Journal of the Faculty of Engineering and Architecture of Gazi University 35:2 (2020) 753-763

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University

Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

# Petri net-based algorithm for maximizing production rate in assembly lines

Özcan Kılınçcı
Department of Industrial Engineering, Dokuz Eylul University, Izmir, 35390, Turkey

**Highlights:**

- A Petri net-based algorithm is presented to solve SALBP-2.
- Firing sequence obtained is used as a priority rule.
- According to comparison results, the presented algorithm is efficient for solving SALBP-2.

**Keywords:**

- SALBP-2
- Assembly line balancing
- Petri net

**Article Info:**

Research Article
Received: 18.01.2019
Accepted: 23.09.2019

**DOI:**

10.17341/gazimmfd.514759

**Correspondence:**

Author: Özcan Kılınçcı
e-mail: ozcan.kilincci@deu.edu.tr
phone: 90 232 301 76 12

**Graphical/Tabular Abstract**

In this study, a new algorithm based on Petri net is presented for simple assembly line balancing problem type-2 (SALBP-2). The presented heuristic algorithm obtains a task order using the properties of Petri net, i.e. firing rule, token movement, and token condition. Tasks are assigned to the workstations using the backward procedure and the task order as a priority rule. The heuristic is a two-stage algorithm; a feasible solution is found in the first stage, and the feasible solution is improved using binary search procedure in the second stage.

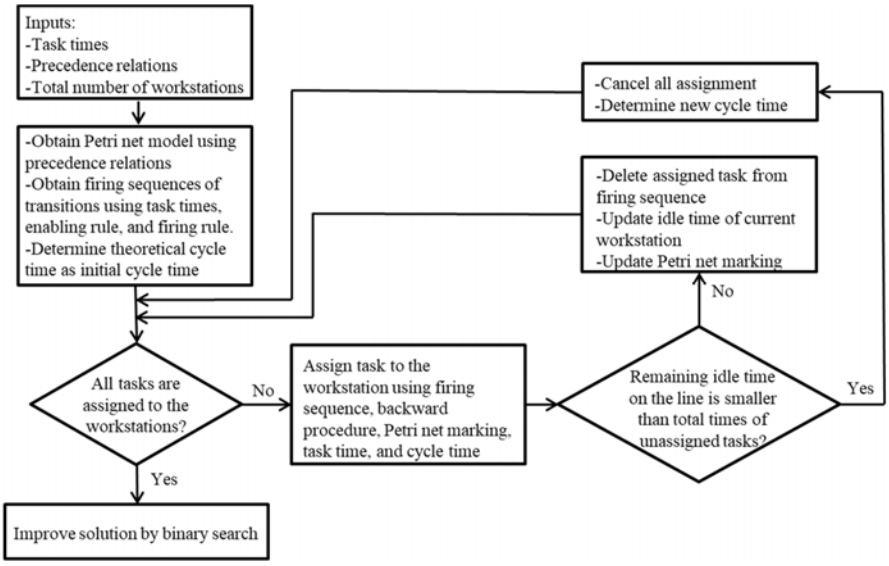



**Figure A.** Flow chart of the presented algorithm.

**Purpose:** The purpose of the study is to develop a new Petri-net based algorithm for solving SALBP-2.

**Theory and Methods:**
The presented algorithm is based on Petri nets which are mathematical and graphical tools used for modeling, formal analysis, and design of discrete event systems. Precedence relations in assembly line balancing problem are represented as a Petri net model. A task order is obtained using enabling rule and firing rule. Tasks are assigned to the workstations using this order.

**Results:**
The presented algorithm is coded in MATLAB and tested on well-known benchmark data set with 302 instances. The comparison studies are given between the presented heuristic algorithm and well-known priority rules, other Petri net-based algorithms, differential evolution algorithms, and genetic algorithms in the literature.

**Conclusion:**
The presented heuristic has the superior performance, especially in large assembly lines. Results show that the developed Petri net-based algorithm is efficient for solving SALBP-2.

Journal of the Faculty of Engineering and Architecture of Gazi University 35:2 (2020) 753-763

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University

Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

# Montaj hatlarında üretim oranını en büyüklemek için petri ağı tabanlı bir algoritma

Özcan Kılınçcı*
Dokuz Eylül Üniversitesi, Endüstri Mühendisliği Bölümü, Tınaztepe Yerleşkesi, Buca İzmir, 35390, Türkiye

## Ö N E Ç I K A N L A R

- BMHDP-2'yi çözmek için Petri ağı tabanlı bir algoritma önerildi
- Oluşturulan tetikleme sırası, bir öncelik kuralı gibi kullanıldı
- Karşılaştırma sonuçlarına göre önerilen algoritma BMHDP-2 çözümünde etkindir

Makale Bilgileri
Araştırma Makalesi
Geliş: 18.01.2019
Kabul: 23.09.2019

DOI:
10.17341/gazimmfd.514759

Anahtar Kelimeler:
BMHDP-2,
montaj hattı dengeleme,
petri ağı

ÖZET

Bu çalışmada, basit montaj hattı dengeleme probleminin ikinci tipi (BMHDP-2) için Petri ağı tabanlı yeni bir sezgisel önerilmiştir. Önerilen sezgisel, Petri ağının tetikleme kuralı, işaret hareketi ve işaret dağılımı gibi özelliklerini kullanarak bir görev sırası oluşturur. Bu görev sırası bir öncelik kuralı gibi kullanılarak, görevler iş istasyonlarına geriye doğru yöntemi ile atanır. Sezgisel iki aşamalı bir algoritmadır; ilk aşamada olurlu bir çözüm bulunur, ikinci aşamada da bulunan bu olurlu çözüm ikili arama prosedürü ile iyileştirilir. Önerilen sezgisel algoritma ile literatürde bilinen öncelik kuralları, diğer Petri ağı tabanlı algoritmalar, diferansiyel evrim algoritmaları ve genetik algoritmaları arasında karşılaştırmalar sunulmuştur. Sonuçlar, sunulan Petri ağı tabanlı algoritmanın BMHDP-2 çözümü için etkin olduğunu göstermektedir.

# Petri net-based algorithm for maximizing production rate in assembly lines

## H I G H L I G H T S

- A Petri net-based algorithm is presented to solve SALBP-2
- Firing order obtained is used as a priority rule.
- According to comporation results, the presented algorithm is efficient for solving SALBP-2

Article Info
Research Article
Received: 18.01.2019
Accepted: 23.09.2019

DOI:
10.17341/gazimmfd.514759

Keywords:
SALBP-2,
assembly line balancing,
petri net

ABSTRACT

In this study, a new simple heuristic based on Petri net is presented for simple assembly line balancing problem type-2 (SALBP-2). The presented heuristic obtains a task order using the properties of Petri net, i.e. firing rule, token movement, and token condition. Tasks are assigned to the workstations using the backward procedure and the task order as a priority rule. The heuristic is a two-stage algorithm; a feasible solution is found in the first stage, and the feasible solution is improved using binary search procedure in the second stage. The comparison studies are presented between the presented heuristic algorithm and well-known priority rules, other Petri net-based algorithms, differential evolution algorithms, and genetic algorithms in the literature. The results show that the presented Petri net-based algorithm is efficient for solving SALBP-2.

*Sorumlu Yazar/Corresponding Author: ozcan.kilincci@deu.edu.tr / Tel: +90 232 301 76 12

## 1. GİRİŞ (INTRODUCTION)

Günümüz rekabet ortamında işletmeler daha az maliyetle üretim yapabilmek amacındadır. Büyük miktarlarda yapılan üretimler ortalama birim üretim maliyetini düşürdüğünden, yığın üretim oldukça tercih edilen bir üretim sistemidir. Yığın üretimin en çok bilinen çeşidi montaj hattı üretimidir. Montaj hattı üretim sistemi, sıralı iş istasyonları ve iş istasyonları arasındaki taşımalar için bir malzeme aktarma sisteminden oluşur. Her iş istasyonunda ürüne ait belirli görevler (operasyonlar) gerçekleştirilir. Üretim sırasında ürünle ilgili teknik ve fiziki kısıtlardan dolayı bazı görevlerin yapılabilmesi için öncesinde başka görevlerin tamamlanması gerekir. İş istasyonlarına görevler atanırken bu öncelik ilişkileri dikkate alınır. Ayrıca her bir iş istasyonunda yapılan görevlerin aynı sürede tamamlanması hedeflenir. Çevrim süresi denilen bu süre sonrasında, belirli görevleri tamamlanmış ürün bir sonraki iş istasyonuna aktarılır. Sonuç olarak, etkin bir montaj hattı üretimi gerçekleştirebilmek için görevlerin hem öncelik ilişkileri hem de süreler dikkate alınarak hat üzerindeki iş istasyonlarına atanması problemine literatürde montaj hattı dengeleme problemi (MHDP) denir. MHDP için son yıllarda yapılan literatür araştırması çalışmaları [1-6]'da bulanabilir.

MHDP, çözüm amacına, montaj hattında üretilen ürün çeşidine, görev sürelerinin özelliklerine ve montaj hattı üretim sisteminin özelliklerine göre farklı sınıflara ayrılmaktadır [1-6]. Bu çalışmada basit montaj hattı dengeleme problemi (BMHDP) dikkate alınmıştır. BMHDP'nin temel özellikleri; hat üzerinde bir ürün çeşidinin üretilmesi, ürüne ait görev sürelerinin sabit ve biliniyor olması, görevlere ait herhangi bir atama kısıtının olmaması, hat üzerindeki tüm iş istasyonlarının aynı yeterliliklere sahip olması olarak belirtilir [1]. Battaia ve Dolgui [5] BMHDP'de kullanılan çözüm tekniklerini üç sınıfa ayırır; kesin çözüm veren algoritmalar, meta sezgiseller ve kurucu sezgiseller. Kesin çözüm veren algoritmalar, dinamik programlama yöntemi ve dal sınır algoritması kullanılarak oluşturulan yöntemleridir [1, 2]. BMHDP NP-zor problem olduğundan, problemdeki görev sayısı arttığında kesin çözüm veren algoritmalar ile çözüm bulunamaz. Bu nedenle optimum ya da optimuma yakın çözümler bulabilmek için kurucu sezgiseller veya meta sezgiseller kullanılır.

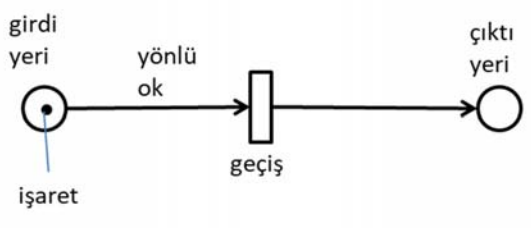
BMHDP amaç fonksiyonuna göre üç farklı şekilde sınıflandırılır [1]. Birinci durumda (BMHDP-1) amaç, problemde verilen çevrim süresine uygun en küçük toplam iş istasyonu sayısını bulmaktır. İkinci durumda (BMHDP-2), verilen iş istasyonu sayısına karşılık gelen en küçük çevrim süresini bulmak hedeflenir. Son durumda da (BMHDP-E), çevrim süresi ve toplam iş istasyonu sayısında değişlikler yapılarak hat etkinliğinin en büyük olması amaçlanır. Bu çalışmada, BMHDP-2 problemi dikkate alınmıştır. Bu nedenle, literatürde sadece BMHDP-2 ile ilgili yapılan çalışmalara yer verilecektir. Kim vd. [7] BMHDP-2 ve diğer BMHDP durumları için bir genetik algoritma kullanmışlardır. Klein ve Scholl [8] BMHDP-2'yi çözmek için dal sınır algoritmasına ve yerel alt sınır metoduna dayalı SALOME-2 yöntemini geliştirmişlerdir. Scholl ve Voß [9] hem BMHDP-1 hem de BMHDP-2 için bir tabu arama prosedürü önermiştir. Uğurdağ vd. [10] problemi çözmek için tam sayılı programlamaya dayalı iki aşamalı bir sezgisel geliştirmişlerdir. Nearchou [11] iki farklı diferansiyel evrim algoritması önermiştir. Kilincci [12] Petri ağlarının erişebilirlik özelliğini (reachability analysis) kullanarak Petri ağı tabanlı bir sezgisel geliştirmiştir. Blum [13] kesin sonuç veren ışın arama tabanlı bir yöntem önermiştir. Zheng vd. [14] BMHDP-2 çözümü için karınca kolonisi optimizasyonu kullanmıştır. Zhang vd. [15] tam sayı kodlamalı diferansiyel evrim algoritması önermiştir. Arıkan [16] iş yükü dengelemelerini de dikkate alarak bir tabu arama algoritması geliştirmiştir. Literatürde, BMHDP-2 ile ilgili çalışmalar BMHDP-1'e göre oldukça azdır. Bu çalışmanın temel amacı, sınırlı sayıda olan BMHDP-2 literatürüne etkin bir kurucu sezgisel kazandırmaktır. Literatürde BMHDP-2 için önerilen yöntemler iki temel mantık içerir; çözümü doğrudan bulmak ya da problemi BMHDP-1'e uyarlayarak farklı çevrim süreleri için deneme yapıp verilen iş istasyonu sayısını sağlayan çevrim süresini belirlemeye çalışmak [1]. Bu çalışmada, Kilincci'nin [17] BMHDP-1 için önerdiği geriye doğru tetikleme sırası (firing sequence backward, FSb) algoritması BMHDP-2'ye uyarlanacaktır. FSb algoritması BMHDP-1 literatüründe en etkin öncelik kuralları arasında gösterilmiştir [18, 19]. BMHDP-2 için uyarlanan FSb algoritması (FSb-2), literatürde var olan ve Petri ağının erişebilirlik özelliğini kullanan algoritmadan [12] farklı olarak, Petri ağının tetikleme özelliğini kullanır. Bu özellik yardımıyla önce atama için kullanılacak bir görev sırası oluşturulur. Sonra geriye doğru prosedür uygulanılarak, sıradaki son görev son iş istasyonuna atanacak şekilde, öncelik ilişkileri ve görev süreleri dikkate alınarak belirli çevrim süreleri için çözümler elde edilir. Daha sonra elde edilen olurlu çözüm ikili arama prosedürü ile iyileştirilmeye çalışılır.

Çalışmanın geriye kalan kısmı şu şekilde planlanmıştır. İkinci bölümde, Petri ağlarının BMHDP'de nasıl kullanıldığı açıklanacaktır. Önerilen FSb-2 algoritması üçüncü bölümde ayrıntılı olarak verilecektir. Çalışma sonuçları dördüncü bölümde özetlenecektir. Bu çalışmalar, önerilen yöntem ile BMHDP-2 için en etkin olarak bilinen öncelik kuralları arasındaki karşılaştırmaları ve yine önerilen yöntem ile literatürdeki diğer Petri ağı tabanlı algoritmalar, bazı genetik algoritmalar ve bazı diferansiyel evrim algoritmaları ile karşılaştırmaları içermektedir. Son bölümde yapılan çalışma değerlendirilecek ve gelecekte yapılabilecek çalışmalar ile ilgili öneriler sunulacaktır.

## 2. MONTAJ HATTI DENGELEME PROBLEMİNDE PETRİ AĞLARI
### (PETRI NETS IN ASSEMBLY LINE BALANCING PROBLEM)

Petri ağları, ayrık olaylı sistemlerin tasarımında, modellenmesinde ve analizinde kullanılan matematiksel ve

grafiksel bir yöntemdir. Bir Petri ağı üç nesneden oluşur; yer (place), geçiş (transition) ve yer ve geçişleri birbirine bağlayan yönlü ok (directed arc) [20, 21]. Okların yönüne göre, geçişten önceki yere girdi yeri, geçişten sonraki yere de çıktı yeri denir. Eğer bir Petri ağında bir geçiş, bir olayı ya da görevi temsil ediyorsa, o geçişe ait girdi yerleri o olay ya da görev öncesinde olması gereken şartları, o geçişe ait çıktı yerleri de o olay ya da görevin gerçeklemesi sonucunda ortaya çıkan durumları gösterir. Petri ağlarının gösteriminde kullanılan bir diğer nesne, yerlerin içinde gösterilen işaretlerdir (token). Basit bir Petri ağı modeli Şekil 1'de verilmiştir.




**Şekil 1.** Bir Petri ağı modeli (A typical Petri net model)

İşaretlerin Petri ağı içindeki hareketi, Petri ağı modellerinin dinamik davranışı olarak tanımlanır. İşaretlerin hareketi iki temel kurala bağlıdır; gerçekleşme (enabling) kuralı ve tetikleme (firing) kuralı. Bir geçişin gerçekleşmesi için geçişe ait her girdi yerinin en az o girdi yeri ile geçişi birbirine bağlayan okun üzerindeki ağırlık değeri kadar işaret içermesi gerekir. Bu gerçekleşme kuralıdır. Gerçekleşme kuralını sağlayan geçişin tetiklenmesi ile geçiş ve girdi yerini bağlayan okun üzerindeki ağırlık sayısı kadar işaret girdi yerinden ayrılır, geçiş ve çıktı yerini birbirine bağlayan okun ağırlık sayısı kadar işaret çıktı yerine yerleşir. Bu da tetikleme kuralıdır. [20, 21]

Bu iki kuralın yerine gelmesi ile işaretlerin Petri ağındaki yerlerde hareket etmesine Petri ağı işaretlemesi (Petri net marking) denir. Petri ağı işaretlemesi, modellenen sistemin o anki durumunu gösterir. Bu gösterim 1xm boyutlarındaki M satır vektöründe yapılır. Vektörün kolon sayısını veren m, Petri ağında bulunan toplam yer sayısıdır. M vektörü negatif olmayan tamsayılardan oluşur ve M(p) de p. yerde bulunan işaret sayısını gösterir. Bir Petri ağının dinamik çalışma yapabilmesi için ya da bir başka tanımla işaretlerin ağ içinde hareket edebilmesi için, başlangıç anında ilgili yerlerde yeterli sayıda işaret olması gerekir. Başlangıç anındaki işaret dağılımı M0 ile gösterilir.

Bir Petri ağında gerçekleşen işaret hareketleri kolayca belirlenebilir. Bunun için öncelikle, geçişler ve yerler arasındaki tüm bağlantıları ve bu bağlantıları sağlayan okların ağırlık değerlerinin gösterildiği ilişkiler matrisi (incidence matrix) oluşturulur [20, 21]. İlişkiler matrisi (A), n adet geçişi temsil eden n adet satır ve m adet yeri temsil eden m adet sütundan oluşur. Geçiş ve yer arasında bir bağlantı varsa, ilgili satır ve sütun değerinin oluşturduğu hücreye bu bağlantıyı sağlayan okun üzerindeki ağırlık değeri yazılır. Eğer yer geçişin girdi yeri ise okun ağırlık değeri negatif, eğer yer geçişin çıktı yeri ise okun ağırlık değeri pozitif olur. Oluşturulan A matrisi yardımıyla, herhangi bir konumdaki işaret dağılımı belirlenebilir. Bunun için kullanılan denklem [20, 21],

$$M_k = M_{k\text{-}1} + A_i \quad k=1, 2,\ldots\ i=1,2,\ldots n \tag{1}$$

Eş.1'de, $M_k$ , 1×m boyutlarında satır vektörüdür ve Petri ağındaki k. işaretlemeyi gösterir. Benzer şekilde $M_{k-1}$ de bir önceki yani ($k-1$). işaretlemeyi gösterir. Eş.1'deki i, $M_{k-1}$ işaretlemesine göre geçerli olan ve tetiklenebilecek geçişin numarasıdır. Son olarak $A_i$ de A matrisinin i. satırıdır.

Bu bilgiler ışığında, Petri ağlarının MHDP'de nasıl kullanılabileceği şu şekilde açıklanabilir. MHDP girdilerinden biri görevler arasındaki öncelik ilişkilerinin gösterildiği öncelik diyagramıdır. Bu öncelik diyagramı bir Petri ağı ile modellenebilir. Petri ağında geçişler görevleri, yerler de öncelik ilişkilerini (görevin yapılmasından önce ve görevin yapılmasından sonraki durumları) gösterir. Geçişe ait girdi yerinde yeterli miktarda işaret varsa, bu durum geçişin geçerli olabileceğini (gerçekleşme kuralı) belirtir. Geçerli olan bir geçiş, öncelik ilişkilerini sağlayan ve mevcut istasyona atanabilecek bir görevi temsil eder. Tetiklenen bir geçiş sonrası, ilgili geçişe ait girdi yerindeki işaret geçişe ait çıktı yerine hareket eder (tetikleme kuralı). Tetikleme kuralı ile tetiklenen bir geçiş, temsil ettiği görevin iş istasyonuna atandığını gösterir. Benzer şekilde Petri ağındaki işaret hareketleri ile tüm görevlerin iş istasyonlarına atanması gerçekleştirilebilir. Bunun için M0 başlangıç işaretlemesi ile atama prosedürüne başlayıp, her atama sonrası oluşan yeni işaret dağılımını Eş. 1 ile belirleyerek tüm atamalar tamamlanır. Atamalar tamamlandığında Petri ağındaki işaretlerin dağılımı, son işaret dağılımını göstereceğinden artık Petri ağı modelinde yeni bir işaret hareketi söz konusu olmayacaktır. Bu başlıkta değinilen Petri ağlarının BMHPD'de nasıl kullanabileceği konusunda daha ayrıntılı bilgiler almak isteyen araştırmacılar Kilincci [12, 17], Kilincci ve Bayhan [22, 23] çalışmalarına bakabilir.

## 3. ÖNERİLEN FSB-2 ALGORİTMASI
**(PROPOSED FSB-2 ALGORITHM)**

Önerilen Petri ağı tabanlı FSb-2 algoritması, Kilincci'nin [17] BMHDP-1 için önerdiği geriye doğru tetikleme sırası (firing sequence backward, FSb) algoritmasının BMHDP-2'ye uyarlanmasıdır. FSb-2, iki aşamalı bir algoritmadır. İlk aşamada algoritma, öncelik ilişkilerinden oluşturulan Petri ağı modeli yardımıyla geçişlerin tetikleme sırasını bulur. Bu sıra yardımıyla ve geriye doğru prosedürü kullanılarak BMHDP-2 çözülür. BMHDP-2 çözüm sürecinde, problem BMHDP-1 olarak düşünülüp, teorik çevrim süresinden başlayarak farklı çevrim süreleri için FSb algoritması [17] kullanılarak çözümler elde edilir. BMHDP-2 için verilen iş istasyonu sayısını sağlayan ilk çözüm, olurlu çözüm olarak belirlenir. Böylece ilk aşama tamamlanmış olur. İkinci aşamada, ilk aşamada bulunan olurlu çözüm, ikili arama prosedürü ile iyileştirilir. Algoritma adımları aşağıda verilmiştir.

Adım 1. Probleme ait verileri gir; görev süreleri, öncelik ilişkileri, toplam iş istasyonu sayısı.

Adım 2. Öncelik ilişkileri bilgisini kullanarak Petri ağı modelini oluştur.

Adım 3. İşlem süreleri bilgisini, gerçekleşme ve tetikleme kuralını kullanarak Petri ağı modelindeki geçişlerin tetikleme sırasını oluştur.

Adım 4. Teorik çevrim süresini, başlangıç çevrim süresi olarak belirle.

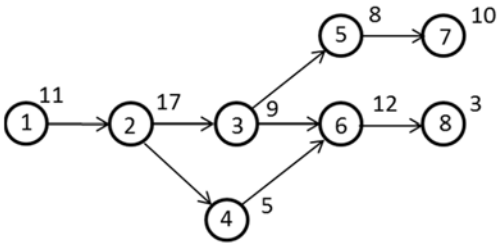
Adım 5. Eğer tüm görevler, iş istasyonlarına atandıysa Adım 8'e git. Atanmadıysa, tetikleme sırası, görev süresi, çevrim süresi ve o anki işaret dağılımı bilgilerini kullanarak geriye doğru prosedürü ile ilgili görevi iş istasyonuna ata.

Adım 6. Geriye kalan boş süreler halen atanmayan görevlerin toplam süreleri için yeterli mi kontrol et. Eğer yeterli ise atanan görevi tetikleme sırasından çıkar, istasyon boş süresini güncelle, ağdaki yeni işaret dağılımını belirle ve Adım 5'e git. Eğer geriye kalan boş süreler yeterli değilse Adım 7'ye git.

Adım 7. Şimdiye kadar yapılan tüm atamaları sil. Yeni çevrim süresi belirle. Yeni çevrim süresini kullanarak atamalara başlamak için Adım 5'e git.

Adım 8. İkili arama prosedürünü kullanarak çözümü iyileştir.

Önerilen algoritmanın işleyişi örnek bir problem üzerinde gösterilecektir. Bunun için Bowman'ın 8 görevli problemi örnek problem olarak belirlenmiştir. Probleme ait bilgiler Şekil 2'de verilmiştir.




**Şekil 2.** Bowman'ın 8 görevli problemi
(Bowman's 8 tasks problem)

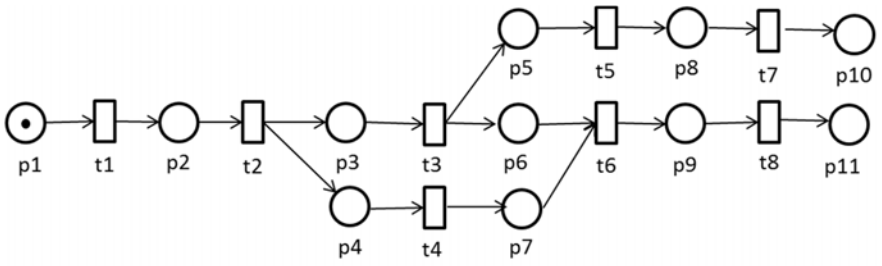
FSb-2 algoritması ilk olarak Şekil 2'de verilen öncelik ilişkileri diyagramını kullanarak probleme ait Petri ağı modelini oluşturur (Şekil 3). Görevler, Petri ağı modelinde geçişlerde gösterilmiştir. Şöyle ki, t1 nolu geçiş birinci görevi, t2 nolu geçiş ikinci görevi gösterir. Geçişler öncesi ve sonrasındaki girdi ve çıktı yerleri, ilgili görevin ön şartlarını ve sonraki durumlarını gösterir. Örnek vermek gerekirse, öncelik ilişkilerine göre birinci görev tamamlandıktan sonra ikinci görev başlar. Petri ağı modelinde birinci görevi gösteren t1 geçişinin tetiklenmesi (yani birinci görevin tamamlanması) ile t1'in çıktı yeri p2'ye bir işaret gelir. p2 aynı zamanda t2'nin (ikinci görevi gösteren geçişin) girdi yeridir. p2'de bir işaret olması, t2'nin tetiklenmesine olanak verir. Böylece ikinci görevin, birinci görev tamamlandıktan sonra yapılması sağlanmış olur. Bir başka tanımla, birinci görev iş istasyonuna atandıktan sonra, ikinci görev de süre kısıtını sağlıyorsa iş istasyonuna atanabilir.

Adım 3'te, algoritma tetikleme sırasını oluşturur. Bu sıra oluşturulurken görev süreleri, gerçekleşme ve tetikleme kurallarından faydalanılır. Her tetikleme sonrası işaret dağılımları Eş. 1 ile belirleneceğinden, Şekil 3'deki Petri ağı modeline ait ilişkiler matrisi (A) oluşturulmalıdır.

$$A = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}$$

A matrisinde satırlar geçişleri, kolonlar da yerleri göstermektedir. Matris değerleri, geçiş ve yer arasındaki bağlantıyı sağlayan okların ağırlık değerleridir. Örneğin t1 geçişinin girdi yeri p1 çıktı yeri p2'dir. Bu bağlantılar, A matrisinde ilk satırda birinci kolon için "-1", ikinci kolon için "1" olarak gösterilmiştir. Benzer şekilde 2 adet çıktı yeri (p3 ve p4) olan t2 için, A matrisinin ikinci satırının üçüncü ve dördüncü kolonlarında "1" olarak gösterilmiştir.

Şekil 3'teki Petri ağı modelininde işaretlerin hareket edebilmesi için başlangıç işaretlemesinin aşağıdaki gibi olması gerekir.




**Sekil 3.** Sekil 2'deki problemin Petri ağı modeli (Petri net model of the problem in figure 2.)

$$M_0 = [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$$

Bu işaret dağılımına göre, sıfır anında sadece t1 gerçekleşebileceğinden ilk olarak t1 seçilir. t1'in tetiklenmesi birinci görev süresi kadar sürer ve 11'de t1 tetiklenmiş olur. Eş.1 kullanılarak t1'in tetiklenmesinde sonraki işaret dağılımı belirlenir.

$$M_1 = M_0 + A_1 = [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0] +$$
$$[-1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$$
$$M_1 = [0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$$

Yeni işaret dağılımına göre t2, tetiklenecek tek geçiştir. t2 11'de tetiklenmeye başlar ve 28'de tetiklenmesi tamamlanır. Benzer şekilde tüm geçişler sıra ile tetiklenerek tetikleme sırası elde edilir. Eğer o anki işaret dağılımı birden fazla tetiklemenin başlamasına olanak veriyorsa tetiklemesi önce tamamlanan geçiş tetikleme sırasına önce eklenecektir. Şekil 2 ve Şekil 3'deki bilgililere göre tetikleme sırasının oluşumu Tablo 1'de verilmektedir.

Adım 4'te, başlangıç çözüm süresi için teorik çevrim süresi belirlenir.

$$Teorik\ çevrim\ süresi = \frac{\sum görev\ süreleri}{iş\ istasyonu\ sayısı} \tag{2}$$

Örnek problem için iş istasyonu sayısı dört alınırsa, Eş. 2'ye göre teorik çevrim süresi 19 olarak bulunur.

Adım 5'te görevlerin atanmasına başlanır. Çevrim süresi, görev süreleri, tetikleme sırası bilgileri kullanılarak geriye doğru prosedür ile görevler son iş istasyonundan başlanılarak atanır. Tetikleme sırasındaki son görev, son iş istasyonuna yerleştirilir. Görevlerin iş istasyonlarına atanması son işaret durumundan geriye doğru yapılacağından, Şekil 3'deki Petri ağı modelinin son işaret durumunun belirlenmesi gerekir. Bu işaret dağılımı da en son yani sekizinci işaret dağılımdır.

$$M_8 = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1]$$

Tetikleme sırasındaki son görev yedi olduğu için ve görev süresi de atamaya uygun olduğu için yedinci görev son iş istasyonuna atanır.

Adım 6'da, her atamadan sonra atanmayan görevlerin geride kalan boş istasyon sürelerini aşıp aşmadığı kontrol edilerek mevcut çevrim süresinin geçerliliği araştırılır. Bunun için Eş. 3'teki şart aranır.

$$\sum t_{ut} \le C \times (w \ - \ w_o) + it_{wl} \tag{3}$$

Eş. 3'te, $t_{ut}$ atanmamış görevlerin toplam süresini, $C$ çevrim süresini, $w$ toplam iş istasyonu sayısını, $w_o$ açılmış olan iş istasyonu sayısını ve $it_{wl}$ açılmış olan son iş istasyonun o anki boş süresini gösterir. Eğer Eş. 3'teki şart sağlanıyorsa, yeni işaret dağılımı belirlenir. Geriye doğru atama yapıldığından, geriye doğru Petri ağı işaretlemesinin güncellenmesi gerekir. Dolayısıyla Eş. 1 düzenlenerek, bir önceki işaret dağılımı Eş. 4'teki gibi bulunabilir.

$$M_{k-1} = M_k - A_i \tag{4}$$

Eş. 4 kullanılarak yedinci görevin atanması öncesi ya da bir başka anlatımla yedinci geçişin tetiklenmesinden önceki işaret dağılımı aşağıdaki gibi hesaplanır.

$$M_7 = M_8 - A_7 = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1] -$$
$$[0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad -1 \quad 0 \quad 1 \quad 0]$$
$$M_7 = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1]$$

İş istasyonu boş süresi güncellenir ve atanan yedinci iş, tetikleme sırasından silinir. Algoritma yeni atamalar için Adım 5'e döner.

Atama sürerken bazen tetikleme sırasındaki son görev, görev süresinin çevrim süresini aşmasından dolayı mevcut iş istasyonuna atanamaz. Bu durumda algoritma, tetikleme sırasındaki atanmamış görevlerden öncelik ilişkisi koşulunu sağlayan görevleri belirler. Bu görevler içinden, iş istasyonunun boş süresini en küçükleyecek görev iş istasyonuna atanır. Eğer herhangi bir atama yapılamazsa yeni iş istasyonu açılır. Her atama sonrasında ya da yeni iş istasyonu açılması sonrasında, Adım 6'ya gidilerek Eş. 3'te verilen şart kontrol edilir. Eğer tüm görevlerin iş istasyonlarına atanması tamamlanmışsa, geçerli çevrim süresi olurlu çözüm olarak not edilir. Olurlu çözümün iyileştirilmesi için Adım 8'e gidilir. Eğer Adım 6'da Eş. 3'teki şart sağlanmazsa, algoritma Adım 7'ye geri döner.

**Tablo 1.** Tetikleme sırasının oluşturulması (Obtaining of firing order)

| Zaman | İşaret dağılımı (M) | Gerçekleşebilecek geçişler | Tetiklenmiş geçiş | Tetikleme sırası |
|---|---|---|---|---|
| 0 | [1 0 0 0 0 0 0 0 0 0 0] | t1 | - | - |
| 11 | [0 1 0 0 0 0 0 0 0 0 0] | t2 | t1 | 1 |
| 28 | [0 0 1 1 0 0 0 0 0 0 0] | t3-t4 | t2 | 1-2 |
| 33 | [0 0 1 0 0 0 1 0 0 0 0] | t3 | t4 | 1-2-4 |
| 37 | [0 0 0 0 1 1 1 0 0 0 0] | t5-t6 | t3 | 1-2-4-3 |
| 45 | [0 0 0 0 0 1 1 1 0 0 0] | t6-t7 | t5 | 1-2-4-3-5 |
| 49 | [0 0 0 0 0 0 0 1 1 0 0] | t7-t8 | t6 | 1-2-4-3-5-6 |
| 52 | [0 0 0 0 0 0 0 1 0 0 1] | t7 | t8 | 1-2-4-3-5-6-8 |
| 55 | [0 0 0 0 0 0 0 0 0 1 1] | - | t7 | 1-2-4-3-5-6-8-7 |

Adım 7'de, mevcut çevrim süresi, atanmayan görevlerin iş istasyonlarına yerleştirilmesine olanak vermeyeceği için yeni bir çevrim süresi hesaplanır. Eş. 5 ile yeni çevrim süresi belirlenir.

$$C_{yeni} = C + \frac{\sum t_{ut} - \left(C \times \left(w - w_o\right) - it_{wl}\right)}{w} \quad (5)$$

Yeni çevrim süresi belirlendikten sonra, o ana dek yapılmış tüm atamalar silinir. Algoritma, Adım 5'e dönerek yeni çevrim süresi için yeniden çözüme başlar. Bu adımlar, Eş. 3'teki şartın sağlandığı bir çevrim süresi bulunana ve bu çevrim süresine göre tüm görevlerin iş istasyonlarına atanmasına kadar devam eder. Bir olurlu çözümün bulunması, algoritmanın ilk aşamasının bittiğini gösterir. Şekil 2 ve Şekil 3'teki problem için ilk aşama adımları Tablo 2'de verilmektedir.

Adım 8'de algoritmanın ikinci aşamasında, bulunan olurlu çözüm ikili arama yöntemi ile iyileştirilir. İkili arama için alt sınır, bir önceki aşamada bulunan olurlu olmayan çevrim süresinin bir fazlası, üst sınır ise birinci aşamada bulunan olurlu çevrim süresidir. İkili arama yöntemi ile bulunan yeni çevrim süresi olurlu çözüm verirse, üst sınır olarak kaydedilir. Eğer çözüm olurlu değilse, çevrim süresinin bir fazlası alt sınır olarak not edilir. İkili arama yöntemi, alt ve üst sınırlar arasındaki fark bir olana kadar devam eder. Fark bir olduğunda, algoritma son çözümü alt sınırdaki çevrim süresi için yapar. Örnek problem için, Tablo 2'den de görüleceği üzere çevrim süresi 23 olduğunda bir olurlu çözüm bulunmuştur. Bu değer ikili aramada üst sınırdır. Birinci aşamadaki son olurlu olmayan çözümde çevrim süresi 21 bulunduğundan, ikili aramadaki alt sınır 22'dir. Alt ve üst sınır arasındaki fark bir olduğundan, algoritma alt sınır olan 22'yi çevrim süresi olarak alır. Her atama sonrası Eş. 3'teki şart sağlandığından, dört iş istasyonu için çevrim süresi 22 olan çözüm bulunmuş olur. Elde edilen çözüme göre, ilk görev ilk iş istasyonuna, 2 ve 4. görevler 2. iş istasyonuna, 3 ve 6. görevler 3. iş istasyonuna, 5, 8 ve 7. görevler de son iş istasyonuna atanmıştır. İyileştirme aşaması Tablo 3'te verilmektedir.

**Tablo 2.** Örnek problem için, ilk aşamadaki atama prosedürü.
(Assignment procedures in the first stage for the example problem.)

| C | İşaret dağılımı | Tetikleme sırası | Atanan görev | $w_o$ | $it_{wl}$ | $t_{ut}$ | $C \times \left(w - w_o\right) + it_{wl}$ | Eş. 3 |
|---|---|---|---|---|---|---|---|---|
| 19 | [0 0 0 0 0 0 0 0 0 1 1] | 1-2-4-3-5-6-8-7 | 7 | 1 | 9 | 65 | 19x(4-1)+9=66 | Doğru |
| 19 | [0 0 0 0 0 0 0 1 0 0 1] | 1-2-4-3-5-6-8 | 8 | 1 | 6 | 62 | 19x(4-1)+6=63 | Doğru |
| 19 | [0 0 0 0 0 0 0 1 1 0 0] | 1-2-4-3-5-6 | 6 | 2 | 7 | 50 | 19x(4-2)+7=45 | Yanlış |
| 21 | [0 0 0 0 0 0 0 0 0 1 1] | 1-2-4-3-5-6-8-7 | 7 | 1 | 11 | 65 | 21x(4-1)+11=74 | Doğru |
| 21 | [0 0 0 0 0 0 0 1 0 0 1] | 1-2-4-3-5-6-8 | 8 | 1 | 8 | 62 | 21x(4-1)+8=71 | Doğru |
| 21 | [0 0 0 0 1 0 0 0 0 0 1] | 1-2-4-3-5-6 | 5 | 1 | 0 | 54 | 21x(4-1)+0=63 | Doğru |
| 21 | [0 0 0 0 1 0 0 0 0 0 1] | 1-2-4-3-6 | 6 | 2 | 9 | 42 | 21x(4-2)+9=51 | Doğru |
| 21 | [0 0 0 0 1 1 1 0 0 0 0] | 1-2-4-3 | 3 | 2 | 0 | 33 | 21x(4-2)+0=42 | Doğru |
| 21 | [0 0 1 0 0 0 1 0 0 0 0] | 1-2-4 | 4 | 3 | 16 | 28 | 21x(4-3)+16=37 | Doğru |
| 21 | [0 0 1 1 0 0 0 0 0 0 0] | 1-2 | 2 | 4 | 4 | 11 | 21x(4-4)+4=4 | Yanlış |
| 23 | [0 0 0 0 0 0 0 0 0 1 1] | 1-2-4-3-5-6-8-7 | 7 | 1 | 13 | 65 | 23x(4-1)+13=82 | Doğru |
| 23 | [0 0 0 0 0 0 0 1 0 0 1] | 1-2-4-3-5-6-8 | 8 | 1 | 10 | 62 | 23x(4-1)+10=79 | Doğru |
| 23 | [0 0 0 0 1 0 0 0 0 0 1] | 1-2-4-3-5-6 | 5 | 1 | 2 | 54 | 23x(4-1)+2=71 | Doğru |
| 23 | [0 0 0 0 1 0 0 0 0 0 1] | 1-2-4-3-6 | 6 | 2 | 11 | 42 | 23x(4-2)+11=57 | Doğru |
| 23 | [0 0 0 0 1 1 1 0 0 0 0] | 1-2-4-3 | 3 | 2 | 2 | 33 | 23x(4-2)+2=48 | Doğru |
| 23 | [0 0 1 0 0 0 1 0 0 0 0] | 1-2-4 | 4 | 3 | 18 | 28 | 23x(4-3)+18=41 | Doğru |
| 23 | [0 0 1 1 0 0 0 0 0 0 0] | 1-2 | 2 | 3 | 1 | 11 | 23x(4-3)+1=24 | Doğru |
| 23 | [0 1 0 0 0 0 0 0 0 0 0] | 1 | 1 | 4 | 12 | 0 | 23x(4-4)+12=12 | Doğru |

**Tablo 3.** Örnek problem için, iyileştirme aşamasındaki atama prosedürü.
(Assignment procedures in the improvement stage for the example problem.)

| *C* | İşaret dağılımı | Tetikleme sırası | Atanan görev | $w_o$ | $it_{wl}$ | $t_{ut}$ | $C \times \left(w - w_o\right) + it_{wl}$ | Eş. 3 |
|---|---|---|---|---|---|---|---|---|
| 22 | [0 0 0 0 0 0 0 0 0 1 1] | 1-2-4-3-5-6-8-7 | 7 | 1 | 12 | 65 | 22x(4-1)+12=78 | Doğru |
| 22 | [0 0 0 0 0 0 0 1 0 0 1] | 1-2-4-3-5-6-8 | 8 | 1 | 9 | 62 | 22x(4-1)+10=76 | Doğru |
| 22 | [0 0 0 0 1 0 0 0 0 0 1] | 1-2-4-3-5-6 | 5 | 1 | 1 | 54 | 22x(4-1)+1=67 | Doğru |
| 22 | [0 0 0 0 1 0 0 0 0 0 1] | 1-2-4-3-6 | 6 | 2 | 10 | 42 | 22x(4-2)+10=54 | Doğru |
| 22 | [0 0 0 0 1 1 1 0 0 0 0] | 1-2-4-3 | 3 | 2 | 1 | 33 | 22x(4-2)+1=45 | Doğru |
| 22 | [0 0 1 0 0 0 1 0 0 0 0] | 1-2-4 | 4 | 3 | 17 | 28 | 22x(4-3)+17=39 | Doğru |
| 22 | [0 0 1 1 0 0 0 0 0 0 0] | 1-2 | 2 | 3 | 0 | 11 | 22x(4-3)+0=22 | Doğru |
| 22 | [0 1 0 0 0 0 0 0 0 0 0] | 1 | 1 | 4 | 11 | 0 | 22x(4-4)+11=11 | Doğru |

## 4. SAYISAL ANALİZLER ( COMPUTATIONAL RESULTS)

Önerilen algoritmanın etkinliğini incelemek için literatürde bilinen 302 test problemi kullanılmıştır. Problem bilgileri Tablo 4'te verilmektedir [9]. Bu bölümde algoritmanın etkinliği iki aşamada incelenmiştir. Birinci aşamada, önerilen FSb-2 algoritması atama sırası oluşturan bir kurucu sezgisel olduğundan, literatürde BMHDP-2 için en iyi çözümü veren öncelik kuralı tabanlı kurucu sezgisel yöntemlerle karşılaştırılacaktır. İkinci aşamada ise, FSb-2 sonuçları literatürdeki diğer Petri ağı tabanlı algoritmalar, genetik algoritmalar ve diferansiyel evrim algoritmaları sonuçları ile karşılaştırılacaktır. Bu sonuçları elde etmek için, algoritma MATLAB R2016a'da kodlanmış ve 3,20-GHz Core i5-3470 işlemcili, 4 GB RAM'li bir bilgisayarda çalıştırılmıştır.

### *4.1. Fsb-2 ve Bilinen Öncelik Kuralları Arasındaki Karşılaştırma Sonuçları*
*(Comparison Results Between Fsb-2 and Well-Known Priority Rules)*

FSb-2 algoritması bir tetikleme sırası oluşturduğundan ve atama için bu sırayı kullandığından, önerilen algoritma bir öncelik kuralı olarak yorumlanabilir. Bu nedenle FSb-2 literatürde BMHDP-2 için en iyi çözümleri veren öncelik kuralları ile karşılaştırılacaktır. Scholl ve Voß [9] çalışmalarında BMHDP-2 için en iyi öncelik kurallarını şu şekilde listelemişlerdir;

- En büyük takip eden sayısı (Maximum number of followers, MaxF): Öncelik ilişkilerine göre, her görevin kendisinden sonra gelen görevlerin sayısı belirlenir ve büyükten küçüğe sıralanır.
- En büyük konumsal ağırlık (Maximum positional weight, MaxPW). Kendisi ve kendisinden sonra gelen görevlerin süreleri toplanır ve büyükten küçüğe sıralanır.
- En büyük görev süresi / en son istasyon (Maximum task time divided by latest station, MaxTimeL): Her bir görev süresi, o görevin en son atanabileceği iş istasyonu sayısına bölünür, değerler büyükten küçüğe sıralanır.
- En büyük görev süresi / boşluk (Maximum task time divided by slack, MaxTimeSlack): Her bir görev süresi, o görevin en son atanabileceği iş istasyonu sayısı ile en erken atanabileceği iş istasyonu sayısı arasındaki farka bölünür, değerler büyükten küçüğe sıralanır.
- En büyük MaxTimeSlack / ileri ve geriye doğru atanabilecek iş sayısı (Maximum MaxTimeSlack divided by number of forward and backward available tasks, MaxTSAvail): Bir önceki öncelik kuralında (MaxTimeSlack) bulunan değerler, o aşamada ileriye ve geriye doğru atanabilecek iş sayına bölünür ve büyükten küçüğe sıralanır.

Karşılaştırmada kullanılacak ilk dört kural, hem ileriye doğru, hem geriye doğru, hem de yönsüz yöntemler uygulanabilen kurallardır. Son kural dinamik bir kural olduğundan sadece yönsüz yöntem uygulanmasına olanak verir. Tüm olası yöntemler uygulanılarak öncelik kuralları ile elde edilen sonuçlar ve FSb-2 sonuçları Tablo 5'te verilmiştir. Sonuçlara bakıldığında optimum çözüm sayısında (# opt.) FSb-2 en iyi olmakla birlikte diğer kurallarla arasında belirgin bir fark yoktur. FSb-2 302 test probleminin 35'inde optimum çözüm bulmuştur. MaxTimeL ve MaxTSAvail öncelik kurallarının yönsüz versiyonları ise, sıra ile 34 ve 33 optimum çözüm elde etmiştir. Optimumdan sapmaların yüzdesel değerlerine (% ort. sp.) bakıldığında, FSb-2 diğer kurallara karşı bir üstünlük kurmuştur. FSb-2, 302 test problemin optimum çözümlerine göre ortalama %1,14'lük sapma ile çözümler bulurken, karşılaştırılan öncelik kuralları içinde ortalama sapma değeri %2,29-3,28 aralığında değişmektedir. En iyi % ortalama sapma değeri

**Tablo 4.** Sayısal analizlerde kullanılan problemlere ait bilgiler
(Information about the problems used in the computational results)

| Problem ismi | Problem boyutu (problemdeki görev sayısı) | Problemdeki en küçük ve en büyük görev süreleri | Problemdeki görevlerin sürelerinin toplamı | Problemden üretilen örnek sayısı |
|---|---|---|---|---|
| Arcus 1 | 83 | 233-3691 | 75707 | 20 |
| Arcus 2 | 111 | 10-5689 | 150399 | 25 |
| Bartholdi 1 | 148 | 3-383 | 5634 | 13 |
| Bartholdi 2 | 148 | 1-83 | 4234 | 25 |
| Buxey | 29 | 1-25 | 324 | 8 |
| Gunther | 35 | 1-40 | 483 | 10 |
| Hahn | 53 | 40-1775 | 14026 | 8 |
| Kilbridge | 45 | 3-55 | 552 | 9 |
| Lutz 1 | 32 | 100-1400 | 14140 | 5 |
| Lutz 2 | 89 | 1-10 | 485 | 20 |
| Lutz 3 | 89 | 1-74 | 1644 | 21 |
| Mukherje | 94 | 8-171 | 4208 | 24 |
| Sawyer | 30 | 1-25 | 324 | 8 |
| Scholl | 297 | 5-1386 | 69655 | 28 |
| Tonge | 70 | 1-156 | 3510 | 23 |
| Warnecke | 58 | 7-53 | 1548 | 27 |
| Wee-Mag | 75 | 2-27 | 1499 | 28 |

**Tablo 5.** FSb-2 ve en iyi öncelik kuralları arasında karşılaştırma sonuçları
(Comparison results between FSb-2 and the best priority rules)

| | Kurallar | # opt. | % ort.sp. |
|---|---|---|---|
| İleriye doğru yöntemler | MaxF | 22 | 2,57 |
| | MaxPW | 22 | 2,57 |
| | MaxTimeL | 31 | 2,74 |
| | MaxTimeSlack | 28 | 2,79 |
| Geriye doğru yöntemler | MaxF | 25 | 2,47 |
| | MaxPW | 25 | 2,52 |
| | MaxTimeL | 29 | 3,28 |
| | MaxTimeSlack | 29 | 3,25 |
| Yönsüz yöntemler | MaxF | 28 | 2,29 |
| | MaxPW | 25 | 2,58 |
| | MaxTimeL | 34 | 2,78 |
| | MaxTimeSlack | 31 | 2,76 |
| | MaxTSAvail | 33 | 2,63 |
| Önerilen yöntem | FSb-2 | 35 | 1,14 |

2,29 ile MaxF öncelik kuralının yönsüz versiyonunda elde edilmiştir ki bu değer, FSb-2 ile elde edilen değerin iki katından fazladır. Scholl ve Voß [9] çalışmalarında, tüm kurallar için ortalama 1 saniye civarında çözüm aldıklarını, en uzun çalışma süresinin de 8 saniye civarında olduğunu belirtmişlerdir. FSb-2 ile elde edilen sonuçlarda ortalama çalışma süresi yarım saniyenin altında, en uzun çalışma süresi de 5 saniyenin altındadır.

### *4.2. Fsb-2, Petri Ağı Tabanlı ve Metasezgisel Tabanlı Algoritmalar Arasında Karşılaştırma Çalışması*
*(Comparison Study Between Fsb-2, Petri Net-Based, and Metaheuristic-Based Algorithms)*

Bu bölümde FSb-2 algoritmasının performansı, BMHDP-2 literatüründeki diğer Petri ağı tabanlı algoritmalar, bazı genetik algoritmalar ve diferansiyel evrim algoritmaları sonuçları ile karşılaştırılacaktır. Nearchou [11] BMHDP-2'yi çözmek için diferansiyel evrim algoritması tabanlı sezgisel geliştirmiştir. Geliştirdiği sezgiselin iki farklı versiyonu olup, kodlamaları oluştururken ya bir öncelik tabanlı (DE_prb) ya da rasgele oluşturulan (DE_rks) bir şema kullanmıştır.

Geliştirdiği bu sezgisellerin performansını test etmek için sonuçları literatürdeki Kim vd.'nin [7] geliştirdiği genetik algoritma (pGA), Goncalves ve Almeida'nın [24] önerdiği evrimsel algoritmalar (rGA_prb ve rGA_rks) ile karşılaştırmıştır. Ayrıca Zhang vd. [15] kendi önerdikleri tamsayı kodlamalı diferansiyel evrim algoritmasının iki farklı versiyonunu (IDEA_rd2 ve IDEA_apt) Nearchou'nun [11] karşılaştırma sonuçları ile kıyaslamışlardır. Kilincci[12] BMHDP-2 problemini çözmek için Petri ağlarının ulaşılabilirlik özelliğini kullanarak Petri ağı tabanlı bir algoritma geliştirmiştir. Bu algoritma için ileriye doğru, geriye doğru ve yönsüz olarak üç farklı versiyon oluşturmuştur (PNA-for, PNA-back ve PNA-bid). Tüm bu yöntemler, bu çalışmada önerilen FSB-2 algoritması ile karşılaştırılmıştır.

Her bir yöntemin optimum çözümden sapma yüzde değerleri Tablo 6'da verilmiştir. Sonuçlar Nearchou'nun[ 11] karşılaştırma tablosu temel alınarak derlenmiştir. Nearchou [11] tüm veri setini iki alt sete bölmüş, sonuçları hem problem bazında hem alt veri setleri bazında hem de bütün olarak değerlendirmiştir. Tablo 6'dan görüleceği üzere Nearchou [11], 94 görevli Mukherje problemine ait çözümleri sonuçlar içinde göstermemiştir. Bunun nedeni olarak, bu problemde elde edilen çözümlerin optimum çözümden sapma yüzde değerlerinin kabul edilebilir değerlerde olmadığını belirtmiştir. Zhang vd. [15], Nearchou'nun [11] sonuçlarını temel aldığından, onlar da Mukherje problemine ait herhangi bir sonuç vermemiştir. Hem Kilincci [12] hem de bu çalışmada, Mukherje problemine ait sonuçlar ayrıca gösterilmiştir. Tüm veri setinin Mukherje'siz ortalama sonuçlarına bakıldığında, FSb-2'nin karşılaştırılan yöntemler içinde en iyi sonucu verdiği görülmektedir. İlk veri setinin ortalamalarına göre, FSb-2 karşılaştırılan yöntemler içinde %2,20 ortalama sapma değeri ile orta sıralarda yer alırken, ikinci veri seti sonuçlarında %1,87 ile en iyi çözüm veren yöntem olmuştur. Tüm bu sonuçlar, önerilen algoritmanın BMHDP-2 için uygulanabilir bir yöntem olduğunu göstermektedir. Ayrıca problem bazlı incelendiğinde FSb-2'nin, Tonge, Arcus 2, Bartholdi 2, Scholl ve Mukherje problemlerinde en iyi yüzde ortalama sapmaları elde ettiği görülmektedir. Bartholdi 1 problemi için de en iyi sonuç veren algoritmaya yakın bir sonuç (%0,25) bulduğu dikkate alınırsa, önerilen FSb-2 algoritmasının özellikle 94 ve üzerindeki görev sayılı problemlerde etkin bir yöntem olduğu söylenebilir. Nearchou'nun [11] çalışmasında tüm yöntemler için ortalama çözüm süreleri, ilk veri seti için 2,66-29,61 saniye aralığında, ikinci veri seti için 3,51-306,57 saniye aralığında verilmiştir. Zhang vd. [15] herhangi bir çözüm süresi bilgisi vermemiştir. FSb-2 ile ilk veri seti için ortalama çözüm süresi 0,30 saniye, ikinci veri seti içinse 0,42 saniyedir. Çözüm süreleri açısından incelendiğinde, FSb-2'nin BMHDP-2 için kısa sürede iyi sonuçlar veren bir algoritma olduğu görülmektedir.

**Tablo 6.** Karşılaştırma sonuçları (Comparative results)

| Problem ismi | Problem boyutu | DE_ prb | DE_ rks | pGA | rGA_ prb | rGA_ rks | IDEA _rd2 | IDEA _apt | PNA -for | PNA- back | PNA- bid | FSb-2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Veri seti 1 | | | | | | | | | | | | |
| Buxey | 29 | 2,69 | 1,16 | 3,26 | 2,46 | 2,15 | 2,21 | 3,07 | 3,07 | 2,92 | 3,76 | 2,62 |
| Sawyer | 30 | 3,52 | 2,27 | 2,61 | 3,67 | 4,51 | 3,34 | 3,52 | 4,00 | 4,39 | 5,24 | 4,44 |
| Lutz 1 | 32 | 0,35 | 0,32 | 0,32 | 0 | 0,71 | 0,35 | 0,71 | 4,09 | 2,93 | 4,09 | 1,99 |
| Gunther | 35 | 1,02 | 0,14 | 0,64 | 1,21 | 1,71 | 1,02 | 1,27 | 1,27 | 2,99 | 1,27 | 3,14 |
| Kilbridge | 45 | 0,6 | 0,66 | 1,38 | 0,8 | 0,96 | 0,60 | 0,60 | 2,19 | 1,46 | 1,91 | 0,68 |
| Tonge | 70 | 2,07 | 1,88 | 2,75 | 3,63 | 3,63 | 2,07 | 1,78 | 2,53 | 3,41 | 2,92 | 1,63 |
| Arcus 1 | 83 | 0,83 | 0,99 | 1,65 | 1,48 | 1,96 | 0,98 | 0,78 | 2,47 | 2,57 | 2,25 | 2,36 |
| Lutz 2 | 89 | 2,54 | 3,08 | 3,13 | 2,84 | 3,76 | 2,64 | 1,99 | 2,99 | 4,44 | 3,92 | 3,17 |
| Arcus 2 | 111 | 4,98 | 4,96 | 5,02 | 5,11 | 5,27 | 4,98 | 4,64 | 2,06 | 4,61 | 2,25 | 1,19 |
| Ortalama | | 2,07 | 1,72 | 2,31 | 2,36 | 2,74 | 2,02 | 2,04 | 2,57 | 3,51 | 2,88 | 2,20 |
| Veri seti 2 | | | | | | | | | | | | |
| Hahn | 53 | 0 | 0 | 0 | 0 | 1,27 | 0 | 0 | 2,52 | 2,08 | 2,61 | 0,81 |
| Warnecke | 58 | 3,31 | 3,74 | 4,29 | 5,25 | 6,40 | 3,98 | 3,51 | 5,57 | 5,90 | 6,01 | 3,84 |
| Wee-Mag | 75 | 1,39 | 1,23 | 2,61 | 2,44 | 2,37 | 1,63 | 1,39 | 1,56 | 1,63 | 1,57 | 2,20 |
| Lutz 3 | 89 | 1,58 | 1,68 | 2,54 | 2,19 | 3,20 | 2,88 | 1,57 | 2,59 | 2,88 | 2,94 | 2,34 |
| Mukherje | 94 | | | | | | | | 1,04 | 1,41 | 1,21 | 0,94 |
| Barthold 1 | 148 | 0,24 | 0,26 | 0,85 | 0,58 | 0,76 | 0,46 | 0,26 | 1,02 | 0,46 | 0,48 | 0,25 |
| Barthold 2 | 148 | 7,37 | 6,85 | 9,64 | 9,88 | 9,77 | 4,79 | 4,79 | 3,97 | 4,79 | 4,32 | 2,10 |
| Scholl | 297 | 9,87 | 9,51 | 10,16 | 10,31 | 11,76 | 10,16 | 9,24 | 3,21 | 2,90 | 2,75 | 0,93 |
| Ortalama | | 3,39 | 3,32 | 4,27 | 4,38 | 5,08 | 3,41 | 2,97 | 2,85 | 3,01 | 2,93 | 1,87 |
| Veri seti 1 ve 2 | | | | | | | | | | | | |
| Ortalama (Mukherje'siz) | | 2,78 | 2,58 | 3,37 | 3,45 | 4,00 | 3,06 | 2,80 | 2,88 | 3,38 | 3,06 | 2,10 |
| Ortalama (Mukherje'li) | | | | | | | | | 2,73 | 3,22 | 2,91 | 2,01 |

## 5. SONUÇLAR (CONCLUSIONS)

Bu çalışmada, BMHDP-2 çözümü için Petri ağı tabanlı FSb-2 algoritması önerilmiştir. Önerilen algoritma iki aşamalı bir algoritmadır. İlk aşamada, problemin öncelik ilişkileri dikkate alınarak oluşturulan Petri ağı modeli yardımıyla görev süreleri de kullanılarak bir tetikleme sırası bulunur. Bu sıra ve geriye doğru prosedürü kullanarak BMHDP-2 için olurlu bir çözüm elde edilir. İkinci aşamada bu olurlu çözüm ikili arama yöntemi ile iyileştirilir.

Önerilen FSb-2'nin performansını test etmek için hem BMHDP-2'de etkin öncelik kuralları ile hem de Petri ağı, genetik algoritma ve diferansiyel evrim algoritmaları tabanlı algoritmalar ile karşılaştırmalar yapılmıştır. Bu karşılaştırmalarda, önerilen FSb-2, ortalama sapma değeri olarak sırası ile %1,14 ve %2,02 değerleri elde ederek, karşılaştırmalar içinde en iyi çözümü veren yöntem olmuştur. FSb-2 ile elde edilen çözümler, en fazla 5 saniyede alınmıştır. Tüm bu sonuçlar FSb-2 algoritmasının BMHDP-2 için etkin bir algoritma olduğunu göstermiştir. Ayrıca, özellikle 94 ve üzeri görevli problemlerde FSb-2 ile elde edilen çözümler, karşılaştırma yapılan diğer tüm algoritmalardan elde edilen çözümlerden oldukça iyidir. Örneğin FSb-2 Arcus 2 için %1,19, Bartholdi 2 için %2,10, Scholl için %0,93 ve Mukherje için %0,94 ortalama sapma değerleri bulurken, karşılaştırma yapılan algoritmaların en iyi sonuçları Arcus 2 için %2,25, Bartholdi 2 için %3,97, Scholl için %2,75 ve Mukherje için %1,04'tür. Bu sonuçlar, FSb-2'nin özellikle çok sayıda görev içeren büyük boyutlu BMHDP-2'de başarılı sonuçlar verdiğini göstermektedir. Çalışma sonuçlarını, karar vericiler açısından yorumlamak gerekirse, bir montaj hattında, aynı anda tek çeşit ürün üreten işletmelerde yeni ürün üretimine geçişlerde, değişim maliyetlerinin olabildiğince az olması istenir. Bu amaçla hattaki iş istasyonu sayısı değiştirilmeden, yeni ürüne ait görevler ve süreleri dikkate alınarak BMHDP-2 çözülüp uygun çevrim süresi belirlenir. Bu çalışmada önerilen FSb-2 algoritmasının, BMHDP-2 için kısa sürede iyi çözümler üreten bir algoritma olduğu görülmüştür. Üretim sahalarında BMHDP-2 ile karşılaşan karar vericilerin, FSb-2 algoritması kullanarak kısa sürede iyi sonuçlar elde edebileceği söylenebilir.

Önerilen FSb-2 algoritması elde ettiği sonuçlara göre BMHDP-2 literatüründe etkin kurucu sezgisellerden biridir. Benzer mantıkla tasarlanan FSb algoritması da BMHDP-1'de etkin çözümler vermiştir. Bu sonuçlar algoritmanın çözüm bulma işleyişinin bu iki problemde başarılı olduğunu göstermektedir. Gelecekte yapılacak çalışmaların bir kolu, bu çözüm bulma işleyişinin diğer MHDP versiyonlarına uygulanmasıdır. Örneğin U tipi MHDP, karışık MHDP, paralel MHDP, iki yönlü MHDP ve benzeri problem versiyonlarında, FSb-2 algoritmasının işleyişine gerekli uyarlamalar yapılarak algoritmanın bu problemlerdeki etkinliği araştırılabilir. Gelecekte yapılacak çalışmalara bir

diğer öneri de, FSb-2 algoritmasının metasezgisellerle birlikte kullanılmasıdır. Son yıllarda literatürdeki birçok NP-zor problemin çözümünde metasezgiseller kullanılmaktadır. Her bir metasezgiselin kendine göre avantajlı ve dezavantajlı yanları vardır. Metasezgisellerin performansını arttırmak için ya birbirleri ile ya da başka sezgisellerle hibrid edilmektedir. FSb-2 algoritması bu amaçla bir metasezgiselle hibrid edilerek, metasezgiselin performansı arttırılabilir.

## 6. SİMGELER (SYMBOLS)

Mk : k. andaki Petri ağı modelindeki işaret dağılımını gösteren satır vektörü (1xm boyutunda)
m : Petri ağı modelindeki yer sayısı
M(p) : p. yerde bulunan işaret sayısı
M0 : Petri ağı modelindeki başlangıç anındaki işaret dağılımı
A : İlişkiler matrisi (mxn boyutunda)
n : Petri ağı modelindeki geçiş sayısı
Ai : A matrsinin i. satırı
$t_{ut}$ : atanmamış görevlerin toplam süresi
$C$ : çevrim süresi
$w$ : toplam iş istasyonu sayısı
$w_o$ : açılmış olan iş istasyonu sayısı
$it_{wl}$ : açılmış olan son iş istasyonun o anki boş süresi
Cyeni : yeni çevrim süresi (Eş. 5)

## KAYNAKLAR (REFERENCES)


1. Scholl A. ve Becker C., State-of-the-art exact and heuristic solution procedures for simple assembly line balancing, Eur. J. Oper. Res., 168 (3), 666-693, 2006.
2. Boysen N., Fliedner M., Scholl A., A classification of assembly line balancing problems, Eur. J. Oper. Res., 183 (2), 674-693, 2007.
3. Boysen N., Fliedner M., Scholl A., Assembly line balancing: Which model to use when?, Int. J. Prod. Econ., 111 (2), 509-528, 2008.
4. Rashid M. F. F., Hutabarat W., Tiwari A., A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches, Int. J. Adv. Manuf. Technol., 59 (1-4), 335-349, 2012.
5. Battaia O. ve Dolgui A., A taxonomy of line balancing problems and their solution approaches, Int. J. Prod. Econ., , 142 (2), 259-277, 2013.
6. Sivasankaran P. ve Shahabudeen P., Literature review of assembly line balancing problems, Int. J. Adv. Manuf. Technol., 73 (9), 1665-1694, 2014.
7. Kim Y. K., Kim Y. J., Kim Y., Genetic algorithms for assembly line balancing with various objectives, Comput. Ind. Eng., 30, 397–409, 1996.
8. Klein R. ve Scholl A., Maximizing the production rate in simple assembly line balancing—a branch and bound procedure, Eur. J. Oper. Res., 91, 367–385, 1996.
9. Scholl A. ve Voß S., Simple assembly line balancing-heuristic approaches, Journal of Heuristics, ,2, 217–244, 1996.
10. Uğurdağ H. F., Rachamadugu R., d Papachritou C. A., Designing paced assembly lines with fixed number of stations, Eur. J. Oper. Res., 102, 488–501,1997.
11. Nearchou A. C., Balancing large assembly lines by a new heuristic based on differential evolution method, Int. J. Adv. Manuf. Technol., 34, 1016–1029, 2007.
12. Kilincci O., A Petri net-based heuristic for simple assembly line balancing problem of type 2, The Int. J. Adv. Manuf. Technol., 46 (1), 329-338, 2010.
13. Blum C., Iterative beam search for simple assembly line balancing with a fixed number of work stations, SORT 35 (2), 145-164, 2011.
14. Zheng Q., Li M., Li Y., Tang Q., Station ant colony optimization for the type 2 assembly line balancing problem, Int. J. Adv. Manuf. Technol., 66 (9-12), 1859-1870, 2013.
15. Zhang H., Yan Q., Liu Y., Jiang Z. An integer-coded differential evolution algorithm for simple assembly line balancing problem of type 2, Assembly Automation, 36 (3), 246-261, 2016.
16. Arıkan M., A tabu search algorithm for the simple assembly line balancing problem of type-2 with workload balancing objective. Journal of the Faculty of Engineering and Architecture of Gazi University 32 (4), 1169-1179, 2017.
17. Kilincci O., Firing sequences backward algorithm for simple assembly line balancing problem of type 1, Comput. Ind. Eng., 60 (4), 830-839, 2011.
18. Otto A., Otto C., Scholl A., Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing, Eur. J. Oper. Res., 228, 33–45, 2013.
19. Otto A. ve Otto C., How to design effective priority rules: Example of simple assembly line balancing, Comput. Ind. Eng., 69 (1), 43-52, 2014.
20. Murata T., Petri nets: properties - analysis and applications, Proceedings of IEEE, 77 (4), 541–580, 1989.
21. Zurawski R. ve Zhou M., Petri nets and industrial applications: A Tutorial, IEEE Trans. Ind. Electron., 41 (6), 567–581, 1994.
22. Kilincci O., ve Bayhan G. M., A Petri net approach for simple assembly line balancing problems, Int. J. Adv. Manuf. Technol., 30 (11–12), 1165–1173, 2006.
23. Kilincci O., ve Bayhan G. M., A P-invariant-based algorithm for simple assembly line balancing problem of type-1, Int. J. Adv. Manuf. Technol., 37 (3–4), 400–409, 2008.
24. Goncalves J. F., ve Almeida J. R., A hybrid genetic algorithm for assembly line balancing, Journal of Heuristics, 8 (6), 629–642, 2002.