CUKUROVA UNIVERSITY

INSTITUTE OF NATURAL AND APPLIED SCIENCES

PhD THESIS

A Novel Optimization Algorithm EMEL: Exploration of Moving and Ever-shrinking Layers

Bilal İŞÇİMEN

Department of Computer Engineering

October, 2024

CUKUROVA UNIVERSITY

INSTITUTE OF NATURAL AND APPLIED SCIENCES

PhD THESIS APPROVAL

A Novel Optimization Algorithm EMEL: Exploration of Moving and Ever-shrinking Layers

---

Bilal İŞÇİMEN

Department of Computer Engineering

This Doctorate Thesis was evaluated by the following Jury Members on 23/10/2024 and was approved by unanimity of votes.

Jury : Asst. Prof. Dr. Mehmet SARIGÜL (Advisor) ……

: Prof. Dr. Selma Ayşe ÖZEL ……

: Assoc. Prof. Dr. Ahmet AYDIN ……

: Assoc. Prof. Dr. Esra SARAÇ EŞSİZ ……

: Assoc. Prof. Dr. Mümine KAYA KELEŞ ……

This Thesis was written in the Department of Computer Engineering, Institute of Natural and Applied Sciences.

Thesis Number:

Prof. Dr. Sadık DİNÇER
Director
Institute of Natural and Applied Sciences

CONTENTS

CUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

PhD THESIS

A Novel Optimization Algorithm EMEL: Exploration of Moving and Ever-shrinking Layers

Bilal İŞÇİMEN

Advisor: Asst. Prof. Dr. Mehmet SARIGÜL

Department of Computer Engineering

ABSTRACT

This study presents a novel optimization algorithm EMEL: Exploration of Moving and Ever-shrinking Layers. EMEL's layer building strategy embodies a balance between exploration and exploitation. The search space is partitioned into layers that are bounded by concentric rectangles whose dimensions increase exponentially. Contrary to its popular counterparts, EMEL does not produce candidate solutions influenced by others. Instead, the candidate solutions are produced anywhere inside a randomly selected layer. The probability of selecting a layer in the close vicinity of the best solution is higher than elsewhere. Therefore, most of the solutions exploit the inner layers, where a few of them still explore the distal layers. To demonstrate its search abilities, EMEL was tested on a total of 120 benchmarks which consist of sixty standard benchmarks, fourteen CEC 2005 benchmarks, twenty-nine CEC 2017 benchmarks, thirteen constrained benchmarks and four real-life constrained engineering design problems. The benchmark set consisting various characteristics such as shifted/rotated functions, hybrid functions, composition functions and also constrained problems include equality and inequality constraints to be satisfied. Four algorithms; Grey Wolf Optimizer, Artificial Bee Colony, Particle Swarm Optimization and Vortex Search Algorithm were included in the experiments to carry out a comparative work. Rather than using fixed parameters, design parameters of all algorithms were adjusted for each benchmark. The results were compared and analysed by utilizing Mann Whitney U test. The results showed that EMEL is superior to its counterparts by achieving high scores in the tests.

Keywords: Optimization Algorithm, Meta-heuristic, Population-based, EMEL

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DOKTORA TEZİ

Yeni Bir Optimizasyon Algoritması EMEL: Hareketli ve Sürekli Küçülen Katmanlarla Arama

Bilal İŞÇİMEN

Danışman: Dr. Öğr. Üyesi Mehmet SARIGÜL

Bilgisayar Mühendisliği Anabilim Dalı

ÖZ

Bu tez çalışmasında popülasyon temelli yeni bir optimizasyon algoritması tasarlanmış ve sunulmuştur. Algoritmanın adı *EMEL: Exploration of Moving and Ever-shrinking Layers* olarak verilmiştir. Algoritmanın çözüm üretme yaklaşımı hareket halinde olan ve sürekli küçülen katmanlar kullanarak arama yapmaya dayanmaktadır. Bu algoritmanın katman oluşturma stratejisi keşif ve sömürü aşamaları arasında doğal bir denge sağlamaktadır. Arama uzayı üzerinde aynı merkeze sahip iç içe katmanlar oluşturulmaktadır. Merkezde o ana kadar bulunmuş en iyi global optimum çözüm yer almaktadır. Bu katmanlar merkeze yakın noktalarda dar, merkezden uzaklaştıkça genişleyen bir yapıya sahiptir. Önerilen algoritmada iyileştirilecek olan bir çözüm öncelikle oluşturulan katmanlardan birini rastgele seçmekte ve pozisyonunu seçtiği bu katman içinde rastgele bir konuma güncellenmektedir. Katman sayısı merkez etrafında fazla, merkezden uzaklaştıkça daha az olduğu için global optimum çözüme daha yakın olan katmanlardan birinin seçilmesinin ihtimali daha yüksektir. Böylece konumu güncellenen çözümlerin birçoğu global optimuma yakın bölgelerde sömürü etkinliği sürdürürken çözümlerin bir kısmı global optimuma uzak bölgelerde keşif çalışması sürdürmektedir. Arama sürecinin başında ama uzayı kadar geniş olabilen katmanlar süreç ilerledikçe daralmaktadır. Bu durum başlangıçta daha geniş alanlarda keşif yapmayı desteklemekte ve çalışma süresinin sonlarına doğru sömürü etkinliği olan yerel en iyileştirme operasyonlarını teşvik etmektedir. Önerilen algoritmanın yetkinliğini kanıtlamak için 120 test fonksiyonu üzerinde deneyler yapılmıştır. Bu fonksiyonların 60 tanesi standart optimizasyon problemleri, 29 tanesi CEC 2017 fonksiyonları, 14 tanesi CEC 2005 fonksiyonları, 13 tanesi kısıtlı optimizasyon problemi ve 4 tanesi kısıtlı mühendislik tasarım problemidir. Karşılaştırmalı bir çalışma yürütmek için aynı deneyler Grey Wolf Optimizer, Particle Swarm Optimization, Artifical Bee Colony ve Vortex Search Optimization algoritmaları ile tekrarlanmış ve elde edilen sonuçlar istatistiksel yöntemlerle karşılaştırılmıştır. Elde edilen sonuçların karşılaştırılmasında Mann Whitney U testi kullanılmış ve EMEL algoritmasının anılan rakiplerden daha başarılı sonuçlar elde ettiğini görülmüştür.

Anahtar Kelimeler: Optimizasyon algoritması, meta-sezgisel, popülasyon temelli, EMEL

GENİŞLETİLMİŞ ÖZET

Optimizasyon, belirli bir amaca uygun olan en iyi çözümü bulma süreci olarak tanımlanabilir. Örneğin bir matematik denkleminin alabileceği en küçük değerin araştırılması bir optimizasyon problemidir. Burada denklemin alabileceği bütün girdilerin oluşturacağı çıktılardan meydana gelen çözümler optimizasyon probleminin arama uzayını ifade eder. Arama işlemi en iyileştirme amacı gözetilerek bu uzay üzerinde gerçekleştirilir. Gerçek hayatta da karşımıza çıkabilecek problemler bilgisayarda temsil edilebilecek şekilde modellenerek bir optimizasyon problemi oluşturulabilir. Ele alınacak problem basitçe bir girdi ve bir çıktından oluşabileceği gibi problemin çözümü sırasında uyulması gereken bazı ek kısıtlılıklar olabilir. Örneğin denklem girdisinin değerlerini alabileceği sayısal bir aralık tanımlanması bir kısıtlılıktır. Arama uzayında problemin çözümünde ihtiyaç duyulan bütün kısıtlamaları sağlayan çözümler makul çözüm olarak isimlendirilir. Elde edilen makul çözümlerin optimizasyon anlamında genellikle olabilecek en küçük veya en büyük değerde olması istenir. Bir sistemin verimini en yüksek seviyeye çıkarmak veya bir sürecin maliyetini en aza indirmek gibi amaçlar bu konuya verilebilecek optimizasyon problemi örnekleridir.

Gerçek hayat uygulamalarında kaynaklar, zaman ve para gibi girdiler her zaman sınırlı olduğundan mühendislik, ekonomi, lojistik gibi günlük hayat uygulamalarının kendi gereksinimleri doğrultusunda optimize edilmesi gereken problemleri bulunmaktadır. Örneğin bir bilgisayarda yürütülen işlemler sırasında bilgisayarın bütün işlemleri aksamadan gerçekleştirmesine olanak sağlayacak en uygun bellek ve işlemci tahsisi bilgisayar bilimleri açısından her zaman üzerinde çalışılan bir problemdir. Gezgin satıcı problemi lojistik alanında en az maliyetli ve en kısa yolun araştırılmasını konu alan bir diğer önemli problemdir. Sınırlı bir alanda kurulan bir tesisin verimliğini en yüksek seviyeye çıkarmak için makine ve teçhizatları uygun şekilde alana yerleştirmek veya depolanması gereken farklı ölçüdeki malzemelerin sınırlı bir alan içindeki en uygun yerleşimini araştırmak da bir optimizasyon problemidir. Bir inşaat sahasına yerleştirilecek kule vinçleri sahada yer alması gereken diğer elemanları da gözeterek en az tehlike oluşturacak şekilde konumlandırmak güvenlik gibi önemli bir ölçütü önceleyen farklı bir araştırma alanıdır. Bu gibi örnekleri birçok farklı günlük hayat uygulamasını ele alarak çoğaltmak mümkündür.

Günlük hayatta her zaman optimize edilmesi arzulanan problemlerin yer alması sebebiyle bunları optimize edecek yöntemler geliştirmekle ilgili çalışmaların tarihi çok eski zamanlara dayanmaktadır. Matematikten ilham alan optimizasyon algoritmalarının ilk örnekleri türev ve diferansiyel tabanlı hesaplamalar gerçekleştiren deterministik yaklaşımlardır. Bu alanda en bilinen yöntemlerden birisi 19. yüzyılın ortalarında sunulmuş olan gradyan iniş algoritmasıdır. Doğrusal programlama ve doğrusal olmayan programlama algoritmaları da 20. yüzyılın ortalarında çalışılmış olup deterministik yöntemlere verilebilecek diğer örnekleridir. Deterministik yöntemler sabit adımlar takip ederek en iyi çözümü tespit etmeye yönelik çalışırlar.

Sezgisel yöntemler ise deterministik algoritmalardan farklı olarak en iyi çözümü değil fakat makul sayılabilecek kadar iyi bir çözümü daha hızlı sürelerde bulmayı amaçlar. Bu yöntemlerden başlıca olanları 1950 ve 1960'lı yıllarda çalışılmış olan en yakın komşu algoritması, tepe tırmanma algoritması ve açgözlü yaklaşım algoritmasıdır. Sezgisel yöntemler genellikle belirli bir problem seti için dizayn edilen, probleme bağımlı algoritmalardır. Buradaki amaç hızlı bir şekilde kabul edilebilir bir çözüm üretmektir. Üretilen çözümün optimum çözüm olmayabileceği, hatta bu çözümden uzak olabileceği göze alınır. Genellikle çözüm uzayı orta ölçekli olan problemler üzerine uygulanması tercih edilir

Günümüze doğru yaklaştıkça bilgisayar gücünün artmasıyla birlikte farklı disiplinlerden ve doğa fenomenlerinden ilham alan meta-sezgisel algoritmalar sergilenmeye başlanmıştır. Genetik algoritma 1975'lerde geliştirilen, ilhamını doğal seçilim süreçlerinden alan ve güncel optimizasyon çalışmalarının temel taşlarından biri olarak kabul edilen bir algoritmadır. Bunu takip eden zamanlarda 1990'larda geliştirilen benzetilmiş tavlama algoritması, karınca kolonisi algoritması, diferansiyel gelişim algoritması, parçacık sürü optimizasyonu algoritması gibi örnekler bu alana öncülük eden en büyük yeniliklerden bazılarıdır.

Meta-sezgisel algoritmaların yaygın hale gelmeye başlamasındaki sebeplerden önde geleni bu algoritmaların problemden bağımsız olmasıdır. Bilgisayarda ifade edilebilecek şekilde modellenen her problemin çözümünde bu algoritmaları kullanmak mümkündür. Ayrıca bu algoritmaların tasarımı oldukça sadece ve anlaşılırdır. Bu algoritmalar ilhamını doğada bulunan fiziksel fenomenler, hayvan sürülerinin davranışları, evrimsel süreçler gibi çok bilindik konseptlerden alır. Tasarlanan algoritmaların sade olması, farklı çalışmaların performans yükseltmeyi sağlayacak özelliklerini kullanarak algoritmada geliştirme yapılmasını kolaylaştırır.

Üzerinde işlem yaptığı çözüm sayısı bakımından meta-sezgisel algoritmalar iki ana sınıfa ayrılabilir. Bunlar tek çözüm temelli ve popülasyon temelli yaklaşımlarıdır. Benzetilmiş tavlama algoritması tek çözüm temelli algoritmaların bir örneğidir. Bu yaklaşımda optimum değeri arama süreci tek bir rastgele çözüm üreterek başlar ve belirli bir sonlandırma kriteri sağlanana kadar bu çözüm iyileştirilerek devam eder. Popülasyon temelli yaklaşımlarda ise aramaya birden fazla sayıda rastgele çözüm üreterek başlanır. Bu çözümler ilerleyen çevrimler boyunca birbirinden ve çevresinden etkilenerek gelişmeye devam eder ve nihayetinde problemi çözecek optimum değerin elde edilmesi amaçlanır.

Popülasyon temelli aramalar gerçekleştiren algoritmalarda üretilen çözümler birbirine yol gösterdiği için çözüm uzayındaki yerel optimumlardan kaçınma kabiliyetleri vardır. Arama uzayının yeterince detaylı aranması optimum çözümü bulmaya giden yolda önemli bir ölçüttür. Meta-sezgisel algoritmalarının popülasyon temelli yaklaşım tercihleri çok geniş arama uzayına sahip problemler için uygun bir altyapı sağlamaktadır. Bu tür algoritmaların geniş arama uzaylarında elverişli olmasının bir diğer nedeni de stokastik bir yapıya sahip olmasıdır. Problem için üretilen çözümleri temsil eden açılış popülasyonu arama uzayı üzerinde rastgele olarak oluşturulur. Arama işleminin

sonraki aşamalarında bu açılış popülasyonunda yer alan çözümleri iyileştirerek en iyi sonuca ulaşılması amaçlanır. Rasgele üretilen çözümlerden faydalanmak, bu algoritmaların optimum noktası bilinmeyen gerçek hayat problemlerine uygun olması için geçerli olan sebeplerden bir diğeridir.

Popülasyon temelli meta-sezgisel algoritmaların önemli alt dallarından birisi sürü zekasıdır. Sürü zekası kavramı doğada bulunan canlıların çeşitli amaçlar doğrultusunda sergiledikleri sürü davranışlarını ifade eder. Bu davranışlar çoğunlukla besin arama, avlanma gibi en temel hayatta kalma ihtiyacına hizmet eder. Karınca kolonisi algoritması bu konuda en bilinen algoritmaların başında gelir. Karıncaların besin arama davranışlarını modelleyen bu algoritma sürü zekası alanında ortaya konulacak onlarca farklı yaklaşıma öncülük etmiştir. Benzer başka bir çalışmada da kuş ve balık sürülerinin besin arama davranışlarından ilham alan parçacık sürü optimizasyonu algoritması sunulmuştur. Takip eden senelerde yine farklı bir besin arama davranışını modelleyen popüler algoritmalardan bir diğeri ise yapay arı kolonisi algoritmasıdır.

Bahsedilen bütün meta-sezgisel algoritmaların arasında bulunan tasarım farklarına bakılmaksızın hepsinde bulunan ortak iki davranış vardır. Bu iki davranış arama sürecinin iki evresi olan keşif ve sömürü süreçleridir. Keşif aşamasında üretilen çözümlerin uzay üzerinde olabildiğince etrafa dağılması ve problemin gereksinimlerini karşılamaya yatkın en verimli çözüm bölgelerini bulması amaçlanır. Bir algoritmanın keşif aşamasını destekleyecek güçlü stokastik operatörlere sahip olması bir sonraki aşamaya geçildiğinde gerçek en iyi çözümlerin bulunmasına öncülük edecektir. Sömürü aşaması ise keşif aşamasında tespit edilen verimli ve sınırları daha küçük bölgeler üzerinde yerel arama ve en iyileştirme yeteneklerini ifade eder. Keşif aşamasında tespit edilen verimli çözüm bölgeleri yalnızca bir kılavuz görevi görecek olup algoritmanın çalışması sona erdiğinde optimum çözümü sunacak olan faktör sömürü aşamasıdır.

Algoritmanın çalışma süresi içinde keşif ve sömürü aşamalarını en iyi çözümü bulacak şekilde dengelemek meta-sezgisel algoritmalar için yıllar boyunca zorlu bir görev olmuş ve araştırmacıların ilgisini cezbetmiştir. Keşif aşamasında zayıf etkinlik gösteren algoritmaların yerel optimum çözümlerin etrafında sıkışarak global optimumu barındıran bölgelere ulaşamaması mümkündür. Bunun tersi olarak güçlü keşif yetenekleri olan algoritmaların yeterince başarılı sömürü operatörleri yoksa global optimumu barındıran bölgeler başarılı bir şekilde tespit edilse bile global optimum çözüm elde edilmeyebilir.

Keşif ve sömürü aşamaları arasındaki dengenin nasıl sağlanacağı dışında meta-sezgisel yöntemlerin verimliliğini yükseltecek yaklaşımları araştıran çalışmalar da literatürde yer almaktadır. Arama operatörlerinin davranışlarının çeşitlendirilmesi, üretilen eski çözümlerden aramanın farklı evrelerinde yeniden faydalanılması gibi metotlar algoritmaların çözüm kalitesini arttırmayı amaçlamaktadır. Bütün bu çabaya rağmen bir algoritmanın her türlü problem üzerinde başarılı olması mümkün değildir. Bir algoritma üzerinde yapılan bütün geliştirmeler araştırmanın doğası gereği sınırlı sayıda problem üzerinde test edilebilmektedir ve bu da henüz test edilmemiş olan problemlerde

başarısız olma ihtimali her zaman mümkün kılmaktadır. Bu durumu açıklayan *No Free Lunch* teorimi bu alandaki çalışmaların sürmesindeki başlıca dayanak noktasıdır.

Bu dayanak çerçevesinde hazırlanan bu tez çalışmasının amacı doğadan ilham alan optimizasyon algoritmalarının keşif mekanizmalarını incelemek ve bu konuda gelişim sağlayacak teknikler sunmaktır. Yapılan çalışmalar neticesinde popülasyon temelli yeni bir optimizasyon algoritması tasarlanmış ve sunulmuştur. Algoritmanın adı *EMEL: Exploration of Moving and Ever-shrinking Layers* olarak verilmiştir. Algoritmanın çözüm üretme yaklaşımı hareket halinde olan ve sürekli küçülen katmanlar kullanarak arama yapmaya dayanmaktadır. Bu algoritmanın katman oluşturma stratejisi keşif ve sömürü aşamaları arasında doğal bir denge sağlamaktadır. Arama uzayı üzerinde aynı merkeze sahip iç içe katmanlar oluşturulmaktadır. Merkezde o ana kadar bulunmuş en iyi global optimum çözüm yer almaktadır. Bu katmanlar merkeze yakın noktalarda dar, merkezden uzaklaştıkça genişleyen bir yapıya sahiptir. Önerilen algoritmada iyileştirilecek olan bir çözüm öncelikle oluşturulan katmanlardan birini rastgele seçmekte ve pozisyonunu seçtiği bu katman içinde rastgele bir konuma güncellenmektedir. Katman sayısı merkez etrafında fazla, merkezden uzaklaştıkça daha az olduğu için global optimum çözüme daha yakın olan katmanlardan birinin seçilmesinin ihtimali daha yüksektir. Böylece konumu güncellenen çözümlerin birçoğu global optimuma yakın bölgelerde sömürü etkinliği sürdürürken çözümlerin bir kısmı global optimuma uzak bölgelerde keşif çalışması sürdürmektedir. Arama sürecinin başında ama uzayı kadar geniş olabilen katmanlar süreç ilerledikçe daralmaktadır. Bu durum başlangıçta daha geniş alanlarda keşif yapmayı desteklemekte ve çalışma süresinin sonlarına doğru sömürü etkinliği olan yerel en iyileştirme operasyonlarını teşvik etmektedir. Önerilen algoritmanın yetkinliğini kanıtlamak için 120 test fonksiyonu üzerinde deneyler yapılmıştır. Bu fonksiyonların 60 tanesi standart optimizasyon problemleri, 29 tanesi CEC 2017 fonksiyonları, 14 tanesi CEC 2005 fonksiyonları, 13 tanesi kısıtlı optimizasyon problemi ve 4 tanesi kısıtlı mühendislik tasarım problemidir. Karşılaştırmalı bir çalışma yürütmek için aynı deneyler Grey Wolf Optimizer, Particle Swarm Optimization, Artifical Bee Colony ve Vortex Search Optimization algoritmaları ile tekrarlanmış ve elde edilen sonuçlar istatistiksel yöntemlerle karşılaştırılmıştır. Elde edilen sonuçların karşılaştırılmasında Mann Whitney U testi kullanılmış ve EMEL algoritmasının anılan rakiplerden daha başarılı sonuçlar elde ettiğini görülmüştür.

ACKNOWLEDGEMENTS

I would like to thank to my supervisor Asst. Prof. Dr. Mehmet SARIGÜL for his guidance, support and his valuable time for the proposed thesis work. His kindness and positive behaviour during this process mean a lot to me.

I would like to express my sincere gratitude to Assoc. Prof. Dr. Mustafa ORAL for his encouragement and his guidance that enabled me to carry out this thesis study. I am really grateful for the opportunity to collaborate with such a wise individual like him during an important time in my life.

I would like to thank to vauable academicians Prof. Dr. Selma Ayşe ÖZEL and Assoc. Prof. Dr. Ahmet AYDIN for their help and support through the study. Their kind and helpful attitude really contributed to my positive experience during my time studying on my thesis.

I would like to thank to the members of PhD thesis evaluation jury, Assoc. Prof. Dr. Esra SARAÇ EŞSİZ and Assoc. Prof. Dr. Mümine KAYA KELEŞ for their valuable time and suggestions.

I would like to express my deepest gratitude and respect to Prof. Dr. Emel ORAL, in whose honour we have named the proposed algorithm. Her warm attitude was one of the things that made me feel happy.

I would like to express my sincere thanks to Asst. Prof. Dr. Merve Nilay AYDIN, who has always been a dear friend to me since the day we met. The fact that she is standing by me in all circumstances is one of the factors that enable me to move forward.

I would like to thank to Dr. Elif Gülfidan TOSUN for her support and sincerely sharing her experience with me through the thesis period.

I would like to express my warmest thanks to my beloved wife Aliye Tuğba İŞÇİMEN and to my dear family. They have always supported me through my studies by love and patience.

LIST OF TABLES

LIST OF FIGURES

SYMBOLS AND ABBREVIATIONS

ABC : Artificial Bee Colony
BFO : Bitterling Fish Optimization
BOA : Bobcat Optimization Algorithm
CEC : Congress on Evolutionary Computation
CFO : Cleaner Fish Optimization
CFOA : Catch Fish Optimization Algorithm
EMEL : Exploration of Moving and Ever-shrinking Layers
GB : Global Best Solution
GWO : Grey Wolf Optimizer
HO : Hippopotamus Optimization
NFL : No Free Lunch
PB : Personal Best Solution
PFA : Polar Fox Optimization
PSO : Particle Swarm Optimization
VSO : Vortex Search Algorithm
WSA : Wave Search Algorithm

1. INTRODUCTION

High dimensionality, multimodality, non-differentiability, and extensive solution spaces complicate the identification of optimal solutions in optimization problems. Numerous algorithms are claimed to be robust against the obstacles that could occur in optimization tasks. These algorithms have drawn inspiration from several sources, including natural phenomena, organisms, and scientific experiences (Boussaïd et al., 2013; Mirjalili et al., 2014). Despite the diversity of inspired sources, optimization algorithms can be categorized into fundamental groups, including heuristics and metaheuristics (Doğan and Ölmez, 2015). Heuristic algorithms tend to be customized designs for specific problems, whereas metaheuristics are comprehensive strategies that guide approaches in the pursuit of problem solutions. Unlike heuristics, metaheuristics are not reliant on specific problem types and could be extensively studied across a wide variety of challenges.

Metaheuristic algorithms have recently gained popularity due to straightforward reasons. The principal motivation is that the inspiration relies on fundamental concepts –such as the behaviours of creatures in nature and physical phenomena– and metaheuristics can be readily implemented. Furthermore, gradient computations are unnecessary, as the search is conducted stochastically. Due to their straightforward structure and stochastic nature, metaheuristics can be employed across a broad spectrum of real-world problems. Another significance is that these algorithms could avoid local optima (Mirjalili et al., 2014; Mirjalili and Lewis, 2016).

Metaheuristic algorithms can also be classified according to the number of solutions in processing batch. This can be single solution-based or population-based (Boussaïd et al., 2013; Doğan and Ölmez, 2015). Single solution-based approach uses a single candidate solution to begin the search process. This single candidate is enhanced throughout the iterations to attain an optimum value. Population-based metaheuristics, on the other hand, use a set of solutions to perform the optimization. In contrast to single solution-based approach, population-based metaheuristic starts searching process with a random initial population. These candidate solutions are enhanced over the course of iterations. It is assumed that population-based metaheuristics have some advantages compared to single solution-based algorithms. The most important of all is exchanging information among individuals. The population is initialized at random over the search space. Thus, the individuals are able to know about whole space from each other. This causes the individuals to displace toward the more promising regions of the search space. Also, better candidates provide assistance to individuals suffering from stagnation on local optimum.

Regardless of search nature, a good optimization algorithm should satisfy two important criteria; exploration and exploitation. Exploration is the process of searching the space for promising solutions at random without any reference to stored information. Exploitation, on the other hand, is refining the solutions by confining the search to a small area within the vicinity of a best candidate solution ever found. Exploration is expected to be intensive at initial steps to find out fruitful regions

worth to effort local search. Exploitation is emphasized at latter stages in the hope of refining the solutions by exhaustively exploring the promising regions. Balancing both exploration and exploitation has been a challenging task for algorithms over the years.

Over the years many algorithms have been presented that are claimed to challenge the obstructions and provide satisfactory solutions to optimization problems. Each algorithm has contributed to the literature with its distinctive features. Despite dozens of algorithms exist in the literature, a question has still seeming to exist: which algorithm is the best for optimization? Although this question is difficult to answer, it can be said that: the one that fits best to the given problem. The No Free Lunch (NFL) theorem (Wolpert and Macready, 1997) confirms this argument by proving that there is no algorithm to solve all optimization problems. According to NFL, a metaheuristic could be able to provide satisfying solutions for some problems while different problems could be tough to be optimized by the same algorithm. Also, it should be emphasized that optimization algorithms are not a solution to a problem, but a way of describing how to search for a solution. Thus, the field of optimization has been open to improvements over the years and seems to be present, at least in the near future. These reasons are our motives to propose a new population-based metaheuristic algorithm.

1.1. The Aim of This Thesis

In this study, we propose a new population-based optimization algorithm that is called EMEL which is the acronym for *Exploration of Moving and Ever-shrinking Layers*. EMEL applies a layer building strategy by using a virtual space in order to provide a balance between exploration and exploitation. This approach aims to ease the searching large solution spaces. Virtual space is an artificial search domain which initially has the same range with the actual solution space. Following a reasonable number of iterations, range of virtual space begins to be scaled down and this process is explained in latter chapters. Initial population of solutions is achieved by randomly scattering the instances on the search space. Best candidate solution, i.e. global best, is identified and the virtual space is relocated to centralize the global best solution. The candidate solutions are created in the vicinity of global best in a more relaxed fashion than EMEL's counterparts. EMELs Layer building strategy leads to divide the virtual space into partitions where each of the partitions called as *Layer*. For the search process, each individual selects an arbitrary *Layer* and updates its position to a random place within this *Layer*. Displaced instances are obligated to relocate on a region where the actual solution space and virtual space overlap. At early stages, preserving the range of virtual space same as the original search domain provides an elaborative exploration. After a reasonable number of iterations, range of virtual space progressively shrink as we obtain the information of where the promising regions are located. EMEL focuses more on exploitation towards the end of iterations.

1.2. Our Contribution

The main contributions in this work are that EMEL provides a framework for developing solutions that differs from classical approaches. The classical approaches such as Particle Swarm Optimization (PSO) algorithm (Kennedy and Eberhart, 1995), Grey Wolf Optimizer (GWO) (Mirjalili et al., 2014), and Artificial Bee Colony (ABC) algorithm (Karaboga and Basturk, 2007) create candidate solutions that are heavily affected by other solutions; mostly the global best and generally a few of other individuals. EMEL, on the other hand, produces the candidate solutions in a very relaxed fashion; anywhere inside a randomly selected layer which is centralize the global best. While the generation of random solutions allows for more relaxed exploration and exploitation, EMEL maintains its position on the global best solution trajectory, which leads to better solutions. Additionally, EMEL assures a continuous balance between exploration and exploitation. In layer building strategy, the probability of selecting a layer in the close vicinity of the best solution is higher than elsewhere. Therefore, most of the solutions exploit the inner layers, however, a few of them still explore the distal layers. Hence, while generating better solutions is promoted, meanwhile search of potentially fertile regions naturally continues.

To demonstrate its search abilities, EMEL was tested on a total of 120 benchmarks which consist of sixty standard benchmarks, fourteen CEC 2005 benchmarks, twenty-nine CEC 2017 benchmarks, thirteen constrained benchmarks and four real-life constrained engineering design problems. The benchmark set consisting various characteristics such as shifted/rotated functions, hybrid functions, composition functions and also constrained problems include equality and inequality constraints to be satisfied. Rather than using fixed parameters, design parameters of all algorithms were adjusted for each benchmark to provide fair chances for all algorithms. The results were compared and analysed by utilizing Mann Whitney U statistical test. The results showed that EMEL is superior to its counterparts by achieving high scores in the tests.

1.3. Thesis Organisation

The remaining sections of this thesis work are organized as follows:

The second chapter provides a literature review and covers the significant studies in field of optimization. Also, applications of optimization algorithms are sampled in this chapter.

Examination on search behaviours of the counterparts, the proposed optimization algorithm EMEL, methodology of the conducted experiments and utilized benchmarks are described and given in the third chapter of the thesis.

The results of the experiments, statistical analysis and discussions about the obtained results are demonstrated in fourth chapter.

The last chapter concludes the conducted work, summarizes the results and extends future research directions.

2. PRELIMINARY WORK

Optimization algorithms have been inspired by a wide range of sources through the past fifty years. Nature inspired approaches appear to be the most prominent algorithms studied by researchers. Though most of them derived from the behaviours of creatures in nature, evolution and physics are also frequently serve as significant sources of inspiration. For instance, Simulated Annealing is a prominent physics-based method, which mimics the annealing process in metallurgy (Kirkpatrick et al., 1983). Other well-known algorithms include Genetic Algorithm (Holland, 1992) and Differential Evolution (Storn, 1996) which are thought to be ancestors of modern evolution strategies. Swarm intelligence, on the other hand, is might be the most attractive domain of optimization algorithms owing to explicit opportunities on observation of the creatures. In nature, struggling on survival in itself requires optimal effort and strategy from hunting to foraging and communicating. This effort presents us an intelligent cooperation such as in foraging of bird or fish flocks in Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995), ant colonies in Ant Colony Optimization (Dorigo and Di Caro, 1999), and bees in Artificial Bee Colony (ABC) (Karaboga and Basturk, 2007). Hunting behaviours, on the other hand, are also founded on principals of swarm behaviour such as pumas in Puma Optimizer (Abdollahzadeh et al., 2024), cats in Cat Hunting Optimization algorithm (Ghaedi et al., 2023), owls in Owl Search Algorithm (Jain et al., 2018), and grey wolves in Grey Wolf Optimizer (GWO) (Mirjalili et al., 2014).

While a wide variety of optimization methods have been introduced from the past to the present, their application in real world is also very diverse. Construction site layout planning with risk assessment approach (Oral et al., 2018), optimal scheduling of appliances in smart homes (Jordehi, 2019), energy saving in workflows scheduling (Xie et al., 2023; Yin et al., 2020), designing a wind power generator considering energy storage (R. Wang and Zhang, 2023), optimizing the dispatch problems in terms of generation cost (Duan et al., 2023), automating switching phases of all signalized intersections in an urban transportation network (Liu et al., 2024), approaches for solving multi-objective problems (C. Wang et al., 2017) are some examples of the research topics covered by optimization algorithms.

The literature on optimization is a vast and rich body of work. It is not possible to discuss all of the algorithms that have been proposed in this field. However, the following subsections provides a brief overview of some of the milestone and most recent nature inspired algorithms.

2.1. Counterpart Algorithms of the Proposed Comparative Study

This study presents a comparative work by including some milestone algorithms to the experiment. A collection of algorithms were formed from state-of-the art and well-established algorithms from the literature to carry out a comparative study for the proposed algorithm EMEL. Grey Wolf Optimizer, Artificial Bee Colony, and Particle Swarm Optimization algorithms have been

included in the collection with this motive. Furthermore, Vortex Search Algorithm is added to the set to represent the algorithms that use similar search strategy with EMEL. This section briefly introduce these counterparts before detailing the current optimization approaches.

Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995) is a population-based algorithm that is milestone in field of optimization. This algorithm is inspired by the social behaviours of bird and fish flocks regarding foraging behaviour. Not all individuals possess knowledge regarding the location of the food source. Nevertheless, they pursue the individual closest to the food during the search. All individuals are moving across the search space based on their proximity to the individual nearest to the food and their distance to the position they have previously been closest to the food.

The Artificial Bee Colony (ABC) (Karaboga and Basturk, 2007) algorithm is another swarm-inspired approach. This algorithm mimics the foraging behaviour of honey bees. The first half of the swarm is referred to as employee bees. Employee bees randomly select a food source to visit in order to generate new solutions. Subsequently, the second half of the swarm, known as onlooker bees, arbitrarily selects a food source that was previously visited by an employee bee, taking into account the quality of the food sources. This behaviour suggests that solutions with better fitness values are more likely to be chosen by onlookers. In the ABC algorithm, if a food source fails to provide an improved fitness value after a certain amount of trials, it is abandoned, and a random food source is determined within the search space to use in subsequent iterations.

The Grey Wolf Optimizer (GWO) (Mirjalili et al., 2014) is a prominent population-based meta-heuristic method. This algorithm is derived on the hunting behaviours of grey wolves. Grey wolves inhabit packs and adhere to a strict hierarchy. The alpha wolf leads the pack with the assistance of its subordinates, the beta and delta wolves. The algorithm's architecture simulates the pack's encircling and hunting behaviour of the prey. Alpha, beta, and delta wolves indicate the global best solution and the subsequent two best solutions. New solutions are derived by using these three optimal solutions.

Vortex Search Algorithm (VSA) (Doğan and Ölmez, 2015) is a single-solution based meta-heuristic that is derived from whirling flow of the stirred fluids. This algorithm mimics the gradually shrinking circles of the fluid vortex. Initially a large outer circle is located on the center of the search space. Candidate solutions are randomly scattered around the initial center by using a Gaussian distribution. Subsequent iterations are recursively based on evaluating the best solution, relocating the center around it, reducing the circle by a step size, and regenerating new random solutions.

2.2. Literature Review on Optimization Algorithms

Fu et al. (2024) derived inspiration from survival skills of secretary birds to propose a population-based metaheuristic algorithm: the Secretary Bird Optimization Algorithm (SBOA). Secretary birds survive by persistently foraging for prey and avoiding predators. The survival skills

of secretary birds are applied to address practical optimization challenges. The exploration phase of the algorithm mimics the capture of snakes by secretary birds, but the exploitation phase represents their avoidance of predators. In this phase, secretary birds survey their environment and determine the optimal path to a safe shelter. The two phases are repeated iteratively, considering the termination criterion, to find the optimal solution to the optimization problem.

Ghiaskar et al. (2024) introduced the Polar Fox Optimization Algorithm (PFA), which is derived from natural phenomena. This algorithm focuses on the herding and hunting methods of polar foxes. The polar fox's jumping approach to hunting, which depends on sensitive hearing, is mathematically modelled and applied to optimize processes in different search spaces.

Fang and Cao (2024) developed a novel metaheuristic algorithm, Leaf in Wind Optimization, based on the natural phenomenon of falling leaves in the wind. The proposed method simulates the motion response of leaves in various wind intensities by creating models for light and strong wind-driven blades. The algorithm incorporates wind-induced motion modes of linear translation and spiral rotation to provide a hybrid search framework that is suitable for both strategies. This method enables exploration and exploitation of the search space.

Amiri et al. (2024) presented a new stochastic method called as Hippopotamus Optimization (HO) algorithm. The HO draws inspiration from the natural behaviors of hippopotamuses, displaying a novel approach to metaheuristic research. The HO is theoretically delineated by a three-phase model encompassing mathematically modelled evasion methods, defensive measures against predators, and positional adjustments in aquatic environments such as rivers or ponds.

W. Zhang et al. (2024) introduced a novel meta-heuristic optimization approach, Cleaner Fish Optimization (CFO), inspired by the behavior of cleaner fish. The CFO models the movement patterns of cleaner fish during "cleaning services," noting that females could experience sex change to become males, and establishes two types of positional update. A two-generation cycle operation method is proposed to enhance the optimization process.

Zareian et al. (2024) developed the Bitterling Fish Optimization (BFO) algorithm inspired by the mating behavior of the bitterling fish. The bitterling fish represents a strong model in nature for survival. The bitterling fish employs the oyster spawning method for offspring care. The female bitterling seeks a male who exhibits superior strength compared to other males to select an appropriate mate.

Benmamoun et al. (2024) proposed a new meta-heuristic method known as the Bobcat Optimization Algorithm (BOA), which resembles the natural behavior of bobcats in their habitat. The fundamental concept for BOA originates from the hunting skills employed by bobcats during their pursuit of prey. The BOA theory is outlined and subsequently mathematically represented in two phases: (i) exploration, which models the bobcat's positional alteration as it approaches the prey, and (ii) exploitation, which models the bobcat's positional alteration as it pursues the prey.

H. Zhang et al. (2024) introduced a novel method for optimization utilizing radar technology: the Wave Search Algorithm (WSA). The WSA algorithm utilizes radar technology for its novel algorithmic design incorporating a new starting approach, boundary limitation rules, several enhanced greedy mechanisms, and the gradient information of the optimization problem.

Jia et al. (2024) derived inspiration from common rural methods of fishing and introduced a novel metaheuristic optimization algorithm based on human behavior: the Catch Fish Optimization Algorithm (CFOA). This algorithm resembles the activities of rural fishermen in ponds, usually categorized into two phases: exploration and exploitation. The exploration phase consists of two stages: the individual capture stage, grounded in personal experience and intuition, and the group capture stage, relying on human expertise with tools and collaborative work. During the exploration phase, a shift is occurs from individual investigation to collaborative acquisition. In exploitation phase all fishermen will encircle the shoal of fish and cooperate to extract the remaining fish, employing a collective catch approach.

Hubálovská et al., (2024) presented the Botox Optimization Algorithm, a new metaheuristic derived from the operational principles of botox. The algorithm is designed for addressing optimization challenges through a human-focused methodology. The botox optimization algorithm is developed and mathematically modeled by drawing inspiration from botox procedures, which focus on identifying and treating imperfections to boost aesthetics.

3. MATERIAL AND METHOD

3.1. Search Behavior of the Counterparts

The motivation behind this study is the gap in the literature where traditional optimization methods attempt to generate new solutions that depend on other individuals in the population, especially the global best. PSO is one of the most powerful algorithms that can be considered as the ancestor of this approach. In this method, when updating the current individual's position in the search space, i.e. generating new solutions, the current individual moves on a composite vector formed by influencing the global best, its own personal best and its current velocity (Kennedy and Eberhart, 1995). The amount of velocity's contribution to the composite vector of displacement is determined by a coefficient called inertia weight (Shi and Eberhart, 1998). The Equation 3.1 and Equation 3.2 determine and the Figure 3.1 illustrates how PSO updates position of an individual. In the given equations $\vec{V}$ is the current velocity, w is the inertia weight, $\vec{X}$ is the current position, $\overrightarrow{PB}$ is the personal best, $\overrightarrow{GB}$ is the global best, C_1 and C_2 are learning factors which are usually assumed as 2, $rand(\,)$ is a random number between [0, 1] to provide randomization. $\vec{V}_{new}$ and $\vec{X}_{new}$ are updated velocity and updated position of the current individual, respectively.

$$\vec{V}_{new} = w \cdot \vec{V} + C_1 \cdot rand(\,) \cdot \left(\overrightarrow{PB} - \vec{X}\right) + C_2 \cdot rand(\,) \cdot \left(\overrightarrow{GB} - \vec{X}\right) \tag{3.1}$$

$$\vec{X}_{new} = \vec{X} + \vec{V}_{new} \tag{3.2}$$

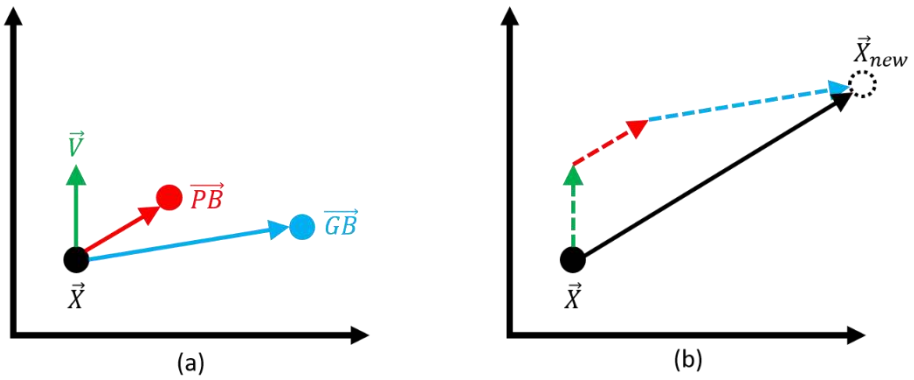

Figure 3.1. New solution generation strategy of PSO algorithm

In this approach, considering different values of $rand(\,)$ provides an area that individuals could update their positions into. However, the displacement is still restricted and could not be manage to land beyond that area. This behaviour prevents exploration of the possible better solutions located around the close vicinity of that area. Most of the subsequent methods inherit this approach in various ways. The aim of the present work is to overcome the limitation in displacement and propose a more relaxed solution generation method.

3.2. The Proposed Algorithm: EMEL

This study presents a novel population-based algorithm using a moving and ever-shrinking virtual search space. This approach intends to eliminate the obstructions resulting from searching large solution spaces. For this purpose, a search space is artificially described apart from the actual problem domain and it is named as virtual search space. Range of virtual search space is initially set to range of actual problem domain. Therefore, both the virtual and actual search spaces fully overlap at the initial stage. Initially, a population of random solutions are scattered on the actual search space. At the beginning of each iteration, the best candidate solution that is called global best (GB) is identified and the virtual search space is re-scaled and re-located to centralize the global best solution. The new solutions are created on the regions where the actual and virtual search spaces overlap. Even though, the creations of new solutions are random, they are encouraged to be located around the global best through the layering mechanism of EMEL. Virtual search space is divided into partitions and each partition is called as *Layer*. For the search process, each individual selects an arbitrary *Layer* and updates its position to a random place within this *Layer*. The layers are bounded by concentric rectangles whose dimensions increase exponentially. While distal layers are vast and sparse, proximal layers are tiny and densely packed around the GB which is the best solution ever encountered. The mentioned *Layer* building strategy will be elaborated in more detail in the subsequent sections of this chapter. Figure 3.2 broadly explains relocation of the virtual search space over the iterations. The pseudo-code of the proposed algorithm is given in Figure 3.3 and a flow-chart of the algorithm is given in the Figure 3.4.

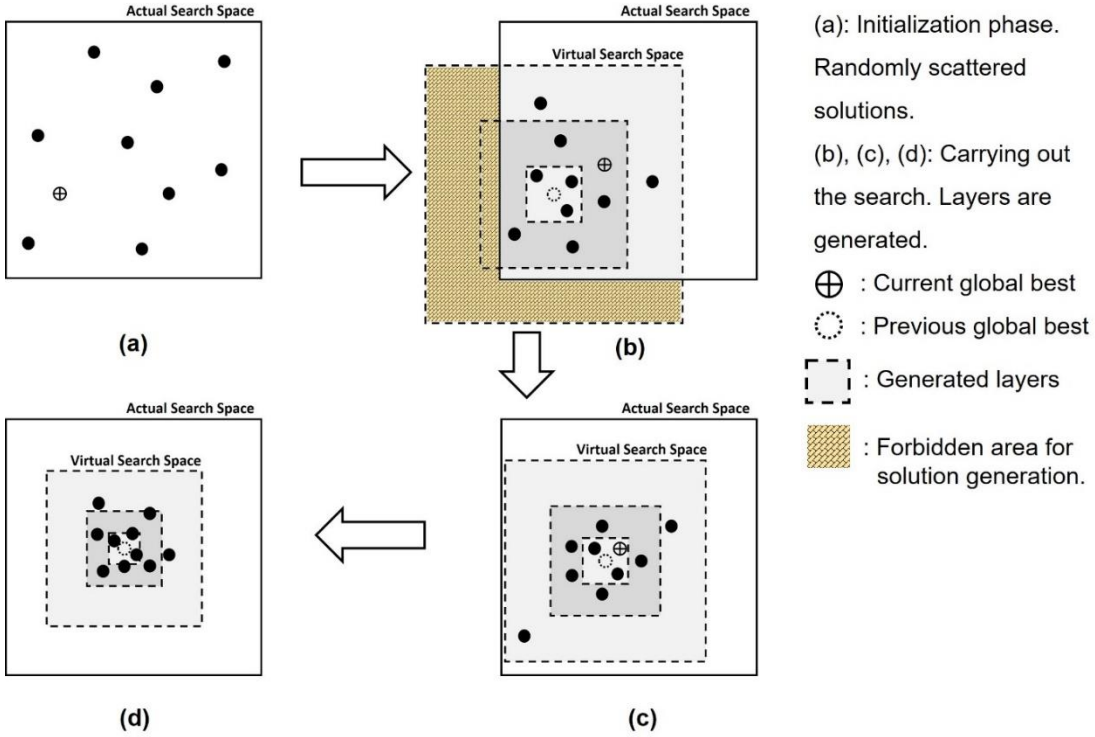

Figure 3.2. Localization and scaling of Virtual Search Space over the iterations.

1: Initialize the population x_i where $i = (1,2, \dots, n)$ and n is total number of individuals

2: Each individual of $x = (x_1, x_2, \dots, x_d)$ and d is total number of dimensions

3: $t = 0$

4: **while** $(t < Maximum\ number\ of\ iterations)$

5: Ensure that all the individuals are within the actual space

6: Find the global best position (GB) among the population

i represents an individual, d represents a dimension

7: **for** $i = 1: Number\ of\ Individuals$

8: **for** $d = 1: Number\ of\ Dimensions$

9: Shift the d^{th} dimension of virtual search space and make the GB_d centre of virtual search space

10: Scale the boundaries at d^{th} dimension of virtual search space by using Equations 3.3 and 3.4

11: Specify the *Layers* over the d^{th} dimension of virtual search space

12: Select a random *Layer* and update the position of i_d to a random place within the selected *Layer*

13: **end-for**

14: **end-for**

15: $t = t + 1$

16: **end-while**

17: GB is the best solution ever found

Figure 3.3. Pseudo-code of the proposed algorithm EMEL

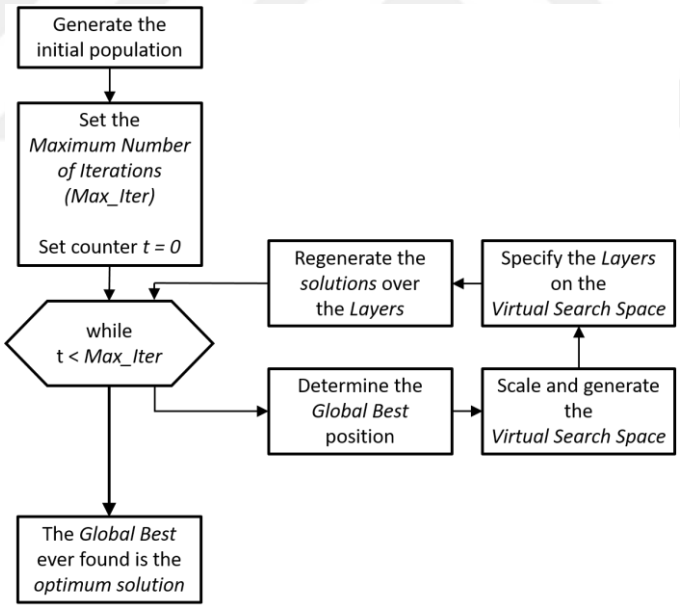

Figure 3.4. Flow-chart of the proposed algorithm EMEL.

At early stages of the search, initial range of virtual search space is preserved to the range of actual search space to allow enough time to explore the space. As the promising regions are discovered, the range of virtual search space is exponentially shrunk at latter stages to provide more convergence and relaxed exploitation. We scale the width of the virtual search space using the variable $C(i)$. Equation 3.3 shows how to calculate $C(i)$, and Equation 3.4 shows how to calculate the virtual search space's boundary values.

$$C(i) = e^{-\frac{i^{\gamma}}{2\sigma^{\gamma}}} \tag{3.3}$$

$${}^{V}_{d}B(i) = {}^{A}_{d}B \cdot C(i) + \varepsilon \tag{3.4}$$

In the given Equation 3.3 and Equation 3.4 i represents the iteration index, $C(i)$ indicates the boundary scaling factor at the i^{th} iteration, e is the Euler's number, σ $(sigma)$ and γ $(gamma)$ are predetermined constants. In Equation 3.4, B indicates the boundary length, the superscripts V and A represents the virtual and actual search spaces, respectively. In most situations, optimization problems have d-dimensional solution space. Hence, the subscript d represents any of the dimensions. As can be seen from ${}^{V}_{d}B(i)$, the virtual space's boundary lengths are evaluated at each iteration i. Note that, in order to obtain the total range of virtual search space, we need to calculate both lower and upper boundaries of it. We will need the total range of virtual search space to specify border limits of *Layers* and this will be explained later on this section. The ε is a predetermined very small real number that prohibits ${}^{V}_{d}B(i)$ to be zero, and it is set to 10^{-300} in this study.

Setting $C(i)$ values close to or equal to 1 in the earlier iterations ensures that the virtual search space width is the same as or extremely close to the actual search space width. Since the virtual and actual search spaces have the same width, solutions can be generated throughout the whole search surface for exploration in the early iterations. In order to enhance the generated solutions at the late iterations, the progressively decreasing $C(i)$ value contributes in shifting the search tendency from exploration to exploitation.

In determining the gamma value, a thorough understanding of its impact on the $C(i)$ values is imperative. The evolution of the $C(i)$ scaling factor over iterations for various gamma values is depicted in Figure 3.5. Gamma plays a pivotal role in influencing the duration of the transition from $C(i)$ values near 1 to $C(i)$ values near 0, indicating a shift from an exploration to an exploitation tendency. With increasing gamma values, C values exhibit a sharp decrease. The utilization of an exceedingly high gamma value, such as 80, results in a $C(i)$ values graph resembling a step function, thereby constraining the algorithm to solely exploration or exploitation. Our intended approach is to avert a rapid transition from exploration to exploitation. The preferred strategy seeks to achieve a balance by gradually increasing the exploitation tendency while decreasing the exploration tendency over iterations. Upon scrutinizing the graphs in Figure 3.5, it becomes evident that the curve with a gamma value of 5 best aligns with this condition. Consequently, we have chosen to employ and recommend for a gamma value of 5 within the framework of Equation 3.3.

Moreover, Equation 3.3 is derived from the normal distribution function for the determination of the $C(i)$ value. The sigma value, representing the standard deviation and indicating the spread of the normal distribution curve, is a critical factor influencing the behavior of C in Equation 3.3. Sigma determines the number of iterations during which C remains close to or equal

to 1, thereby influencing the duration of the exploration phase. The variation in the C scaling factor over iterations for different sigma values, with gamma set to 5, is illustrated in Figure 3.6. Specifically, Figure 3.6 (a) highlights that selecting a small sigma, such as 50, results in an abundance of C values close to 0, while values close to 1 are scarce. Conversely, opting for a large sigma value, such as 300, extends the exploration period but significantly reduces the time allocated for exploitation. The preference for sigma as an adjustable parameter stems from the aim of attaining an optimal curve spread that ensures sufficient durations for both exploration and exploitation, aligning with the requirements of the problem. Therefore, the course-to-fine method was employed in this study to explore the [50, 300] range and determine the optimal sigma value for each benchmark.

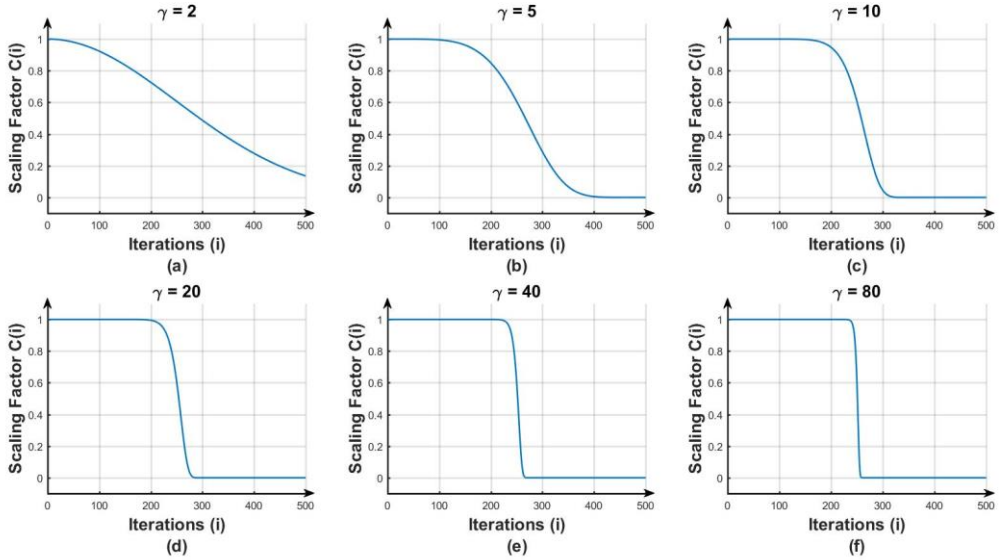

Figure 3.5. $C(i)$ curve for various Gamma values. Sigma value is set to 250 in given curves as an example case.

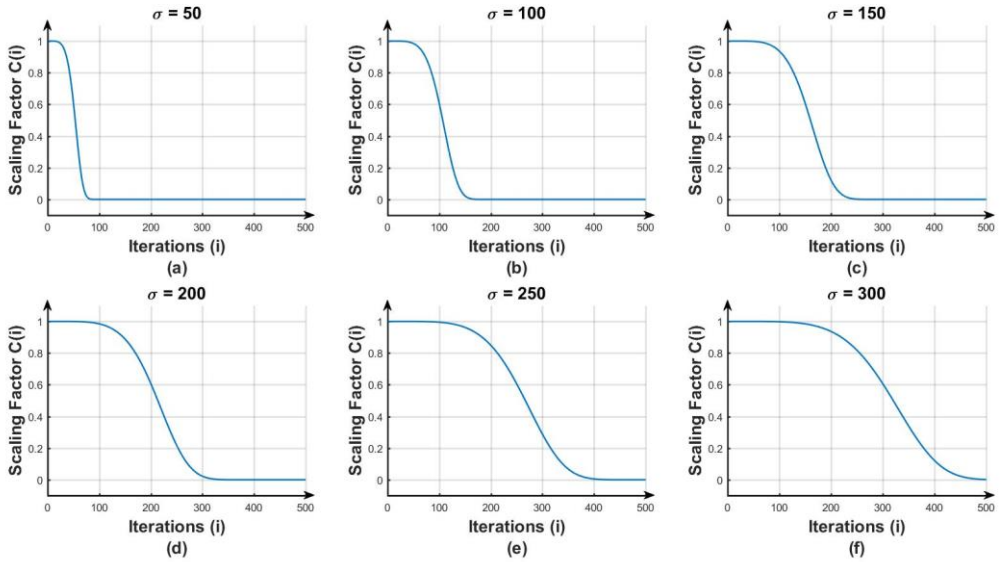

Figure 3.6. $C(i)$ curve for various Sigma values. Gamma value is set to 5 in given curves.

Having decided the scale of virtual search space, it should be centralized around the global best solution. For this reason, all the candidate solutions are evaluated for goodness by using a fitness function specific to the optimization problem at hand. The best solution is called global best (GB) and the origin of virtual search space is shifted to the global best solution's position. It is intended to create new solutions around the global best in the quest of producing better individuals. However, the global best could be a local optimum point and this intention may result with an early convergence to a local optimum. Therefore, a mechanism should be devised which promotes to create solutions tightly positioned around global best for exploitation, as well as allowing to create new individuals within the valid regions of actual search space for exploration. This can be accomplished by defining *Layer*s that are smaller within the close neighbourhood of global best, and larger at the outer boundary (See Figure 3.8).

Specification of the *Layer*s is inspired by the logarithmic spiral. The distances between the turnings of a logarithmic spiral increase in geometric progression (Darling, 2004). Distances of *Layer*s mimic this manner to subdivide the space. The logarithmic spiral is drawn over a domain by using two dimensions, but optimization problems could have more than two dimensions. Hence, the inspired manner is simplified so that can be easily applied on each single dimension of any problem. Border limits of *Layer*s are calculated by using Equation 3.5.

$$Lim(L_d) = \frac{Range_d^i}{2} \cdot \frac{P^{L_d}-1}{P^N-1} \tag{3.5}$$

In Equation 3.5 L_d indicates the identification number of an arbitrarily selected *Layer* on d^{th} dimension and $Lim(L_d)$ is the upper border limit of the selected *Layer*. Note that GB is always at the origin of the virtual search space. Therefore, upper border limit of a Layer (L_d) is measured starting from the origin. $Range_d^i$ is the total range of d^{th} dimension of virtual search space at i^{th} iteration, N is the total number of specified *Layer*s and P is a coefficient. The N and P are predetermined constant values. Distance of each *Layer* is found to be P times of the previous one, except the first one (Figure 3.7).

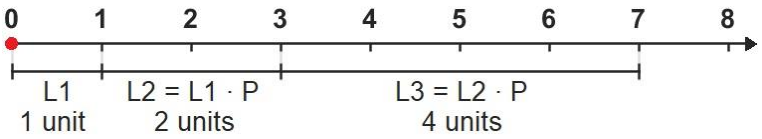

Figure 3.7. Range calculation of *Layers* at a single dimension. Assume that P = 2. L: Layer.

As mentioned earlier in this section, an individual selects a random *Layer* and updates its position to random place within the borders of the selected *Layer*. Remember that border limit of any *Layer* is calculated starting from the origin where the GB is located. Displacement of individuals is formulated as in Equation 3.6 in order to enable positions to be moved into both sides of GB.

$$\begin{cases} if\ \ rand()_d < 0.5\ then,\ -Lim(L_d) \le X_d^{new} \le -Lim(L_d - 1) \\ if\ \ rand()_d \ge 0.5\ then,\ Lim(L_d - 1) \le X_d^{new} \le Lim(L_d) \end{cases} \tag{3.6}$$

In Equation 3.6, just as in Equation 3.5, L_d indicates the identification number of an arbitrarily selected *Layer* on d^{th} dimension and $Lim(L_d)$ is the upper border limit of the selected *Layer*. X^{new} indicates the updated position of an individual on d^{th} dimension. It is obvious on Figure 3.8 that individuals will move over the area of selected *Layer* excluding the areas hold by previous *Layers*. Furthermore, Figure 3.8 illustrates a sample of two dimensional search space where an individual selects the arbitrary *Layer*s as (3, 2) along the axis X_1 and X_2, respectively. Therefore, this individual, by using Equation 3.6, will be allowed to update its position to any random place in one of the feasible areas for solution generations in Figure 3.8.

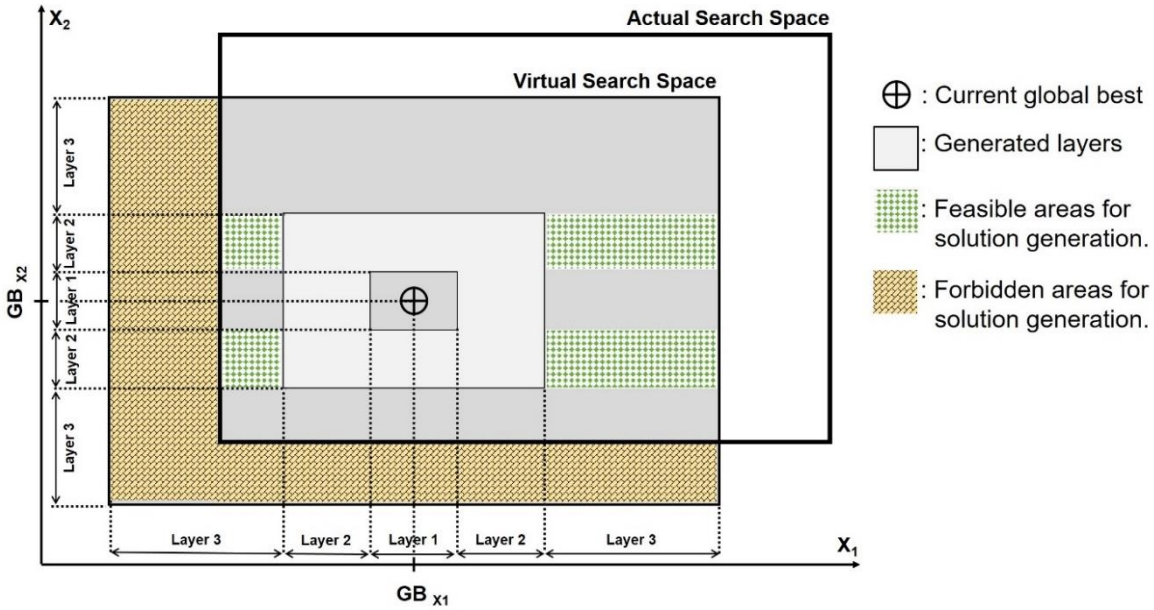

Figure 3.8. Carrying out random *Layer* selection for any individual to relocate its position into. Example case: for an individual *Layer* #3 and #2 are selected for dimensions X_1 and X_2, respectively.

The *Layer* strategy is proposed aiming a relaxed and more randomized search in order to find more feasible areas. As can be seen back in Figure 3.8 the *Layers* have smaller range around the close vicinity of GB and gradually enlarge through outer boundaries of virtual search space. Since more *Layers* are located around GB, the probability of individuals to displace around GB increases. Figure 3.9 depicts a numerical example of specifying the *Layers*. In the given figure $P = 1.5$, total number of *Layers* is set to 5 and total range of dimension X_1 is equals to 20. Considering the given figure, it could be argued that the probability of an individual displacing into the first four out of five *Layers*, i.e. 80% of the time, provides an opportunity to update position into the area that covers 62%

of the search space around GB. Furthermore, the first three *Layers* construct a closer vicinity of GB via covering 36% of search space by having a chance to be displaced into in three out of five *Layers*, i.e. 60% of the time.

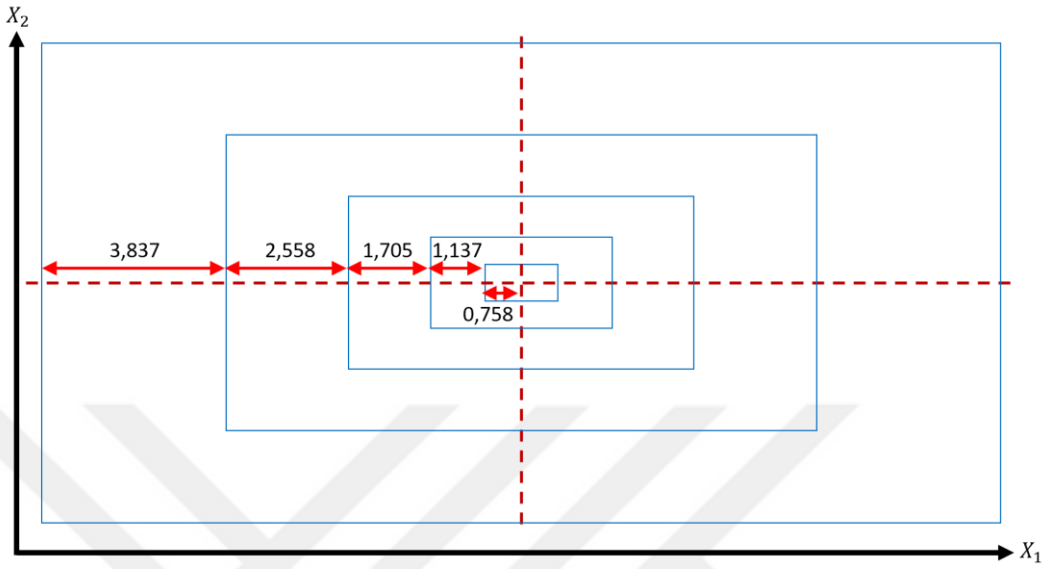

Figure 3.9. A numerical example of specifying the *Layers*.

Figure 3.10 provides a closer examination on effects of *Layers*. In the given figure, total range of dimension X_1 is set to 20 and the depicted *Layer* ranges are evaluated for one side of GB as in Figure 3.9. For ease of illustration pie chart is preferred instead of axis. Abbreviations *L1* - *L7* represent the *Layers* and the range of relevant *Layer* is given in brackets. Figure 3.10 (a) discusses the effect of total number of *Layers* in comparison with Figure 3.9, which uses equal P value and total range. According to the Figure 3.10 (a), 65% of the search space that located around GB holds a chance of six out of seven *Layers*, i.e. 86% of the time, to host updated individuals. A more detailed inspect figures out that 41% of the area holds a closer vicinity of GB with a chance of 71% to be visited, i.e. five out of seven *Layers*.

Figure 3.10 (b), on the other hand, represent a comparison according to Figure 3.10 (a). In Figure 3.10 (b) value of design parameter P is set to 2 and this led a denser area located around GB consist of six out of seven *Layers*. According to Figure 3.10 (b), 50% of the search space is located around GB and 86% of time the individuals could be displaced into these inner *Layers*. Furthermore, in comparison with Figure 3.10 (a), Figure 3.10 (b) shows that 71% of individuals could update their positions into closer area to the GB that covers 25% of the search space.

Considering the given examples, this study aims to establish balanced values of design parameters. In this context, values of total number of *Layers* and the P are meant to be adjusted for each separate benchmark in order to provide a proper exploration nature for proposed algorithm EMEL.

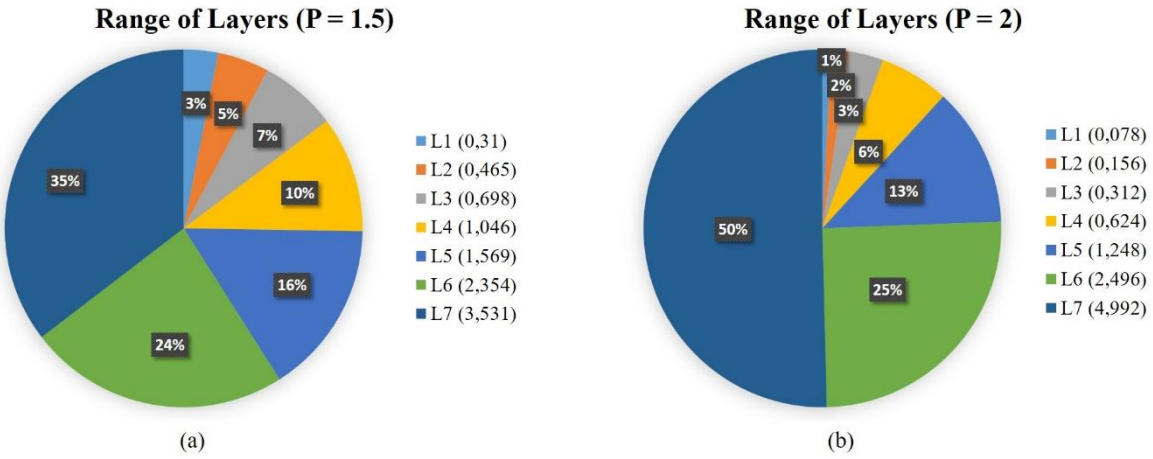

Figure 3.10. Various numerical examples of specifying the *Layers*.

3.3. Methodology of the Experiments

In order to validate a new algorithm, the researchers have used benchmarks that present various challenges similar to real-world problems. Unlike some of the real-world problems, the benchmarks whose optimum solutions are known provide an opportunity to test the outcomes of the proposed algorithm and to compare the results with notable algorithms exist in the literature. In this study total of 120 benchmarks are utilized for the experiments and descriptions of these benchmarks are given in Section 3.5 of this thesis.

There are numerous nature inspired optimization algorithms that present various performances for the benchmarks. A collection of algorithms were formed from state-of-the art and well-established algorithms from the literature to carry out a comparative study for the proposed algorithm EMEL. GWO, ABC, and PSO have been included in the collection with this motive. Furthermore, VSA is added to the set to represent the algorithms that use similar search strategy.

The common control parameters of the algorithms consist of population size, number of maximum iterations and iteration termination criteria. These parameters are set to the same values for all algorithms to satisfy fair chances in the experiments. The population size has been set to thirty and the number of maximum iterations has been set to five hundred as the only termination criterion. Also, same seed values were used for random number generator of each separate run. Hence, the same initial population was reserved for all algorithms at each run and also the experiments were made repeatable.

Algorithms, also, have their own control parameters which also required to be set. The control parameters of the algorithms are usually set as given in the articles that they have been presented. It is mentioned that an algorithm is not able to provide satisfying solutions for all problems according to the No Free Lunch theorem. Hence, as the number of benchmarks increases and their characteristics diverse in experiments, it will get harder for an algorithm to achieve an overall success by using the same values of control parameters. Besides, it should be considered that the benchmark set may have fitting characteristics in favour of some algorithm(s) while being tough to other(s). The

hard path have been chosen for this study that many of the previous researchers avoided because of the extensive workload. The values of the control parameters have been adjusted separately for each benchmark to provide fair chances to the algorithms in the experiments.

Coarse-to-fine parameter tuning method (Moshkelgosha et al., 2017) has been utilized for tuning the control parameters of the algorithms in order to find the values providing the best optimization results for each benchmark. In this method the domain range of a parameter is discretized into a coarse grid of values. A search along these values provides an observation about how the increment/decrement of a parameter value affect the performance of an algorithm. After the initial search, the vicinity of the most promising value is discretized into another grid for a finer search around it. A better fitting parameter value is obtained in the finer search, and the process recursively continues.

Ten grids have been set for each parameter of algorithms. Assume that there are N control parameters to be tuned. Since the domain ranges have been divided into ten grids including the minimum and maximum values, there will be 11^N candidate parameter set for an algorithm at each level of the coarse-to-fine tuning. Furthermore, in order to eliminate the effect of randomness, the algorithms were run multiple times with same number of runs using each candidate parameter set. The parameter set providing the best median value out of performed multiple runs is selected as the best one for related search level. A finer search is carried out by using these parameters, and the process retains. The one, which is obtained after the last searching level, is assumed as the best parameter set of the algorithm for related benchmark. The described exhaustive search process was separately performed over all benchmarks for each algorithm.

Following the course-to-fine searching procedure, the resulting values of the algorithm design parameters for all benchmarks are provided in Appendices A, B, C, D, and E. The remaining experiments for statistical analysis are performed using these values of design parameters for the relevant benchmarks.

3.4. Sensitivity Analysis of the Tuned Parameters

The number of tuned parameters varies for algorithms. Table 3.1 shows the tuned parameters of the experimented algorithms and the search ranges for relevant parameter. Brief descriptions about the tuned parameters are given below following the provided table.

Table 3.1. Tuned parameters of experimented algorithms

Algorithms	Tuned parameters	Range of values	Type of values
EMEL	N	[10, 150]	Discrete
	P	[1.1, 3]	Continuous
	σ	[50, 300]	Continuous
ABC	L	[10, 150]	Discrete
	N	[1, #of dimensions /2]	Discrete
	T	[1, 10]	Continuous
PSO	ω	[0, 5]	Continuous
	C_1	[1, 5]	Continuous
	C_2	[1, 5]	Continuous
GWO	$\vec{a}$	[1, 5]	Continuous
	$\vec{C}$	[1, 5]	Continuous
VSA	x	[10e-3, 1-10e-3]	Continuous

According to Table 3.1 the given parameters *N* and *P* of EMEL algorithm are utilized in Equation 3.5 of this thesis. The parameters *N* and *P* indicate the number of specified *Layers* and the value of coefficient influencing the range of *Layers*, respectively. The σ value is given in Equation 3.3 and influences the time when the scaling factor begins to be effective, i.e. the time when the algorithm starts to shift from exploration through exploitation.

Karaboga and Basturk (2007) configures the ABC algorithm to explore the vicinity of a food source at least *'limit'* trials before abandoning it if it could not be improved. The *'limit'* value is set to 100 by default in ABC. In this study, the number of trials is denoted as *L*. Furthermore, ABC algorithm allows the solution to be contributed by neighbours to be improved. This contribution affects only one dimension of a solution by default. In this study, different number of dimensions allowed to be contributed and number of allowed dimensions is indicated by *N*. The number of modified dimensions, *N*, is allowed to be up the half of the dimensions of related benchmark. The amount of contribution to a dimension, on the other hand, is influenced by ϕ as described in Equation 2.2 of Karaboga and Basturk (2007). The default ABC generates random values of ϕ between [-1, 1]. In this study, ϕ is meant to be a random number between [-*T/2*, *T/2*].

PSO, on the other hand, is influenced by a solution's own personal best value found over iterations as well as the global best value of the search space ever found. Contribution of personal best and global best values are controlled by the C_1 and C_2 and both values are originally set to 2 by Kennedy and Eberhart (1995). Inertia weight, ω, aims a balanced search between exploration and exploitation by utilizing current position of a solution. The initial introduction of inertia weigh was by Shi and Eberhart (1998) and initial value range is suggested between [0.9, 1.2].

The design parameters of GWO are consist of $\vec{a}$ and $\vec{C}$. The parameter $\vec{a}$ is initially set to 2 and is controls exploitation by decreasing from its initial value to 0 over the course of iterations. Exploration is favoured by $\vec{C}$ and its default value is also 2. $\vec{C}$ preserves its value during iterations and meant to contain random values between [0, 2].

The x value of VSA is used in Equation 11 of Doğan and Ölmez (2015) and influences the step sizes over the iterations while reducing the range of the search space. The initial value of x originally is set to 0.1.

Effects of adjusting the value of design parameters are separately analysed by utilizing Friedman test. The Friedman test is a nonparametric test used for paired data samples to compare the results produced by different methods (Gençal and Oral, 2022). The null hypothesis H_0 for this test states that "*there is no difference between the compared selection methods*", while the alternative hypothesis H_1 suggests that "*there is difference between compared selection methods*". The significance of the test is determined by p values. If the calculated p value is smaller than the specified level of significance, which is $\alpha = 0.05$, it is concluded that there is statistically significant difference between the compared data groups.

To test the effect of a parameter, a grid of values is formed in the search range of the relevant parameter from back in Table 3.1. The values of remaining design parameters of the relevant algorithm are kept constant and the algorithm was run twenty-five times for tested parameter with each of the values taken from the previously formed grid. Each of these twenty-five run's results is considered as one data group. All data groups of tested parameter are compared using Friedman test aiming to produce a p value smaller than the level of significance $\alpha = 0.05$. Thus, it could be concluded that data groups consisting of results generated by different values of the tested parameter are significantly differs from each other. This experiments are repeated for each tested parameter separately over total of 120 benchmarks. Table 3.2 reports the total number of benchmarks out of 120, where the tested parameter managed to produce p value indicating the significant difference between varying values of it. According to given table it is clear that it is worth searching for an adjusted value of these design parameters.

Table 3.2. Results of Friedman test applied on data groups obtained by different values of tested design parameter

Algorithms	**Separately analysed parameters**	Number of p values less than 0.05 out of 120 benchmarks
EMEL	N	104
	P	89
	σ	117
ABC	L	110
	N	23
	T	114
PSO	ω	119
	C_1	118
	C_2	118
GWO	$\vec{a}$	113
	$\vec{C}$	94
VSA	x	119

3.5. Benchmark Set of the Experiments

The proposed algorithm EMEL was tested on a large benchmark set consisting of 120 benchmarks that have various characteristics such as shifted/rotated, hybrid, composition, constrained functions and also real-world engineering design problems. This set includes sixty standard global optimization functions including both unimodal and multimodal structures which are experimented in many works in literature (Jamil and Yang, 2013; Karaboga and Akay, 2009). Fourteen of shifted/rotated, hybrid and composition functions are taken from CEC 2005 benchmarks (Suganthan et al., 2005) and twenty-nine of them are taken from CEC 2017 benchmarks (Awad et al., 2016). Additionally, in order to present constraint handling abilities of the algorithms thirteen constrained optimization functions (Karaboga and Akay, 2011) and four constrained real-world engineering design problems are included in the benchmark set.

The benchmark set includes functions that have varying characteristics in purpose of measuring the exploration skills of an algorithm in searching the global optimum. This section summarizes the basic characteristics of the benchmarks in order to consider during the evaluation of the experimental results. A multimodal function is a function that involves more than one local optimum. A local optimum is a best solution in a narrow vicinity of itself but not globally satisfies the problem. An efficient algorithm is expected to avoid stagnation on local optimums by its exploration abilities. Beside multimodality, shifting and rotating the global optimum from the origin increase the complexity of the search space and forces any algorithms' explorations skills. Furthermore, hybrid and composition benchmarks challenge the exploration abilities by combining multiple basic functions with different properties to generate various search spaces involving many local optimums.

Constrained problems, on the other hand, involve many equality and/or inequality constraints. In this study the commonly used thirteen constrained benchmarks (Karaboga and Akay, 2011) and four of constrained problems which are welded beam (Cagnina et al., 2008; Ragsdell and Phillips, 1976), speed reducer (Cagnina et al., 2008; Golinski, 1970), piston lever (Gandomi et al., 2013), and I-beam (Gandomi et al., 2013; Gold and Krishnamurty, 1997) design problems have been experimented. Detailed description about the engineering design problems is provided in the Appendices F, G, H, and I, respectively. The constrained problems are optimized considering the feasibility in terms of constraints while enhancing the fitness of the solutions. A feasible solution is an individual that is not violating the borders specified by constraints. Constrained handling is a challenging task and many constrained handling methods are presented in literature (Coello Coello, 2002; Deb, 2012). Penalty method is one of the easily implemented methods. This method transforms the constrained problem into unconstrained by adding a penalty term to the fitness value according to amount of constraint violations (Arora, 2015; Deb, 2012).

The penalty method have been utilized due to constraint handling. The penalty method handles constraints by adding a penalty term to the fitness value. The amount of penalization is

calculated considering the violation level of the constraints. A basic formulation is given in Equation 3.7 (Arora, 2015). In this formulation $f(x)$ refers to the standard objective function of the benchmark and $F(x)$ is the penalized objective function. The terms h and g represent the equality and inequality constraints, respectively. The term R is a large number which is used to penalize the fitness value in case of violations (Deb, 2012).

$$F(x) = f(x) + R \cdot \sum_{j=1}^{n} h_j^2(x) + R \cdot \sum_{i=1}^{m} \langle g_i(x) \rangle^2 \tag{3.7}$$

The term R can be used as a constant large number like 10^{20} (Deb, 2012), however, it would be better to begin with a small value and increasing later (Arora, 2015). Thus, we multiplied the constant 10^{20} by another $M(t)$ value in order to calculate the term R which is individually used in each iteration. The multiplier term $M(t)$ and R are calculated as in Equation 3.8 and Equation 3.9, respectively. The term t represents the number of current iteration and $MaxIter$ is the total number of iterations. The term $M(t)$ eases the exploration at initial steps by providing a smooth penalty amount. At later iterations, on the other hand, the exploitation gets tighter as the term $M(t)$ increases.

$$M(t) = \frac{t}{MaxIter} \tag{3.8}$$

$$R = M(t) \cdot 10^{20} \tag{3.9}$$

3.6. Mann Whitney U Test

Mann Whitney U is one of the statistical analysis methods recently popular in evaluating the results in the field of optimization. Mann Whitney U is a nonparametric statistical test which is implemented on independent small size groups. It is used to test whether the samples are derived from distributions with equal medians or not (Heiberger and Holland, 2015; Heumann et al., 2016).

In this study, all algorithms are executed within same number of runs on each benchmark, utilizing parameter settings optimized through a course-to-fine parameter search. Number of function evaluations were same in all experiments due to using same number of search agents and maximum number of iterations with no other termination criteria for all algorithms. The outcomes of each optimization run were carefully documented, resulting in a comprehensive dataset. This dataset is facilitated pairwise comparisons of algorithms for each benchmark.

Pairwise comparisons of the competing algorithms were made using the data, i.e. the results of multiple runs of these algorithms, for each benchmark handled individually. The Mann Whitney U test was employed to assess whether there existed a statistically significant difference between the medians of two independent data groups. In cases where a statistically significant difference was observed between algorithm pairs for a given benchmark, the algorithm with the smaller median was

deemed more successful for that specific benchmark. Conversely, if no statistically significant difference was found between algorithm pairs, the performance of the compared algorithms was denoted as "*no difference*" for that benchmark. Executing these procedures for all available benchmarks resulted in a score table illustrating the number of benchmarks for which each algorithm demonstrated superior performance overall, along with the number of benchmarks where no statistically significant difference was observed.

4. RESULTS AND DISCUSSIONS

Meta-heuristic algorithms run based on randomness; creation of initial solutions and modifications on solutions are made randomly. For this reason, an algorithm is run multiple times on the same problem to provide that the obtained success is not by chance. The statistical values such as best, mean, standard deviation, median etc., are calculated from this batch of results. Some or all of these values are presented as performance measures of the algorithms for a related problem. Some studies highlight the results that outperform the others for each benchmark. In such comparisons there might be slight difference between the results. Hence, using a single result value to consider one algorithm is superior to other is a controversial issue. In recent years, beyond showing the existence of difference between results, the studies focus on determining how significant the difference is. In this context, the statistical analysis has become more popular and important in the field of optimization (Mattos et al., 2021). For this reason, this study utilizes Mann Whitney U test in order to analyse and present the obtained results.

Tables 4.1, 4.2, 4.3, 4.4 and 4.5 present distinct score tables derived from pairwise comparisons among all algorithms for standard benchmarks, CEC 2005 benchmarks, CEC 2017 benchmarks, constrained benchmarks and the real-world design problems, respectively. While arranging a table, an algorithm from a row is taken as reference and another one from column is taken as opponent (*Opp.*) to compare the obtained scores according to the Mann Whitney U test. The given numbers under the column of *Opponent Algorithm* are the number of benchmarks that shows the pairwise performance comparison of the algorithms. The number aligned to the name of the reference algorithm refers to the number of benchmarks which reference algorithm achieves significantly better results compared to opponent algorithm, and the number aligned to the abbreviation of *Opp.* is vice versa. The number aligned to the *No Difference* presents the number of benchmarks in which there is no significant difference between two algorithms.

4.1. Results Based on Standard Benchmarks

Compared algorithms used the design parameters that given in Appendix A in experiments of sixty standard benchmarks. Values of common control parameters for all algorithms were set same in order to provide fair chances in the experiments. The population size were set to thirty and the number of maximum iterations were set to five hundred without other termination criterion. All algorithms were run twenty-five times on each benchmark to construct the dataset. Also, same seed values of random number generator of each separate run reserved the same initial population for all algorithms at each run.

Table 4.1. Mann Whitney U test comparison results based on standard benchmarks

60 STANDARD BENCHMARKS						
			Opponent Algorithm			
			GWO	**ABC**	**PSO**	**VSA**
Reference Algorithm	**EMEL**	EMEL is better	33	35	21	29
		No Difference	9	17	23	22
		Opp. is better	18	8	16	9
	GWO	GWO is better		20	20	24
		No Difference		7	12	9
		Opp. is better		33	28	27
	ABC	ABC is better			13	22
		No Difference			19	17
		Opp. is better			28	21
	PSO	PSO is better				27
		No Difference				19
		Opp. is better				14

Table 4.1 shows the pairwise comparison results of algorithms over sixty standard global optimization functions. The results clearly states that EMEL significantly outperforms its opponents. EMEL manages being superior to its competitors despite the existence of competitive results. VSA, the one with the most similar search strategy, have been competed on substantial number of functions. However, the number of functions that EMEL outperforms make this competition negligible.

ABC is one of the most tough optimization algorithms in literature. Even so, the optimization power of ABC could not achieve a satisfactory competition. EMEL is significantly better at more than half of the functions, while ABC achieves success only at eight functions.

GWO is the opponent which the most statistically significant difference appears between algorithms. The number of functions that marked as *no difference* is considerably less than the other algorithms. Although GWO outperforms EMEL in many functions, EMEL has better optimized almost twice as many functions than GWO.

PSO is one of the milestones in field of optimization and it has been the most challenging opponent in experiments of this benchmark set. The results have the highest amount of *no difference* for PSO. The scores of EMEL and PSO are close, however, the difference between them is in favour of EMEL.

4.2. Results Based on CEC 2005 Benchmarks

The design parameters of the compared algorithms were used as in Appendix B in experiments of fourteen benchmarks that taken from CEC 2005. Common control parameters were given the same values to provide fair chances in the experiments. The population size were set to thirty and the number of maximum iterations were set to five hundred without other termination criterion. All algorithms were run twenty-five times on each benchmark to construct the dataset.

Also, same seed values of random number generator of each separate run reserved the same initial population for all algorithms at each run.

Table 4.2. Mann Whitney U test comparison results based on CEC 2005 benchmarks

14 BENCHMARKS OF CEC 2005						
			Opponent Algorithm			
			GWO	**ABC**	**PSO**	**VSA**
Reference Algorithm	**EMEL**	EMEL is better	9	10	11	9
		No Difference	3	3	3	4
		Opp. is better	2	1	0	1
	GWO	GWO is better		6	2	4
		No Difference		2	6	4
		Opp. is better		6	6	6
	ABC	ABC is better			6	4
		No Difference			2	2
		Opp. is better			6	8
	PSO	PSO is better				3
		No Difference				4
		Opp. is better				7

Table 4.2 shows the pairwise comparison results of algorithms over fourteen benchmarks of the CEC 2005. Comparing with results of the sixty standard benchmarks, superiority of EMEL is more obvious in context of pairwise comparisons of CEC 2005 benchmarks. The first thing that stands out is that EMEL is superior on at least nine out of fourteen benchmarks against each opponent. There are only a few competitive results but EMEL still manages being superior to its competitors. Contrary to standard benchmarks, the competition between EMEL and VSA is almost disappeared for CEC 2005 benchmarks. Even though VSA is merely successful, the superiority of EMEL is clear with no doubt. ABC, on the other hand, lost its competing abilities against EMEL. Even VSA could compete against EMEL slightly better than ABC. Also, GWO is the most competitive algorithm in CEC 2005 benchmarks but this effort still not satisfactory since EMEL is significantly superior to it. On the other hand, tough competition of PSO becomes absent for the second category functions. EMEL achieves the most significant difference of these benchmarks against PSO.

4.3. Results Based on CEC 2017 Benchmarks

Compared algorithms used the design parameters that given in Appendix C for experimenting the thirty-nine CEC 2017 benchmarks. Values of common control parameters for all algorithms were set same in order to provide fair chances in the experiments. The population size were set to thirty and the number of maximum iterations were set to five hundred without other termination criterion. All algorithms were run thirty times on each benchmark to construct the dataset. Also, same seed values of random number generator of each separate run reserved the same initial population for all algorithms at each run.

Table 4.3. Mann Whitney U test comparison results based on CEC 2017 benchmarks

29 BENCHMARKS OF CEC 2017						
			Opponent Algorithm			
			GWO	**ABC**	**PSO**	**VSA**
Reference Algorithm	**EMEL**	EMEL is better	14	12	11	17
		No Difference	9	11	13	10
		Opp. is better	6	6	5	2
	GWO	GWO is better		6	7	13
		No Difference		8	8	6
		Opp. is better		15	14	10
	ABC	ABC is better			12	15
		No Difference			10	5
		Opp. is better			7	9
	PSO	PSO is better				14
		No Difference				8
		Opp. is better				7

Table 4.3 demonstrates the results of pairwise comparison of the algorithms over twenty-nine CEC 2017 benchmarks. GWO displayed one of the highest number of functions that have significantly better results than EMEL among the tested algorithms. Although GWO outperforms EMEL in six of functions, EMEL is still superior to GWO by achieving significantly better results in fourteen of functions. Also, the number of functions whose optimization results are marked as *No Difference* is considerably less than the other algorithms. This shows significant polarization in outcomes comparing to the results of other algorithms. It can be concluded that EMEL outperforms GWO, especially, if the search space is challenging such as in CEC 2017 benchmarks.

Superiority of ABC over EMEL is considerably high among the tested algorithms. Its results are significantly better than EMEL in six of functions. Furthermore, ABC generates less polarization, which leads EMEL having significantly better results in twelve of functions. Though ABC challenges EMEL slightly more than GWO, EMEL is still an obvious choice against ABC.

The results have the highest amount of *No Difference* in functions for PSO. However, EMEL still manages having more than twofold significantly better results than PSO. PSO's performance is slightly less than GWO's and ABC's with being superior in five functions. Considering ABC and GWO, PSO is the only one causes EMEL's performance to place under twelve. However, in spite of power of PSO, there is strong confidence to favour EMEL as a safe choice as an optimization tool comparing to PSO.

Regardless of their strategic similarity, EMEL's performances is distinctively superior to VSA. VSA produces significantly better results than EMEL in two of functions. EMEL, on the other hand, is far superior to VSA by achieving significantly better results in seventeen of functions. The superiority of EMEL over VSA is clear without a doubt.

4.4. Results Based on Constrained Benchmarks

The design parameters of the compared algorithms were used as in Appendix D in experiments of thirteen constrained benchmarks. The experimented competitor algorithms had fair chances in experiments due to setting same values of common control parameters. The population size were used as thirty and the number of maximum iterations were set to five hundred without other termination criterion. All algorithms were run twenty-five times on each benchmark to collect the data. Also, the random number generator were seeded with same values for each separate run in order to reserving the same initial population for all algorithms at each run.

Table 4.4. Mann Whitney U test comparison results based on constrained benchmarks

13 CONSTRAINED BENCHMARKS						
			Opponent Algorithm			
			GWO	**ABC**	**PSO**	**VSA**
Reference Algorithm	**EMEL**	EMEL is better	9	8	2	6
		No Difference	3	3	7	6
		Opp. is better	1	2	4	1
	GWO	GWO is better		3	1	3
		No Difference		4	3	4
		Opp. is better		6	9	6
	ABC	ABC is better			2	3
		No Difference			3	5
		Opp. is better			8	5
	PSO	PSO is better				7
		No Difference				6
		Opp. is better				0

EMEL successfully proves its exploration and exploitation abilities on former benchmarks that have challenging search spaces. Constraint handling, on the other hand, is also must be satisfied by an optimization algorithm due to the problems which demand extended criterions beyond capturing the optimum solution. It would be fair considering the constrained functions tougher than benchmarks chasing only the optimum solution. Constrained functions might have multiple equality and inequality constraints to be satisfied. Therefore the search spaces of such functions more challenging due to the obligation that the found solutions must be in feasible areas in terms of meeting the constraint requirements.

In this context, it can be argued using Table 4.4 that comparison results based on constrained functions resemble the formers, except the competition of PSO. The results show that EMEL readily manages outperforming ABC and GWO. According to table, ABC and GWO have the smallest count of *No Difference* along with a vast polarization in comparison with EMEL. VSA, on the other hand, differs from ABC and GWO in terms of obtaining a greater number of *No Difference* than them. However, even if VSA could manage reducing the polarization, EMEL still clearly dominate by being superior to VSA on six of the constrained benchmarks.

PSO must be highlighted for this context due to its competition finally comes to a superiority to EMEL. As can be seen in the Table 4.4 PSO prevents a difference on seven out of thirteen benchmarks. EMEL outperforms PSO on two benchmarks, however, PSO manages being superior by capturing four of the benchmarks.

4.5. Results Based on Engineering Design Problems

The design parameters of the compared algorithms were used as in Appendix E for engineering design problems. Fair chances are provided to algorithms in experiments by using same values for common control parameters. The population size were used as thirty and the number of maximum iterations were set to five hundred without other termination criterion. All algorithms were run twenty-five times on each benchmark to collect the data. Also, the random number generator were seeded with same values for each separate run in order to reserving the same initial population for all algorithms at each run.

Table 4.5. Mann Whitney U test comparison results based on engineering design problems

CONSTRAINED ENGINEERING DESIGN PROBLEMS						
			Opponent Algorithm			
			GWO	**ABC**	**PSO**	**VSA**
Reference Algorithm	**EMEL**	EMEL is better	2	2	1	4
		No Difference	2	1	3	0
		Opp. is better	0	1	0	0
	GWO	GWO is better		1	1	4
		No Difference		0	1	0
		Opp. is better		3	2	0
	ABC	ABC is better			2	3
		No Difference			0	0
		Opp. is better			2	1
	PSO	PSO is better				3
		No Difference				1
		Opp. is better				0

The engineering design problems are also have several equality and inequality constraints to be satisfied. Table 4.5 shows the comparison results of the constrained engineering design problems. It could be concluded that EMEL's overall performance on the constrained engineering design problems is similar with the former results. None of the opponents, except ABC in one of the problems, attained significantly better results than EMEL. Performance of VSA was lesser to according to all the tested algorithms along with EMEL. Contrary to VSA, GWO at least manages to have scores denoting no difference. However, remaining problems are resulted in favour of EMEL. Following the former constrained benchmark, PSO manages to maintain a tough competition to EMEL. While three of obtained results could not be present significant difference, the remaining one captured by EMEL.

4.6. Overview of Combined Statistical Results

The statistical results are presented separately in former subsections according to sources of the collected benchmarks. EMEL attained superiority over its rivals by producing significantly better results for overall benchmarks. Furthermore, combining the results together into one table would highlight the superiority of EMEL. Table 4.6 presents the combined statistical results of 120 benchmarks.

Table 4.6. Mann Whitney U test comparison results based on combined benchmarks

COMBINED RESULTS OF 120 BENCHMARKS						
			Opponent Algorithm			
			GWO	**ABC**	**PSO**	**VSA**
Reference Algorithm	**EMEL**	EMEL is better	67	67	46	65
		No Difference	26	35	49	42
		Opp. is better	27	18	25	13
	GWO	GWO is better		36	31	48
		No Difference		21	30	23
		Opp. is better		63	59	49
	ABC	ABC is better			35	47
		No Difference			34	29
		Opp. is better			51	44
	PSO	PSO is better				54
		No Difference				38
		Opp. is better				28

GWO displayed one of the highest number of functions that have significantly better results than EMEL among the tested algorithms. Although GWO outperforms EMEL in 23% of functions, EMEL outperforms GWO by achieving 56% significantly better results in functions, which is more than two folds of the success achieved by GWO. Also, competition of GWO results in the smallest number of functions whose optimization results are marked as *no difference* among all algorithms. This shows significant polarization in outcomes comparing to the results of algorithms. If there is no significant difference between the optimization performances of two algorithms for a function, favouring either of them will not be a big deal. In fact, considering both of them successful at that function would be a fair evaluation. If we re-distribute the functions whose results show no significant difference in favour of one of the algorithms, an increase in the performances will be observed. GWO achieves significantly better than or as good results as EMEL in 44% of functions. On the contrary, EMEL's performance increases to %78 for functions.

ABC, on the other hand, seems to be obtain less polarization due to the number of functions with no significant difference is thirty-five. However, the results still polarize in favour of EMEL. This comparison results in that EMEL being superior to ABC in 56% of the functions while ABC manages to outperform EMEL in 15% of the set. If we re-distribute the functions whose optimization results show no significant difference as we did for GWO discussions, EMEL provides significantly

better results than ABC in 85% of the functions. In return, ABC's performance will be 44% in same context. EMEL is an obvious choice in this comparison even though ABC present a competition.

PSO is one of the cornerstone algorithms in the field of optimization and it has been a challenging opponent for EMEL in this study. The results have the highest total amount of *no difference* in functions for PSO. However, EMEL still manages having almost twofold significantly better results than PSO. PSO's performance is slightly less than GWO's with the score of 21%. Considering ABC, GWO, and VSA, PSO is the only one causes EMEL's performance to place under 54% with score of 38%. However, re-distributing the *no difference* data will lead EMEL to produce just as good or significantly better results than PSO in 79% of the functions, while PSO obtains success on 62%. However, in spite of power of PSO, there is strong confidence to favour EMEL comparing to PSO.

The last algorithm that we investigated in this work is VSA which has a similar search strategy to EMEL's. Regardless of their strategic similarity, EMEL's performances is distinctively superior to VSA. VSA produces significantly better results than EMEL in 11% of functions. EMEL, on the other hand, is far superior to VSA by achieving significantly better results in 54% of functions. When we take into account insignificant differences in favour of algorithms, striking results will be observed. The competition almost disappears by EMEL's performance of 89%, which is higher score of this context among the algorithms. VSA, on the other hand, provides just as good or better optimization performance than EMEL in 46% of the functions. The superiority of EMEL over VSA is clear without a doubt.

4.7. Analysis of Convergence Curves

A convergence curve is a graphical representation of an optimization algorithm's progress across iterations. It commonly depicts the objective function value on the y-axis and the number of iterations on the x-axis. The convergence curve assists in measuring the algorithm's effectiveness and efficiency by demonstrating how quickly and effectively it approaches the optimal solution. Analysing the convergence curve provides insights regarding the algorithm's stability, speed, and dependability. For instance, a rapidly dropping curve indicates fast convergence, but a plateau may indicate a local minimum or the need for parameter tuning.

In order to provide an insight, the convergence curves of the algorithms are sampled in Figure 4.1, 4.2, and 4.3. Convergence curves of six benchmarks selected from shifted/rotated, hybrid and composition functions of CEC 2017 are presented for all algorithms. According to the figures, the proposed algorithm EMEL maintains a consistent, gradual decrease in objective function values across all benchmark functions. This steady descent implies that EMEL avoids large oscillations or erratic behaviour, which is frequently desired for stability in optimization tasks. Although EMEL does not achieve rapid convergence, its approach to convergence is consistent, with relatively smooth

curves and few sudden changes in function values. This steady approach could be useful in applications where gradual optimization is preferred over aggressive convergence.

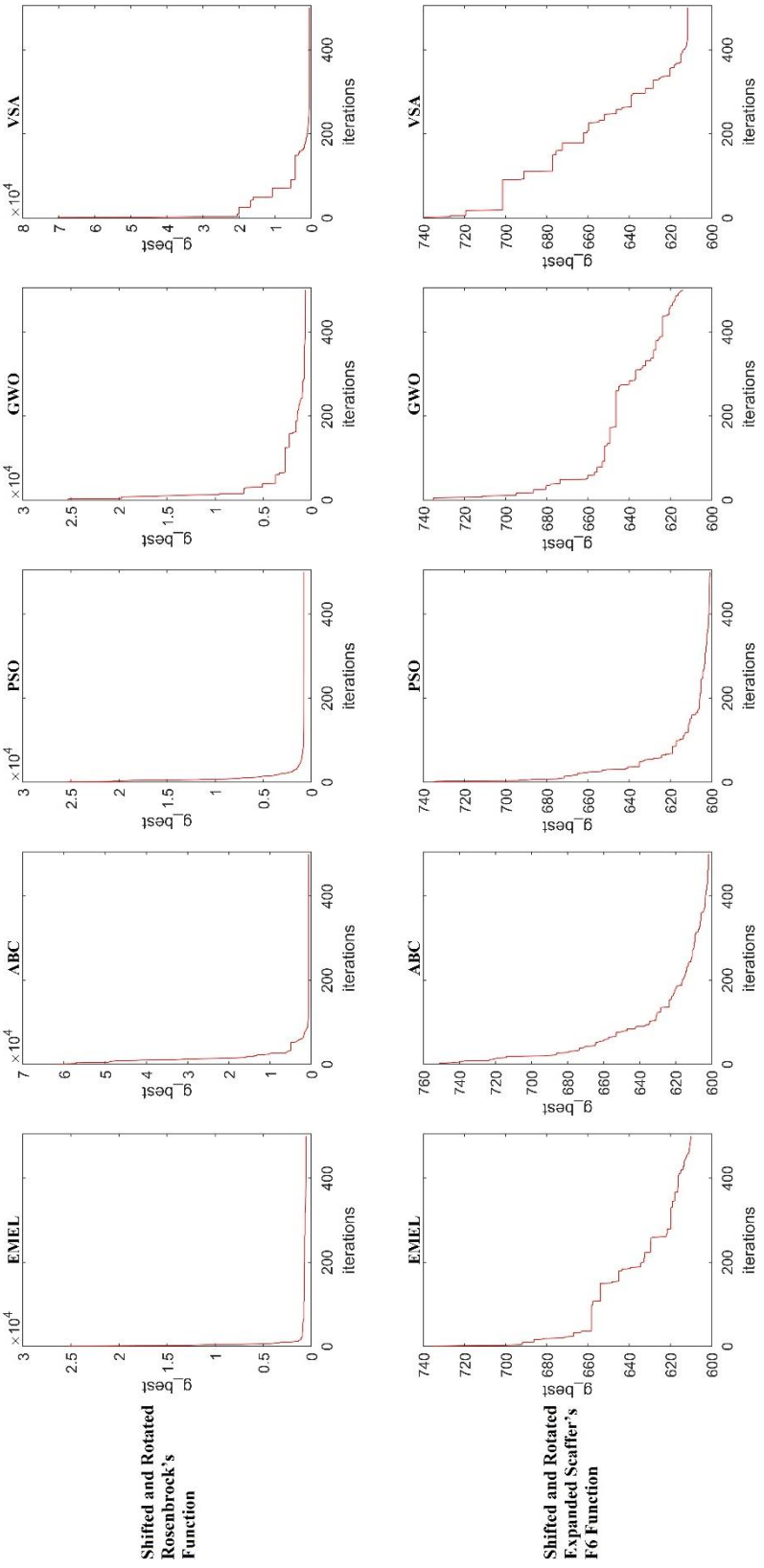

Figure 4.1. Convergence curves of the algorithms for selected benchmarks.

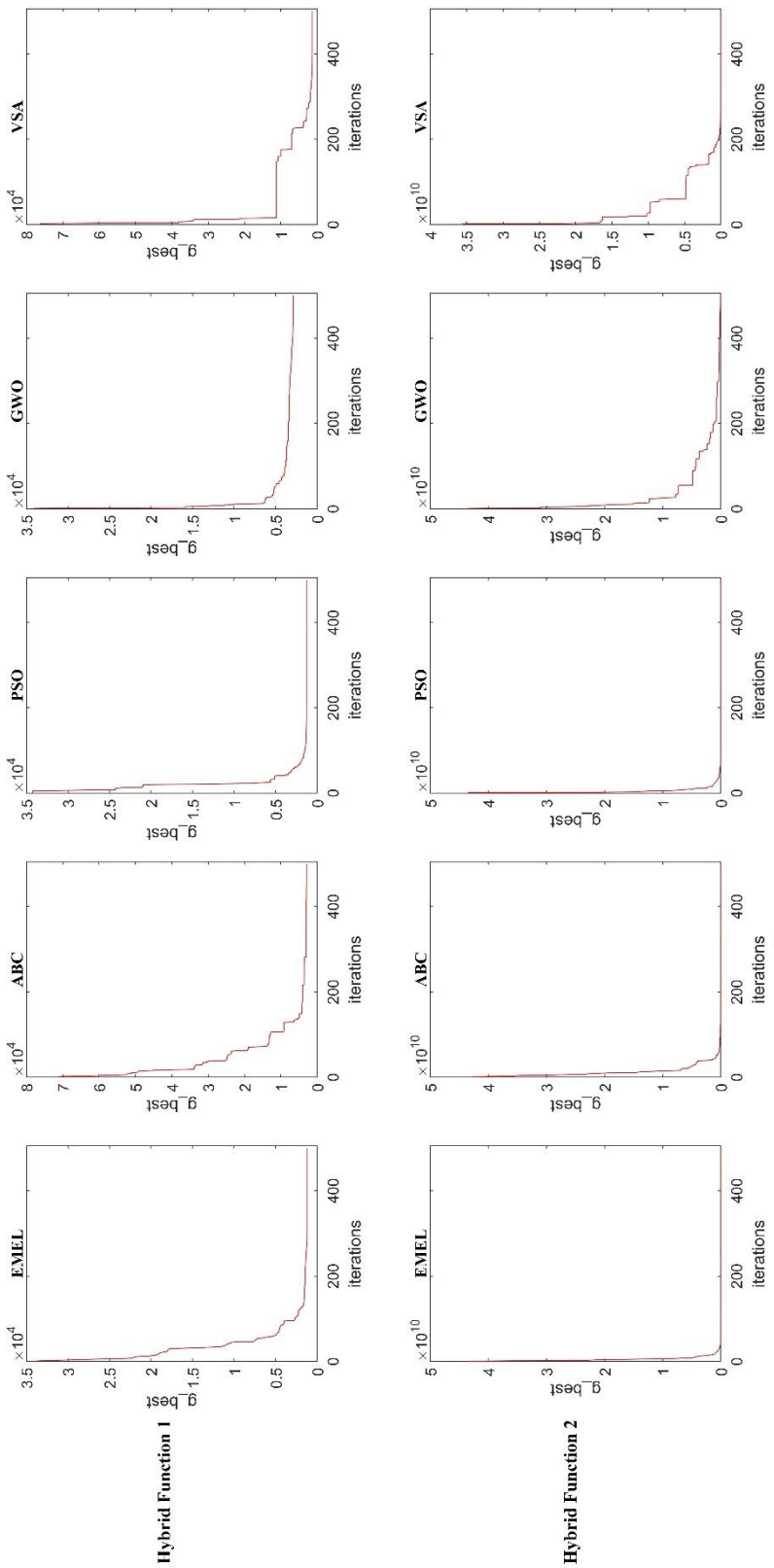

Figure 4.2. Convergence curves of the algorithms for selected benchmarks (continues)

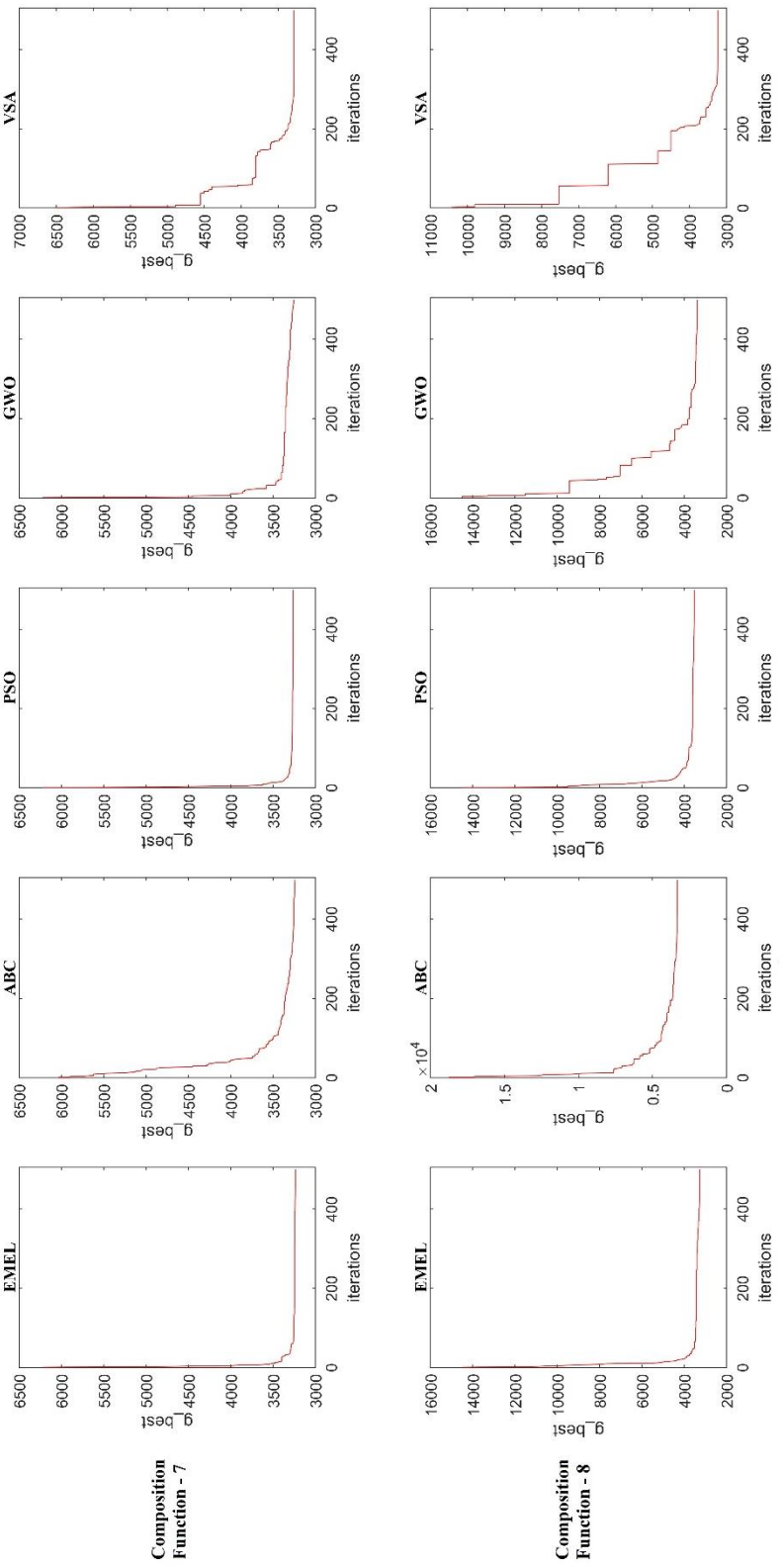

Figure 4.3. Convergence curves of the algorithms for selected benchmarks (continues)

4.8. Future Works of the Proposed Algorithm

The proposed algorithm EMEL presents a basic optimization approach by using fundamental and free of complex operations. Despite its simplicity, EMEL manages to outperform its opponents. However, it is found out that some contexts, such as constraint handling, could be able to challenge EMEL. Since optimization field is vast and rich in terms of variety of problems the emerging challenges are insignificant as a result of No Free Lunch theorem, which states that there is no algorithm could handle all problems with same performance. Nevertheless, literature involves some methods that aim performance improvement on optimization algorithms. For instance, history-based methods use a memory consisting of previous solutions either to prevent or contribute visiting them in next iterations of the search. Levy flights, on the other hand, apply random huge walks over search space in order to prevent stuck on local optimum solutions. Adaptive parameter control is other approach that dynamically adjusts the values of one or more design parameters for an effective balance between exploration and exploitation. Considering such methods existing in literature, EMEL has potential improvements and future researches would focus on that subject.

5. CONCLUSIONS

A novel population-based optimization algorithm called EMEL, which stands for *Exploration of Moving and Ever-shrinking Layers*, is proposed in this thesis work. In order to validate the effectiveness of the proposed algorithm a benchmarks set consist of 120 functions is used. The most commonly studied benchmarks from the literature are collected to form the benchmark set of this study. The formed benchmark set involves various and challenging characteristics such as shifted/rotated search spaces, hybrid and composition functions, constrained functions and also real world engineering design problems.

Optimization of constraint handling requires further calculations due to the equality and inequality constraints that are needed to be satisfied. EMEL and other algorithms are properly modified in order to import penalized fitness function. The experimental framework was constructed to provide fair chances to algorithms during optimization processes. For this purpose, values of appropriate design parameters of algorithms were tuned to obtain best optimization performances. The results were compared according to statistical significance.

In order to carry out a comparative study, well-known metaheuristics PSO, ABC, GWO and VSA are used in the experiments. EMEL’s overall performance is identified as outstanding comparing to its counterparts after the tests. We have carried out our analysis utilizing Mann Whitney U test to observe EMEL's behaviour. The statistical results are presented separately in former subsections according to sources of the collected benchmarks. EMEL attained superiority over its rivals by producing significantly better results for overall benchmarks. Furthermore, if the results are combined together, the obtained table would highlight the EMEL's performance. According to discussion of combined statistical results, it could be confidentially argued that EMEL would be a promising optimization algorithm.

In spite of the slight competition EMEL proved to be obviously superior to tested algorithms for benchmarks. The toughest challenge have come with PSO's *no difference* score, which places its percentage of being just as good or better than EMEL to top among all other competitors. However, this does not produce a superiority. Still, EMEL manages to demonstrably outperform PSO in benchmarks. Also, PSO was the second best choice for real-world problems. Once again, this was not a superiority against EMEL, yet a satisfactory competition with other rivals.

On the other hand, the proposed algorithm has some drawbacks. For example, the proposed algorithm cannot be applied to problems requiring binary solutions in its current state. Another limitation originates from the virtual search space's gradually shrinking structure. Once the virtual search space begins to shrink, even if it continues to move across the space, the overlapping area gradually decreases, reducing the possibility of generating solutions anywhere in the search space. As a drawback, this situation may cause the algorithm to converge around a local optimum and

become stuck there. Furthermore, this situation limits the algorithm's adaptability to dynamic problems.

Constraint handling abilities of EMEL are also required to be carefully investigated. Though EMEL manages being favourable among ABC, GWO and VSA, the reasons that allows PSO to show such a competition are needed to be disclosed. Furthermore, following a detailed research, proper methods enhancing the optimization skills and constraint handling abilities could be inherited from the literature to improve the performance of EMEL.

It can be concluded that EMEL is a simple but efficient optimization algorithm. Its performance is quite efficient comparing to some of the cornerstone optimization algorithms. Its simple search strategy is promising for further improvements. Future studies will focus on improvements that aim at eliminating the aforementioned drawbacks of the proposed algorithm.

REFERENCES

Abdollahzadeh, B., Khodadadi, N., Barshandeh, S., Trojovský, P., Gharehchopogh, F. S., El-kenawy, E.-S. M., Abualigah, L., & Mirjalili, S. (2024). Puma optimizer (PO): a novel metaheuristic optimization algorithm and its application in machine learning. *Cluster Computing*, *27*(4), 5235–5283. https://doi.org/10.1007/s10586-023-04221-5

Amiri, M. H., Mehrabi Hashjin, N., Montazeri, M., Mirjalili, S., & Khodadadi, N. (2024). Hippopotamus optimization algorithm: a novel nature-inspired optimization algorithm. *Scientific Reports*, *14*(1), 5032. https://doi.org/10.1038/s41598-024-54910-3

Arora, R. K. (2015). Optimization: Algorithms and Applications. In *Optimization: Algorithms and Applications*. Chapman and Hall/CRC. https://doi.org/10.1201/b18469

Awad, N. H., Ali, M. Z., Suganthan, P. N., Liang, J. J., & Qu, B. Y. (2016). Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real parameter optimization. *Technical Report, Nanyang Technological University & Jordan University of Science and Technology & Zhengzhou University*.

Benmamoun, Z., Khlie, K., Bektemyssova, G., Dehghani, M., & Gherabi, Y. (2024). Bobcat Optimization Algorithm: an effective bio-inspired metaheuristic algorithm for solving supply chain optimization problems. *Scientific Reports*, *14*(1), 20099. https://doi.org/10.1038/s41598-024-70497-1

Boussaïd, I., Lepagnot, J., & Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, *237*, 82–117. https://doi.org/10.1016/j.ins.2013.02.041

Cagnina, L. C., Esquivel, S. C., & Coello, C. A. C. (2008). Solving engineering optimization problems with the simple constrained particle swarm optimizer. *Informatica - Journal of Computing and Informatics*, *32*(3), 319–326.

Coello Coello, C. A. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, *191*(11–12), 1245–1287. https://doi.org/10.1016/S0045-7825(01)00323-1

Darling, D. (2004). *The Universal Book of Mathematics: From Abracadabra to Zeno's Paradoxes*. John Wiley & Sons.

Deb, K. (2012). *Optimization for Engineering Design: Algorithms and Examples* (2nd ed.). PHI Learning. https://books.google.com.tr/books?id=cN_kjtySMhIC

Doğan, B., & Ölmez, T. (2015). A new metaheuristic for numerical function optimization: Vortex search algorithm. *Information Sciences*, *293*, 125–145. https://doi.org/10.1016/j.ins.2014.08.053

Dorigo, M., & Di Caro, G. (1999). Ant colony optimization: A new meta-heuristic. *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, 1470–1477.

https://doi.org/10.1109/CEC.1999.782657

Duan, Y., Zhao, Y., & Hu, J. (2023). An initialization-free distributed algorithm for dynamic economic dispatch problems in microgrid: Modeling, optimization and analysis. *Sustainable Energy, Grids and Networks*, *34*, 101004. https://doi.org/10.1016/j.segan.2023.101004

Fang, N., & Cao, Q. (2024). Leaf in Wind Optimization: A New Metaheuristic Algorithm for Solving Optimization Problems. *IEEE Access*, *12*, 56291–56308. https://doi.org/10.1109/ACCESS.2024.3390670

Fu, Y., Liu, D., Chen, J., & He, L. (2024). Secretary bird optimization algorithm: a new metaheuristic for solving global optimization problems. *Artificial Intelligence Review*, *57*(5), 123. https://doi.org/10.1007/s10462-024-10729-y

Gandomi, A. H., Yang, X.-S., & Alavi, A. H. (2013). Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. *Engineering with Computers*, *29*(1), 17–35. https://doi.org/10.1007/s00366-011-0241-y

Gençal, M. C., & Oral, M. (2022). Bipolar Mating Tendency: Harmony Between the Best and the Worst Individuals. *Arabian Journal for Science and Engineering*, *47*(2), 1849–1871. https://doi.org/10.1007/s13369-021-06105-5

Ghaedi, A., Bardsiri, A. K., & Shahbazzadeh, M. J. (2023). Cat hunting optimization algorithm: a novel optimization algorithm. *Evolutionary Intelligence*, *16*(2), 417–438. https://doi.org/10.1007/s12065-021-00668-w

Ghiaskar, A., Amiri, A., & Mirjalili, S. (2024). Polar fox optimization algorithm: a novel meta-heuristic algorithm. *Neural Computing and Applications*, *36*(33), 20983–21022. https://doi.org/10.1007/s00521-024-10346-4

Gold, S., & Krishnamurty, S. (1997, September 14). Trade-offs in robust enginering design. *Volume 2: 23rd Design Automation Conference*. https://doi.org/10.1115/DETC97/DAC-3757

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

Heiberger, R. M., & Holland, B. (2015). Nonparametrics. In *Statistical Analysis and Data Display* (pp. 577–592). https://doi.org/10.1007/978-1-4939-2122-5_16

Heumann, C., Schomaker, M., & Shalabh. (2016). Hypothesis testing. In *Introduction to Statistics and Data Analysis* (pp. 209–247). Springer International Publishing. https://doi.org/10.1007/978-3-319-46162-5_10

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems*. The MIT Press. https://doi.org/10.7551/mitpress/1090.001.0001

Hubálovská, M., Hubálovský, Š., & Trojovský, P. (2024). Botox Optimization Algorithm: A New Human-Based Metaheuristic Algorithm for Solving Optimization Problems. *Biomimetics*, *9*(3), 137. https://doi.org/10.3390/biomimetics9030137

Jain, M., Maurya, S., Rani, A., & Singh, V. (2018). Owl search algorithm: A novel nature-inspired heuristic paradigm for global optimization. *Journal of Intelligent & Fuzzy Systems*, *34*(3), 1573–1582. https://doi.org/10.3233/JIFS-169452

Jamil, M., & Yang, X. S. (2013). A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, *4*(2), 150. https://doi.org/10.1504/IJMMNO.2013.055204

Jia, H., Wen, Q., Wang, Y., & Mirjalili, S. (2024). Catch fish optimization algorithm: a new human behavior algorithm for solving clustering problems. *Cluster Computing*, *27*(9), 13295–13332. https://doi.org/10.1007/s10586-024-04618-w

Jordehi, A. R. (2019). Binary particle swarm optimisation with quadratic transfer function: A new binary optimisation algorithm for optimal scheduling of appliances in smart homes. *Applied Soft Computing*, *78*, 465–480. https://doi.org/10.1016/j.asoc.2019.03.002

Karaboga, D., & Akay, B. (2009). A comparative study of Artificial Bee Colony algorithm. *Applied Mathematics and Computation*, *214*(1), 108–132. https://doi.org/10.1016/j.amc.2009.03.090

Karaboga, D., & Akay, B. (2011). A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems. *Applied Soft Computing*, *11*(3), 3021–3031. https://doi.org/10.1016/j.asoc.2010.12.001

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, *39*(3), 459–471. https://doi.org/10.1007/s10898-007-9149-x

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, *4*, 1942–1948. https://doi.org/10.1109/ICNN.1995.488968

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

Liu, W.-L., Zhong, J., Liang, P., Guo, J., Zhao, H., & Zhang, J. (2024). Towards explainable traffic signal control for urban networks through genetic programming. *Swarm and Evolutionary Computation*, *88*, 101588. https://doi.org/10.1016/j.swevo.2024.101588

Mattos, D. I., Bosch, J., & Olsson, H. H. (2021). Statistical models for the analysis of optimization algorithms with benchmark functions. *IEEE Transactions on Evolutionary Computation*, *25*(6), 1163–1177. https://doi.org/10.1109/TEVC.2021.3081167

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, *95*, 51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, *69*, 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007

Moshkelgosha, V., Behzadi-Khormouji, H., & Yazdian-Dehkordi, M. (2017). Coarse-to-fine parameter tuning for content-based object categorization. *2017 3rd International Conference*

on Pattern Recognition and Image Analysis (IPRIA), 160–165. https://doi.org/10.1109/PRIA.2017.7983038

Oral, M., Baazati, S., Aydınlı, S., & Oral, E. (2018). Construction Site Layout Planning: Application of Multi-Objective Particle Swarm Optimization*. *Teknik Dergi*, *29*(6), 8691–8713. https://doi.org/10.18400/tekderg.389638

Ragsdell, K. M., & Phillips, D. T. (1976). Optimal design of a class of welded structures using geometric programming. *Journal of Engineering for Industry*, *98*(3), 1021–1025. https://doi.org/10.1115/1.3438995

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, 69–73. https://doi.org/10.1109/ICEC.1998.699146

Storn, R. (1996). On the usage of differential evolution for function optimization. *Proceedings of North American Fuzzy Information Processing*, 519–523. https://doi.org/10.1109/NAFIPS.1996.534789

Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y. P., Auger, A., & Tiwari, S. (2005). Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. *Technical Report, Nanyang Technological University, Singapore, May 2005 AND KanGAL Report 2005005, IIT Kanpur, India*.

Wang, C., Wang, Y., Wang, K., Dong, Y., & Yang, Y. (2017). An Improved Hybrid Algorithm Based on Biogeography/Complex and Metropolis for Many-Objective Optimization. *Mathematical Problems in Engineering*, *2017*, 1–14. https://doi.org/10.1155/2017/2462891

Wang, R., & Zhang, R. (2023). Techno-economic analysis and optimization of hybrid energy systems based on hydrogen storage for sustainable energy utilization by a biological-inspired optimization algorithm. *Journal of Energy Storage*, *66*, 107469. https://doi.org/10.1016/j.est.2023.107469

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82. https://doi.org/10.1109/4235.585893

Xie, Y., Wang, X.-Y., Shen, Z.-J., Sheng, Y.-H., & Wu, G.-X. (2023). A Two-Stage Estimation of Distribution Algorithm With Heuristics for Energy-Aware Cloud Workflow Scheduling. *IEEE Transactions on Services Computing*, *16*(6), 4183–4197. https://doi.org/10.1109/TSC.2023.3311785

Yin, L., Zhuang, M., Jia, J., & Wang, H. (2020). Energy Saving in Flow-Shop Scheduling Management: An Improved Multiobjective Model Based on Grey Wolf Optimization Algorithm. *Mathematical Problems in Engineering*, *2020*, 1–14. https://doi.org/10.1155/2020/9462048

Zareian, L., Rahebi, J., & Shayegan, M. J. (2024). Bitterling fish optimization (BFO) algorithm.

Multimedia Tools and Applications, *83*(31), 75893–75926. https://doi.org/10.1007/s11042-024-18579-0

Zhang, H., San, H., Sun, H., Ding, L., & Wu, X. (2024). A novel optimization method: wave search algorithm. *The Journal of Supercomputing*, *80*(12), 16824–16859. https://doi.org/10.1007/s11227-024-06078-w

Zhang, W., Zhao, J., Liu, H., & Tu, L. (2024). Cleaner fish optimization algorithm: a new bio-inspired meta-heuristic optimization algorithm. *The Journal of Supercomputing*, *80*(12), 17338–17376. https://doi.org/10.1007/s11227-024-06105-w

CURRICULUM VITAE

Bilal İŞÇİMEN received a Bachelor's degree in 2013 from Mustafa Kemal University, Department of Computer Engineering and he received a M.Sc. degree in 2016 from Mustafa Kemal University, Department of Informatics. He worked as a research assistant in the Department of Computer Engineering in Mustafa Kemal University between 2013 and 2016. Since 2017, he has been working as a lecturer at Kırıkhan Vocational School of Hatay Mustafa Kemal University.

APPENDICES

Appendix A: List of the sixty standard global optimization benchmarks

Table A1. The values of adjusted design parameters of the algorithms for standard benchmarks

		Design parameters of the algorithms															
		EMEL			ABC			PSO				GWO		VSA	
#	Function Name	N	P	σ	L	N	T	w	C_1	C_2	$\vec{a}$	$\vec{C}$	x
1	Ackley	134	3	255	101	15	1,6	0,6	2,55	1,18	1,2	5	0,1277
2	Adjiman	10	1,3	175	38	1	3,7	0	1	1,4	1,4	1,6	0,0531
3	Beale	10	1,1	200	147	3	2,3	0	1	2,2	1,5	1,7	0,0100
4	Bird	24	2,4	200	114	1	5,5	0,4	1,2	2,2	1,5	1,3	0,0100
5	Bohachevsky 1	10	1,1	50	150	1	3,4	0	1	1,4	1	1	0,0100
6	Bohachevsky 2	10	1,1	50	21	1	4	0	1	1,4	1,4	2,6	0,0100
7	Bohachevsky 3	10	1,1	75	114	1	1,7	0	1	1,8	1	5	0,0100
8	Booth	10	1,1	75	117	1	3,2	0,1	1,24	1,56	1	1,5	0,0100
9	Branin	10	1,1	50	22	1	4,2	0	1	1,4	1,2	1,1	0,0100
10	Colville	22	1,2	248,4	124	2	2,3	0,6	1,01	1,9	2,8	1,7	0,8277
11	Dixon-Price	73	2,3	289	86	1	1,2	0,6	2,34	1,24	1,1	4,7	0,0394
12	Easom	10	1,1	50	122	1	3,7	0	1	1,4	1,1	1,3	0,0100
13	Exponential	150	2,9	254,4	32	15	1,5	0,6	2,26	1,29	1	3,8	0,0188
14	Fletcher-Powell 10	15	1,1	225,6	138	1	1,5	0,5	3,13	1	3	2,9	0,0178
15	Fletcher-Powell 2	10	1,1	75	122	1	3,7	0	1	1,4	1	1,6	0,0100
16	Fletcher-Powell 5	26	1,2	210	145	3	2,4	0,5	3,08	1	1,8	1,9	0,2044
17	Foxholes	10	1,3	175	94	1	4,6	0,3	1	2,12	3,7	1,6	0,0100
18	Goldstein-Price	52	2,4	175	150	1	2,6	0,1	1,98	1,07	1,9	4,1	0,0414
19	Griewank	136	3	213	30	3	2,1	0,5	2,11	1,69	1	4,2	0,0198
20	Hartman 6	38	1,3	150	68	2	3,7	0,5	1	1	1	1,8	0,0296

Table A2. The values of adjusted design parameters of the algorithms for standard benchmarks

		Design parameters of the algorithms											
		EMEL			**ABC**			**PSO**			**GWO**		**VSA**
#	**Function Name**	N	P	σ	L	N	T	w	C_1	C_2	$\vec{a}$	$\vec{C}$	x
21	Hartmann 3	10	1,3	150	78	2	10	0	1	1,8	1	1,7	0,0198
22	HolderTable	10	1,1	125	82	1	2,3	0	1	1,4	1,1	1,4	0,0296
23	Keane	10	1,3	175	42	1	6	0	1,04	1,35	3	1,4	0,0100
24	Kowalik	12	2,8	283,4	80	2	1,1	0,6	1,83	1,2	1,1	1,8	0,7745
25	Langerman 10	108	1,1	125	148	2	1	0,3	2,96	1,09	1,4	1,8	0,4984
26	Langerman 5	23	1,1	243,9	135	3	2,3	0,5	1,8	1,4	1	1,6	0,0100
27	Langermann 2	10	1,5	125	24	1	3,5	0	1	1,8	1	1,5	0,1080
28	Levy	138	2,7	255	33	1	2,5	0,6	2,67	1,03	2,8	1,6	0,9880
29	Matyas	14	1,1	173,9	101	1	1	0,1	1,01	1,72	1,7	5	0,0112
30	Michalewicz 10	104	1,8	243,9	65	2	2,5	0,4	3,1	1,24	1,2	1,7	0,0346
31	Michalewicz 2	10	1,1	100	104	1	9,9	0	1,04	1,8	1,4	1,3	0,0100
32	Michalewicz 5	24	1,7	200	47	3	4,5	0,4	2,7	1,01	1,4	1,2	0,5003
33	Penalized	146	2,2	253,9	49	1	3	0,7	2,36	1,06	3,1	1,6	0,9896
34	Penalized 2	138	2,7	255	69	13	1,6	0,6	2,33	1,19	3	1,8	0,0165
35	Perm	10	1,2	262,6	140	2	1	0,6	1,08	2,76	3,1	3,2	0,3040
36	Powell	74	2,3	286	147	11	1	0,6	2,64	1,12	1,2	4,9	0,0786
37	Powell Sum	130	3	294,5	138	2	2,4	0,2	1,12	2,46	1,1	5	0,0276
38	Power sum	24	1,3	287,9	129	2	1,3	0,5	1,26	1,57	1,1	1	0,2060
39	Qing	148	2,9	255	112	1	2,1	0,6	2,09	1,41	1	1,3	0,0167
40	Quartic	72	1,7	296,1	147	9	1,5	0,6	2,04	1,37	1,1	4,3	0,0664

Table A3. The values of adjusted design parameters of the algorithms for standard benchmarks

		Design parameters of the algorithms											
		EMEL			**ABC**			**PSO**			**GWO**		**VSA**
#	**Function Name**	N	P	σ	L	N	T	w	C_1	C_2	$\vec{a}$	$\vec{C}$	x
41	Rastrigin	136	3	298,9	40	1	2,3	0,6	2,69	1,09	1	4,2	0,9900
42	Rosenbrock	11	1,9	300	114	4	1,1	0,6	2,28	1,28	2,8	1,3	0,2640
43	Salomon	52	2,2	276,9	143	14	1,4	0,6	2,45	1,16	1,3	4	0,1078
44	Schaffer	10	1,7	200	117	1	2,8	0,3	1,87	1,65	1,4	4,2	0,4216
45	Schwefel	136	2,4	255,9	65	2	2,2	0,5	3	1,06	2,5	2,1	0,9900
46	Schwefel 1.2	87	1,3	291,1	139	12	1	0,5	2,43	1,24	1	4,7	0,0897
47	Schwefel 2.21	102	1,4	280	150	14	1	0,2	1,16	2,54	1,1	5	0,0157
48	Schwefel 2.22	147	3	252	63	15	1,6	0,6	2,7	1,02	1,1	5	0,0210
49	Shekel 10	10	1,1	100	58	2	2,3	0,5	2,2	1	1,1	1,6	0,0100
50	Shekel 5	10	1,1	170	150	2	2,4	0,5	1,73	3,4	1,1	2,3	0,0100
51	Shekel 7	10	1,1	175	82	2	3,6	0,3	1,48	1,64	1,2	1,3	0,0100
52	Six-Hump Camel	10	1,1	50	16	1	4	0	1	1,4	1,1	1,3	0,0100
53	Sphere	150	2,9	254,4	63	15	1,5	0,6	2,26	1,29	1,1	5	0,0188
54	Step	24	2,8	275	80	6	2,9	0,5	1,9	2,02	1,4	3	0,0512
55	Stepint	10	1,1	50	10	1	1	0	1	1,8	1,1	2,2	0,1080
56	Styblinski-Tang	60	2,9	272,8	42	2	2,4	0,6	2,78	1,07	2,9	1,9	0,6941
57	Sum Squares	149	2,6	258,1	47	15	1,4	0,6	2,28	1,37	1,1	5	0,0568
58	Trid 10	12	1,2	255	121	1	1,1	0,6	1,34	1,73	3	1,4	0,0473
59	Trid 6	10	1,1	205	121	3	1	0,4	1,45	2,1	1,1	1,6	0,0110
60	Zakharov	41	1,4	299	149	11	1	0,1	2,68	1,05	1,1	4,9	0,0219

Table B. The values of adjusted design parameters of the algorithms for CEC 2005 benchmarks

		Design parameters of the algorithms											
		EMEL			**ABC**			**PSO**			**GWO**		**VSA**
#	**Function Name**	N	P	σ	L	N	T	w	C_1	C_2	$\vec{a}$	$\vec{C}$	x
1	Shifted Sphere Function	148	2,7	254	59	15	1,6	0,5	3,32	1	2,8	1,7	0,01
2	Shifted Schwefel’ s Problem 1.2	48	1,3	300	142	8	1	0,6	2,25	1,03	3,8	1,7	0,022
3	Shifted Rotated High Conditioned Elliptic Function	13	1,6	286,3	52	1	1	0,6	2,55	1,14	2,6	1,8	0,029
4	Shifted Schwefel’ s Problem 1.2 with Noise in Fitness	10	1,9	300	144	14	1	0,5	2,67	1,01	3,8	1,6	0,069
5	Schwefel’ s Problem 2.6 with Global Optimum on Bounds	40	1,1	297,3	142	14	1,3	0,6	2,87	1,02	3,2	1,8	0,304
6	Shifted Rosenbrock’ s Function	139	2,7	285	113	1	2,4	0,5	3,23	1,01	4,2	1,4	0,079
7	Shifted Rotated Griewank’ s Function without Bounds	146	2,9	250	52	7	3,7	0	4,71	1,77	3,9	2,5	0,079
8	Shifted Rotated Ackley’ s Function with Global Optimum on Bounds	144	2,1	271,1	81	13	8,2	0,3	1,02	1,52	3,8	4	0,098
9	Shifted Rastrigin’ s Function	102	2,9	277,5	38	1	2,3	0,5	2,74	1,01	4	2	0,402
10	Shifted Rotated Rastrigin’ s Function	10	1,6	282,6	139	13	2,3	0,7	2,12	1,01	3,8	1,8	0,136
11	Shifted Rotated Weierstrass Function	94	1,1	299,9	140	2	1,2	0,7	1,62	1,82	1,3	2,1	0,034
12	Schwefel’ s Problem 2.13	104	1,4	290	69	1	3	0,6	2,75	1,01	2,8	1,4	0,038
13	Expanded Extended Griewank’ s plus Rosenbrock’ s Function	146	2,2	299,5	143	1	2	0	1,01	2,68	2,2	1,4	0,237
14	Shifted Rotated Expanded Scaffer’ s	22	1,7	275,1	131	1	3,3	0,2	1,89	1,2	2,2	2,5	0,124

Table C1. The values of adjusted design parameters of the algorithms for CEC 2017 benchmarks

		Design parameters of the algorithms											
		EMEL			**ABC**			**PSO**			**GWO**		**VSA**
#	**Function Name**	N	P	σ	L	N	T	w	C_1	C_2	$\vec{a}$	$\vec{C}$	x
1	Shifted and Rotated Bent Cigar Function	114	2,9	251,4	114	2	2,1	0,5	3,04	1,02	3	1,8	0,02
2	Shifted and Rotated Zakharov Function	24	1,7	297,6	11	2	1,1	0,6	1,19	1,37	2	3,6	0,037
3	Shifted and Rotated Rosenbrock’ s Function	123	3	245,3	115	2	1,4	0,7	2,24	1,05	4,7	1,3	0,018
4	Shifted and Rotated Rastrigin’ s Function	13	2,8	292,5	132	1	3,2	0,6	2,54	1,01	2,8	1,9	0,206
5	Shifted and Rotated Expanded Scaffer’ s F6 Function	120	3	280	82	4	2,3	0,6	2,69	1,01	2,8	1,6	0,049
6	Shifted and Rotated Lunacek Bi_Rastrigin Function	150	1,2	245	66	1	2,8	0,7	2,36	1,11	2,5	1,9	0,041
7	Shifted and Rotated Non-Continuous Rastrigin’ s Function	44	1,2	259,9	131	1	2,8	0,6	2,5	1,01	3	1,7	0,186
8	Shifted and Rotated Levy Function	44	1,1	247	120	1	3,7	0,6	2,74	1	3,1	1,3	0,079
9	Shifted and Rotated Schwefel’ s Function	24	2,6	187	108	1	2,8	0,6	1,85	1,16	1,5	2	0,199
10	Hybrid Function 1 (N=3)	44	1,7	286,1	142	14	1,1	0,7	2,27	1,05	3,8	1,7	0,03
11	Hybrid Function 2 (N=3)	140	2,4	286	131	3	1,1	0,6	2,62	1,08	2,3	1,4	0,305
12	Hybrid Function 3 (N=3)	81	2,6	227,9	128	5	1	0,5	3,08	1,06	1,3	1	0,461
13	Hybrid Function 4 (N=4)	40	1,1	261	112	10	1	0,5	2,06	1,17	2,6	2,8	0,132
14	Hybrid Function 5 (N=4)	116	1,5	299,5	138	7	1	0,5	3,1	1	1	1,2	0,502
15	Hybrid Function 6 (N=4)	30	1,4	297	96	4	1,3	0,6	2,66	1,07	3	1,6	0,081

Table C2. The values of adjusted design parameters of the algorithms for CEC 2017 benchmarks (continues)

		Design parameters of the algorithms											
		EMEL			**ABC**			**PSO**			**GWO**		**VSA**
#	**Function Name**	N	P	σ	L	N	T	w	C_1	C_2	$\vec{a}$	$\vec{C}$	x
16	Hybrid Function 6 (N=5)	13	1,4	270	129	5	1,9	0,6	2,75	1,15	3,2	1,9	0,186
17	Hybrid Function 6 (N=5)	34	1,1	256,2	132	3	1,3	0,5	2,08	1,1	1,4	2,6	0,586
18	Hybrid Function 6 (N=5)	68	1,3	249,8	138	7	1,7	0,7	2,66	1,03	2	2,5	0,209
19	Hybrid Function 6 (N=6)	79	1,2	289,1	148	5	2	0,6	2,61	1,21	1,8	3	0,18
20	Composition Function 1 (N=3)	12	2,6	252,7	68	7	1	0,6	2,65	1,03	3,4	1,5	0,419
21	Composition Function 2 (N=3)	48	1,5	297,5	78	2	1,4	0,6	2,68	1,02	1,2	4	0,039
22	Composition Function 3 (N=4)	68	1,1	282,4	116	1	2,6	0,5	3,02	1,04	4,2	1,7	0,186
23	Composition Function 4 (N=4)	24	1,1	284,9	147	2	3,2	0	3,1	1,14	3,5	1,7	0,503
24	Composition Function 5 (N=5)	24	2,4	275,3	136	15	1,5	0,7	2,44	1,03	4,5	1,7	0,059
25	Composition Function 6 (N=5)	22	1,5	295	102	2	1,1	0,5	2,87	1	3,5	1,5	0,016
26	Composition Function 7 (N=6)	10	2,8	297	126	6	3	0,4	3,33	1,15	3,4	2,2	0,1
27	Composition Function 8 (N=6)	100	1,6	293,1	75	9	1,1	0,6	2,95	1,03	4,3	1,3	0,015
28	Composition Function 9 (N=3)	23	1,7	298	136	3	1,7	0,6	2,77	1,01	3,4	2,6	0,265
29	Composition Function 10 (N=3)	92	1,6	288,3	132	2	1,3	0,5	3,13	1	2,6	1,8	0,205

Appendix D: List of the thirteen constrained benchmarks

Table D. The values of adjusted design parameters of the algorithms for constrained benchmarks

		Design parameters of the algorithms																	
		EMEL			ABC			PSO			GWO			VSA	
#	Function Name	N	P	σ	L	N	T	w	C_1	C_2	$\vec{a}$	$\vec{C}$	x
1	G_01	26	2,3	233,6	59	6	2,8	0,1	4,58	1,02	1	1,4	0,798
2	G_02	137	1,2	294	145	6	2,3	0,7	2,58	1,05	2,6	1,3	0,328
3	G_03	34	3	63,12	11	1	9,5	0	1,07	1,17	3,8	1,6	0,014
4	G_04	42	1,1	240	147	3	2,8	4,5	1	1	1,2	1,5	0,041
5	G_05	10	2,6	217,4	94	2	1	0,6	2,48	1,23	1,2	1,4	0,076
6	G_06	11	1,1	219	64	1	1	0,5	1,34	1,11	1	1,5	0,011
7	G_07	13	1,5	275	147	4	1,8	0,7	2,26	1	3,6	2	0,149
8	G_08	10	1,1	75	148	1	4,3	0	1	1,8	1,1	1	0,01
9	G_09	56	1,1	290	137	4	1,9	0,6	1,93	1,75	3,8	2,2	0,167
10	G_10	19	1,2	260	104	2	1,1	0,7	1,65	1,67	1	1	0,284
11	G_11	110	2	52,65	24	1	7,3	0,2	1,17	1,21	2,3	2	0,461
12	G_12	10	1,3	50	10	2	1,9	0,1	1	1,49	1	1,6	0,484
13	G_13	24	1,4	215	147	1	1,2	0,7	1,47	1,24	1,5	1,8	0,104

Appendix E: List of the four constrained engineering design problems

Table E. The values of adjusted design parameters of the algorithms for constrained engineering design problems

		Design parameters of the algorithms														
		EMEL			**ABC**			**PSO**			**GWO**		**VSA**
#	**Function Name**	N	P	σ	L	N	T	w	C_1	C_2	$\vec{a}$	$\vec{C}$	x
1	Welded Beam	23	1,2	209,4	131	2	6	0,5	2,46	1,48	1,1	1,5	0,206
2	Speed Reducer	24	2,4	175	38	4	2,8	0	1,03	2,28	1,1	1,6	0,421
3	I-Beam	10	1,1	125	80	2	3,7	0,5	1	1	1,1	1,1	0,108
4	Piston Lever	38	2,5	196,9	147	2	3	0,9	1,72	2,05	2,7	3,2	0,403

Appendix F: Welded Beam Design Problem

Welded beam design problem (Cagnina et al., 2008; Ragsdell and Phillips, 1976) aims to minimize the cost of welded beam in the manufacturing process. This objective is subject to shear stress, bending stress in the beam, buckling load on the bar, and maximum end deflection of the beam. The design variables are the width and length of the welded area, the depth and the thickness of the main beam. The objective function and constraints of welded beam design are given as follows:

Minimize:

$f(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2)$

Subject to:

$g_1(\vec{x}) = \tau(\vec{x}) - 13600 \leq 0$

$g_2(\vec{x}) = \sigma(\vec{x}) - 30000 \leq 0$

$g_3(\vec{x}) = x_1 - x_4 \leq 0$

$g_4(\vec{x}) = 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5 \leq 0$

$g_5(\vec{x}) = 0.125 - x_1 \leq 0$

$g_6(\vec{x}) = \delta(\vec{x}) - 0.25 \leq 0$

$g_7(\vec{x}) = 6000 - P_c(\vec{x}) \leq 0$

$$\tau(\vec{x}) = \sqrt{(\tau')^2 + (2\tau'\tau'')\frac{x_2}{2R} + (\tau'')^2}, \qquad \tau' = \frac{6000}{\sqrt{2}x_1x_2}, \qquad \tau'' = \frac{MR}{J}$$

$$M = 6000\left(14 + \frac{x_2}{2}\right), \qquad R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}, \qquad J = 2\left(x_1x_2\sqrt{2}\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right)$$

$$\sigma(\vec{x}) = \frac{504 \cdot 10^3}{x_4x_3^2}, \qquad \delta(\vec{x}) = \frac{65856 \cdot 10^3}{(30 \cdot 10^6)x_4x_3^3}$$

$$P_c(\vec{x}) = \frac{4.013(30 \cdot 10^6)\sqrt{\frac{x_3^2x_4^6}{36}}}{196}\left(1 - \frac{x_3\sqrt{\frac{30 \cdot 10^6}{4(12 \cdot 10^6)}}}{28}\right)$$

with:

$0.1 \leq x_1, x_4 \leq 2$ and $0.1 \leq x_2, x_3 \leq 10$

Appendix G: Speed Reducer Design Problem

The objective of speed reducer design (Cagnina et al., 2008; Golinski, 1970) is minimizing the total weight of the speed reducer. This problem considers face width, module of teeth, number of teeth on pinion, length of the first shaft between bearings, length of the second shaft between bearings, diameter of the first shaft and diameter of the second shaft. The weight of the speed reducer is minimized subject to constraints on bending stress of the gear teeth, surface stress, transverse deflection of the shafts and stress in the shafts. The objective function and constraints of speed reducer design are given as follows:

Minimize:

$$f(\vec{x}) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$$

Subject to:

$$g_1(\vec{x}) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0$$

$$g_2(\vec{x}) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \leq 0$$

$$g_3(\vec{x}) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \leq 0$$

$$g_4(\vec{x}) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \leq 0$$

$$g_5(\vec{x}) = \frac{1.0}{110x_6^3}\sqrt{\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9 \cdot 10^6} - 1 \leq 0$$

$$g_6(\vec{x}) = \frac{1.0}{85x_7^3}\sqrt{\left(\frac{745x_5}{x_2x_3}\right)^2 + 157.5 \cdot 10^6} - 1 \leq 0$$

$$g_7(\vec{x}) = \frac{x_2x_3}{40} - 1 \leq 0$$

$$g_8(\vec{x}) = \frac{5x_2}{x_1} - 1 \leq 0$$

$$g_9(\vec{x}) = \frac{x_1}{12x_2} - 1 \leq 0$$

$$g_{10}(\vec{x}) = \frac{1.5x_6 + 1.9}{x_4} - 1 \leq 0$$

$$g_{11}(\vec{x}) = \frac{1.1x_7 + 1.9}{x_5} - 1 \leq 0$$

with:

$2.6 \leq x_1 \leq 3.6, 0.7 \leq x_2 \leq 0.8, 17 \leq x_3 \leq 28, 7.3 \leq x_4 \leq 8.3, 7.8 \leq x_5 \leq 8.3, 2.9 \leq x_6 \leq 3.9$ and $5.0 \leq x_7 \leq 5.5$

Appendix H: Piston Lever Design Problem

The aim of piston lever design problem (Gandomi et al., 2013) is to locate the piston components by minimizing the oil volume when the lever of the piston is lifted up from 0° to 45°. The problem involves four design variables which are H, B, D and C. This problem can be formulated as follows:

$\vec{X} = [x_1, x_2, x_3, x_4] = [H, B, D, C]$

Minimize:

$f(\vec{X}) = \frac{1}{4}\pi D^2(L_2 - L_1)$

Subject to:

$g_1 = QL\cos\theta - RF \leq 0$

$g_2 = Q(L - C) - M_{max} \leq 0$

$g_3 = 1.2(L_2 - L_1) - L_1 \leq 0$

$g_4 = D/2 - B \leq 0$

where:

$R = \frac{|-C(C\sin\theta + H) + H(B - C\cos\theta)|}{\sqrt{(C-B)^2 + H^2}}, \qquad F = \pi PD^2/4$

$L_1 = \sqrt{(C-B)^2 + H^2}, \qquad L_2 = \sqrt{(C\sin\theta + H)^2 + (B - C\cos\theta)^2}$

where payload $Q = 10000\ lbs$, $L = 240\ in$, $\theta = 45°$, the maximum bending moment of the lever is $M_{max} = 1.8 \times 10^6\ lbs \cdot in$, and the oil pressure is given as $P = 1500\ psi$

with:

$0.05 \leq H, B, D \leq 500$ and $0.05 \leq C \leq 120$

Appendix I: I-Beam Design Problem

The I-beam design problem (Gandomi et al., 2013; Gold and Krishnamurty, 1997) aims to design an I-shaped beam having minimum vertical deflection. In this case, there is four variables representing length, height and two thicknesses. The problem is minimized subject to cross-section area and stress constraints. The objective function and constraints of I-beam design are given as follows:

Minimize:

$$f(b,h,t_w,t_f) = \frac{5000}{\frac{t_w(h-2t_f)^3}{12} + \frac{bt_f^3}{6} + 2bt_f\left(\frac{h-t_f}{2}\right)^2}$$

Subject to cross section area less than 300 cm^2:

$$g_1 = 2bt_w + t_w(h-2t_f) \leq 300$$

Subject to bending stress:

$$g_2 = \frac{18h \cdot 10^4}{t_w(h-2t_f)^3 + 2bt_w\left(4t_f^2 + 3h(h-2t_f)\right)} + \frac{15b \cdot 10^3}{(h-2t_f)t_w^3 + 2t_w b^3} \leq 56$$

with:

$10 \leq h \leq 80$, $10 \leq b \leq 50$, $0.9 \leq t_w \leq 5$ and $0.9 \leq t_f \leq 5$