e-04	9.8e-06	1.02e-04	1.0e-05
ZDT1	4.32e-04	7.2e-05	1.86e-04	1.2e-05	1.35e-04	8.3e-07
ZDT2	6.53e-03	3.0e-03	1.89e-04	8.1e-06	1.41e-04	1.9e-06
ZDT3	4.60e-04	1.4e-04	1.29e-04	7.3e-06	9.99e-05	1.3e-06
ZDT4	9.37e-03	3.3e-03	2.42e-04	1.6e-04	3.57e-03	3.8e-03
ZDT6	1.18e-03	3.9e-04	2.21e-04	2.0e-05	1.33e-04	7.7e-07

Table G.5. Mean and standard deviation values of HV metric obtained by CB-EFO and the competitors.

	Algorithms											
	NSGA-II		SPEA2		IBEA		MOEAD		MOCell		CB-EFO	
Problem	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**
DTLZ1	7.62e-01	4.6e-03	7.87e-01	8.7e-04	1.92e-01	7.3e-02	7.54e-01	7.1e-04	7.61e-01	9.9e-03	4.33e-01	3.6e-01
DTLZ2	3.75e-01	6.1e-03	4.05e-01	1.8e-03	4.21e-01	3.3e-04	3.72e-01	6.7e-04	3.79e-01	6.3e-03	3.77e-01	4.7e-03
DTLZ3	3.73e-01	9.7e-03	4.09e-01	5.0e-03	3.86e-04	1.2e-03	3.66e-01	4.0e-03	3.78e-01	8.6e-03	0.00e+00	0.0e+00
DTLZ4	3.75e-01	4.9e-03	3.30e-01	1.0e-01	2.56e-01	1.2e-01	3.55e-01	2.8e-02	2.90e-01	1.6e-01	3.78e-01	4.9e-03
DTLZ5	9.29e-02	1.7e-04	9.33e-02	1.5e-04	9.31e-02	1.1e-04	9.04e-02	2.6e-05	9.41e-02	2.7e-05	9.40e-02	2.8e-05
DTLZ6	8.29e-02	1.2e-02	9.07e-02	7.0e-03	7.12e-02	1.4e-02	9.13e-02	5.5e-06	8.98e-02	8.2e-03	9.50e-02	3.1e-05
DTLZ7	2.85e-01	3.9e-03	2.96e-01	7.1e-03	2.51e-01	4.7e-02	2.36e-01	8.6e-03	2.71e-01	1.8e-02	2.89e-01	3.7e-03
LZ09$_{F1}$	6.53e-01	5.9e-04	6.58e-01	3.7e-04	6.58e-01	3.9e-04	6.61e-01	4.0e-05	6.59e-01	2.8e-04	6.59e-01	3.1e-04
LZ09$_{F2}$	5.29e-01	3.2e-02	5.23e-01	3.0e-02	5.08e-01	4.0e-02	6.37e-01	1.7e-02	4.93e-01	4.3e-02	5.77e-01	2.7e-02
LZ09$_{F3}$	6.19e-01	6.0e-03	6.11e-01	8.3e-03	6.01e-01	1.5e-02	6.31e-01	2.5e-02	5.92e-01	1.9e-02	6.25e-01	8.0e-03
LZ09$_{F4}$	6.30e-01	3.5e-03	6.24e-01	3.4e-03	6.21e-01	4.8e-03	6.44e-01	3.2e-03	6.20e-01	2.0e-03	6.30e-01	5.6e-03
LZ09$_{F5}$	6.26e-01	5.7e-03	6.23e-01	6.5e-03	6.20e-01	7.9e-03	6.36e-01	1.2e-02	6.13e-01	1.2e-02	6.36e-01	5.7e-03
LZ09$_{F6}$	1.92e-01	3.9e-02	2.65e-01	2.4e-02	8.22e-02	9.6e-02	3.15e-01	2.1e-02	1.63e-01	4.5e-02	2.35e-01	5.8e-02
LZ09$_{F7}$	4.56e-01	6.8e-02	4.40e-01	5.6e-02	3.99e-01	5.5e-02	6.51e-01	6.3e-03	3.56e-01	6.9e-02	3.80e-01	5.5e-02
LZ09$_{F8}$	4.54e-01	3.6e-02	4.39e-01	4.3e-02	4.01e-01	4.3e-02	4.70e-01	8.6e-02	3.54e-01	5.8e-02	3.64e-01	5.6e-02
LZ09$_{F9}$	1.85e-01	4.8e-02	1.86e-01	4.0e-02	1.77e-01	5.5e-02	2.99e-01	1.0e-02	1.80e-01	4.6e-02	2.40e-01	3.1e-02
WFG1	5.89e-01	7.2e-02	5.54e-01	9.3e-02	5.09e-01	1.1e-01	6.27e-01	4.3e-03	6.03e-01	6.4e-02	6.31e-01	1.6e-03
WFG2	5.62e-01	1.4e-03	5.63e-01	1.5e-03	5.60e-01	7.2e-04	5.62e-01	8.6e-05	5.61e-01	9.6e-03	5.65e-01	6.6e-05
WFG3	4.41e-01	2.4e-04	4.42e-01	1.7e-04	4.42e-01	1.6e-04	4.42e-01	2.0e-05	4.42e-01	8.2e-05	4.42e-01	2.8e-05
WFG4	2.18e-01	2.0e-04	2.19e-01	1.5e-04	2.18e-01	1.0e-04	2.17e-01	3.1e-04	2.19e-01	4.2e-05	2.19e-01	3.7e-04
WFG5	1.95e-01	3.8e-04	1.96e-01	1.3e-04	1.96e-01	5.3e-05	1.95e-01	1.1e-03	1.96e-01	3.2e-05	1.97e-01	1.8e-03
WFG6	2.01e-01	9.5e-03	2.00e-01	1.0e-02	1.97e-01	1.2e-02	2.09e-01	2.5e-05	1.85e-01	2.4e-02	2.05e-01	7.1e-03
WFG7	2.09e-01	2.7e-04	2.10e-01	1.4e-04	2.10e-01	1.1e-04	2.09e-01	2.2e-05	2.11e-01	1.8e-05	2.11e-01	2.4e-05
WFG8	1.59e-01	1.8e-02	1.59e-01	1.7e-02	1.52e-01	1.2e-02	1.65e-01	2.2e-02	1.56e-01	1.3e-02	1.58e-01	1.9e-02

Table G.5. (continued.)

	Algorithms											
	NSGA-II		SPEA2		IBEA		MOEAD		MOCell		CB-EFO	
Problem	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**
WFG9	2.38e-01	1.2e-03	2.39e-01	1.2e-03	2.39e-01	1.2e-03	2.39e-01	5.8e-04	2.39e-01	2.2e-03	2.39e-01	1.8e-03
ZDT1	6.60e-01	3.1e-04	6.62e-01	9.8e-05	6.61e-01	3.0e-04	6.61e-01	2.4e-04	6.62e-01	1.8e-05	6.62e-01	5.0e-05
ZDT2	3.27e-01	2.9e-04	3.28e-01	1.0e-04	3.28e-01	7.8e-05	3.28e-01	2.5e-04	3.29e-01	1.7e-05	3.29e-01	3.9e-05
ZDT3	5.15e-01	9.9e-05	5.16e-01	7.7e-05	5.13e-01	4.9e-04	5.13e-01	1.6e-04	5.16e-01	4.9e-04	5.16e-01	3.1e-05
ZDT4	6.60e-01	3.4e-04	6.62e-01	2.3e-04	2.27e-01	7.9e-02	6.61e-01	3.1e-04	6.62e-01	1.9e-04	5.01e-01	1.6e-01
ZDT6	3.98e-01	4.8e-04	4.01e-01	2.7e-04	3.97e-01	3.4e-04	4.01e-01	4.8e-07	4.01e-01	2.4e-04	4.01e-01	3.2e-05

Table G.6. Mean and standard deviation values of SPREAD metric obtained by CB-EFO and the competitors.

	Algorithms											
	NSGA-II		SPEA2		IBEA		MOEAD		MOCell		CB-EFO	
Problem	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**
DTLZ1	8.16e-01	4.6e-02	6.75e-01	2.6e-01	1.69e+00	1.5e-01	7.83e-01	2.4e-02	8.40e-01	1.3e-01	7.46e-01	4.3e-02
DTLZ2	7.04e-01	5.9e-02	5.30e-01	3.0e-02	5.22e-01	3.5e-02	8.06e-01	2.3e-02	6.90e-01	4.2e-02	6.16e-01	4.3e-02
DTLZ3	7.42e-01	5.0e-02	6.02e-01	1.9e-01	1.33e+00	4.1e-01	8.24e-01	2.7e-02	6.86e-01	4.8e-02	6.62e-01	4.6e-02
DTLZ4	6.88e-01	4.1e-02	4.86e-01	1.4e-01	6.25e-01	1.4e-01	8.52e-01	7.5e-02	7.39e-01	1.5e-01	6.32e-01	3.8e-02
DTLZ5	4.41e-01	4.4e-02	2.20e-01	3.9e-02	5.36e-01	3.3e-02	9.18e-01	3.6e-03	1.31e-01	4.7e-02	1.59e-01	4.5e-02
DTLZ6	6.33e-01	8.5e-02	2.71e-01	1.7e-01	1.11e+00	6.7e-02	9.12e-01	6.7e-04	2.58e-01	2.4e-01	1.10e-01	1.8e-02
DTLZ7	7.39e-01	4.7e-02	6.07e-01	4.5e-02	7.54e-01	1.1e-01	1.04e+00	9.8e-02	6.87e-01	6.2e-02	6.97e-01	4.4e-02
LZ09$_{F1}$	4.62e-01	8.6e-02	2.44e-01	1.4e-01	6.43e-01	4.5e-02	2.97e-01	3.0e-02	2.56e-01	2.0e-01	1.71e-01	1.4e-02
LZ09$_{F2}$	1.43e+00	8.6e-02	1.50e+00	9.5e-02	1.47e+00	1.1e-01	6.09e-01	1.4e-01	1.39e+00	1.4e-01	1.45e+00	7.3e-02
LZ09$_{F3}$	6.59e-01	4.7e-02	6.48e-01	8.2e-02	1.11e+00	9.4e-02	5.62e-01	1.2e-01	5.51e-01	6.8e-02	8.21e-01	6.7e-02
LZ09$_{F4}$	5.45e-01	5.0e-02	4.49e-01	5.6e-02	9.91e-01	5.1e-02	8.15e-01	2.0e-01	3.82e-01	4.6e-02	5.87e-01	7.6e-02
LZ09$_{F5}$	6.07e-01	4.8e-02	5.52e-01	9.0e-02	1.07e+00	7.2e-02	5.65e-01	8.5e-02	4.77e-01	7.7e-02	7.66e-01	5.9e-02
LZ09$_{F6}$	9.38e-01	7.5e-02	6.95e-01	6.1e-02	1.40e+00	5.6e-01	8.41e-01	6.3e-02	9.12e-01	8.3e-02	6.96e-01	8.5e-02
LZ09$_{F7}$	1.41e+00	1.5e-01	1.36e+00	1.8e-01	1.12e+00	1.5e-01	8.86e-01	2.7e-01	1.30e+00	2.0e-01	1.35e+00	2.1e-01
LZ09$_{F8}$	1.50e+00	1.0e-01	1.47e+00	1.2e-01	1.23e+00	1.8e-01	1.18e+00	1.3e-01	1.34e+00	1.8e-01	1.31e+00	1.8e-01
LZ09$_{F9}$	1.62e+00	1.6e-01	1.67e+00	1.3e-01	1.65e+00	1.2e-01	5.91e-01	2.4e-01	1.51e+00	1.9e-01	1.63e+00	1.5e-01
WFG1	5.95e-01	4.0e-02	4.74e-01	1.1e-01	7.79e-01	1.9e-02	5.31e-01	7.0e-02	3.57e-01	6.8e-02	2.58e-01	2.1e-02
WFG2	7.90e-01	1.3e-02	7.54e-01	6.4e-03	1.21e+00	6.6e-02	1.10e+00	1.2e-03	7.49e-01	4.3e-03	7.58e-01	7.7e-04
WFG3	5.84e-01	2.4e-02	4.36e-01	6.2e-03	4.86e-01	1.4e-02	5.62e-01	8.6e-05	3.59e-01	6.8e-03	3.61e-01	5.1e-03
WFG4	4.13e-01	3.3e-02	2.75e-01	1.7e-02	5.37e-01	2.8e-02	4.41e-01	1.1e-02	1.35e-01	1.4e-02	1.36e-01	1.4e-02
WFG5	4.25e-01	3.3e-02	2.86e-01	2.3e-02	5.34e-01	2.6e-02	4.64e-01	7.5e-03	1.41e-01	1.4e-02	1.33e-01	1.8e-02
WFG6	3.93e-01	3.1e-02	2.53e-01	2.1e-02	5.58e-01	3.5e-02	4.10e-01	1.1e-03	1.32e-01	2.8e-02	1.12e-01	1.6e-02
WFG7	3.88e-01	2.6e-02	2.51e-01	1.9e-02	5.21e-01	4.6e-02	4.10e-01	1.3e-04	1.13e-01	1.4e-02	1.17e-01	1.3e-02
WFG8	6.46e-01	4.9e-02	5.68e-01	7.3e-02	1.30e+00	8.5e-02	6.24e-01	9.8e-02	5.42e-01	4.7e-02	6.28e-01	5.5e-02

Table G.6. (continued.)

	Algorithms											
	NSGA-II		SPEA2		IBEA		MOEAD		MOCell		CB-EFO	
Problem	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**
WFG9	4.14e-01	3.2e-02	2.99e-01	1.4e-02	5.15e-01	3.7e-02	4.64e-01	2.5e-03	1.36e-01	1.2e-02	1.61e-01	2.1e-02
ZDT1	3.78e-01	3.0e-02	1.48e-01	1.5e-02	3.11e-01	1.6e-02	2.84e-01	8.7e-04	5.36e-02	1.6e-02	7.58e-02	1.1e-02
ZDT2	3.82e-01	3.4e-02	1.36e-01	1.4e-02	2.71e-01	2.1e-02	1.37e-01	1.7e-03	6.79e-02	1.8e-02	7.49e-02	1.2e-02
ZDT3	7.46e-01	1.7e-02	7.05e-01	3.7e-03	1.10e+00	2.6e-02	1.00e+00	2.4e-03	7.01e-01	1.3e-03	7.04e-01	2.1e-03
ZDT4	4.12e-01	3.8e-02	1.15e-01	1.2e-02	1.12e+00	5.0e-02	2.84e-01	1.4e-03	6.37e-02	1.4e-02	3.54e-01	2.3e-01
ZDT6	6.42e-01	4.4e-02	1.38e-01	1.3e-02	7.78e-01	3.0e-02	1.50e-01	1.0e-04	3.74e-02	7.6e-03	1.44e-01	3.0e-01

Table G.7. Mean and standard deviation values of EPSILON metric obtained by CB-EFO and the competitors.

	Algorithms											
	NSGA-II		SPEA2		IBEA		MOEAD		MOCell		CB-EFO	
Problem	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**
DTLZ1	5.48e-02	5.9e-03	3.38e-02	1.9e-03	2.89e-01	1.7e-02	4.94e-02	8.9e-04	5.19e-02	1.0e-02	1.89e-01	1.7e-01
DTLZ2	1.28e-01	1.7e-02	8.29e-02	9.2e-03	7.25e-02	3.5e-03	1.39e-01	4.2e-03	1.30e-01	1.9e-02	1.25e-01	1.4e-02
DTLZ3	1.30e-01	1.6e-02	7.80e-02	8.9e-03	7.28e-01	2.8e-01	1.37e-01	7.0e-03	1.24e-01	1.6e-02	2.93e+00	1.1e+00
DTLZ4	1.14e-01	1.5e-02	2.70e-01	2.8e-01	4.79e-01	3.0e-01	1.42e-01	4.8e-02	3.16e-01	3.7e-01	1.15e-01	2.0e-02
DTLZ5	1.05e-02	1.6e-03	7.46e-03	9.5e-04	1.28e-02	1.5e-03	2.57e-02	1.8e-04	4.45e-03	5.4e-04	4.74e-03	5.6e-04
DTLZ6	2.36e-02	1.5e-02	9.94e-03	7.7e-03	5.56e-02	1.9e-02	2.66e-02	1.4e-05	1.03e-02	1.0e-02	4.05e-03	2.1e-04
DTLZ7	1.39e-01	4.2e-02	1.41e-01	2.3e-01	1.07e+00	9.7e-01	5.60e-01	7.6e-01	7.06e-01	6.6e-01	1.40e-01	3.0e-02
LZ09$_{F1}$	1.72e-02	1.8e-03	1.11e-02	1.3e-03	1.52e-02	1.7e-03	7.94e-03	8.0e-04	9.80e-03	1.3e-03	1.07e-02	1.0e-03
LZ09$_{F2}$	2.04e-01	4.7e-02	2.05e-01	4.4e-02	2.26e-01	6.0e-02	6.78e-02	5.5e-02	2.61e-01	7.7e-02	1.37e-01	3.5e-02
LZ09$_{F3}$	1.31e-01	3.3e-02	1.45e-01	3.7e-02	1.78e-01	6.2e-02	1.18e-01	8.8e-02	2.26e-01	7.6e-02	1.04e-01	2.4e-02
LZ09$_{F4}$	1.38e-01	2.6e-02	1.61e-01	1.7e-02	1.84e-01	2.5e-02	7.77e-02	1.3e-02	1.68e-01	1.2e-02	1.55e-01	3.4e-02
LZ09$_{F5}$	1.04e-01	2.0e-02	1.22e-01	2.7e-02	1.31e-01	3.6e-02	9.81e-02	4.7e-02	1.62e-01	6.1e-02	7.55e-02	1.4e-02
LZ09$_{F6}$	2.94e-01	4.1e-02	2.64e-01	7.3e-02	5.08e-01	1.5e-01	2.06e-01	4.0e-02	3.80e-01	1.5e-01	3.40e-01	1.8e-01
LZ09$_{F7}$	3.60e-01	1.3e-01	3.85e-01	1.1e-01	4.60e-01	1.2e-01	5.63e-02	2.4e-02	5.19e-01	1.4e-01	4.46e-01	9.6e-02
LZ09$_{F8}$	3.17e-01	7.5e-02	3.59e-01	8.9e-02	4.28e-01	9.7e-02	3.10e-01	1.3e-01	4.91e-01	1.2e-01	4.62e-01	1.2e-01
LZ09$_{F9}$	2.29e-01	5.7e-02	2.36e-01	6.6e-02	2.76e-01	8.3e-02	7.37e-02	3.3e-02	3.69e-01	1.7e-01	1.60e-01	6.1e-02
WFG1	1.18e-01	1.1e-01	2.59e-01	1.7e-01	2.33e-01	1.8e-01	9.34e-02	3.4e-02	1.60e-01	1.0e-01	5.58e-02	1.7e-02
WFG2	4.04e-01	3.5e-01	3.49e-01	3.5e-01	4.93e-01	3.2e-01	2.71e-02	4.6e-04	5.72e-01	3.2e-01	9.37e-03	3.7e-04
WFG3	2.00e+00	4.6e-04	2.00e+00	4.2e-04	2.00e+00	3.8e-04	2.00e+00	5.9e-05	2.00e+00	2.8e-04	2.00e+00	2.9e-05
WFG4	3.58e-02	6.7e-03	2.46e-02	2.1e-03	2.64e-02	4.3e-03	2.72e-02	2.8e-03	1.38e-02	4.5e-04	1.45e-02	6.7e-04
WFG5	8.61e-02	9.0e-03	7.35e-02	2.5e-03	7.16e-02	3.4e-03	7.27e-02	2.1e-03	6.32e-02	2.8e-04	6.28e-02	2.7e-03
WFG6	4.37e-02	1.7e-02	3.68e-02	1.6e-02	4.73e-02	1.9e-02	2.29e-02	2.2e-04	5.69e-02	4.2e-02	2.21e-02	1.1e-02
WFG7	3.63e-02	6.5e-03	2.59e-02	3.1e-03	2.34e-02	3.3e-03	2.38e-02	1.5e-04	1.37e-02	3.9e-04	1.40e-02	4.4e-04
WFG8	3.95e-01	1.2e-01	3.88e-01	1.2e-01	4.55e-01	1.2e-01	4.01e-01	1.2e-01	4.53e-01	1.0e-01	4.30e-01	1.1e-01

Table G.7. (continued.)

	Algorithms											
	NSGA-II		SPEA2		IBEA		MOEAD		MOCell		CB-EFO	
Problem	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**
WFG9	3.74e-02	5.8e-03	2.78e-02	3.3e-03	2.59e-02	4.1e-03	2.50e-02	1.9e-03	1.66e-02	2.9e-03	1.91e-02	2.9e-03
ZDT1	1.28e-02	2.4e-03	7.95e-03	8.3e-04	1.11e-02	3.0e-03	8.15e-03	1.1e-04	5.38e-03	3.8e-04	5.56e-03	1.8e-04
ZDT2	1.30e-02	2.8e-03	7.80e-03	8.2e-04	9.11e-03	1.1e-03	6.62e-03	1.4e-04	5.23e-03	1.5e-04	5.41e-03	1.4e-04
ZDT3	8.40e-03	1.5e-03	8.79e-03	1.5e-03	1.66e-02	2.1e-03	1.58e-02	2.4e-04	1.11e-02	4.3e-02	5.22e-03	4.8e-04
ZDT4	1.25e-02	2.3e-03	7.29e-03	6.1e-04	7.63e-01	8.8e-02	8.75e-03	2.3e-04	5.35e-03	1.6e-04	9.27e-01	4.2e+00
ZDT6	1.29e-02	2.1e-03	6.06e-03	5.2e-04	1.09e-02	9.1e-04	4.85e-03	1.3e-05	4.37e-03	1.3e-04	4.46e-03	1.2e-04

Table G.8. Mean and standard deviation values of IGD metric obtained by CB-EFO and the competitors.

	Algorithms											
	NSGA-II		SPEA2		IBEA		MOEAD		MOCell		CB-EFO	
Problem	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**
DTLZ1	5.98e-04	3.3e-05	4.42e-04	1.1e-05	4.10e-03	3.2e-04	5.88e-04	3.5e-06	5.94e-04	9.6e-05	4.40e-03	4.7e-03
DTLZ2	7.77e-04	3.3e-05	5.88e-04	1.1e-05	9.90e-04	2.4e-05	7.62e-04	4.5e-06	7.55e-04	3.1e-05	7.03e-04	2.0e-05
DTLZ3	1.26e-03	8.0e-05	9.14e-04	1.4e-05	1.12e-02	2.3e-03	1.26e-03	1.2e-05	1.19e-03	5.8e-05	6.35e-02	2.5e-02
DTLZ4	1.21e-03	1.2e-04	2.80e-03	2.9e-03	4.81e-03	2.9e-03	1.24e-03	2.2e-04	3.15e-03	3.5e-03	1.24e-03	1.2e-04
DTLZ5	1.98e-05	1.1e-06	1.56e-05	5.0e-07	5.58e-05	3.3e-06	4.26e-05	1.8e-07	1.40e-05	3.5e-07	1.44e-05	4.0e-07
DTLZ6	1.39e-04	1.0e-04	6.55e-05	5.3e-05	4.10e-04	1.2e-04	9.58e-05	1.0e-07	7.22e-05	6.3e-05	3.39e-05	1.1e-06
DTLZ7	2.25e-03	1.3e-04	2.38e-03	2.7e-03	1.40e-02	9.3e-03	8.13e-03	6.8e-03	8.77e-03	7.2e-03	2.41e-03	2.0e-04
LZ09$_{F1}$	3.94e-04	1.7e-05	2.52e-04	1.0e-05	4.11e-04	3.9e-05	2.33e-04	1.1e-05	2.37e-04	9.0e-06	2.44e-04	8.9e-06
LZ09$_{F2}$	5.94e-03	1.4e-03	6.17e-03	1.4e-03	7.48e-03	2.1e-03	1.48e-03	1.7e-03	8.39e-03	2.9e-03	3.85e-03	1.3e-03
LZ09$_{F3}$	3.11e-03	1.2e-03	3.47e-03	1.2e-03	5.75e-03	2.6e-03	2.28e-03	2.3e-03	6.58e-03	3.1e-03	2.76e-03	8.4e-04
LZ09$_{F4}$	2.52e-03	6.8e-04	3.18e-03	6.8e-04	5.70e-03	9.6e-04	1.17e-03	2.3e-04	3.08e-03	5.2e-04	4.19e-03	1.3e-03
LZ09$_{F5}$	2.20e-03	5.4e-04	2.57e-03	9.2e-04	3.78e-03	1.4e-03	1.72e-03	1.4e-03	4.28e-03	2.3e-03	1.80e-03	4.3e-04
LZ09$_{F6}$	6.22e-03	1.3e-03	4.07e-03	1.2e-03	1.80e-02	2.6e-03	4.19e-03	1.0e-03	7.93e-03	2.7e-03	6.52e-03	2.8e-03
LZ09$_{F7}$	1.12e-02	4.4e-03	1.28e-02	5.1e-03	1.92e-02	5.6e-03	7.57e-04	4.0e-04	1.84e-02	6.1e-03	1.56e-02	4.2e-03
LZ09$_{F8}$	9.47e-03	2.6e-03	1.13e-02	3.9e-03	1.64e-02	4.6e-03	7.60e-03	4.0e-03	1.67e-02	5.0e-03	1.59e-02	4.7e-03
LZ09$_{F9}$	7.54e-03	2.6e-03	7.42e-03	2.3e-03	8.17e-03	2.9e-03	1.17e-03	3.9e-04	9.77e-03	5.0e-03	4.32e-03	1.5e-03
WFG1	1.18e-03	1.4e-03	2.35e-03	2.1e-03	2.97e-03	2.5e-03	5.16e-04	2.1e-04	1.26e-03	1.2e-03	3.41e-04	5.8e-05
WFG2	4.48e-03	3.6e-03	3.85e-03	3.7e-03	6.31e-03	2.4e-03	1.49e-03	1.2e-05	6.29e-03	3.5e-03	2.93e-04	1.0e-05
WFG3	3.01e-03	4.9e-07	3.01e-03	5.1e-07	3.01e-03	4.3e-07	3.01e-03	1.1e-07	3.01e-03	3.1e-07	3.01e-03	6.6e-08
WFG4	1.77e-04	1.0e-05	1.61e-04	5.9e-06	2.58e-04	3.7e-05	1.96e-04	6.4e-06	1.24e-04	2.1e-06	1.24e-04	1.8e-06
WFG5	9.64e-04	6.4e-06	9.50e-04	2.1e-06	9.72e-04	6.9e-06	9.51e-04	3.1e-05	9.42e-04	3.3e-07	9.27e-04	5.1e-05
WFG6	4.95e-04	3.2e-04	4.86e-04	3.4e-04	8.45e-04	3.5e-04	2.77e-04	2.2e-07	1.13e-03	9.9e-04	3.48e-04	2.1e-04
WFG7	1.26e-04	1.2e-05	9.82e-05	3.2e-06	1.11e-04	4.0e-06	1.22e-04	4.5e-08	8.44e-05	1.0e-06	8.42e-05	1.2e-06
WFG8	2.95e-03	9.9e-04	3.00e-03	9.1e-04	3.31e-03	6.6e-04	2.88e-03	1.1e-03	3.38e-03	7.6e-04	3.14e-03	1.0e-03

Table G.8. (continued.)

	Algorithms											
	NSGA-II		SPEA2		IBEA		MOEAD		MOCell		CB-EFO	
Problem	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**	**Mean**	**S.D.**
WFG9	1.37e-04	9.7e-06	1.09e-04	7.9e-06	1.56e-04	1.6e-05	1.24e-04	1.6e-06	9.92e-05	1.6e-05	1.04e-04	1.3e-05
ZDT1	1.85e-04	8.8e-06	1.45e-04	2.3e-06	2.16e-04	3.6e-05	1.62e-04	1.5e-06	1.35e-04	7.0e-07	1.35e-04	7.8e-07
ZDT2	1.91e-04	1.1e-05	1.44e-04	2.2e-06	3.55e-04	2.9e-05	1.41e-04	8.5e-07	1.39e-04	1.5e-06	1.41e-04	1.7e-06
ZDT3	1.34e-04	8.1e-06	1.20e-04	5.5e-06	7.80e-04	8.9e-05	2.83e-04	3.1e-06	1.48e-04	3.4e-04	9.98e-05	1.3e-06
ZDT4	1.85e-04	9.4e-06	1.42e-04	1.6e-06	2.27e-02	2.5e-03	1.69e-04	2.7e-06	1.35e-04	1.0e-06	2.86e-02	1.3e-01
ZDT6	2.27e-04	1.3e-05	1.39e-04	1.5e-06	2.37e-04	7.3e-06	1.40e-04	8.7e-09	1.33e-04	4.9e-07	1.32e-04	6.2e-07

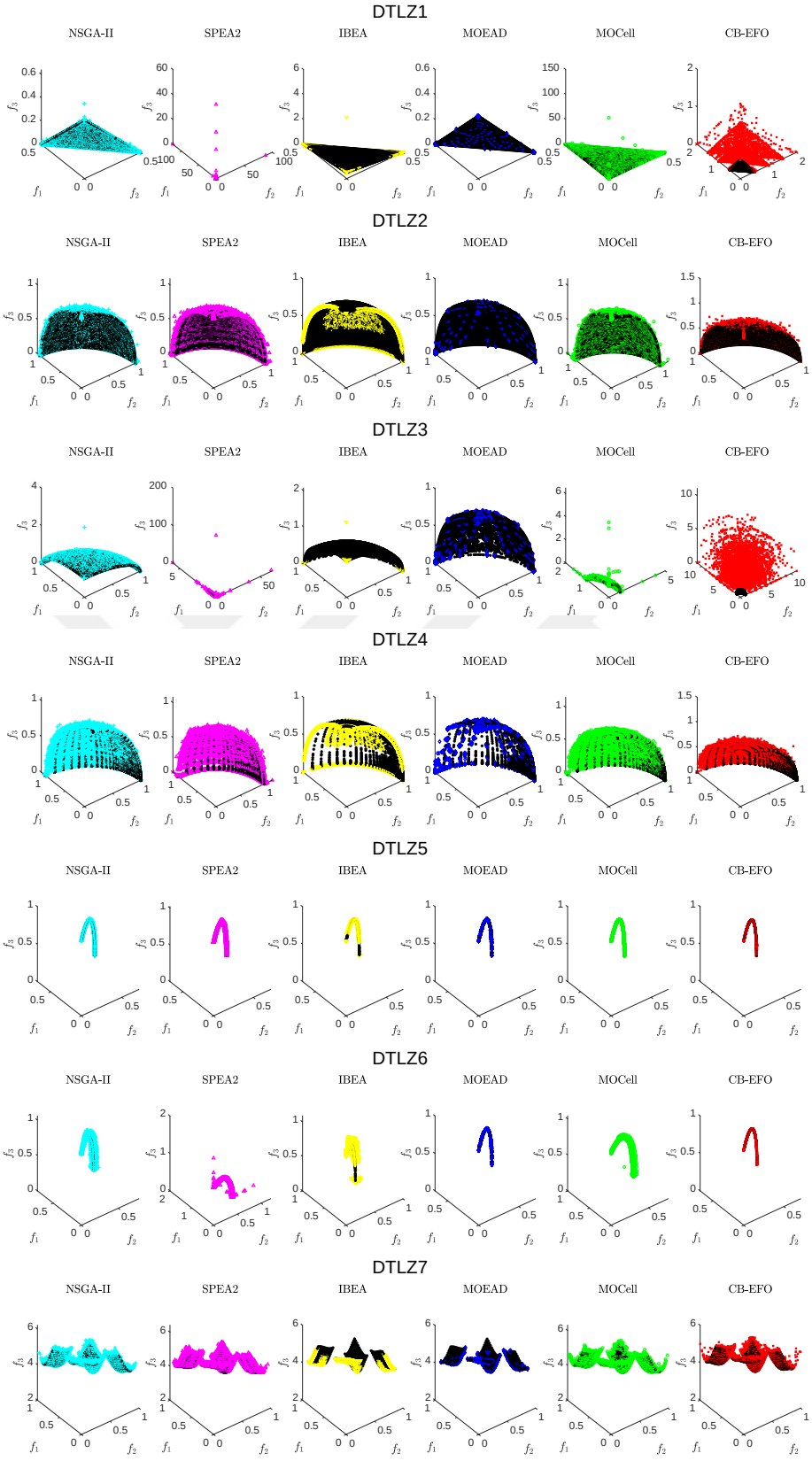

Figure G.1. Convergence to optimal Pareto fronts from DTLZ1 to DTLZ7.

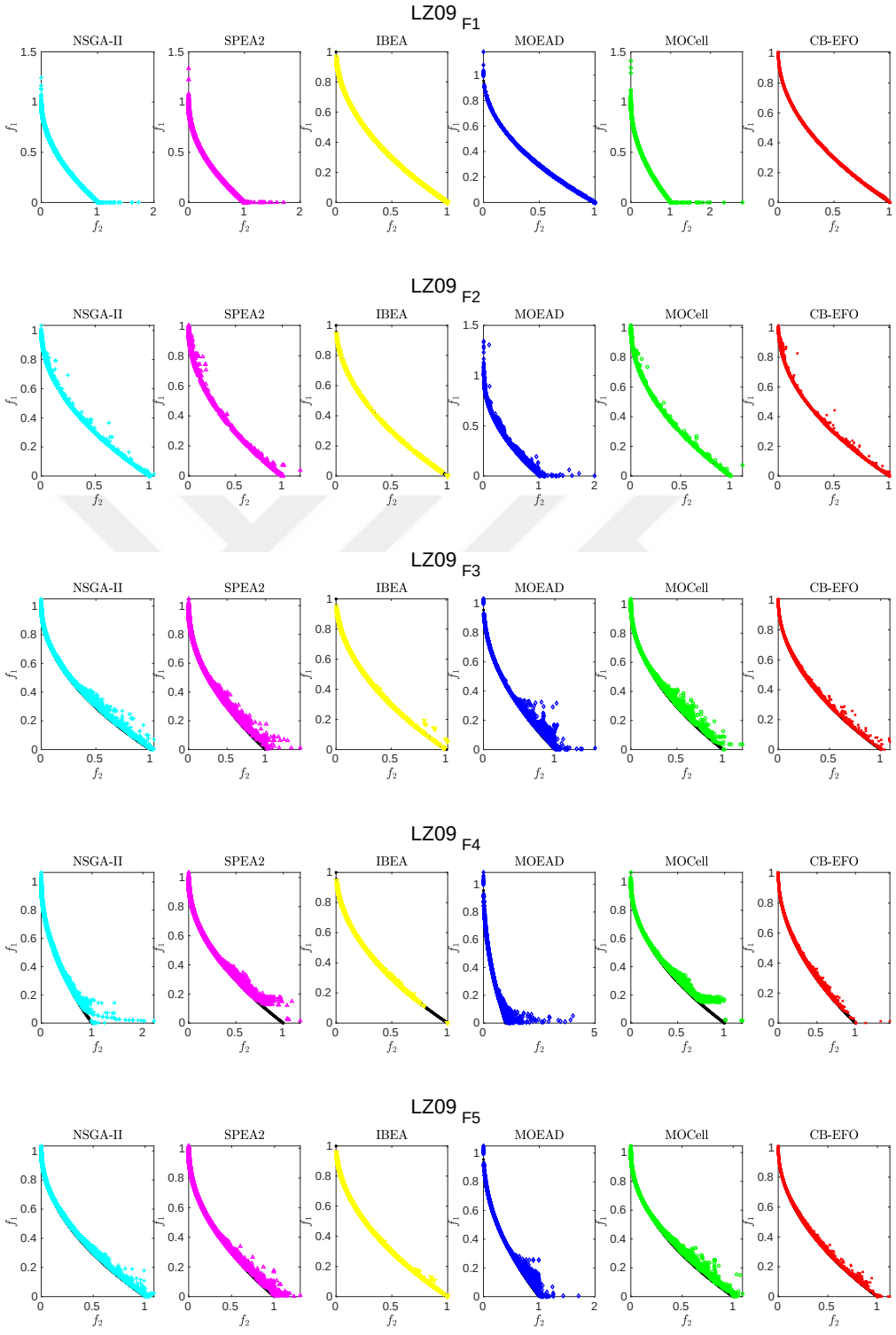

Figure G.2. Convergence to optimal Pareto fronts from $LZ09_{F1}$ to $LZ09_{F5}$.

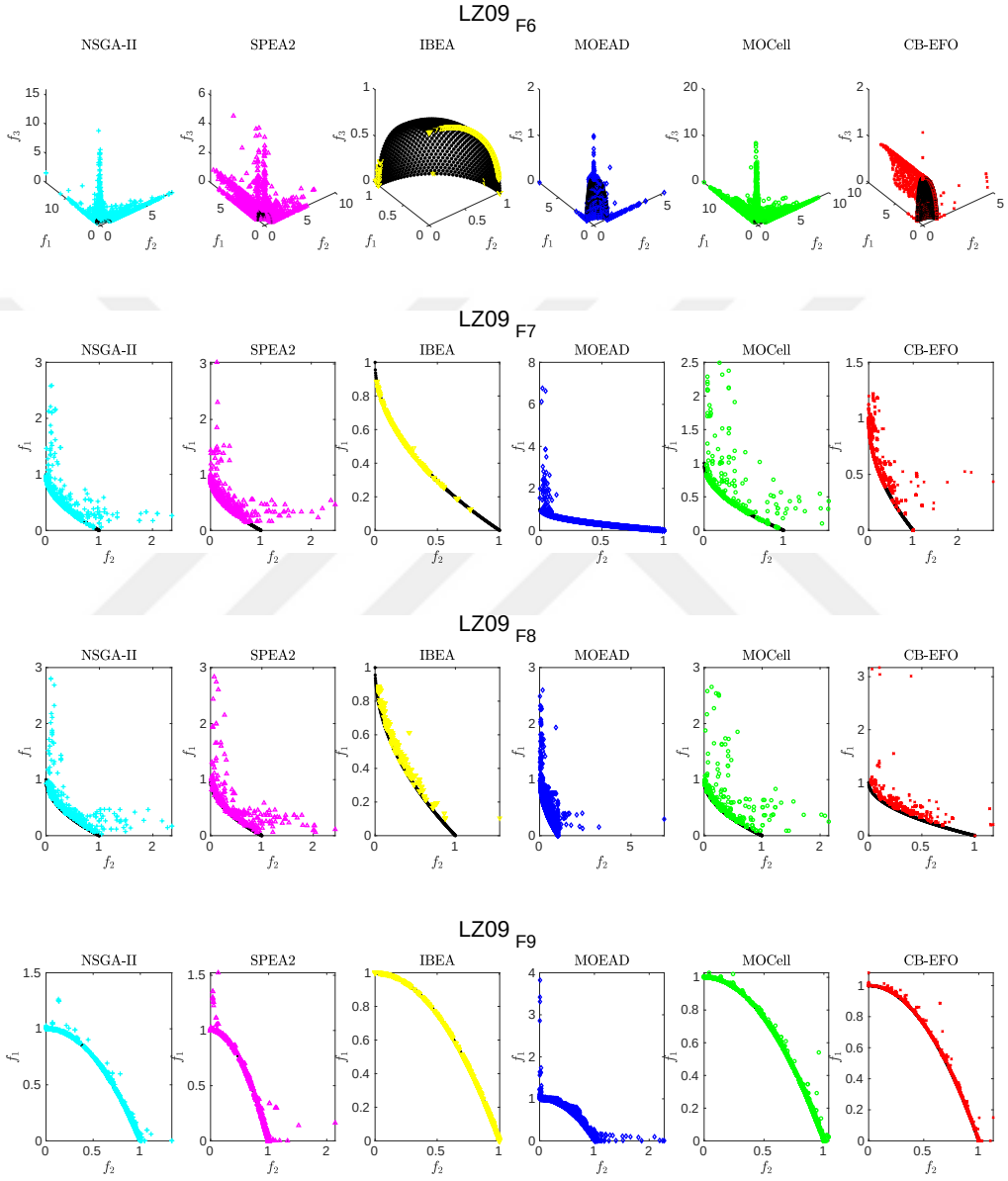

Figure G.3. Convergence to optimal Pareto fronts from $LZ09_{F6}$ to $LZ09_{F9}$.

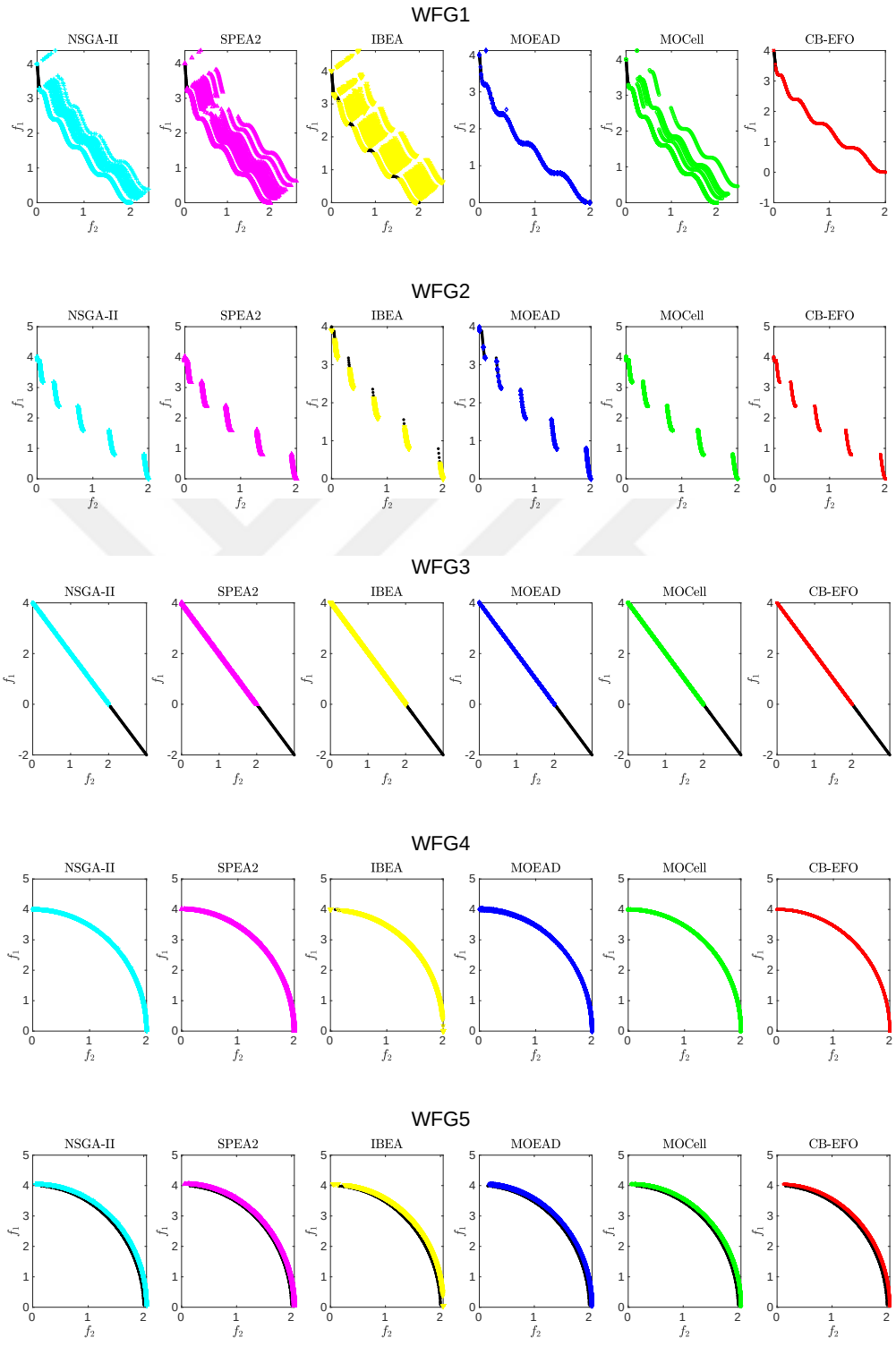

Figure G.4. Convergence to optimal Pareto fronts from WFG1 to WFG5.

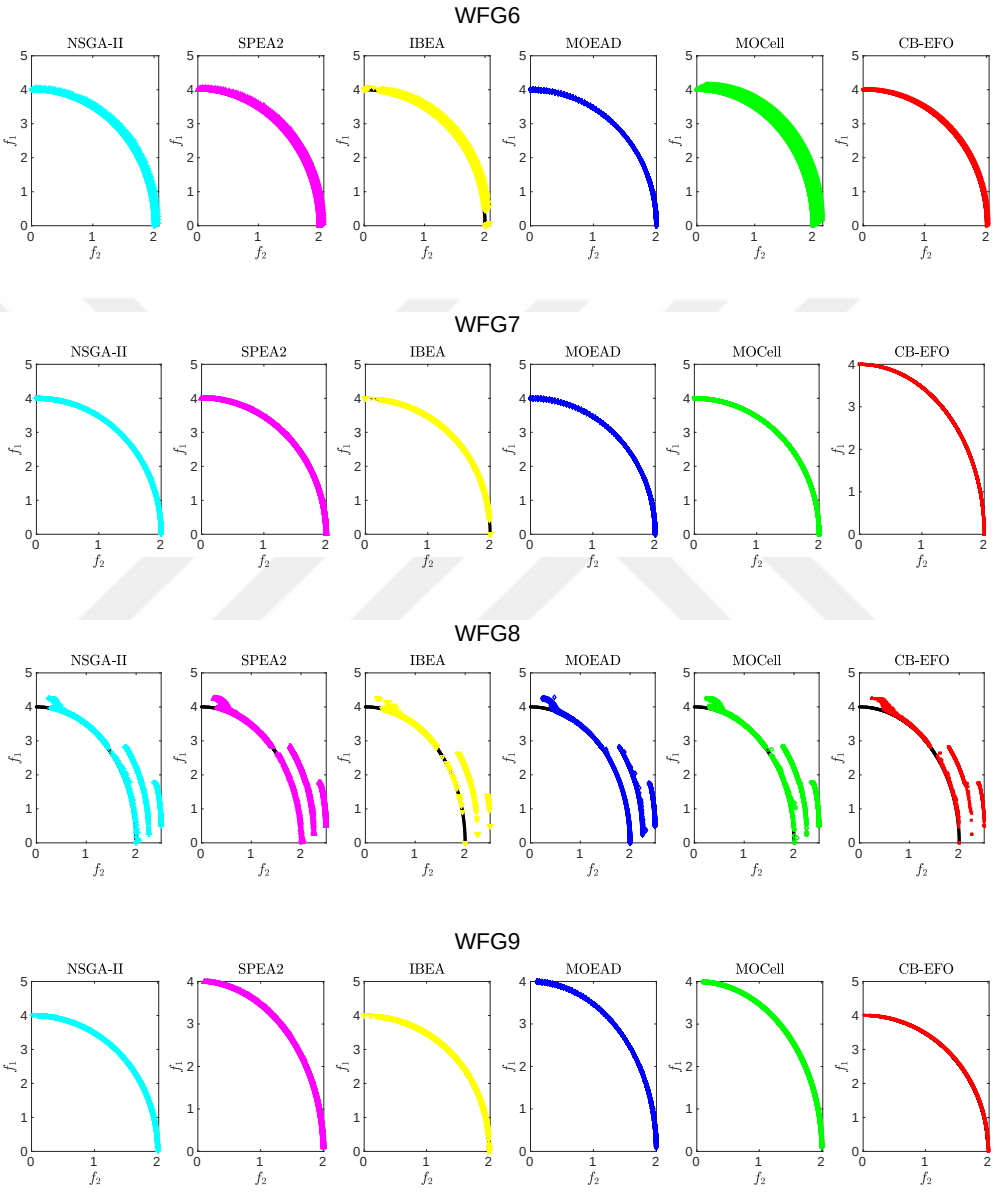

Figure G.5. Convergence to optimal Pareto fronts from WFG6 to WFG9.

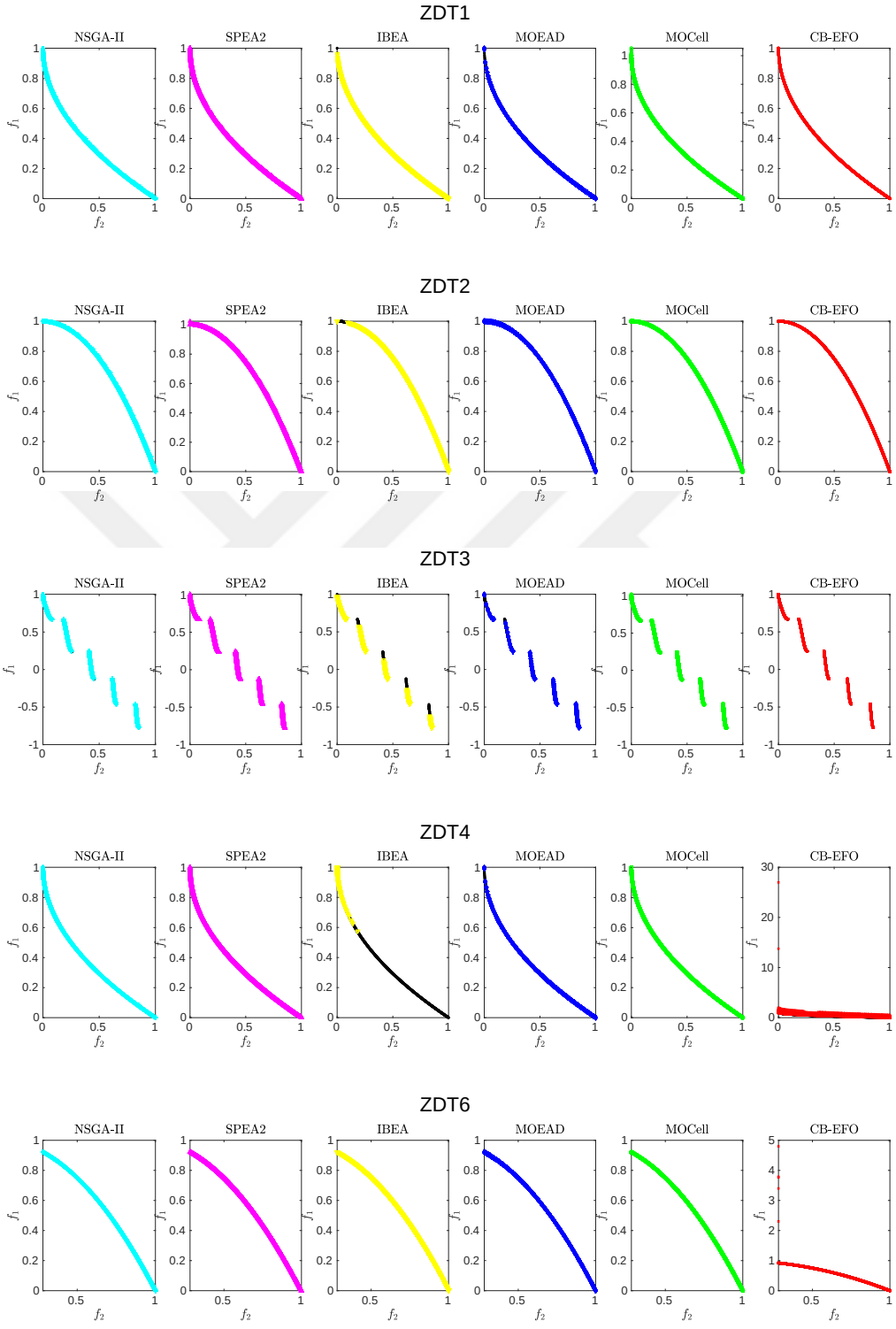

Figure G.6. Convergence to optimal Pareto fronts from ZDT1 to ZDT6.

HACETTEPE UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING
THESIS/DISSERTATION ORIGINALITY REPORT

HACETTEPE UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING
TO THE DEPARTMENT OF COMPUTER ENGINEERING

Date: 20/01/2020

Thesis Title: ELECTRIC FISH OPTIMIZATION: A NEW HEURISTIC ALGORITHM BASED ON ELECTROLOCATION

According to the originality report obtained by myself by using the *Turnitin* plagiarism detection software and by applying the filtering options stated below on 20/01/2020 for the total of 238 pages including the a) Title Page, b) Introduction, c) Main Chapters, d) Conclusion sections of my thesis entitled as above, the similarity index of my thesis is 10 %.

Filtering options applied:

1. Bibliography/Works Cited excluded
2. Quotes included
3. Match size up to 5 words excluded

I declare that I have carefully read Hacettepe University Graduate School of Sciene and Engineering Guidelines for Obtaining and Using Thesis Originality Reports; that according to the maximum similarity index values specified in the Guidelines, my thesis does not include any form of plagiarism; that in any future detection of possible infringement of the regulations I accept all legal responsibility; and that all the information I have provided is correct to the best of my knowledge.

I respectfully submit this for approval.

20/01/2020

Name Surname:	Selim YILMAZ
Student No:	N13143017
Department:	Computer Engineering
Program:	Computer Engineering
Status:	☐ Masters ☒ Ph.D. ☐ Integrated Ph.D.

ADVISOR APPROVAL

APPROVED.

Assoc. Prof. Dr. Sevil Şen AKAGÜNDÜZ

CURRICULUM VITAE

Credentials

Name,Surname : Selim YILMAZ

Place of Birth : 19/02/1988

Marital Status : Married

E-mail : selimy@cs.hacettepe.edu.tr

Address : Hacettepe University, Beytepe Campus, Computer Engineering Dept.

Education

B.Sc. : Electronics and Computer Education Department, Selcuk University,

M.Sc. : Computer Engineering Department, Suleyman Demirel University,

Ph.D. : Computer Engineering Department, Hacettepe University

Foreign Languages

English

Work Experience

Research Assistant at Hacettepe University

Areas of Experiences

Optimization, Heuristic Algorithms, Machine Learning, Data Mining

Projects and Budgets

Researcher on a joint Project with Mid Sweden University

Publications

Yilmaz, S., & Sen, S. (2019). Electric fish optimization: a new heuristic algorithm inspired by electrolocation. Neural Computing and Applications, 1-36.

Aydogan, E., Yilmaz, S., Sen, S., Butun, I., Forsström, S., & Gidlund, M. (2019). A Central Intrusion Detection System for RPL-Based Industrial Internet of Things. In 2019 15th IEEE International Workshop on Factory Communication Systems (WFCS) (pp. 1-5). IEEE.

Yilmaz, S., & Sen, S. (2019). Early Detection of Botnet Activities Using Grammatical Evolution. In International Conference on the Applications of Evolutionary Computation (Part of EvoStar) (pp. 395-404). Springer, Cham.

Hasdemir, S., Yilmaz, S., & Sen, S. (2019). A novel multi-featured metric for adaptive routing in mobile ad hoc networks. Applied Intelligence, 1-19.

Dilmen, E., Yilmaz, S., & Beyhan, S. (2017). An Intelligent Hybridization of ABC and LM Algorithms With Constraint Engineering Applications. In Handbook of Neural Computation (pp. 87-107). Academic Press.

Yılmaz, S., & Kücüksille, E. U. (2015). A new modification approach on bat algorithm for solving optimization problems. Applied Soft Computing, 28, 259-275.

Yılmaz, S., Kucuksille, E. U., & Cengiz, Y. (2014). Modified bat algorithm. Elektronika ir Elektrotechnika, 20(2), 71-78.

Yilmaz, S., & Kucuksille, E. U. (2013). Improved bat algorithm (IBA) on continuous optimization problems. Lecture Notes on Software Engineering, 1(3), 279.

Dilmen, E., Yılmaz, S., & Beyhan, S. (2013). Cascaded ABC-LM algorithm based optimization and nonlinear system identification. In 2013 International Conference on Electronics, Computer and Computation (ICECCO) (pp. 243-246). IEEE.