Hızlı Ateş Böceği Algoritması
A Fast Firefly Algorithm

Rustu AKAY[1] , Alper BASTURK[2]
[1]Mekatronik Mühendisliği Bölümü, Erciyes Üniversitesi, Kayseri, Türkiye
akay@erciyes.edu.tr
[2]Bilgisayar Mühendisliği Bölümü, Erciyes Üniversitesi, Kayseri, Türkiye
ab@erciyes.edu.tr

***Özetçe*—Bu çalışmada, yeni bir optimizasyon algoritması olan ateş böceği algoritmasını hızlandırmak için paralel hesaplama sistemlerinin getirdiği avantajlardan faydalanılmıştır. Gerçekleştirilen modelde popülasyon alt popülasyonlara bölünmüş ve her bir alt popülasyonun farklı bir işlemcide çalışması sağlanmıştır. Yaygın olarak kullanılan bazı test fonksiyonları üzerinde elde edilen sonuçlardan, gerçekleştirilen modelin algoritmanın performansını etkilemeden hızını belirgin bir şekilde arttırdığı görülmüştür.**

Anahtar Kelimeler — Ateş böceği algoritması; paralel hesaplama; MPI.

***Abstract*— In this study, the advantages of the parallel compution paradigms are utilized in a recent optimization algorithm, firefly algorithm. In the proposed implementation, the population is divided into subpopulations and each subpopulation is run on a different processing node. From the results on commonly used benchmark functions, the proposed model enhances the computation cost without comprosing on the solution quality.**

Keywords — Global optimization, Firefly algorithm; parallel computing; MPI.

I. GİRİŞ

Ateş böceği (Firefly, FF) algoritması 2008 yılında Yang tarafından geliştirilen yeni bir optimizasyon algoritmasıdır [1,2]. Farklı tipteki optimizasyon problemlerine çözümler sunmayı amaçlayan bu algoritma ateş böceklerinin doğal yaşamlarından esinlenmektedir. Algoritmanın basit ve etkili olması sebebiyle birçok problemin çözümünde kullanılmaktadır [3]. Yapılan bazı çalışmalarda algoritmanın performansını iyileştirmek için bazı modifikasyonlar ve diğer optimizasyon algoritmaları ile beraber çalışmasını sağlayan hybrid modeller geliştirilmiştir [4,5,6]. Ayrıca algoritmanın gerçek hayattaki uygulama problemleri çözümlerinde kullanılmasına yönelik bazı çalışmalarda bulunmaktadır. Mohanty'in ekonomi alanındaki toplam yıllık maliyet kestirimi problemi [7], Erdal'ın yeni nesil çelik kirişleri için minimum ağırlık tasarım problemi [8], Upadhyay ve arkadaşlarının IIR filtre tasarımı [9] ve Setiadi ve Jones'un güç sistemi tasarımı [10] bunlardan bazılarıdır. Bu çalışmalarla ateş böceği algoritmasının optimizasyon problemlerinde kullanılabilecek basit ve etkili bir algoritma olduğu gösterilmiştir. Ateş böceği algoritmasının çalışma zamanı incelendiğinde diğer algoritmalara nazaran daha uzun sürdüğü görülmektedir. Bu çalışmada ateş böceği algoritmasının farklı popülasyon büyüklüklerinde performansı ve çalışma zamanı analiz edilecektir. Ayrıca algoritmanın çalışma zamanındaki dezavantajı gidermek için popülasyon tabanlı optimizasyon algoritmalarında yaygın olarak kullanılan alt popülasyon paralel modelinin gerçekleştirilmesi amaçlanmıştır.

Bu amaçlar doğrultusunda Bölüm 2'de ateş böceği algoritması, Bölüm 3'de gerçekleştirimi yapılan paralel model, Bölüm 4'te test fonksiyonları çözümleri ile elde edilen simülasyon sonuçları verilmiştir. Bölüm-5'de edilen sonuçlar yorumlanmıştır.

II. ATEŞ BÖCEĞİ ALGORİTMASI

Ateş böceği algoritması, ateş böceklerinin sosyal davranışlarından esinlenerek Yang tarafından geliştirilmiş bir optimizasyon algoritmasıdır [1]. Bu algoritmanın diğer popülasyon tabanlı algoritmalarla bir çok ortak özelliği bulunmaktadır. Algoritmada ateş böceği sürüsü popülasyon olarak düşünülmektedir. Algoritma ateş böceklerinin ışıklarını yakıp söndürmesi ve bu sayede diğer ateş böceklerini kendisine çekmesi prensibine dayanmaktadır.

Algoritmada verilen bir optimizasyon probleminin çözümünde, popülasyonun parlak ve daha çekici yerlere gitmesi çözüm kalitesi olarak tanımlanan yanıp sönen ışıklar ya da ışık şiddeti ile ilişkilidir. Popülasyondaki tüm bireyler çözüm kaliteleri ve mesafeleri oranında birbirlerini çekmektedir. Yani ateş böceği ne kadar parlak olursa çözüm kalitesi o kadar iyidir ve diğer ateş böcekleri için o kadar çekicidir. Bununla beraber mesafe, parlaklığı azalttığı için çekim kuvvetini de azaltmaktadır. FF algoritmasının temel adımları Şekil 1 ile gösterilmiştir.

Başlangıç popülasyonunun oluşturulması
Uygunluk değerinin hesaplanması (ışık şiddeti)
Işığın emilim katsayısının belirlenmesi (γ)
Repeat
Her bir adayın kendinden daha iyi çözümler tarafından etkilendiği yeni çözüm üretilir;
Uygunluk değerinin hesaplanması
Güncelleme;
Popülasyon sıralaması ve en iyi çözümün belirlenmesi;
Until

Şekil 1. FF algoritmasının temel adımları

Bu çalışma ASYU-2017 Konferansında sunulmuş ve sadece özet olarak
Bildiri Özetleri Kitabı (ISBN: 978-605-84722-8-0)'nda yayımlanmıştır.

FF algoritmasında iki önemli nokta vardır. Bunlar ışık yoğunluğunun değişimi ve popülasyondaki bireylerin çekicilik formülüdür. Algoritmada ateş böceğinin çekiciliğinin parlaklığı tarafından belirlendiği varsayılır. Buda popülasyondaki bireyin amaç fonksiyon değeri ile ilişkilidir.

Bir optimizasyon problemi için parlaklık $I(x) \infty f(x)$ şeklinde gösterilebilir. Bunun yanında çekicilik değeri (β) görecelidir ve ateş böceği i ile ateş böceği j arasındaki mesafe (r_{ij}) değerine bağlı olarak değişecektir. Dolayısı ile ışık kaynağından uzaklaştıkça ışık emilerek şiddeti azalacaktır. En basit haliyle ışık şiddeti mesafenin karesine bağlı olarak Denklem (1)'deki gibi formülüze edilebilir.

$$I(r) = \frac{I_s}{r^2} \quad (1)$$

Burada r mesafe, I_s de ışık şiddetidir. Herhangi iki ateş böceği arasındaki mesafe Denklem (2) ile yeni çözüm üretme işlemi de Denklem (3) ile hesaplanır.

$$r_{ik} = \sqrt{\sum_{j=1}^{D}(x_{i,j} - x_{k,j})^2} \quad (2)$$

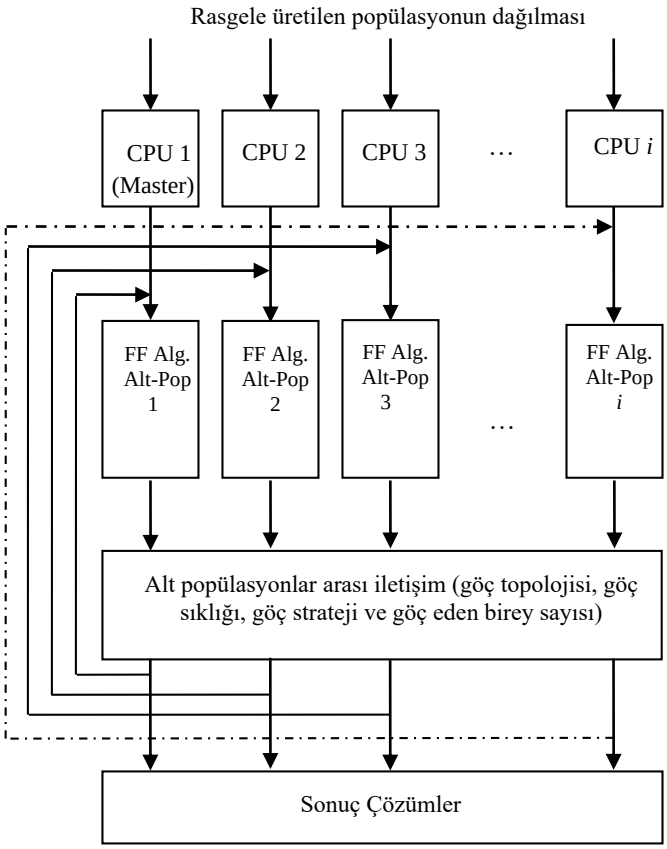
$$x_i(t+1) = x_i(t) + \beta_0 e^{-\gamma r_{ik}^2}\big(x_k(t) - x_i(t)\big) + \alpha\varepsilon_i \quad (3)$$

Burada $k \in [1, NP]$ aralığında $k \neq i$ olmak şartıyla komşu bir çözüm vektörü, α adım büyüklüğü, ε_i Gauss veya uniform dağılımlı bir vektör, γ ateş böcekleri arasında çekicilik değişimlerini belirleyen bir sabit, r ateş böcekleri arasındaki mesafe ve β_0'da onların çekiciliğini göstermektedir.

III. PARALEL ATEŞ BÖCEĞİ ALGORİTMASI

Çalışmada FF algoritmasının paralel gerçekleştirimi (Paralel Firefily, PFF) yapılmıştır. Algoritmanın paralel gerçekleştirimi yapılırken algoritma alt popülasyonlara bölünmüş ve bu alt popülasyonların bazı kriterlere göre birbirleri ile haberleşmesi sağlanmıştır. Bu model genelde popülasyon tabanlı algoritmaların paralelleştirilmesinde kullanılan alt popülasyon (subpopulation) modeli olarak bilinir ve kolay uyarlanabilir olması sebebi ile de yaygın olarak kullanılır [11]. Modelde eş zamanlı olarak her bir alt popülasyonun farklı bir işlemcide çalışması sağlandığı için zamandan kazanç sağlanmaktadır. Ancak bu alt popülasyonlar arası iletişim de, göç topolojisi, göç sıklığı, göç strateji ve göç eden birey sayısı gibi farklı parametreleri beraberinde getirmektedir. Bu parametreler algoritmaların çözüm kalitesini ve hızını etkilemekte ve algoritmalar için ayrı bir araştırma konusu oluşturmaktadır. Bu çalışmada bu parametre değerlerinin etkisi PFF algoritması üzerinde detaylıca araştırılmamış, bunun yerine bilinen, kolayca uyarlanabilen ve yapılan çeşitli pratik çalışmaların sunduğu yaklaşık değerler seçilmiştir. Şekil 2'de gerçekleştirilen modelin blok diyagramı verilmiştir.

Şekil 2'den görüldüğü gibi, gerçekleştirilen PFF algoritmasında paralel çalışan yerel FF algoritmaları farklı başlangıç popülasyonları üzerinde çözüm uzayını araştırmakta ve belirli bir işlem sayısı sonunda çözümler diğer çözümlerle birleştirilerek ve/veya karşılaştırılarak diğer FF algoritmalarına aktarılmaktadır. Bu aktarım sayesinde bilginin yerel algoritmalar arası değişimi gerçekleşmiş olur. Bu işlemler algoritmanın çalışma süresince tekrarlanır.



Şekil 2. PFF algoritmasının blok şeması

IV. SİMÜLASYON SONUÇLARI

Gerçekleştirilen model C programlama dilinde MPI paralel programlama kütüphanesi kullanılarak kodlanmıştır [12]. Problem çözümleri TÜBİTAK ULAKBİM, Yüksek Başarım ve Grid Hesaplama Merkezindeki bilgisayar sistemlerinde gerçekleştirilmiştir.

FF ve PFF algoritmalarının performansı ve çalışma zamanının incelenmesi için literatürde yaygın olarak kullanılan 12 adet sürekli test fonksiyonu dikkate alınmıştır. Tablo 1'de verilen bu fonksiyonlardan her biri farklı zorluk derecesine ve karakteristiğe sahiptir. Elde edilen sonuçların güvenilirliği açısından problemler 30'ar kez çözülmüş ve sonuç tablolarında bu çözümlerin ortalama değerleri verilmiştir. Algoritmanın hızlanma etkisini görebilmek için de problem boyutları özellikle büyük seçilmiştir. Problem boyutu, beta, alfa ve gama katsayısı sırası ile 100, 0.2, 0.25 ve 1 alınmış, maksimum fonksiyon hesaplama sayısı ise 500.000 seçilmiştir.

Çalışmada ilk olarak popülasyon tabanlı algoritmaların ortak parametresi olan popülasyon büyüklüğü değerinin FF algoritmasına etkisi algoritmanın seri modeli üzerinde incelenmiştir. Popülasyon büyüklüğünün 12, 24, 48 ve 96 değerleri için elde edilen performans ve çalışma zamanı değerleri Tablo 2 ve Tablo 3'de sırası ile verilmiştir.

Bu çalışma ASYU-2017 Konferansında sunulmuş ve sadece özet olarak
Bildiri Özetleri Kitabı (ISBN: 978-605-84722-8-0)'nda yayımlanmıştır.

	Fonksiyon	Aralık	Özellikler
f_1	Shifted Sphere	[-100, 100]	TM, AY
f_2	Shifted Schwefel	[-100, 100]	TM, AT
f_3	Shifted Rosenbrock	[-100, 100]	CM, AT
f_4	Shifted Rastrigin	[-5,5]	CM, AY
f_5	Shifted Griewank	[-600,600]	CM, AT
f_6	Shifted Ackley	[-32,32]	CM, AY
f_7	Step	[-100, 100]	TM, AY
f_8	Quartic	[-1.28-1.28]	TM, AY
f_9	Schwefel 2.22	[-10, 10]	TM, AT
f_{10}	Dixon Price	[-10, 10]	TM, AT
f_{11}	Penalized	[-50, 50]	CM, AT
f_{12}	Penalized2	[-50, 50]	CM, AT

Tablo 1. Test fonksiyonları (TM: Tek modlu, CM: Çok modlu, AY: Ayrıştırılabilir, AT: Ayrıştırılamaz)

	Popülasyon büyüklüğü			
	12	24	48	96
f_1	**1.32E-03**	1.75E-03	2.75E-03	4.87E-03
f_2	1.40E-01	1.28E-01	1.18E-01	**9.98E-02**
f_3	2.99E+03	2.76E+03	**2.69E+03**	3.17E+03
f_4	2.72E+02	2.06E+02	1.70E+02	**1.46E+02**
f_5	2.58E-03	**2.25E-03**	2.68E-03	3.22E-03
f_6	**4.69E-03**	5.50E-03	6.55E-03	9.06E-03
f_7	8.67E-01	6.67E-02	**0.00E+00**	**0.00E+00**
f_8	9.81E-02	2.98E-02	9.27E-03	**4.55E-03**
f_9	**6.12E-02**	1.05E-01	1.49E-01	2.73E-01
f_{10}	**6.15E-01**	1.12E+00	1.58E+00	6.22E+00
f_{11}	**9.86E-07**	1.04E-03	2.09E-06	4.53E-06
f_{12}	1.88E-03	8.01E-04	**1.06E-04**	2.32E-04

Tablo 2. FF algoritması çözüm değerleri

	Popülasyon büyüklüğü			
	12	24	48	96
f_1	15.77	30.64	62.84	118.96
f_2	15.76	30.65	60.68	118.91
f_3	19.81	34.76	66.35	122.99
f_4	20.41	35.42	65.39	123.56
f_5	26.91	41.97	71.82	129.87
f_6	19.82	34.75	64.74	122.98
f_7	10.01	16.60	29.71	55.15
f_8	23.54	38.67	68.74	126.88
f_9	15.67	30.58	60.72	118.86
f_{10}	18.03	33.15	63.48	121.27
f_{11}	24.23	39.26	69.08	127.10
f_{12}	24.00	38.81	68.69	126.84
Ortalama	19.50	33.77	62.69	117.78

Tablo 3. FF algoritması çalışma süreleri (sn)

Sonuçlar incelendiğinde popülasyon büyüklüğünün FF algoritmasının çalışma süresini son derece etkilediği görülmektedir. Popülasyonun büyümesi algoritmanın çalışma zamanını arttırmaktadır. İncelenen tüm test fonksiyonlarının ortalama çalışma zamanları dikkate alındığında algoritmanın popülasyon büyüklüğü 12 olduğundaki çalışma zamanının popülasyon büyüklüğü 96 olduğundakine oranla 6 kat daha hızlı olduğu görülmektedir. Çalışma süresinin bu denli farklı olması algoritmada her bir yeni aday çözümünün üretilmesinde popülasyondaki kendisinden daha iyi diğer tüm çözümlerden faydalanılma prensibinden kaynaklanmaktadır. Popülasyon büyüdükçe iyileştirilecek her bir çözümden daha fazla sayıda iyi bireyler bulunmakta buda çalışma süresini arttırmaktadır. Buna karşın popülasyon büyüklüğündeki değişim çözüm kalitesi bakımından incelendiğinde düşük yada yüksek popülasyon sayısının etkisi genelleştirilememiştir. f_1, f_9 ve f_{11} gibi bazı test fonksiyonlarında düşük popülasyon sayısı ile daha kaliteli çözümler elde edilirken, f_2, f_7 ve f_8 gibi bazı test fonksiyonlarında yüksek popülasyon sayısı ile daha kaliteli çözümler elde edilmiştir. Bu test fonksiyonları için etkin bir çözüm elde etmek için kabul edilebilir sürelerin üzerinde çözüm sürelerine ihtiyaç duyulduğu gözlemlenmiştir. Yüksek popülasyon sayılarında oldukça yavaş çalışan FF algoritması için bu bir dezavantaj oluşturmaktadır.

Çalışmanın ikinci bölümünde FF algoritmasının alt popülasyon paralel gerçekleştirimi incelenmiştir. Bu modelin gerçekleştirimi algoritmaların özel parametrelerine ek yeni kontrol parametreleri getirdiği bilinmektedir. Bu parametrelerden göç topolojisi halka, göç sıklığı 10, göç eden birey sayısı 1 ve göç stratejisi olarak en iyi bireyin göç ederek en kötü bireyle yer değiştirdiği strateji seçilmiştir. Gerçekleştirilen paralel modelde işlemci sayısı alt popülasyon sayısına eşittir. İşlemci sayısının etkisi 2, 4, 8 ve 16 işlemci için incelenmiştir. Paralel modelin performans ve çalışma zamanı değerleri Tablo 4 ve Tablo 5'de sırası ile verilmiştir.

Paralel algoritmaların performanslarını ölçmek için kullanılan çeşitli yöntemler ve ölçümler bulunmaktadır. Hızlanma ve verimlilik bu yöntem ve ölçümlerin en önemlileridir. Hızlanma, problem çözümünün tek bir işlemci ile gerçekleştirildiğinde harcanan zamanın paralel hesaplama sistemleri ile gerçekleştirildiğinde harcanan zamana oranı olarak tanımlanır [13]. Verimlilik ise ne kadar işlemci ile ne kadar hızlanma sağlandığının ölçüsüdür [13]. Bu çalışmada hızlanma değerleri incelenmiştir ve Şekil 3'de PFF algoritması için ilk 4 fonksiyon üzerinde hızlanma grafikleri verilmiştir.

	İşlemci sayısı			
	2	4	8	16
f_1	3.76E-03	3.53E-03	3.45E-03	3.23E-03
f_2	9.92E-02	1.26E-01	1.39E+00	2.16E+01
f_3	5.06E+02	9.97E+01	9.76E+01	6.94E+01
f_4	1.43E+02	1.70E+02	2.31E+02	3.63E+02
f_5	2.50E-03	2.03E-03	1.71E-03	1.54E-03
f_6	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_7	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_8	4.41E-03	4.93E-03	8.99E-03	1.25E-01
f_9	1.84E-01	1.98E-01	2.05E-01	2.71E-01
f_{10}	1.16E+00	7.07E-01	7.01E-01	7.21E-01
f_{11}	3.54E-06	3.35E-06	3.16E-06	3.24E+00
f_{12}	1.90E-04	1.69E-04	1.58E-04	4.81E-04

Tablo 4. PFF algoritması çözüm değerleri

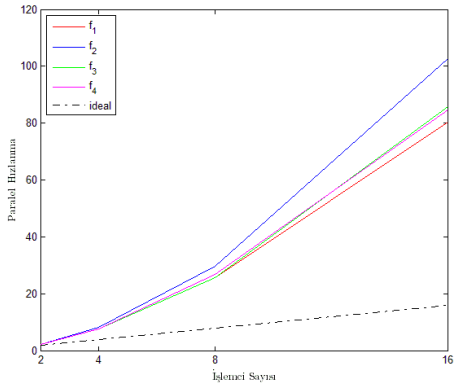
Bu çalışma ASYU-2017 Konferansında sunulmuş ve sadece özet olarak
Bildiri Özetleri Kitabı (ISBN: 978-605-84722-8-0)'nda yayımlanmıştır.

	İşlemci sayısı				
	FF	PFF			
	1	2	4	8	16
f_1	118.96	55.76	14.16	3.64	0.91
f_2	118.91	55.84	14.24	3.64	0.90
f_3	122.99	57.96	15.16	4.15	1.17
f_4	123.56	58.45	15.29	4.21	1.21
f_5	129.87	61.60	16.92	5.04	1.62
f_6	122.98	58.13	15.17	4.14	1.20
f_7	55.15	26.46	7.34	2.14	0.64
f_8	126.88	60.10	16.48	4.74	1.50
f_9	118.86	55.90	14.17	3.60	0.90
f_{10}	121.27	57.09	14.79	3.92	1.05
f_{11}	127.10	60.27	16.28	4.74	1.53
f_{12}	126.84	60.12	16.18	4.71	1.50
Ortalama	117.78	55.64	14.68	4.06	1.18

Tablo 5. PFF algoritması çalışma süreleri (sn)

Tablo 4'deki sonuçlar incelendiğinde alt popülasyon sayısının algoritmanın çözüm kalitesine etkisinin sınırlı olduğu görülmüştür. Diğer taraftan Tablo 5'daki çalışma zamanları ve Şekil 3'deki hızlanma grafikler incelendiğinde paralel gerçekleştirim ile FF algoritmasının en büyük dezavantajının giderilerek zamansal kazanç sağladığı görülmektedir.

İncelenen tüm test fonksiyonlarının ortalama çalışma zamanları dikkate alındığında 16 işlemci ile çalışan PFF algoritmasının FF algoritmasına göre yaklaşık 99 kat daha hızlı sonuç verdiği görülmektedir. Bu kadar yüksek zamansal kazanç problem boyutunun büyüklüğünden de kaynaklanmaktadır. Problem boyutlarını daha da büyüttüğümüzde (değişken sayısını 100 yerine 1000 aldığımıza) elde edilen bu kazancın daha fazla olacağı söylenebilir. Özetle bu çalışmada PFF algoritması ile doğrusal hızlanmanın da üzerinde süper doğrusal hızlanma elde edilmiştir.



Şekil 3. PFF algoritması için hızlanma grafiği

V. SONUÇLAR

Çalışmanın ilk bölümünde seri FF algoritmasının farklı popülasyon büyüklüklerindeki çalışma zamanı ve performansı diğer bölümünde ise paralel gerçekleştirimi yapılan algoritmanın hızlanma değerleri incelenmiştir.

FF algoritmasında popülasyon büyüdükçe çalışma zamanının belirgin bir şekilde arttığı gözlemlenmiştir. Özellikle yüksek popülasyon sayılarında daha etkili çözümlerin bulunduğu test problemlerinde çözüm süreleri kabul edilebilir süreleri aşmaktadır. Algoritma için dezavantaj oluşturan bu durum algoritmanın paralel gerçekleştirimi ile giderilmiştir. Popülasyonun alt popülasyonlara bölünmesi ve her bir alt popülasyonunda farklı işlemcilerde çalışması ile gerçekleştirilen model, algoritmanın hem düşük popülasyonlar ile çalışması sağlanmış hem de paralel hesaplama sistemlerinin getirdiği avantajlardan faydalanarak çok yüksek bir oranda hızlanma elde edilmiştir.

Algoritmanın yeni çözüm üretme mekanizmasına farklı stratejilerin entegre edilmesi gelecekte yapılabilecek çalışmalar olarak düşünülebilir.

KAYNAKLAR

[1] *Yang, X.S., Nature-Inspired Metaheuristic Algorithms, Luniver Press, UK, 2008.*

[2] *Yang, X.S., "Firefly algorithms for multimodal optimization", in: Stochastic Algorithms: Foundations and Appplications (Eds O. Watanabe and T. Zeugmann), SAGA 2009, Lecture Notes in Computer Science, 5792, Springer-Verlag, Berlin, 2009, p 3-6.*

[3] *Tilahun, S.L. ve Ngnotchouyei J.M.T., Firefly Algorithm for Discrete Optimization Problems: A Survey, Journal of Civil Engineering, Vol. 21, No. 2, 2017, p 535-545.*

[4] *Rahmani, A. and MirHassani, S., "A hybrid firefly-genetic algorithm for the capacitated facility location problem." Information Sciences, Vol. 283, 2014, p 70-78.*

[5] *M., Zhang, Y., B. Zeng, H. Z., and Liu, J,. "The modified firefly algorithm considering fireflies' visual range and its application in assembly sequences planning." Int J Adv Manuf Technol, Vol. 82, No. 5, 2016, p 1381-1403.*

[6] *Huang, S.-J., Liu, X.-Z., Su, W.-F., and Yang, S.-H., "Application of hybrid firefly algorithm for sheath loss reduction of underground transmission systems." IEEE Transactions on Power Delivery, Vol. 28, No. 4, 2013, p 2085-2092.*

[7] *Mohanty, D.K.,. "Application of firefly algorithm for design optimization of a shell and tube heat exchanger from economic point of view." International Journal of Thermal Sciences, Vol. 102, 2016, pp. 228-238.*

[8] *Erdal, F., "A firefly algorithm for optimum design of newgeneration beams." Engineering Optimization, 2016, pp. 1-17.*

[9] *Upadhyay, P., Kar, R., Mandal, D., and Ghoshal, S., "A new design method based on firefly algorithm for IIR system identification problem." Journal of King Saud University-Engineering Sciences, Vol. 28, No. 2, 2016, p 174-198.*

[10] *Setiadi, H. and Jones, K. O., "Power system design using firefly algorithm for dynamic stability enhancement." Indonesian Journal of Electrical Engineering and Computer Science, Vol. 1, 2016, p 446-455.*

[11] *Alba E., Parallel Metaheuristics: A New Class of Algorithms, Wiley-Interscience, 2005.*

[12] *Pacheco ,P.S., Parallel programming with MPI, Morgan Kaufmann Publishers USA,, 1996.*

[13] *Grama, A., Gupta, A., Karypis, G. and Kumar, V., Introduction to Parallel Computin, Addison Wesley, Edinburg, 2003.*

Bu çalışma ASYU-2017 Konferansında sunulmuş ve sadece özet olarak
Bildiri Özetleri Kitabı (ISBN: 978-605-84722-8-0)'nda yayımlanmıştır.