DEÜ MÜHENDİSLİK FAKÜLTESİ
FEN ve MÜHENDİSLİK DERGİSİ
Cilt: 6 Sayı: 2 s. 1-15 Mayıs 2004

BASİT MONTAJ HATTI DENGELEME PROBLEMİ ÇÖZÜMÜ İÇİN BİR PETRİ AĞI YAKLAŞIMI

(*A PETRI NET APPROACH FOR SOLVING SIMPLE ASSEMBLY LINE BALANCING PROBLEM)*

Özcan KILINÇCI*

ÖZET / *ABSTRACT*

Montaj hattı dengeleme problemi bir atama problemidir. İş elemanları, hat üzerindeki iş istasyonlarına birbirleri arasındaki öncelik ilişkileri sağlanarak atanırlar. Hat üzerinde sadece bir ürün işleniyorsa ve bu ürüne ait iş elemanlarının süreleri biliniyor ve sabit ise problem, basit montaj hattı dengeleme problemi olarak adlandırılır. Eğer hat üzerindeki iş istasyonu sayısı belli ise çevrim süresi azaltılmalıdır (BMHDP-1). BMHDP-1'i çözmek için Dal & Sınır algoritmaları, tabu araştırması, genetik algoritmalar vb. Araştırma tekniklerine dayalı sezgisel yöntemler geliştirilmektedir. Bu makalede, bu problemin çözümü için yeni bir yaklaşım olarak Petri ağları sunulmuştur. Petri ağları, kesikli olaylı sistemleri modelleyen, tasarlayan ve analiz eden bir tekniktir. Önerilen Petri ağlarına dayalı algoritma özellikle ulaşılabilirlik analizini kullanır. Algoritma MATLAB 6.0'da kodlanmıştır. Önerilen algoritmanın testi, Tonge'nin 70 iş elemanlı probleminde yapılmış, sonuçlar mevcut yedi yöntemin sonuçları ile karşılaştırılmıştır. Algoritma, 13 çevrim süresinin 12'sinde en iyi sonuçları vermiştir.

Assembly line balancing problem is an assignment problem. Tasks are assigned to work stations on the line by providing the precedence relations between tasks. When the only one product is produced on the line and its tasks have deterministic times, the problem is called simple assembly line balancing problem (SALBP). If number of work stations on the line is fixed, then cycle time should be reduced (SALBP-1). Heuristics based on branch and bound procedures, tabu search metaheuristics, genetic approaches, and etc. have been developed to solve SALBP-1. In this article, Petri nets are represented as a new approach to solve this problem. Petri nets are mathematical and graphical tool to model, design, and analyze discrete event systems. Proposed algorithm based on Petri nets uses the especially reachability analysis to determine available tasks and select task into them. Algorithm is coded in MATLAB 6.0. Proposed algorithm is tested on Tonge's 70-tasks problem, and then results are compared with existing seven methods' results. The algorithm gave best results for 12 of 13 cycle times.

ANAHTAR KELİMELER / *KEY WORDS*

Montaj hattı dengeleme problemi, Petri ağları, Montaj üretim, Ulaşılabilirlik analizi
Assembly line balancing problem, Petri nets, Assembly production, Reachability analysis

1. GİRİŞ

Montaj hattı, yüksek üretkenlik ve otomasyon sağlayan bir yığın üretim sistemi çeşididir. Bu avantajlarından yararlanmak için en önemli unsur, hat üzerindeki aynı işlem zamanına sahip iş istasyon sayısını ya da çevrim süresini en küçüklemektir. Bu amaçla iş elemanları, birbirleri ile öncelik ilişkilerine ve istasyon boş sürelerine göre iş istasyonlarına atanır. Bu problem literatürde montaj hattı dengeleme problemi (MHDP) olarak adlandırılır. Eğer montaj hattında, sadece bir ürüne ait işlemler yapılacaksa problem, tekli montaj hattı dengeleme problemi adını alırken, birden fazla ürünün hat üzerindeki iş istasyonlarında işlenmesi ile karmaşık montaj hattı dengeleme problemine dönüşür. İş istasyonlarına atanacak iş elemanlarının işlem süreleri bilgisi de montaj hattı dengeleme problemini yeni sınıflara ayırır. Eğer süreler biliniyorsa ve değişmiyorsa deterministik montaj hattı dengeleme problemi, eğer süreler rasgele değişiyorsa stokastik montaj hattı dengeleme problemi karşımıza çıkar. Deterministik süreli iş elemanlarına sahip tekli montaj hatlarındaki montaj hattı dengeleme problemi, basit montaj hattı dengeleme problemi (BMHDP) olarak tanımlanır. BMHDP'deki amaca göre 3 farklı durum söz konusudur. Eğer hat üzerindeki çevrim süresi biliniyorsa (yada veriliyorsa), amaç montaj hattındaki iş istasyonu sayısını en küçüklemektir. Bu durum birinci tip BMHDP'dir. Bilinen (yada verilen) iş istasyonu sayısına göre çevrim süresi en küçüklenmeye çalışılıyorsa ikinci tip BMHDP geçerlidir (Baybars, 1986a). Montaj hattı çıktı oranının yüksek olması isteniyorsa BMHDP-1 ile ilgilenilir. Eğer ürettim alanının hacimsel kısıtları söz konusuysa BMHDP-2 kullanılmalıdır. Son durumda, iş istasyon sayısı ve çevrim süresi değiştirilerek montaj hattının etkinliği en büyüklenmeye çalışılır (BMHDP-E). Hat etkinliği, hat üzerindeki iş istasyonlarının toplam sürelerinin ne kadarının iş elemanları işlemlerine ayrıldığını gösteren bir performans ölçütüdür. Yüksek hat etkinliği, iş istasyonlarındaki boş sürelerin az olduğunu gösterir. Eğer hat etkinliği %100 ise, hat üzerindeki hiç bir iş istasyonunda boş süre söz konusu değildir.

Salveson tarafından 1955 yılında matematiksel olarak ifade edilmesine değin MHDP, deneme yanılma ile çözülmekteydi. İzleyen yıllarda problemin doğrusal programlama, tam sayı programlama vb. ile ifade edilmesi ile çalışmalar yoğunlaşmıştır. Fakat problemimin boyutu büyüdükçe, yada bir başka deyişle hat üzerindeki iş elemanı sayısı arttıkça, çözüm uzayı da büyüdüğünden matematiksel programlama ile optimum sonucu bulmak imkansız hale gelmiştir. Bu aşamada optimum yada optimuma yakın sonuç üreten sezgisel yöntemler geliştirilmiştir. Bu makalenin üzerinde yoğunlaştığı BMHDP-1 için de literatürde bir çok sezgisel yöntem içeren çalışmalar bulunmaktadır. En çok bilinen yöntemlerden biri Baybars'ın geliştirmiş olduğu LBHA-1'dir (Line Balancing Heuristic Algorithm-type 1) (Baybars, 1986a). LBHA-1 problemin boyutunu küçülterek ve mümkünse problemi daha küçük problemlere dönüştürerek, BMHDP-1'i basitleştirerek çözer. BMHDP-1 için sezgisel yöntem geliştiren araştırmacıların yararlandığı metotlardan biri Dal ve sınır algoritmalarıdır. Johnson tarafından geliştirilen FABLE (Fast Algorithm for Balancing Lines Effectively), Nourie ve Venta'nın geliştirdiği OptPack, Hoffmann'ın yöntemi EUREKA, Scholl ve Klein tarafından geliştirilen SALOME (Simple Assembly Line balancing Optimization MEthod) ve Sprecher'in yöntemi AGSA (Adapted General Sequencing Algorithm) literatürde en çok bilinen dal ve sınır algoritmalarına dayalı sezgisel çözüm yöntemleridir (Johnson, 1988; Nourie ve Venta, 1991; Hoffmann, 1992; Scholl ve Klein, 1997; Sprecher, 1999). Genetik Yaklaşım yöntemi, BMHDP-1'e sezgisel çözüm yöntemi geliştiren araştırmacılar için yeni bir çalışma alanıdır. Rubinovitz ve Levitin , Kim vd., Bautista vd. ve Sabuncuoglu vd. çalışmalarında genetik yaklaşımdan yaralanmışlardır (Rubinovitz ve Levitin, 1995; Kim vd., 1996; Bautista vd., 2000; Sabuncuoğlu vd., 2001). Scholl ve Voss ve Chiang tabu araştırması yöntemini kullanarak, BMHDP-1 için sezgisel yöntemler geliştirmişlerdir (Scholl ve Hoss,

1996; Chiang, 1998). Burada değinilen sezgisel yöntemler ve ilgili başka çalışmalar hakkında daha fazla bilgi edinmek isteyen araştırmacılar, Baybars, Erel ve Sarin ve Kılınçcı'nın çalışmalarından faydalanabilirler (Baybars, 1986b; Erel ve Sarin, 1998; Kılınçlı, 2003).

Bu çalışmada, BMHDP-1 için geliştirilen Petri ağları tabanlı yeni bir çözüm yöntemi sunulacaktır. İzleyen bölümde Petri ağları ile ilgili genel bilgilere değinilecektir. Daha sonra geliştirilen algoritma açıklanarak işleyişi örnek problem üzerinde verilecek, son olarak da uygulama sonuçları sunulacaktır.

2. PETRİ AĞLARI

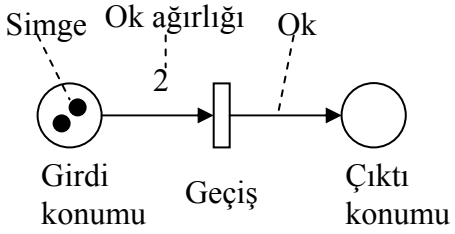
Bu bölümde özet olarak Petri ağları ile ilgili genel bilgilere değinilecektir. Özelikle izleyen bölümde sunulacak olan algoritmada kullanılan özellikler ayrıntılı olarak verilecektir.

Petri ağları, Carl Adam Petri tarafından geliştirilen ve onun adıyla anılan, kesikli olaylı sistemlerin modellenmesinde, analizinde ve tasarımında kullanılan grafiksel ve matematiksel bir tekniktir. Petri ağları, eş zamanlı, eş zamansız, paralel, deterministik, stokastik, vb. işlemlerin yer aldığı sistemlerle çalışma imkanı sağlar. Ayrıca modelde yer alan semboller yardımıyla, sistemdeki olaylar için bir benzetim imkanı sağlar. Matematiksel bir araç olarak, sistemin davranışlarını açıklayan durum denklemlerinin elde edilmesine, cebirsel sonuçların bulunmasına ve diğer matematiksel modellerin geliştirilmesine yardımcı olur (Murata, 1989).

Temeli Carl Adam Petri'nin doktora çalışmasına dayanan Petri ağları, yazılım sistemlerinden esnek imalat sistemlerine, endüstriyel kontrol sistemlerinden çok işlemcili hafıza sistemlerine, veri akışı işleyen sistemlerden programlanabilir mantık kontrol devrelerine kadar geniş bir uygulama alanına sahiptir. Benzer şekilde, eş zamanlı, eş zamansız, paralel, vb. işlemler içeren üretim sistemlerinde de yaygın olarak kullanılmaktadır. Üretim sistemlerindeki modelleme, sıralama, ölü noktaların kontrolü ve giderilmesi, kontrol ve performans değerlendirme ile ilgili bir çok uygulama, Petri ağları ile yapılabilmektedir. Montaj hatlı üretim sistemleri, özellikle sistemin modellenmesi, tasarımı ve kontrolü, hat üzerindeki işlerin veya iş elemanlarının sıralanması açısından popüler bir uygulama alanıdır. Teng ve Zhang, Ramaswamy vd., Adamou vd. ve Zha vd., Petri ağları ile üretim sisteminin modellemesi ve kontrolü üzerinde çalışmışlardır (Teng ve Zhang, 1993; Ramaswamy vd., 1997; Adamou vd., 1998; Zha vd., 1998). Cao ve Sanderson, McCarragher, D'Souza ve Khator, Zimmermann ve Hommel, Yeung ve Moore, Zha ve Zha vd. Petri ağları ile robotlu montaj hattı sistemlerini ve sistemdeki robotların işlemlerini modellemişlerdir (Cao ve Sanderson, 1994; McCarragher, 1994; D'Souza ve Khator, 1997; Zimmermann ve Hommel, 1999; Yeung ve Moore, 2000; Zha, 2000; Zha vd., 2001). Montaj işlem planlarını ve işlem sırasını belirlemek için Petri ağları, Thomas vd., Jeng vd., Yee ve Ventura ve Frey'in çalışmalarında kullanılmıştır (Thomas vd., 1996; Jeng vd., 1999; Yee ve Ventura, 1999; Frey, 2000). Moore vd., Zussman ve Zhou, Zha ve Lim ve Tang vd. Petri ağları yardımıyla montaj/demontaj işlem planlarını belirleyen yöntemler geliştirmişlerdir (Moore vd., 1998; Moore vd., 2001; Zussman ve Zhou, 1999; Zha ve Lim, 2000; Tang vd., 2001). Burada değinilen çalışmalar ve üretim sistemlerinde yapılan diğer Petri ağı çalışmaları hakkında bilgi almak isteyen araştırmacılar Moore ve Gupta ve Kılınçcı'nın çalışmalarından yararlanabilir (Moore ve Gupta, 1996; Kılınçlı, 2003). Yukarıda da değinildiği gibi, literatürdeki bir çok Petri ağı çalışması montaj hattı sistemlerinin çeşitli problemlerine odaklansa da, Petri ağları ile yapılmış bir montaj hattı dengeleme problemine rastlanılmamıştır.

Önerilen Petri ağ tabanlı algoritmayı açıklamadan önce, Petri ağlarla ilgili gerekli bilgilerin verilmesi gerekmektedir. Bir Petri ağı dört bileşenden oluşur. Bunlar, grafiksel gösterimde çember biçiminde ifade edilen konum (place), dikdörtgen kutu ya da çubuk şeklinde gösterilen geçiş (transition), konum ve geçişi birbirine bağlayan yönlendirilmiş ok

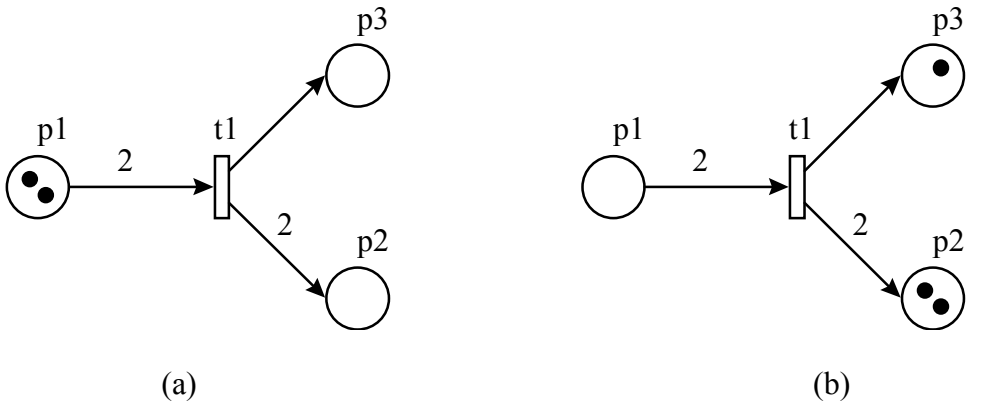
(arc), ve konumlar içinde nokta biçiminde gösterilen simge (token)'dir (Zurawski ve Zhou, 1994). Bir konum bir başka konuma, ya da bir geçiş bir başka geçişe doğrudan bağlı olamaz. Geçiş ve konumu birbirine bağlayan oklar üzerindeki değerler, geçiş ve konum arasındaki paralel bağlantı sayısını gösterir. Herhangi bir geçiş iki çeşit konuma sahiptir. Kendinden önceki konumlar girdi konumu, kendinden sonraki konumlar çıktı konumu olarak adlandırılır. Petri ağı bileşenleri Şekil 1'de gösterilmiştir.



Şekil 1. Petri ağı bileşenleri

Simgelerin hareketi, Petri ağları ile dinamik bir çalışma imkanı sağlar. Simgelerin konumlar arasında hareket edebilmesi için, ilk anda yeterli sayıda simgenin gerekli konum veya konumlara yerleştirilmesi gerekir. Başlangıç işaretlemesi, ilk anda hangi konumda ne kadar simge olduğu bilgisini verir. Bu sağlandıktan sonra, simge konumlar arasında hareket eder. Bunun için iki konum arasındaki geçişin oluşması gereklidir. Geçişin oluşmasından önce de geçerlilik kuralının yerine gelmesi gerekir. Bir geçişin geçerli olması için, girdi konumu ile geçiş arasındaki paralel ok sayısı kadar simgenin geçişe ait girdi konumunda yer alması şarttır. Bu şart sağlandıktan sonra geçiş geçerli olur. Geçiş gerçekleştikten sonra, girdi konumu ile geçiş arasındaki paralel ok sayısı kadar simge girdi konumundan silinir. Geçiş ile çıktı konumu arasındaki paralel ok sayısı kadar simge çıktı konumuna yerleştirilir. Geçerlilik kuralı ve oluşum kuralı ile ilgili gösterimler Şekil 2'de verilmiştir (Zurawski ve Zhou, 1994).

Petri ağları bazı özellikler taşır. Bunlardan literatürde en çok bilinenleri, ulaşılabilirlik (reachability), canlılık (liveness), geri dönebilirlik (reversibility) ve sınırlandırılmışlık (boundedness) özellikleridir. Canlılık özelliğinden, özellikle sistemlerin ölü noktalarının belirlenmesinde yararlanılır. Geri dönebilirlik özelliği, belirli bir işlem yapıldıktan sonra, sistemin işlem öncesi durumuna gelmesini sağlar. Sınırlandırılmışlık özelliği de, sistemdeki kapasite sınırlamalarının modelde gösterilmesine imkan verir. Ulaşılabilirlik özelliği, sistemin işleyişinin istenen konumlara gelip gelmediğini araştırır. Bu özelliğe dayalı olarak geliştirilen ulaşılabilirlik analizi yardımıyla, sistemin başlangıç durumundan itibaren gelebileceği tüm olası durumlar belirlenir. Bu çalışmada sunulacak algoritma, Petri ağlarının ulaşılabilirlik analizinden yararlanmaktadır.



Şekil 2. (a) Geçiş t_1 geçerli, (b) Geçerli geçiş t_1 oluştu

Petri ağları ile ilgili bir başka bilinmesi gereken konu, ilişki (incidence) matrisidir. İlişki matrisi, Petri ağı modelindeki konum ve geçişlerin bağlantı durumunu ve bağlantıyı sağlayan okların ağırlık bilgilerini içeren bir tamsayı matrisidir. Matrisin kolon sayısı, modeldeki konum sayısına, satır sayısı da modeldeki geçiş sayısına eşittir. İlişki matrisinde yer alan hücrelerdeki sayılar, hücrenin yer aldığı satırdaki geçişle ilgili bilgiler içerir. Örneğin *i*. satır ve *j*. kolondaki değer, *i*. geçiş ile ilgili bilgiler taşır. Buradaki değer, *i*. geçiş ile çıktı konumları arasındaki paralel ok sayısının, *i*. geçiş ile girdi konumları arasındaki paralel ok sayısının farkına eşittir. Bir başka deyişle, *i*. geçişin oluşması sonucunda, çıktı konumlarına yerleştirilecek simge sayısından girdi konumlarından silinecek simge sayısının çıkarılması ile bulunan değerdir. Ayrıca ilişki matrisi, simgelerin hareketi sonucu (ya da herhangi bir geçişin geçerli olması sonucu) değişecek olan Petri ağı modelindeki simge durumunu belirlememize imkan verir. Simge durumları, tek satır ve konum sayısı kadar kolona sahip bir vektörde gösterilir. Her konumdaki simge sayısı ilgili kolonda belirtilir. Eğer mevcut duruma göre *i*. geçiş geçerli olacaksa, geçişten sonra modelde oluşacak yeni simge durumu, mevcut simge durumunu gösteren kolon vektör ile ilişki matrisinin *i*. satırının toplanmasıyla bulunur. Yeni simge durumu, yeni geçiş imkanları sağlar. Böylece simgeler model üzerinde, konumlar arasındaki hareketlerini sürdürürler.

Bu bölümde Petri ağları ile ilgili özet bilgiler sunulmuştur. Ayrıntılı bilgiye ulaşmak isteyen araştırmacılar Murata, Zurawski ve Zhou, Desrochers ve Al-Jaar, Zhou ve Venkatesh'in çalışmalarından yararlanabilir (Murata, 1989; Zurawski ve Zhou, 1994; Desrochers ve Al-Jaar, 1995; Zhou ve Venkatesh, 1999). Bu bölümde verilen bilgiler ışığında izleyen bölümde, BMHDP-1 için geliştirilen Petri ağ tabanlı algoritma sunulacaktır.

3. PETRİ AĞLARI İLE BMHDP-1'İN ÇÖZÜLMESİ

Geliştirilen algoritma açıklanmadan önce, Petri ağlarının BMHDP-1 çözümünde kullanılmasının nedenlerini belirtmek gerekir. Petri ağları, paralel veya eşanlı olaylar içeren ve grafiksel olarak bir akış diyagramı şeklinde gösterilebilen tüm sistemlerde kullanılabilmektedir (Murata, 1989). Bir montaj hattı sistemi, paralel ve eşanlı işlemler içermektedir. Ayrıca montaj hattında yapılacak işe ait iş elemanlarının öncelik ilişkisi, bir akış diyagramı olarak gösterilebilir. Yine önceki bölümde de değinildiği gibi, simgelerin konumlar arasındaki hareketi, dinamik çalışma imkanı sağlar. Bu dinamik çalışma da, iş istasyonlarındaki boş sürelere ve öncelik ilişkilerine göre, iş istasyonlarına atanmaya aday iş elemanlarının belirlenmesine ve içlerinden birinin seçimine olanak verir. Petri ağları kullanarak geliştirilen algoritmanın temeli bu iki noktaya dayanmaktadır.

Algoritmayı uygulamadan önce, montaj hattının Petri ağı modeli oluşturulmalıdır. Petri ağı modeli, montaj hattındaki iş elemanlarının öncelik ilişkilerini yansıtan bir modeldir. İş elemanları, modeldeki geçişlerde tanımlanır. Dolayısıyla Petri ağı modelindeki geçiş sayısı, problemdeki iş elemanı sayısına eşittir. Ayrıca iş elemanı sıra numarası ile modeldeki geçiş sıra numarası aynı olmalıdır. Yani 3. iş elemanı 3. geçişte ifade edilmelidir. Böylece algoritma sonucunun yorumlanması kolay olur. İlişki matrisi ve başlangıç simge durumunu gösteren vektör, oluşturulan bu Petri ağı modelinden elde edilir. Geliştirilen algoritma verilmeden önce, algoritmada kullanılan semboller açıklanacaktır.

A: İlişki matrisi (*tn* x *pn*)
a: Aynı simge dağılımında, birden fazla iş elemanı atanmasını önleyen kontrol değişkeni
B: *A*'nın *j*. satırı (geçerli geçişin numarası *j* ise)
Bz: Uygun her iş elemanı için belirlenen boş zaman
C: Çevrim süresi

LE: Montaj hattının hat etkinliği
M: Başlangıç simge durumun gösteren vektör (1 x *pn*)
minBz: İş istasyonunun minimum boş zamanı
MN: Seçilen iş elemanına göre, *MY* matrisinden seçilen vektör (1 x *pn*)
MT: Herhangi bir geçişin girdi konumundaki toplam simge sayısını içeren
MY: Geçerli tüm geçişler için oluşturulan *B* vektörlerini içeren matris
O: Tüm *Q* vektörlerini içeren matris
Ot: Geçerli geçişin süresi
Oz: Geçerli geçişin numarası
pn: Petri ağındaki kolon sayısı (ya da *A* matrisinin kolon sayısı)
Q: Geçerli geçişin numarası ve süresinin içeren vektör
SN: İş istasyonuna atanan iş elemanı bilgilerini içeren vektör. İş istasyonu numarası (*w*), atanan iş elemanının numarası (*Oz*), atanan iş elamanının süresi (*Ot*) ve iş istasyonu boş süresi (*minBz*) bilgilerini içerir
SON: *SN* vektörlerinden oluşan sonuç matrisi
SumOt: Atanan iş elemanlarının toplam süresi
Sumt: İş elemanlarının toplam süresi
Sw: İş istasyonuna atanmış iş elemanlarının toplam süresi
t: İş elemanlarının sürelerini içeren vektör (1 x *tn*)
tn: Petri ağındaki geçiş sayısı (ya da *A*'nın satır sayısı)
vr: Atanabilir iş elemanlarının sayısı (ya da *O* matrisinin satır sayısı)
w: İş istasyonu numarası

Geliştirilen algoritmanın adımları aşağıda verilmiştir.

Adım 1. *A, M, t ve C'yi gir*
A'yı kullanarak pn ve tn'i belirle, Sumt'yi bul
Başlangıç değerleri: w=1, Sw=0, Bz=C, MinBz=C, SumOt=0, MT=[], MY=[], MN=[], Ot=0, Oz=0, O=[], SN=[], SON=[], a=0

Adım 2. *Atanmamış iş elemanı kalıncaya dek (Sumt>SumOt)*

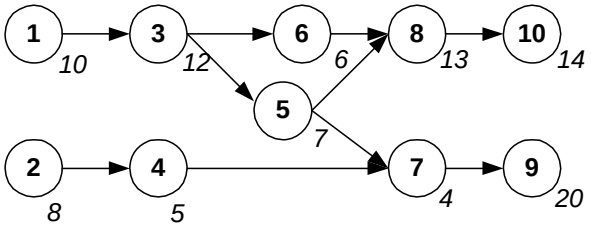
a. *Geçerli geçişleri (atanabilir iş elemanlarını) belirle*
j=1, 2, ... ,tn için
i. *i=1,2,...,pn için*
eğer A(j,i)=-1
y=M(1,i),
y'yi MT'ye ekleyerek MT'yi güncelle
ii. *eğer MT'deki tüm değerler 1'e eşitse*
B=A(j,:)
B'yi MY'ye ekleyerek MY'yi güncelle
Q=[j t(j)]
Q'yu O'ya ekleyerek O'yu güncelle
iii. *MT=[]*

b. *Minimum iş istasyonu boş süresini veren iş elemanını belirleyen ve bu iş elemanını iş istasyonu atayan işlemler*
vr'yi belirle
v=1,2,...vr için
i. *eğer O(v,2)=C-Sw ve a≠1*
minBz=0, MN=MY(v,:), Oz=O(v,1), Ot=O(v,2)
SN=[w Oz Ot minBz], a=1
ii. *eğer O(v,2) < C-Sw ve a≠1*

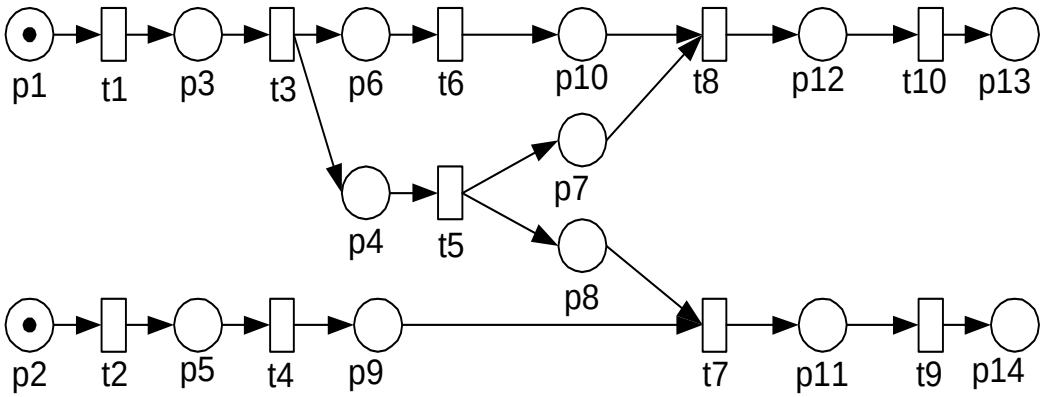
Bz=C-Sw-O(v,2)
aa. *eğer Bz<minBz ve Bz>0*
minBz=Bz, MN=MY(v,:), Oz=O(v,1), Ot=O(v,2),
SN=[w Oz Ot minBz]
c. *SN'yi SON'a ekleyerek SON'u güncelle*
tim=SON(:,3)
SumOt=∑tim; Sw=C-minBz
i. *eğer Oz=[]*
w=w+1; Bz=C; minBz=C; Sw=0; MN sıfır vektör (1xpn)
ii. *M=M+MN; MN=[]; MY=[]; O=[]; Oz=0; Ot=0; B=[]; SN=[]*
Adım 3. *Sonuçları göster (SON)*
Hat etkinliğini hesapla, LE=Sumt/(wxC)
Hat etkinliğini göster (LE)

Algoritmanın ilk adımı veri girişi aşamasıdır. Petri ağı modelinden elde edilen ilişki matrisi, başlangıç simge durumunu gösteren vektör ile iş elemanları süreleri ve çevrim süresi algoritmanın girdileridir. Herhangi bir iş elemanı atanmayan iş istasyonunun boş süresi çevrim süresine eşittir. Bir başka deyişle, iş istasyonuna atanan iş elemanlarının sürelerinin toplamı çevrim süresini aşamaz. Ayrıca bu adımda, sonraki adımlarda farklı değerler alacak olan değişkenlerin başlangıç değerleri belirlenir. 2. adım, atanmaya uygun iş elemanlarının belirlenip, bu adaylar içinden en küçük iş istasyonu boş zamanını veren iş elemanının geçerli iş istasyonuna atanması işlemlerini kapsar. *Adım 2.a*'da ilişki matrisi ve simge durumunu gösteren vektör yardımıyla geçerli olabilecek geçişler belirlenir. Geçerli olabilecek geçişler, (varsa) önceki iş elemanları iş istasyonlarına atanarak öncelik şartları sağlanmış iş elemanlarını ifade eder. *Adım 2.b*'de, *Adım 2.a*'da belirlenen öncelik şartı sağlanmış ve süreleri iş istasyonu boş süresinden küçük ya da ona eşit olan iş elemanları sıra ile araştırılarak, iş istasyonuna atanmaları durumunda ortaya çıkacak olan iş istasyonu boş süreleri hesaplanır. En küçük iş istasyon boş süresini veren iş elemanı, geçerli iş istasyonuna atanır ve atama bilgilerini içeren *SN* vektörü oluşturulur. Tüm bu işlemler *Adım 2.b.ii*'de yapılır. Eğer aday iş elemanları içinde, işlem süresi iş istasyonu boş süresine eşit iş elemanı varsa, diğer adaylar araştırılmadan bu iş elemanı geçerli iş istasyonuna atanır ve atama bilgileri oluşturulur (*Adım 2.b.i*). Oluşturulan atama bilgileri (*SN*), sonuç matrisine eklenir ve iş istasyonu boş süresi güncellenir (*Adım 2.c*). Eğer herhangi bir atama işlemi yapılmadıysa (iş istasyonu boş süresi, tüm aday iş elemanlarının sürelerinden küçükse), algoritma yeni bir iş istasyonu açar. Herhangi bir atama yapılmadığından *SN* oluşturulamaz ve simge durumunu gösteren vektörde de bir değişiklik yapılmaz (*Adım 2.c.i*). Atama olsa da olmasa da, ilgili değişkenlerin değerleri yeni atama süreci için güncellenir (*Adım 2.c.ii*). *Adım 2*, tüm iş elemanları iş istasyonlarına atanana dek sürer. Tüm atama işlemleri tamamladıktan sonra, son adımda sonuçlar gösterilir (*SON* ve *LE*). Sonuçlar, hangi iş elemanının hangi iş istasyonuna atandığını, iş istasyonlarının her atama sonucundaki toplam işlem ve boş sürelerini ve atamalar sonucunda montaj hattında oluşan hat etkinliğini içerir.

Algoritmanın işleyişini anlatmak için Şekil 3'de 10 iş elemanlı bir montaj hattının öncelik ilişkileri diyagramı verilmiştir. Daire içlerinde iş elemanlarının numaraları, dairelerin sağ alt köşelerinde ise ilgili iş elemanın süresi gösterilmiştir. Algoritmayı kullanmadan önce yapılacak ilk işlem, öncelik diyagramının Petri ağı modelini oluşturmaktır. İş elemanlarının geçişlerde gösterildiği Petri ağı modeli Şekil 4'de verilmiştir. Petri ağı modelinden oluşturan ilişkiler matrisi, başlangıç simge durumunu gösteren vektör, iş elemanları sürelerini içeren vektör ve çevrim süresi, algoritmanın girdileri olarak Şekil 5'de sunulmuştur.



Şekil 3. İş elemanlarının öncelik ilişkilerini gösteren diyagram



Şekil 4. Öncelik diyagramının Petri ağı modeli

$$A = \begin{array}{c} \begin{matrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13} & p_{14} \end{matrix} \\ \begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \\ t_{10} \end{matrix}$$

$$\begin{matrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13} & p_{14} \end{matrix}$$

$$M = [1 \;\; 1 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0], \quad \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 & t_{10} \end{matrix}$$

$$t = [10 \;\; 8 \;\; 12 \;\; 5 \;\; 7 \;\; 6 \;\; 4 \;\; 13 \;\; 20 \;\; 14], \quad C=35$$

Şekil 5. Algoritmanın girdileri

İlişki matrisi, Peri ağı modelindeki konum ve geçişleri arasındaki ilişkinin varlığını ve ilişki varsa, konum ve geçiş arasındaki okun ağırlık derecesi bilgilerini içerir. Şekil 4'deki Petri ağı modelinde yer alan 1.geçişin (t1) bir girdi konumu (p1) ve bir çıktı konumu (p3) vardır. Bu ilişkiler, ilişki matrisinin 1.satırında (1.geçiş bilgilerinin olduğu satır) gösterilmiştir. 1.kolondaki "–1" değeri, p1'in 1.geçişin girdi konumu, 3.kolondaki "1" değeri, p3'ün 1.geçişin çıktı konumu olduğunu gösterir. Diğer ilişkiler de benzer şekilde gösterilmiştir. Başlangıç durumunda Petri ağındaki simge dağılımı *M* vektöründe verilmiştir. İlk anda sadece 1. ve 2.konumlarda birer adet simge bulunmaktadır, bu da M vektörünün 1. ve 2.kolonlarında "1" değerleri ile gösterilmiştir. İş elemanları süreleri ya da geçişlerin işlem süreleri *t* vektöründe verilmiştir. Örnek problem için çevrim süresi (*C*) 35 olarak belirlenmiştir.

Algoritmanın işleyişi ve ilgili değişkenlerin aldıkları değerler Çizelge 1'de verilmiştir. İlk aşamada, geçerli olabilecek geçişler 1 ve 2.geçişlerdir. Bu adaylar içinden en küçük boş iş istasyon süresini veren iş elemanı 1.iş elemanıdır ve ilk iş istasyonuna atanır. 2.aşamada, yeni simge durumuna göre adaylar 2 ve 3.geçişler olarak belirlenir. En küçük boş süreyi veren 2.iş elemanı ilk istasyona atanır. 3.aşamadaki adaylar arasından (2, 5 ve 6.geçişler) benzer mantıkla 2.iş elemanı seçilerek atama işlemi yapılır. 4.aşamada belirlenen adaylar arasından (4, 5 ve 6.geçişler), iş istasyonu boş süresine eşit süreye sahip 4.iş elemanı, 1.iş istasyonuna atanır. 5.aşama sonunda, belirlenen adayların işlem süreleri iş istasyonu boş süresinden küçük olduğu için 2.iş istasyonu açılır. Algoritma sırasıyla, 6.aşamada 5 ve 6.iş elemanları arasından 5.'yi, 7.aşamada 6 ve 7.iş elemanları arasından 6.'yı, 8.aşamada 7 ve 8.iş elemanları arasından 8.'yi ve son olarak 9.aşamada 7 ve 10.iş elemanları arasından 7.iş elemanını seçerek 2.iş istasyonuna atar. 10.aşamada belirlenen her iki iş elemanının (9 ve 10) süreleri, 2.iş istasyonunun boş süresinden küçük olduğu için algoritma 3.iş istasyonu açar. Son iki aşamada da sırasıyla 9 ve 10.iş elemanları 3.iş istasyonuna atanır. Böylece algoritmanın *Adım 2*'si tamamlanmış olur. Her atama işlemi sonunda güncellenen sonuçlar (*SON*) ve *Adım 3*'e belirlen hat etkinliği (*LE*) aşağıda verilmiştir.

w	tn	t	Bz
1	1	10	25
1	3	12	13
1	2	10	5
1	4	5	0
2	5	7	28
2	6	6	22
2	8	13	9
2	7	4	5
3	9	20	15
3	10	14	1

Hat etkinliği (*LE*) = 0.9429

Çizelge 1. Örnek problemin çözüm aşamaları

Aş.	*M*	*MY*	*O*	*w*	*Sw*	*Bz*	*minBz*	*SN*	*MN*	*SumOt*
0	[1 1 0 0 0 0 0 0 0 0 0 0 0 0 0]	[]	[]	1	0	35	35	[]	[]	0
1	[0 1 1 0 0 0 0 0 0 0 0 0 0 0 0] [1 0 0 1 0 0 0 0 0 0 0 0 0 0 0]	[–1 0 1 0 0 0 0 0 0 0 0 0 0 0 0] [0 –1 0 0 1 0 0 0 0 0 0 0 0 0 0]	[1 10] [2 8]	1	0 10	25 (1. iş el.) 27 (2. iş el.)	25 25 25	[1 1 10 25]	[-1 0 1 0 0 0 0 0 0 0 0 0 0 0 0]	10
2	[0 1 0 1 0 1 0 0 0 0 0 0 0 0 0] [0 0 1 1 0 1 0 0 0 0 0 0 0 0 0]	[0 –1 0 0 1 0 0 0 0 0 0 0 0 0 0] [0 0 –1 1 0 1 0 0 0 0 0 0 0 0 0]	[2 8] [3 12]	1	22	17 (2. iş el.) 13 (3. iş el.)	17 13 13	[1 3 12 13]	[0 0 -1 1 0 1 0 0 0 0 0 0 0 0 0]	22
3	[0 0 0 1 1 1 0 0 0 0 0 0 0 0 0] [0 0 0 1 0 1 0 0 0 0 0 0 0 0 0] [0 0 0 1 0 1 0 0 0 0 0 0 0 0 0]	[0 –1 0 0 1 0 0 0 0 0 0 0 0 0 0] [0 0 0 –1 0 0 1 1 0 0 0 0 0 0 0] [0 0 0 0 0 –1 0 0 0 1 0 0 0 0 0]	[2 8] [5 7] [6 6]	1	30	5 (2. iş el.) 6 (5. iş el.) 7 (6. iş el.)	5 5 5 5	[1 2 10 5]	0 -1 0 0 1 0 0 0 0 0 0 0 0 0 0]	30
4	[0 0 0 1 0 1 0 0 1 0 0 0 0 0 0] [0 0 0 1 0 1 0 0 1 0 0 0 0 0 0] [0 0 0 1 0 1 0 0 1 0 0 0 0 0 0]	[0 0 0 0 –1 0 0 0 1 0 0 0 0 0 0] [0 0 0 –1 0 0 1 1 0 0 0 0 0 0 0] [0 0 0 0 0 –1 0 0 0 1 0 0 0 0 0]	[4 5] [5 7] [6 6]	1	35	0 (4. iş el.)	0 0	[1 4 5 0]	[0 0 0 0 -1 0 0 0 1 0 0 0 0 0 0]	35
5	[0 0 0 1 0 1 0 0 1 0 0 0 0 0 0] [0 0 0 1 0 1 0 0 1 0 0 0 0 0 0]	[0 0 0 –1 0 0 1 1 0 0 0 0 0 0 0] [0 0 0 0 0 –1 0 0 0 1 0 0 0 0 0]	[5 7] [6 6]	1 2	0	-7 (5. iş el.) -6 (6. iş el.)	35	[]	[]	35
6	[0 0 0 0 0 1 1 1 1 0 0 0 0 0 0] [0 0 0 1 0 1 0 0 1 0 0 0 0 0 0]	[0 0 0 –1 0 0 1 1 0 0 0 0 0 0 0] [0 0 0 0 0 –1 0 0 0 1 0 0 0 0 0]	[5 7] [6 6]	2	7	28 (5. iş el.) 29 (6. iş el.)	28 28 28	[2 5 7 28]	[0 0 0 -1 0 0 1 1 0 0 0 0 0 0 0]	42

Çizelge 1. Örnek problemin çözüm aşamaları (devamı)

Aş.	***M***	***MY***	***O***	***w***	***Sw***	***Bz***	***minBz***	***SN***	***MN***	***SumOt***
7	[0 0 0 0 0 0 1 1 1 1 0 0 0 0]	[0 0 0 0 0 –1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 –1 –1 0 1 0 0 0]	[6 6 7 4]	2	13	22 (6. iş el.) 24 (7. iş el.)	22 22 22	[2 6 6 22]	[0 0 0 0 0 -1 0 0 0 1 0 0 0 0]	48
8	[0 0 0 0 0 0 0 1 1 0 0 1 0 0]	[0 0 0 0 0 0 0 –1 –1 0 1 0 0 0 0 0 0 0 0 0 –1 0 0 –1 0 1 0 0]	[7 4 8 13]	2	26	15 (7. iş el.) 9 (8. iş el.)	15 9 9	[2 8 13 9]	[0 0 0 0 0 0 -1 0 0 -1 0 1 0 0]	61
9	[0 0 0 0 0 0 0 0 0 0 0 1 1 0 0]	[0 0 0 0 0 0 0 –1 –1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 –1 1 0]	[7 4 10 14]	2	30	5 (7. iş el.) -5 (10. iş el.)	5 5	[2 7 4 5]	[0 0 0 0 0 0 0 -1 -1 0 1 0 0 0]	65
10	[0 0 0 0 0 0 0 0 0 0 0 1 1 0 0]	[0 0 0 0 0 0 0 0 0 0 0 –1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 –1 1 0]	[9 20 10 14]	2 3	0	-15 (9. iş el.) -9 (10. iş el.)	35	[]	[]	65
11	[0 0 0 0 0 0 0 0 0 0 0 0 1 0 1]	[0 0 0 0 0 0 0 0 0 0 0 –1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 –1 1 0]	[9 20 10 14]	3	20	15 (9. iş el.) 21 (10. iş el.)	15 15 15	[3 9 20 15]	[0 0 0 0 0 0 0 0 0 0 0 -1 0 0 1]	85
12	[0 0 0 0 0 0 0 0 0 0 0 0 0 1 1]	[0 0 0 0 0 0 0 0 0 0 0 0 –1 1 0]	[10 14]	3	34	1 (10. iş el.)	1 1	[3 10 14 1]	[0 0 0 0 0 0 0 0 0 0 0 0 -1 1 0]	99

4. UYGULAMA

Önerilen algoritmanın etkinliğini araştırmak için Tonge'nin 70 iş elemanlı problemi test problemi olarak seçilmiştir. Aynı problem Baybars'ın ve Sabuncuoğlu vd.'nin çalışmalarında da kullanılmıştır (Baybars, 1986b; Sabuncuoğlu vd., 2000). Baybars, Tonge'nin problemini 13 farklı çevrim süresi için, kendi geliştirdiği algoritma (LBHA-1) ile çözmüş, sonuçları Moodie ve Young'ın 1965'deki çalışmasının sonuçları ile, Tonge'nin yine 1965 yılındaki çalışmasının sonuçları ile ve Nevins'in 1972 yılındaki çalışmasının sonuçları ile karşılaştırmıştır (Baybars, 1986b). Tonge çalışmasında üç sezgisel yöntem kullanmıştır, bunlar; en çok hemen takip eden (MIF), rasgele seçim (RC) ve yanlı olasılıklı seçim (BPC) yöntemleridir. LBHA-1 5 aşamadan oluşan bir algoritmadır. İlk aşamada, belirli işler elenerek problem boyutu küçültülür. Atama kararlarının belirlendiği ikinci aşamadan sonra, problem daha küçük boyutlu problemlere dönüştürülür. 4.aşamada aynı iş istasyonuna atanacak iş elemanları bir araya toplandıktan sonra, son aşamada değişik sezgisel yöntemler kullanılarak çözüme ulaşılır. Moodie ve Young'ın geliştirdiği algoritma en büyük süreli iş elemanından başlayarak atamaları yapar. Tonge'nin MIF algoritması, atama sürecinde, en çok öncelik ilişkisi kısıtının kalkmasını sağlayan iş elemanını belirleyerek atama yapar. Bir diğer Tonge algoritması olan RC rasgele seçimler yaparak atama sürecini tamamlar. Son Tonge algoritması BPC, seçme işleminde yanlı olasılık ile karar verir. Nevin'in algoritması bir optimizasyon yöntemidir, bir üst sınır kullanılarak problem çözülür. Eğer üst sınırda belirtilen iş istasyonu sayısı sonuç olarak bulunmuşsa, bu sayı bir azaltılarak problem yeniden çözülür. Çözümsüz olan iş istasyonu sayısında çözüm süreci kesilir ve minimum iş istasyonu, çözümsüz aşamadaki iş istasyonu sayısına bir eklenerek elde edilmiş olur. Sabuncuoğlu vd. algoritması genetik yaklaşım mantığına kullanarak, her bir alternatif atama sırasını bir birey olarak kabul etmiş, bu atama sıralarındaki (bireylerdeki) en iyi atamaları (üstün özellikleri) bir sonraki atama sürecine (bir sonraki nesile) taşıyarak problemi çözer (Sabuncuoğlu vd., 2000). Sabuncuoğlu vd. yaptıkları çalışmada Baybars'ın bu karşılaştırma sonuçlarını kullanarak, kendi geliştirdikleri Genetik Algoritmaya dayalı yöntemin sonuçları ile bir kıyaslama yapmışlardır (Sabuncuoğlu vd., 2000). Sabuncuoğlu vd.'nin kullandıkları karşılaştırma tablosu, bu makalede önerilen algoritmanın etkinliğini test etmek için kullanılacaktır. Algoritma MATLAB 6.0'da kodlanarak yazılmış, program yardımıyla Tonge'nin 70 iş elemanlı problemi 13 farklı çevrim süresi için çözülmüştür. Diğer çalışmaların ve önerilen algoritmanın sonuçları (toplam iş istasyonu sayısı) Çizelge 2'de verilmiştir.

Çizelge 2. Karşılaştırma sonuçları

Çevrim Süresi	Moodie ve Young	Tonge MIF	Tonge RC	Tonge BPC	Nevins	Baybars	GA	ÖPA
83	48	50	50	49	47	47	48	47
86	47	47	48	47	46	46	46	45
89	44	45	46	44	43	43	43	42
92	43	43	44	43	42	42	42	41
95	42	43	43	41	40	40	40	40
170	24	24	24	23	23	23	23	23
173	24	24	24	23	22	23	23	23
176	22	24	23	22	22	23	22	22
179	22	23	23	21	21	22	22	21
182	22	23	22	21	21	22	22	21
346	11	11	12	11	11	11	11	11
349	11	11	11	11	11	11	11	11
364	11	11	11	11	11	11	11	10

Çizelge 1'deki ilk kolon Tonge'nin problemi için kullanılan çevrim sürelerini verir. Test için 83'den 364'e kadar 13 farklı çevrim süresi kullanılmıştır. İkinci kolon Moodie ve Young'ın sonuçlarını (MY), üç, dört ve beşinci kolonlar Tonge'nin sonuçlarını, altıncı kolon Nevins'in sonuçlarını, yedinci kolon Baybars'ın sonuçlarını, sekizinci kolon Sabuncuoglu vd.'nin sonuçlarını (GA) ve son kolon bu makalede önerilen Petri ağ tabanlı algoritmanın (ÖPA) sonuçlarını verir. ÖPA, çevrim süresi 176, 346 ve 349 için MY ile aynı sonuçları verirken, diğer 10 çevrim süresinde MY'den daha iyi sonuçlar vermiştir. MIF ile karşılaştırıldığında, ÖPA 346 ve 349 çevrim süreleri için MIF ile aynı sonuçları verirken diğer 11 çevrim süresinde daha iyi sonuçlar sağlamıştır. RC ile kıyaslamada, ÖPA sadece çevrim süresi 349'da benzer sonuç verirken diğer 12 çevrim süresinde RC'ye üstünlük sağlamıştır. BPC ile kıyaslamada ise, ÖPA 7 çevrim süresi için aynı sonucu verirken geriye kalan 6 çevrim süresinde BPC'den daha küçük iş istasyon sayıları bulmuştur. Nevins'in yöntemi çevrim süresi 173 olduğunda ÖPA'dan daha iyi sonuç vermektedir. Bununla birlikte ÖPA 4 çevrim süresi için Nevins'in yönteminden daha iyi sonuçlar bulmaktadır. Geri kalan 8 çevrim süresinde ÖPA ve Nevins'in yöntemi benzer sonuçlar üretir. Baybars'ın metoduyla ÖPA karşılaştırıldığında, ÖPA'nın 7 çevrim süresinde daha iyi sonuçlar verdiği görülmektedir. Diğer 6 çevrim süresi için Baybars'ın ve ÖPA'nın sonuçları aynıdır. Son olarak ÖPA, GA ile karşılaştırılmıştır. Yine 7 çevrim süresi için ÖPA daha iyi sonuçlar üretirken, geri kalan 6 çevrim süresinde GA ve ÖPA benzer sonuçlar bulmuşlardır. Sonuç olarak, çevrim süresinin 173 olduğu durum dışında, diğer çevrim süreleri için ÖPA her zaman ya en iyi sonucu veren algoritma (çevrim süreleri 86, 89, 92 ve 364 olduğu durumlarda) ya da en iyi sonucu veren yöntemlerden biri olmuştur. Bu sonuçlar, önerilen Petri ağ tabanlı algoritmanın BMHDP-1 çözümünde etkin olabileceğini göstermektedir.

5. SONUÇ

Montaj hattı dengeleme problemi, farklı biçimlerde tanımlanmasından dolayı araştırmacılar arasında popüler bir problem olmuştur. Çeşitli yaklaşımlar kullanılarak bu tip problemlerin çözülmesine çalışılmıştır. Bu makalede BMHDP-1 için, Petri ağları ve onun özelikleri kullanılarak bir algoritma sunulmuştur. Petri ağları, kesikli olaylı sistemlerin modellenmesinde ve analizinde kullanılan, geniş uygulama alanı olan bir tekniktir. Makalede, montaj hattı dengeleme problemi tanıtıldıktan ve ilgili çalışmalara değinildikten sonra Petri ağları ile ilgili bilgiler verilmiştir. Önerilen algoritmanın açıklanmasının ve örnek bir problemin çözülmesinin ardından, uygulama bölümünde önceki çalışmalarda geliştirilen yedi yöntemin sonuçları ile önerilen algoritmanın sonuçları karşılaştırılmıştır. Karşılaştırma sonuçları ışığında, bir çevrim süresi dışında, diğer 12 çevrim süresi için önerilen algoritmanın en iyi sonuçları verdiği gözlenmiştir. Dört çevrim süresi için, en iyi sonuçları veren tek yöntem, önerilen algoritma olmuştur. Bundan sonraki çalışmalarda BMHDP'nin farklı tipleri için, geliştirilen bu algoritmanın uygulanabilirliği araştırılacaktır.

KAYNAKLAR

Adamou M., Zerhoni S.N., Bourjault A. (1998): "Hierarchical Modeling and Control of Flexible Assembly Systems Using Object-oriented Petri Nets", International Journal of Computer Integrated Manufacturing. Vol. 11, No. 1, pp. 18-33.

Bautista J., Suarez R., Mateo M., Companys R. (2000): "Local Search Heuristics for Assembly Lien Balancing Problem with Incompatibilities between Tasks", Proceedings of the 2000 IEEE International Conference on Robotics & Automation, San Francisco, CA., pp. 2404-2409.

Baybars I. (1986b): "A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem", Management Science, Vol. 32, No. 8, pp. 909-932.

Baybars I. (1986a): "An Efficient Heuristic Method for the Simple Assembly Line Balancing Problem", International Journal of Production Research, Vol. 24, No. 1, pp. 149-166.

Cao T., Sanderson A.C. (1994): "Task Decomposition and Analysis of Robotic Assembly Task Plans Using Petri Nets", IEEE Transactions on Industrial Electronics, Vol. 41, No.6, pp. 620-629.

Chiang W.C. (1998): "The Application of a Tabu search Metaheuristic to the Assembly Line Balancing Problem", Annals of Operations Research, No. 77, pp. 209-227.

D'Souza K.A., Khator S.K. (1997): "System Reconfiguration to Avoid Deadlocks in Automated Manufacturing Systems", Computers Industrial Engineering, Vol. 32, No.2. pp 455-465.

Desrochers A.A., Al-Jaar R.Y. (1995): "Applications of Petri Nets in Manufacturing Systems, Modeling, Control, and Performance Analysis", IEEE Press, New York.

Erel E., Sarin S.C. (1998): "A Survey of the Assembly Line Balancing Procedures", Production Planning and Control, Vol. 9, No. 5, pp. 414-434.

Frey G. (2000): "Assembly Line Sequencing based on Petri Net T-Invariants", Control Engineering Practice, Vol. 8, pp. 63-69

Hoffmann T.R. (1992): "EUREKA: A Hybrid System for Assembly Line Balancing", Management Science, Vol. 38, pp. 39-47

Jeng M.D. (1997): "Petri Nets for Modeling Automated Manufacturing Systems with Error Recovery", IEEE Transaction on Robotics and Automation, Vol. 13, No. 5, pp. 752-760.

Johnson R.V. (1988): "Optimally Balancing Large Assembly Lines with FABLE", Management Science, Vol. 34, No. 2, pp. 240-253.

Kılınççı Ö. (2003): "Simple Assembly Line Balancing Problem: A Petri Net Approach", Doktora Tezi, İzmir, Dokuz Eylül Üniversitesi, Fen Bilimleri Enstitüsü.

Kim Y.K., Kim Y.J., Kim Y. (1996): "Genetic Algorithms for Assembly Line Balancing with various Objectives", Computers and Industrial Engineering, Vol. 30, No. 3, pp. 397-409.

McCarragher B.J. (1994): "Petri Net Modeling for Robotic Assembly and Trajectory Planning", IEEE Transactions on Industrial Electronics, Vol. 41, No. 6, pp. 631-640.

Moore K.E., Güngör A., Gupta S.M. (2001): "Petri Net Approach to Disassembly Process Planning for Products with Complex AND/OR Precedence Relationships", European Journal of Operational Research, Vol. 135, pp. 428-449.

Murata T. (1989): "Petri Nets: Properties, Analysis and Applications", Proceedings of IEEE, Vol. 77, No. 4, pp. 541-580.

Nourie F.J., Venta E.R. (1991): "Finding the Optimal Line Balances with OptPack", Operations Research Letters, Vol. 10, pp. 165-171.

Ramaswamy S., Valavanis K.P., Barber S. (1997): "Petri Net Extensions for the Development of MIMO Net Models of Automated Manufacturing Systems", Journal of Manufacturing Systems, Vol. 16, No. 3, pp. 175-191.

Rubinovitz J., Levitin G. (1995): "Genetic Algorithm for Assembly Line Balancing", International Journal of Production Economics, Vol. 41, pp. 343-354.

Sabuncuoğlu I., Erel E., Tanyer M. (2000): "Assembly Line Balancing Using Genetic Algorithms", Journal of Intelligent Manufacturing, No. 11, pp. 295-310.

Scholl A., Klein R. (1997): "SALOME: A Bidirectional Branch-and-bound Procedure for Assembly Line Balancing", INFORMS Journal on Computing, Vol. 9, No. 4, pp. 319-334.

Scholl A., Voss S. (1996): "Simple Assembly Line Balancing-heuristic Approaches", Journal of Heuristics, No. 2, pp. 217-244.

Sprecher A. (1999): "A Competitive Branch-and-bound algorithm for the Simple Assembly Line Balancing Problem", International Journal of Production Research, Vol. 37, No. 8, pp. 1787-1816.

Teng S.H., Zhang J. (1993): "A Petri Net-based Decomposition Approach in Modeling of Manufacturing Systems", International Journal of Production Research, Vol. 31, No. 6, pp. 1423-1439.

Zha X.F., Lim S.Y.E., Fok S.C. (1998): "Integrated Knowledge-based Petri Net Intelligent Flexible Assembly Planning", Journal of Intelligent Manufacturing, Vol. 9, pp. 235-250.