

T.C.
ERCIYES ÜNİVERSİTESİ
BİLİMSEL ARAŞTIRMA PROJELERİ
KOORDİNASYON BİRİMİ



**YAPAY ARI KOLONİSİ ALGORİTMASININ TEK MODLU SÜREKLİ
FONKSİYONLAR ÜZERİNDE HESAPLAMA KARMAŞIKLIĞININ
ANALİZİ**

Proje No: FBA-10-2959

Normal Araştırma Projesi

SONUÇ RAPORU

Proje Yürütücüsü:

Doç. Dr. Bahriye Akay
Mühendislik Fakültesi Bilgisayar Mühendisliği

Prof. Dr. Derviş Karaboğa
Mühendislik Fakültesi Bilgisayar Mühendisliği

Kasım 2013

KAYSERİ

TEŞEKKÜR

FBA-10-2959 kodlu "Yapay Arı Kolonisi Algoritmasının Tek Modlu Sürekli Fonksiyonlar Üzerinde Hesaplama Karmaşıklığının Analizi" isimli proje ERÜ BAP Birimi tarafından desteklenmiştir.

İÇİNDEKİLER

	Sayfa No
ÖZET	1
ABSTRACT	1
1. GİRİŞ	2
2.GENEL BİLGİLER	6
3. YAPAY ARI KOLONİSİ ALGORİTMASI	8
4. GEREÇ VE YÖNTEM	10
5. BULGULAR	14
6. TARTIŞMA VE SONUÇ	16

YAPAY ARI KOLONİSİ ALGORİTMASININ TEK MODLU SÜREKLİ FONKSİYONLAR ÜZERİNDE HESAPLAMA KARMAŞIKLIĞININ ANALİZİ

ÖZET: Evrimsel algoritmalar doğal seleksiyon sürecini modelleyerek tasarım problemlerini çözmede başarılı sonuçlar üreten algoritmalarlardır. Evrimsel algoritmaların uygulamalarının ve başarımlarının test edilmesine yönelik literatürde çok sayıda yayın olmasına rağmen yakınsaklık yada çalışma zamanlarının hesaplanmasına dair teorik çalışmaların sayısı kısıtlıdır. Yapay arı kolonisi algoritması da evrimsel algoritmalar gibi seleksiyon prensibine dayalı sürü zekası da kullanan bir algoritmadır. Yani sürü zekasının sağlamış olduğu positif geri besleme, negatif geri besleme, çoklu etkileşim ve salınım yapma gibi özellikleri taşımaktadır. Literatüre sunulan çok sayıda çalışma yapay arı kolonisi algoritmasının başarısını ortaya koymaktadır. Bu proje kapsamında da bu algoritmanın teorik incelemelerinin temeli atılmaktadır. Analizlerde kolaylık sağlaması bakımından yapay arı kolonisi algoritmasının binary çalışan versiyonunun tek modlu problemlerden olan ONEMAX problemi üzerinde analizi yapılmıştır.

ABSTRACT

Evolutionary algorithms modelling the natural selection are useful tools for solving the design problems. Although there are very large number of studies on the applications and performance tests of evolutionary algorithms in the literature, the number of studies on the theoretical analysis of evolutionary algorithms are very limited. Artificial bee colony algorithm is also based on the selection principle and swarm intelligence properties which are positive feedback, negative feedback, multiple interactions and fluctuations. In the literature there are too many studies related to the success of the artificial bee colony algorithm. In this project, the first fundamental theoretical analysis of artificial bee colony algorithm is presented. For the sake of simplicity, the binary version of the artificial bee colony algorithm is employed in the analysis on the unimodal test problem ONEMAX.

GİRİŞ / AMAÇ VE KAPSAM

Doğal seleksiyona dayalı sezgisel algoritmalar olan Evrimsel Algoritmalar çok çeşitli optimizasyon problemlerine uygulandığında iyi sonuçlar üretmektedir. Ancak uygulamalarının çokluğuna rağmen Evrimsel Algoritmaların teorik analizi ile ilgili çalışma sayısı kısıtlıdır. Bu çalışmanın amacı popülasyon tabanlı sezgisel algoritmalarından biri olan Yapay Arı Kolonisi (Artificial Bee Colony, ABC) algoritmasının tek modlu problemler üzerinde teorik incelemesinin gerçekleştirilmesidir. Rastgelelik içeren sezgisel arama algoritmalarının analizinde kullanılan olasılıksal teknikler bu çalışma kapsamında incelenmiş ve ABC algoritmasının tek modlu sürekli fonksiyonların optimizasyonundaki karmaşıklığının hesaplanmasında kullanılmıştır.

Gelişime dayalı algoritmalarda bir başlangıç popülasyonu (n) rastgele üretilerek bir maliyet fonksiyonu tarafından değerlendirilir. Bu işlem sonucunda bir uygunluk değeri atanır ve rastgelelik içeren değişim operatörleri (reproduction/mutasyon) ile yeni çözümler ($n' < \infty$) üretilir. $n + n'$ tane bireye sahip yeni popülasyondan n tane birey seleksiyon işlemi sonucunda bir sonraki jenerasyonda yaşamına devam etmek üzere hayatta kalır. Bu adımlar bir durdurma kriteri sağlanana kadar devam eder. O anki popülasyon bir önceki popülasyonla stokastik olarak ilişkili olduğundan Markov özelliği gösterir. Dolayısıyla evrimsel algoritmaların stokastik davranışları araştırma uzayı ve popülasyon büyüklüğü sonlu olduğu için sonlu Markov süreci olarak ifade edilebilir. k . adımdaki rastgele popülasyon X_k ve $F_k = \max\{f(X_{k,1}), f(X_{k,2}), \dots, f(X_{k,n})\}$ bu adımdaki bireylere ait uygunluk değerlerinden en iyisi olmak üzere, F_k 'nın $f^* = \max\{f(x) : x \in X\}$ optimum değerine yakınsaması popülasyonun global optimumu içermesi ile garantilenir. Bu olay başlangıç popülasyonunun ne olduğundan bağımsız olarak $p=1$ olasılıkla sonlu sayıda basamak sonrası oluşacaktır. Evrimsel algoritmanın optimum çözümü ilk olarak yakaladığı adıma ilk vuruş zamanı (first hitting time) denmektedir ve şu şekilde formüle edilir: Evrimsel algoritma $\Pr[T < \infty] = 1$ olasılıkla, yani kesinlikle sonlu sayıda adımda, ilk vuruş zamanı $T = \min\{t \geq 0 : F_t = f^*\}$ sürede optimum çözümü yakalar.

Evrimsel algoritmaların NP-Hard türdeki problemleri polinomsal zamanda çözmesi beklenmediğinden bu tür problemler üzerinde yakınsama oranları değerlendirilerek en iyi yakınsak algoritmalar ile kıyaslanırlar.

Evrimsel algoritmaların durum geçişlerinin stokastik bir yapısı olmasından dolayı burada optimuma yakınsama deterministik bir yapıda gerçekleşmez. Buradaki yakınsama stokastik yakınsamadır (Rudolph, 1998).

$D_0, D_1, \dots$ olasılık uzayında negatif olmayan rasgele değişkenler olmak üzere D_k değerleri tüm $\varepsilon > 0$ değerleri için $\sum_{k=0}^{\infty} P\{D_k > \varepsilon\} < \infty$ ise, tamamen sifıra yakınsıyor (converge completely to zero);

$P\{\lim_{k \rightarrow \infty} D_k = 0\} = 1$ ise, 1 olasılıklı yakınsama (converge with probability 1, almost surely to zero);

$k \rightarrow \infty$ için $P\{D_k > \varepsilon\} = o(1)$ ise, sifıra olasılıkta zayıf yakınsama (converge in probability to zero);

$k \rightarrow \infty$ için $E[D_k] = o(1)$ ise, ortalamada yakınsaklık (converge in mean to zero) olduğu söylenir.

Tam yakınsama 1 olasılıkla yakınsamayı kapsar; 1 olasılıkla yakınsama ve ortalamada yakınsaklık olasılıklı yakınsamayı kapsar. D_k değerleri bir sabitle üstten sınırlı ise olasılıklı yakınsama ortalamada yakınsamayı kapsar. Bu tanımlar çerçevesinde evrimsel algoritmaların yakınsaklığı daha iyi anlaşılabilir.

X_k evrimsel algoritma tarafından üretilen popülasyonlar ve k jenerasyon sayısı, n problemin boyutu iken $F_k = \max\{f(X_{k,1}), f(X_{k,2}), \dots, f(X_{k,n})\}$ en iyi amaç fonksiyonu olsun. Bir evrimsel algoritmanın global maksimum $f^* = \max\{f(x) : x \in X\}$ 'e tamamen yakınsak (1 olasılıkla yakınsak, zayıf olasılıkla, ortalamada yakınsak) olabilmesi için negatif olmayan rastgele seri $D_k = f^* - F_k$ değerlerinin sifıra tamamen yakınsaması gerekir.

Bu durumda yakınsama sağlanabilmesi için global çözümün 1 olasılıkla ziyaret edilmesi önşarttır.

$(x_1, x_2, \dots, x_n) \in X^n$ popülasyondaki bireyleri temsil etmek üzere bireyler sırasıyla aşağıdaki operatörlerden geçirilirler:

Eş seçimi: $mat = X^n \rightarrow X^\rho$, $2 \leq \rho \leq n$.

Re-kombinasyon: $reco = X^\rho \rightarrow X$. Eş seçimi ile seçilen 2 veya daha fazla bireyin bilgileri kullanılarak yeni bir birey üretilir.

Mutasyon: $mut : X \rightarrow X$. re-kombinasyon ile üretilen bireyde mutasyon operatörü aracılığı ile bazı değişimler yapılır.

Seleksiyon: $sel : X^k \rightarrow X^n$. üretilen yeni bireyler dahilinde oluşan k bireye sahip popülasyon popülasyon büyüklüğünü n değerine sabitlemek amacıyla seleksiyona tabi tutulurlar. k bireyden bazı seleksiyon operatörleri (rulet tekerleği, turnuva, sıralama vb.) kullanılarak n tanesi seçilir.

Her bir jenerasyonunda bu işlemlerin yapıldığı evrimsel algoritma ile ilgili şu kabuller yapılabilir:

A1: $\forall x \in (x_1, \dots, x_n) : P\{x \in reco(mat(x_1, \dots, x_n))\} \geq \delta_r > 0$. Her bir bireyin seçilme şansı vardır ve re-kombinasyon operatörü ile değişme olasılığı δ_r 'den büyüktür.

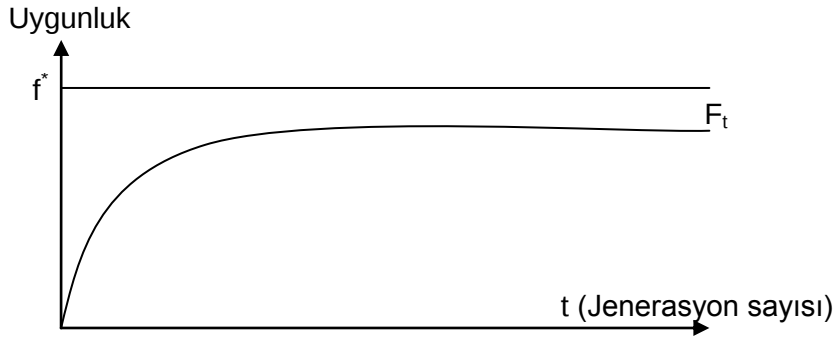
A2: Her $x, y \in X$ çifti için $P\{x_{i+1} = mut(x_i)\} \geq \delta_m > 0$ olacak şekilde $x_1 = x$, $x_k = y$, $x_1, x_2, \dots, x_k$ dizisi vardır. Yani her bir birey ardışıl mutasyonlar sonucu başka bir bireye dönüşebilir.

A2': Her $x, y \in X$ çifti $P\{y = mut(x)\} \geq \delta_m > 0$ sağlar. Her nokta başka bir noktanın mutasyona uğramasıyla elde edilebilir. A2'den farklı olarak tek bir mutasyonla diğer bireye dönüşebilir.

A3: $\forall x \in (x_1, \dots, x_n) : P\{x \in sel(x_1, \dots, x_n)\} \geq \delta_s > 0$. Her elemanın seçilme olasılığı vardır.

A4: $v_k^*(x_1, \dots, x_k) = \max\{f(x_i) : i = 1, \dots, k\}$ popülasyon içerisindeki k ($k \geq n$) tane bireyden en iyi uygunluk değerini göstermek üzere seleksiyon metodu

$P\{v_n^*(sel(x_1, \dots, x_k)) = v_k^*(x_1, \dots, x_k)\} = 1$ şartını sağlar. Yani en iyi bireyin seleksiyon operatörü ile hayatta kalma olasılığı 1'dir; kaybolma olasılığı yoktur. Evrimsel algoritma uygulamalarında en iyi çözüm popülasyon içinde tutulmasa bile hafıza da saklandığından dolayı bu özellik sonlu bir zaman içerisinde global optimumun bulunacağını ve kaybedilmeyeceğini garantiler.



$f : S \rightarrow R$ amaç fonksiyonu olmak üzere evrimsel algoritmanın yakınsama olasılığı 1'dir.

$f^* = \max_{x \in S} f(x)$, f fonksiyonunun maksimum değeri

$F_t = \max_{x \in P_t} f(x)$, t anındaki P_t popülasyonun bireyleri arasındaki maksimum değer olmak üzere

$\Pr[\lim_{t \rightarrow \infty} f^* - F_t = 0] = 1$ 'dir. Yada başka bir deyişle global çözümün ilk vuruş zamanı

$T = \min\{t \geq 0 : F_t = f^*\}$ olmak üzere $\Pr[T < \infty] = 1$ 'dir.

Evrimsel algoritmanın beklenen ilk vuruş zamanının değeri evrimsel algoritmanın türüne ve

ele alınan probleme göre değişmektedir. Örneğin $f(x) = \sum_{i=1}^{\ell} \prod_{j=1}^i x_j$ olan bir fonksiyonun

beklenen ilk vuruş zamanı rastgele seçilen tek bir bit mutasyona uğradığı zaman $E[T] \leq \ell^2$,

birden fazla bit mutasyon oranı dahilinde rastgele olarak değişime uğruyorsa $p = 1/\ell$

olasılıkla $E[T] \leq \ell^2(\exp(1) - 1)$ olarak elde edilmiştir. Unimax probleminde ise rastgele seçilen

tek bir bit mutasyona uğradığı zaman $E[T] \leq 3 \cdot \ell \cdot 2^{(\ell-1)/2} - 2\ell$, birden fazla bit mutasyon oranı

dahilinde rastgele olarak değişime uğruyorsa $p = 1/\ell$

olasılıkla $E[T] \leq (\ell^3 - \ell)(\exp(1)/2)$ olarak elde edilmiştir (Rudolph, 1997).

Yapay Arı Kolonisi Algoritması (Karaboga, 2005) gerçek arıların yiyecek arama davranışından esinlenilerek geliştirilmiş bir optimizasyon algoritmasıdır. Sınırlamasız ve sınırlamalı türden optimizasyon ve mühendislik tasarım problemlerinin çözümünde oldukça başarılı sonuçlar üretmektedir. Literatürde kombinasyonel türden olan problemlerin çözümünde de kullanılmıştır. Doğru ve kararlı sonuçlar üretmenin yanı sıra bir optimizasyon

algoritmasından beklenen hesaplama karmaşıklığının düşük olmasıdır. Yapay Arı Kolonisi algoritmasının doğruluğu ve kararlılığını ortaya koyan çok sayıda çalışma olmasına rağmen ortalama hesaplama süresi yada karmaşıklığı ile ilgili çalışma bulunmamaktadır. Bu çalışmada tek modlu problemlerin çözümünde yapay arı kolonisi algoritmasının ilk vuruş zamanının alt ve üst limitlerinin çıkarılarak algoritmanın problem türünde olan verimliliği incelenmiştir.

GENEL BİLGİLER

Evrimsel Algoritmaların Teorik Analizi üzerine Yapılan Çalışmalar

Evrimsel algoritmaların çalışma sürelerinin analizi ile ilgili literatürde sayıları kısıtlı olmakla birlikte bazı çalışmalar bulunmaktadır. Eiben ve ark.. (1991) evrimsel algoritmanın gelişim sürecini bir Markov zinciri olarak tanımlayarak EA'ların optimum çözümü $p=1$ olasılıkla bulabilmesinin garanti olduğu şartları tanımlamışlardır. Droste ve ark. (1998) binary girişli doğrusal fonksiyonların çözümünde farklı mutasyon oranlarına sahip $(1+1)$ EA'ların hesaplama karmaşıklıklarını analiz ederek n değişkenli çoğu doğrusal fonksiyonda bu türdeki evrimsel algoritmaların beklenen çalışma zamanlarının $O(n \ln n)$ olduğunu belirtmişlerdir. Rudolph (1998) sonlu bir uzayda ayırık zamanda limit davranışı üzerinde bir inceleme yapmıştır. He ve Huang (1998) birden fazla bireye sahip, çaprazlama ve mutasyon uygulayabilen EA'ların çalışma zamanını inceleyerek bazı alt küme toplamı (subset sum) problemlerinin EA'lar ile polinomsal zamanda çözülebilirken bazılarının ise en az üstel zaman gerektirdiğini göstermişlerdir. He ve Yao (2001) EA'ların en fazla polinomsal zaman ve problemlerin ise en az üstel zaman gerektirdiği koşullar altında bir drift (kayma) analizi yapmışlardır. Bu çalışmada incelenen EA çaprazlama, mutasyon ve seleksiyon operatörleri içermektedir ve sonlu bir popülasyon ile çalışmaktadır. Yapılan drift analiz ile ilk vuruş zamanının alt ve üst sınırlarını elde etmişlerdir. He ve Yao (2002) $(N+ N)$ evrimsel algoritmanın ilk vuruş zamanını elde ederek EA'ların ortalama çalışma zamanları üzerinde popülasyonun etkisini incelemişlerdir. He ve Yao (2003) ilk vuruş zamanlarının analizinde Markov zincirlerini kullanan bir çalışma yaparak farklı EA'ları sistematik olarak incelemişlerdir. He ve Yao (2004) drift analiz ile iki EA'ya ait ilk vuruş zamanlarının alt ve üst limitlerini elde etmişlerdir. He ve Yao (2006) ölçeklenebilir paralel bir EA ve ilk vuruş zamanına dayalı olarak hızlanmasını tanımlamışlar ve popülasyon büyüklüğünün hızlanması üzerindeki etkisini incelemişlerdir.

Ding ve Bi (2007) Markov zincirlerine dayalı ilk vuruş zamanlarının analitik bir ifadesini vermişlerdir ve drift analizle birlikte Dynkin denklemi kullanarak alt ve üst limit elde

etmişlerdir. Zhou ve He (2007) EA'ların karmaşıklığını sınırlamalı problemler üzerinde inceleyerek penaltı katsayılarının zaman karmaşıklığı üzerindeki etkisini araştırmışlardır. Penaltı katsayılarındaki polinomsal bir değişikliğin zaman karmaşıklığı üzerinde üstel bir etki yarattığını belirtmişlerdir. Chen ve ark. (2009) tek modlu problemler (leadingones, onemax) üzerinde popülasyon tabanlı (N+N) evrimsel algoritmaların zaman karmaşıklığını çıkarmak için genel bir yaklaşım geliştirmişlerdir. Seleksiyon biriminin süresini formülize ederek ilk vuruş zamanının ortalama değerini elde etmişlerdir. Yua ve Zhou (2008) farklı konfigürasyondaki evrimsel algoritmaların (üç mutasyon operatörü, popülasyonlu ve popülasyonsuz, çaprazlama ve zamanla değişken mutasyonlu) yakınsama hızı ve ilk vuruş zamanını ilişkilendirerek incelemişlerdir. Olivetto ve Witt (2008) iki şartın sağlanması gereken drift analizin ispatı için kolay bir metot önermişlerdir. Olivetto ve ark. (2007) (1+1) EA'ların vertex cover örnekleri üzerinde beklenen çalışma sürelerinin teorik analizini sunmuşlardır ve çoğul koşullar kullanılarak her örnek için polinomsal zaman içerisinde optimumuna erişilebileceğini belirtmişlerdir.

Witt (2005) (1+1)-EA ve bir RLS algoritmasının her ikisinin de NP-Hard olan bölümlenme problemi üzerinde $O(n^2)$ beklenen karmaşıklıkla en az $4/3$ -yakınsak bir çözüm üretebileceklerini ve çoğul koşullar yapılırsa her iki algoritmanın da polinomsal yakınsal şemalar (PRAS) olduklarını göstermiştir. Storch (2006a) Metropolis algoritmasının rastgele ve yarı rastgele düzlemsel (planar) graphlar üzerinde yüksek olasılıkla $\Theta(n)$ zamanda maksimum klikleri bulabildiğini ve (1+1)-EA'nın da $\Theta(n^6)$ jenerasyonda bulabildiğini göstermiştir. Storch (2006b) (1+1)-EA'lerle kıyaslandığında popülasyon kullanımının faydalı olduğunu göstermiştir. Friedrich ve ark. (2007) (1+1)-EA'lar bazı vertex cover problemleri üzerinde kötü yakınsaklığa sahipken, basit çok kriterli evrimsel iyileyicinin (simple evolutionary multi-objective optimizer, SEMO) daha verimli olduğunu göstermişlerdir (Olivetto et al. 2007b).

Olivetto ve ark. (2007a) çoğul koşullarla (1+1)-EA'nın bazı vertex cover problemleri üzerinde yakınsayabildiğini göstermiştir. Olivetto ve ark. (2009) RLS algoritması ve (1 + 1)-EA ile ilgili teorik analiz sunarak (1 + 1)-EA'nın bipartite örneği üzerinde yakınsaklığını ispat etmiştir ve restart stratejisi ile de grafin minimum cover'ını hızlı bir şekilde bulabildiğini göstermişlerdir. Vertex derecesi en fazla iki iken algoritmanın çözümü polinomsal zamanda çözebildiğini belirtmişlerdir.

Baswana ve ark. (2009) n düğümlü yönlendirilmiş bir grafta en kısa yol probleminin çözümünün analizinde beklenen optimizasyon süresinin üst sınırını bulabilmek için optimal

özüm ile o anki en iyi özümün uygunluęu arasındaki fark gap i'nin beklenen deęerinin ifadesini vermiřlerdir.

YAPAY ARI KOLONİSİ ALGORİTMASI

Yapay arı kolonisi algoritmasında (Karaboga, 2005), bir koloni de üç grup arı bulunmaktadır: işi arılar, gözcü arılar ve kařif (scout) arılar. Modelimizde, koloninin yarısı işi, yarısı gözcü arı olarak seçilmiřtir. Her bir nektar kaynaęı için sadece bir işi arı bulunmaktadır. Yani işi arıların sayısı nektar kaynaęı sayısına eřittir. Algoritmanın temel adımları ise řu řekildedir:

Initialization

REPEAT

- İşi arıları kaynaklara gönder ve nektar miktarlarını hesapla
- Gözcü arıları kaynaklara gönder ve nektar miktarlarını hesapla
- Rasgele yeni kaynaklar bulmaları için kařif arıları gönder
- O ana kadarki en iyi kaynaęı hafızada tut

UNTIL (durma kriteri saęlanana kadar)

Her bir çevrim üç adımdan oluřmaktadır: işi ve gözcü arıların kaynaklara gönderilmesi, gidilen kaynakların nektar miktarlarının hesaplanması, kařif arının belirlenerek yeni bir kaynaęa rasgele konumlanması. Yiyecek kaynakları optimize edilmeye alışılan problemin olası özümlerine karşılık gelmektedir. Bir kaynaęa ait nektar miktarı, o kaynakla ifade edilen özümün kalite deęerini ifade etmektedir. Gözcü arılar rulet tekerleęi prensibine göre gidecekleri kaynakları belirlemektedirler. Her kolonide rastgele araştırma yapan kařif arılar bulunmaktadır. Bu arılar yiyecek ararken herhangi bir ön bilgi kullanmamakta, tamamen rastgele araştırma yapmaktadırlar. Dolayısıyla arama maliyetleri düşüktür ve de buldukları kaynaęın ortalama kalite deęeri düşüktür. Zengin nektar kaynaęına sahip keřfedilmemiř kaynakları bulmaları da olasıdır. ABC algoritmasında işi arılardan biri seçilerek kařif arı haline gelmektedir. Bu seçme işleminin “limit” parametresine göre yapılmaktadır. Bir kaynaęı ifade eden özüm belli sayıdaki deneme ile geliřtirilememiře bu kaynak terk edilir ve bu kaynaęa gidip gelen işi arı kařif arı haline gelir. Kaynaęın terk edilmesi için belirlenmiř deneme sayısı “limit” parametresi ile belirlenmektedir.

Gürbüz bir arama sürecinde keşif (exploration) ve keşfedilenden faydalanma (exploitation) aynı anda gerçekleşmelidir. ABC algoritmasında gözcü ve işçi arılar keşfedilen kaynaklardan faydalanma işleminde, kaşif arılar ise keşif sürecinde görev alırlar. Gerçek arılarda taşıma hızı koloninin bir kaynağı bulması ve onu kovana getirmesi ile belirlenirken, yapay arılar durumunda bulunan çözümün kalite değeri yani uygunluğu ile belirlenir.

Diğer sosyal yiyecek arayıcıları gibi, arılar E/T değerini yani birim zamanda yuvaya getirilen yiyecek miktarını belirten enerji fonksiyonunu maksimize etmek için çalışırlar. Bir maksimizasyon probleminde de amaç fonksiyonunun $F(\theta_i)$, $\theta_i \in R^p$, maksimize edilmesi işlemi gerçekleşir. θ_i , i. kaynağın pozisyonu olmak üzere $F(\theta_i)$ bu nektar miktarına karşılık gelir ve $E(\theta_i)$ ile orantılıdır. c çevrim sayısı (cycle), S: kovan etrafındaki nektar kaynağı sayısı olmak üzere $P(c) = \{\theta_i(c) | i = 1, 2, \dots, S\}$ tüm kaynakların pozisyon bilgilerini içeren nektar kaynağı popülasyonudur. Daha önce belirtildiği gibi gözcü arıların bir kaynağı seçmeleri $F(\theta)$ değerine bağlı idi. Kaynağın nektar miktarı ne kadar fazla olursa, bu kaynağı bir gözcü arı tarafından seçilme olasılığı o kadar fazla olmaktadır. Yani θ_i pozisyonundaki bir kaynağın seçilme olasılığı şu şekildedir:

$$P_i = \frac{F(\theta_i)}{\sum_{k=1}^S F(\theta_k)} \quad (1)$$

Gözcü arı, işçi arıların dansını izledikten ve (1) eşitliğindeki olasılık değeri ile θ_i konumundaki kaynağı seçtikten sonra, bu kaynağın komşuluğunda bir kaynak belirler ve kaynağın nektarını almaya başlar. Yani θ_i civarındaki kaynaklar arasında bir kıyaslama yapar. Seçilen komşuya ait pozisyon bilgisi şu şekilde hesaplanmaktadır:

$$\theta_i(c+1) = \theta_i(c) \pm \phi_i(c) \quad (2)$$

$\phi_i(c)$, θ_i civarında daha fazla nektara sahip bir kaynak bulabilmek için kullanılan, rastgele üretilen adım büyüklüğüdür. $\phi_i(c)$, k i'den farklı rastgele üretilen popülasyondaki bir çözüme ait indis olmak üzere $\theta_i(c)$ ve $\theta_k(c)$ çözümlerinin bazı bölümlerinin farkının alınması ile hesaplanır. $\theta_i(c+1)$ e ait nektar miktarı $F(\theta_i(c+1))$, $\theta_i(c)$ konumundaki kaynağa ait nektar miktarından daha fazla ise arı kovana giderek bu bilgisini diğerleri ile paylaşır ve yeni pozisyon olarak $\theta_i(c+1)$ aklında tutar, aksi durumda $\theta_i(c)$ yi hafızasında saklamaya devam

eder. θ_i konumundaki nektar kaynağı “limit” parametresi sayısınca gelişmemiş ise θ_i deki kaynak terk edilir ve o kaynağın arısı kaşif arı haline gelerek rasgele araştırma yapar, yeni bulunduğu kaynak θ_i ye atanır.

GEREÇ VE YÖNTEM

Evrimsel algoritmaların detaylı analiz edilebilmesi içi olasılık teorisi ile ilgili bazı tekniklere ihtiyaç duyulmaktadır. X_f bir evrimsel algoritmanın bir f fonksiyonu üzerindeki çalışmasına karşılık gelen rastgele bir değişken olsun. Teorik analizde, $E(X_f)$ 'nin kestiriminde en iyi (best), ortalama (average) ve en kötü (worst) durum analizleri yapılarak başarı olasılığı olan $P_r(X_f \leq T)$ hesaplanır.

Devralma Süresi (takeover time) [6]: Gelişime dayalı algoritmalarda kullanılan seleksiyon şemasının devralma süresi en iyi bireylerin başka operatörler olmaksızın sadece seleksiyon operatörü ile bir popülasyonu doldurması için gerekli iterasyon sayısıdır. A tekrarlanan bir seleksiyon süreci olmak üzere, seleksiyon operatörü her bir iterasyonda bir sonraki popülasyonu oluşturmak üzere o anki popülasyondan popülasyon boyutu sabit kalmak üzere bazı bireyleri seçer. Nt t. iterasyonda ($t \in \mathbb{N}_0$) en iyi uygunluk değerine sahip bireylerin sayısı olmak üzere, A seleksiyon sürecinin devralma süresi $\eta = \min\{t; Nt = N\}$ ile ifade edilir. Seleksiyon operatörü için geliştirilmiş bu metrik mutasyon operatörü içinde genelleştirilerek kullanılabilir.

Evrimsel algoritmaların kuyruk eşitsizlikleri (tail inequality) ile analizinde en sık kullanılan rastgelen değişken türü binary yada Bernoulli rastsal değişkenidir. Bernoulli rastsal değişkeni olası iki sonuç olayından birini ifade eden bir indikatör değişkendir ve genellikle sayma için kullanılır. Binary bir rastgele değişkenin olasılık dağılımı $\Pr(X = 1) = p$ ve $\Pr(X = 0) = 1 - p$ ile tanımlanır. Eşit p olasılıklı rastgele değişkenlerle tanımlanan n tane olayın kombinasyonu olan bir X rastgele değişkeni rastsal binom değişkeni olarak isimlendirilir. Tüm $k \in \{0, \dots, n\}$ değerleri için X değişkeninin k değerini alma olasılığı $\Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$ ile tanımlanır. Bazı durumlarda belli bir olayın oluşması için ne kadar zaman geçmesi gerektiği ile ilgileniriz. Yani $Y = \min\{k \in \mathbb{N} | X_k = 1\}$ olan Y rastgele değişkeni ile ilgileniriz. Bu Y rastgele değişkenine geometrik rastgele

değişken adı verilir. k. denemede olayın oluşma ihtimali yani k-1 kere başarısız ve 1 kere başarılı olma olayının ihtimalini binom dağılımından $\Pr(Y = k) = (1 - p)^{k-1} p$ ile bulunur.

Bekleme Zamanı Argümanı (Waiting Time Argument) Başarılı olma olasılığı p olan bir geometrik rastgele değişken X için beklenen zaman $E(X) = 1/p$ 'dir. Bu bir başarılı olay elde edene kadar beklenen deneme sayısının $1/p$ olduğu anlamına gelmektedir.

$$E(X) = \sum_{i=1}^{\infty} iP(X_i = k) = \sum_{i=1}^{\infty} i(1-p)^{i-1} p$$

$$E(X) = \sum_{i=1}^{\infty} (1-p)^{i-1} p + \left(\sum_{i=2}^{\infty} (i-1)(1-p)^{i-2} p \right) (1-p)$$

$$E(X) = 1 + \left(\sum_{t=1}^{\infty} t(1-p)^{t-1} p \right) \rightarrow (1-p)^{t-1} p = P(t)$$

$$E(X) = 1 + E(X)(1-p)$$

$$E(X)p = 1$$

$$E(X) = 1/p$$

Tek Arı ile çalışan Binary ABC algoritmasının $\{0,1\}^n$ uzayında tanımlı bir fonksiyon için beklenen çalışma zamanı en fazla n^n 'dir.

O anki çözüm ile global optimum arasındaki farklı bit sayısı i olsun. Dolayısı ile (n-i) tane bit olması gereken pozisyonlardır. Optimum çözüme ulaşmak için doğru pozisyonları bitlerin değişmemesi ve i tane yanlış pozisyonda olan bitin ise değişmesi, mutasyona uğraması gerekmektedir. Her bir bitin değişme olasılığı $1/n$ 'dir. Değişmeme olasılığı ise $(1-1/n)$ 'dir. Optimuma ulaşma olasılığının alt sınırı:

$$P(x^* | x) = \left(\frac{1}{n} \right)^i \left(1 - \frac{1}{n} \right)^{(n-i)} \geq \left(\frac{1}{n} \right)^n = n^{-n}$$

Buradan beklenen zamanın üst sınırı ise n^n olarak bulunur.

Kuyruk Eşitsizlikleri

Bir sezgisel algoritmanın analizinde rastgele bir değişkenin beklenen değerlerinden ziyade yüksek olasılıkla sağlanan sınırlarla ilgileniriz. Öncelikle, çalışma zamanının beklenen değeri hesaplanır ve çalışma zamanının bu beklentiyi belli bir miktar aşma olasılığı sınırlanır.

Bu eşitsizliklere kuyruk eşitsizlikleri yada büyük sapma eşitsizlikleri adı verilir (Doerr, 2011).

Bağımsız Rastgele Değişkenlerin Toplamı için Chernoff Sınırları (Doerr, 2011)

Poisson denemeleri olarak bilinen bağımsız 0-1 değişkenlerinin toplamının kuyruk dağılımının Chernoff sınırı Markov eşitsizliği uygulanarak elde edilir.

Teorem 1. $X_1, \dots, X_n$ $[0,1]$ aralığında değer alan bağımsız rastgele değişkenler ve

$X = \sum_{i=1}^n X_i$ bunların toplamı olmak üzere:

$$a) \delta \in [0,1] \text{ ise } \Pr(X \leq (1 - \delta)E(X)) \leq \left(\left(\frac{1}{1 - \delta} \right)^{1 - \delta} e^{-\delta} \right)^{E(X)}$$

$$b) \delta > 0 \text{ ise } \Pr(X \geq (1 + \delta)E(X)) \leq \left(\frac{e^{\delta}}{(1 + \delta)^{1 + \delta}} \right)^{E(X)}$$

İyi belirlenmiş bir t değeri için e^{tX} 'ye Markov eşitsizliği uygulayarak Chernoff sınırını elde edebiliriz. ($e^x \geq 1 + x$)

$$t > 0 \text{ için } \Pr[X \geq a] = \Pr[e^{tX} \geq e^{ta}] \leq \frac{E[e^{tX}]}{e^{ta}}. \text{ Dolayısıyla } \Pr[X \geq a] \leq \min_{t > 0} \frac{E[e^{tX}]}{e^{ta}}$$

$$t < 0 \text{ için } \Pr[X \leq a] = \Pr[e^{tX} \geq e^{ta}] \leq \frac{E[e^{tX}]}{e^{ta}}. \text{ Dolayısıyla } \Pr[X \leq a] \leq \min_{t < 0} \frac{E[e^{tX}]}{e^{ta}}$$

Chernoff sınırı beklenen değer en azından logaritmik ise polinomsal başarısızlık olasılıklarını, beklenen değer $\Theta(n)$ ise (yani X_i 'ler sabit olasılıklı iseler) exponansiyel başarısızlık olasılıklarını verir.

$X_1, \dots, X_n$, $\Pr[X_i = 1] = p_i$ olasılıklı bağımsız Poisson denemeleri serisi ve $X = \sum_{i=1}^n X_i$ olmak

üzere

$$\mu = E[X] = E\left[\sum_{i \in [n]} X_i\right] = \sum_{i \in [n]} E[X_i] = \sum_{i \in [n]} p_i$$

$\delta > 0$ için $\Pr[X \geq (1 + \delta)\mu]$ ve $\Pr[X \leq (1 - \delta)\mu]$ sınırları aranmaktadır.

Chernoff sınırını elde edebilmek için X 'in moment çıkaran fonksiyonunun hesaplanması gerekmektedir. Moment çıkaran fonksiyon aşağıdaki gibidir:

$$M_X(t) = E[e^{tX}, t \in R]$$

$$M_X(0) = 1$$

$$E(e^{tX}) = \sum_{k=0}^{\infty} e^{tk} f(k; \lambda) = \sum_{k=0}^{\infty} e^{tk} \frac{\lambda^k e^{-\lambda}}{k!} = e^{\lambda(e^t - 1)}$$

$$M_{X_i}(t) = E[e^{tX_i}]$$

$$= p_i e^{t \cdot 1} + (1 - p_i) e^{t \cdot 0}$$

$$= p_i e^t + (1 - p_i)$$

$$M_{X_i}(t) \leq e^{p_i(e^t - 1)}$$

$$M_X(t) = \prod_{i \in [n]} M_{X_i}(t)$$

$$M_X(t) \leq \prod_{i \in [n]} e^{p_i(e^t - 1)}$$

$$M_X(t) = \exp \left\{ \sum_{i \in [n]} p_i (e^t - 1) \right\}$$

$$M_X(t) = e^{(e^t - 1)\mu}$$

$$\Pr[X \geq (1 + \delta)\mu] = \Pr[e^{tX} \geq e^{t(1+\delta)\mu}]$$

$$\leq \frac{E[e^{tX}]}{e^{t(1+\delta)\mu}}$$

$$\leq \frac{e^{(e^t - 1)\mu}}{e^{t(1+\delta)\mu}}$$

$$t = \ln(1 + \delta) > 0$$

$$\Pr[X \geq (1 + \delta)\mu] < \left(\frac{e^\delta}{(1 + \delta)e^{(1+\delta)}} \right)^\mu$$

$$0 < \delta < 1$$

$$\Pr[X \leq (1 - \delta)\mu] \leq \left(\frac{e^{-\delta}}{(1 - \delta)e^{(1-\delta)}} \right)^\mu$$

Kupon Toplama Problemi (Coupon Collector)

Yerine koyma gerçekleştirmeden toplanması gereken n tane kuponumuz olduğunu farzedelim. n tane farklı kuponun hepsinin de toplanmış olması için gereken deneme sayısının

t denemeden fazla olması olasılığı nedir sorusuyla ilgilenir. Problemin matematiksel analizi ile denemelerin beklenen sayısının $\Theta(n \log(n))$ olduğunu göstermektedir.

Teorem 2. (Kupon toplama sayısı beklenen değeri). $H_n := \sum_{i=1}^n \frac{1}{i}$, n. Harmonik sayı olmak üzere n tane kuponun toplanması için gereken zaman nH_n 'dir. $\ln(n) < H_n < 1 + \ln(n)$ olduğu için toplayıcının $(1 + o(1))n \ln(n)$ zamana ihtiyacı vardır.

$$\begin{aligned} E(T) &= E(t_1) + E(t_2) + \dots + E(t_n) = \frac{1}{p_1} + \frac{1}{p_2} + \dots + \frac{1}{p_n} \\ &= \frac{n}{n} + \frac{n}{n-1} + \dots + \frac{n}{1} = n \left(\frac{1}{1} + \frac{1}{2} + \dots + \frac{1}{n} \right) = nH_n = n \ln n + \gamma_n + \frac{1}{2} + O(1) \end{aligned}$$

BULGULAR

Tek aralı binary ABC algoritmasının tek modlu ONEMAX problemi üzerindeki beklenen çalışma zamanı $\Omega(n \ln n)$ 'dir.

t adım süresince belli bir bitin değişmeme olasılığı $(1 - 1/n)^t$ 'dir. Dolayısıyla t adımda en azından bir kere değişme olasılığı $1 - (1 - 1/n)^t$. Başlangıç fazında 1/2 sabit olasılıkla en fazla n/2 tane 1 biti üretilir (Droste ve ark., 1998). n/2 tane bitin hepsinin en azından bir kere değişme olasılığı için ilgili olasılığın (n/2). kuvveti alınır ve bu olasılığında komplementi alınırsa n/2 bitten en az birinin $t = (n-1) \ln n$ adımda değişmeme olasılığının sınırını elde ederiz (Oliveto and Yao, 2011):

$$1 - \left(1 - \left(1 - \frac{1}{n}\right)^{(n-1) \ln n}\right)^{n/2} \geq 1 - (1 - e^{-\ln n})^{n/2} = 1 - (1 - 1/n)^{n/2} \geq 1 - e^{-1/2}$$

Beklenen çalışma zamanı da en az $E(t) \geq (n-1) \ln n (1/2)(1 - e^{-1/2}) = \Omega(n \ln n)$ 'dır.

He ve Yao (2003) EA'ların ilk vuruş zamanlarının analizi için analitik bir yapı sunmuşlardır. Bir evrimsel algoritma Markov zinciri olarak modellendiğinde her bir populasyon Markov zincirindeki bir duruma karşılık gelir ve bir sonraki durum sadece bir önceki duruma bağlıdır. Bir evrimsel algoritmanın tüm gelişim süreci Markov zincirinin geçiş matrisinde depolanır. Algoritmanın ilk vuruş zamanı ve ilk vuruş zamanı için beklenen değerlerin ortalaması evrimsel algoritmanın karmaşıklığının bulunmasında kullanılır. Lineer geçiş matrisi sisteminin çözümü oldukça güç olduğundan Drift analizi (Hajek, 1982) ile ilk vuruş zamanının alt ve üst limitleri elde edilir. Büyük ölçekli problemlerde o anki popülasyonun optimal set ile arasındaki mesafenin hesabı zor olduğu için durumlar üzerinde

decomposition uygulanır. Stokastik bir süreç ξ durumlar ve t zamanı gösteren bağımsız değişkenler olmak üzere $x(\xi, t)$ şeklinde bir fonksiyon olarak tanımlanır. Durumlardaki değişimler geçişler olarak isimlendirilir ve bir durumdan diğerine geçiş olasılığına geçiş (transition) olasılığı denir. Ayrık durum uzaylı Markov sürecine Markov zinciri denir. $E = \{x_1, x_2, \dots, x_n\}$ durumlar kümesi olmak üzere X_n Markov zincirindeki $P_j(n) = P\{S_n = j\}$ ve $P_{jk}(n, m) = P\{S_m = k \mid S_n = j\}$ şartlı olasılıklarının tüm $n \geq m \geq 0$ zaman parametreleri ve $j, k \in E$ durumları için hesaplanması gerekir. Geçiş matrisinin özellikleri şu şekildedir:

$$P_{ij}(m, n) \geq 0, \text{ for } \forall i, j \in E, \forall m, n \in T$$

$$\sum_j P_{ij} = 1 \text{ for } \forall i \in E$$

$$P(m, n) = P(m, u)P(u, n), m \leq u \leq n$$

Geçiş matrisini kanonik formda yazacak olursak T geçici (transient) durumlar ve $S-T$ absorbe edici durumlar kümesi olmak üzere

$$P = \begin{pmatrix} I & 0 \\ R & T \end{pmatrix}$$

şeklinde ifade edebiliriz. T matrisindeki T_{ij} değerleri geçici durum i 'den geçici durum j 'ye geçiş olasılığıdır. R matrisi $i \in T$ geçici durumundan $j \in S - T$ absorbe edici duruma geçiş olasılıklarını tutar. $N = (I - T)^{-1}$ absorbe edici Markov zincirinin temel (fundamental) matrisidir. zincir i . durumda başladığında tekrarlı (recurrent) durumlara erişmek için beklenen zaman m_i olmak üzere $m = N1 = (I - T)^{-1}1$ ortalama soğurma (absorption) süreleri vektörüdür (Oliveto and Yao, 2011). $(I-T)$ 'nin inversinin alınmasından dolayı çoğu problemin temel matrisini elde etmek zordur. Tek aralı binary ABC için olasılık matrisi

$$P = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ p_{(1,0)} & p_{(1,1)} & 0 & \dots & 0 \\ p_{(2,0)} & p_{(2,1)} & p_{(2,2)} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{(N,0)} & p_{(N,1)} & p_{(N,2)} & \dots & p_{(N,N)} \end{pmatrix}$$

olarak elde edilir. Buradan ilk vuruş zamanı $m = N1 = (I - T)^{-1}1$ aşağıdaki gibi elde edilebilir:

$$\begin{cases} m_0 = 0, & i \in S_0 \\ m_1 = 1/p_{(1,0)}, & i \in S_1 \\ m_2 = \frac{1 + p_{(i,1)}m_1 + p_{(i,2)}m_2 + \dots + p_{(i,i-1)}m_{i-1}}{p_{(i,0)} + p_{(i,1)} + p_{(i,2)} + \dots + p_{(i,i-1)}}, & i = 2, \dots, N \end{cases}$$

Binary ABC algoritması i tane sıfır bit içeren bir çözümle başlatılırsa ONEMAX problemini optimize etmek için nH_i adıma ihtiyaç duyar.

ONEMAX probleminin araştırma uzayı $n+1$ farklı uygunluk seviyesine sahip olduğundan matris boyutu $(n+1) \times (n+1)$ 'dir. O anki çözümün i tane sıfır bitine ve $n-i$ tane bir bitine sahip olsun. Binary ABC algoritması her adımda sadece bir biti değiştireceğinden bir tane sıfır bitinin değiştirilmesi olasılığı i/n ve bir tane bir bitinin değiştirilme olasılığı ise $(n-i)/n$ 'dir. Bir bitlerinin bir tanesi değiştiği zaman uygunluk değeri azalacak ve yeni çözüm aç gözlü seleksiyon işleminde atılacaktır. Durum RLS algoritmasının kanonik matrisi (Oliveto and Yao, 2011) ile aynı hale gelecektir:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 1/n & (n-1)/n & 0 & 0 & \dots & 0 & 0 \\ 0 & 2/n & (n-2)/n & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & (n-1)/n & 1/n & 0 \\ 0 & 0 & 0 & 0 & \dots & 1 & 0 \end{pmatrix}$$

Sistemin çözümü aşağıdaki gibidir:

$$\begin{cases} m_0 = 0 \\ m_j = \frac{1 + p_{(j,j-1)}m_{j-1}}{p_{(j,j-1)}} = \frac{1 + (j/n)m_{j-1}}{j/n} = n/j + m_{j-1}; 0 < j \leq n \end{cases}$$

Algoritma i tane sıfıra sahip bir çözüm ile başlarsa beklenen çalışma zamanı RLS algoritmasının (Oliveto and Yao, 2011) kadar olacaktır:

$$\sum_{k=1}^i (n/k) = n \sum_{k=1}^i (1/k) = nH_i$$

TARTIŞMA VE SONUÇ

Yapay arı kolonisi algoritması literatüre son yıllarda kazandırılmış optimizasyon problemleri üzerinde başarılı sonuçlar üreten sürü zekası temelli bir algoritmadır. Algoritmanın

performansına ve uygulamalarına dönük oldukça fazla sayıda bilimsel yayınlar üretilmiştir. Bu proje kapsamında da yapay arı kolonisi algoritmasının tek modlu problemler üzerindeki çalışma zamanı analizi yapılmıştır. Bazı olasılık kuramı tekniklerinin kullanılması ile ilk vuruş zamanının beklenen değerlerinin hesaplamalarının yapılmasında kolaylık olması açısından analizlerde tek arılı, binary uzayda çalışan ve komşuluk operatörü olarak bit değişimi uygulayan yapay arı kolonisi modeli kullanılmıştır ve tek modlu problemlerden olan ONEMAX üzerinde analizler yapılmıştır. Binary ABC algoritmasının ilk vuruş zamanının beklenen değeri için Markov zincirleri kullanılmıştır. Yani algoritmanın gelişim sürecinde o anki popülasyondan bir sonraki popülasyona geçiş Markov zinciri olarak modellenmiştir. Geçiş olasılıkları hesaplanmış ve kanonik matris formunda verilmiştir. Kanonik formda verilen sistem çözüldüğünde i tane sıfır bit içeren bir çözümle başlatıldığı zaman algoritmanın ONEMAX problemini optimize etmek için nH_i adıma ihtiyaç duyduğu sonucu elde edilmiştir.

$$\sum_{k=1}^i (n/k) = n \sum_{k=1}^i (1/k) = nH_i$$

Projenin amacı olan ABC algoritmasının tek modlu problemler üzerinde karmaşıklık analizi gerçekleştirilmiş olup sonuçlar literatüre sunulacaktır.

Çalışmanın ileriki safhalarında popülasyon ve komşuluk operatörlerine dayalı olarak algoritmanın modellenmesi yapılacak ve ilk vuruş zamanının beklenen değerinin alt ve üst limitleri elde edilecektir.

KAYNAKLAR

1. Hajek, B., Hitting-Time And Occupation-Time Bounds Implied By Drift Analysis with Applications, Adv. Appl. Prob. 14, 502-525, 1982.
2. Droste, S., Jansen, T., Wegener, I., A rigorous complexity analysis of the $(1 + 1)$ evolutionary algorithm for linear functions with boolean inputs, Evolutionary Computation 6 (2) , 185–196, 1998.
3. Rudolph, G., Finite Markov chain results in evolutionary computation: A tour d’horizon, Fundamenta Informaticae 35 (1–4), 67–89, 1998.

4. He J., Huang, H., The computational time analysis of genetic algorithms, in: Proc. Fifth Chinese Joint Conference on Artificial Intelligence, Xi'an Jiaotong University Press, Xi'an, pp. 440–443, 1998
5. He J., Yao, X, Drift analysis and average time complexity of evolutionary algorithms, Artificial Intelligence 127, 57–85, 2001.
6. He J., Yao, X., From an Individual to a Population: An Analysis of the First Hitting Time of Population-Based Evolutionary Algorithms, IEEE Transactions On Evolutionary Computation, Vol. 6, No. 5, 2002.
7. He J., Yao, X., Towards an analytic framework for analysing the computation time of evolutionary algorithms, Artificial Intelligence 145, 59–97, 2003.
8. He J., Yao, X., A study of drift analysis for estimating computation time of evolutionary algorithms, Natural Computing 3: 21–35, 2004.
9. He J., Yao, X., Analysis of Scalable Parallel Evolutionary Algorithms, 2006 IEEE Congress on Evolutionary Computation Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, 120-127, July 16-21, 2006.
10. Oliveto, P. S., He, J. , Yao, X., Evolutionary algorithms and the vertex cover problem, in Proc. IEEE CEC, Singapore, pp. 1870–1877, 2007.
11. Ding, L., Bi, Y., About the Time Complexity of Evolutionary Algorithms Based on Finite Search Space, CIS 2006, LNAI 4456, pp. 209–219, Springer-Verlag Berlin Heidelberg, 2007.
12. Zhou, Y., He, J., A Runtime Analysis of Evolutionary Algorithms for Constrained Optimization Problems, IEEE Transactions On Evolutionary Computation, Vol. 11, No. 5, October 2007.

13. Yua, Y., Zhou, Z-H., A new approach to estimating the expected first hitting time of evolutionary algorithms, *Artificial Intelligence*, 172 (15), 1809-1832, 2008.
14. Oliveto, P. S., Witt, C., Simplified Drift Analysis for Proving Lower Bounds in Evolutionary Computation, *LNCS: Parallel Problem Solving from Nature – PPSN X*, 5199/2008, 82-91, 2008.
15. Chen, T. , He, J., Sun, G., Chen, G., Yao, X., A New Approach for Analyzing Average Time
16. Complexity of Population-Based Evolutionary Algorithms on Unimodal Problems, *IEEE Transactions On Systems, Man, And Cybernetics—Part B: Cybernetics*, Vol. 39, No. 5, 2009.
17. Rudolph, G., *Convergence Properties of Evolutionary Algorithms*, Kovac, Hamburg, 1997.
18. Doerr, B. Book: *Theory of Randomized Search Heuristics*, Chapter: Analyzing Randomized Search Heuristics: Tools from Probability Theory, World Scientific Publishing, 2011.
19. Oliveto, P. Yao, X. Book: *Theory of Randomized Search Heuristics*, Chapter: Runtime Analysis of Evolutionary Algorithms for Discrete Optimization, World Scientific Publishing, 2011.
20. Oliveto, P., He, J., Yao, X, Time Complexity of Evolutionary Algorithms for Combinatorial Optimization: A Decade of Results, *International Journal of Automation and Computing*, 04(3), 281-293, 2007.
21. C. Gunia, On the Analysis of the Approximation Capability of Simple Evolutionary Algorithms for Scheduling Problems, *GECCO'05*, June 25–29, 2005, pp. 571-578, Washington, DC, USA.

22. C. Witt (2005), Worst-case and Average-case Approximations by Simple Randomized Search Heuristics. In Proceedings of the 22nd Annual Symposium on Theoretical Aspects of
23. Computer Science, Lectures Notes in Computer Science, Springer-Verlag, Stuttgart, Germany, vol. 3404, pp. 44-56, 2005.
24. T. Storch (2006a) How randomized search heuristics find maximum cliques in planar graphs. In Proceedings of the Annual Conference on Genetic and Evolutionary Computation, Seattle, Washington, USA, pp. 567-574, 2006.
25. T. Storch (2006b) Finding Large Cliques in Sparse Semi-random Graphs by Simple Randomised Search Heuristics, Technical Report No. CI-211/06, Fachbereich Informatik, Universitat Dortmund, 2006.
26. T. Friedrich, J. He, N. Hebbinghaus, F. Neumann, C. Witt (2007). Approximating Covering Problems by Randomized Search Heuristics Using Multi-objective Models. In Proceedings of
27. the 9th Annual Conference on Genetic and Evolutionary Computation, 2007.
28. F. Neumann, I. Wegener (2007), Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. Theoretical Computer Science, 378(1):32-40, 2007.
29. P. S. Oliveto, J. He, and X. Yao (2007a), Evolutionary algorithms and the vertex cover problem, in Proc. Congr. Evol. Comput. (CEC '07), Singapore, Sep. 2007, pp. 1870–1877.
30. P. Oliveto, J. He, X. Yao (2007b), Time Complexity of Evolutionary Algorithms for Combinatorial Optimization: A Decade of Results, International Journal of Automation and Computing, 04(3), 281-293, 2007.

31. P. S. Oliveto, J. He , and X. Yao (2009), Analysis of the (1+1)-EA for Finding Approximate Solutions to Vertex Cover Problems. *IEEE Transactions on Evolutionary Computation*, 13 (5):1006 -1029, 2009
32. S. Baswana, S. Biswas, B. Doerr, T. Friedrich, P. P. Kurur, and F. Neumann (2009), Computing single source shortest paths using single-objective fitness. In *FOGA'09: Proceedings of the 10th ACM Workshop on Foundations of Genetic Algorithms*, pages 59-66. ACM, 2009.
33. F. Neumann, C. Witt (2010), Makespan Scheduling, Book Title: *Bioinspired Computation in Combinatorial Optimization– Algorithms and Their Computational Complexity*, Springer.
34. R. Graham (1966), Bounds for Certain Multiprocessing Anomalies. In *Bell System Technical Journal*, number 45, pages 1563–1581. 1966.
35. S. Sahni (1976), Algorithms for Scheduling Independent Tasks. In *Journal of the Association for Computing Machinery*, number 23 (1), pages 116–127. 1976.
36. C. Witt (2005), Worst-Case and Average-Case Approximations by Simple Randomized Search Heuristics. In *Proceedings of STACS*, volume 3404 of LNCS, pages 44–56, 2005.
37. Karaboga, D., An Idea Based On Honey Bee Swarm For Numerical Optimization, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.