
GAZİ ÜNİVERSİTESİ
Gazi
1926

HİBRİT ESNEK AKIŞ TİPİ ROBOTİK HÜCRELERDE ROBOT HAREKET DİZİSİ BELİRLEME VE ÇOKLU PARÇA ÇİZELGELEME

Gül Didem BATUR SİR

DOKTORA TEZİ
ENDÜSTRİ MÜHENDİSLİĞİ ANABİLİM DALI

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HAZİRAN, 2014

Gül Didem BATUR SİR tarafından hazırlanan "Hibrit Esnek Akış Tipi Robotik Hücrelerde Robot Hareket Dizisi Belirleme ve Çoklu Parça Çizelgeleme" adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ ile Gazi Üniversitesi Endüstri Mühendisliği Anabilim Dalında DOKTORA TEZİ olarak kabul edilmiştir.

Danışman: Prof. Dr. Serpil EROL

Endüstri Mühendisliği Anabilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum. ……..

İkinci Danışman: Doç. Dr. Oya KARAŞAN

Endüstri Mühendisliği Anabilim Dalı, Bilkent Üniversitesi

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum. ...……

Başkan: Prof. Dr. Selim AKTÜRK

Endüstri Mühendisliği Anabilim Dalı, Bilkent Üniversitesi

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum. ...……

Üye: Prof. Dr. Orhan TÜRKBEY

Endüstri Mühendisliği Anabilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum. ...……

Üye: Prof. Dr. Hasan BAL

İstatistik Anabilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum. ...……

Üye: Doç. Dr. Ediz ATMACA

Endüstri Mühendisliği Anabilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum. .....……

Üye: Yrd. Doç. Dr. Bahar ÖZYÖRÜK

Endüstri Mühendisliği Anabilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum. ...……

Tez Savunma Tarihi: 05 / 06 / 2014

Jüri tarafından kabul edilen bu tezin Doktora Tezi olması için gerekli şartları yerine getirdiğini onaylıyorum.

……

Prof. Dr. Şeref SAĞIROĞLU

Fen Bilimleri Enstitüsü Müdürü

ETİK BEYAN

Gazi Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Gül Didem BATUR SİR

05.06.2014

HİBRİT ESNEK AKIŞ TİPİ ROBOTİK HÜCRELERDE ROBOT HAREKET DİZİSİ BELİRLEME VE ÇOKLU PARÇA ÇİZELGELEME

(Doktora Tezi)

Gül Didem BATUR SİR

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Haziran 2014

ÖZET

Bu tez çalışması, hibrit esnek akış tipi hücrede çoklu parça tiplerinden oluşan bir parça kümesinin tekrarlı olarak üretildiği ve makineler arası taşımaların bir robot yardımıyla gerçekleştirilmesi ile ortaya çıkan çizelgeleme problemini göz önüne almıştır. Hücrenin çevrim zamanı, robot hareket dizisi, parça/makine atamaları ve parça sıralamalarından etkilenmektedir. Aşamaların birinde tek diğerinde iki makinenin bulunduğu hibrit esnek akış tipi bir hücrede en iyi çevrim zamanının belirlenmesi problemi öncelikle Gezgin Satıcı Probleminin özel bir şekli olarak modellenmiş ve GAMS/CPLEX Çözücüsü kullanılarak çözüm yapılmıştır. Daha sonra, büyük boyutlu problemlerin çözümü için, Tavlama Benzetimi tabanlı bir sezgisel algoritma geliştirilerek, algoritma içerisinde tanımlanan iki farklı komşuluk yapısı ile çözüm aranmış ve sonuçlar önerilen alt sınır değeri ile karşılaştırılmıştır.

Bilim Kodu : 906.1.141
Anahtar Kelimeler : Paralel makine çizelgeleme, Hibrit hücre, Robotik sistemler, Matematiksel modelleme, Tavlama benzetimi
Sayfa Adedi : 121
Danışman : Prof. Dr. Serpil EROL
İkinci Danışman : Doç. Dr. Oya KARAŞAN

ROBOT MOVE SEQUENCE DETERMINING AND MULTIPLE PART TYPE SCHEDULING IN HYBRID FLEXIBLE FLOW SHOP ROBOTIC CELLS

(Ph.D. Thesis)

Gül Didem BATUR SİR

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

June 2014

ABSTRACT

This thesis considers the scheduling problem arising in hybrid flexible flow shops which repeatedly produce a set of multiple part-types and where transportation of the parts between the machines is performed by a robot. The cycle time of the cell is effected by the robot move sequence, part/machine assignments and part sequences. In a hybrid flexible flow shop in which there exist one machine in the first and two machines in the second stage, the problem of determining the best cycle time is first modeled as a special case of Travelling Salesman Problem and solved using GAMS/CPLEX Solver. Then, for large sized problems, a Simulated Annealing based heuristic is constructed and the problem is solved using two different neighborhood structures defined. The results are also compared against a proposed lower bound value.

Science Code : 906.1.141
Key Words : Parallel machine scheduling, Hybrid cells, Robotic systems, Mathematical modelling, Simulated Annealing
Page Number : 121
Supervisor : Prof. Dr. Serpil EROL
Co-supervisor : Assoc. Prof. Dr. Oya KARAŞAN

TEŞEKKÜR

Tez çalışmamdaki yardım ve katkılarının yanı sıra iyi ve kötü günlerimde yanımda olup sonsuz ilgi ve sabrı ile beni yönlendiren, kıymetli tecrübelerinden her zaman faydalandığım, değerli danışmanım Prof. Dr. Serpil EROL'a içten teşekkürlerimi sunarım.

Kendisini tanıdığım günden beri hep yanımda olan, anlayış ve desteği her zaman büyük güç veren, sevgili hocam ve danışmanım Doç. Dr. Oya KARAŞAN'a teşekkürü bir borç bilirim.

Değerli hocam Prof. Dr. Selim AKTÜRK'e, gerek yüksek lisans gerekse doktora çalışmalarım sırasındaki paha biçilemez rehberliği ve önerileri için; kıymetli hocam Prof. Dr. Orhan TÜRKBEY'e doktora sürecim boyunca verdiği destek için teşekkür ederim.

Tez savunmam sırasında yanımda olmayı ve değerli fikirlerini benimle paylaşmayı kabul eden jüri üyelerim Prof. Dr. Hasan BAL, Doç. Dr. Ediz ATMACA ve Yrd. Doç. Dr. Bahar ÖZYÖRÜK'e teşekkürlerimi sunarım.

Çalışmamın zorlandığım aşamalarında sağladığı yardımlar ve manevi desteğiyle yanımda olan hocam Yrd. Doç. Dr. İsmail KARAOĞLAN'a; tüm süreç boyunca bunaldığım her noktada çekinmeden aradığım ve beni hiçbir zaman yanıtsız bırakmayan arkadaşlarım Dr. Emre ÇALIŞKAN, Dr. Turan BİROL ve Kutay KAZBEK'e çok teşekkür ederim.

Hayatımın her alanında ve her anında gösterdikleri ilgi, sabır, anlayış ve destekleri için, sahip olduğum en büyük şanslar olan annem Nevin BATUR, babam Yaşar BATUR ve biricik kardeşim Dilşat BATUR'a; yanımda olduğu için her gün şükrettiğim, beni benden iyi tanıyan ve benden çok düşünen sevgili eşim Ender SİR'e -teşekkür etmenin yeterli olmayacağını bilsem de- teşekkür ederim. Bu dört insanın sevgisi olmasaydı çalışmamın bir önemi kalmayacağı gibi hayatımın amacı ve anlamı da eksik kalırdı.

Sonuncu ama en önemli olarak, birkaç ay sonra yanımızda olacağını umduğum canım kızıma; şimdiden hissettirdiği benzersiz sevgisi, çalışmamın bu son adımlarını benimle birlikte geçirdiği ve böyle bir dönemde beni hiç yormayarak gösterdiği desteği için minnettarım. Bu tez, kızıma adanmıştır…

İÇİNDEKİLER

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış bazı kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

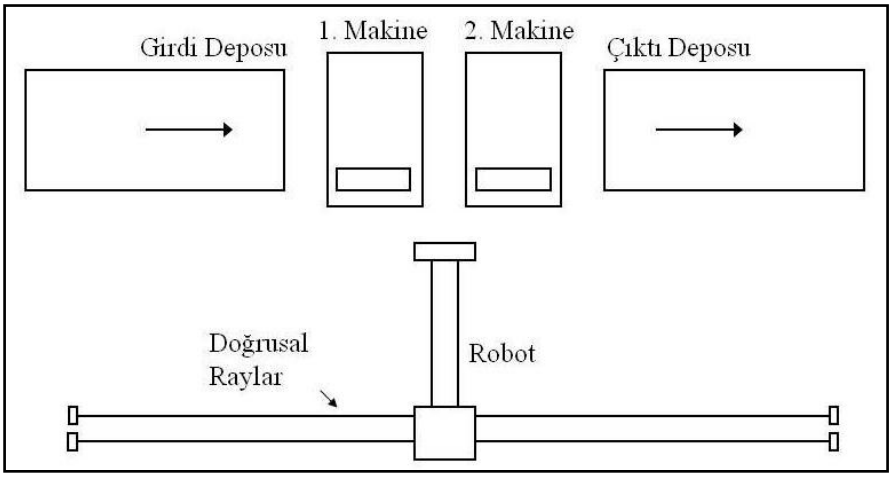
Kısaltmalar	Açıklama
EAH	Esnek Akış Hattı
EAPK	En Az Parça Kümesi
GSP	Gezgin Satıcı Problemi
HAT	Hibrit Akış Tipi
HEAT	Hibrit Esnek Akış Tipi
TB	Tavlama Benzetimi

1. GİRİŞ

Bu çalışmanın çıkış noktası imalat endüstrilerindeki otomasyon seviyesindeki artıştır. Günümüzün yüksek rekabete dayalı dünyasında, imalat sektörleri için verimlilik önemli bir faktördür. Mevcut teknolojik gelişmeler endüstriler için yeni olanaklar sağlamaktadır. Kamoun ve arkadaşlarının belirttiği üzere, esnekliği artırmaya yönelik olarak hazırlık zamanları düşürüldüğünden malzeme taşıma zaman ve maliyeti darboğaz haline gelmiş; etkin malzeme taşıma önem kazanmıştır (Kamoun, Hall ve Sriskandarajah, 1999). Bu amaçla günümüzde robotlar otomotiv, elektrik, elektronik, makine endüstrilerinde sıkça kullanılmaktadır. Browne, Harhen ve Shivnan, robotların aşağıdaki amaçlara yönelik olarak kullanıldığını belirtmişlerdir:

- İşgücü maliyetini azaltmak,
- Çıktı oranını arttırmak,
- Daha esnek üretim sistemleri sağlamak,
- Tehlikeli şartlarda çalışan işçilerin yerini almak,
- Kalifiye işgücünün yokluğunu telafi etmek,
- Daha tutarlı kalite kontrolü sağlamak (Browne, Harhen ve Shivnan, 1996).

Belirli sayıda makine ve malzeme taşımakta kullanılan bir robottan oluşan üretim hücreleri robotik hücre olarak adlandırılmaktadır. Şekil 1.1'de iki makineli hatsal bir robotik hücre gösterilmiştir. Sistemde, girdi deposundan alınan parça, makinelerin durumu ve sistemin yapısına uygun olarak, robot tarafından ilgili makinelere taşınmakta ve burada işlem gördükten sonra yine robot tarafından çıktı deposuna götürülmektedir. Bu hücrelerin etkin kullanımı önemli ve zor problemlerin çözümü ile mümkün olmaktadır.



Şekil 1.1. Hatsal robotik hücre yerleşimi

Bu çalışmada CNC makinelerden oluşan esnek robotik üretim hücreleri üzerinde durulmaktadır. Ele alınan robotik hücreler, farklı parça tipleri içeren partiler için işlem yapabilmektedir. Genel olarak, bu tip parçalar belirli bir makine için farklı işlem zamanlarına sahiptirler. Çoklu parça tipi problemleri, karşıtı olan özdeş parça tipi problemlerinden daha zordur. Tam zamanında üretimle bağlantılı olarak her partideki parça tiplerinin ilgili oranları talepteki ilgili oranlarla aynı olmalıdır. Bundan dolayı, araştırmacılar En Az Parça Kümesi (EAPK) olarak adlandırılan ve bu eşit oranları içeren çevrimler üzerine yoğunlaşmıştır. Bu terim toplam talepteki ürün miktarına bağlı olarak, tüm parça tiplerinden aynı oranlarda parça içeren kümeyi tanımlamaktadır. Örneğin, bir şirketin toplam talebinin üç ürüne, A ürünü 40%, B ürünü 35% ve C ürünü 25%'ini oluşturacak şekilde bölündüğü durumda bir EAPK 8 adet A ürünü, 7 adet B ürünü ve 5 adet C ürünü olmak üzere toplam 20 üründen oluşacaktır. Birçok uygulamada EAPK'lerin tekrarlı veya çevrimsel olarak üretilmesinde robotik hücreler kullanılmaktadır. Çoklu parça tipi probleminde üretim hedefindeki oranlarla aynı parça oranlarına sahip en küçük mümkün küme olan EAPK tekrarlı olarak üretilmektedir. Amaç çevrimsel üretim çerçevesinde bir EAPK üretimi için gerekli ortalama zamanın minimize edilmesidir. Çıktı oranı bu ortalama zamanın tersi olarak tanımlanmaktadır. Genel olarak, hücrede k farklı tipte parça üretilmektedir. Bir EAPK'de $i = 1,2,\dots,k$ olmak üzere i tip parçadan r_i adet üretim olmaktadır. Çevrimdeki tamamlanan parça sayısı $n = r_1 + r_2 + \cdots + r_k$'dır. Bir EAPK çevrimi, EAPK parçalarının girdi deposundan sisteme girdiği, işlendiği, çıktı deposundan sistemden ayrıldığı ve sistemin başlangıç durumuna döndüğü çevrimdir. EAPK çevrimi, EAPK parçalarının hücreye gireceği sıralamanın ve bu parçaların hücrede

göreceği işlemlerin çizelgesinin belirlenmesiyle tanımlanmaktadır. EAPK parçalarının sıralaması, *EAPK parça dizisi* veya basitçe *parça dizisi* olarak adlandırılmaktadır. *EAPK robot hareket dizisi* veya *robot hareket dizisi* ise EAPK çevrimi süresince gerçekleştirilecek robot aktivitelerinin sıralamasıdır. Minimum çevrim zamanını veren EAPK çevrimini bulmaktaki zorluk iki yönlüdür, yani verilmesi gereken iki karar vardır: (a) robot hareket dizisinin seçilmesi, (b) parça dizisinin belirlenmesi. Robotik bir hücrede parçaların optimal olarak çizelgelenmesinde amaç minimum çevrim zamanını veren EAPK çevrimini bulmaktır. Bu amaç, çevrim zamanını minimize edecek şekilde, parça dizisi ve robot hareket dizisinin eşzamanlı olarak belirlenmesini gerektirmektedir (Dawande, Geismar, Sethi ve Sriskandarajah, 2005).

Birçok farklı hücre tipinde robotlardan yararlanılması mümkündür. Bu çalışmada hibrit akış tipi hücrelerde (HAT) robot kullanımı incelenecektir. Bir HAT hücrede n sayıda parça m aşamada işlem görmekte ve bu aşamaların her birinde özdeş paralel makineler bulunmaktadır. Aşamalarda birden fazla makine yer almasının tercih edilmesinin nedeni aşamaların kapasitelerinin dengelenmesi, sistemin toplam kapasitesinin arttırılması veya sistemdeki darboğaz aşamaların etkilerinin önüne geçilmesi olarak ifade edilebilir.

Literatürde, birbirine benzer özellikler taşımak üzere, paralel makineli akış tipi, hibrit akış tipi ve esnek akış hatları başlıkları altında çalışmalar bulunmaktadır. Bu çalışmaların farklılıkları ve üzerinde çalışılacak olan sistemle ilişkileri Bölüm 2'de ele alınacaktır. HAT'lerin birçok farklı şekli olmasına rağmen, aşağıdaki özellikler çoğunda ortak olarak görülmektedir:

- En az 2 işlem aşaması bulunmaktadır.
- Her k aşamasında $M^k \geq 1$ makine bulunmakta ve en az bir aşamada $M^k > 1$ olmaktadır.
- Tüm parçalar aynı üretim akışını izleyerek işlem görmektedir (sırasıyla, 1. aşama, 2. aşama, … , m. aşama).
- j parçasının k. aşamada işlem göreceği süre P_{jk}'dır. Bu işlem, o_{jk} operasyonu olarak adlandırılmaktadır.

HAT probleminin standart formunda şu varsayımlar yapılmaktadır (Ruiz ve Vázquez-Rodríguez, 2010):

- Tüm parça ve makineler $t = 0$ anında hazır durumdadır.
- Belirli bir aşamadaki tüm makineler özdeştir.
- Bir makine aynı anda yalnız bir operasyonu gerçekleştirebilmekte ve bir parça yalnız bir makinede işlem görebilmektedir.
- Hazırlık zamanları ihmal edilmektedir.
- İş keserek öncelik vermeye izin verilmez.
- Aşamalar arası stok alanları limitsizdir.
- Problem verileri deterministik yapıdadır ve kesin olarak bilinir.

Ruiz ve Vázquez-Rodríguez'in tanımına göre, bir parçanın bazı aşamalarda işlem görmüyor olması mümkündür ancak her parça en az bir aşamada işlenmek zorundadır (Ruiz ve Vázquez-Rodríguez, 2010). Bunun yanı sıra, HAT'lerde n parçanın tümünün aynı operasyonel sırayı (işlem rotasını) izlediğini ve 1. aşamadan m. aşamaya dek tüm aşamalarda işlem görmekte olduğunu öne süren çalışmalar da bulunmaktadır (Naderi, B., Zandieh, Khaleghi Ghoshe Balagh ve Roshanaei, 2009a).

Bu çalışmada, daha güncel olması göz önüne alınarak, Ruiz ve Vázquez-Rodríguez'in 'hibrit esnek akış tipi' tanımına göre yol alınacak; parçaların bazı aşamalarda işlem görmemesinin mümkün olduğu durum, 'esnek akış hatları' başlığı altında ayrıca incelenecektir (Ruiz ve Vázquez-Rodríguez, 2010).

HAT probleminin çözümü de iki yöne sahiptir: parçaların her aşamadaki çeşitli makinelere atanması ve her aşamadaki parçaların sıralanması (Gupta, 1988). Bu problemler robotik hücreler ile bir araya getirildiğinde, çalışmada ele alınacak problemler parça dizisi, tüm aşamalar için parça-makine atamaları ve robot hareket dizisinin belirlenmesi halini almaktadır.

Literatürde robotik hücreler ile hibrit esnek akış tipinin birlikte dikkate alındığı bir çalışmaya rastlanmamıştır. Farklı işlem gereksinimlerine sahip olan çoklu parçaların göz önünde bulundurulması problem zorluğunu arttırmakta, robot hareketlerinin belirlenmesi

hedefi ise problemi daha da karmaşık bir yapıya sokmaktadır. Söz konusu problemin çözümü için öncelikle karışık tamsayılı bir programlama modeli kurulmuştur.
Ancak problem karmaşıklığından dolayı, tamsayılı programlama çözüm yöntemlerinin etkin olarak kullanılamadığı görülmüştür. Bu nedenle problem çözümünde sezgisel yöntemlerin kullanılması üzerine araştırmalar yapılmış, bulunan yöntemler arasından Tavlama Benzetimi (TB)'nin etkin çözüm verdiği gözlenmiştir.

Sonraki bölümde konuyla ilgili geniş bir literatür taraması verilmiştir. Bölüm 3'te çalışmayla ilgili notasyonlar ve temel varsayımlar sunulmuş ve ele alınan problem tanımlanmıştır. Bölüm 4'te önerilen matematiksel model verilmektedir. Bölüm 5'te oluşturulan sezgisel algoritma tanımlanmış ve Bölüm 6'da sözkonusu algoritma, probleme yönelik geliştirilen alt sınır değeri ile karşılaştırılarak değerlendirilmiştir. Bölüm 7 çalışma ile elde edilen sonuçları açıklamakta ve gelecek çalışmalar için olası konuları sunmaktadır.

2. LİTERATÜR ARAŞTIRMASI

2.1. Paralel Makineli (Çok İşlemcili) Akış Tipi

Tipik bir akış tipi hücre, tüm ürünlerin belirli sayıda aşamadan aynı sırayla geçmek zorunda olduğu çok aşamalı bir üretimi içermektedir. Klasik akış tipinde her aşama, aynı anda en fazla bir operasyonu yerine getirebilen tek bir makineden oluşmaktadır. Bazı çalışmalarda ise her aşamada, birbirine paralel olarak çalışan, belirli sayıda özdeş makinenin var olduğu varsayılmış ve bunlar çok işlemcili akış tipi hücreler olarak adlandırılmıştır (Low, 2005). Bu şekilde bir çizelgeleme ortamı yaygın olarak kullanılmakta ve yarı-iletkenler, elektronik imalatı ve petrokimyasal üretim gibi alanlarda gözlenmektedir. Her aşamasında özdeş makinelerin bulunduğu çok aşamalı akış tipi hücreler literatürde 'hibrit akış tipi' olarak da adlandırılmaktadır (Gupta, 1988). Ancak, hibrit akış tipi literatüründe özdeş olmayan parçalar ve özdeş makineler üzerinde durulurken, paralel makineli akış tipi çalışmalarında, genellikle, özdeş parçalar ve benzer paralel makineler üzerine odaklanılmıştır (Dessouky, Dessouky ve Verma, 1998).

Sriskandarajah, 'paralel makineli beklemesiz akış tipi' olarak bilinen bir imalat sistemindeki parçaların çizelgelenmesine yönelik algoritmaların performansı hakkında çalışmıştır. Beklemesiz akış tipinde her parça ilk makine merkezindeki başlangıcından son makine merkezindeki tamamlanışına dek, herhangi bir kesilmeye ve makine merkezleri arasında beklemeye uğramaksızın, sürekli olarak işlem görmek zorundadır. Çalışmada sezgisel algoritmalar önerilmiş ve en kötü durum performansları açısından analiz edilmiştir (Sriskandarajah, 1993).

Dessouky ve arkadaşları, akış tipi düzenindeki çok aşamalı sistemler üzerinde durmuş ve 2-aşamalı problemin polinom zamanda çözülebileceğini, aşama sayısı 2'den fazla olduğu durumlarda ise problemin NP-zor olduğunu göstermişlerdir (Dessouky ve diğerleri, 1998).

Nowicki ve Smutnicki, paralel makineli akış tipi hücrede maksimum tamamlanma zamanının minimizasyonu problemi ile ilgilenmiş ve özel bir komşuluk yapısına sahip, tabu arama tabanlı bir algoritma geliştirmişlerdir. Göz önüne aldıkları sistem her bir merkezinde birden fazla sayıda özdeş makinenin bulunduğu çok sayıda makine merkezinden oluşmaktadır (Nowicki ve Smutnicki, 1998).

Low, özdeş olmayan çoklu makineye sahip akış tipi problemini dikkate almış ve bağımsız hazırlık zamanları ile bağımlı hareket zamanları gibi bazı teknik kısıtlamaları göz önünde tutmuştur (Low, 2005).

2.2. Hibrit Akış Tipi

İmalat işlemleri karmaşıklaştıkça ve fabrikalar, işlem görebilecek parça sayısına yönelik kapasitelerini arttırmayı arzuladıkça temel akış hattı sisteminden uzaklaşılmaktadır. Bir akış hattında, tüm parçalar, ilk aşamadan son aşamaya doğru, doğrusal bir yapıyla, aynı makinelerde işlem görmekte ve her aşamada tüm işlemler tek bir makine tarafından gerçekleştirilmektedir. Bir aşamanın kapasitesinin arttırılması için ilave paralel makinelerin satın alınması mümkündür. Aşamalarda birden çok sayıda (çoğunlukla özdeş) makinenin bulunmasına izin vermeye yönelik bu genişleme, akış hattını hibrit akış hattı'na dönüştürmektedir (Kurz ve Askin, 2003).

HAT problemi, karmaşıklığı ve pratikteki kullanılırlığı göz önünde bulundurularak, birçok çalışmada dikkate alınmaktadır. Bu alanda Vignier, Billaut ve Proust, Linn ve Zhang, Wang, Quadt ve Kuhn tarafından yapılmış literatür araştırmaları bulunmakla birlikte, bilinen son çalışma 2010 yılında Ruiz ve Vázquez-Rodríguez tarafından yapılmıştır (Vignier, Billaut ve Proust, 1999; Linn ve Zhang, 1999; Wang, 2005; Quadt ve Kuhn, 2007; Ruiz ve Vázquez-Rodríguez, 2010) .

2.2.1. Özdeş paralel makineli hibrit akış tipi

Rao, Bolat, Al-Harkan ve Al-Harbi, Guirchoun, Martineau ve Billaut, Arthanary ve Ramaswamy, Mittal ve Bagga, birinde tek makine, diğerinde iki özdeş makine bulunan 2-aşamalı durumu ele alarak problemin en basit yapısı üzerinde çalışmış ve kesin algoritmalar geliştirmişlerdir (Rao, 1970; Bolat, Al-Harkan ve Al-Harbi, 2005; Guirchoun, Martineau ve Billaut, 2005; Arthanary ve Ramaswamy, 1971; Mittal ve Bagga, 1973).

Gupta, her aşamada özdeş makinelerin bulunduğu, 2-aşamalı akış tipi problemi tanımlamış ve bu problemin NP-tam olduğunu göstermiştir. Yazar, 1. aşamada m özdeş makinenin, 2. aşamada tek makinenin bulunduğunu varsaydığı 2-aşamalı HAT problemini

dikkate alarak maksimum tamamlanma zamanını minimize etmeye yönelik bir sezgisel algoritma geliştirmiştir (Gupta, 1988).

Gupta ve Tunc, Gupta'nın çalışmasının tam tersi olarak, 1. aşamasında tek makine, 2. aşamasında m özdeş makinenin bulunduğu 2-aşamalı HAT problemini dikkate almış ve problemin NP-tam olmasını göz önünde tutarak, kabul edilebilir (optimal veya optimale yakın) bir sonuç bulmak için polinom sınırlı iki sezgisel algoritma geliştirmişlerdir (Gupta ve Tunc, 1991).

Gupta ve Tunc, 1. aşamada tek makine ve 2. aşamada herhangi sayıda özdeş makinenin olduğu problemi ayrılabilir ve sıradan bağımsız hazırlık ve hareket zamanlarıyla birlikte maksimum tamamlanma zamanı kriteri altında ele almışlardır. Çalışmada, 2. aşamadaki makine sayısının toplam parça sayısına eşit veya ondan büyük olması durumu için polinom zamanlı bir optimizasyon algoritması geliştirilmiştir (Gupta ve Tunc, 1994).

Lee ve Kim ile Gupta, Hariri ve Potts, 2-aşamalı ve aşamalarından birinde herhangi sayıda özdeş makinenin bulunduğu durumlar üzerinde çalışmış ve probleme Dal-Sınır metodu uygulamışlardır (Lee ve Kim, 2004; Gupta, Hariri ve Potts, 1997).

Haouari, Hidri ve Gharbi, her aşamasında en az iki özdeş makinenin bulunduğu iki aşamalı HAT problemine yönelik olarak bir dal-sınır algoritması geliştirmişlerdir. Algoritma içerisinde alt sınır değerleri ile birlikte ayarlama prosedürleri, baskınlık kuralları ve bazı sezgiseller de bulunmaktadır (Haouari, Hidri ve Gharbi, 2006).

Naderi ve diğerleri, hazırlık zamanlarının sıra bağımlı olduğu hibrit akış tipi hücrelerin çizelgelenmesi problemine, maksimum tamamlanma zamanı ve maksimum gecikmeyi minimize etme amacıyla yaklaşmışlardır. Problemin çözümü için TB'yi basit bir yerel arama tekniğiyle birleştirmiş ve önerilen algoritmanın performansını değerlendirmişlerdir (Naderi ve diğerleri, 2009a).

Bir başka çalışmada, Naderi, Zandieh ve Roshanaei, hibrit akış tipi problemini toplam tamamlanma zamanı ve toplam gecikmeyi minimize etmek amaçlarıyla birlikte göz önüne almışlardır. Sıra bağımlı hazırlık ve taşıma zamanlarını probleme katmış ve TB tabanlı bir metasezgisel uygulamışlardır (Naderi, Zandieh ve Roshanaei, 2009b).

Li, Wang ve Huo, sıra bağımlı hazırlık zamanları ve permütasyon kısıtları gevşetmesini içeren bir hibrit akış tipi problemi üzerinde çalışmışlardır. Tüm parçaların aynı üretim rotasını izlediğinin varsayılmasının yanında çeşitli taleplerin farklı üretim aşamalarına ihtiyaç duyulabileceği de kabul edilmiştir. Bu, parçanın herhangi bir işleme ihtiyaç duymaması durumunda o aşamadaki işlem süresinin sıfır olarak alınması anlamına gelmektedir (Li, Wang ve Huo, 2010).

Mirsanei, Zandieh, Moayed ve Khabbazi, sıra bağımlı hazırlık zamanlı HAT çizelgeleme problemi için TB tabanlı bir algoritma geliştirmiş ve sonuçları rassal anahtarlı genetik algoritma ve bağışıklık algoritması ile karşılaştırmışlardır (Mirsanei, Zandieh, Moayed ve Khabbazi, 2011).

Carpov, Carlier, Nace ve Renaud, 2-aşamalı HAT probleminde 2. aşamada paralel makineler ile öncelik kısıtlarının bulunduğu durum üzerinde çalışmışlardır. Sözkonusu öncelik kısıtları ile bir işin işlenmesine başlandıktan sonra tüm aşamalarda aralıksız olarak işlem göreceği varsayılmaktadır. Çalışmada bu probleme yönelik olarak sezgisel yöntemler geliştirilmiş ve önerilen alt sınır değeri kullanılarak üretilen test problemleri üzerinden karşılaştırmalar yapılmıştır (Carpov, Carlier, Nace ve Renaud, 2012).

Bozejko, Pempera ve Smutnicki, HAT problemine yönelik olarak tabu arama algoritmasını geliştirerek paralel tabu arama algoritması kullanmışlardır. Çalışmada sözü edilen algoritma içerisinde iki metot önerilmiştir: birincisi kriter fonksiyonlarının paralel hesaplamalarının yapıldığı hızlı metot, ikincisi ise tabu aramanın içine lokal komşu paralel arama yönteminin katıldığı metottur (Bozejko, Pempera ve Smutnicki, 2013).

Pan, Wang, Li ve Duan, tüm parçaların aynı işlem sırasını takip ettikleri HAT çizelgelemesinde maksimum tamamlanma zamanı minimizasyonu için kesikli yapay arı kolonisi algoritması kullanmış ve kendilerinden önce problemi ele almış olan farklı araştırmacıların sonuçları ile karşılaştırmalar yapmışlardır (Pan, Wang, Li ve Duan, 2014).

Khalili, toplam gecikme ve maksimum tamamlanma zamanının minimize edilmeye çalışıldığı HAT problemine yönelik olarak öncelikle matematiksel bir model oluşturmuş, ardından çok amaçlı elektromanyetizma algoritması geliştirmiştir. Önerilen algoritma,

parçaların optimale doğru gidişi için bir çekme/itme mekanizması kullanan popülasyon tabanlı, esnek ve etkin bir algoritma olarak tanımlanmaktadır (Khalili, 2014).

Marichelvam, Parabaharan ve Yang, maksimum tamamlanma zamanının minimize edildiği HAT çizelgeleme problemi için bir guguk kuşu arama algoritması geliştirmişlerdir. Çalışmada ele alınan bir problem genetik algoritma, tavlama benzetimi, karınca kolonisi optimizasyonu gibi farklı algoritmalarla da çözülerek önerilen algoritma ile karşılaştırma yapılmıştır (Marichelvam, Parabaharan ve Yang, 2014).

2.2.2. Özdeş olmayan paralel makineli hibrit akış tipi

Özdeş olmayan makinelerin bulunduğu HAT problemi, özdeş makineli durumdan daha karmaşık bir yapı göstermektedir. Bilinen ilk çalışmada Salvador, her aşamasında özdeş olmayan makinelerin bulunduğu, 2- ve 3-aşamalı HAT'ler üzerinde çalışmış ve bu problem için Dal-Sınır kullanarak çözüm aramıştır (Salvador, 1973).

Huang ve Li, 2-aşamalı bir HAT hücreyi, 1. aşamada tek makine ve 2. aşamada farklı işlem hızları ile karakterize edilmiş, özdeş olmayan makineler bulunduğu varsayımı altında çalışmışlardır. Sistem, grup teknolojisi kapsamında ve her iki aşamada da, bir makine yeni bir parça grubunun işlenmesine geçtiğinde temel bir hazırlığa ihtiyaç duyulduğu göz önünde tutularak ele alınmıştır (Huang ve Li, 1998).

Lin ve Liao, gerçek bir etiket imalatı problemini ele almış ve 2-aşamalı bir HAT olarak modellemişlerdir. Çalışmada sıra bağımlı hazırlık zamanları ve makine uygunluğu göz önünde tutulmuştur (Lin ve Liao, 2003).

Ruiz ve Maroto, seramik fayans üretimi sektöründe bulunan, aşamalarda özdeş olmayan makinelerin ve makine uygunluğunun dikkate alınmasının yanı sıra, sıra bağımlı hazırlık zamanlarının bulunduğu bir problem üzerinde çalışmışlardır. Çalışmada, söz konusu problem için genetik algoritma formunda bir metasezgisel geliştirilmiştir (Ruiz ve Maroto, 2006).

Her aşamasında özdeş olmayan makinelerin bulunduğu ve hazırlık zamanına sahip olan probleme Jungwattanakit, Reodecha, Chaovalitwongse ve Werner tarafından önce Genetik

Algoritmalar ile çözüm önerilmiş; daha sonra aynı probleme Reodecha, Chaovalitwongse ve Werner tarafından dağıtım kuralları, Genetik Algoritma, Tabu Arama ve Tavlama Benzetimini içeren sezgiseller uygulanmıştır. Her iki çalışmada da yazarlar maksimum tamamlanma zamanı ve gecikmiş iş sayısının lineer bir kombinasyonunu amaç fonksiyonu olarak almıştır (Jungwattanakit, Reodecha, Chaovalitwongse ve Werner, 2008; Reodecha, Chaovalitwongse ve Werner, 2009)

Behnamian ve Fatemi Ghomi, sıra bağımlı hazırlık zamanı ile birlikte makine ve kaynak bağımlı işlem zamanına sahip HAT hücrelerde maksimum tamamlanma zamanı ve toplam kaynak ataması maliyetini minimize etmek için genetik algoritma ile yerel komşu aramasını içeren hibrit bir algoritma geliştirmişlerdir (Behnamian ve Fatemi Ghomi, 2011).

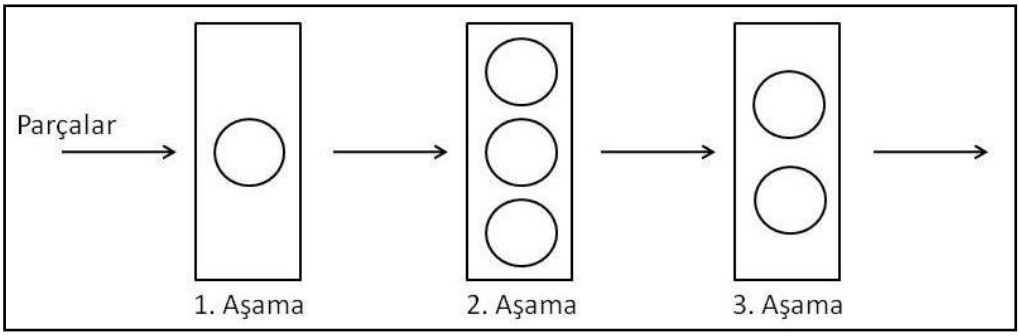
Abyaneh ve Zandieh, sıra bağımlı hazırlık zamanlı HAT çizelgelemesinde limitli ara stok alanı ile maksimum tamamlanma zamanı ve toplam gecikmeyi minimize etmeye yönelik çalışmışlardır. Çalışmada, çok aşamalı iki adet yöntem önerilmiş ve literatürdeki bazı örnekler ile denemeler yapılarak test edilmiştir (Abyaneh ve Zandieh, 2012).

2.3. Esnek Akış Hatları / Hibrit Esnek Akış Tipi

Esnek akış hattı (EAH) problemleri, paralel makineli akış tipi problemlerine bazı belirli varsayımların eklenmesiyle elde edilmektedir. Bu varsayımlardan bazıları belirli parçaların bir veya daha fazla merkezi işlem görmeden atlayabileceği, makineler ve/veya merkezlerden önce sınırlı kapasiteye sahip stok alanlarının bulunması, merkezler arasında parça taşıma zamanın var olması vb.'dir (Nowicki ve Smutnicki, 1998). Tüm parçaların tüm aşamalarda işlem görmesi kısıtlaması kaldırıldığında sıradan bir akış hattı'ndan EAH'ye geçilmiş olmaktadır. Ancak, EAH'de de parçaların ilk aşamadan son aşamaya doğru ilerlemesi şartı geçerlidir (Kurz ve Askin, 2003).

Şekil 2.1, makineler ve aşamalar arasındaki fiziksel ilişkiyi, örnek bir EAH üzerinde göstermektedir. Bu hat üç aşamadan oluşmaktadır. 1. aşamada bir makine, 2. aşamada üç paralel özdeş makine ve 3. aşamada iki paralel özdeş makine bulunmaktadır. Parçalar 1. aşamadan 3. aşamaya doğru hareket etmekte, ancak tüm aşamalara uğrama zorunluluğu taşımamaktadır. 3-aşamalı esnek bir akış hattında, hangi aşamalarda işlem göreceklerine

bağlı olarak yedi tip parça vardır. Parçaların, virgüllerle birbirinden ayrılmış olarak gösterilen, şu rotalardan birini izlemesi mümkündür: $1, 2, 3, 1-2, 1-3, 2-3, 1-2-3$. Bir parçanın, örneğin, 2. aşamaya gideceği bilindiğinde, aşamadaki üç makineden hangisinde işlem göreceğine karar verilmesi gerekmektedir (Kurz ve Askin, 2003).



Şekil 2.1. 3-aşamalı esnek akış hattı

Wittrock, gerçek bir elektronik üretim sistemi üzerinde çalışmış ve her aşamada özdeş makinelerin bulunduğu 3-aşamalı problemde, parçaların bir veya iki üretim aşamasını atlayabilmesi özelliğini dikkate almıştır. Çalışmada pseudo-dinamik bir programlama sezgiseli önerilmiştir. Bilindiği kadarıyla, HAT literatüründe birden fazla amacı ilk defa olarak göz önünde bulunduran bu çalışmada, maksimum tamamlanma zamanı minimize edilmeye çalışılmış ve minimum değere sahip sıralamalar arasında işlemi süren parçalara bakılmıştır (Wittrock, 1985). Wittrock, daha sonraki bir çalışmasında, aynı uygulamayı aşamalar arasında beklemeye izin verilmeyen ve bunun sonucu olarak makine boşaltılmadan üretime devam edilemeyen durum ile birlikte ele almıştır (Wittrock, 1988).

Chen, Usher ve Palanimuthu, 2-aşamalı bir EAH için maksimum tamamlanma zamanını minimize etme amacıyla bir Tabu Arama sezgiseli sunmuşlardır (Chen, Usher ve Palanimuthu, 1998).

Botta-Genoulaz, paralel özdeş makinelerden oluşan, çok aşamalı HAT probleminde parçaların öncelik kısıtları, duraklamalar ve teslim zamanları dikkate alınarak çizelgelenmesi üzerinde çalışmışlardır. Bu duraklamalar robotik bir hücredeki taşıma zamanlarına benzetilebileceği gibi, sıradan bağımsız makine hazırlıklarının da yükleme/boşaltma operasyonlarına benzetilmesi mümkündür. Çalışmada maksimum gecikmeyi minimize etmek üzere altı adet sezgisel geliştirilmiştir (Botta-Genoulaz, 2000).

Kurz ve Askin, EAH'lerin çizelgelenmesini araştırmışlardır. Bu çalışma konusunun kilit özelliklerinden biri, her aşamada, parçalar arasında sıra bağımlı hazırlık zamanlarının bulunmasıdır. Bir parçanın işleminin tamamlanmasının ardından sonraki parçanın işlemine başlamadan önce bazı hazırlık işlemlerinin gerçekleştirilmesi gerekmektedir. Hazırlık için gerekli zamanın uzunluğu önceki parça ile işlem görecek parçaya bağlı olduğundan bu hazırlık zamanları sıra bağımlıdır. Çalışmada üç tip sezgisel kullanılmıştır: ekleme sezgiselleri (gezgin satıcı problemi, GSP, için kullanılan ekleme sezgiselleri tabanlı), Johnson algoritması ve diğer bazı sezgiseller. Elde edilen sonuçlarla hangi aralık durumlarında hangi metodun daha iyi çalıştığına dair yorumlar yapılmıştır (Kurz ve Askin, 2003).

Kurz ve Askin, sıra bağımlı hazırlık zamanlarının görüldüğü esnek akış hatlarında maksimum tamamlanma zamanını minimize etmeye yönelik çizelgeleme üzerinde araştırma yapmışlardır. Çalışmada bir tamsayılı programlama modeli formüle edilmiş ve modelin direkt olarak çözülmesindeki zorluktan dolayı bazı sezgisel algoritmalar (obur metodlar, akış hattı metodları, ekleme sezgiseli ve rassal anahtarlı genetik algoritma önerilmiştir (Kurz ve Askin, 2004).

Ruiz, Serifoğlu ve Urlings, her aşamada özdeş olmayan makinelerin bulunduğu m aşamada, n parçanın bazı aşamaların atlanmasına izin verilerek çizelgelenmesi problemi için karmaşık tamsayılı modelleme ile bir formülasyon sağlamış ve bazı sezgiseller oluşturmuşlardır. Her makinede, makine duraklamaları, serbest kalma zamanları, makine uygunluğu ve parçalar arası öncelik ilişkilerinin yanı sıra, sıra bağımlı hazırlık zamanlarını dikkate almışlardır (Ruiz, Serifoğlu ve Urlings, 2008).

Zandieh ve Karimi, sıra bağımlı hazırlık zamanlı hibrit esnek akış tipi (HEAT) hücrelerde toplam ağırlıklı gecikme ve toplam tamamlanma zamanını eşzamanlı olarak minimize etmeye yönelik çok amaçlı grup çizelgeleme problemi üzerinde çalışmış, probleme yönelik olarak genetik algoritma tabanlı bir yaklaşım uygulamışlardır (Zandieh ve Karimi, 2011).

Singh ve Mahapatra, esnek akış tipi çizelgeleme problemine yönelik olarak parçacık sürü optimizasyonu (PSO) yaklaşımını kullanmış, elde edilen çizelgeleri toplam tamamlanma zamanı veya maksimum tamamlanma zamanı bazında değerlendirerek, sonuçları Oguz, Zinder, Do, Janiak ve Linchtenstein tarafından önerilen alt sınır değerinden yüzde

sapmalarına göre sunmuşlardır (Singh ve Mahapatra, 2012; Oguz, Zinder, Do, Janiak ve Linchtenstein, 2004).

Sawik, sınırlı ara stoğun bulunduğu ve sınırlı veya sürekli makine uygunluğu kısıtlarına sahip esnek akış tipi çizelgelemesinde çevrimsel veya parti tipi çizelgelemesine yönelik matematiksel modeller geliştirerek aynı zamanda çevrimsel ve parti tipi çizelgelemenin karşılaştırılmasına yönelik olarak çalışmıştır. Belli kısıtlar altında sözkonusu karşılaştırmaları yapmakla birlikte, sonuçta çevrimsel çizelgelemenin çevrimsel olmayan çizelgeleme tiplerinden daha güçlü olduğu görülmüştür (Sawik, 2012).

Çizelge 2.1'de buraya kadar anlatılan üç ana başlık altında yapılmış olan çalışmalar verilmiştir. Bu çalışmada hibrit esnek akış tipi hücreler üzerinde durulmaktadır. HEAT probleminde, paralel makineli akış tipi'nden farklı olarak özdeş olmayan parçaların işlemleri gerçekleştirilmekte ve literatürde bulunan son tanımlara göre, EAH tanımıyla paralel olarak bazı parçaların bazı aşamalarda işlem görmemesine olanak sağlanmaktadır.

Çizelge 2.1. Akış tipi literatürü

Paralel Makineli (Çok İşlemcili) Akış Tipi	Hibrit Akış Tipi		Esnek Akış Hatları
	Özdeş Paralel Makineli	Özdeş Olmayan Paralel Makineli	
- Sriskandarajah, 1993 - Dessouky ve diğerleri, 1998 - Nowicki ve Smutnicki, 1998 - Low, 2005	- Rao, 1970 - Arthanary ve Ramaswamy, 1971 - Mittal ve Bagga, 1973 - Gupta, 1988 - Gupta ve Tunc, 1991 -Gupta ve Tunc, 1994 - Gupta ve diğerleri, 1997 - Lee ve Kim, 2004 - Bolat ve diğerleri, 2005 - Guirchoun ve diğerleri, 2005 - Haouari ve diğerleri, 2006 - Naderi ve diğerleri, 2009a - Naderi ve diğerleri, 2009b - Li ve diğerleri, 2010 - Mirsanei ve diğerleri, 2011 - Carpov ve diğerleri, 2012 - Bozejko ve diğerleri, 2013 - Pan ve diğerleri, 2014 - Khalili, 2014 - Marichelvam ve diğerleri, 2014	- Salvador, 1973 - Huang ve Li, 1998 - Lin and Liao, 2003 - Ruiz ve Maroto, 2006 - Jungwattanakit ve diğerleri, 2008 - Jungwattanakit ve diğerleri, 2009 - Behnamian ve Fatemi Ghomi, 2011 - Abyaneh ve Zandieh, 2012	- Wittrock, 1985 - Wittrock, 1988 - Chen ve diğerleri, 1998 - Botta-Genoulaz, 2000 - Kurz ve Askin, 2003 - Kurz ve Askin, 2004 - Oguz ve diğerleri, 2004 - Ruiz ve diğerleri, 2008 - Zandieh ve Karimi, 2011 - Singh ve Mahapatra, 2012 - Sawik, 2012

2.4. Robotik Hücreler

Üreticilerin daha büyük ve karmaşık robotik hücreler kullanmasıyla birlikte bu sistemlerin optimize edilebilmesi için daha gelişmiş model ve algoritmaların kullanılması ihtiyacı ortaya çıkmıştır. Bu ihtiyacı karşılamak üzere literatürde kimi 1970'lere kadar uzanan ama çoğu 1990'lardan sonra yapılmış birçok çalışma bulunmaktadır. Lee, Lei ve Pinedo, Crama, Kats, Van de Klundert ve Levner, Dawande ve diğerleri ve Brauner'in konuyla ilgili geniş kapsamlı araştırmaları bulunmaktadır (Lee, Lei ve Pinedo, 1997; Crama, Kats, Van de Klundert ve Levner, 2000; Dawande ve diğerleri, 2005; Brauner, 2008).

2.4.1. Tekli parça tipi (özdeş parçalar)

Robotik hücrelerdeki tekli parça tipi çizelgeleme problemleri, parça sıralama problemini içermediğinden, çoklu parça tipi problemlerine kıyasla daha kolay yapıdadır. Robotik hücrelerdeki parçaların ve robot hareketlerinin uzun dönemli çıktıyı maksimize edecek şekilde optimal sırasının belirlenmesi problemine yönelik sistematik çalışma Sethi ve arkadaşları tarafından başlatılmıştır. Bu çalışma, robotik hücre çizelgeleme probleminin başlangıç noktası olarak düşünülebilir. Çalışmada amaç çıktıyı maksimize etmek, diğer bir deyişle çevrim zamanını minimize etmektir. Ele alınan problemlerden biri 2-makineli hücrelerdeki tekli parça tipi problemidir. Bu problem için optimal sonucun 1-birimlik çevrimler olduğu ispatlanmıştır. 2-makineli hücrelerde 1-birimlik iki adet uygun çözüm bulunduğundan, çalışmada bu çözümler için çevrim zamanları karşılaştırılarak optimal sonucu veren şartlar tanımlanmıştır. Ele alınan diğer bir problem 3-makineli sistemlerdeki tekli parça tipi problemidir. Çevrim zamanını minimize edecek 1-birimlik çevrimi veren robot hareket sırasının belirlenmesi üzerinde durulmuştur. Bu problem için yalnız 1-birimlik çevrimlerin dikkate alınmasının sebebi bu kısıtlama olmadan problemin analizinin çok zor olmasıdır. Bu probleme çözüm olarak optimal politikayı belirlemek üzere bir karar ağacı oluşturulmuştur. Çalışmada ayrıca m-makineli durumda 1-birimlik çevrimlerin sayısının $m!$ olduğu gösterilmiştir (Sethi, Sriskandarajah, Sorger, Blazewicz ve Kubiak, 1992).

Crama ve Van de Klundert, m-makineli robotik hücre problemi üzerinde durmuşlardır. Makine sayısı parametre olarak alınırsa, yalnız bir parça tipi olduğunda sadece 1-birimlik çevrimleri göz önüne alarak problemin polinom zamanda çözülebildiğini göstermişlerdir.

Çalışmada belirli bir çizelgenin çevrim zamanını hesaplayan bir algoritma verilmiş; son olarak da her çevrimde 1 birim üretildiği varsayımı altında tekli parça tipi çizelgeleme problemini çözen ve m makine sayısını göstermek üzere, karmaşıklığı $O(m^3)$ olan bir dinamik programlama yaklaşımı sunulmuştur. Çalışmanın bir diğer sonucu optimal çevrim zamanı üzerinde üst ve alt sınırların türetilmesidir (Crama ve Van de Klundert, 1997).

Hall, Kamoun ve Sriskandarajah, tekli parça tipi üretimi yapılan 3-makineli hücreler üzerinde çalışmış ve 1-birimlik çevrimlerin 2-birim üreten daha karmaşık yapıdaki politikalar üzerinde baskın olduğunu göstermişlerdir (Hall, Kamoun ve Sriskandarajah, 1997a). Sethi ve diğerlerinin 3-makineli robotik hücrelere yönelik öne sürdükleri tahminin geçerliliği Crama ve Van de Klundert tarafından sunulmuştur (Sethi ve diğerleri, 1992; Crama ve Van de Klundert, 1999). Brauner ve Finke ise bu ispatı basitleştirmiştir (Brauner ve Finke, 1999). İleriki bir çalışmada ise Brauner ve Finke 4- veya daha büyük makineli hücrelerde optimal çözümün 1-birimlik çevrimler ile elde edilmesinin şart olmadığını göstermiş ve bu gibi durumlara yönelik örnekler sunmuşlardır (Brauner ve Finke, 2001).

Kats ve Levner, malzeme taşımanın otomatik araçlar, vinçler veya robotlar aracılığıyla yapıldığı plastik kalıplama, elektrolizle kaplama ve çelik üretimi gibi sistemlerde kullanılan bir sistemi dikkate almışlardır. Çalışmada çevrim zamanını minimize edecek 1-birimlik robot hareket çevriminin bulunması amacına sahip m-makineli tekli parça tipi robotik hücre çizelgelemesi problemi üzerinde durulmuştur. Parçaların işlem sıraları boyunca herhangi bir makineye bir defadan fazla uğrayabileceği varsayılmış ve problemi çözmek üzere karmaşıklığı $O(K^5)$ olan bir algoritma sunulmuştur. Burada K işlem aşamalarının sayısını göstermektedir. Söz konusu kısıt gevşetildiğinde aynı algoritmanın karmaşıklığının, m makine sayısını göstermek üzere, $O(m^4)$ olduğu gösterilmiştir (Kats ve Levner, 1997). İleriki bir çalışmada, Levner, Kats ve Levit, aynı problemi yeniden-giriş kısıtları olmadan ele almış ve karmaşıklığı $O(m^3 \, log \, m)$ olan bir algoritma sunmuşlardır (Levner, Kats ve Levit, 1997).

Dawande, Sriskandarajah ve Sethi, robotik hücrelerde tekli parça tipi çizelgelemesi probleminin farklı bir durumunu dikkate almışlar; yalnızca 1-birimlik çevrimleri göz önüne alarak, m-makineli robotik bir hücrede çevrim zamanını minimize edecek optimal robot hareket dizisinin belirlenmesi üzerinde çalışmışlardır. Çalışmada literatürden farklı

olarak herhangi iki makine arasındaki robot hareket zamanının sabit olduğu varsayımı yapılmıştır. Optimal 1-birimlik çevrimi bulmak üzere polinom zamanlı bir algoritma geliştirilmiştir (Dawande, Sriskandarajah ve Sethi, 2002).

Gultekin, Akturk ve Karasan, 2-makineli, özdeş parçalı robotik hücreler üzerinde çalışmışlardır. Çalışmada, işlem ataması esnekliğinin hücredeki makinelerin CNC makineler olmasının doğrudan bir sonucu olduğu belirtilmiştir. Bazı işlemlerin sadece ilk makinede işlenebileceği, bazılarının ise sadece ikinci makinede işlenebileceği varsayılmış; kalan işlemlerin ise her iki makinede de gerçekleşebileceği belirtilmiştir. Ele alınan problem, her iki makinede de yapılabilecek olan bu işlemlerin makinelere atanması ve çevrim zamanını minimize edecek şekilde ilgili robot hareket çevriminin belirlenmesidir. Probleme çözüm olarak, optimal sonucun 1-birimlik veya 2-birimlik çevrimler olacağı kanıtlanmış ve bu robot hareket çevrimleri için optimalite aralıkları tanımlanmıştır (Gultekin, Akturk ve Karasan, 2006).

Gultekin, Karasan ve Akturk, m-makineli robotik hücreler üzerinde durmuşlardır. Ele aldıkları hücrede kullanılan makineler yüksek esnekliğe sahip CNC makineler olup bunun bir sonucu olarak, her parçanın belirli sayıda işlem gerektirdiği ve her bir makinenin bu işlemlerin tamamını gerçekleştirebilecek yetenekte olduğu belirtilmiştir. Makinelerin esnekliğinden yararlanarak yeni bir robot hareket çevrimi sınıfı tanımlamış ve bu sınıftaki çevrimlerin tüm akış tipi robot hareket çevrimlerine baskın olduğunu ispatlanmışlardır. Sonuçlar, önerilen bu çevrimlerin basitlikle birlikte oldukça pratik olduğunu ve oldukça etkin çözüm verdiğini göstermiştir. Her bir makinenin bir parçanın tüm işlemlerini gerçekleştirebileceği varsayılarak, esneklik özelliğinin yeni çevrimlerin bulunmasına imkan verdiği gösterilmiştir (Gultekin, Karasan ve Akturk, 2009).

Kats ve Levner, m-makineli üretim hattında özdeş parçaların beklemesiz ve çevrimsel olarak üretilmesinin çizelgelenmesi üzerinde çalışmışlardır. Çalışmada amaç, her bir çevrimde iki parçanın sisteme girdiği ve sistemi terk ettiği 2-birimlik çizelgelerin çevrim zamanının minimize edilmesi olarak alınmıştır. Çözüme yönelik olarak $O(m^5 logm)$ karmaşıklığında bir sezgisel algoritma önerilmiştir (Kats ve Levner, 2009).

2.4.2. Çoklu parça tipi (özdeş olmayan parçalar)

Çoklu parça tipi alanına bakıldığında, Sethi ve diğerleri, çoklu parça tipi üreten 2-makineli bir hücredeki iki mümkün çevrim ile ilişkili parça sıralama problemini çözmüştür. Çalışmada bir EAPK'nin çevrim zamanını minimize etmek amaçlanmıştır. Çevrimlerin biri için problem önemsiz olarak değerlendirilmişken, diğerinin Gezgin Satıcı Problemi (GSP)'nin özel bir durumuna denk olduğu ve Gilmore ve Gomory'nin polinom zamanlı algoritması kullanılarak optimal olarak çözülebileceği gösterilmiştir (Sethi ve diğerleri, 1992; Gilmore ve Gomory, 1964).

Stern ve Vitner, 2-makineli çoklu parça tipi problemini, maksimum tamamlanma zamanının minimize edilmesi amacıyla, makineler arası taşıma zamanlarının parçaya bağlı olduğu durum için ele almışlardır. Problemin asimetrik gezgin satıcı problemine denk ve NP-zor olduğu gösterilmiştir (Stern ve Vitner, 1990). Aynı problemde, taşıma zamanlarının parçaya bağlı olmadığı durum için, $O(n^3)$ karmaşıklığında bir algoritma Kise, Shioyama ve Ibaraki tarafından önerilmiştir (Kise, Shioyama ve Ibaraki, 1993).

Yine 2-makineli çoklu parça tipi problemi için, Logendran ve Sriskandarajah, üç farklı yerleşimi dikkate almış ve bu yerleşimler için optimal robot dizilerini belirlemiştir. Çoklu parça tiplerinin optimal sıralamasının belirlenmesi probleminin 2-makineli beklemesiz akış tipi problemine denk olduğu gösterilmiş ve söz konusu problem çözülmüştür. Tek bir EAPK'in analizinin yanı sıra birden çok EAPK'in üretilmesine yönelik analizler de verilmiştir (Logendran ve Sriskandarajah, 1996).

Hall ve diğerleri, 2- veya 3-makineli robotik hücrelerdeki işlemlerin çizelgelenmesini dikkate almışlardır. 2-makineli hücredeki çoklu parça tipi problemleri için robot hareket ve parça sıralama problemlerini eş zamanlı olarak çözen etkin bir algoritma geliştirmişlerdir. Çoklu parça tipi durumunda çevrim zamanı minimizasyonu problemi, hangi parçaların iki mümkün çevrim ($S1$ ve $S2$) arasından hangisine göre işlem göreceğinin ve optimal parça sırası da eşzamanlı olarak belirlenirken bir çevrimden diğerine nasıl geçileceğinin belirlenmesi olarak gösterilmiştir. n parça sayısını belirtmek üzere, problemi optimal olarak çözen $O(n^4)$ karmaşıklığında bir algoritma geliştirilmiş, ayrıca 3-makineli hücrelerde tekli parça tipi üretildiğinde 1-birimlik çevrimlerin 2-birim üreten daha karmaşık politikalara baskın olduğu gösterilmiştir (Hall ve diğerleri, 1997a).

Hall, Kamoun ve Sriskandarajah da bir robotun kullanıldığı üretim hücresindeki işlemlerin çizelgelenmesi üzerine çalışmıştır. Çoklu parça tipi üreten 3-makineli bir hücre için 1-birim üreten, optimal olma ihtimaline sahip altı adet robot hareket çevriminin ikisinde parça sıralama probleminin tanıma versiyonunun NP-tam yapıda olduğu ispatlanmış, 3-makineli bir hücrede herhangi bir robot hareket çevrimiyle kısıtlandırılmamış genel parça sıralama probleminin de NP-tam olduğu gösterilmiştir. Çalışmada çoklu parça tiplerinin denge durumu üretiminde hücrenin bir veya iki EAPK sonra aynı duruma döndüğü gözlenmiştir (Hall, Kamoun ve Sriskandarajah, 1997b).

Aneja ve Kamoun da Hall ve diğerleri gibi 2-makineli robotik hücrelerde uzun dönemli ortalama zamanın minimizasyonu problemini göz önüne almış ve onların algoritmasını iyileştirerek optimal robot hareket dizisi ve parça girdi sırasını bulmuşlardır. Problemi GSP'nin özel bir durumu olarak modellemiş ve karmaşıklığı $O(n\,log\,n)$ olan bir algoritma geliştirmişlerdir (Aneja ve Kamoun, 1999; Hall ve diğerleri, 1997a).

Sriskandarajah, Hall ve Kamoun, daha büyük $(m \geq 4)$ hücrelerde parça çizelgeleme probleminin karmaşıklığını belirlemek üzere kriterler geliştirmişlerdir. Çalışmada, robot hareket dizileriyle ilişkili parça sıralama problemleri aşağıdaki kategorilere göre sınıflandırılmıştır:

- Bağımsız sıralama,
- GSP olarak modellenebilen ama polinom zamanda çözülebilen,
- GSP olarak modellenebilen ve NP-zor yapıda,
- NP-zor yapıda ama GSP olarak modellenemeyen.

Bu sınıflandırmanın sonucu olarak, $m!$ adet mümkün robot hareket çevriminin $2m-2$ adediyle ilişkili parça sıralama probleminin polinom zamanda çözülebilir olduğu gösterilmiştir. Geri kalan çevrimler ise NP-zor yapıda parça sıralama problemleri içermektedir (Sriskandarajah, Hall ve Kamoun, 1998).

Sriskandarajah ve diğerleri, 3-makineli çoklu parça tipi probleminin zorlu olduğunu göstermişlerdir (Sriskandarajah ve diğerleri, 1998). Bundan dolayı, Kamoun ve diğerleri, benzer parçalardan oluşan ürün ailelerinin tekrarlı olarak üretildiği ve makinelerin akış tipi

üretim sistemi şeklinde yerleştirildiği, robot kullanılan bir üretim hücresinde meydana gelen çizelgeleme problemlerini dikkate almışlardır. Çalışmada genel 3- ve 4-makineli hücreler için birkaç sezgisel yöntem geliştirilmiş ve test edilmiştir. Ayrıca robot hareket dizisi ve parça sırasının belirlenmesi genel problemine yönelik etkin sezgiseller tanımlanmış ve test edilmiştir. Amaç fonksiyonu olarak EAPK'lerin tekrarlı üretimi için ortalama denge durumu çevrim zamanının minimizasyonu alınmıştır. Çalışmada ayrıca makinelerin hücreler içinde gruplandırılması, iyi parça sıralarının belirlenmesi ve hücreler arasında uygun stok büyüklükleri belirlenmesine dayalı olarak robotik hücrelerin etkin performans göstermesinin nasıl sağlanabileceği üzerinde durulmuş; bu yaklaşımların daha büyük hücrelere genişletilmesine yönelik öneriler sunulmuştur (Kamoun ve diğerleri, 1999).

Kamalabadi, Gholami ve Mirzaei, 3-makineli robotik hücrede çevrimsel çoklu parça-tipi probleminin çözümü için Parçacık Sürü Optimizasyonu algoritması uygulamışlardır (Kamalabadi, Gholami ve Mirzaei, 2007).

2.4.3. Paralel makineler

Bazı araştırmacılar tüm makinelerin benzer olup paralel olarak çalıştığı, robotik veya bir servis sağlayıcının kullanıldığı hücrelerdeki çizelgeleme ve sıralama problemleri üzerinde çalışmışlardır. Bu araştırmacılar, benzer yapıda makinelere sahip olduğundan, bir işin makinelerden (çoğunlukla iki makineden) herhangi birinde işlem görebileceğinin varsayıldığı sistemlerde, hazırlık işlemine ihtiyaç duyan işlerin çizelgelenmesi problemini ele almışlardır. Tekli veya çoklu parça tipi problemleri tanımı yapma gereği duymamış; farklı hazırlık veya işlem zamanı kısıtlarını dikkate alarak algoritmalar geliştirmeye veya problem karmaşıklıklarını tanımlamaya çalışmışlardır.

Çeşitli araştırmacılar, her işin kendi işlemlerine ek olarak, tek bir servis sağlayıcı tarafından sağlanacak bir hazırlık işlemine ihtiyaç duyduğu paralel makineli sistemler üzerinde çalışmış ve problem karmaşıklıklara dair sonuçlar sunmuşlardır. Koulamas ve Smith, ortak servis sağlayıcının bulunduğu, sürekli parça gelişi olan paralel makineli çizelgeme problemi için ileriye dönük bir sezgisel yöntem geliştirmişlerdir (Koulamas ve Smith, 1988). Koulamas, servis sağlayıcının uygun olmamasına bağlı olarak bazı makinelerin boş kalması şeklinde tanımlanan zorunlu boş zamanı minimize etmeye

çalıştıkları problemin, 2-makineli ve tek servis sağlayıcıya sahip paralel sistemlerde NP-zor olduğunu göstermiş; yerel arama ve demet araması algoritmaları sunmuşlardır (Koulamas, 1996). Morton ve Pentico da ortak ancak harici bir servis sağlayıcı problemini dikkate almışlardır (Morton ve Pentico, 1993).

Kravchenko ve Werner, tüm hazırlık zamanlarının bire eşit olduğu 2-makineli durum için pseudo-polinom zamanlı bir algoritma geliştirmiş ve problemin büyük oranda NP-tam olduğunu göstermişlerdir (Kravchenko ve Werner, 1997). Daha genel bir durumda, Hall, Potts ve Sriskandarajah, bir robotun parça değiştirme ve parça hazırlık amaçlarıyla birden fazla ekipman arasında paylaşıldığı problem için yaygın bazı çizelgeleme amaçları doğrultusunda karmaşıklık analizleri sunmuşlardır. Aynı çalışmada, maksimum tamamlanma zamanı çizelgelemesine yönelik iki sezgisel analiz edilmiştir (Hall, Potts ve Sriskandarajah, 2000).

Abdekhodaee ve Wirth, eşit süreli işlerin bulunduğu durumu bir sıralama algoritması kullanarak çözmüşlerdir. Çalışmada, genel durum için birkaç sezgisel önerilmiş ve bu yöntemlerin performansları, yayınlanmış sonuçlarla karşılaştırılmıştır (Abdekhodaee ve Wirth, 2002). Diğer bir çalışmada, Abdekhodaee, Wirth ve Gan, iki özel durumun, eşit işlem zamanı ve eşit hazırlık zamanı problemlerinin, karmaşıklığını göz önüne almış ve bu durumlar için farklı sezgisellerin performansını test etmişlerdir (Abdekhodaee, Wirth ve Gan, 2004). Yine bir başka çalışmada, Abdekhodaee, Wirth ve Gan, ilk işlemi (veya hazırlık işlemini) gerçekleştirmek üzere tek bir servis sağlayıcının bulunduğu iki işlemli, bölünemeyen işlerden oluşan belirli bir iş kümesinin üretiminin, maksimum tamamlanma zamanını minimize edecek şekilde, yarı-otomatik 2-makineli sistemde çizelgelenmesi problemini dikkate almışlardır. Genel durum ile ilgilenmek amacıyla, düzgün problemlerin çalışıldığı geçmiş çalışmalar üzerinde durulmuştur. Bu çalışmanın sonuçları, geçmiş çalışmalarla birlikte genel duruma yönelik geniş sezgisel çözümler sağlıyor görünmektedir (Abdekhodaee, Wirth ve Gan, 2006).

Benzer bir hücre yapısı, Geismar, Dawande ve Sriskandarajah tarafından sunulan çalışmada kullanılmış ve tek bir robotun bulunduğu çok aşamalı robotik hücreler üzerinde durulmuştur. Her parça her aşamadaki paralel makinelerden herhangi birinde işlem görmekte ve her parça aşamalar arasında aynı sırayı takip etmektedir. Çalışmada özdeş parçalar üzerinde durulmuş ve robot hareketlerinin belirlenmesi amaçlanmıştır. Yazarlar,

önemli ve sezgisel bir çevrim sınıfının varlığını tespit etmiş ve çevrim zamanları için genel bir ifade oluşturmuş, bunun yanı sıra hücrenin belirli bir çıktıyı karşılayabilmesi için her aşamada ihtiyaç duyduğu makine sayısını belirleyen bir formül geliştirmişlerdir. Ayrıca, paralel makine kullanımının gereksiz masrafa yol açtığı durumlar belirlenmiştir (Geismar, Dawande ve Sriskandarajah, 2004).

Elmi ve Topaloğlu, hibrit akış tipindeki robotik hücrelerde birden fazla robot kullanımını incelemiş; çoklu parça tipi ve özdeş olmayan paralel makinelerin bulunduğu sisteme yönelik çizelgeleme çalışmasında öncelikle bir matematiksel model önermiş, problem karmaşıklığı sebebiyle TB tabanlı bir sezgisel oluşturarak çözüm elde etmişlerdir. Ele aldıkları sistem, çalışmamız ile benzerlik göstermekle birlikte; esneklik imkanı dikkate alınmamış, birden fazla robotun bulunması sayesinde robot hareketlerinin optimize edilmesi problemi üzerinde çalışılmamış, ayrıca sezgisel yöntemin çözüm değerlendirmelerinde alt sınır değeri yerine çalışmada önerilen algoritmanın iki versiyonunun karşılaştırması yapılmıştır (Elmi ve Topaloğlu, 2013).

Çizelge 2.2. Robotik hücre literatürü

Tekli Parça Tipi (Özdeş Parçalar)	Çoklu Parça Tipi (Özdeş Olmayan Parçalar)	Paralel Makineler
- Sethi ve diğerleri, 1992 - Crama ve Van de Klundert, 1997 - Kats ve Levner, 1997 - Levner ve diğerleri, 1997 - Hall ve diğerleri, 1997a - Crama ve Van de Klundert, 1999 - Brauner ve Finke, 1999 - Brauner ve Finke, 2001 - Dawande ve diğerleri, 2002 - Gultekin ve diğerleri, 2006 - Gultekin ve diğerleri, 2009 - Kats ve Levner, 2009	- Sethi ve diğerleri, 1992 - Stern ve Vitner, 1990 - Kise ve diğerleri, 1993 - Logendran ve Sriskandarajah, 1996 - Hall ve diğerleri, 1997a - Hall ve diğerleri, 1997b - Aneja ve Kamoun, 1999 - Sriskandarajah ve diğerleri, 1998 - Kamoun ve diğerleri, 1999 - Kamalabadi ve diğerleri, 2007	- Koulamas ve Smith, 1988 - Koulamas, 1996 - Morton ve Pentico, 1993 - Kravchenko ve Werner, 1997 - Hall ve diğerleri, 2000 - Abdekhodaee ve Wirth, 2002 - Abdekhodaee ve diğerleri, 2004 - Geismar ve diğerleri, 2004 - Abdekhodaee ve diğerleri, 2006 - Elmi ve Topaloğlu, 2013
Literatür Araştırmaları	- Lee ve diğerleri, 1997 - Crama ve diğerleri, 2000 - Dawande ve diğerleri, 2005 - Brauner, 2008	

Çizelge 2.2’de robotik hücre başlığı altında yapılmış olan çalışmalar verilmiştir. Bu çalışmada hibrit akış tipi tanımı ile birlikte iki aşamalı ve aşamaların birinde birbirine paralel çalışan iki makinenin bulunduğu hücrede çoklu parça tipi üretimi üzerinde durulmaktadır.

3. ELE ALINAN SİSTEM

Bu çalışmada çoklu parça tipi üretimi yapılan ve makineler/aşamalar arasındaki taşımanın bir robot tarafından gerçekleştirildiği, sabit hareket zamanlı hibrit esnek akış tipi hücrelerde gözlenen çizelgeleme problemi üzerinde durulmaktadır.

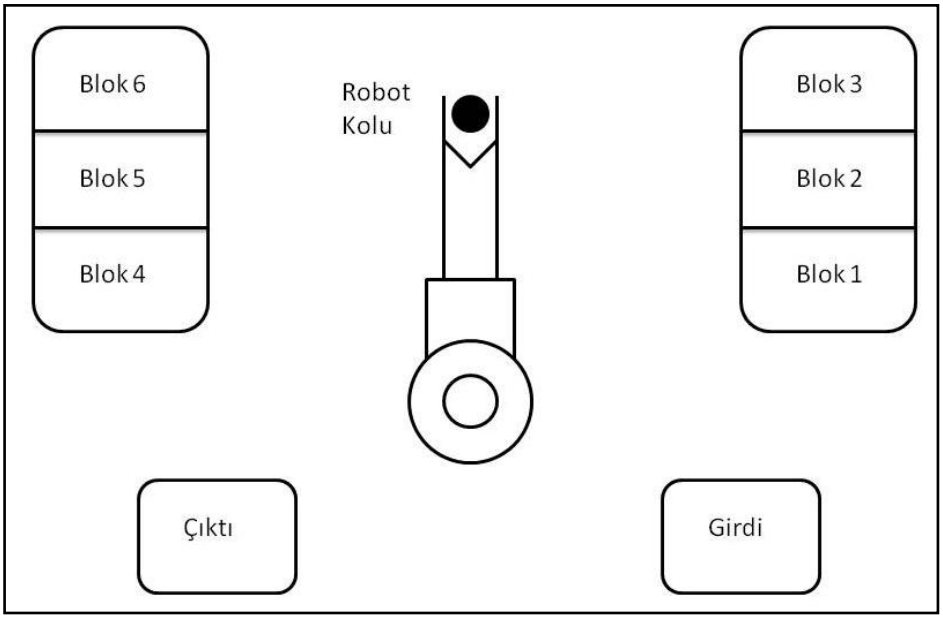
Sistemde, parçalar, çok aşamalı bir üretim işlemine ihtiyaç duymakta; söz konusu aşamalardaki farklı işlem yüklerinin dengelenebilmesi veya darboğaz aşamaların varlığının önüne geçilebilmesi için, birbirine paralel olarak çalışan makinelerden yararlanılmaktadır. Sabit hareket zamanlı robotik hücre tanımıyla birlikte, ele alınan sistemde herhangi iki makine arasındaki hareket zamanları, farklı iki makine arası hareket zamanları ile çok değişiklilik göstermemekte, yaklaşık olarak eşit kabul edilmektedir.

Robotik hücrelerde, makineler ve robot, hücre denetçisinin yardımıyla, gerçek zaman ekseninde etkileşim halindedir. Robotik bir hücrede aranılan çözüm, robot hareketlerini tam olarak tanımlayacak şekilde olmalıdır. Bu, taşınmakta, yüklenmekte veya boşaltılmakta olan parçayla birlikte ilgili makinenin de bilinmesini gerektirmektedir. Robotik hücre ve hibrit esnek akış tipi yapılarının tümü bir arada dikkate alındığında, problemdeki zorluk üç yönlü olmaktadır: (a) parça sıralamasının belirlenmesi, (b) parçaların makinelere atanması, (c) robot hareket dizisinin belirlenmesi.

Çalışmada ele alınan hücre, 2 aşamadan oluşan esnek akış hattına göre düzenlenmiştir. Oluşturulan sistemde hatsal robotik hücre yerleşimi kullanılmış ve ilk aşamada yalnız bir, ikinci aşamada ise iki paralel özdeş makinenin bulunduğu varsayılmıştır.

Hücrede; girdi deposundan alınan parça, makinelerin durumu ve sistemin yapısına uygun olarak, robot tarafından ilgili makine(ler)e taşınmakta ve bu makine(ler)de işlem gördükten sonra yine robot tarafından çıktı deposuna götürülmektedir. Parçalar birinci aşamadan ikinci aşamaya doğru hareket etmekte, ancak, esneklik tanımı ile birlikte, tüm aşamalara uğrama zorunluluğu taşımamaktadır. 2-aşamalı esnek bir akış hattında, hangi aşamalarda işlem göreceklerine bağlı olarak üç tip parça vardır. Parçaların, virgüllerle birbirinden ayrılmış olarak gösterilen, şu rotalardan birini izlemesi mümkündür: $1,\ 2,\ 1-2$. Bir parçanın, ikinci aşamaya gideceği bilindiğinde, aşamadaki iki makineden hangisinde işlem göreceğine karar verilmesi gerekmektedir.

Çalışmada sabit hareket zamanlı robotik hücreler ele alınmaktadır. Bu tip hücrelerde robot, değişken ivmeyle hareket etmekte ve herhangi iki makine arasındaki hareket zamanı sabite yakın değer almaktadır. Bu hücre yapısı ilk olarak Dawande ve diğerleri tarafından ele alınmış ve Şekil 3.1'deki yerleşim planına sahip bir robotik hücre üzerinde çalışılmıştır. Bu hücrede makineler, aralarındaki ikili uzaklıklar yaklaşık olarak eşit olacak şekilde yerleşmiştir. Şekilde görülen her bir blokta 6 adede kadar makine bulunmakta, bu makineler dikey olarak yığılmaktadır (Dawande ve diğerleri, 2002).



Şekil 3.1. Sabit hareket zamanlı hücre yapısı

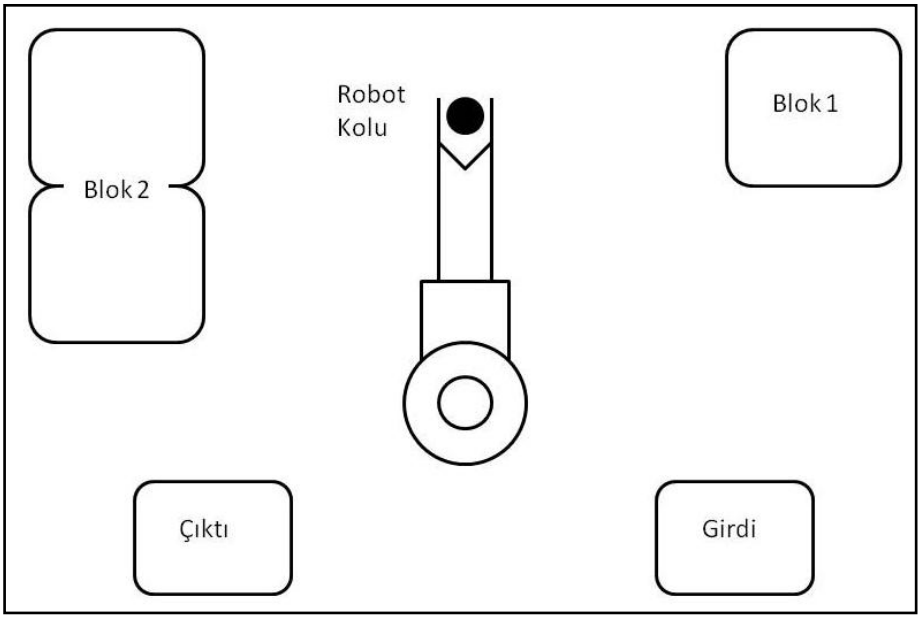
Robot tarafından herhangi bir makine çifti arasında hareket edilirken izlenen hareket yörüngesi üç boyutta karmaşık bir yol oluşturmakla birlikte şu özellikler ile karakterize edilmektedir:

1. Herhangi iki makine arasındaki robot hareket ivmesinin artış/azalış hızları, farklı hareket uzaklıkları için farklılık göstermektedir. Bu hız, robotun hareketi süresince değişmekte ve fiziksel yerleşim veya güvenlik gibi etkenlere bağlı olarak değer almaktadır. Genel olarak, robotun herhangi iki makine arasındaki ortalama hızı aradaki uzaklık arttıkça artmaktadır.
2. Robotun ulaşabileceği bir maksimum hız değeri bulunmaktadır. Ancak, hücre yerleşimi karmaşık bir yapıya sahip olduğundan çok az sayıda makine çifti arasındaki hareketlerde bu maksimum hız değerine ulaşılabilmekte ve bu hıza ulaşıldığında da ancak çok kısa süre için aynı hızın korunması mümkün olmaktadır.

Bu özelliklerin sonucu olarak, makine çiftleri arasındaki hareket zamanlarının kabaca eşit olduğu bulunmuştur.

Makineler arası hareket zamanları ile ilgili iki varsayımın, toplamsal hareket zamanlı ve sabit hareket zamanlı modellerin, her ikisi de aslında gerçek hareket zamanlarına birer yaklaşım olarak ele alınmakta; sabit hareket zamanlı yaklaşım ise bazı robotik hücrelere daha uygun özellik göstermektedir (Dawande ve diğerleri, 2002).

Üzerinde durulan hücre yapısını göstermek üzere Şekil 3.2'de verilen robotik hücrede 2 aşamaya karşılık olarak 2 blok bulunmakta; ilk blokta yalnız 1, ikinci blokta 2 adet makine görülmektedir.



Şekil 3.2. Ele alınan hücre yapısı

Çalışmada kullanılan hücre yapısı Şekil 3.1'de verilen hücre yapısına benzemekle birlikte, içerisinde farklı özellikler barındırmaktadır. Daha önce de belirtildiği üzere, ele alınan yapıda özdeş parça tipi yerine çoklu parça tipi üretimi üzerinde durulmuştur. Bu kullanımın sonucu olarak parça sıralamasının belirlenmesi problemi ile karşı karşıya kalınmaktadır. Çalışmanın bir diğer önemli noktası, esneklik tanımıyla birlikte bazı parçaların bazı aşamalarda işlem görmeme olanağı olmasıdır. Böylece her parçanın farklı işlem gereksinimleri ortaya çıkmakta ve bu gereksinimler dikkate alınarak çözüm bulunmaya çalışılmaktadır. Bunun yanı sıra, parçaların işlem göreceği aşamalarda birden

çok özdeş makine bulunması ile birlikte makineler arasında seçim yapılması gündeme gelmekte, parçaların makinelere en uygun şekilde atanması gerekmektedir.

4. MATEMATİKSEL MODEL YAPISI VE VARSAYIMLARI

Problem uzaklık matrisinin parametre değerlerinin yanı sıra karar değişkenlerinden oluştuğu, Gezgin Satıcı Problemi (GSP)'nin özel bir durumu olarak formüle edilmiştir. Ele alınan sistemin aşağıdaki özellikleri taşıdığı varsayılmaktadır:

- Hücrede 2 işlem aşaması bulunmaktadır. Birinci aşamada yalnız bir makine, ikinci aşamada özdeş yapıda 2 makine mevcuttur.
- İşlem görecek parçalar özdeş değildir. Herhangi bir j parçasının k. aşamada işlem göreceği süre P_{jk}'dır.
- Bir parçanın bazı aşamalarda işlem görmemesi mümkün olmakla birlikte her parça en az bir aşamada işlenmek zorundadır. Ancak, tüm parçalar aynı üretim akışını izlemektedir: 1. aşama, 2. aşama. Herhangi bir j parçasının k. aşamada işlem görmemesi durumunda o aşamaya ait işlem zamanı, $P_{jk} = 0$ olmaktadır.
- Bir makine aynı anda yalnız bir parçanın işlemini gerçekleştirebilmekte ve bir parça yalnız bir makinede işlem görebilmektedir.
- Hazırlık zamanları ihmal edilmekte, iş keserek öncelik vermeye ve aşamalar arası stoğa izin verilmemektedir.
- Problem verileri deterministik yapıda olup kesin olarak bilinmektedir.
- Tüm parça ve makineler $t = 0$ anında hazır durumdadır.

Formülasyonda kullanılan tanımlardan biri düğüm terimidir.

Tanım 1. i_k düğümü i parçasının k istasyonunda olması durumunu belirtmektedir. Girdi deposunun 1 istasyonu ve çıktı deposunun 5 istasyonu olarak adlandırıldığı sistemde $2, 3$ ve 4 sırasıyla birinci aşamada bulunan 1. makine ile ikinci aşamaya ait 2. ve 3. makineleri göstermektedir. Bir EAPK çevrimi süresince, robot bir parçayı yükledikten sonra işlemin tamamlanmasını beklemek yerine başka aktiviteleri gerçekleştirmeyi tercih edebileceğinden, makinelerin aynı parça için, biri yükleme diğeri boşaltma için olmak üzere iki defa ziyaret edilmesi gerekebilmektedir. Bu nedenle, olası çevrimleri tanımlamak üzere, 6, 7 ve 8 istasyonları da, sırasıyla, 2, 3 ve 4 istasyonlarının kopyası olarak oluşturulmuştur.

Formülasyon için N düğüm kümesi ve A ark kümesi oluşturulmuştur.

N={i_k; i=1,2,…,n, k=1,2,…8}

A={$i_k j_l$; $i_k, j_l \in N$ ve i parçasının k istasyonunda bulunduğu i_k düğümünden j parçasının l istasyonunda bulunduğu j_l düğümüne olan hareket mümkün}

Robot hareketleri düğümler arasında tanımlanmıştır. İki tip hareket vardır: birincisinde parça bir istasyondan diğerine taşınmakta, ikincisinde robot bir istasyona parçayı bırakmakta ve başka bir parçayı almak üzere diğer bir istasyona gitmektedir. Çalışmada k' notasyonu k düğümünün kopyasını göstermek üzere kullanılmaktadır. Çizelge 4.1'de bu dönüşümün görülmesi mümkündür.

Çizelge 4.1. k istasyonları ile k' istasyonları arası ilişki

k	k'
2	6
3	7
4	8
6	2
7	3
8	4

Problemin 8 istasyonun her biri için bir düğümün oluşturulduğu bir GSP olarak formüle edilmesi için aşağıdaki parametre ve değişkenler kullanılmaktadır:

Parametreler:

- n : EAPK'de üretilecek toplam parça sayısı
- $P_{i1(2)}$: Üretilecek parçaların $1.(2.)$ aşamaya ait işlem zamanları, $i = 1,2,\dots,n.$
- ε : Robotun makinelere yükleme / boşaltma zamanı
- δ : İki makine arasındaki robot hareket zamanı
- $cost_{i_k j_l}$: i parçasının k istasyonunda olduğu durumdan j parçasının l istasyonunda olduğu duruma geçişle ilgili robot hareketi için gerekli toplam zaman, toplam maliyet
- $wait_{i_k j_l}$: $\begin{cases} 1 & \text{robot hareketinde bekleme zamanı gözlenmesi olasılığı varsa,} \\ 0 & \text{diğer durum} \end{cases}$

Karar değişkenleri:

- $y_{i_k j_l}$: $\begin{cases} 1 & \text{robot } i_k \text{ düğümünden } j_l \text{ düğümüne gidiyorsa,} \\ 0 & \text{diğer durum} \end{cases}$
- P'_{i_k} : i parçasının k istasyonundaki işlem zamanı, *k=2,3,4,6,7,8*
- s_{i_k} : i parçasının işleminin k istasyonunda başladığı zaman
- c_{i_k} : Robotun i_k düğümüyle ilgili aktivitesinin tamamlanma zamanı
- w_{i_k} : i parçası için k istasyonu önünde robot tarafından beklenilen zaman
- z_{i_k} : $\begin{cases} 1 & \text{bir çevrim içerisinde, } i \text{ parçasının } k \text{ istasyonundaki} \\ & \text{başlama zamanı, parça işleminin bitiş zamanından} \\ & \text{önce olarak değerlendiriliyorsa,} \\ 0 & \text{diğer durum} \end{cases}$
- $m2_{i_k}$: Robot i_k düğümündeyken 2. istasyonda yüklenmiş olan parça
- $m3_{i_k}$: Robot i_k düğümündeyken 3. istasyonda yüklenmiş olan parça
- $m4_{i_k}$: Robot i_k düğümündeyken 4. istasyonda yüklenmiş olan parça

  $m2_{i_k}$, $m3_{i_k}$, $m4_{i_k}$ değişkenlerinin her üçü de $\{0,1,\dots,n\}$ kümesinden değerler almaktadır. Değişkenin 0 değerini alması istasyonun o anda boş olduğunu, 1 ile n arası değerler alması ise istasyonun i_k düğümünde o parça ile yüklenmiş durumda olduğunu göstermektedir.
- C : Çevrim zamanı değeri

Bu problem için bir parça ile yüklenmiş durumda olan bir makinenin yeniden yüklenemeyeceği ve boş durumda olan bir makinenin boşaltılamayacağını belirten fizibilite şartlarının sağlanması gerekmektedir. Ayrıca, bazı hareketlerin yapılması gereksizdir; örneğin, bir parça girdi deposundan alınıp hiçbir işlem görmeden çıktı deposuna götürülmesi, çıktı deposundan parça alınması veya girdi deposuna parça bırakılmasına izin verilmemektedir. Tüm olası hareketler bu gözlemlere göre belirlenmekte ve çevrimdeki bazı hareketler yasaklanmaktadır.

Parametre olarak tanımlanan $cost_{i_k j_l}$ değerleri robotun i_k düğümünden j_l düğümüne gitmesi sırasında harcadığı toplam zamanı göstermektedir. Örneğin, i_1 düğümünden i_2 düğümüne gidiş; robotun girdi deposundan bir parça alması (ε), bu parçayı 2. istasyona yani 1. makineye taşıması (δ) ve buraya yüklemesi (ε) durumuna karşılık gelmektedir ki bu durumda toplam zaman $2\varepsilon + \delta$ olmaktadır. Daha önce de belirtildiği üzere, i ve j

parçalarının eşitliğine bağlı olarak iki farklı durumun temsil edilmesi mümkündür; bundan dolayı $cost_{i_k j_l}$ değerleri de farklılık göstermektedir. Toplam hareket zamanları, ilk çizelge $i = j$ durumuna, ikinci çizelge ise $i \neq j$ durumuna karşılık gelmek üzere, Çizelge 4.2 ve 4.3'te verilmiştir.

Çizelge 4.2. Aynı parçaya yönelik hareketlere ait toplam zaman

	i_1	i_2/i_6	i_3/i_7	i_4/i_8	i_5
i_1	x	$2\varepsilon + \delta$	$2\varepsilon + \delta$	$2\varepsilon + \delta$	x
i_2/i_6	x	x	$2\varepsilon + \delta$	$2\varepsilon + \delta$	$2\varepsilon + \delta$
i_3/i_7	δ	x	x	x	$2\varepsilon + \delta$
i_4/i_8	δ	x	x	x	$2\varepsilon + \delta$
i_5	δ	x	x	x	x

Çizelge 4.3. Farklı parçalara yönelik hareketlere ait toplam zaman

	j_1	j_2/j_6	j_3/j_7	j_4/j_8	j_5
i_1	x	x	x	x	x
i_2/i_6	δ	x	δ	δ	x
i_3/i_7	δ	δ	x	δ	x
i_4/i_8	δ	δ	δ	x	x
i_5	δ	δ	δ	δ	x

Çizelge 4.2'ye bakıldığında, sistemin esnekliği dolayısıyla mümkün kılınmış bazı hareketler dikkat çekmektedir. Akış tipi hücrelerde sisteme parça girişi ilk aşamadan olmaktayken esnek akış tipinde parçanın işlemine ilk aşama atlanarak ikinci (veya daha sonraki) bir aşamada başlanması seçeneği bulunmaktadır. Bu nedenle 1. istasyondan 2. (6.) istasyonlara parça taşıması olduğu gibi, 1. istasyondan 3. (7.) veya 4. (8.) istasyonlara da parça taşıması olabilmektedir. Benzer şekilde, çıktı deposuna parça gidişinin yalnız son aşamadan çıktı deposuna doğru bir taşımayla mümkün görülmesi yerine, ele alınan sistemde ilk aşamada tüm işlemi tamamlanan bir parçanın da direkt olarak çıktı deposuna götürülmesi olası bir hareket halini almaktadır. Bunun sonucu olarak da 2. (6.) istasyondan 5. istasyona doğru parça hareketine izin verilmiştir.

Çizelgelerde 'x' ile gösterilmiş olan hareketler gerçekleştirilmesi mümkün olmayan hareketlerdir. Örneğin, $y_{i_2 i_1}$ hareketi i parçasının 2. istasyondan alınıp 1. istasyona yani girdi deposuna bırakılmasını gerektirmektedir ki böyle bir durumun olması söz konusu değildir.

Makinede yüklü olan parçanın işlemi tamamlanmamasına rağmen, robot, makineyi boşaltmak üzere geldiğinde gözlenen bekleme zamanı, çevrim zamanı hesaplamalarındaki ana bileşenlerden birisidir. Bu değer robot makinenin önüne geldiği anda parçanın işlemi tamamlanmışsa sıfıra; aksi takdirde kalan işlem zamanına eşit olmaktadır. Şu şekilde gösterilmesi mümkündür:

$$w_{i_k} = maks\{0,\ P'_{i_k} - v_{i_k}\} \quad \forall i,k \tag{4.1}$$

Burada i, k istasyonunda yüklü durumda olan parçayı; v_{i_k} ise robotun parçayı k istasyonu olarak ifade edilen makineye yüklemesiyle aynı makineye parçayı almak amacıyla tekrar gelmesi arasında yaptığı hareketlerin toplam zamanını göstermektedir.

$wait_{i_k j_l}$ değerleri hangi hareketler için robotun k istasyonunda i parçasını beklemesi gerekebileceğini görmek amacıyla kullanılmaktadır. Tahmin edileceği üzere, farklı parçalara ait hareketlerde ($i \neq j$ olduğunda) bekleme zamanı gözlenmeyecektir; bu nedenle sadece $i = j$ durumları tanımlanmıştır. Bekleme zamanı olasılıkları Çizelge 4.4'te verilmiştir. Örneğin, i_2 düğümünden i_3 düğümüne gidileceğinde i parçası 2. istasyondan alınıp 3. istasyona yüklenecektir ki bu durumda parçanın 2. istasyondaki işlemi tamamlanana kadar beklenmesi gerekebilmektedir.

Çizelge 4.4. Aynı parçaya yönelik hareketlere ait bekleme zamanı olasılıkları

	i_1	i_2/i_6	i_3/i_7	i_4/i_8	i_5
i_1	-	-	-	-	-
i_2/i_6	-	-	*1*	*1*	*1*
i_3/i_7	-	-	-	-	*1*
i_4/i_8	-	-	-	-	*1*
i_5	-	-	-	-	-

Tanımlanan parametre ve değişkenlerle bir matematiksel model oluşturulmuştur. Öncelikle y değişkenlerinin uygun değerler almasına çalışılmaktadır. Bir düğüme giren bir ark olduğunda bu düğümden çıkan bir ark da olması sağlanmalıdır. Bu şart Eş. (4.2) ile sağlanmıştır.

$$\sum_{j_l:(j_l,i_k)\in A} y_{j_l i_k} = \sum_{j_l:(i_k,j_l)\in A} y_{i_k j_l}, \quad \forall i_k \in N \tag{4.2}$$

Modelde bazı düğümlerin kullanılmamasına izin verilmektedir. Bu nedenle GSP'nin atama kısıtları Eş. (4.3)'teki şekli almıştır.

$$\sum_{j_l:(i_k,j_l)\in A} y_{i_k j_l} \leq 1, \quad \forall i_k \in N \quad \text{ve} \quad \sum_{j_l:(j_l,i_k)\in A} y_{j_l i_k} \leq 1, \quad \forall i_k \in N \tag{4.3}$$

Bu problemde sistemdeki tüm parçaların işlem görmesi gerekmektedir. Bu da Eş. (4.4) ile gösterildiği üzere, her bir parçanın girdi deposundan tam olarak bir defa alınması ve çıktı deposuna tam olarak bir defa bırakılmasını gerektirmektedir.

$$\sum_{j_l} y_{i_1 j_l} = 1 \quad \forall i \quad \text{ve} \quad \sum_{j_l} y_{j_l i_5} = 1 \quad \forall i \tag{4.4}$$

Söz konusu sistem çok aşamalı olup, esneklik özelliği ile birlikte girdi deposundan herhangi bir aşamaya gidebileceği gibi herhangi bir aşamadan da çıktı deposuna gidebilme imkanı bulunduğundan, tüm parçaların girdi deposundan alınıp çıktı deposuna bırakılması gerekliliğini gösteren Eş. (4.5) şu şekilde yazılabilmektedir:

$$\begin{aligned} &y_{i_1 i_2} + y_{i_1 i_6} + y_{i_1 i_3} + y_{i_1 i_7} + y_{i_1 i_4} + y_{i_1 i_8} = 1 \quad \forall i \\ &y_{i_2 i_5} + y_{i_3 i_5} + y_{i_4 i_5} + y_{i_6 i_5} + y_{i_7 i_5} + y_{i_8 i_5} = 1 \quad \forall i \end{aligned} \tag{4.5}$$

Aşamalar arasındaki taşımayı tanımlamak üzereyse Eş. (4.6) oluşturulmuştur.

$$y_{i_2 i_3} + y_{i_2 i_7} + y_{i_6 i_3} + y_{i_6 i_7} + y_{i_2 i_4} + y_{i_2 i_8} + y_{i_6 i_4} + y_{i_6 i_8} \leq 1 \quad \forall i \tag{4.6}$$

Genelliği kaybetmeden, 1_1 düğümü problemin başlangıç noktası olarak alınmıştır. Yani, sistemin robotun 1 numaralı parçayı almak üzere girdi deposunun önüne gelmesiyle başladığı varsayılmaktadır. Bunun sağlanması için Eş. (4.7) kullanılır.

$$c_{1_1} = 0 \tag{4.7}$$

Eş. 4.8'de Miller-Tucker-Zemlin tipi kısıtlar düğümlerin tamamlanma zamanlarının hareketlere, maliyet ve bekleme zamanlarına göre hesaplanmasına yardımcı olacak şekilde kullanılmaktadır (Miller, Tucker ve Zemlin, 1960). Burada Eş. (4.7)'ye bağlı olarak, $j_1 \neq 1_1$'dir.

$$c_{j_l} \geq c_{i_k} + cost_{i_k j_l} \cdot y_{i_k j_l} - M\left(1 - y_{i_k j_l}\right) + wait_{i_k j_l} \cdot w_{i_k} \quad \forall i_k, j_l \tag{4.8}$$

Eş. (4.8), i_k düğümünden j_l düğümüne doğru bir ark kullanıldığında j_l düğümünün tamamlanma zamanının i_k düğümünün tamamlanma zamanı, i_k düğümünden j_l düğümüne gitmenin maliyeti ve k istasyonundaki i parçası için bir bekleme varsa bu beklemeye ait bekleme zamanının toplamı ile belirlendiğini göstermektedir.

Eş. (4.9), bir parçanın her bir aşamaya ait tüm işleminin tamamlanması gerektiğini belirtmektedir.

$$P'_{i_2} = P_{i1} \quad \text{ve} \quad P'_{i_3} + P'_{i_4} = P_{i2} \quad \forall i \tag{4.9}$$

Burada $2-6$, $3-7$ ve $4-8$ istasyonlarının aynı olmasından dolayı, bu istasyon çiftlerine atanan işlem zamanları da aynı olmaktadır. Bu amaçla, $P'_{i_2} = P'_{i_6}$, $P'_{i_3} = P'_{i_7}$ ve $P'_{i_4} = P'_{i_8}$ eşitlikleri dikkate alınmaktadır.

Makinelerle işlem zamanları arasındaki ilişkinin tanımlanması için Eş. (4.10) ve Eş. (4.11) kullanılmaktadır. İşlemler girdi ve çıktı depolarında değil, yalnızca makinelerde gerçekleştiğinden bu eşitsizlik yalnızca istasyonlar 2, 3 ve 4 için oluşturulmuştur.

$$P'_{i_k} \leq P_{i_1} * \sum_{j_l:(i_k,j_l)\in A \; veya \; (i_{k'},j_l)\in A} (y_{i_k j_l} + y_{i_{k'} j_l}) \quad \forall i_k \in N \quad s.t. \quad k = \{2\} \tag{4.10}$$

$$P'_{i_k} \leq P_{i_2} * \sum_{j_l:(i_k,j_l)\in A \; veya \; (i_{k'},j_l)\in A} (y_{i_k j_l} + y_{i_{k'} j_l}) \quad \forall i_k \in N \quad s.t. \quad k = \{3,4\} \tag{4.11}$$

Eş. (4.10) ve Eş. (4.11), bir turda i_k düğümü kullanıldığında istasyon k'da, en fazla i parçasının toplam işlem zamanına eşit olacak kadar bir değerde işlem yapılabileceğini göstermektedir. Ayrıca, düğüm i_k bir çözümde kullanılmamışsa sağ taraf değeri sıfır olmakta ve istasyon k'da işlem yapılmamakta, yani $P'_{i_k} = 0$ olmaktadır.

s_{i_k} ve c_{i_k} değişkenleri, bir istasyondaki işlemin başlangıcı ve ilgili hareketin düğümde gerçekleştirildiği zaman olarak tanımlanmıştır. İşlemin başlangıç zamanı, parçanın ilgili istasyona yüklendiği zamana eşittir. Bu değerin, parçanın istasyon k veya k''ne yüklenmiş

olmasına bağlı olarak, s_{i_k} veya $s_{i_{k'}}$ ile gösterilmesi mümkündür. Bu iki istasyon birbirinin kopyası olduğundan Eş. (4.12)'deki sonuç elde edilmektedir:

$$s_{i_k} = s_{i_{k'}} \quad \forall i_k \in N \quad s.t. \quad k = \{2,3,4\} \tag{4.12}$$

Yukarıdaki açıklamalardan anlaşılacağı üzere, s_{i_k} değerleri, c_{i_k} değişkenlerine göre belirlenmektedir. Bu hesaplamalarda Eş. (4.13)-(4.14) kullanılmakta ve $l \neq 5, k \neq l, k' \neq l', k \neq l'$ olmaktadır.

$$s_{i_k} \geq c_{i_k} - M\left(1 - y_{i_l i_k}\right), \quad \forall i_k \in N, \quad s.t. \quad k = \{2,3,4\} \tag{4.13}$$

$$s_{i_k} \geq c_{i_{k'}} - M\left(1 - y_{i_l i_{k'}}\right), \quad \forall i_k \in N, \quad s.t. \quad k = \{2,3,4\} \tag{4.14}$$

Bu eşitsizliklere göre, i_l düğümünden i_k veya $i_{k'}$ düğümüne bir ark varsa, yani i parçası l istasyonundan k veya k' istasyonuna taşınıyorsa, i parçasının k veya k' istasyonuna denk gelen makinedeki işlemi, hareketin tamamlandığı zamana eşit olarak, parça makineye bırakıldığı anda başlamaktadır. Çıktı deposundan parça alınması mümkün olmadığından l, 5. istasyona eşit olamamaktadır. s_{i_k} ve c_{i_k} değerleri arasındaki ilişki göz önüne alındığında, Eş. (4.13)-(4.14)'e bakılarak, s_{i_k} değerinin i_k veya $i_{k'}$ düğümünün tamamlanma zamanına eşit olacağı, yani $s_{i_k} = c_{i_k}$ veya $s_{i_k} = c_{i_{k'}}$ olacağı söylenebilir.

Bekleme zamanlarını hesaplamaya yönelik olarak her düğüm için z_{i_k} ile gösterilen bir değişken kullanılmaktadır. Bu değişken belirli bir çevrimde parçanın yüklenmesinin mi boşaltılmasının mı daha önce gerçekleştirildiğinin anlaşılmasına yardımcı olur. z değerleri Eş. (4.15)-(4.16) ile belirlenmektedir:

$$c_{i_k} \geq s_{i_k} + {P'}_{i_k} - M\left(1 - z_{i_k}\right), \quad \forall i_k \in N \tag{4.15}$$

$$s_{i_k} \geq c_{i_k} + 4\varepsilon + 3\delta - M z_{i_k}, \quad \forall i_k \in N \tag{4.16}$$

Burada, $z_{i_k} = 1$ olması $c_{i_k} \geq s_{i_k} + {P'}_{i_k}$ olmasını, $z_{i_k} = 0$ olması ise $s_{i_k} \geq c_{i_k} + 4\varepsilon + 3\delta$ olmasını sağlamaktadır. İkinci durumda, yüklenmiş durumda olan bir parçanın boşaltılması ve tekrar yüklenmesi gerekmektedir. $4\varepsilon + 3\delta$ makinenin boşaltılması ile yeniden yüklenmesi arasında geçecek minimum toplam zamanı vermektedir.

Eş. (4.13)-(4.16)'da M, başlangıç ve tamamlanma zamanlarının daha büyük değer alamayacağı kadar büyük bir sayıyı temsil etmektedir. Sabit hareket zamanlı bir hücrede, bir makineye yüklenecek ve makineden boşaltılacak her bir parça için en fazla $4\varepsilon + 3\delta$ birim zaman gerektiğinden, $M = \sum_i (P_{i1} + P_{i2}) + n(4\varepsilon + 3\delta)$ olarak kullanılması mümkündür.

Önceden belirtildiği üzere, bekleme zamanı, robot makineyi boşaltmak üzere geldiğinde parçanın işlemi bitmemiş ise görülür. Bu formülasyonda, bekleme zamanı düğümlerin başlangıç ve tamamlanma zamanlarıyla (s_{i_k} ve c_{i_k} değerleriyle) birlikte z_{i_k} değişkenlerine göre hesaplanmaktadır.

$$w_{i_k} \geq P'_{i_k} - \left(c_{i_k} - s_{i_k}\right) - M\left(1 - z_{i_k}\right), \quad \forall i_k \in N \tag{4.17}$$

$$w_{i_k} \geq P'_{i_k} - \left(C - s_{i_k} + c_{i_k}\right) - M z_{i_k}, \quad \forall i_k \in N \tag{4.18}$$

Eş. (4.17)-(4.18)'de görüldüğü üzere, bekleme zamanı, $z_{i_k} = 1$ ise $maks\{0, P'_{i_k} - \left(c_{i_k} - s_{i_k}\right)\}$'ya, $z_{i_k} = 0$ ise $maks\{0, P'_{i_k} - \left(C - s_{i_k} + c_{i_k}\right)\}$'ya eşittir. Örneğin ikinci durumda, s_{i_k}'dan c_{i_k}'ya kadar geçen zaman $C - s_{i_k} + c_{i_k}$ toplamına eşittir. Bu durumda, elde edilen değer ile toplam işlem zamanı karşılaştırılır. Hangi hareketler için bekleme zamanı oluşacağının belirlenmesinde $wait_{i_k j_l}$ parametreleri kullanılmaktadır.

Bir diğer eşitlik kümesi, hareketler ile istasyonlarda yüklenmiş olan parçalar arasındaki ilişkinin tanımlanması için oluşturulmuştur. $(i_k j_l)$ hareketleri için aşağıda açıklandığı üzere, iki mümkün durum bulunmaktadır:

- Robot aktivitesi farklı parçalar için gerçekleşir, yani $i \neq j$'dir.
  Bu durumda robot i parçasını k istasyonuna bırakır ve j parçası için l istasyonuna gider. Bu tip bir hareket herhangi bir parça taşıması içermediğinden makinelerde yüklenmiş olan parçalarda da herhangi bir değişikliğe yol açmamaktadır.
- Robot aktivitesi aynı parça için gerçekleşir, yani $i = j$'dir.
  Bu durumda robot k istasyonundan bir parça alır ve parçayı yüklemek üzere l istasyonuna gider. Böyle bir hareketin yapılabilmesi için aşağıdaki durumların sağlanmış olması gereklidir:

- l istasyonuna karşılık gelen makinenin, yani yüklenecek makinenin, i_k düğümünde boş olması gerekmektedir. Çıktı deposunda her zaman boş yer olacağı varsayımından dolayı, $l \neq 5$ alınmaktadır.
- k istasyonuna karşılık gelen makinenin, yani boşaltılacak makinenin, j_l düğümünde boş olması gerekmektedir. Tanım gereği, girdi deposunda her zaman parça olacağından, $k \neq 1$ alınmaktadır.
- Bu şekilde bir hareket k, k' ve l, l' dışındaki istasyonlarda yüklenmiş olan parçalarda değişikliğe yol açmamaktadır.

1 ve 5 istasyonları, sırasıyla, girdi ve çıktı depolarına karşılık geldiğinden, yukarıdaki ilişkiler bu iki istasyon dışındaki istasyonlar için kontrol edilir. Bu durumları sağlamak için kullanılan eşitsizlikler aşağıdaki şekildedir:

- i_k düğümünden j_l düğümüne bir hareket varsa ve bu hareket makinelerde yüklenmiş olan parçalarda bir değişikliğe yol açmıyorsa, $m2_{i_k} = m2_{j_l}$, $m3_{i_k} = m3_{j_l}$ ve $m4_{i_k} = m4_{j_l}$ olması gereklidir. Bu sonuç Eş. (4.19)-(4.24) ile sağlanır:

$$-m2_{i_k} + m2_{j_l} \leq n(1 - y_{i_k j_l}) \quad (4.19)$$

$$m2_{i_k} - m2_{j_l} \leq n\left(1 - y_{i_k j_l}\right) \quad (4.20)$$

$$-m3_{i_k} + m3_{j_l} \leq n(1 - y_{i_k j_l}) \quad (4.21)$$

$$m3_{i_k} - m3_{j_l} \leq n(1 - y_{i_k j_l}) \quad (4.22)$$

$$-m4_{i_k} + m4_{j_l} \leq n(1 - y_{i_k j_l}) \quad (4.23)$$

$$m4_{i_k} - m4_{j_l} \leq n(1 - y_{i_k j_l}) \quad (4.24)$$

- Herhangi bir i parçası k istasyonundan l istasyonuna taşınırsa,
  - l istasyonuna karşılık gelen makine (2., 3. veya 4. istasyon) boş olmalıdır. Bu, ilgili değişkenin, robot önceki istasyonda iken 0'a eşit olması anlamına gelir.

$$m2_{i_k} \leq n(1 - y_{i_k i_l}),\ m3_{i_k} \leq n(1 - y_{i_k i_l}) \text{ veya } m4_{i_k} \leq n(1 - y_{i_k i_l}) \quad (4.25)$$

- Önceki duruma benzer olarak, k istasyonuna karşılık gelen makine (2., 3. veya 4. istasyon) boş hale geçer. Yani, robot sonraki istasyonda iken, ilgili değişken 0'a eşit olmalıdır.

$$m2_{i_l} \leq n(1 - y_{i_k i_l}),\ m3_{i_l} \leq n(1 - y_{i_k i_l}) \text{ veya } m4_{i_l} \leq n(1 - y_{i_k i_l}) \quad (4.26)$$

- Bu durumda, diğer makineler için herhangi bir değişiklik gözlenmez. İlgili makinelere göre, Eş. (4.19)-(4.24) kullanılmaktadır.

Sistemin yapısı gereği bazı hareketler birbirini engelleyebilmektedir. Bu hareketlere örnek olarak 3. ve 4. istasyonlar arası taşımaların gösterilmesi mümkündür. Bir parçanın herhangi bir aşamaya ait işleminin tamamı, söz konusu aşamada bulunan makinelerden yalnız biri tarafından gerçekleştirileceğinden, bu gibi taşımalara ve bu taşımalara sebep olacak atamalara izin verilmemektedir. Eş. (4.27)-(4.28) bu tip hareketleri belirlemek için oluşturulmuştur:

$$y_{i_k i_l} + y_{i_l i_k} + y_{i_k i_{l'}} + y_{i_l i_{k'}} + y_{i_{k'} i_l} + y_{i_{l'} i_k} + y_{i_{k'} i_{l'}} + y_{i_{l'} i_{k'}} = 0, \quad \forall i \quad (4.27)$$

$$\sum_{j_k} y_{j_k i_l} + y_{j_k i_m} \leq 1, \quad \forall i, \quad l, m = \{3,4,7,8\}, l \neq m, l \neq m', l' \neq m, l' \neq m' \quad (4.28)$$

Ele alınan sistemin esneklik özelliği sonucu girdi deposundan alınan bir parçanın ilk aşamada işlemi yoksa bu aşamaya uğramadan ikinci aşamaya gelebileceği; bununla birlikte ilk aşamada işlem gören bir parçanın ikinci aşamada işlemi olmaması durumunda direk olarak çıktı deposuna gidebileceği bilinmektedir. Bu ilişkiyi sağlamak amacıyla, her iki aşamaya yönelik olarak aşağıdaki eşitsizlikler kullanılmıştır:

$$y_{i_1 i_2} + y_{i_1 i_6} \leq P_{i1}, \quad \forall i \quad (4.29)$$

$$y_{i_1 i_2} + y_{i_1 i_6} \geq {P_{i1}}/{M}, \quad \forall i \quad (4.30)$$

$$y_{i_3 i_5} + y_{i_4 i_5} + y_{i_7 i_5} + y_{i_8 i_5} \leq P_{i2}, \quad \forall i \quad (4.31)$$

$$y_{i_3 i_5} + y_{i_4 i_5} + y_{i_7 i_5} + y_{i_8 i_5} \geq {P_{i2}}/{M}, \quad \forall i \quad (4.32)$$

Eş. (4.29) ve Eş. (4.30) ile ilk aşamadaki işlem zamanının sıfır olması durumunda girdi deposundan $2.(6.)$ istasyonlara gidilmesi önlenirken, sıfırdan büyük olması durumunda bu hareket garantilenmiş olmakta; benzer şekilde Eş. (4.31) ve Eş. (4.32) ile de ikinci aşamada işlem zamanının sıfır olması durumunda $3.(7.)$ ve $4.(8.)$ istasyonlardan çıktı deposuna gidilmesi önlenip sıfırdan büyük olması durumunda bu hareket garantilenmiş olmaktadır. Bu formülasyonda amaç minimum çevrim zamanının bulunmasıdır. Aşağıdaki eşitsizlik bu tanımlama için kullanılır:

$$C \geq c_{i_k} + y_{i_k 1_1} \cdot cost_{i_k 1_1}, \quad \forall i_k \in N \tag{4.33}$$

Bu kısıt sayesinde, çevrimin tamamlanma zamanını veren, son düğümün tamamlanma zamanı bulunmaktadır. 1_1 düğümünün kullanılmasının sebebi, gezinin başlangıç noktası olarak 1_1 düğümünün alınmış olmasıdır.

Sonuç olarak, aşağıdaki karışık tamsayılı doğrusal programlama modeli oluşturulmuştur:

min C

s.t.

$\sum_{j_l:(j_l,i_k)\in A} y_{j_l i_k} = \sum_{j_l:(i_k,j_l)\in A} y_{i_k j_l}, \quad \forall i,k$

$\sum_{j_l:(i_k,j_l)\in A} y_{i_k j_l} \leq 1, \quad \forall i,k$

$\sum_{j_l:(j_l,i_k)\in A} y_{j_l i_k} \leq 1, \quad \forall i,k$

$\sum_{j_l} y_{i_1 j_l} = 1, \quad \forall i$

$\sum_{j_l} y_{j_l i_5} = 1, \quad \forall i$

$y_{i_1 i_2} + y_{i_1 i_6} + y_{i_1 i_3} + y_{i_1 i_7} + y_{i_1 i_4} + y_{i_1 i_8} = 1, \quad \forall i$

$y_{i_2 i_5} + y_{i_3 i_5} + y_{i_4 i_5} + y_{i_6 i_5} + y_{i_7 i_5} + y_{i_8 i_5} = 1, \quad \forall i$

$y_{i_2 i_3} + y_{i_2 i_7} + y_{i_6 i_3} + y_{i_6 i_7} + y_{i_2 i_4} + y_{i_2 i_8} + y_{i_6 i_4} + y_{i_6 i_8} \leq 1, \quad \forall i$

$c_{1_1} = 0$

$c_{j_l} \geq c_{i_k} + cost_{i_k j_l} \cdot y_{i_k j_l} - M\left(1 - y_{i_k j_l}\right) + wait_{i_k j_l} \cdot w_{i_k}, \quad \forall i,k,j,l$

$P'_{i_2} = P_{i1}, \quad \forall i$

$P'_{i_3} + P'_{i_4} = P_{i2}, \quad \forall i$

$P'_{i_2} = P'_{i_6}, \quad \forall i$

$P'_{i_3} = P'_{i_7}, \quad \forall i$

$$P'_{i_4} = P'_{i_8}, \quad \forall i$$

$$P'_{i_k} \leq P_{i1} * \sum_{j_l:(i_k,j_l)\in A \; veya \; (i_{k'},j_l)\in A} \left(y_{i_k j_l} + y_{i_{k'} j_l}\right), \quad \forall i, \; k = \{2\}$$

$$P'_{i_k} \leq P_{i2} * \sum_{j_l:(i_k,j_l)\in A \; veya \; (i_{k'},j_l)\in A} \left(y_{i_k j_l} + y_{i_{k'} j_l}\right), \quad \forall i, \; k = \{3,4\}$$

$$s_{i_k} = s_{i_{k'}}, \quad \forall i, \; k = \{2,3,4\}$$

$$s_{i_k} \geq c_{i_k} - M\left(1 - y_{i_l i_k}\right), \quad \forall i, \quad s.t. \quad k = \{2,3,4\}$$

$$s_{i_k} \geq c_{i_{k'}} - M\left(1 - y_{i_l i_{k'}}\right), \quad \forall i, \quad s.t. \quad k = \{2,3,4\}$$

$$c_{i_k} \geq s_{i_k} + P'_{i_k} - M\left(1 - z_{i_k}\right), \quad \forall i, k$$

$$s_{i_k} \geq c_{i_k} + 4\varepsilon + 3\delta - M z_{i_k}, \quad \forall i, k$$

$$w_{i_k} \geq P'_{i_k} - \left(c_{i_k} - s_{i_k}\right) - M\left(1 - z_{i_k}\right), \quad \forall i, k$$

$$w_{i_k} \geq P'_{i_k} - \left(C - s_{i_k} + c_{i_k}\right) - M z_{i_k}, \quad \forall i, k$$

$$m2_{i_k} = i, \quad \forall i, \quad k = \{2,6\}$$

$$m3_{i_k} = i, \quad \forall i, \quad k = \{3,7\}$$

$$m4_{i_k} = i, \quad \forall i, \quad k = \{4,8\}$$

$$m2_{i_k} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \; k = \{1\}, \; l = \{2,6\}$$

$$-m3_{i_k} + m3_{i_l} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \; k = \{1\}, \; l = \{2,6\}$$

$$m3_{i_k} - m3_{i_l} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \; k = \{1\}, \; l = \{2,6\},$$

$$-m4_{i_k} + m4_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \; k = \{1\}, \; l = \{2,6\}$$

$$m4_{i_k} - m4_{i_l} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \; k = \{1\}, \; l = \{2,6\}$$

$$m3_{i_k} \leq n\left(1 - y_{i_k i_l}\right), \; k = \{1\}, \quad \forall i, \; l = \{3,7\}$$

$$-m2_{i_k} + m2_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \; k = \{1\}, \; l = \{3,7\}$$

$$m2_{i_k} - m2_{i_l} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \; k = \{1\}, \; l = \{3,7\}$$

$$-m4_{i_k} + m4_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \; k = \{1\}, \; l = \{3,7\}$$

$$m4_{i_k} - m4_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \; k = \{1\}, \; l = \{3,7\}$$

$$m4_{i_k} \leq n\left(1 - y_{i_k i_l}\right), \quad k = \{1\}, \quad \forall i, \; l = \{4,8\}$$

$$-m2_{i_k} + m2_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \quad k = \{1\}, \; l = \{4,8\}$$

$$m2_{i_k} - m2_{i_l} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \quad k = \{1\}, \; l = \{4,8\}$$

$$-m3_{i_k} + m3_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \quad k = \{1\}, \; l = \{4,8\}$$

$$m3_{i_k} - m3_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \quad k = \{1\}, \; l = \{4,8\}$$

$$-m2_{i_k} + m2_{i_l} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \; k = \{3,4,7,8\}, \; l = \{1\}$$

$$m2_{i_k} - m2_{i_l} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \ \ k = \{3,4,7,8\}, \quad l = \{1\}$$
$$-m3_{i_k} + m3_{i_l} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \ \ k = \{3,4,7,8\}, \ \ l = \{1\}$$
$$m3_{i_k} - m3_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \ \ k = \{3,4,7,8\}, \quad l = \{1\}$$
$$-m4_{i_k} + m4_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \ \ k = \{3,4,7,8\}, \quad l = \{1\}$$
$$m4_{i_k} - m4_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \ \ k = \{3,4,7,8\}, \quad l = \{1\}$$
$$m2_{i_k} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \ \ k = \{2,6\}, \ \ l = \{5\}$$
$$-m3_{i_k} + m3_{i_l} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \ \ k = \{2,6\}, \ \ l = \{5\}$$
$$m3_{i_k} - m3_{i_l} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \ \ k = \{2,6\}, \ \ l = \{5\}$$
$$-m4_{i_k} + m4_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \ \ k = \{2,6\}, \ \ l = \{5\}$$
$$m4_{i_k} - m4_{i_l} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \ \ k = \{2,6\}, \ \ l = \{5\}$$
$$m3_{i_k} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \ \ k = \{3,7\}, \ \ l = \{5\}$$
$$-m2_{i_k} + m2_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \ \ k = \{3,7\}, \ \ l = \{5\}$$
$$m2_{i_k} - m2_{i_l} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \ \ k = \{3,7\}, \ \ l = \{5\}$$
$$-m4_{i_k} + m4_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \ \ k = \{3,7\}, \ \ l = \{5\}$$
$$m4_{i_k} - m4_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \ \ k = \{3,7\}, \ \ l = \{5\}$$
$$m4_{i_k} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \ \ k = \{4,8\}, \ \ l = \{5\}$$
$$-m2_{i_k} + m2_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \ \ k = \{4,8\}, \ \ l = \{5\}$$
$$m2_{i_k} - m2_{i_l} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \ \ k = \{4,8\}, \ \ l = \{5\}$$
$$-m3_{i_k} + m3_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \ \ k = \{4,8\}, \ \ l = \{5\}$$
$$m3_{i_k} - m3_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \ \ k = \{4,8\}, \ \ l = \{5\}$$
$$m2_{i_l} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \ \ k = \{2,6\}, \ \ l = \{3,7\}$$
$$m3_{i_l} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \ \ k = \{2,6\}, \ \ l = \{3,7\}$$
$$-m4_{i_k} + m4_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \ \ k = \{2,6\}, \ \ l = \{3,7\}$$
$$m4_{i_k} - m4_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \ \ k = \{2,6\}, \ \ l = \{3,7\}$$
$$m2_{i_l} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \ \ k = \{2,6\}, \ \ l = \{4,8\}$$
$$m4_{i_l} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \ \ k = \{2,6\}, \ \ l = \{4,8\}$$
$$-m3_{i_k} + m3_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \ \ k = \{2,6\}, \ \ l = \{4,8\}$$
$$m3_{i_k} - m3_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \ \ k = \{2,6\}, \ \ l = \{4,8\}$$
$$-m2_{i_k} + m2_{i_l} \leq n(1 - y_{i_k i_l}), \quad \forall i, \ \ k = \{5\}, \ \ l = \{1\}$$
$$m2_{i_k} - m2_{i_l} \leq n\left(1 - y_{i_k i_l}\right), \quad \forall i, \ \ k = \{5\}, \ \ l = \{1\}$$

$$-m3_{i_k} + m3_{i_l} \le n(1 - y_{i_k i_l}), \quad \forall i, \ k = \{5\}, \ l = \{1\}$$
$$m3_{i_k} - m3_{i_l} \le n(1 - y_{i_k i_l}), \quad \forall i, \ k = \{5\}, \ l = \{1\}$$
$$-m4_{i_k} + m4_{i_l} \le n(1 - y_{i_k i_l}), \quad \forall i, \ k = \{5\}, \ l = \{1\}$$
$$m4_{i_k} - m4_{i_l} \le n(1 - y_{i_k i_l}), \quad \forall i, \ k = \{5\}, \ l = \{1\}$$
$$-m2_{i_k} + m2_{j_l} \le n\left(1 - y_{i_k j_l}\right), \quad \forall i, j, \ i \ne j, \ k = \{2,6\}, \ l = \{1,3,7,4,8\}$$
$$m2_{i_k} - m2_{j_l} \le n\left(1 - y_{i_k j_l}\right), \quad \forall i, j, \ i \ne j, \ k = \{2,6\}, \ l = \{1,3,7,4,8\}$$
$$-m3_{i_k} + m3_{j_l} \le n\left(1 - y_{i_k j_l}\right), \quad \forall i, j, \ i \ne j, \ k = \{2,6\}, \ l = \{1,3,7,4,8\}$$
$$m3_{i_k} - m3_{j_l} \le n\left(1 - y_{i_k j_l}\right), \quad \forall i, j, \ i \ne j, \ k = \{2,6\}, \ l = \{1,3,7,4,8\}$$
$$-m4_{i_k} + m4_{j_l} \le n\left(1 - y_{i_k j_l}\right), \quad \forall i, j, \ i \ne j, \ k = \{2,6\}, \ l = \{1,3,7,4,8\}$$
$$m4_{i_k} - m4_{j_l} \le n\left(1 - y_{i_k j_l}\right), \quad \forall i, j, \ i \ne j, \ k = \{2,6\}, \ l = \{1,3,7,4,8\}$$
$$-m2_{i_k} + m2_{j_l} \le n\left(1 - y_{i_k j_l}\right), \quad \forall i, j, \ i \ne j, \ k = \{3,7\}, \ l = \{1,2,6,4,8\}$$
$$m2_{i_k} - m2_{j_l} \le n\left(1 - y_{i_k j_l}\right), \quad \forall i, j, \ i \ne j, \ k = \{3,7\}, \ l = \{1,2,6,4,8\}$$
$$-m3_{i_k} + m3_{j_l} \le n(1 - y_{i_k j_l}), \quad \forall i, j, \ i \ne j, \ k = \{3,7\}, \ l = \{1,2,6,4,8\}$$
$$m3_{i_k} - m3_{j_l} \le n(1 - y_{i_k j_l}), \quad \forall i, j, \ i \ne j, \ k = \{3,7\}, \ l = \{1,2,6,4,8\}$$
$$-m4_{i_k} + m4_{j_l} \le n(1 - y_{i_k j_l}), \quad \forall i, j, \ i \ne j, \ k = \{3,7\}, \ l = \{1,2,6,4,8\}$$
$$m4_{i_k} - m4_{j_l} \le n(1 - y_{i_k j_l}), \quad \forall i, j, \ i \ne j, \ k = \{3,7\}, \ l = \{1,2,6,4,8\}$$
$$-m2_{i_k} + m2_{j_l} \le n\left(1 - y_{i_k j_l}\right), \quad \forall i, j, \ i \ne j, \ k = \{4,8\}, \ l = \{1,2,6,3,7\}$$
$$m2_{i_k} - m2_{j_l} \le n\left(1 - y_{i_k j_l}\right), \quad \forall i, j, \ i \ne j, \ k = \{4,8\}, \ l = \{1,2,6,3,7\}$$
$$-m3_{i_k} + m3_{j_l} \le n(1 - y_{i_k j_l}), \quad \forall i, j, \ i \ne j, \ k = \{4,8\}, \ l = \{1,2,6,3,7\}$$
$$m3_{i_k} - m3_{j_l} \le n(1 - y_{i_k j_l}), \quad \forall i, j, \ i \ne j, \ k = \{4,8\}, \ l = \{1,2,6,3,7\}$$
$$-m4_{i_k} + m4_{j_l} \le n(1 - y_{i_k j_l}), \quad \forall i, j, \ i \ne j, \ k = \{4,8\}, \ l = \{1,2,6,3,7\}$$
$$m4_{i_k} - m4_{j_l} \le n(1 - y_{i_k j_l}), \quad \forall i, j, \ i \ne j, \ k = \{4,8\}, \ l = \{1,2,6,3,7\}$$
$$-m2_{i_k} + m2_{j_l} \le n\left(1 - y_{i_k j_l}\right), \quad \forall i, j, \ i \ne j, \ k = \{5\}, \ l = \{1,2,6,3,7,4,8\}$$
$$m2_{i_k} - m2_{j_l} \le n\left(1 - y_{i_k j_l}\right), \quad \forall i, j, \ i \ne j, \ k = \{5\}, \ l = \{1,2,6,3,7,4,8\}$$
$$-m3_{i_k} + m3_{j_l} \le n(1 - y_{i_k j_l}), \quad \forall i, j, \ i \ne j, \ k = \{5\}, \ l = \{1,2,6,3,7,4,8\}$$
$$m3_{i_k} - m3_{j_l} \le n(1 - y_{i_k j_l}), \quad \forall i, j, \ i \ne j, \ k = \{5\}, \ l = \{1,2,6,3,7,4,8\}$$
$$-m4_{i_k} + m4_{j_l} \le n(1 - y_{i_k j_l}), \quad \forall i, j, \ i \ne j, \ k = \{5\}, \ l = \{1,2,6,3,7,4,8\}$$
$$m4_{i_k} - m4_{j_l} \le n(1 - y_{i_k j_l}), \quad \forall i, j, \ i \ne j, \ k = \{5\}, \ l = \{1,2,6,3,7,4,8\}$$
$$y_{i_k i_l} + y_{i_l i_k} + y_{i_k i_{l'}} + y_{i_l i_{k'}} + y_{i_{k'} i_l} + y_{i_{l'} i_k} + y_{i_{k'} i_{l'}} + y_{i_{l'} i_{k'}} = 0, \quad \forall i$$
$$\sum_{j_k} y_{j_k i_l} + y_{j_k i_m} \le 1, \quad \forall i, \quad l, m = \{3,4,7,8\}, \quad l \ne m, \quad l \ne m', \quad l' \ne m, \quad l' \ne m'$$

$$y_{i_1 i_2} + y_{i_1 i_6} \leq P_{i1}\,, \quad \forall i$$

$$y_{i_1 i_2} + y_{i_1 i_6} \geq {}^{P_{i1}}/_{M}, \quad \forall i$$

$$y_{i_3 i_5} + y_{i_4 i_5} + y_{i_7 i_5} + y_{i_8 i_5} \leq P_{i2}\,, \quad \forall i$$

$$y_{i_3 i_5} + y_{i_4 i_5} + y_{i_7 i_5} + y_{i_8 i_5} \geq {}^{P_{i2}}/_{M}, \quad \forall i$$

$$C \geq c_{i_k} + y_{i_k 1_1} \cdot cost_{i_k 1_1}, \quad \forall i, k$$

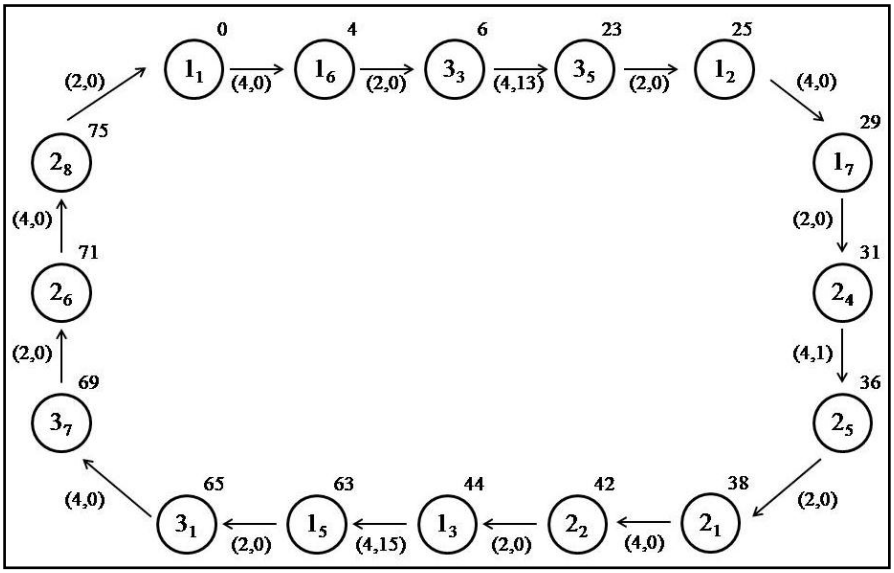
Oluşturulan matematiksel model GAMS/CPLEX Çözücüsü kullanılarak kodlanmıştır. EK-1'de ilgili GAMS/CPLEX kodu bir örnek probleme yönelik olarak verilmiş, EK-2'de ilgili probleme yönelik çözüm çıktısı gösterilmiştir.

4. 1. Örnek Uygulama

Oluşturulan matematiksel model, 3 parçanın bulunduğu bir örnek sistem için çalıştırılmış ve sonuç alınmıştır. Problem verileri şu şekildedir:

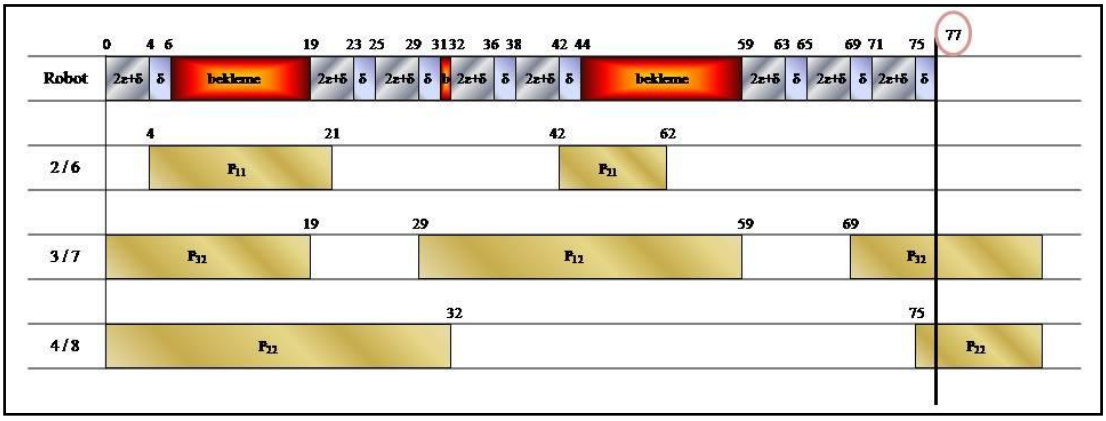
Örnek 1. 3 parçanın üretileceği 2 aşamalı bir sistemde parçaların işlem zamanları 1. aşama için sırasıyla $P_{11} = 17$, $P_{21} = 20$, $P_{31} = 0$, 2. aşama için sırasıyla $P_{12} = 30$, $P_{22} = 34$, $P_{32} = 27$ birim zamandır. Yükleme/boşaltma (ε) ve haraket zamanları (δ) sırasıyla 1 ve 2 birim zamandır.

Bulunan çözümdeki tur Şekil 4.1'de verilmiştir. Şekilde, düğümlerin üzerindeki sayılar düğümün tamamlanma zamanını, okların üzerindeki sayılar ise sırasıyla düğümler arasındaki hareket zamanı ve bu hareketin doğurduğu bekleme zamanını vermektedir. Örneğin, 3_3 düğümünden 3_5 düğümüne gitmek için 4 birimlik robot hareket zamanı gerekiyorken, 13 birimlik de bekleme gerekmiş ve toplamda bu hareket 17 birim zamanda tamamlanmıştır.



Şekil 4.1. Örnek 1 için elde edilen gezgin satıcı turu gösterimi

Elde edilmiş olan turun gerçek sistemdeki yansımasının Şekil 4.2'de verilen Gantt şemasından görülmesi mümkündür. Buna göre, $t = 0$ anında 3. ve 4. istasyonlar dolu, 2. istasyon dolu olarak başlanmakta; yine bu anda, robot girdi deposunun önünde hazır bulunmaktadır. Robot 1 no'lu parçayı girdi deposundan alarak bu parçayı 2. istasyona yüklemektedir. Söz konusu hareket 1 no'lu parçanın girdi deposundan alınması (ε), girdi deposundan 2. istasyona gidilmesi (δ) ve parçanın bu istasyona bırakılması (ε) şeklinde olup toplamda $2\varepsilon + \delta$ birim zaman almaktadır. Sonraki hareketlerin de benzer şekilde takip edilmesi mümkündür. Şekilden görüleceği üzere çevrimin tamamlanması 77 birim zaman almakta ve $t = 77$ anında tüm istasyonlar $t = 0$ anındaki başlangıç durumlarına geri dönmektedir.



Şekil 4.2. Çözümün Gantt-şeması üzerinde gösterimi

GSP'nin NP-Zor yapıda bir problem olduğu bilinmektedir. Bu çalışmada oluşturulan formülasyon ise klasik GSP'den daha genel bir yapıda olup düşük makine sayılarına sahip hücrelerdeki problemlerin çözümünde bile yüksek bilgisayar gücü gerektirmektedir. Yapılan denemelerde modelin düşük parça sayılarına sahip küçük boyutlu problemler için optimal çözüm verdiği ancak büyük boyutlu problemler için yetersiz kaldığı gözlenmiş, bu nedenle problemin çözümüne yönelik olarak TB tabanlı bir sezgisel algoritma oluşturulması düşünülmüştür. Sonraki bölümde, önerilen algoritmanın çalışma prensibi anlatılmaktadır.

5. SEZGİSEL YÖNTEM

Optimum sonuca ulaşmanın çok zor olduğu veya mümkün olmadığı bu tip problemlerin çözüm aşamalarında matematiksel modeller yetersiz kalmakta ve araştırmacılar problem karmaşıklığını göstererek çözüme yönelik farklı, etkin algoritmalar geliştirmeye çalışmaktadır.

Bu çalışmada ele alınan problemde verilecek karar üç boyutludur: bunlar, parça sıralaması, parça/makine atamaları ve robot hareket dizisinin belirlenmesidir. İleriki bölümlerde anlatılacağı üzere, bu üç kararı belirtebilmek üzere kullanılacak çözüm gösterimi ilki parça sırası, ikincisi parça/makine atamaları olmak üzere 2 bölümden oluşmalıdır.

Bu şekilde oluşturulan bir yapı üzerinde arama yapılabilmesi için kullanılabilecek sezgisel algoritmalar arasında karşılaştırma yapıldığında Tavlama Benzetimi kullanımının en uygun seçim olduğu görülmektedir. Problem birden fazla karar içermekte olup bunu sağlayabilecek bir çözüme yönelik olarak örneğin Tabu Arama yaklaşımı kullanılmak istendiğinde birden fazla tabu listesi tutulması gibi ihtiyaçlar veya Genetik Algoritma kullanımında yine çözümün her bir parçasına yönelik ayrı mutasyon yapıları geliştirilmesi gibi gereksinimler ortaya çıkacağından, uygulanması daha kolay ve bununla birlikte oldukça etkin bir yapıya sahip olan TB kullanımı tercih edilmiştir. Bunun yanı sıra; maksimum tamamlanma zamanı, akış zamanı, boş zaman, gecikme vb. amaçların optimizasyonuna yönelik olarak geliştirilen yöntemler arasından TB'nin bu amaçlara ulaşılmasında faydalı bir algoritma olduğu da mevcut literatürde belirtilmektedir (Hooda ve Dhingra, 2011).

TB'nin doğada bulunan iyileştirme prosedürlerinden biri olarak görülmesi mümkündür. Yöntem, Metropolis, Rosenbluth ve Resenbluth tarafından geliştirilmiş, Kirkpatrick, Gelatt Jr. ve Vecchi tarafından popülerlik kazandırılmıştır (Metropolis, Rosenbluth ve Resenbluth, 1953; Kirkpatrick, Gelatt Jr. ve Vecchi, 1983). Çizelgeleme problemlerine yönelik olarak birçok araştırmacı TB kullanımını tercih etmiştir. Kim, Kim, Jang ve Chen, Low, Yeh ve Huang, Lee, Lin ve Ying, Behnamian, Zandieh ve Fatemi Ghomi, Zhang ve Wu tarafından yapılan çalışmalar bunlara örnektir (Kim, Kim, Jang ve Chen, 2002; Low, Yeh ve Huang, 2004; Lee, Lin ve Ying, 2010; Behnamian, Zandieh ve Fatemi Ghomi, 2009; Zhang ve Wu, 2010).

Kim ve diğerleri sıra-bağımlı hazırlık zamanını ve parti üretimini dikkate alarak toplam gecikmeyi minimize etmeye yönelik paralel makine çizelgeleme probleminin çözümünde TB'yi kullanmıştır (Kim ve diğerleri, 2002). Low ve diğerleri, akış tipi çizelgeleme problemleri için iyi bir çözümün karakteristiklerini kaydetmiş ve TB'nin arama prosedürünün etkinliğini artırmak üzere bu özellikleri algoritma içerisinde kullanmışlardır (Low ve diğerleri, 2004). Lee ve diğerleri, sınırlı bir TB algoritması sunmuş, en iyi komşunun bulunması için bazı etkin olmayan hareketlerin elenmesini sağlayarak sınırlı bir arama stratejisi oluşturarak dinamik paralel makine çizelgeleme problemlerinin çözümüne çalışmışlardır (Lee ve diğerleri, 2010). Behnamian ve diğerleri, paralel makineli ve sıra bağımlı hazırlık zamanlı çizelgeleme probleminde maksimum tamamlanma zamanının minimizasyonuna yönelik olarak tavlama benzetimini karınca kolonisi ve değişken komşu arama ile birlikte kullanan hibrit bir metasezgisel kullanmıştır (Behnamian ve diğerleri, 2009). Zhang ve Wu, atölye tipi çizelgeleme probleminde toplam ağırlıklı gecikmeyi minimize etmek için yine TB tabanlı bir algoritma geliştirmişlerdir (Zhang ve Wu, 2010). Mirsanei ve diğerleri de sıra bağımlı hazırlık zamanlı, özdeş paralel makineli hibrit akış tipi çizelgeleme probleminde maksimum tamamlanma zamanı minimizasyonuna yönelik olarak TB tabanlı bir yaklaşım önermişlerdir (Mirsanei ve diğerleri, 2011).

Bu yaklaşımı temel alan yöntemler, geleneksel sezgiseller ile sağlanamayan yerel optimumdan kaçış yeteneğine sahiptir. Bu avantajından dolayı farklı optimizasyon problemlerine uygulandığında yüksek kaliteli çözümler elde edilmesine imkan sağlamaktadır (Zegordi, Itoh ve Enkawa, 1995).

TB, yüksek bir sıcaklık değeriyle çalışmaya başlamaktadır. İlk çözüm oluşturulduktan sonra mevcut çözümden onun komşularının birine hareket edilmesi denenmekte, bu denemenin gerçekleşip gerçekleşmeyeceğinin belirlenmesi için amaç fonksiyonundaki değişiklikler hesaplanmaktadır. Yeni çözümün amaç fonksiyonu değeri öncekinden iyiyse yeni çözüm kabul edilmekte, değilse önceden tanımlanmış bir olasılık fonksiyonuna bağlı olarak yeni çözüme kabul edilme şansı verilmektedir. Olasılık fonksiyonu şu şekilde tanımlanır: $P(\Delta E) = e^{-\Delta E/T}$, burada ΔE amaç fonksiyonundaki değişiklik oranını, T ise mevcut sıcaklığı göstermektedir. Bu kontrol için öncelikle $(0,1)$ arasından rassal bir değer seçilmekte, seçilen değer olasılık değerinden küçük veya bu değere eşitse, yeni çözüm kabul edilmekte, değilse reddedilmektedir. Böylelikle, daha kötü amaç fonksiyonu değeri

veren çözümlerin kabul edilmesi sayesinde yerel optimumda sıkışık kalınması önlenir. TB ısı dengesine ulaşmak için bu işlem her sıcaklıkta L defa tekrarlanmaktadır, burada L Markov zincir uzunluğu olarak adlandırılan bir kontrol parametresidir. TB, sonlandırma kriterine ulaşılana kadar devam ettiği sürece, T parametresi önceden tanımlanan başlangıç sıcaklığından, bir soğuma fonksiyonu ile azar azar düşürülerek yine önceden tanımlanan bitiş sıcaklığına ulaşıldığında algoritma sonlandırılmaktadır.

TB'yi belli bir kombinatoryal optimizasyon problemine uyarlamak için verilmesi gereken bir takım kararlar vardır. Bunlardan ilki başlangıç sıcaklığı olup bu değer hemen hemen tüm komşuluklara gidişe izin verebilecek kadar yüksek olmalıdır. Bu şekilde ayarlanmazsa son elde edilen çözüm başlangıç çözümü ile aynı veya ona çok yakın değer alabilecektir. Ancak, başlangıç sıcaklığının çok yüksek olması da aramanın herhangi bir komşuya gidebilmesine imkan verecek ki bu da özellikle aramanın ilk aşamalarının rassal aramaya dönüşmesine sebep olacaktır. Bir diğer karar olan bitiş sıcaklığına bakıldığında ise, literatürde sıcaklık derecesinin sıfıra ulaşana kadar düşürülmesi yaygın bir kullanımdır. Bazı uygulamalarda ise sıcaklık değeri belirli şartların sağlanmasına bağlı olarak düşürülmeye devam edilmektedir; örneğin en iyi durumda belirli bir sure boyunca değişiklik olmaması durumunda sıcaklığın düşürülmesine son verilebilmektedir.

Yapılan çalışmada başlangıç ve bitiş sıcaklık değerleri Karaoğlan, Altiparmak, Kara ve Dengiz, tarafından yapılmış çalışmaya dayanılarak belirlenmiştir (Karaoğlan, Altiparmak, Kara ve Dengiz, 2011). Sözkonusu çalışmada, TB'nin başlangıcında mevcut çözümden 70% oranında daha kötü bir çözümün 0,90 olasılıkla kabul edilmesi, bitişine gelindiğinde ise mevcut çözümden 1% oranında daha kötü bir çözümün 0,001 olasılıkla kabul edilmesine izin verilmesinin uygun olacağı düşünülmüştür. Bunun için yukarıda tanımlanan olasılık fonksiyonu değerinden yola çıkılarak aşağıdaki eşitlikler oluşturulmuştur:

$$P(70) = e^{-70/T} = 0{,}90 \rightarrow \ln(0{,}90) = -70/T \quad \rightarrow \quad T = -70/\ln(0{,}90) = 665 \qquad (5.1)$$

$$P(1) = e^{-1/T} = 0{,}001 \rightarrow \ln(0{,}001) = -1/T \quad \rightarrow \quad T = -1/\ln(0{,}001) = 0{,}15 \qquad (5.2)$$

Verilen eşitlikler ışığında, çalışmada bitiş sıcaklık değeri için 0,15 değeri sabit tutularak, başlangıç sıcaklık değeri için farklı olasılık değerleri denenmiş ve Çizelge 5.1'de verilen

sıcaklık değerleri ile birlikte literatürde sıkça rastlanılan $T_i = 100$ değeri için, toplamda beş farklı başlangıç değerine yönelik denemeler yapılmıştır.

Çizelge 5.1. Başlangıç sıcaklık değerleri

ΔE oranı	Kabul olasılığı	Başlangıç sıcaklığı
- 60%	0,70	168
- 70%	0,70	196
- 70%	0,80	313
- 70%	0,90	665

Sıcaklık değerinin düşürülmesinde kullanılan yol da algoritmanın başarısı üzerinde kritik öneme sahiptir. Yaygın kullanım, d_r azaltma oranını göstermek ve $d_r < 1$ olmak üzere $t = t \times d_r$ şeklindeki geometrik azalma fonksiyonunun kullanımıdır. Mevcut literatür azaltım oranının $0{,}8$ ile $0{,}99$ arasında olması gerektiğini, daha iyi sonuçların ise yüksek oran ile elde edildiğini göstermektedir. Bu çalışmada sıcaklık değeri düşürülürken $d_r = 0{,}99$ olduğu kabul edilmiştir.

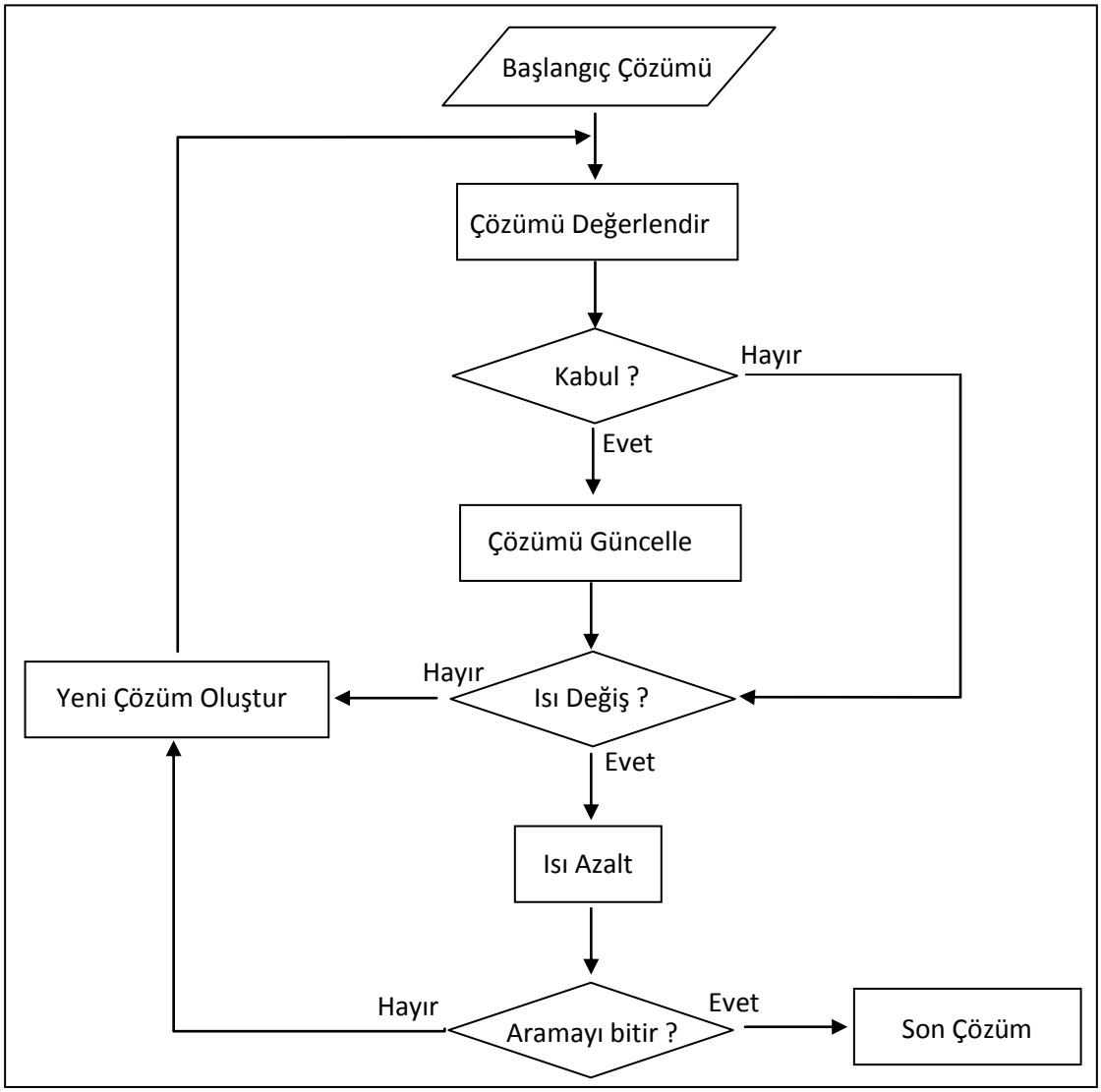
Bir diğer karara gelindiğinde, TB teorisine göre, sıcaklık düşürülmeden önce her sıcaklıkta sistemin denge durumuna ulaşmasına yetecek kadar yüksek sayıda iterasyon gerçekleştirilmelidir. Bu çalışmada her sıcaklıkta problemde yer alan parça sayısı (n) kadar tekrar yapılmaktadır.

Buraya kadar tanımlanan tavlama planına özgü kararlardan sonra sırada probleme özgü kararların belirlenmesi olacaktır. Ele alınan problem için kritik noktalar parça sıralamaları ve birden fazla makineye sahip aşamalar için makine atamalarıdır. Bununla birlikte sistem çevrimsel olarak üretimine devam ettiğinden dolayı, oluşturulan çözümde başlangıç durumunun da bilinmesi gerekmektedir. Problemin çözüm gösterimi, parça sıralamaları ile birlikte ilgili parçanın her bir aşamada hangi makineye atandığını gösteren, n parça sayısı ve m sistemdeki aşama sayısını belirtmek üzere, $n + (m \times n)$ boyutlu bir vektörle sağlanmıştır. Bu vektör $m + 1$ parçalı olarak oluşturulup ilk kısım işlem görecek parçaların sırasını, ikinci ve sonraki kısımlar söz konusu parçaların işleneceği aşamalardaki makineleri göstermektedir. Bu şekilde elde edilen sıralama ve atamalara göre, robot tüm

parçaları girdi deposundan ilgili makinelere taşımakta ve işlemleri tamamlandığında çıktı deposuna teslim etmektedir.

Problemin eniyilenmesi gereken tek bir amacı vardır. Tüm parçaların girdi deposundan alınması ile işlemi biten parçaların çıktı deposuna bırakılması arasında geçen çevrim zamanının minimize edilmesi söz konusu problemin nihai amacıdır. Çevrim zamanı hesabı eldeki sıralama ve atamalara göre algoritmanın çalışması sırasında yapılmaktadır.

Problemin çözümünde kullanılan TB algoritmasının akış diyagramı Şekil 5.1'de verilmiştir.



Şekil 5.1. Tavlama benzetimi algoritmasına ait akış diyagramı

Şekilden hareketle, sözkonusu yapının şu şekilde özetlenmesi mümkündür:

- İlk çözümün rassal olarak oluşturulması ve çevrim zamanının hesaplanması.
- Mevcut çözümden onun komşularının birine hareket edilmesi ve bu diziye dayalı olarak amaç fonksiyonu değerinin hesaplanması.
    - Çevrim zamanındaki değişikliklerin hesaplanmasıyla birlikte yeni çözümün amaç fonksiyonu değeri öncekinden iyiyse yeni çözüm kabul edilecek, değilse önceden tanımlanmış olasılık fonksiyonuna bağlı olarak yeni çözümün kabul edilme şansı kontrol edilecektir.
    - Bu işlem her sıcaklıkta parça sayısı kadar tekrarlanacaktır.
- T parametresinin soğuma fonksiyonu ile azar azar düşürülmesi ile birlikte bitiş sıcaklığına kadar söz konusu işlemlerin tekrarı.

Yapılan açıklamalar ışığında, Şekil 5.2'de TB'nin çalışma şekline yönelik pseudo-kodu verilmiştir. Pseudo-kod içinde kullanılan notasyonlar şunlardır:

n : parça sayısı
T : Mevcut sıcaklık
T_i : Başlangıç sıcaklığı
T_f : Bitiş sıcaklığı
S^* : İterasyondaki en iyi çözüm
C^* : S^* çözümünün amaç fonksiyonu değeri
S : Mevcut çözüm
C : S çözümünün amaç fonksiyonu değeri
S_p : Geçici çözüm
C_p : S_p çözümünün amaç fonksiyonu değeri
r : Rassal sayı
d_r : Sıcaklık azaltma oranı
m_g : g aşamasında bulunan makine sayısı
ΔE : Amaç fonksiyonundaki değişiklik oranı

Algoritma A. Tavlama Benzetimi

1: Girdi: T_i, T_f, S, C
2: Çıktı: S, C
3: $T = T_i$
4: **while** ($T > T_f$) **do**
5: $C^* = +\infty$
6: $r = Rassal$
7: **for** ($i = 0,\ i < n,\ i++$)
8: $S_p = S$
9: **if** ($r >$ 0,5) **then**
10: **while** ($pos1 = pos2$) **do**
11: $pos1 = Rassal(n)$
12: $pos2 = Rassal(n)$
13: **end while**
14: **Swap**($S_p[pos1],\ S_p[pos2]$)
15: **else**
16: **while** ($k_c = k_n$ veya $m_g = 1$) **do**
17: $j = Rassal(n)$
18: $g = Rassal(m)$
19: $k_c = S[j + (g + 1) \times n]$
20: $k_n = Rassal(m_g)$
21: **end while**
22: $S_p[j + (g + 1)n] = k_n$
23: **end if**
24: C_p =**AmacFonksiyonuHesapla**(S_p)
25: **if** ($C_p < C^*$) **then**
26: $C^* = C_p$
27: $S^* = S_p$
28: **end if**
29: **end for**
30: $\Delta = 100 \times (C^* - C)/C$
31: **if** ($\Delta <$ 0 veya $r <$ exp $(-\Delta/T)$ **then**
32: $C = C^*$
33: $S = S^*$
34: **end if**
35: $T = T \times d_r$
36: **end while**

Şekil 5.2. Tavlama benzetimi algoritmasına ait pseudo-kodu

Pseudo-kod üzerinden gidilecek olursa, S, mevcut çözüm ve C, bu çözüme ait çevrim zamanı değerinin elde edilmesinden itibaren satır 3'teki önceden belirlenmiş bir sıcaklık değeriyle karşılaştırmalara başlanmaktadır. Satır $7 - 23$ arasında tanımlanan komşu çözüm

oluşturma işlemleri sonrasında elde edilen parça sayısı kadar komşu çözüm arasından, satır $25-28$'de, en iyi çevrim zamanı değerine sahip olan komşu çözüm tutularak satır $30-34$'te bu çözüm eldeki mevcut çözüm ile karşılaştırılır. Yapılan karşılaştırmalarda amaç fonksiyonundaki değişiklik oranı hesaplanarak daha iyi bir çözüm elde edilmişse yeni çözüm olarak alınmakta; elde edilen çözümün amaç fonksiyonu değeri daha kötüyse rassal bir değer ile tanımlanmış olasılık fonksiyonuna bağlı olarak hangi çözümle devam edileceğine karar verilmektedir. Sıcaklık değerinin satır 35'teki d_r oranına göre düşürülmesiyle birlikte bitiş sıcaklığına gelinene kadar bu işlemler devam etmektedir.

Probleme yönelik başlangıç çözümü rassal olarak oluşturulmaktadır. Kullanılan vektör yapısının ilk kısmında işlenmesi gereken tüm parçaların sıralanmasıyla birlikte, vektörün ikinci kısmındaki atamalar da 1. veya 2. makine olarak ayarlanmakta ve bir öncül çözüm bulunmaktadır. Örnek verilecek olursa; 4 parçanın üretileceği bir problemde parçaların işlem göreceği aşamalar Çizelge 5.2'de verildiği gibi olsun.

Çizelge 5.2. Parça-aşama eşlikleri

	1	2
1	✓	✓
2	x	✓
3	✓	✓
4	x	✓

Böyle bir sistemde, 1. ve 3. parçalar her iki aşamada da işlem görecekken, 2. ve 4. parçalar yalnızca 2. aşamada işlem görecektir. Sözkonusu probleme yönelik 12 elemanlı bir çözüm vektörünün aşağıdaki şekilde oluşturulması mümkündür:

$[\![3|4|1|2||1|1|1|1||2|1|1|2]\!]$

Bu çözümde ifade edilen; parçaların $3-4-1-2$ sırasıyla sisteme gireceği ve 3. ile 2. parçaların 2. aşamadaki 2. makinede, 4. ile 1. parçaların ise 2. aşamadaki 1. makinede işlem göreceğidir. Çizelge 5.2'de verilen bilgiler ve oluşturulan vektör ışığında makinelere ait parça atamaları (sıralı olarak) Çizelge 5.3'teki gibi olacaktır.

Çizelge 5.3. Makine / parça atamaları

1. aşama (1. makine)	3 ve 1
2. aşama (1. makine)	4 ve 1
2. aşama (2. makine)	3 ve 2

Mevcut çözümün iyileştirilmesi ve optimum sonuca yaklaşılması amacıyla daha iyi komşu çözümlerin aranması için iki farklı strateji kullanılmıştır. Ancak her iki algoritmada da her bir iterasyonda parça sayısı kadar yeni çözüm üretilmekte ve iterasyondaki en iyi çözüm komşu çözüm olarak alınarak devam edilmektedir.

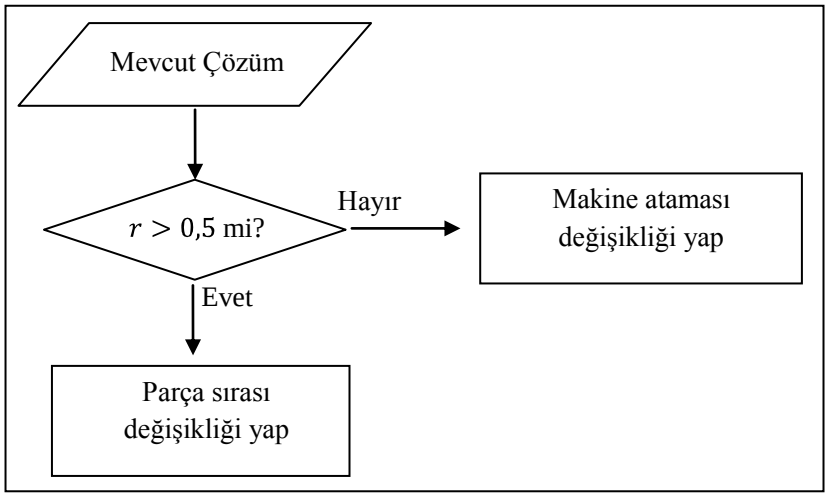
Rassal komşuluk

Önerilen ilk komşu çözüm üretme algoritması, Algoritma A'nın 7. ve 23. satırları arasındaki komşu çözümün üretiminde kullanılmak üzere verilmiştir. Bu algoritma için mevcut çözümden komşu bir çözüme geçişte rassal değişim hareketinden ve rassal seçimden yararlanılır. Mevcut çözüme uygulanacak hareket yine rassal olarak belirlenmektedir. $0-1$ değerleri arasından seçilecek rassal değer (r),

- $r > 0{,}5$ ise parça sırasında değişim hareketi (satır 9 ila 14),
- $r \leq 0{,}5$ ise makine atamalarında rassal değişim gerçekleştirilmektedir (satır 15 ila 22).

Rassal komşuluğa ait akış şeması Şekil 5.3'te verilmiştir.

Algoritma esnasında parça sırasında değişim yapılmasına karar verildiği takdirde vektörün ilk bölümünde, parçaların sisteme giriş sırasını gösteren değerler arasında rassal olarak seçilen iki parçanın ikili değişimi yapılmaktadır. Örneğin, $[\![3|\mathbf{4}|1|\mathbf{2}||1|1|1|1||2|1|1|2]\!]$ çözümünden üretilecek yeni bir çözümde 2. parça ile 4. parça yer değiştirdiğinde, $[\![3|\mathbf{2}|1|\mathbf{4}||1|1|1|1||2|1|1|2]\!]$ çözümü elde edilmekte; yeni durumda parçalar $3-2-1-4$ sırasıyla sisteme girerek, Çizelge 5.4'teki atamalara göre işlem görmektedir.



Şekil 5.3. Rassal komşuluk akış şeması

Çizelge 5.4. Değişim hareketi sonucu makine / parça atamaları

1. aşama (1. makine)	3, 1
2. aşama (1. makine)	2, 1
2. aşama (2. makine)	3, 4

Komşu çözüm oluşturulmasında rassal olarak seçilen bir parça için geçerli makine atamasının değiştirilmesi durumunda ise vektörün ikinci bölümündeki makine atamalarından biri değiştirilerek yeni çözüm elde edilmektedir. Örnek üzerinden devam edilecek olursa, ⟦3|4|1|2||1|1|1|1||2|1|**1**|2⟧ çözümünde 3. sırada sisteme giren 1 numaralı parçanın 2. aşamada atandığı makine değiştirilirse, ⟦3|4|1|2||1|1|1|1||2|1|**2**|2⟧ çözümü, Çizelge 5.5'te gösterilen makine/parça atamalarını veren olası bir komşu çözümdür.

Çizelge 5.5. Makine ataması seçimi sonucu makine/parça atamaları

1. aşama (1. makine)	3, 1
2. aşama (1. makine)	4
2. aşama (2. makine)	3, 1, 2

Bilinçli komşuluk

İkinci komşuluk yapısında ise önceki komşuluk yapısının rassallığını azaltmak amacıyla eldeki çözüm üzerinde belirlenen bazı değerlere göre değişiklik yapılmasına izin verilecek şekilde bir komşuluk yapısı tanımlanmıştır. Bu amaçla, komşu çözüm oluşturulmaya geçilmeden önce mevcut çözümdeki boş zaman ve bekleme zaman değerleri belirlenerek analiz edilir. Bu analizlerle sistemde görülen bekleme zamanları ve boş zamanlar sıralanarak öncelikle hangi noktalarda iyileştirme yapılabileceği belirlenmiş olur. Tüm aşama ve makineler için ilgili boş zaman ve bekleme zamanları hesaplandıktan sonra aşağıda verilen kurallara göre komşu çözüm oluşturulur.

Adım 1. En büyük boş zamanın (b_j) görüldüğü makineyi (j) belirle.

Adım 2. j makinesinin ait olduğu aşamada bekleme zamanı olup olmadığını kontrol et;

Bekleme varsa, ilgili makineyi (k) ve en büyük beklemenin (w_{ik}) görüldüğü parçayı (i) belirle.

Bekleme yoksa boş zaman listesini güncelleyerek Adım 1'e dön.

Adım 3. Belirlenen j ve k makinelerine göre *Kural 1* veya *Kural 2*'yi seçerek komşu çözüm için uygulanacak değişimi belirle.

Kural 1. En büyük boş zamanın olduğu makine (j) ile bu makinenin bulunduğu aşamadaki en büyük beklemenin olduğu makine (k) aynıysa, X_A değeri hesaplanır ve ilgili makinedeki en büyük beklemeye sahip olan parça ile bu makineye atanmış diğer parçalar arasından X_A değerine en yakın işlem zamanına sahip olan parçanın sırası değiştirilerek yeni çözüm elde edilir.

P_{ik} : en büyük beklemenin olduğu parçanın ilgili aşamadaki işlem zamanı

w_{ik} : belirlenen en büyük bekleme zamanı

$$X_A = P_{ik} - w_{ik} \tag{5.3}$$

Bu kural ile en büyük beklemeye sahip olan parçanın bulunduğu yere, bu noktada beklemeye sebep olmaması veya daha az beklemeye sebep olmasının muhtemel olduğu düşünülen bir parça getirilmeye çalışılmakta; böylece çevrim zamanını uzatan bekleme değerinden tasarruf edilerek toplam çevrim zamanının düşürülmesi hedeflenmektedir.

Beklemeye sebep olan parçalar ile boş zamana sebep olan parçalar arası değişim yapılmasıyla sistemdeki boş zaman ve bekleme zamanlarının dengelenmesine çalışılmakta, mevcut çözümdeki boş zamanlar doldurularak beklemelerin azaltılacağı ve alt sınır değerine yaklaşılacağı öngörülmektedir.

Kural 2. En büyük boş zamanın olduğu makine (j) ile bu makinenin bulunduğu aşamadaki en büyük beklemenin olduğu makine (k) farklı makinelerse, j makinesi için X_B ve k makinesi için X_C değerleri hesaplanır. k makinesindeki ilgili parça ile j makinesindeki parçalar arasından $|X_B - X_C|$ farkına en yakın işlem zamanına sahip olan parçanın makine atamaları değiştirilir.
$b_{j(k)}$: $j(k)$ makinesindeki boş zaman

$$X_B = b_j \tag{5.4}$$

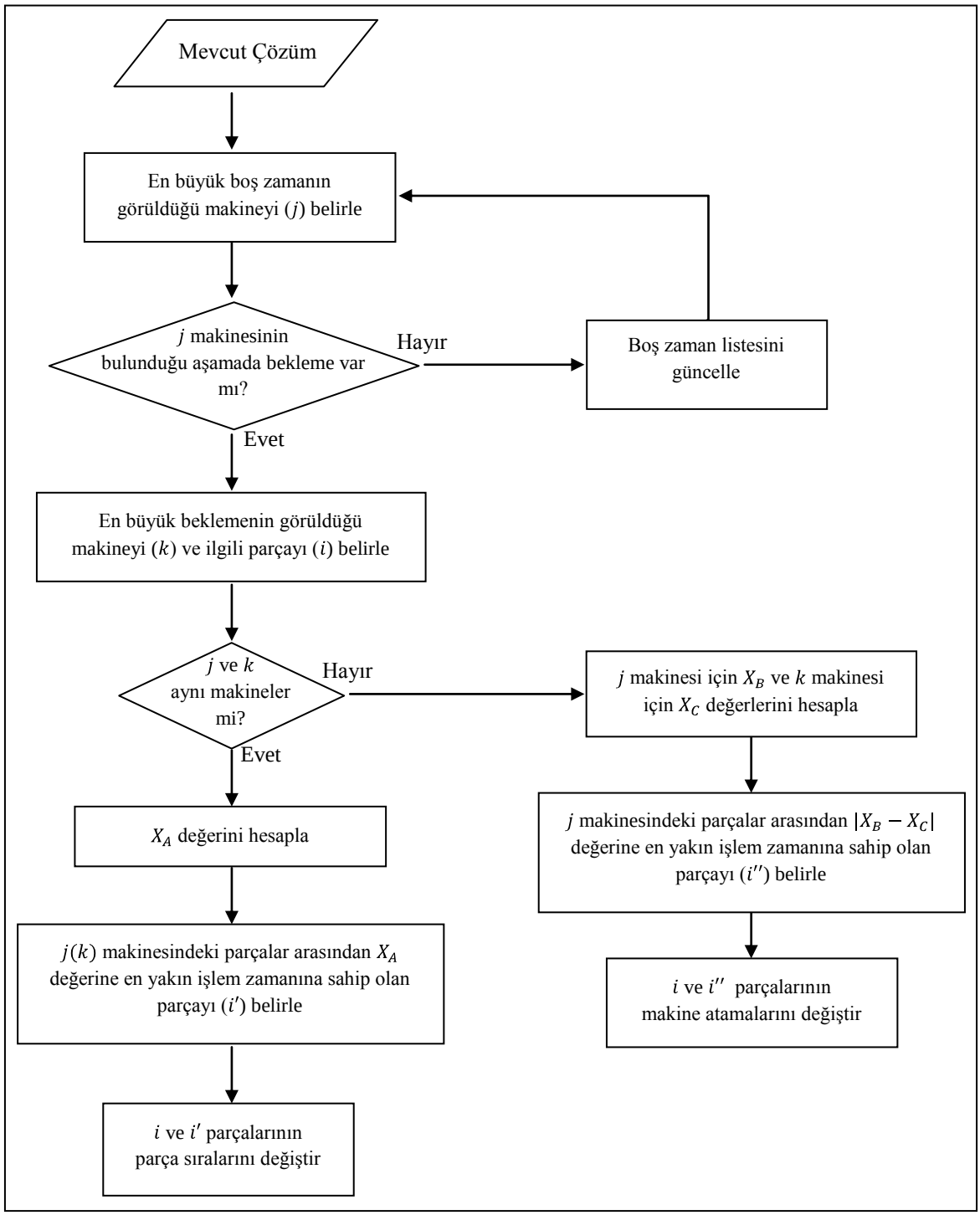
$$X_C = b_k + P_{ik} \tag{5.5}$$

Bu kural da sistemdeki boş zaman ve bekleme zamanlarının dengelenmesine yardımcı olmaktadır. En büyük beklemenin görüldüğü i parçası ilgili makineden alındığında makinede görülen mevcut boş zamanla birlikte bu parçanın işlem zamanı kadar yer açılmaktadır. j makinesindeki boş zaman ile k makinesinde elde edilen bu boş yer arasındaki farka yakın bir parça değişimi sağlandığında boş zamanlardan tasarruf edilerek çevrim zamanında iyileşme olabileceği ve böylece alt sınır değerine yaklaşılacağı öngörülmektedir.

Bilinçli komşuluğa yönelik akış şeması Şekil 5.4'te verilmiştir. Açıklanan adımların daha iyi anlaşılabilmesi için yine bir örnek problem üzerinden gidilecektir.

Örnek 2. 4 parçanın üretileceği 2 aşamalı bir sistemde ilk aşamada yalnız 1, ikinci aşamada 2 paralel makine bulunmaktadır. Aşamalara ait işlem zamanları aşağıda verildiği gibi olup eldeki mevcut çözüm ⟦3|4|2|1||1|1|1|1||1|2|2|1⟧ şeklindedir.
İşlem zamanları:
1. aşama: $10 - 0 - 0 - 15$
2. aşama: $15 - 17 - 14 - 16$

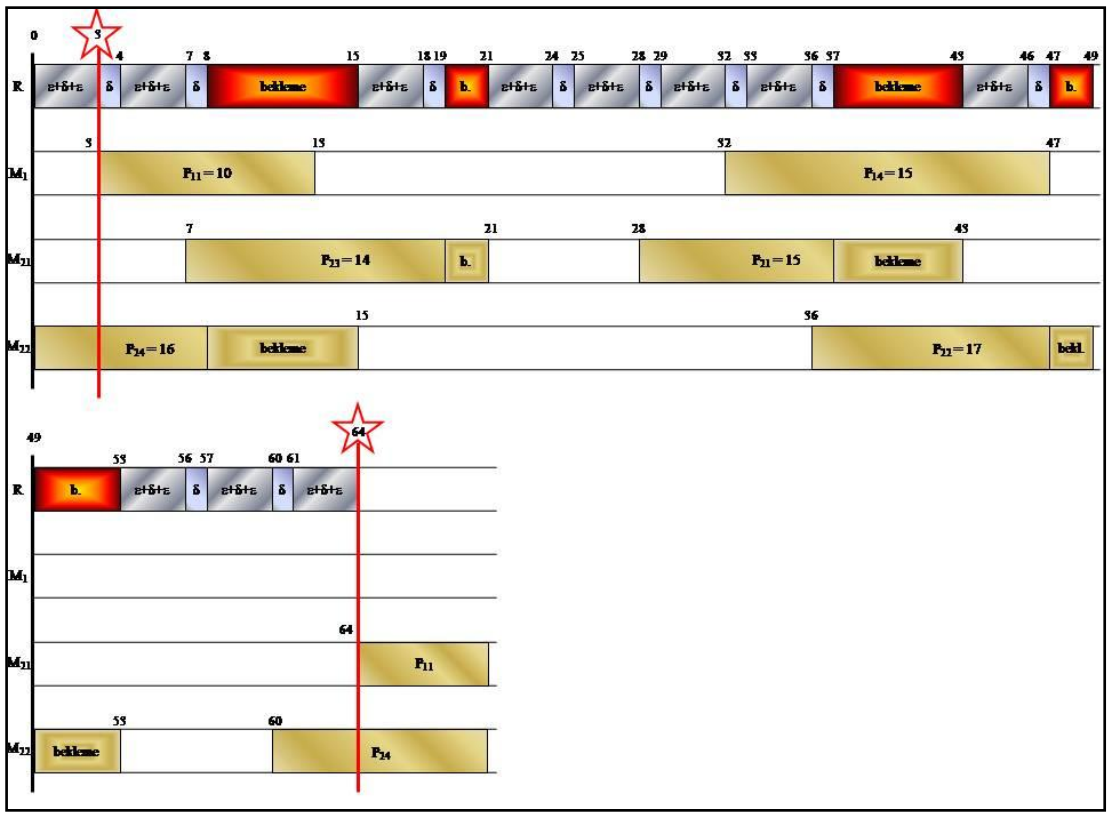
Eldeki vektörü yapısına göre parçaların sisteme giriş sırası $3-4-2-1$ şeklinde olup 2. aşama için parça/makine atamaları sırasıyla $1-2-2-1$ şeklindedir.



Şekil 5.4. Bilinçli komşuluk akış şeması

Örnek 2'de için elde edilmiş olan çözüme ait gantt-şeması gösterimi Şekil 5.5'te verilmiştir. Örnek çözümde 1. makine için toplamda 22 bz., 2. aşamanın ilk makinesi için

18 bz., ikinci makinesi için 14 bz.'lik boş zaman ile birlikte 2. aşamanın ilk makinesinde 3. parça için 2 bz., 1. parça için 6 bz., ikinci makinesinde 4. parça için 7 bz., 2. parça için 6 bz.'lık bekleme görülmektedir.



Şekil 5.5. Örnek 2 için Gantt-şeması gösterimi

Bu verilerden yola çıkılarak bilinçli bir yeni komşu çözüm oluşturulurken şu şekilde ilerlenmektedir:

Komşu çözüm algoritmasının ilk adımında, tüm sistemde en büyük boş zamanın hangi makinede oluşmuş olduğuna bakılır. Örnek problemde en büyük boş zaman 1. makinedeki 22 bz.'lık boşluktur.

Adım 2'de, mevcut çözümde en büyük boş zamanın görüldüğü makinede bekleme zamanı varlığı kontrol edildiğinde böyle bir parça bulunmamıştır. Bu sebeple bir önceki adıma geri dönülür.

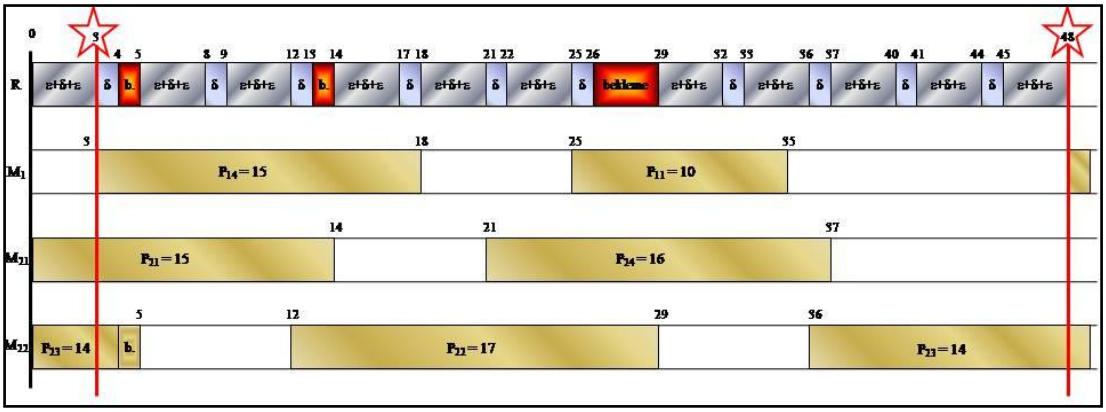
Sistemdeki boş zaman listesinde güncelleme yapılarak en büyük ikinci boş zaman olan makine belirlenir. Bu makine 2. aşamanın 1. makinesi olup ilgili boş zaman 18 bz. olarak karşımıza çıkmıştır.

Tekrar Adım 2'ye gelinerek ilgili aşamadaki bekleme zamanına bakılır. Bu aşamadaki mevcut en büyük bekleme 2. aşamanın 2. makinesinde 4 numaralı parçada 7 bz. olarak görülmektedir. Böylece yeni komşu çözüm ile ilgili ilk parça 4 numaralı parça olarak belirlenmiş olur.

Bu noktada boş zamanın olduğu makine ile en büyük beklemenin olduğu makinenin aynı makineler olup olmadığı kontrol edilir. Boş zamanın olduğu makine ile en büyük beklemenin olduğu makine farklı makineler olduğundan, Kural 2'ye göre komşu çözüm oluşturulacaktır. Boş zamanın olduğu makine için X_B ve beklemenin olduğu makine için X_C değerleri hesaplanır. Boş olan makineye atanmış parçalar arasından bu iki değer arasındaki mutlak farka en yakın işlem zamanına sahip olan parça ikinci parça olarak belirlenir.

En büyük beklemenin olduğu parça 4 numaralı parça olarak belirlenmiştir. İkinci parçanın belirlenmesi için $X_B = 18$ ile $X_C = 14 + 16 = 30$ değerleri hesaplanır. Aradaki fark $30 - 18 = 12$'dir. Boş olan makineye (2. aşamanın 1. makinesi) atanmış olan iki parça vardır ve bunların işlem zamanları 14 ve 15 bz.'dır. Bu ikisinden 12'ye yakın olan değer 14 olduğundan, ikinci parça olarak 3. parça belirlenir.

Belirlenen iki parça ayrı makinelere atanmış parçalar olduğundan parça sıralarına dokunmayıp parça/makine atamalarını değiştirerek yeni çözüm elde edilir. Bu durumda, 3. ve 4. parçaların atamaları değiştirildiğinde yeni çözümün parça sırası $3 - 4 - 2 - 1$ olarak aynı kalıp parça/makine atamaları $2 - 1 - 2 - 1$ olarak görülmektedir. Şekil 5.6'dan görülebileceği üzere bu çözümün çevrim zamanı 45 bz. çıkmıştır.

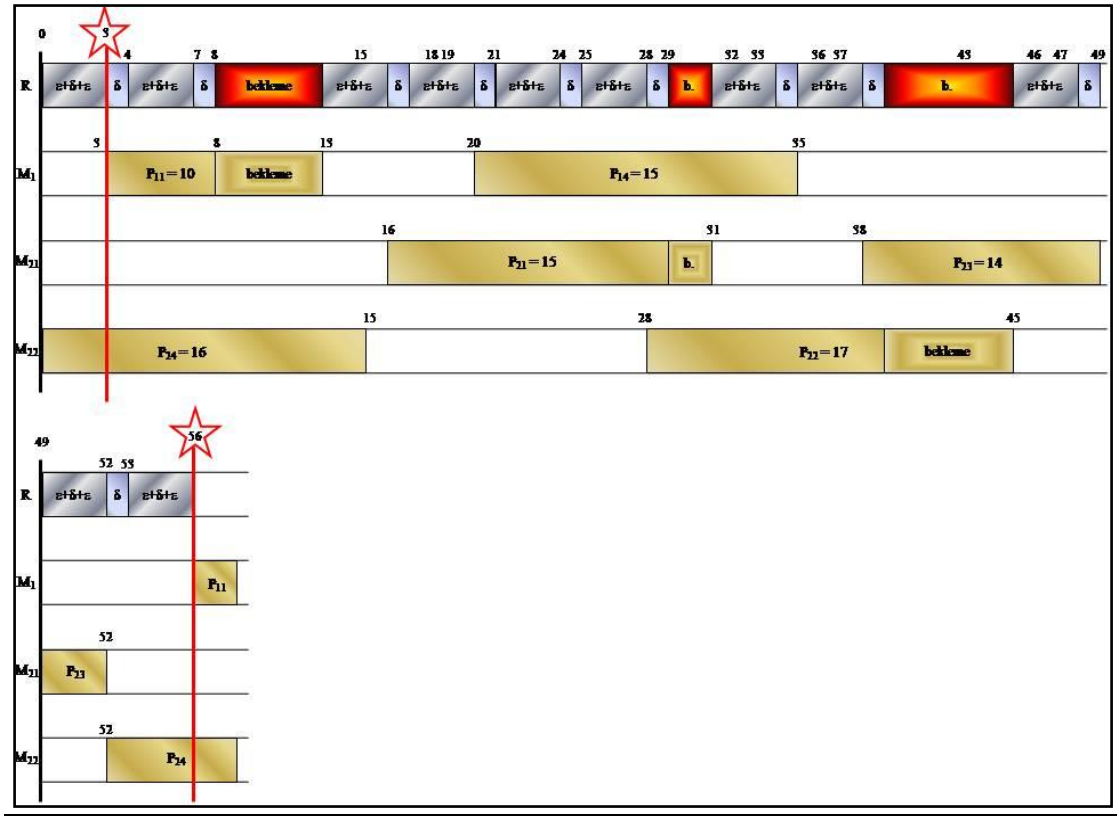


Şekil 5.6. Komşu çözüme ait Gantt-şeması

Alternatif bir çözümün daha görülebilmesi amacıyla, örnek üzerinden ikinci bir komşu çözüm oluşturulabilir. Bunun için, Adım 3'e gelindiğinde boş zamanın olduğu makine ile en büyük beklemenin olduğu makinenin aynı olduğu varsayılmıştır.

Bu durumda ikinci parçanın belirlenmesi için, Kural 1 gereğince, X_A değeri hesaplanır ve ilgili makineye atanmış diğer parçalar arasından bu değere en yakın işlem zamanına sahip olan parça ikinci parça olarak belirlenir.

Böyle bir seçim için, en büyük beklemenin 1 numaralı parçada olduğunu varsayarsak, ikinci parça olarak da ($15 - 6 = 9$'a en yakın işlem zamanına sahip olan parça) 3 numaralı parça olarak seçilir. Bu kararla birlikte, tüm parça/makine atamaları aynı kalarak iki parçanın sırası değiştirilerek yeni çözüm elde edilecektir. Yani; 3 ve 1 numaralı parçaların sırasını değiştirerek elde etiğimiz yeni çözümümüzün parça sırası $1 - 4 - 2 - 3$ olacak ve 2. aşama için parça/makine atamaları $1 - 2 - 2 - 1$ olarak kalacaktır. Şekil 5.7'den görüleceği üzere, bu şekilde elde edilecek bir çözümün çevrim zamanı 53 olarak hesaplanmıştır.



Şekil 5.7. Muhtemel komşu çözüme ait Gantt-şeması

Çözüm yönteminin önemli noktalarından biri çevrim zamanı hesaplanmasıdır. Söz konusu hesaplamanın pseudo-kodu Şekil 5.8-5.9'da verilmiştir. Kullanılan notasyonlar şunlardır:

V : çözüm vektörü

ε : yükleme/boşaltma zamanı

δ : hareket zamanı

P_{nm} : n parçasının m aşamasındaki işlem zamanı

S : hareket listesi

S' : mümkün hareket listesi

Z_j : parçanın girdi deposunda olup olmadığını belirten indis

mc_j : j parçasının bulunduğu makine

mn_j : j parçasının gideceği makine

D_i : i makinesinde yüklü olan parça

Algoritma B. Çevrim Zamanı

Girdi: V, ε, δ, $P_{11}, \ldots, P_{nm}$
Çıktı: S, C
$S = \emptyset$
for $(i = 0,\ i < n,\ i++)$
    $Z_i = 1$
end for
$CiktiDeposu = \sum_{i=1}^{m} m_i + 1$
$i = Vec[0]$
$S = SU\{i\}$
while $CevrimZamaniDongusu$ **do**
    while $TekrarDene$ **do**
        for $(i = 0,\ i < n_s,\ i++)$
            $j = S[i]$
            if $(Z_j = 1)$ **then**
                if $(D_{mn_j} = 0)$ **then**
                    $A_1 = 1$
            else if $(Z_j = 0)$ **then**
                if $(mn_j = CiktiDeposu)$ **then**
                    $A_2 = 1$
                else if $(D_{mn_j} = 0)$ **then**
                    $A_3 = 1$
            end if
            if $(A_1 = 1$ veya $A_2 = 1$ veya $A_3 = 1$ ve $t_j \leq t)$ **then**
                $S' = S'U\{j\}$
            end if
        end for
        if $(S' = \emptyset)$ **then**
            for $(i = 0,\ i < n_s,\ i++)$
                if $(Z_i = 0$ ve $t_i < t_{min})$ **then**
                    if $(mn_i = CiktiDeposu$ veya $D_{mn_i} = 0)$ **then**
                        $t_{min} = t_i$
                    end if
                end if
            end for
            $t = t_{min}$
            $TekrarDene = true$
        else if $(n_{S'} = 1)$ **then**
            $j = S'[0]$
        else
            for $(i = 0,\ i < n_s,\ i++)$
                $k = S'[i]$
                if $(Q[mn_k] < Q)$ **then**
                    $Q = Q[mn_k]$
                    $j = S'[i]$
                end if
            end for

Şekil 5.8. Çevrim zamanı hesaplamasına ait pseudo-kodu

```
47:             end if
48:     end while
49:     if (Z_j = 1) then
50:             Z_j = 0
51:             Guncelle(mn_j, mc_j, t, t_j)
52:             if (j = S[1]) then
53:                     Y_j = IlkParca
54:                     l = l + 1
55:                     L_S(l) = t
56:                     C_Tp = L_S(l−1) − L_S(l−2)
57:                     C_Tc = L_S(l) − L_S(l−1)
58:                     if (C_Tp = C_Tc)
59                              CevrimZamaniDongusu = false
60:                     else
61:                             CevrimZamaniDongusu = true
62:                     end if
63:             end if
64:             i = Vec[j]
65:             S = SU{j}
66:     else if (Z_j = 0) then
67:             if (mn_j = CiktiDeposu) then
68:                     Guncelle(mn_j, mc_j, t, t_j)
69:             else
70:                     D_mc_j = 0
71:                     Guncelle(mn_j, mc_j, t, t_j)
72:             end if
73:     end if
74: end while
```

Şekil 5.9. Çevrim zamanı hesaplamasına ait pseudo-kodu (devam)

Sözkonusu algoritmanın çalışma adımları şu şekildedir:

Adım 1. Tüm parçalar girdi deposuna yüklenir (satır 4)

Adım 2. Satır 9'da görüldüğü gibi, çözüm vektörünün ilk elemanı olan parça olay kuyruğuna eklenir.

Adım 3. Olay kuyruğundaki parçalar için mümkün olay kümesi belirlenir.

Eğer olay kuyruğundaki ilk sıradaki parça, satır 14'teki gibi, girdi deposunda ise ve eğer parçanın gideceği makine boşsa $A_1 = 1$ olarak belirlenir.

Değilse,

Eğer parçanın gideceği yer çıktı deposuysa $A_2 = 1$ olarak belirlenir.

Değilse,

Eğer parçanın gideceği makine boşsa $A_3 = 1$ olarak belirlenir.

Algoritmanın 23. satırındaki gibi, $A_1 = 1$ veya $A_2 = 1$ veya $A_3 = 1$ ve parçalardan en az birinin işlemi tamamlanmış ise ilgili parça mümkün olay kümesine eklenir.

Adım 4. Olay kümesindeki tüm parçaların kontrolü bittiğinde, eğer mümkün olay kümesi, satır 27'deki gibi, boşsa (ki bu makinelerin hepsinin dolu veya robotun meşgul olmasını gerektirir) sistem saati satır 31'deki gibi en erken tamamlanabilecek işe göre güncellenir ve Adım 3'e dönülür.

Değilse,

Eğer, satır 37'deki gibi, yapılabilecek tek bir iş varsa o iş yapılır.

Değilse,

Eğer birden fazla mümkün olay varsa, $42 - 45.$ satırlardaki gibi, makineler arası öncelik durumuna göre seçim yapılarak iş seçilir.

Adım 5. Eğer yapılacak iş, satır 49'daki gibi, bir parçanın girdi deposundan alınması ise parça, makine ve zaman güncellemeleri yapılarak parça ilgili makineye taşınarak yüklenir.

Eğer seçilen iş çözüm vektöründeki ilk iş ise bu işin son iki çevrimde gerçekleştirildiği zamanlar arasındaki farklar alınarak çevrim zamanları hesaplanır ve son iki çevrim zamanının eşit olması durumunda $CevrimZamaniDongusu$ olarak adlandırılan çevrim zamanının döngüye girdiğine işaret eden değişken güncellenir.

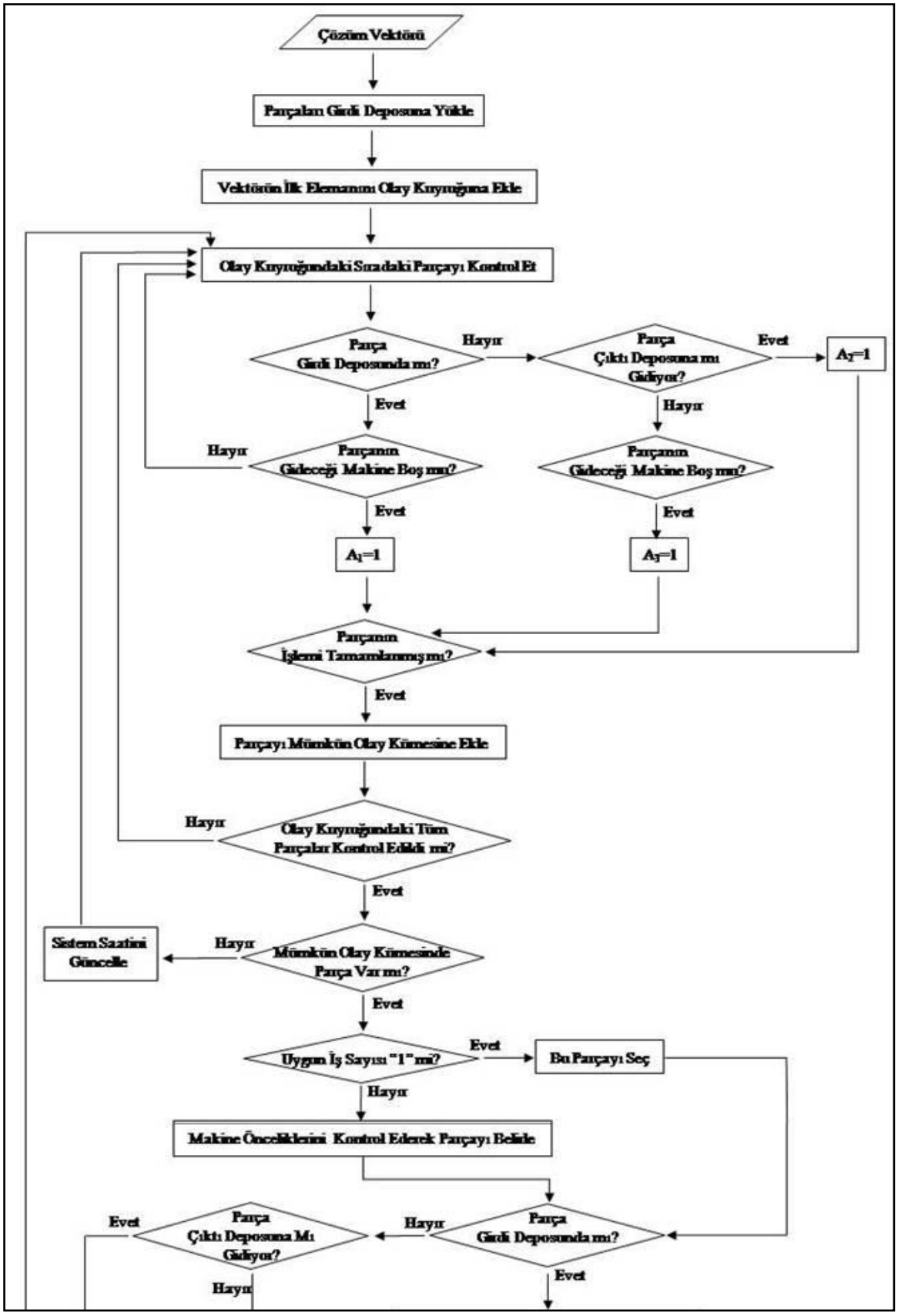
Değilse,

Eğer parça çıktı deposuna veya başka bir makineye gidecekse parça yüklü olduğu makineden alınarak çıktı deposuna götürülüp ilgili güncellemeler yapılır.

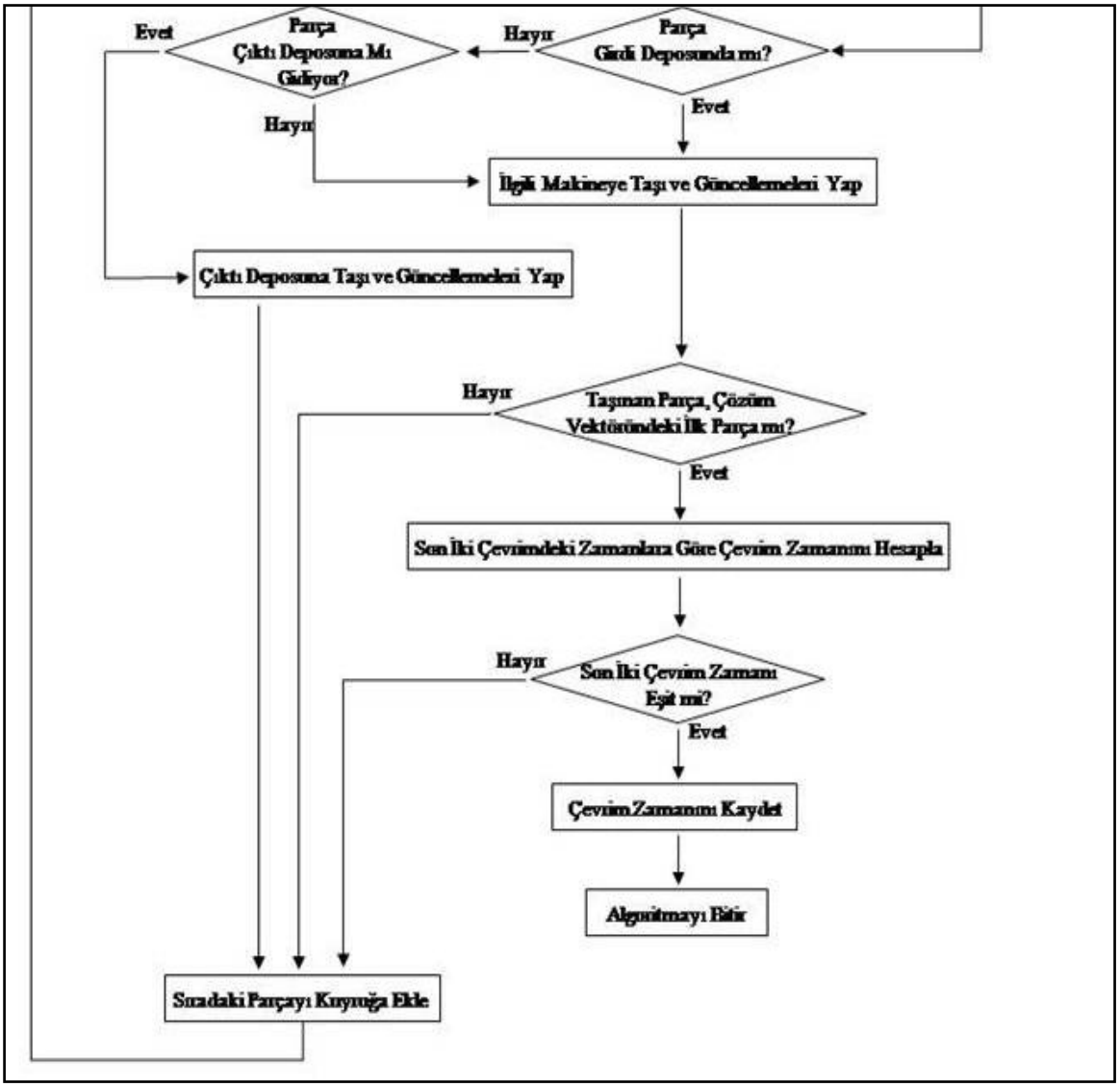
Adım 6. Satır 64'teki gibi, vektörün sıradaki parçası olay kuyruğuna eklenir ve Adım 3'e dönülür.

Adım 7. Çözüm vektöründeki tüm parçalar sisteme sokulduğunda Adım 1'e dönülerek girdi deposuna yeni bir parça kümesi yüklenir. $CevrimZamaniDongusu$ değişkeni ile tutulan, son iki çevrim zamanının eşitliğine kadar bu işlemlere devam edilir. Son iki çevrim zamanı eşit olarak bulunduğunda bulunan çevrim zamanı sonucu verecektedir.

Sözkonusu hesaplamalara yönelik akış şeması Şekil 5.10 ve Şekil 5.11'de verilmiş olup ilgili adımlar aşağıda Örnek 3 üzerinden açıklanmıştır.



Şekil 5.10. Çevrim zamanı akış şeması



Şekil 5.11. Çevrim zamanı akış şeması (devam)

Örnek 3: 3 parçanın işlem göreceği bir sistemde ilgili işlem zamanları ilk aşama için $P_{11} = 6$, $P_{12} = 12$ ve $P_{13} = 0$, ve ikinci aşama için $P_{21} = 10$, $P_{22} = 14$ ve $P_{23} = 20$ olarak bilinmektedir. Yükleme/boşaltma (ε) ve hareket (δ) zamanlarının her ikisi de 1 birim zamandır.

Ele alınan sistem için [1|3|2||1|1|1||2|1|2] şeklinde bir çözüm elde edilmiştir. Eldeki işlem zamanı verilerine bakıldığında 1 ve 2. parçaların her iki aşamada da, 3. parçanın ise yalnızca 2. aşamada işlem göreceği görülmektedir. Eldeki çözüm vektörüne göre ise 1 ve 2 numaralı parçaların 2. aşamanın 2. makinesinde, 3 numaralı parçanın ise aynı aşamanın 1. makinesinde işlem görmesi gerekmektedir.

Başlangıç ($t = 0$) anında tüm parçalar hazır olarak girdi deposunda bulunmaktadır. Bu noktada eldeki çözüm vektörüne göre Adım 2'deki olay kuyruğuna ilk eklenecek parça 1 numaralı parçadır. Mümkün olay listesi oluşturulurken mevcut olay kuyruğundaki bu ilk parça girdi deposunda olup gideceği makine de boş olduğundan Adım 3'te $A_1 = 1$ olarak belirlenir ve 1 numaralı parça mümkün olay kümesine eklenir. Bu durumda yapılabilecek olan tek bir iş olduğundan Adım 4'te ve algoritmanın 37. satırında belirtildiği üzere, bu iş seçilir.

Verilen kararla birlikte, robot, $t = 0$ anından itibaren sırasıyla, 1 numaralı parçayı, girdi deposundan ε zamanda alacak, işlem göreceği ilk aşama olan 1. aşamanın tek makinesi olan 1. makinesine δ zamanda taşıyacak ve ε zamanda bu makineye yükleyecektir. Bu hareketlerle birlikte $t = 2\varepsilon + \delta = 3$ olarak güncellenecektir. Bu zaman, ilk çevrimin başladığı zaman olarak da kaydedilmektedir.

Robotun sonraki adımda ilgileneceği parça belirlenirken, Adım 6'dan takip edileceği üzere, tekrar çözüm vektörüne bakılarak sıradaki parça olay kuyruğuna eklenir. Sıradaki parça *3* numaralı parçadır. Parçanın işlem göreceği makine 2. aşamanın 1. makinesi olup ilgili makine boş durumdadır. Adım 3'e bakıldığında yine $A_1 = 1$ olarak belirlenmekte ve parça, mümkün olay kuyruğuna eklenmektedir. 1 numaralı parçada olduğu gibi bu parça da girdi deposundan alınıp ilgili makineye taşınacak ve yüklenecektir. Bu adımlar sonunda $t = 7$ olmuştur.

Çözüm vektöründe sırada 2 numaralı parça vardır. Bu parçanın işlem göreceği ilk makine 1. aşamanın 1. makinesi olarak görülmektedir. Bu makine mevcut durumda dolu olduğundan kendisiyle ilgili bir hareket yapılması mümkün değildir. Bu durumda makinelerde yüklü olan diğer parçalar dikkate alınarak Adım 3'teki A_2 veya A_3 şartlarını sağlayan bir parça olup olmadığı kontrol edilir. Mevcut durumda tüm parçaların işlemleri devam etmekte olup bu şartları sağlayabilecek bir parça olmadığından Adım 4'e gidilir. Bu anda 1. aşamanın 1. makinesinde yüklü olan 1 numaralı parçanın tamamlanma zamanı $t = 9$, 2. aşamanın 1. makinesinde yüklü olan 3 numaralı parçanın tamamlanma zamanı $t = 27$'dir. Adım 4'e göre, sistem saati $t = 9$ anına getirilerek 1 numaralı parça için Adım 3'teki şartlar tekrar kontrol edilir ve parçanın mümkün olay listesine eklenip eklenemeyeceğine bakılır. Bu noktada 1 numaralı parçanın sonraki işlem noktası olan 2.

aşamanın 2. makinesi boş olduğundan $A_3 = 1$ olmakta ve parça önce mümkün olay kümesine alınıp daha sonra da bu kümedeki tek iş olduğundan ilgili hareketler gerçekleştirilir. Böylece $t = 9$ anından itibaren, parça 1. aşamanın 1. makinesinden alınacak (ε), 2. aşamanın 2. makinesine taşınacak (δ) ve yüklenecektir (ε).

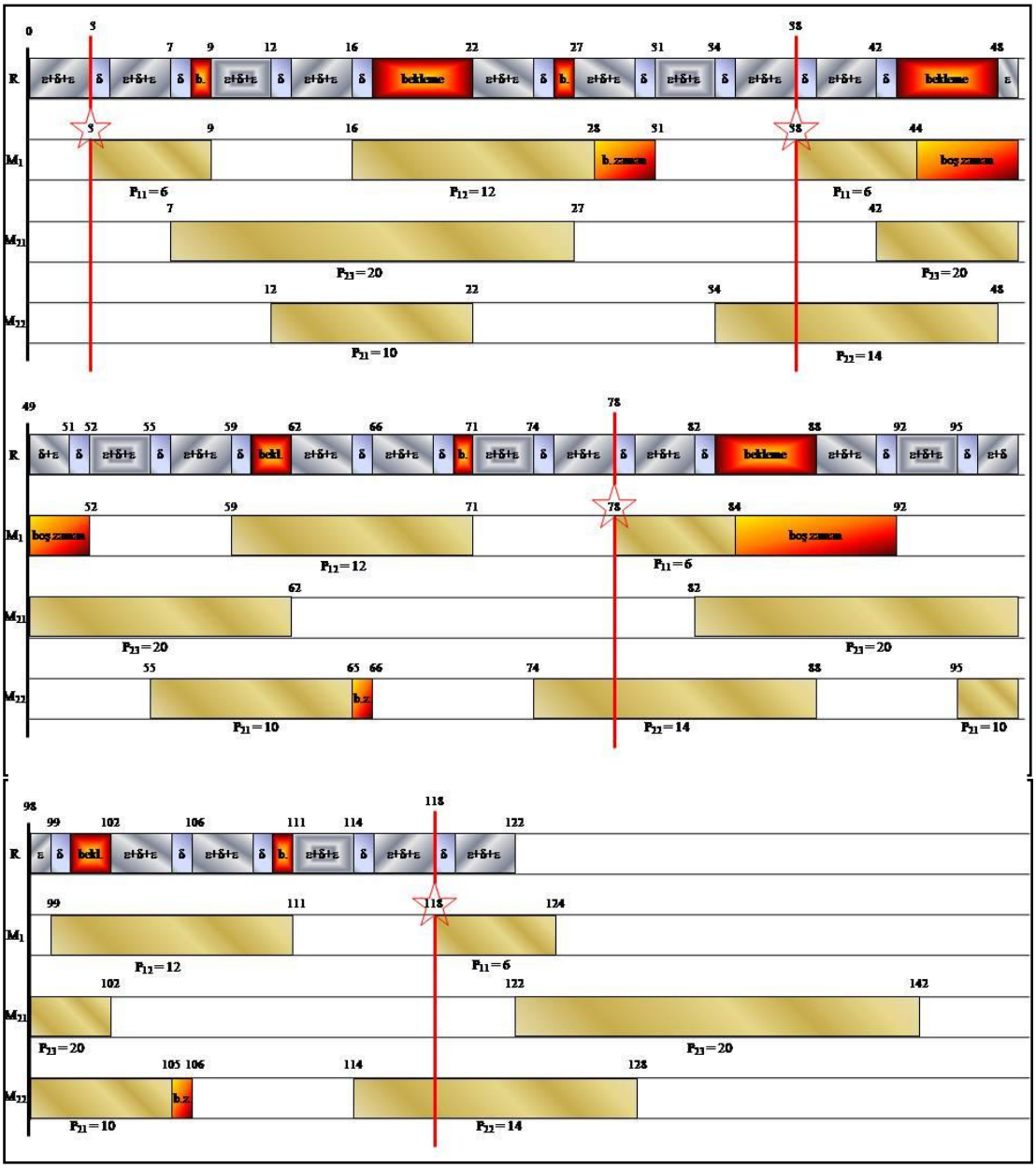
Olay kuyruğuna tekrar bakıldığında halen sırada olan 2 numaralı parça için $A_1 = 1$ şartı sağlanmıştır. Bu parça mümkün olay kümesine eklenerek ilgili hareketler gerçekleştirilir. Bunun için robot tekrar girdi deposuna gidecek (δ), 2 numaralı parçayı alacak (ε), parçayı 1. aşamanın 1. makinesine taşıyacak (δ) ve yükleyecektir (ε).

Bu anda bir parça setine ait tüm parçalar sisteme girmiş durumdadır. Girdi deposuna yeni bir parça kümesi yüklenecektir. Yeni kümedeki parçalar da önceki ile aynı sırada işleme sokulacaktır. Buna göre sırada 1 numaralı parça vardır, ancak bu parçanın işlem göreceği 1. aşamanın 1. makinesinin dolu olduğu görülmektedir. Bu durumda daha önce de yapıldığı gibi, Adım 4'e gidilerek makinelerin boşaltılabileceği en erken zamanlar karşılaştırılır. Boşaltılabilecek ilk parça 2. aşamanın 2. makinesinde yüklü olan 1 numaralı parçadır. Makinenin boşaltılması için parçanın işleminin bittiği ana ($t = 22$) kadar beklenip parça bu makineden alınacak, çıktı deposuna taşınacak ve bırakılacaktır. Sıradaki parça olan 1 numaralı parçanın yüklenmesi için uygun şart halen sağlanmamış olduğundan, yine en erken boşaltılabilecek makine belirlenir. $t = 25$ anında boşaltılabilecek ilk parça 2. aşamanın 1. makinesinde yüklü olan 3 numaralı parçadır. Makinenin boşaltılması için parçanın işleminin bittiği ana ($t = 27$) kadar beklenip parça bu makineden alınarak çıktı deposuna bırakılacaktır. Bu hareketin ardından boşaltılabilecek makine *1inci* aşamanın *1inci* makinesidir. Bu makinede 2 numaralı parça yüklüdür. Boşaltma için robot $t = 31$ anında makinenin önüne gidebilmiş, parçanın işleminin tamamlandığı $t = 28$ anından bu ana kadar makine boş kalmıştır. 2 numaralı parçanın bir sonraki işlem noktası 2. aşamanın 2. makinesidir; bunun için parça 1. aşamanın 1. makinesinden alınıp 2. aşamanın 2. makinesine taşınacak ve $t = 34$'te makineye yüklenecektir.

Tekrar kontrol edildiğinde 1 numaralı parçanın yüklenmesinin mümkün olduğu görülmektedir. Parça girdi deposundan alınıp $t = 38$ anında 1. aşamanın 1. makinesine yüklenecektir. Bu zaman ikinci çevrimin başladığı zaman olarak kayıt altına alınmaktadır. Bu şekilde sürekli yenilenerek devam eden adımlar sırasında her bir kümenin başlangıç

zamanları tutularak bu zamanlar arasındaki farklar hesaplanmakta ve ardışık kümelere ait zamanların eşitliğinde çevrim zamanı değeri bulunmaktadır.

Açıklanan adımların Şekil 5.12'den de takip edilmesi mümkündür. Problemde ilk çevrimin başlama zamanı $t = 3$, ikinci çevrimin başlama zamanı $t = 38$, üçüncü çevrimin başlama zamanı $t = 78$, dördüncü çevrimin başlama zamanı $t = 118$'dir. Bu aralıklarda çevrim zamanı değerleri sırasıyla $C_1 = 35$, $C_2 = 40$, $C_3 = 40$ olarak hesaplanmış; son iki değer birbirinin aynısı olduğundan çözümün sonucu $C = 40$ olmuştur.



Şekil 5.12. Örnek 3 için Gantt Şeması gösterimi

Oluşturulan sezgisel algoritma MS Visual C++ programlama dili kullanılarak kodlanmış ve çalıştırılmıştır. Kullanılan kod EK-3'te verilmiş olup, algoritmaya yönelik bir örnek problem veri kümesi ile ilgili probleme yönelik çözüm çıktısı EK-4'te gösterilmiştir.

6. ALT SINIR DEĞERİ VE PERFORMANS ANALİZİ

Algoritmanın etkinliği üzerine net fikir yürütebilmek için öncelikle probleme yönelik bir alt sınır değeri geliştirilmiştir. Bu noktada, aşamalarda işlem görecek parçalara bakılarak her bir aşamaya ait toplam işlem zamanı, ilgili aşamadaki makine sayısına bölünerek aşamaya ait minimum gerekli zaman bulunmuş ve tüm aşamalar için hesaplanan bu değerden yararlanılarak alt sınır değeri bulunmuştur.

Teorem:

Çoklu parça tipi üretimi yapılan ve makineler/aşamalar arasındaki taşımanın bir robot tarafından gerçekleştirildiği, sabit hareket zamanlı hibrit esnek akış tipi hücrelerde gözlenen çizelgeleme problemi için alt sınır değeri şu şekildedir:

$$Alt\ Sınır\ (A.S.) = maks_j \left\{ \sum_i p_{ij} + n_j(4\varepsilon + 3\delta) \Big/ m_j \right\} \tag{6.1}$$

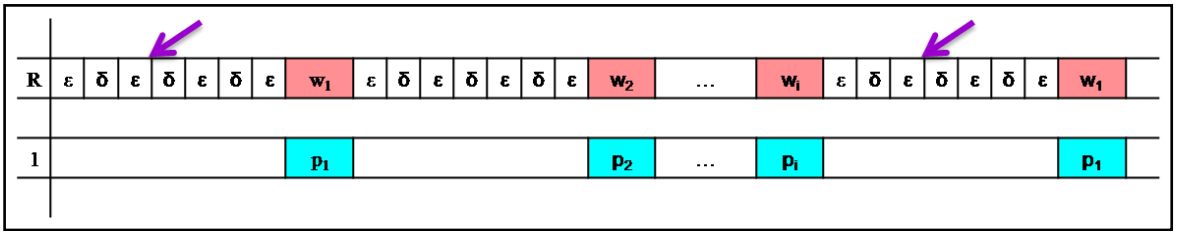
Burada j aşamaları, i işlem görecek parçaları, p_{ij} i parçasının j aşamasındaki işlem zamanını, n_j j aşamasında işlem görecek parça sayısını, m_j ise j aşamasındaki makine sayısını belirtmektedir. Denklemde dikkat çeken $4\varepsilon + 3\delta$ değeri, sistemdeki tüm parçaların işlem göreceği tüm aşamalar için geçerli olup, sözkonusu parçanın girdi deposundan veya bir önceki aşamadan alınması (ε) buradan ilgili aşamaya taşınarak (δ) bırakılması (ε) ve tekrar oradan alınarak (ε) diğer bir aşamaya veya çıktı deposuna taşınması (δ) ve bırakılması (ε) esnasında ihtiyaç duyacağı toplam zamanı göstermektedir. Sözü edilen alt sınır değerinin aşağıdaki açıklama ve şekillerden görülmesi mümkündür.

İspat:

Alt sınır değeri belirlenirken en küçük sistemden başlanarak ele aldığımız hibrit akış tipi sisteme varılmıştır. Bunun için önce tek aşamalı tek makineli hücre yapısı ile başlanacaktır.

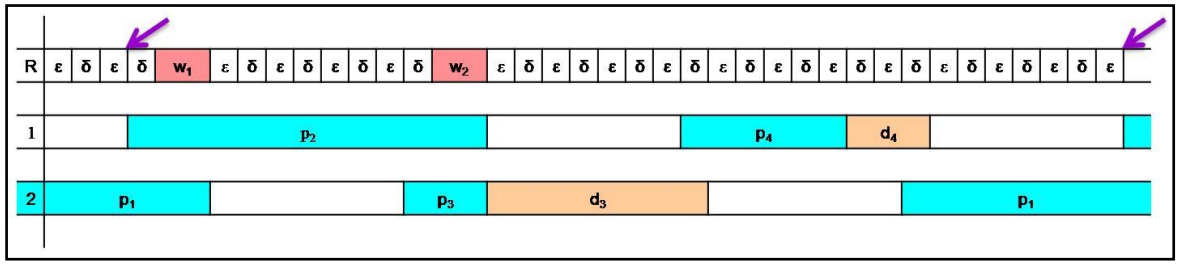
Durum 1. Şekil 6.1'deki gibi bir tek aşamalı tek makineli sistemde, robot girdi deposundan işlem görecek olan parçayı alacak (ε), makineye taşıyacak (δ), yükleyecek (ε), parçanın

işlem süresi kadar bekleyecek ($w_1 = p_1$), işlemi tamamlanan parçayı alacak (ε), çıktı deposuna taşıyacak (δ), bırakacak (ε) ve yeni parça almak için girdi deposuna geri dönecektir. Bu şekilde bir i parçası için geçecek olan toplam zaman $p_i + 4\varepsilon + 3\delta$ olmaktadır. Tüm parçalar için aynı şekilde olacak olan hareket dizisinin tamamlanmasıyla çevrim zamanı $\sum_i w_i + n(4\varepsilon + 3\delta) = \sum_i p_i + n(4\varepsilon + 3\delta)$ olur. Şekilde a ve b noktaları arasındaki çevrimde bu zamanın görülmesi mümkündür.



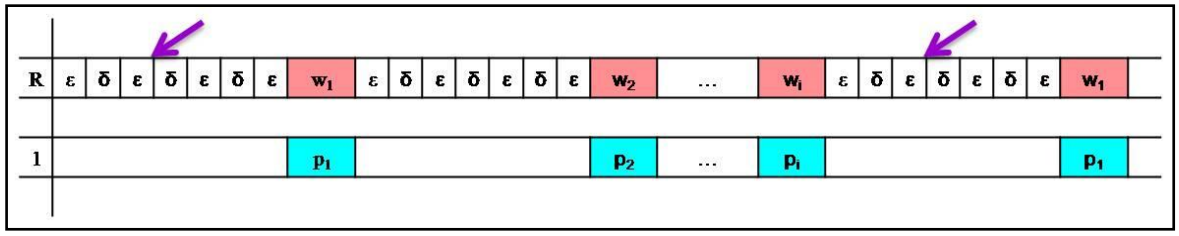
Şekil 6.1. Tek aşamalı tek makineli sistemde çevrim zamanı

Durum 2. Buradan Şekil 6.2'deki, yine tek aşamalı ancak aşamada birden fazla makine olduğu duruma geçersek, herbir parça için gerekli toplam zaman yine $p_i + 4\varepsilon + 3\delta$ olacak, fakat bu sistemde robot tek makine yerine birden fazla makine ile ilgileneceğinden bekleme zamanlarının yanı sıra makinelerde boş zamanlar da oluşabilecektir. Bu durumda bir j makinesi için geçecek toplam zaman, p_{ij} ilgili makineye atanmış işlem zamanını ve d_{ij} boş zaman değerlerini vermek üzere, $\sum_i p_{ij} + 4\varepsilon + 3\delta + \sum_i d_{ij}$ olacaktır. Sözkonusu boş zaman değeri robot hareketlerine göre belirlenecek olduğundan, alt sınır hesabında olası en küçük değeri olan sıfıra eşit kabul edilmesi mümkündür. Bu durumda j makinesi için geçecek toplam zamanın en az $\sum_i p_{ij} + 4\varepsilon + 3\delta$ kadar olacağı söylenebilir. Birden fazla makinenin olduğu sistemde, ideal duruma makinelerin tam olarak eşit yüklenmesi ile ulaşılacağını tahmin etmek zor değildir. Bu mantıktan yola çıkıldığında, tüm sistemdeki parçalar için gerekli toplam zaman, n parça sayısını vermek üzere, $\sum_j \sum_i p_{ij} + n(4\varepsilon + 3\delta)$ olmakta; bu zamanın makinelere eşit dağıtılmasıyla ise, m makine sayısını vermek üzere, $(\sum_j \sum_i p_{ij} + n(4\varepsilon + 3\delta))/m$ değeri bulunmaktadır.



Şekil 6.2. Tek aşamalı çok makineli sistemde çevrim zamanı

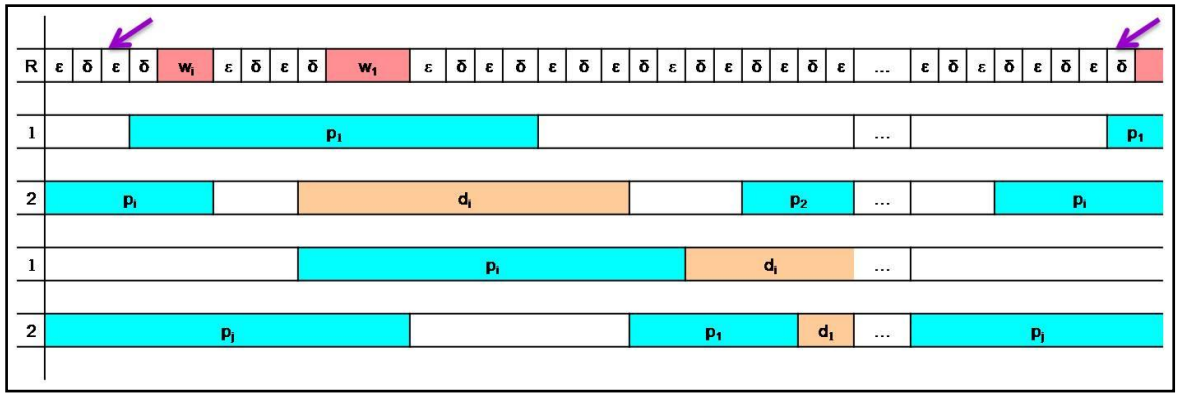
Durum 3. Çok aşamalı ve aşamaların her birinde yalnız bir makine olduğunda, robot birden fazla makine ile ilgileneceğinden, sistemde yine boş zamanların gözlenme olasılığı olacaktır. Bir önceki durumdaki gibi her bir makine için gerekli toplam zaman $\sum_i p_{ij} + 4\varepsilon + 3\delta + \sum_i d_{ij}$ olacak, boş zamanın sıfır kabul edilmesiyle her bir makine için alt sınır $\sum_i p_{ij} + 4\varepsilon + 3\delta$ olacaktır. Şekil 6.3'te görülen, çok aşamalı sistemde tüm parçalar tüm aşamalarda işlem görecek olup, her bir aşama da yalnızca kendi yapacağı işlemlerden sorumlu olduğundan makineler arası paylaşım yapılamayacak, bunun yerine aşamalar arasından işlemleri geç tamamlananana göre çevrim zamanı belirlenecektir. Bu durumda alt sınır değeri, $maks_j\{\sum_i p_{ij} + n(4\varepsilon + 3\delta)\}$ olmaktadır.



Şekil 6.3. Çok aşamalı tek makineli sistemde çevrim zamanı

Durum 4. Son olarak, ele aldığımız çok aşamalı ve aşamaların birinde birden fazla makinenin bulunduğu sistemlere gelecek olursak, Şekil 6.4'ten bu sistemlerin önceki iki yapının karması olduğu görülebilir. Aşamalara ayrı ayrı bakıldığında; gerekli toplam zaman, tek makinenin olduğu aşama için $\sum_i p_{ij} + 4\varepsilon + 3\delta$, birden fazla makinenin olduğu aşama için ise makinelere eşit olarak dağıtılabileceğinden $(\sum_j \sum_i p_{ij} + n(4\varepsilon + 3\delta))/m$ olmaktadır. Çok aşamalı sistemlerde aşamalardan geç bitene göre çevrim zamanının belirlendiğini bildiğimize göre, çok aşamalı ve aşamaların birinde birden fazla makinenin bulunduğu sistemde çevrim zamanı, j aşamaları ve m_j aşamalardaki makine sayısını göstermek üzere, $maks_j\{(\sum_i p_{ij} + n(4\varepsilon + 3\delta))/m_j\}$ olacaktır. Sistemimizde esneklik

özelliği bulunup, bazı parçaların bazı aşamalarda işlem görmemesine izin verildiğinden, n ile gösterilen parça sayısı yerine j aşamasında işlem görecek parça sayısını göstermek üzere n_j değişkeni kullanılması daha uygundur. Bu değişiklikle birlikte alt sınır değeri $maks_j\{(\sum_i p_{ij} + n_j(4\varepsilon + 3\delta))/m_j\}$ olmuştur.



Şekil 6.4. Çok aşamalı çok makineli sistemde çevrim zamanı

□

Üretilen test problemlerinin önerilen algoritma ile çözülmesi ile elde edilen sonuçlar tanımlanan A.S. değeri ile karşılaştırılırken aşağıdaki eşitlik kullanılmıştır:

$$\%\ iyileşme = {A.S. - C(Algoritma)}\Big/{C(Algoritma)} \times 100 \qquad (6.2)$$

Geliştirilen algoritmanın gözlemlenebilmesi için farklı boyutlarda test problemleri üretilmiş ve denenmiştir. Hall ve arkadaşları taşımalarının bir robot tarafından gerçekleştirildiği bir üretim sistemindeki işlerin çizelgelenmesi üzerinde durmuş ve çalışmalarında çoklu parça tiplerinin denge durumu üretiminde hücrenin bir veya iki EAPK sonra aynı duruma döndüğünü gözlemlemişlerdir (Hall ve diğerleri, 1997b). Benzer şekilde, yapılan denemelerde de genellikle iki veya üç EAPK sonrasında hücreye özgü bir çevrimsel üretim planına ulaşıldığı görülmüştür.

Ele alınan probleme ait başlıca parametreler parça sayıları, parçalara ait işlem zamanları, yükleme/boşaltma ve hareket zamanları ile sistemde bulunan aşama ve her bir aşamaya ait makine sayılarıdır. Bu adımda 2 aşamalı bir sistemde aşamalardan birinde yalnız 1,

diğerinde 2 makinenin bulunduğu yapıya yönelik test problemleri kullanılmıştır. Çizelge 6.1'de verilen parça sayısı, işlem zamanı, yükleme/boşaltma ve hareket zamanı özelliklerine göre her bir problem tipinden 10'ar örnek üretilerek 2430 problem elde edilmiş ve her bir örnek problem verilen sıcaklık değerlerinin her biri için 5'er kez çalıştırılarak bulunan en iyi sonuçlar dikkate alınarak analizler tamamlanmıştır.

Çizelge 6.1. Test problemlerine yönelik parametre değerleri

Parça sayısı	$4-5-10-15-20-25-50-100-150$
İşlem zamanı	$0-200,\ 0-300,\ 0-500$ birim zaman aralıkları
Yükleme/boşaltma zamanı	$1, 6, 10$ % (ortalama işlem zamanı üzerinden)
Hareket zamanı	$1, 6, 10$ % (ortalama işlem zamanı üzerinden)
Başlangıç sıcaklığı	$T_i = 100, 168, 196, 313, 665$

Algoritma, Microsoft Visual Studio 2010 kullanılarak C++ programlama dili ile kodlanmış ve çalıştırılmıştır. Yukarıda verilen Eş. 6.2 yardımıyla yapılan karşılaştırmalar sonucunda denenen tüm problemler için alt sınır değerinden ortalama yüzde sapma değerleri hesaplanmıştır. Elde edilen sonuçların Çizelge 6.2 ve 6.3'ten detaylı olarak görülmesi mümkündür. Çizelgelerde 'd', 'o' ve 'y' harfleri, sırasıyla, 'düş$ük$', '$orta$' ve 'yü$ksek$' parametre seviyelerini belirtmek üzere kullanılmıştır. $Epsilon/Delta$ başlığı altında verilen '$d+d$', '$d+o$' vb. ifadeler ise yükleme/boşaltma ve hareket zamanı parametrelerinin birlikte dikkate alındığı gözlemleri vermektedir.

İlgili çizelgelerden görülebileceği üzere, her iki algoritma için de düşük işlem zamanı aralıklarında 3,2 % civarında seyreden ortalama sapmalar, orta boyutlu aralıklarda 3,3 % civarında gözlenmekte, yüksek boyutlu aralıklarda ise 4,6 ila 4,7 % gibi değerler almıştır. Söz konusu işlem zaman aralıklarına bakıldığında aralık genişledikçe sapmanın arttığı görülmektedir. Buna sebep olarak işlem zamanları arasındaki farklar arttıkça sıralama ve çizelgeleme probleminin ve tanımlanan alt sınır değerine ulaşmanın daha zor bir hal almasının görülmesi mümkündür.

Çizelge 6.2. Rassal komşuluk sonuçları

		P = 0-200					P = 0-300					P = 0-500				
Başlangıç Sıcaklığı		**100**	**168**	**196**	**313**	**665**	**100**	**168**	**196**	**313**	**665**	**100**	**168**	**196**	**313**	**665**
Problem Boyutu	**4**	5,13%	5,13%	5,13%	5,13%	5,13%	6,27%	6,27%	6,27%	6,27%	6,27%	8,09%	8,03%	8,03%	8,09%	8,09%
	5	4,73%	4,73%	4,73%	4,73%	4,71%	9,97%	10,00%	9,97%	9,97%	9,97%	8,69%	8,56%	8,56%	8,56%	8,56%
	10	4,26%	4,22%	4,18%	4,15%	4,11%	2,66%	2,56%	2,46%	2,39%	2,32%	3,95%	3,91%	3,84%	3,82%	3,78%
	15	2,96%	2,91%	2,84%	2,79%	2,70%	2,05%	1,97%	1,90%	1,87%	1,80%	4,48%	4,40%	4,35%	4,28%	4,21%
	20	1,85%	1,82%	1,79%	1,73%	1,69%	3,59%	3,55%	3,50%	3,46%	3,38%	3,37%	3,32%	3,26%	3,24%	3,20%
	25	2,32%	2,27%	2,22%	2,17%	2,14%	0,92%	0,87%	0,83%	0,79%	0,77%	3,48%	3,44%	3,40%	3,36%	3,28%
	50	2,24%	2,21%	2,19%	2,16%	2,12%	2,84%	2,82%	2,80%	2,78%	2,75%	3,61%	3,57%	3,54%	3,52%	3,48%
	100	2,83%	2,81%	2,81%	2,80%	2,77%	1,82%	1,81%	1,81%	1,80%	1,79%	4,84%	4,83%	4,82%	4,81%	4,80%
	150	3,10%	3,08%	3,06%	3,05%	3,04%	0,88%	0,87%	0,87%	0,86%	0,85%	2,76%	2,75%	2,75%	2,74%	2,73%
Epsilon	**d**	1,52%	1,49%	1,47%	1,45%	1,41%	1,70%	1,67%	1,61%	1,49%	1,54%	2,60%	2,57%	2,54%	2,52%	2,49%
	o	3,14%	3,12%	3,08%	3,04%	3,00%	3,84%	3,80%	3,77%	3,74%	3,71%	5,11%	5,07%	5,04%	5,01%	4,97%
	y	5,15%	5,12%	5,10%	5,08%	5,05%	4,80%	4,77%	4,75%	4,74%	4,72%	6,71%	6,63%	6,61%	6,61%	6,59%
Delta	**d**	1,50%	1,47%	1,45%	1,43%	1,39%	2,33%	2,29%	2,24%	2,21%	2,17%	2,44%	2,34%	2,31%	2,31%	2,27%
	o	3,24%	3,23%	3,19%	3,17%	3,14%	2,67%	2,65%	2,60%	2,59%	2,55%	3,96%	3,93%	3,90%	3,87%	3,84%
	y	5,05%	5,03%	5,01%	4,97%	4,94%	5,33%	5,31%	5,30%	5,27%	5,24%	8,02%	8,00%	7,98%	7,96%	7,93%
Epsilon / Delta	**d+d**	0,79%	0,76%	0,74%	0,71%	0,69%	1,74%	1,71%	1,61%	1,58%	1,53%	1,93%	1,89%	1,86%	1,84%	1,81%
	d+o	1,55%	1,55%	1,52%	1,51%	1,46%	1,55%	1,52%	1,46%	1,44%	1,40%	2,38%	2,35%	2,33%	2,32%	2,29%
	d+y	2,20%	2,16%	2,14%	2,12%	2,09%	1,81%	1,78%	1,77%	1,73%	1,70%	3,49%	3,47%	3,43%	3,41%	3,36%
	o+d	1,70%	1,67%	1,64%	1,63%	1,57%	2,54%	2,47%	2,44%	2,40%	2,38%	2,69%	2,66%	2,63%	2,60%	2,54%
	o+o	2,94%	2,92%	2,87%	2,82%	2,79%	2,94%	2,92%	2,87%	2,86%	2,81%	3,49%	3,43%	3,39%	3,36%	3,31%
	o+y	4,78%	4,76%	4,74%	4,67%	4,63%	6,03%	6,01%	6,00%	5,97%	5,95%	9,14%	9,12%	9,09%	9,07%	9,06%
	y+d	2,01%	2,00%	1,96%	1,94%	1,90%	2,71%	2,67%	2,66%	2,65%	2,62%	2,69%	2,48%	2,45%	2,48%	2,45%
	y+o	5,25%	5,21%	5,20%	5,18%	5,16%	3,53%	3,50%	3,47%	3,47%	3,45%	6,02%	6,00%	5,97%	5,94%	5,92%
	y+y	8,19%	8,15%	8,14%	8,13%	8,11%	8,16%	8,14%	8,13%	8,11%	8,08%	11,43%	11,42%	11,40%	11,40%	11,38%
Ortalama		3,27%	3,24%	3,22%	3,19%	3,16%	3,45%	3,42%	3,38%	3,36%	3,32%	4,81%	4,76%	4,73%	4,71%	4,68%

düşük	orta	yüksek

Çizelge 6.3. Bilinçli komşuluk sonuçları

		P = 0-200					P = 0-300					P = 0-500				
Başlangıç Sıcaklığı		**100**	**168**	**196**	**313**	**665**	**100**	**168**	**196**	**313**	**665**	**100**	**168**	**196**	**313**	**665**
Problem Boyutu	**4**	5,05%	5,05%	5,05%	5,05%	5,05%	6,42%	6,42%	6,42%	6,42%	6,42%	7,91%	7,91%	7,91%	7,91%	7,91%
	5	4,71%	4,71%	4,71%	4,71%	4,71%	9,83%	9,88%	9,88%	9,88%	9,88%	8,20%	8,13%	8,13%	8,12%	8,12%
	10	4,26%	4,18%	4,14%	4,09%	4,07%	2,70%	2,57%	2,50%	2,40%	2,36%	3,86%	3,83%	3,82%	3,80%	3,75%
	15	3,07%	2,96%	2,90%	2,85%	2,80%	1,96%	1,88%	1,83%	1,78%	1,74%	4,23%	4,18%	4,15%	4,09%	4,04%
	20	1,87%	1,85%	1,81%	1,77%	1,71%	3,55%	3,51%	3,48%	3,44%	3,35%	3,39%	3,35%	3,32%	3,26%	3,19%
	25	2,38%	2,30%	2,25%	2,22%	2,16%	0,85%	0,80%	0,78%	0,76%	0,71%	3,53%	3,48%	3,45%	3,39%	3,33%
	50	2,23%	2,21%	2,18%	2,16%	2,13%	2,87%	2,85%	2,83%	2,81%	2,80%	3,62%	3,58%	3,56%	3,54%	3,51%
	100	2,83%	2,80%	2,78%	2,77%	2,75%	1,83%	1,81%	1,80%	1,77%	1,76%	4,88%	4,86%	4,85%	4,85%	4,83%
	150	3,08%	3,08%	3,06%	3,05%	3,03%	0,87%	0,86%	0,86%	0,85%	0,83%	2,78%	2,77%	2,76%	2,75%	2,74%
Epsilon	**d**	1,52%	1,48%	1,46%	1,43%	1,41%	1,66%	1,63%	1,60%	1,52%	1,53%	2,61%	2,55%	2,54%	2,51%	2,48%
	o	3,16%	3,12%	3,08%	3,05%	3,01%	3,86%	3,81%	3,79%	3,76%	3,74%	5,07%	5,05%	5,03%	5,00%	4,97%
	y	5,14%	5,11%	5,09%	5,08%	5,05%	4,78%	4,75%	4,74%	4,71%	4,69%	6,45%	6,43%	6,41%	6,39%	6,36%
Delta	**d**	1,50%	1,46%	1,44%	1,42%	1,39%	2,35%	2,31%	2,28%	2,23%	2,20%	2,37%	2,32%	2,30%	2,27%	2,22%
	o	3,27%	3,22%	3,19%	3,17%	3,14%	2,63%	2,59%	2,56%	2,55%	2,52%	3,97%	3,94%	3,93%	3,90%	3,88%
	y	5,06%	5,03%	5,00%	4,97%	4,94%	5,32%	5,30%	5,29%	5,26%	5,23%	7,79%	7,77%	7,76%	7,73%	7,70%
Epsilon / Delta	**d+d**	0,80%	0,76%	0,74%	0,72%	0,69%	1,75%	1,73%	1,67%	1,62%	1,57%	2,00%	1,91%	1,88%	1,85%	1,80%
	d+o	1,57%	1,50%	1,48%	1,46%	1,44%	1,49%	1,44%	1,42%	1,40%	1,39%	2,35%	2,33%	2,32%	2,30%	2,28%
	d+y	2,21%	2,17%	2,15%	2,12%	2,11%	1,74%	1,73%	1,71%	1,69%	1,63%	3,47%	3,42%	3,42%	3,39%	3,35%
	o+d	1,69%	1,65%	1,63%	1,61%	1,56%	2,57%	2,50%	2,48%	2,44%	2,42%	2,56%	2,55%	2,53%	2,51%	2,46%
	o+o	2,98%	2,94%	2,90%	2,87%	2,82%	2,93%	2,89%	2,86%	2,84%	2,79%	3,49%	3,46%	3,43%	3,40%	3,39%
	o+y	4,80%	4,76%	4,72%	4,67%	4,64%	6,08%	6,06%	6,03%	6,00%	5,99%	9,16%	9,14%	9,13%	9,10%	9,06%
	y+d	2,02%	1,98%	1,95%	1,93%	1,92%	2,73%	2,69%	2,68%	2,63%	2,62%	2,54%	2,51%	2,48%	2,45%	2,41%
	y+o	5,25%	5,21%	5,18%	5,18%	5,16%	3,46%	3,43%	3,41%	3,40%	3,38%	6,07%	6,03%	6,03%	6,00%	5,98%
	y+y	8,17%	8,15%	8,14%	8,12%	8,09%	8,15%	8,13%	8,12%	8,09%	8,07%	10,74%	10,74%	10,73%	10,72%	10,70%
Ortalama		3,28%	3,24%	3,21%	3,19%	3,16%	3,43%	3,40%	3,38%	3,35%	3,32%	4,71%	4,68%	4,66%	4,63%	4,60%

düşük	orta	yüksek

Probleme yönelik belirtilen parametre değerlerine göre karşılaştırmalar yapılarak problem boyutu, yükleme/boşaltma zamanı ve hareket zamanının farklı seviyelerinde gözlemler yapılmıştır. Bu gözlemler için Çizelge 6.4 ve 6.5'ten yararlanılması mümkündür.

Parametreler bazında yapılan karşılaştırmalardan, problem sonuçlarında yüksek öneme sahip olan iki parametrenin yükleme/boşaltma ($epsilon$) ve hareket ($delta$) zamanları olduğu görülmektedir. Bu iki değerin toplamının yüksek olduğu durumda her iki algoritmanın da ilgili alt sınır değeri üzerinden elde ettiği sapmaların daha yüksek olduğu görülmektedir. Söz konusu iki değere ayrı ayrı bakılacak olursa, hareket zamanının daha fazla önemi olduğu görülür ki sistemdeki yükleme/boşaltma sayıları algoritmadan bağımsızken oluşturulan algoritmalarda amaçlanan robot hareket dizisinin belirlenmesiyle hareket zamanından tasarruf sağlanmaya çalışıldığından bu sonuç oldukça beklenebilirdir.

Geliştirilen komşuluk yapılarına göre elde edilen sonuçlara bakacak olursak aradaki fark çok yüksek olmamakla birlikte, özellikle yüksek işlem zamanı aralıklarına, yüksek yükleme/boşaltma ve yüksek hareket zamanlarına sahip sistemlerin her birinde ortalama 0,08 % iyileşme sağladığı görülmüştür. Yüksek işlem zamanı aralıklarında bu sapmadaki iyileşmenin daha fazla olmasının altında, ilgili problemlerdeki işlem zamanları arasındaki farkların daha fazla olup Algoritma 2 ile tanımlanan daha bilinçli diyebileceğimiz komşuluk yapısı sayesinde bu farkların daha fazla dikkate alınıyor olmasının yattığını söylemek mümkündür.

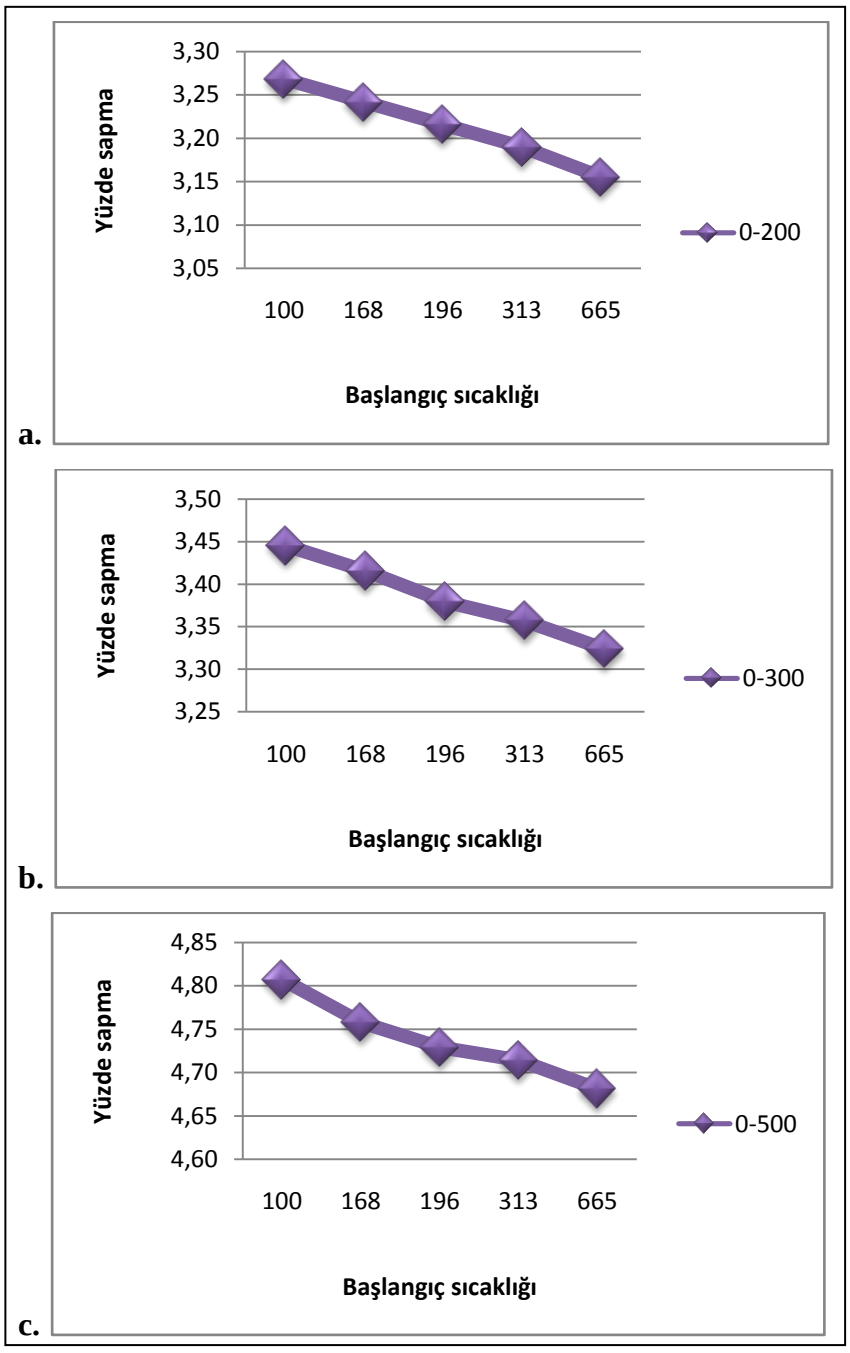
Çizelge 6.4. Rassal komşuluk için parametre bazında sonuçlar

Başlangıç Sıcaklığı		100	168	196	313	665
İşlem Zamanı	Düşük	3,27%	3,24%	3,22%	3,19%	3,16%
	Orta	3,45%	3,42%	3,38%	3,36%	3,32%
	Yüksek	4,81%	4,76%	4,73%	4,71%	4,68%
Epsilon	Düşük	1,94%	1,91%	1,87%	1,82%	1,81%
	Orta	4,03%	4,00%	3,96%	3,93%	3,89%
	Yüksek	5,55%	5,51%	5,49%	5,48%	5,45%
Delta	Düşük	2,09%	2,03%	2,00%	1,98%	1,94%
	Orta	3,29%	3,27%	3,23%	3,21%	3,18%
	Yüksek	6,14%	6,11%	6,09%	6,07%	6,04%
Epsilon&Delta	Düşük	1,87%	1,84%	1,80%	1,78%	1,74%
	Orta	2,70%	2,65%	2,62%	2,60%	2,56%
	Yüksek	6,95%	6,92%	6,90%	6,88%	6,86%

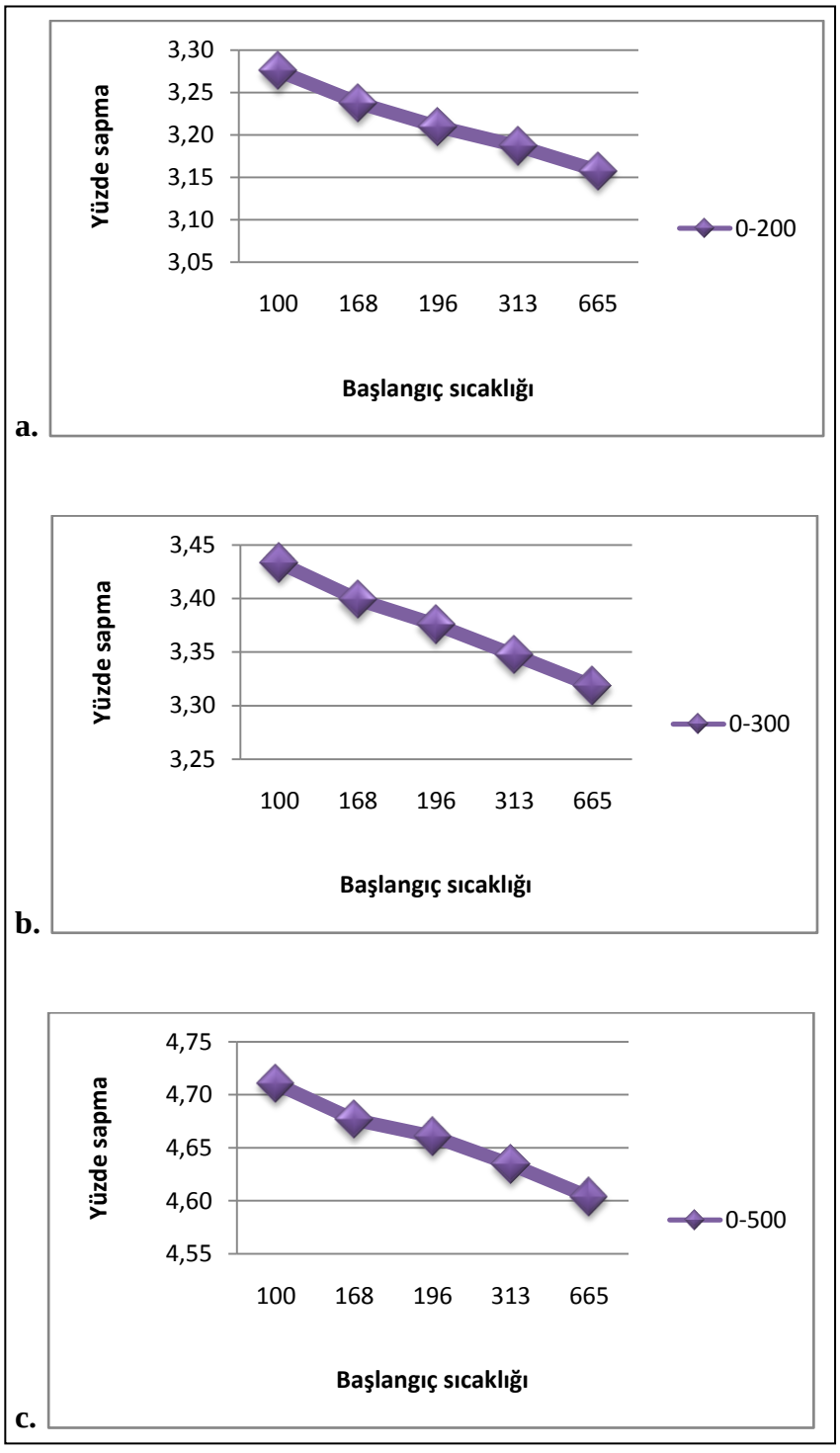
Çizelge 6.5. Bilinçli komşuluk için parametre bazında sonuçlar

Başlangıç Sıcaklığı		100	168	196	313	665
İşlem Zamanı	Düşük	3,28%	3,24%	3,21%	3,19%	3,16%
	Orta	3,43%	3,40%	3,38%	3,35%	3,32%
	Yüksek	4,71%	4,68%	4,66%	4,63%	4,60%
Epsilon	Düşük	1,93%	1,89%	1,86%	1,82%	1,81%
	Orta	4,03%	3,99%	3,97%	3,94%	3,90%
	Yüksek	5,46%	5,43%	5,41%	5,39%	5,37%
Delta	Düşük	2,07%	2,03%	2,00%	1,97%	1,94%
	Orta	3,29%	3,25%	3,23%	3,20%	3,18%
	Yüksek	6,06%	6,03%	6,02%	5,99%	5,96%
Epsilon&Delta	Düşük	1,86%	1,82%	1,79%	1,77%	1,73%
	Orta	2,68%	2,64%	2,62%	2,59%	2,56%
	Yüksek	6,88%	6,85%	6,83%	6,81%	6,79%

Denenen farklı başlangıç sıcaklıkları için elde edilen sonuçlar karşılaştırıldığında, bekleneceği üzere, yüksek sıcaklık derecelerinde elde edilen sapmaların daha düşük olduğu görülmüştür. Her iki komşuluk yapısına yönelik grafikler Şekil 6.5 ve 6.6'da verilmiştir.



Şekil 6.5. Başlangıç sıcaklığı değişimlerine göre rassal komşuluğa ait sonuçlar



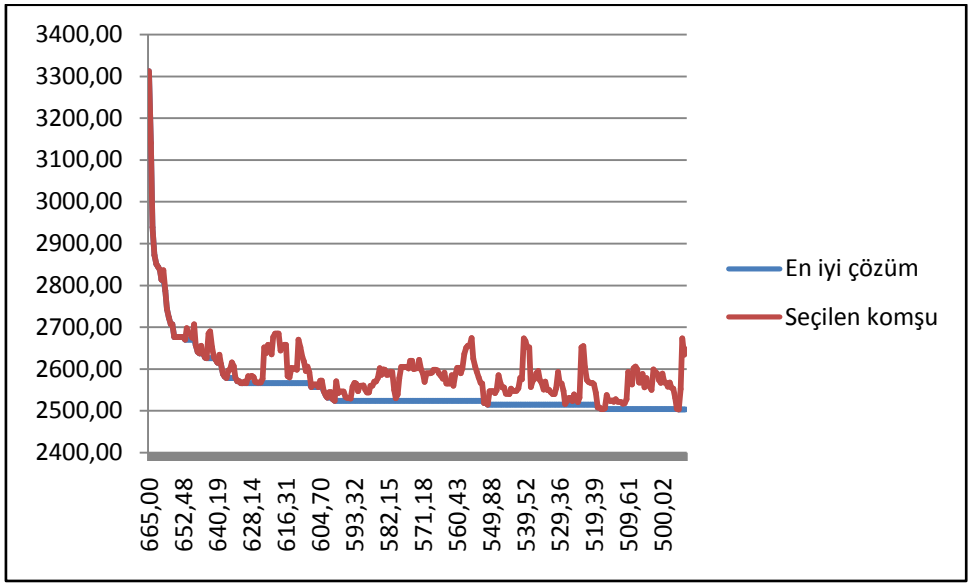
Şekil 6.6. Başlangıç sıcaklığı değişimlerine göre bilinçli komşuluğa ait sonuçlar

Şekil 6.5 ve Şekil 6.6'daki sonuçlara bakıldığında yüzde sapmalardaki farkın oldukça düşük olduğu görülmüştür. Algoritmanın çalışması süresince elde ettiği çözümlerdeki iyileşmelerin sıcaklık bazında gözlenmesi amacıyla, eldeki örnek veri kümeleri arasından rassal olarak seçilen orta boyutlu bir problem, dikkate alınmış en yüksek başlangıç sıcaklık değeri olan $T_i=665$ değeri ile çözülmüş ve her bir iterasyonda bulunan en iyi çözümler ile

en iyi komşulara ait sonuçlar kaydedilmiştir. Bu deneme sonucunda EK-5'te verilen grafik elde edilmiştir. İlgili grafikten görülebileceği üzere ilk sıcaklık değerlerindeki amaç fonksiyonu değişimi oldukça yüksek olmakta, daha sonraki değerlerde ise önemli farklılıklar görülmemektedir.

Sıcaklık değeri düştükçe gözlenen bir diğer değişiklik ise, TB'nin mantığına uygun olarak, bulunan komşu çözümlerde kötü sonuçlara daha az izin verilmesi olmuştur. Örneğin, sıcaklık değerinin 140,47 ila 125,71 arasında olduğu bölgede 2750'ye yakın çözümler komşu çözüm olarak kabul edilirken, sonraki sıcaklıklarda en büyük değere sahip komşu çözümlerin bile çevrim değerleri 2700'ün altında seyretmiştir.

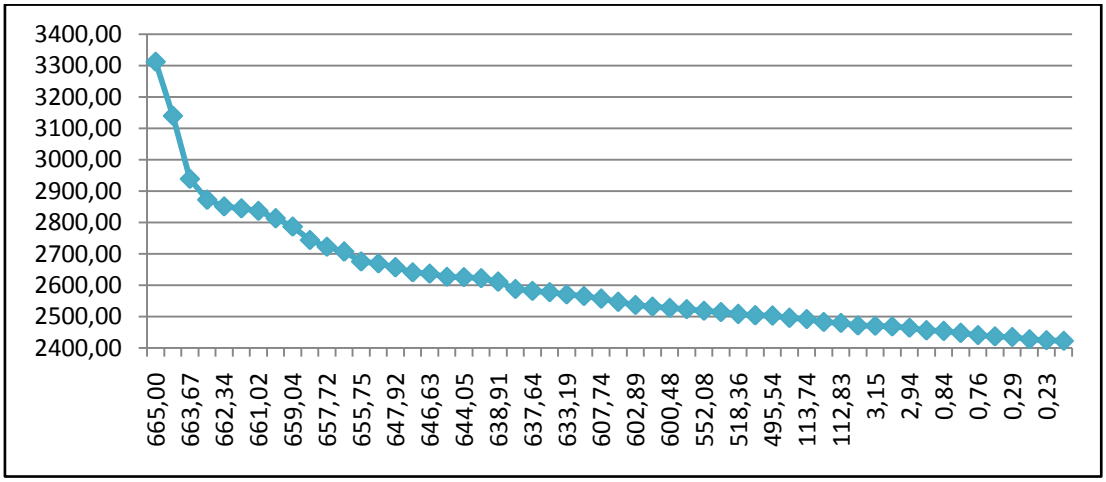
Şekil 6.7'de değişim hızının yüksek olduğu ilk sıcaklıklardaki değişimin daha iyi görülebilmesi için EK-5'teki grafiğin ilk bölümü büyütülerek alınmıştır. Şekilde algoritma süresince bulunan en iyi çözümlere bakıldığında, ilk sıcaklık değerlerinde çevrim zamanı değerlerinde yüksek değişim gözlenirken ilerleyen aşamalarda değişim oranının azaldığı görülmektedir.



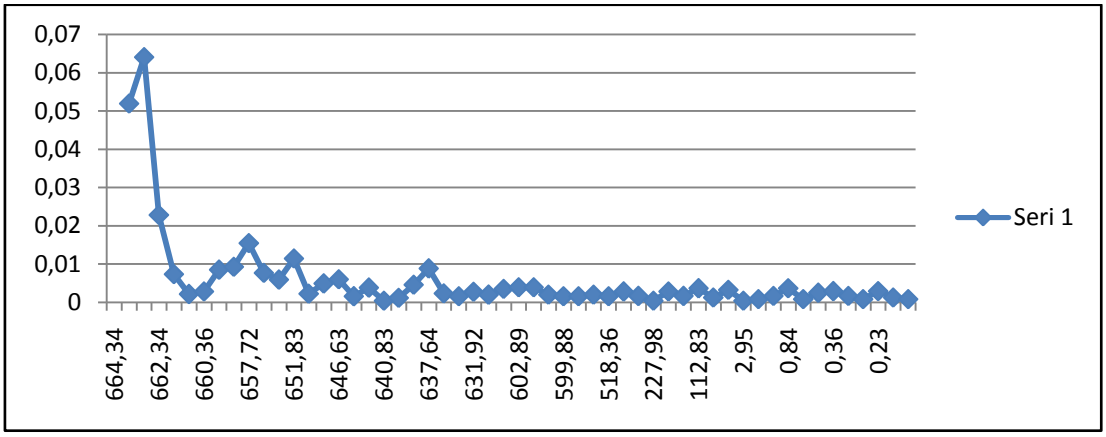
Şekil 6.7. İterasyon bazında en iyi çevrim zamanı ve en iyi komşu çözümler

Şekil 6.8 ve 6.9'da, bulunan en iyi çözümlere göre iterasyon bazındaki değişimler gösterilmiştir. Şekil 6.7'deki sonuçlarla uyumlu olarak çevrim zamanı değişiminin ilk

sıcaklık değerlerinde oldukça yüksek olduğu, sıcaklık arttıkça daha yavaş ve daha düşük oranlı değişim gerçekleştiği görülmektedir.



Şekil 6.8. İterasyon bazında çevrim zamanı değişim noktaları



Şekil 6.9. İterasyon bazında çevrim zamanı değişim oranları

7. SONUÇ

Bu çalışmada odak noktası çoklu parça tipinin tekrarlı olarak üretildiği ve makineler arası taşımaların robot yardımıyla gerçekleştirildiği, sabit hareket zamanlı esnek hibrit akış tipi hücrelerde gözlenen çizelgeleme problemidir. Ele alınan sistemde hibrit akış tipi özelliği olarak, birden fazla üretim aşaması ve bu aşamalardan en az birinde birden fazla paralel makine bulunmaktadır. Problemde çoklu parça tipinin dikkate alınması nedeniyle, En Az Parça Kümesi (EAPK) olarak adlandırılan, toplam üretim miktarı ile aynı oranlarda parça içeren kümelerin üretimi gerçekleştirilmektedir. Üzerinde çalışılan amaç değeri bir EAPK'nin tekrarlı olarak üretilmesi sırasında gereken ortalama zamanın minimizasyonudur. Bu problemdeki zorluk üç yönlü olmaktadır: (a) parça sıralamasının belirlenmesi, (b) parçaların makinelere atanması, (c) robot hareket dizisinin belirlenmesi. Sistemde varlığı kabul edilen esneklik özelliği sonucu, bazı parçaların bazı aşamalarda işlem görmemesine de izin verilmektedir.

Aranılan çözüm, taşınan/yüklenen/alınan parçayı gösterip ilgili makineleri de belirterek, robot hareketlerini tam olarak tanımlayacak şekilde olmaktadır. Problem öncelikle, uzaklık matrisinin parametre değerlerinin yanı sıra karar değişkenlerinden oluştuğu, Gezgin Satıcı Probleminin özel bir şekli olarak modellenmiştir. Bu formulasyon, klasik GSP'den çok daha genel bir yapıda olup küçük boyutlu problemlerin çözümü için bile oldukça zorlanmaktadır. Bu durum sonucunda, sezgisel yaklaşımlar üzerinde durulması düşünülmüş ve literatürde yaygın olarak kullanılan TB yaklaşımı tabanlı bir sezgisel yöntem geliştirilmiştir. Elde edilen sonuçların incelenebilmesi için probleme yönelik olarak bir alt sınır değeri geliştirilmiştir. Rassal olarak üretilen problemler, önerilen algoritma ile çözülerek bulunan sonuçlar probleme özgü alt sınır değerine yakınlıklarına göre değerlendirilmiş ve yapılan bu değerlendirmeler sonucu, algoritmanın alt sınır değerine oldukça yakın sonuçlar verdiği görülmüştür. Sözkonusu denemeler üzerinden parametre analizleri de yapılmış, algoritmanın hemen hemen tüm parametre değerleri için iyi sonuçlar verdiği gözlenmiştir.

Bildiğimiz kadarıyla, bu çalışma hibrit esnek akış tipi hücrelerde tek robot kullanımın ve çoklu parça tipi üretiminin ele alındığı ilk çalışmadır. Çalışmanın ilerletilmesi ve gelecek araştırmalar için imkan bulunmaktadır. Olası bir çalışma konusu olarak her bir aşamada en

az 2 olmak üzere, tüm aşamalarda m-makinenin bulunduğu hücrelerin gözlenmesi alınabilir. Böyle bir varsayım problem büyüklüğünü oldukça artıracak, boyut arttıkça probleme uygun çözüm bulunması da güçleşecektir. Diğer bir çalışma konusu, makineler arası teknolojik farkların olduğu durumların dikkate alınması olabilir. Aşamalarda özdeş olmayan makinelerin bulunmasıyla birlikte, herhangi bir makinenin üretilecek bir parçaya yönelik olarak ilgili aşamada yapılması gereken tüm işlemleri gerçekleştirmeye yeterli olmayabileceği düşünülebilecektir. Bunun sonucu olarak bazı işlemlere ihtiyaç duyan parçaların belirli makinelere atanması kısıtı gerekebilecek olup bu tanımların da probleme katılmasıyla problem karmaşıklığı oldukça artacaktır. Bir başka konu, birden fazla robotun kullanılması ile elde edilebilir. Bu durumda sistemdeki olası robot hareketi sayısı artacak olmakla birlikte sistemdeki boş zamanların oldukça düşeceği öngörülebilir.

KAYNAKLAR

Abdekhodaee, A.H. and Wirth, A. (2002). Scheduling parallel machines with a single server: some solvable cases and heuristics. ***Computers & Operations Research***, 29, 295-315.

Abdekhodaee, A.H., Wirth, A., and Gan, H. S. (2004). Equal processing and equal setup time cases of scheduling parallel machines with a single server. ***Computers & Operations Research***, 31, 1867-1889.

Abdekhodaee, A.H., Wirth, A., and Gan, H. S. (2006). Scheduling two parallel machines with a single server: the general case. ***Computers & Operations Research***, 33, 994-1009.

Abyaneh, S.H. and Zandieh, M. (2012). Bi-objective hybrid flow shop scheduling with sequence-dependent setup times and limited buffers. ***International Journal of Advanced Manufacturing Technology***, 58, 309-325.

Aneja, Y.P. and Kamoun, H. (1999). Scheduling of parts and robot activities in two-machine robotic cell. ***Computers & Operations Research***, 26, 297-312.

Arthanary, T.S. and Ramaswamy, K.G. (1971). An extension of two machine sequencing problem. ***OPSEARCH: The Journal of Operational Research Society of India***, 8(4), 10-22.

Behnamian, J., Zandieh, M., and Fatemi Ghomi, S.M.T. (2009). Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm. ***Expert Systems with Applications***, 36, 9637-9644.

Behnamian, J. and Fatemi Ghomi, S.M.T. (2011). Hybrid flowshop scheduling with machine and resource-dependent processing times. ***Applied Mathematical Modelling***, 35, 1107-1123.

Bolat, A., Al-Harkan, I., and Al-Harbi, B. (2005). Flow-shop scheduling for three serial stations with the last two duplicate. ***Computers and Operations Research***, 32(3), 647-667.

Botta-Genoulaz, V. (2000). Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness. ***International Journal of Production Economics***, 64, 101-111.

Bozejko, W., Pempera, J., and Smutnicki, C. (2013). Parallel tabu search algorithm for the hybrid flow shop problem. ***Computers and Industrial Engineering***, 65, 466-474.

Brauner, N. (2008). Identical part production in cyclic robotic cells: Concepts, overview and open questions. ***Discrete Applied Mathematics***, 156, 2480-2492.

Brauner, N. and Finke, G. (1999). On the conjecture in robotic cells: new simplified prof for the three-machine case. ***INFOR***, 37(1), 20-36.

Brauner, N. and Finke, G. (2001). On cycles and permutations in robotic cells. ***Mathematical and Computer Modeling***, 34, 565-591.

Browne, J., Harhen, J., and Shivnan, J. (1996). *Production Management Systems*. New York: Addison-Wesley.

Carpov, S., Carlier, J., Nace, D., and Renaud, S. (2012). Two-stage hybrid flow shop with precedence constraints and parallel machines at second stage. ***Computers and Operations Research***, 39, 736-745.

Chen, C.L., Usher, J.M., and Palanimuthu, N. (1998). A tabu search based heuristic for a flexible flow line with minimum flow time criterion. ***International Journal of Industrial Engineering-Theory Applications and Practice***, 5(2), 157-168.

Crama, Y. and Van de Klundert, J. (1997). Cyclic scheduling of identical parts in a robotic cell. ***Operations Research***, 45(6), 952-965.

Crama, Y., and Van de Klundert, J. (1999). Cyclic scheduling in 3-machine robotic flow shops. ***Journal of Scheduling***, 4, 35-54.

Crama, Y., Kats, V., Van de Klundert, J., and Levner, E. (2000). Cyclic scheduling in robotic flowshops. ***Annals of Operations Research***, 96, 97-124.

Dawande, M., Geismar, H.N., Sethi, S.P., and Sriskandarajah, C. (2005). Sequencing and scheduling in robotic cells: Recent developments. ***Journal of Scheduling***, 8(5), 387-426.

Dawande, M., Sriskandarajah, C., and Sethi, S.P. (2002). On throughput maximization in constant travel-time robotic cells. ***Manufacturing and Service Operations Management***, 4(4), 296-312.

Dessouky, M.M., Dessouky, M.I., and Verma, S.K. (1998). Flowshop scheduling with identical jobs and uniform parallel machines. ***European Journal of Operational Research***, 109, 620-631.

Elmi, A. and Topaloglu, S. (2013). A scheduling problem in blocking hybrid flow shop robotic cells with multiple robots. ***Computers and Operations Research***, 40, 2543-2555.

Geismar, H.N., Dawande, M., and Sriskandarajah, C. (2004). Robotic Cells With Parallel Machines: Throughput Maximization in Constant Travel-TimeCells. ***Journal of Scheduling***, 7, 375-395.

Gilmore, P.C. and Gomory, R.E. (1964). Sequencing in one state-variable machine:a solvable case of the travelling salesman problem. ***Operations Research***, 12, 655-679.

Guirchoun, S., Martineau, P., and Billaut, J.C. (2005). Total completion time minimization in a computer system with a server and two parallel processors. ***Computers and Operations Research***, 32(3), 599-611.

Gultekin, H., Akturk, M.S., and Karasan, O.E. (2006). Robotic cell scheduling with tooling constraints. ***European Journal of Operational Research***, 174, 777-796.

Gultekin, H., Karasan, O.E., and Akturk, M.S. (2009). Pure cycles in flexible robotic cells. ***Computers & Operations Research***, 36(2), 329-344.

Gupta, J.N.D. (1988). Two-stage, hybrid flowshop scheduling problem. ***Journal of the Operational Research Society***, 39(4), 359-364.

Gupta, J.N.D. and Tunc, E.A. (1991). Schedules for a two-stage hybrid flowshop with parallel machines at the second stage. ***International Journal of Production Research***, 29(7), 1489-1502.

Gupta, J.N.D. and Tunc, E.A. (1994). Scheduling a two-stage hybrid flowshop with separable setup and removal times. ***European Journal of Operational Research***, 77(3), 415-428.

Gupta, J.N.D., Hariri, A.M.A., and Potts, C.N. (1997). Scheduling a two-stage hybrid flow shop with parallel machines at the first stage. ***Annals of Operations Research***, 69, 171-191.

Hall, N.G., Kamoun, H., and Sriskandarajah, C. (1997a). Scheduling in robotic cells: classification, two and three machine cells. ***Operations Research***, 45(3), 421-439.

Hall, N.G., Kamoun, H., and Sriskandarajah, C. (1997b). Scheduling in robotic cells: Complexity and steady state analysis. ***European Journal of Operational Research***, 109(1), 43-65.

Hall, N.G., Potts, C.N., and Sriskandarajah, C. (2000). Parallel machine scheduling with a common server. ***Discrete Applied Mathematics***, 102, 223-243.

Haouari, M., Hidri, L., and Gharbi, A. (2006). Optimal scheduling of a two-stage hybrid flow shop. ***Mathematical Methods of Operations Research***, 64, 107-124.

Hooda, N. and Dhingra, A.K. (2011). Flow shop scheduling using simulated annealing: A review. ***International Journal of Applied Engineering Research***, 2(1), 234-249.

Huang, W. and Li, S. (1998). A two-stage hybrid flowshop with uniform machines and setup times. ***Mathematical and Computer Modelling***, 27(2), 27-45.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., and Werner, F. (2008). Algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. ***International Journal of Advanced Manufacturing Technology***, 37(3-4), 354-370.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., and Werner, F. (2009). A comparison of scheduling algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. ***Computers & Operations Research***, 36, 358-378.

Karaoglan, I., Altiparmak, F., Kara, I. and Dengiz, B. (2011). A branch and cut algorithm for the location-routing problem with simultaneous pickup and delivery. ***European Journal of Operational Research***, 211, 318-332.

Kamalabadi, I.N., Gholami, S., and Mirzaei, A.H. (2007). *Considering a cyclic multiple-part type three-machine robotic cell problem*. ***IEEM, IEEE International Conference***, 704-708.

Kamoun, H., Hall N.G., and Sriskandarajah, C. (1999). Scheduling in robotic cells: heuristics and cell design. ***Operations Research,*** 47(6), 821-835.

Kats, V. and Levner, E. (1997). A strongly polynomial algorithm for no-wait cyclic robotic flowshop scheduling. ***Operations Research Letters***, 21, 171-179.

Kats, V. and Levner, E. (2009). A polynomial algorithm for 2-cyclic robotic scheduling: A non-Euclidean case. ***Discrete Applied Mathematics***, 157(2), 339-355.

Khalili, M. (2014). A multi-objective electromagnetism algorithm for a bi-objective hybrid no-wait flowshop scheduling problem. ***International Journal of Advanced Manufacturing Technology***, 70, 1591-1601.

Kim, D.W., Kim, K.H., Jang, W., and Chen F.F. (2002). Unrelated parallel machine scheduling with setup times using simulated annealing. ***Robotics and Computer Integrated Manufacturing***, 18, 223-231.

Kirkpatrick, S., Gelatt Jr., C.D., and Vecchi M.P. (1983). Optimization by Simulated Annealing", ***Science***, 220, 671-680.

Kise, H., Shioyama, T., and Ibaraki, T. (1993). Automated two machine flowshop scheduling: a solvable case. ***IIE Transactions***, 23, 80-87.

Koulamas, C.P. (1996). Scheduling two parallel semiautomatic machines to minimize machine interference. ***Computers and Operations Research***, 23(10), 945-956.

Koulamas, C.P. and Smith, M.L. (1988). Look-ahead scheduling for minimizing machine interference. ***International Journal of Production Research***, 26, 1523-1533.

Kravchenko, S.A. and Werner, F. (1997). Parallel Machine Scheduling Problems with a single server. ***Mathematical and Computer Modelling***, 26(12), 1-11.

Kurz, M.E. and Askin, R.G. (2003). Comparing scheduling rules for flexible flow lines. ***International Journal of Production Economics***, 85, 371-388.

Kurz, M.E. and Askin, R.G. (2004). Scheduling flexible flow lines with sequence-dependent setup times. ***European Journal of Operational Research***, 159, 66-82.

Lee, C.Y., Lei, L., and Pinedo, M. (1997). Current trends in deterministic scheduling. ***Annals of Operations Research***, 70, 1-41.

Lee, G.-C. and Kim, Y.D. (2004). A branch-and-bound algorithm for a two-stage hybrid flowshop scheduling problem minimizing total tardiness. ***International Journal of Production Research***, 42(22), 4731-4743.

Lee, Z.J., Lin, A.W., and Ying K.C. (2010). Scheduling jobs on dynamic parallel machines with sequence-dependent setup times. ***International Journal of Advanced Manufacturing Technology***, 47, 773-781.

Levner, E., Kats, V., and Levit, V.E. (1997). An improved algorithm for cyclic flowshop scheduling in a robotic cell. ***European Journal of Operational Research***, 97, 500-508.

Li, L., Wang, L., and Huo, J., "Hybrid flowshop scheduling with setup times for cold treating process in Baoshan Iron & Steel Complex", ***Proceedings of the Logistics Systems and Intelligent Management, 2010 International Conference***, Jan. 9-10, 2010, Harbin, 357-363 (2010).

Lin, H.-T. and Liao, C.J. (2003). A case study in a two-stage hybrid flow shop with setup time and dedicated machines. ***International Journal of Production Economics***, 86(2), 133-143.

Linn, R. and Zhang, W. (1999). Hybrid flow shop scheduling: a survey. ***Computers and Industrial Engineering***, 37(1-2), 57-61.

Logendran, R. and Sriskandarajah, C. (1996). Sequencing of robot activities and parts in two-machine robotic cells. ***International Journal of Production Research***, 34, 34-47.

Low, C., (2005). Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. ***Computers & Operations Research***, 32, 2013-2025.

Low, C., Yeh, J.Y., and Huang, K.I. (2004). A robust simulated annealing heuristic for flow shop scheduling problems. ***Interantional Journal of Advanced Manufacturing Technology***, 23, 762-767.

Marichelvam, M.K., Parabaharan, T., and Yang, X.S. (2014). Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan. ***Applied Soft Computing***, 19, 93-101.

Metropolis, N., Rosenbluth, A., and Resenbluth, M. (1953). Equation of state calculations by fast computing machines. ***Journal of Chemical Physics***, 21, 1087–1092.

Miller, C., Tucker, A., and Zemlin, R. (1960). Integer programming formulation of traveling salesman problems. ***Journal of the ACM***, 7(4), 326-329.

Mirsanei, H.S., Zandieh, M., Moayed, M.J., and Khabbazi, M.R. (2011). A simulated annealing algorithm approach to hybrid flow shop scheduling with sequence-dependent setup times. ***Journal of Intelligent Manufacturing***, 22, 965-978.

Mittal, B.S. and Bagga, P.C. (1973). Two machine sequencing problem with parallel machines. ***OPSEARCH: Journal of the Operational Society of India***, 10, 50-61.

Morton, T.E. and Pentico, D.W. (1993). *Heuristic scheduling systems: with applications to production systems and project management*. New York:Wiley.

Naderi, B., Zandieh, M., Khaleghi Ghoshe Balagh, A., and Roshanaei, V. (2009a). An improved simulated annealing for hybrid flowshops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness. ***Expert Systems with Applications***, 36, 9625-9633.

Naderi, B., Zandieh, M., and Roshanaei, V. (2009b). Scheduling hybrid flowshops with sequence dependent setup times to minimize makespan and maximum tardiness. ***The International Journal of Advanced Manufacturing Technology***, 41, 1186-1198).

Nowicki, E., Smutnicki, C. (1998). The flow shop with parallel machines: A tabu search approach. ***European Journal of Operational Research***, 106, 226-253.

Oguz C., Zinder Y., Do V.H., Janiak A., and Linchtenstein M. (2004). Hybrid flow shop scheduling problems with multiprocessor task systems. ***European Journal of Operational Research***, 152,115-131.

Pan, Q., Wang, L., Li, J., Duan, J. (2014). A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespan minimisation. ***Omega***, 45, 42-56.

Quadt, D. and Kuhn, D. (2007). A taxonomy of flexible flow line scheduling procedures", ***European Journal of Operational Research***, 178(3), 686-698.

Rao, T.B.K. (1970). Sequencing in the order a, b, with multiplicity of machines for a single operation. ***OPSEARCH: Journal of the Operational Society of India***, 7, 135-144.

Ruiz, R. and Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. ***European Journal of Operational Research***, 169, 781-800.

Ruiz, R., Serifoğlu, F.S. and Urlings, T. (2008). Modeling realistic hybrid flexible flowshop scheduling problems. ***Computers & Operations Research***, 35, 1151-1175.

Ruiz, R. and Vázquez-Rodríguez, J.A. (2010). The hybrid flow shop scheduling problem. ***European Journal of Operational Research***, 205, 1-18.

Salvador, M.S. (1973). A solution to a special case of flow shop scheduling problems. S.E. Elmaghraby (Ed.). *Symposium of the Theory of Scheduling and Applications*, Springer-Verlag, pp. 83-91.

Sawik, T. (2012). Batch versus cyclic scheduling of flexible flow shops by mixed-integer programming. ***International Journal of Production Research***, 50(18), 5017-5034.

Sethi, S.P., Sriskandarajah, C., Sorger, G. Blazewicz, J. and Kubiak W. (1992). Sequencing of parts and robot moves in a robotic cell. ***International Journal of Flexible Manufacturing Systems***, 4, 331-358.

Singh, M.,R. and Mahapatra, S.,S. (2012). A swarm optimization approach for flexible flow shop scheduling with multiprocessor tasks. ***International Journal of Advanced Manufacturing Technology***, 62, 267-277.

Sriskandarajah, C. (1993). Performance of scheduling algorithms for no-wait flowshops with parallel machines. ***European Journal of Operational Research***, 70, 365-378.

Sriskandarajah, C., Hall, N.G., and Kamoun, H. (1998). Scheduling large robotic cells without buffers. ***Annals of Operations Research***, 76, 287-321.

Stern, H. I. and Vitner, G. (1990). Scheduling Parts in a Combined Production-Transportation Work Cell. ***The Journal of the Operational Research Society,*** 41(7), 625-632.

Vignier, A., Billaut, J.C. and Proust, C. (1999). Les problFmes d'ordonnancement de type flow-shop hybride. ***Ttat de l'art, RAIRO Recherche OpTrationnelle***, 33(2), 117-183.

Wang, H. (2005). Flexible flow shop scheduling: optimum, heuristics and artificial intelligence solutions. ***Expert Systems***, 22(2), 78-85.

Wittrock, R.J. (1985). Scheduling algorithms for flexible flow lines. ***IBM Journal of Research and Development***, 29(4), 401-412.

Wittrock, R.J. (1988). An adaptable scheduling algorithm for flexible flow lines. ***Operations Research***, 36(3), 445-453.

Zandieh, M. and Karimi, N. (2011). An adaptive multi-population genetic algorithm to solve the multi-objective group scheduling problem in hybrid flexible flowshop with sequence-dependent setup times. ***Journal of Intelligent Manufacturing***, 22, 979-989.

Zegordi, A.H., Itoh, K., and Enkawa T. (1995). Minimizing makespan for flow shop scheduling by combining simulated annealing with sequencing knowledge. ***European Journal of Operational Research***, 85, 515-531.

Zhang, R. and Wu, C. (2010). A hybrid immune simulated annealing algorithm for the job shop scheduling problem. ***Applied Soft Computing***, 10, 79-89.

EKLER

EK-1. Matematiksel modele ait GAMS/CPLEX kodu

```
option solprint=off;
options iterlim=50000000, reslim=360000000000;
option optcr = 0.0;

Set i /1*3/;
Set k /1*8/;

alias(k,kk,l);
alias(i,ii,j);
option seed=123221;

parameter proca(i);
parameter procb(i);
proca('1')=17;
proca('2')=20;
proca('3')=0;
procb('1')=30;
procb('2')=34;
procb('3')=27;

parameter delta;
parameter epsilon;
epsilon=1;
delta=2;

parameter cost(i,k,j,l);

parameter possible(i,k,j,l);
possible(i,k,j,l)=1$(cost(i,k,j,l)>0);

parameter notpossible(i,k,j,l);
notpossible(i,k,j,l)=1-possible(i,k,j,l);

option cost:3:0:1;
display cost;

option possible:3:0:1;
display possible;

option notpossible:3:0:1;
display notpossible;

parameter wait(i,k,j,l);

option wait:3:0:1;
display wait;
```

EK-1.(devam) Matematiksel modele ait GAMS/CPLEX kodu

```
parameter bigm;
bigm=sum(i,proca(i))+sum(i,procb(i))+sum(i,1)*(4*epsilon+3*delta);
bigm=bigm*100;

positive variables c(i,k),p(i,k),cmax,w(i,k),start(i,k),m2val(i,k),m3val(i,k),m4val(i,k);
binary variables y(i,k,j,l),normal(i,k),turn(k,l);

con1.. sum((i,k,j,l)$(not possible(i,k,j,l)),y(i,k,j,l)) =e= 0;
con2(i,k).. sum((j,l),y(j,l,i,k)) =e= sum((j,l),y(i,k,j,l));
con31(i,k).. sum((j,l),y(i,k,j,l)) =l= 1;
con32(i,k).. sum((j,l),y(j,l,i,k)) =l= 1;
con41(i).. sum((j,l),y(i,'1',j,l)) =e= 1;
con42(i).. sum((j,l),y(j,l,i,'5')) =e= 1;
con5.. c('1','1') =e= 0;
con6(i,k,j,l)$(not ((ord(j) = 1) and (ord(l) = 1))).. c(j,l) =g= c(i,k) + cost(i,k,j,l)*y(i,k,j,l) -
bigm*(1-y(i,k,j,l)) + wait(i,k,j,l)*w(i,k);
con71(i).. p(i,'6') =e= p(i,'2');
con72(i).. p(i,'7') =e= p(i,'3');
con73(i).. p(i,'8') =e= p(i,'4');
con81(i).. p(i,'2') =e= proca(i);
con82(i).. p(i,'3') + p(i,'4') =e= procb(i);

con91(i)..  p(i,'3') =l= procb(i)*sum((l,j),y(i,'3',j,l)+y(i,'7',j,l));
con92(i)..  p(i,'7') =l= procb(i)*sum((l,j),y(i,'3',j,l)+y(i,'7',j,l));
con93(i)..  p(i,'4') =l= procb(i)*sum((l,j),y(i,'4',j,l)+y(i,'8',j,l));
con94(i)..  p(i,'8') =l= procb(i)*sum((l,j),y(i,'4',j,l)+y(i,'8',j,l));
con95(i)..  p(i,'2') =l= proca(i)*sum((l,j),y(i,'2',j,l)+y(i,'6',j,l));
con96(i)..  p(i,'6') =l= proca(i)*sum((l,j),y(i,'2',j,l)+y(i,'6',j,l));

con011(i)..y(i,'1',i,'3')+y(i,'1',i,'4')+y(i,'1',i,'7')+y(i,'1',i,'8')+y(i,'2',i,'3')+y(i,'2',i,'4')+y(i,'2',i,'7
')+y(i,'2',i,'8')+y(i,'6',i,'3')+y(i,'6',i,'4')+y(i,'6',i,'7')+y(i,'6',i,'8')=l=procb(i);
con012(i).. y(i,'1',i,'2')+y(i,'1',i,'6')=l=proca(i);
con013(i).. y(i,'1',i,'2')+y(i,'1',i,'6')=g=proca(i)/bigm;
con014(i).. y(i,'3',i,'5')+y(i,'4',i,'5')+y(i,'7',i,'5')+y(i,'8',i,'5')=l=procb(i);
con015(i).. y(i,'3',i,'5')+y(i,'4',i,'5')+y(i,'7',i,'5')+y(i,'8',i,'5')=g=procb(i)/bigm;

cons11(i).. start(i,'6') =e= start(i,'2');
cons12(i).. start(i,'7') =e= start(i,'3');
cons13(i).. start(i,'8') =e= start(i,'4');

cons21(i,k)$((ord(k)=1) or (ord(k)=3) or (ord(k)=7) or (ord(k)=4) or (ord(k)=8))..
start(i,'2') =g= c(i,'2')-bigm*(1-y(i,k,i,'2'));
cons22(i,k)$((ord(k)=1) or (ord(k)=3) or (ord(k)=7) or (ord(k)=4) or (ord(k)=8))..
start(i,'2') =l= c(i,'2')+bigm*(1-y(i,k,i,'2'));
cons23(i,k)$((ord(k)=1) or (ord(k)=3) or (ord(k)=7) or (ord(k)=4) or (ord(k)=8))..
start(i,'2') =g= c(i,'6')-bigm*(1-y(i,k,i,'6'));
```

EK-1.(devam) Matematiksel modele ait GAMS/CPLEX kodu

```
cons24(i,k)$((ord(k)=1) or (ord(k)=3) or (ord(k)=7) or (ord(k)=4) or (ord(k)=8))..
start(i,'2') =l= c(i,'6')+bigm*(1-y(i,k,i,'6'));

cons31(i,k)$((ord(k)=1) or (ord(k)=2) or (ord(k)=6) or (ord(k)=4) or (ord(k)=8))..
start(i,'3') =g= c(i,'3')-bigm*(1-y(i,k,i,'3'));
cons32(i,k)$((ord(k)=1) or (ord(k)=2) or (ord(k)=6) or (ord(k)=4) or (ord(k)=8))..
start(i,'3') =l= c(i,'3')+bigm*(1-y(i,k,i,'3'));
cons33(i,k)$((ord(k)=1) or (ord(k)=2) or (ord(k)=6) or (ord(k)=4) or (ord(k)=8))..
start(i,'3') =g= c(i,'7')-bigm*(1-y(i,k,i,'7'));
cons34(i,k)$((ord(k)=1) or (ord(k)=2) or (ord(k)=6) or (ord(k)=4) or (ord(k)=8))..
start(i,'3') =l= c(i,'7')+bigm*(1-y(i,k,i,'7'));

cons41(i,k)$((ord(k)=1) or (ord(k)=2) or (ord(k)=6) or (ord(k)=3) or (ord(k)=7))..
start(i,'4') =g= c(i,'4')-bigm*(1-y(i,k,i,'4'));
cons42(i,k)$((ord(k)=1) or (ord(k)=2) or (ord(k)=6) or (ord(k)=3) or (ord(k)=7))..
start(i,'4') =l= c(i,'4')+bigm*(1-y(i,k,i,'4'));
cons43(i,k)$((ord(k)=1) or (ord(k)=2) or (ord(k)=6) or (ord(k)=3) or (ord(k)=7))..
start(i,'4') =g= c(i,'8')-bigm*(1-y(i,k,i,'8'));
cons44(i,k)$((ord(k)=1) or (ord(k)=2) or (ord(k)=6) or (ord(k)=3) or (ord(k)=7))..
start(i,'4') =l= c(i,'8')+bigm*(1-y(i,k,i,'8'));

cons51(i,k).. c(i,k) =g= start(i,k) - bigm*(1-normal(i,k));
cons52(i,k).. start(i,k) =g= c(i,k) + 4*epsilon + 3*delta - bigm*normal(i,k);

cons6(i,k).. w(i,k) =l= p(i,k);

cons71(i,k).. w(i,k) =g= p(i,k)-(c(i,k)-start(i,k))-bigm*(1-normal(i,k));
cons72(i,k).. w(i,k) =g= p(i,k)-(cmax-start(i,k)+c(i,k))-bigm*normal(i,k);

cons81(i,k)$((ord(k)=2) or (ord(k)=6)).. m2val(i,k) =e= ord(i);
cons82(i,k)$((ord(k)=3) or (ord(k)=7)).. m3val(i,k) =e= ord(i);
cons83(i,k)$((ord(k)=4) or (ord(k)=8)).. m4val(i,k) =e= ord(i);

set11(i,k)$((ord(k)=2) or (ord(k)=6)).. m2val(i,'1') =l= sum(ii,1)*(1-y(i,'1',i,k));
set12(i,k)$((ord(k)=2) or (ord(k)=6)).. -m3val(i,'1')+m3val(i,k) =l= sum(ii,1)*(1-
y(i,'1',i,k));
set13(i,k)$((ord(k)=2) or (ord(k)=6)).. m3val(i,'1')-m3val(i,k) =l= sum(ii,1)*(1-y(i,'1',i,k));
set14(i,k)$((ord(k)=2) or (ord(k)=6)).. -m4val(i,'1')+m4val(i,k) =l= sum(ii,1)*(1-
y(i,'1',i,k));
set15(i,k)$((ord(k)=2) or (ord(k)=6)).. m4val(i,'1')-m4val(i,k) =l= sum(ii,1)*(1-y(i,'1',i,k));

set011(i,k)$((ord(k)=3) or (ord(k)=7)).. m3val(i,'1') =l= sum(ii,1)*(1-y(i,'1',i,k));
set012(i,k)$((ord(k)=3) or (ord(k)=7)).. -m2val(i,'1')+m2val(i,k) =l= sum(ii,1)*(1-
y(i,'1',i,k));
set013(i,k)$((ord(k)=3) or (ord(k)=7)).. m2val(i,'1')-m2val(i,k) =l= sum(ii,1)*(1-
y(i,'1',i,k));
```

EK-1.(devam) Matematiksel modele ait GAMS/CPLEX kodu

```
set014(i,k)$((ord(k)=3) or (ord(k)=7)).. -m4val(i,'1')+m4val(i,k) =l= sum(ii,1)*(1-
y(i,'1',i,k));
set015(i,k)$((ord(k)=3) or (ord(k)=7)).. m4val(i,'1')-m4val(i,k) =l= sum(ii,1)*(1-
y(i,'1',i,k));

set0011(i,k)$((ord(k)=4) or (ord(k)=8)).. m4val(i,'1') =l= sum(ii,1)*(1-y(i,'1',i,k));
set0012(i,k)$((ord(k)=4) or (ord(k)=8)).. -m2val(i,'1')+m2val(i,k) =l= sum(ii,1)*(1-
y(i,'1',i,k));
set0013(i,k)$((ord(k)=4) or (ord(k)=8)).. m2val(i,'1')-m2val(i,k) =l= sum(ii,1)*(1-
y(i,'1',i,k));
set0014(i,k)$((ord(k)=4) or (ord(k)=8)).. -m3val(i,'1')+m3val(i,k) =l= sum(ii,1)*(1-
y(i,'1',i,k));
set0015(i,k)$((ord(k)=4) or (ord(k)=8)).. m3val(i,'1')-m3val(i,k) =l= sum(ii,1)*(1-
y(i,'1',i,k));

set51(i,k)$((ord(k)=3) or (ord(k)=7)).. -m2val(i,k)+m2val(i,'1') =l= sum(ii,1)*(1-
y(i,k,i,'1'));
set52(i,k)$((ord(k)=3) or (ord(k)=7)).. m2val(i,k)-m2val(i,'1') =l= sum(ii,1)*(1-y(i,k,i,'1'));
set53(i,k)$((ord(k)=3) or (ord(k)=7)).. -m3val(i,k)+m3val(i,'1') =l= sum(ii,1)*(1-
y(i,k,i,'1'));
set54(i,k)$((ord(k)=3) or (ord(k)=7)).. m3val(i,k)-m3val(i,'1') =l= sum(ii,1)*(1-y(i,k,i,'1'));
set55(i,k)$((ord(k)=3) or (ord(k)=7)).. -m4val(i,k)+m4val(i,'1') =l= sum(ii,1)*(1-
y(i,k,i,'1'));
set56(i,k)$((ord(k)=3) or (ord(k)=7)).. m4val(i,k)-m4val(i,'1') =l= sum(ii,1)*(1-y(i,k,i,'1'));

set61(i,k)$((ord(k)=4) or (ord(k)=8)).. -m2val(i,k)+m2val(i,'1') =l= sum(ii,1)*(1-
y(i,k,i,'1'));
set62(i,k)$((ord(k)=4) or (ord(k)=8)).. m2val(i,k)-m2val(i,'1') =l= sum(ii,1)*(1-y(i,k,i,'1'));
set63(i,k)$((ord(k)=4) or (ord(k)=8)).. -m3val(i,k)+m3val(i,'1') =l= sum(ii,1)*(1-
y(i,k,i,'1'));
set64(i,k)$((ord(k)=4) or (ord(k)=8)).. m3val(i,k)-m3val(i,'1') =l= sum(ii,1)*(1-y(i,k,i,'1'));
set65(i,k)$((ord(k)=4) or (ord(k)=8)).. -m4val(i,k)+m4val(i,'1') =l= sum(ii,1)*(1-
y(i,k,i,'1'));
set66(i,k)$((ord(k)=4) or (ord(k)=8)).. m4val(i,k)-m4val(i,'1') =l= sum(ii,1)*(1-y(i,k,i,'1'));

set081(i,k)$((ord(k)=2) or (ord(k)=6)).. m2val(i,'5') =l= sum(ii,1)*(1-y(i,k,i,'5'));
set082(i,k)$((ord(k)=2) or (ord(k)=6)).. -m3val(i,k)+m3val(i,'5') =l= sum(ii,1)*(1-
y(i,k,i,'5'));
set083(i,k)$((ord(k)=2) or (ord(k)=6)).. m3val(i,k)-m3val(i,'5') =l= sum(ii,1)*(1-
y(i,k,i,'5'));
set084(i,k)$((ord(k)=2) or (ord(k)=6)).. -m4val(i,k)+m4val(i,'5') =l= sum(ii,1)*(1-
y(i,k,i,'5'));
set085(i,k)$((ord(k)=2) or (ord(k)=6)).. m4val(i,k)-m4val(i,'5') =l= sum(ii,1)*(1-
y(i,k,i,'5'));

set81(i,k)$((ord(k)=3) or (ord(k)=7)).. m3val(i,'5') =l= sum(ii,1)*(1-y(i,k,i,'5'));
```

EK-1.(devam) Matematiksel modele ait GAMS/CPLEX kodu

```
set82(i,k)$((ord(k)=3) or (ord(k)=7)).. -m2val(i,k)+m2val(i,'5') =l= sum(ii,1)*(1-
y(i,k,i,'5'));
set83(i,k)$((ord(k)=3) or (ord(k)=7)).. m2val(i,k)-m2val(i,'5') =l= sum(ii,1)*(1-y(i,k,i,'5'));
set84(i,k)$((ord(k)=3) or (ord(k)=7)).. -m4val(i,k)+m4val(i,'5') =l= sum(ii,1)*(1-
y(i,k,i,'5'));
set85(i,k)$((ord(k)=3) or (ord(k)=7)).. m4val(i,k)-m4val(i,'5') =l= sum(ii,1)*(1-y(i,k,i,'5'));

set91(i,k)$((ord(k)=4) or (ord(k)=8)).. m4val(i,'5') =l= sum(ii,1)*(1-y(i,k,i,'5'));
set92(i,k)$((ord(k)=4) or (ord(k)=8)).. -m2val(i,k)+m2val(i,'5') =l= sum(ii,1)*(1-
y(i,k,i,'5'));
set93(i,k)$((ord(k)=4) or (ord(k)=8)).. m2val(i,k)-m2val(i,'5') =l= sum(ii,1)*(1-y(i,k,i,'5'));
set94(i,k)$((ord(k)=4) or (ord(k)=8)).. -m3val(i,k)+m3val(i,'5') =l= sum(ii,1)*(1-
y(i,k,i,'5'));
set95(i,k)$((ord(k)=4) or (ord(k)=8)).. m3val(i,k)-m3val(i,'5') =l= sum(ii,1)*(1-y(i,k,i,'5'));

set101(i,l,k)$(((ord(l)=2) or (ord(l)=6)) and ((ord(k)=3) or (ord(k)=7))).. m2val(i,k) =l=
sum(ii,1)*(1-y(i,l,i,k));
set102(i,l,k)$(((ord(l)=2) or (ord(l)=6)) and ((ord(k)=3) or (ord(k)=7))).. m3val(i,l) =l=
sum(ii,1)*(1-y(i,l,i,k));
set103(i,l,k)$(((ord(l)=2) or (ord(l)=6)) and ((ord(k)=3) or (ord(k)=7))).. -
m4val(i,l)+m4val(i,k) =l= sum(ii,1)*(1-y(i,l,i,k));
set104(i,l,k)$(((ord(l)=2) or (ord(l)=6)) and ((ord(k)=3) or (ord(k)=7))).. m4val(i,l)-
m4val(i,k) =l= sum(ii,1)*(1-y(i,l,i,k));

set111(i,l,k)$(((ord(l)=2) or (ord(l)=6)) and ((ord(k)=4) or (ord(k)=8))).. m2val(i,k) =l=
sum(ii,1)*(1-y(i,l,i,k));
set112(i,l,k)$(((ord(l)=2) or (ord(l)=6)) and ((ord(k)=4) or (ord(k)=8))).. m4val(i,l) =l=
sum(ii,1)*(1-y(i,l,i,k));
set113(i,l,k)$(((ord(l)=2) or (ord(l)=6)) and ((ord(k)=4) or (ord(k)=8))).. -
m3val(i,l)+m3val(i,k) =l= sum(ii,1)*(1-y(i,l,i,k));
set114(i,l,k)$(((ord(l)=2) or (ord(l)=6)) and ((ord(k)=4) or (ord(k)=8))).. m3val(i,l)-
m3val(i,k) =l= sum(ii,1)*(1-y(i,l,i,k));

set161(i).. m2val(i,'5')-m2val(i,'1') =l= sum(ii,1)*(1-y(i,'5',i,'1'));
set162(i).. -m2val(i,'5')+m2val(i,'1') =l= sum(ii,1)*(1-y(i,'5',i,'1'));
set163(i).. m3val(i,'5')-m3val(i,'1') =l= sum(ii,1)*(1-y(i,'5',i,'1'));
set164(i).. -m3val(i,'5')+m3val(i,'1') =l= sum(ii,1)*(1-y(i,'5',i,'1'));
set165(i).. m4val(i,'5')-m4val(i,'1') =l= sum(ii,1)*(1-y(i,'5',i,'1'));
set166(i).. -m4val(i,'5')+m4val(i,'1') =l= sum(ii,1)*(1-y(i,'5',i,'1'));

set171(i,j,k,l)$(((ord(k)=2) or (ord(k)=6)) and (ord(i) ne ord(j)) and ((ord(l)=1) or
(ord(l)=3) or (ord(l)=7) or (ord(l)=4) or (ord(l)=8)))..
-m2val(i,k)+m2val(j,l)  =l= sum(ii,1)*(1-y(i,k,j,l));
set172(i,j,k,l)$(((ord(k)=2) or (ord(k)=6)) and (ord(i) ne ord(j)) and ((ord(l)=1) or
(ord(l)=3) or (ord(l)=7) or (ord(l)=4) or (ord(l)=8)))..
+m2val(i,k)-m2val(j,l)  =l= sum(ii,1)*(1-y(i,k,j,l));
```

EK-1.(devam) Matematiksel modele ait GAMS/CPLEX kodu

```
set173(i,j,k,l)$(((ord(k)=2) or (ord(k)=6)) and (ord(i) ne ord(j)) and ((ord(l)=1) or
(ord(l)=3) or (ord(l)=7) or (ord(l)=4) or (ord(l)=8)))..
-m3val(i,k)+m3val(j,l)  =l= sum(ii,1)*(1-y(i,k,j,l));
set174(i,j,k,l)$(((ord(k)=2) or (ord(k)=6)) and (ord(i) ne ord(j)) and ((ord(l)=1) or
(ord(l)=3) or (ord(l)=7) or (ord(l)=4) or (ord(l)=8)))..
+m3val(i,k)-m3val(j,l)  =l= sum(ii,1)*(1-y(i,k,j,l));
set175(i,j,k,l)$(((ord(k)=2) or (ord(k)=6)) and (ord(i) ne ord(j)) and ((ord(l)=1) or
(ord(l)=3) or (ord(l)=7) or (ord(l)=4) or (ord(l)=8)))..
-m4val(i,k)+m4val(j,l)  =l= sum(ii,1)*(1-y(i,k,j,l));
set176(i,j,k,l)$(((ord(k)=2) or (ord(k)=6)) and (ord(i) ne ord(j)) and ((ord(l)=1) or
(ord(l)=3) or (ord(l)=7) or (ord(l)=4) or (ord(l)=8)))..
+m4val(i,k)-m4val(j,l)  =l= sum(ii,1)*(1-y(i,k,j,l));

set191(i,j,k,l)$(((ord(k)=3) or (ord(k)=7)) and (ord(i) ne ord(j)) and ((ord(l)=1) or
(ord(l)=2) or (ord(l)=6) or (ord(l)=4) or (ord(l)=8)))..
-m2val(i,k)+m2val(j,l)  =l= sum(ii,1)*(1-y(i,k,j,l));
set192(i,j,k,l)$(((ord(k)=3) or (ord(k)=7)) and (ord(i) ne ord(j)) and ((ord(l)=1) or
(ord(l)=2) or (ord(l)=6) or (ord(l)=4) or (ord(l)=8)))..
+m2val(i,k)-m2val(j,l)  =l= sum(ii,1)*(1-y(i,k,j,l));
set193(i,j,k,l)$(((ord(k)=3) or (ord(k)=7)) and (ord(i) ne ord(j)) and ((ord(l)=1) or
(ord(l)=2) or (ord(l)=6) or (ord(l)=4) or (ord(l)=8)))..
-m3val(i,k)+m3val(j,l)  =l= sum(ii,1)*(1-y(i,k,j,l));
set194(i,j,k,l)$(((ord(k)=3) or (ord(k)=7)) and (ord(i) ne ord(j)) and ((ord(l)=1) or
(ord(l)=2) or (ord(l)=6) or (ord(l)=4) or (ord(l)=8)))..
+m3val(i,k)-m3val(j,l)  =l= sum(ii,1)*(1-y(i,k,j,l));
set195(i,j,k,l)$(((ord(k)=3) or (ord(k)=7)) and (ord(i) ne ord(j)) and ((ord(l)=1) or
(ord(l)=2) or (ord(l)=6) or (ord(l)=4) or (ord(l)=8)))..
-m4val(i,k)+m4val(j,l)  =l= sum(ii,1)*(1-y(i,k,j,l));
set196(i,j,k,l)$(((ord(k)=3) or (ord(k)=7)) and (ord(i) ne ord(j)) and ((ord(l)=1) or
(ord(l)=2) or (ord(l)=6) or (ord(l)=4) or (ord(l)=8)))..
+m4val(i,k)-m4val(j,l)  =l= sum(ii,1)*(1-y(i,k,j,l));

set211(i,j,k,l)$(((ord(k)=4) or (ord(k)=8)) and (ord(i) ne ord(j)) and ((ord(l)=1) or
(ord(l)=2) or (ord(l)=6) or (ord(l)=3) or (ord(l)=7)))..
-m2val(i,k)+m2val(j,l)  =l= sum(ii,1)*(1-y(i,k,j,l));
set212(i,j,k,l)$(((ord(k)=4) or (ord(k)=8)) and (ord(i) ne ord(j)) and ((ord(l)=1) or
(ord(l)=2) or (ord(l)=6) or (ord(l)=3) or (ord(l)=7)))..
+m2val(i,k)-m2val(j,l)  =l= sum(ii,1)*(1-y(i,k,j,l));
set213(i,j,k,l)$(((ord(k)=4) or (ord(k)=8)) and (ord(i) ne ord(j)) and ((ord(l)=1) or
(ord(l)=2) or (ord(l)=6) or (ord(l)=3) or (ord(l)=7)))..
-m3val(i,k)+m3val(j,l)  =l= sum(ii,1)*(1-y(i,k,j,l));
set214(i,j,k,l)$(((ord(k)=4) or (ord(k)=8)) and (ord(i) ne ord(j)) and ((ord(l)=1) or
(ord(l)=2) or (ord(l)=6) or (ord(l)=3) or (ord(l)=7)))..
 +m3val(i,k)-m3val(j,l)  =l= sum(ii,1)*(1-y(i,k,j,l));
set215(i,j,k,l)$(((ord(k)=4) or (ord(k)=8)) and (ord(i) ne ord(j)) and ((ord(l)=1) or
(ord(l)=2) or (ord(l)=6) or (ord(l)=3) or (ord(l)=7)))..
-m4val(i,k)+m4val(j,l)  =l= sum(ii,1)*(1-y(i,k,j,l));
```

EK-1.(devam) Matematiksel modele ait GAMS/CPLEX kodu

```
set216(i,j,k,l)$(((ord(k)=4) or (ord(k)=8)) and (ord(i) ne ord(j)) and ((ord(l)=1) or
(ord(l)=2) or (ord(l)=6) or (ord(l)=3) or (ord(l)=7)))..
 +m4val(i,k)-m4val(j,l)  =l= sum(ii,1)*(1-y(i,k,j,l));

set231(i,j,l)$((ord(j) ne ord(i)) and (ord(l) ne 5)).. m2val(i,'5')-m2val(j,l) =l= sum(ii,1)*(1-
y(i,'5',j,l));
set232(i,j,l)$((ord(j) ne ord(i)) and (ord(l) ne 5)).. -m2val(i,'5')+m2val(j,l) =l=
sum(ii,1)*(1-y(i,'5',j,l));
set233(i,j,l)$((ord(j) ne ord(i)) and (ord(l) ne 5)).. m3val(i,'5')-m3val(j,l) =l= sum(ii,1)*(1-
y(i,'5',j,l));
set234(i,j,l)$((ord(j) ne ord(i)) and (ord(l) ne 5)).. -m3val(i,'5')+m3val(j,l) =l=
sum(ii,1)*(1-y(i,'5',j,l));
set235(i,j,l)$((ord(j) ne ord(i)) and (ord(l) ne 5)).. m4val(i,'5')-m4val(j,l) =l= sum(ii,1)*(1-
y(i,'5',j,l));
set236(i,j,l)$((ord(j) ne ord(i)) and (ord(l) ne 5)).. -m4val(i,'5')+m4val(j,l) =l=
sum(ii,1)*(1-y(i,'5',j,l));
cons9(i,k).. cmax =g= c(i,k)+y(i,k,'1','1')*cost(i,k,'1','1');

bolme1(i).. y(i,'3',i,'4') + y(i,'4',i,'3') + y(i,'3',i,'8') + y(i,'4',i,'7') +  y(i,'7',i,'4') + y(i,'8',i,'3') +
y(i,'7',i,'8') + y(i,'8',i,'7') =e= 0;
bolme2(i).. sum((j,k),(y(j,k,i,'3') + y(j,k,i,'4'))) =l= 1;
bolme3(i).. sum((j,k),(y(j,k,i,'3') + y(j,k,i,'8'))) =l= 1;
bolme4(i).. sum((j,k),(y(j,k,i,'7') + y(j,k,i,'4'))) =l= 1;
bolme5(i).. sum((j,k),(y(j,k,i,'7') + y(j,k,i,'8'))) =l= 1;

scon11(i).. y(i,'1',i,'2') + y(i,'1',i,'6') + y(i,'1',i,'3') + y(i,'1',i,'7') + y(i,'1',i,'4') + y(i,'1',i,'8')
=e= 1;
scon12(i).. y(i,'2',i,'5') + y(i,'6',i,'5') + y(i,'3',i,'5') + y(i,'7',i,'5') + y(i,'4',i,'5') + y(i,'8',i,'5')
=e= 1;
scon13(i).. y(i,'2',i,'3') + y(i,'2',i,'7') + y(i,'6',i,'3') + y(i,'6',i,'7') + y(i,'2',i,'4') + y(i,'2',i,'8') +
y(i,'6',i,'4') + y(i,'6',i,'8') =l= 1;

objdef..     obj =e= cmax;

Model simple /all/;
solve simple using mip minimizing obj;

option c:3:0:1;
display c.l;

option y:3:0:1;
display y.l;

display cmax.l;
```

EK-2. Matematiksel modele ait GAMS/CPLEX çözümü çıktısı

```
G e n e r a l  A l g e b r a i c  M o d e l i n g  S y s t e m
Model Statistics    SOLVE simple Using MIP From line 427

MODEL STATISTICS
BLOCKS OF EQUATIONS     137     SINGLE EQUATIONS     2754
BLOCKS OF VARIABLES      11     SINGLE VARIABLES      770
NON ZERO ELEMENTS     11988     DISCRETE VARIABLES    600
GENERATION TIME      =        0.047 SECONDS    2.1 Mb      WIN212-136
EXECUTION TIME       =        0.047 SECONDS    2.1 Mb      WIN212-136

Solution Report     SOLVE simple Using MIP From line 427

          S O L V E     S U M M A R Y

   MODEL   simple           OBJECTIVE  obj
   TYPE    MIP              DIRECTION  MINIMIZE
   SOLVER  CPLEX            FROM LINE  427

**** SOLVER STATUS     8 USER INTERRUPT
**** MODEL STATUS      8 INTEGER SOLUTION
**** OBJECTIVE VALUE              77.0000

 RESOURCE USAGE, LIMIT      86344.750 360000000000.000
 ITERATION COUNT, LIMIT 841043886      50000000

GAMS/Cplex    Sep  3, 2003 WIN.CP.CP 21.2 023.026.041.VIS For Cplex 8.1
Cplex 8.1.0, GAMS Link 23
Cplex licensed for 1 use of lp, mip and barrier.

MIP Solution:          77.000000    (841043863 iterations, 15832377 nodes)
Final Solve:           77.000000    (23 iterations)

Best integer solution possible:         68.000000
Absolute gap:            9.000000
Relative gap:            0.116883

**** REPORT SUMMARY :       0     NONOPT
                            0 INFEASIBLE
                            0  UNBOUNDED

E x e c u t i o n

----    433 VARIABLE w.L

1.3 30.000
1.6 17.000
1.7  6.000
2.2 20.000
2.4  1.000
2.8 34.000
3.3  9.000
3.7 27.000
```

EK-2.(devam) Matematiksel modele ait GAMS/CPLEX çözümü çıktısı

---- 439 VARIABLE c.L

1.2 21.000
1.3 25.000
1.5 59.000
1.6 4.000
1.7 49.000
2.1 34.000
2.2 47.000
2.4 27.000
2.5 32.000
2.6 67.000
2.8 71.000
3.1 61.000
3.3 6.000
3.5 19.000
3.7 65.000

---- 444 VARIABLE y.L

1.1.1.6 1.000
1.2.1.3 1.000
1.3.2.4 1.000
1.5.3.1 1.000
1.6.3.3 1.000
1.7.1.5 1.000
2.1.2.2 1.000
2.2.1.7 1.000
2.4.2.5 1.000
2.5.2.1 1.000
2.6.2.8 1.000
2.8.1.1 1.000
3.1.3.7 1.000
3.3.3.5 1.000
3.5.1.2 1.000
3.7.2.6 1.000

---- 448 VARIABLE m2val.L

1.2 1.000
1.5 2.000
1.6 1.000
1.7 2.000
2.2 2.000
2.6 2.000
3.1 2.000
3.2 3.000
3.3 1.000
3.5 1.000
3.6 3.000
3.7 2.000

EK-2.(devam) Matematiksel modele ait GAMS/CPLEX çözümü çıktısı

---- 450 VARIABLE m3val.L

1.1 3.000
1.3 1.000
1.6 3.000
1.7 1.000
2.1 1.000
2.2 1.000
2.3 2.000
2.4 1.000
2.5 1.000
2.6 3.000
2.7 2.000
2.8 3.000
3.3 3.000
3.7 3.000

---- 452 VARIABLE m4val.L

1.1 2.000
1.2 2.000
1.3 2.000
1.4 1.000
1.6 2.000
1.8 1.000
2.4 2.000
2.8 2.000
3.2 3.000
3.3 2.000
3.4 3.000
3.5 2.000
3.6 3.000
3.8 3.000

---- 457 VARIABLE cmax.L = 77.000

EXECUTION TIME = 0.109 SECONDS 1.5 Mb WIN212-136

EK-3. Sezgisel algoritmaya ait MS Visual C++ kodu

- *Problem*

```
void ClsProblem::take_data(string ProblemRootDirectory, string ProblemFileName){
	int i,j;
	string tmpstr;
	ifstream infile;
	PrbName=ProblemFileName;
	infile.open(ProblemRootDirectory+ProblemFileName);

	infile>>tmpstr>>JobNum;
	infile>>tmpstr>>StageNum;
	infile>>tmpstr>>RobotLoadUnloadTime;
	infile>>tmpstr>>RobotMoveTime;

	MacNum.resize(StageNum);
	ProcTime.resize(JobNum);
	for (i=0; i<JobNum; i++)
		ProcTime[i].resize(StageNum);

	infile>>tmpstr;
	for (i=TotalMacNum=0; i<StageNum; i++){
		infile>>MacNum[i];
		TotalMacNum+=MacNum[i];
	}

	infile>>tmpstr;
	for (i=0; i<JobNum; i++)
		for (j=0; j<StageNum; j++)
			infile>>ProcTime[i][j];

	infile.close();
}
```

- *FeasibleSolution*

```
void ClassFeasSol::Solve(){
	int CurObs;
	double SolTime;
	clock_t Start;
	Initialize();

	for (CurObs=0; CurObs<ObsNum; CurObs++){
		srand(RandomizedSeedsForMe[CurObs]);
		Start=clock();
		ObtainFeasSol();
		SA();
		if (IsLT(BestObjVal, OverAllBestObjVal)){
			OverAllBestObjVal=BestObjVal;
			OverAllBestSol=BestSol;
		}
		SolTime=(((double)clock()-Start)/CLOCKS_PER_SEC);
		GeneralReport(CurObs+1, SolTime);
```

EK-3.(devam) Sezgisel algoritmaya ait MS Visual C++ kodu

```
        }
        WriteSolToMonitor(OverAllBestSol, OverAllBestObjVal);
        WriteSolToFile(OverAllBestSol, OverAllBestObjVal);
}

void ClassFeasSol::SA(){
        int i,j;
        int Pos1, Pos2;
        int MaxVal, SelJob1, SelStage1, SelJob2, SelStage2;
        double Delta, Temperature, SelectionProb=0.5;
        bool Accept;
        Temperature=PrmSAInitT;
        vector<int> TmpSol, IterBestSol;
        double TmpObjVal, IterBestObjVal;

        int NumNewSol=Prb->JobNum;
        do{
                IterBestObjVal=INT_MAX;
                MaxVal=INT_MIN;

                for (i=0; i<NumNewSol; i++){
                        TmpSol=Sol;
                        if (RandomNumber()>0.5){
                                do{
                                        Pos1=RandomIntNumber(Prb->JobNum);
                                        Pos2=RandomIntNumber(Prb->JobNum);
                                }while(Pos1==Pos2);
                                swap(TmpSol[Pos1], TmpSol[Pos2]);
                        }
                        else{
                                int CurJob, CurStg, CurMac, NewMac;
                                do{
                                        CurJob=RandomIntNumber(Prb->JobNum);
                                        CurStg=RandomIntNumber(Prb->StageNum);
                                        CurMac=Sol[ CurJob + (CurStg+1)*Prb->JobNum ];
                                        NewMac=RandomIntNumber(Prb->MacNum[CurStg]);
                                }while(CurMac==NewMac || Prb->MacNum[CurStg]==1);
                                TmpSol[ CurJob + (CurStg+1)*Prb->JobNum ]=NewMac;
                        }

                        TmpObjVal=ObjFuncEval(TmpSol);
                        if (TmpObjVal<IterBestObjVal){
                                IterBestObjVal=TmpObjVal;
                                IterBestSol=TmpSol;
                        }
                }

                Delta=100*((IterBestObjVal-ObjVal)/(double)ObjVal);
                Accept=(RandomNumber()<exp(-Delta/Temperature));
                if (Delta<0 || Accept){
                        ObjVal=IterBestObjVal;
                        Sol=IterBestSol;
```

EK-3.(devam) Sezgisel algoritmaya ait MS Visual C++ kodu

```
                        if (IsLT(ObjVal, BestObjVal)){
                                BestObjVal=ObjVal;
                                BestSol=Sol;
                        }
                }

                Temperature*=PrmSADecRatio;
        }while (Temperature>PrmSAFinalT);
}

void ClassFeasSol::ObtainFeasSol(){
        int i,j;
        for(i=0; i<Prb->JobNum; i++){
                Sol[i]=i;
                for (j=0; j<Prb->StageNum; j++)
                        Sol[i+(j+1)*Prb->JobNum]=rand()%Prb->MacNum[j];
        }
        random_shuffle(Sol.begin(),Sol.begin()+Prb->JobNum);

        ObjVal=BestObjVal=InitObjVal=ObjFuncEval(Sol);
        BestSol=Sol;
}

void ClassFeasSol::Empty(){
        int i;
        for (i=0; i<Ganth.size(); i++)
                Ganth[i].Empty();
        EventList.resize(0);
        FeasibleEvents.resize(0);
        LotInf.resize(0);
        Robot.Empty();
}

void ClassFeasSol::Initialize(){
        int i,j,k;

        srand(12345);
        RandomizedSeedsForMe.resize(ObsNum);
        for (i=0; i<ObsNum; i++){
                RandomizedSeedsForMe[i]=rand();
        }

        Ganth.resize(Prb->TotalMacNum);
        GanthPtr.resize(Prb->StageNum);
        for (i=k=0; i<Prb->StageNum; i++){
                GanthPtr[i].resize(Prb->MacNum[i]);
                for (j=0; j<Prb->MacNum[i]; j++, k++){
                        Ganth[k].Stage=i;
                        Ganth[k].Mac=j;
                        GanthPtr[i][j]=&Ganth[k];
                }
```

EK-3.(devam) Sezgisel algoritmaya ait MS Visual C++ kodu

```
	}

	Robot.Prb=Prb;
	OverAllBestObjVal=INT_MAX;
	EncodingDimension=Prb->JobNum*(Prb->StageNum+1);
	Sol.resize(EncodingDimension);
	LowerBound=INT_MIN;
	int CurStageJobNum;
	double CurStageProcTime;
	for (i=0; i<Prb->StageNum; i++){
		for (CurStageJobNum=CurStageProcTime=0, j=0; j<Prb->JobNum; j++){
			CurStageProcTime+=Prb->ProcTime[j][i];
			if (Prb->ProcTime[j][i])
				CurStageJobNum++;
		}
		CurStageProcTime += (CurStageJobNum*(4*Prb->RobotLoadUnloadTime +
3*Prb->RobotMoveTime));
		CurStageProcTime/=Prb->MacNum[i];

		if (CurStageProcTime>LowerBound)
			LowerBound=CurStageProcTime;
	}
}

double ClassFeasSol::ObjFuncEval(vector<int> &Solution){
	int i,j;
	double PrevCycleTime, CurCycleTime;
	int NumLoadedJobToQueue=0, NumLoadedLot=0, JobIndex=0;
	vector<double> CycleTimes, LotCycleTimes;
	bool SteadyState=false;
	Empty();

	JobMacAss.resize(Prb->JobNum);
	for (int CurJob=0; CurJob<Prb->JobNum; CurJob++){
		JobMacAss[CurJob].JobNo=CurJob;
		JobMacAss[CurJob].StageNum=0;
		JobMacAss[CurJob].StageSeq.resize(0);
		JobMacAss[CurJob].MacSeq.resize(0);
		JobMacAss[CurJob].WaitTimeSeq.resize(0);
		for (int CurMac, CurStage=0; CurStage<Prb->StageNum; CurStage++){
			if (Prb->ProcTime[CurJob][CurStage]!=0){
				CurMac=Solution[CurJob+(CurStage+1)*Prb->JobNum];
				JobMacAss[CurJob].StageSeq.push_back(CurStage);
				JobMacAss[CurJob].MacSeq.push_back(CurMac);
				JobMacAss[CurJob].WaitTimeSeq.push_back(0);
				JobMacAss[CurJob].StageNum++;
				GanthPtr[CurStage][CurMac]->TotalAssignedJobNum++;
				GanthPtr[CurStage][CurMac]->TotalJobDuration+=Prb-
>ProcTime[CurJob][CurStage];
				GanthPtr[CurStage][CurMac]->TotalJobDuration+=(4*Prb-
>RobotLoadUnloadTime+3*Prb->RobotMoveTime);
```

EK-3.(devam) Sezgisel algoritmaya ait MS Visual C++ kodu

```
                                GanthPtr[CurStage][CurMac]-
>MacPriority=GanthPtr[CurStage][CurMac]-
>TotalJobDuration*10+GanthPtr[CurStage][CurMac]->TotalAssignedJobNum;
                        }
                }
        }

        ClsEvent FirstJob;
        FirstJob.LoadNextJobInQueue(Solution[0], NumLoadedJobToQueue, GanthPtr,
JobMacAss);
        EventList.push_back(FirstJob);

        bool CycleTimeLoop=true;
        do{
                bool TryAgain;
                ClsEvent *CurEvent;
                FeasibleEvents.resize(0);
                do{
                        TryAgain=false;
                        for (i=0; i<EventList.size(); i++){
                                CurEvent=&EventList[i];
                                bool AcceptCriterion, AcceptCriterion1=false,
AcceptCriterion2=false, AcceptCriterion3=false;
                                if(CurEvent->EventType==EVENT_TYPE::SEIZE){
                                        AcceptCriterion1 = CurEvent->CurMachineGanth-
>Status==STATUS_TYPE::IDLE;
                                }
                                else if(CurEvent->EventType==EVENT_TYPE::PROCESS){
                                        if (CurEvent->NextMachineGanth==NULL)
                                        AcceptCriterion2=true;
                                        else if (CurEvent->NextMachineGanth-
>Status==STATUS_TYPE::IDLE)      AcceptCriterion3=true;
                                }

                                AcceptCriterion=AcceptCriterion1 || AcceptCriterion2 ||
AcceptCriterion3;
                                if (AcceptCriterion && CurEvent->CurMachineGanth-
>ReadyTime<=Robot.ReadyTime){
                                        FeasibleEvents.push_back(CurEvent);
                                }
                        }

                        if (FeasibleEvents.size()==0){
                                int MinTime=INT_MAX;
                                for (i=0, MinTime=INT_MAX; i<EventList.size(); i++){
                                        ClsEvent *CurEvent=&EventList[i];
                                        if (CurEvent->EventType==EVENT_TYPE::PROCESS
&& CurEvent->CurMachineGanth->ReadyTime<MinTime){
                                                if (CurEvent->NextMachineGanth==NULL ||
                                                        CurEvent->NextMachineGanth-
>Status==STATUS_TYPE::IDLE){
```

EK-3.(devam) Sezgisel algoritmaya ait MS Visual C++ kodu

```
                                                    MinTime=CurEvent->CurMachineGanth->ReadyTime;
                                                }
                                            }
                                        }
                                        Robot.ReadyTime=MinTime;
                                        TryAgain=true;
                                    }
                                    else if (FeasibleEvents.size()==1){
                                        CurEvent=FeasibleEvents[0];
                                    }
                                    else{
                                        int MaxPriority=INT_MAX;
                                        for (i=0; i<FeasibleEvents.size(); i++){
                                            if (FeasibleEvents[i]->CurMachineGanth->ReadyTime<MaxPriority){
                                                MaxPriority=FeasibleEvents[i]->CurMachineGanth->ReadyTime;
                                                CurEvent=FeasibleEvents[i];
                                            }
                                        }
                                    }
                                }while(TryAgain);

                                if (CurEvent->EventType==EVENT_TYPE::SEIZE){
                                    CurEvent->EventType=EVENT_TYPE::PROCESS;
                                    Robot.EvaluateWorkTime(CurEvent->PrevStage, CurEvent->PrevMac);
                                    Robot.AtStage=CurEvent->CurStage;
                                    Robot.AtMachine=CurEvent->CurMachine;
                                    CurEvent->CurMachineGanth->ReadyTime=Robot.ReadyTime+Prb->ProcTime[CurEvent->Job][CurEvent->CurStage];
                                    CurEvent->CurMachineGanth->Status=STATUS_TYPE::BUSY;

                                    JobIndex=NumLoadedJobToQueue%Prb->JobNum;
                                    NumLoadedLot=(NumLoadedJobToQueue-1)/Prb->JobNum;
                                    CurEvent->WhichLot=NumLoadedLot;
                                    if (JobIndex==0){
                                        CurEvent->JobType=JOB_TYPE::LASTJOB;
                                    }
                                    else if (JobIndex==1){
                                        ClsLot NewLot;
                                        LotInf.push_back(NewLot);
                                        CurEvent->JobType=JOB_TYPE::FIRSTJOB;
                                        LotInf[NumLoadedLot].NumLoadedJob++;
                                        LotInf[NumLoadedLot].StartTime=Robot.ReadyTime;
                                        int TotalLoadedLotNum=LotInf.size();
                                        int AvgNum=Prb->StageNum;
                                        if (SteadyState){
                                            PrevCycleTime=LotInf[ TotalLoadedLotNum-1 ].StartTime - LotInf[ TotalLoadedLotNum-2 ].StartTime;
                                            CurCycleTime =LotInf[ TotalLoadedLotNum-2 ].StartTime - LotInf[ TotalLoadedLotNum-3 ].StartTime;
```

EK-3.(devam) Sezgisel algoritmaya ait MS Visual C++ kodu

```
                                        if(PrevCycleTime==CurCycleTime){
                                                CycleTimeLoop=false;
                                        }else{
                                                if (LotInf.size()>AvgNum){
                                                        double CycleTime=0;
                                                        for (j=0, i=LotInf.size()-1; j<AvgNum; i--,
j++){
                                                                CycleTime+=LotInf[i].StartTime -
LotInf[i-1].StartTime;
                                                        }

        CycleTimes.push_back(CycleTime/AvgNum);
                                                        if (CycleTimes.size()>1){
                                                                if (CycleTimes[
CycleTimes.size()-1 ] == CycleTimes[ CycleTimes.size()-1 ]){
                                                                        CycleTimeLoop=false;
        CurCycleTime=CycleTimes[ CycleTimes.size()-1 ];
                                                                }
                                                        }
                                                }
                                        }
                                }
                                else{
                                        CycleTimeLoop=true;
                                }
                        }

                        ClsEvent NextJob;
                        NextJob.LoadNextJobInQueue(Solution[JobIndex],
NumLoadedJobToQueue, GanthPtr, JobMacAss);
                        EventList.push_back(NextJob);
                        }
                else if (CurEvent->EventType==EVENT_TYPE::PROCESS){
                        if (CurEvent->NextMachineGanth==NULL){
                                Robot.EvaluateWorkTime(CurEvent->CurStage, CurEvent-
>CurMachine);
                                Robot.AtStage=-2;
                                Robot.AtMachine=-2;
                                CurEvent->CurMachineGanth->Status=STATUS_TYPE::IDLE;
                                CurEvent->EventType=EVENT_TYPE::FINISHED;
                                if (CurEvent->JobType==JOB_TYPE::LASTJOB){
                                        LotInf[ CurEvent->WhichLot
].CompletionTime=CurEvent->CurMachineGanth->ReadyTime;
                                        LotInf[ CurEvent->WhichLot ].CycleTime=LotInf[
CurEvent->WhichLot ].CompletionTime-LotInf[ CurEvent->WhichLot ].StartTime;
                                        if (CurEvent->WhichLot>=1 && LotInf[ CurEvent-
>WhichLot ].CycleTime==LotInf[ CurEvent->WhichLot-1 ].CycleTime)
                                                SteadyState=true;

                                        int AvgNum=Prb->StageNum;
                                        if (!SteadyState && CurEvent->WhichLot>=(AvgNum-
1)){
```

EK-3.(devam) Sezgisel algoritmaya ait MS Visual C++ kodu

```
                                        double LotCycleTime=0;
                                        for (j=0, i=CurEvent->WhichLot; j<AvgNum; i--,
j++){
                                                LotCycleTime+=LotInf[i].CycleTime;
                                        }

        LotCycleTimes.push_back(LotCycleTime/AvgNum);
                                        if (LotCycleTimes.size()>1){
                                                if (LotCycleTimes[ LotCycleTimes.size()-
1 ] == LotCycleTimes[ LotCycleTimes.size()-1 ]){
                                                        SteadyState=true;
                                                }
                                        }
                                }

                        }
                }
                else{
                        CurEvent->CurMachineGanth->Status=STATUS_TYPE::IDLE;
                        CurEvent->PrevMachineGanth=CurEvent->CurMachineGanth;
                        CurEvent->PrevStage=CurEvent->CurStage;
                        CurEvent->PrevMac=CurEvent->CurMachine;
                        CurEvent->CurStage=CurEvent->NextStage;
                        CurEvent->CurMachine=CurEvent->NextMachine;
                        CurEvent->CurMachineGanth=CurEvent->NextMachineGanth;
                        CurEvent->StageIndex++;

                        Robot.EvaluateWorkTime(CurEvent->PrevStage, CurEvent-
>PrevMac);
                        Robot.AtStage=CurEvent->CurStage;
                        Robot.AtMachine=CurEvent->CurMachine;

                        int Diff;
                        Diff=MAX(Robot.ReadyTime, CurEvent->NextMachineGanth-
>ReadyTime);
                        Diff-=CurEvent->PrevMachineGanth->ReadyTime;
                        JobMacAss[CurEvent->Job].WaitTimeSeq[CurEvent-
>CurStage]+=Diff;

                        CurEvent->CurMachineGanth-
>ReadyTime=Robot.ReadyTime+Prb->ProcTime[CurEvent->Job][CurEvent->CurStage];
                        CurEvent->CurMachineGanth->Status=STATUS_TYPE::BUSY;
                        if(JobMacAss[CurEvent->Job].StageNum>(CurEvent-
>StageIndex+1)){
                                CurEvent->NextStage=JobMacAss[CurEvent-
>Job].StageSeq[CurEvent->StageIndex+1];
                                CurEvent->NextMachine=JobMacAss[CurEvent-
>Job].MacSeq[CurEvent->StageIndex+1];
                                CurEvent->NextMachineGanth= GanthPtr[CurEvent-
>NextStage][CurEvent->NextMachine];
                        }
                        else{
```

EK-3.(devam) Sezgisel algoritmaya ait MS Visual C++ kodu

```
                                        CurEvent->NextStage=-1;
                                        CurEvent->NextMachine=-1;
                                        CurEvent->NextMachineGanth= NULL;
                                }
                        }
                }
        }while(CycleTimeLoop);
        UpdateStats(CurCycleTime);
        return CurCycleTime;
}
```

EK-4. Sezgisel algoritmaya ait örnek problem verileri ve çözüm çıktısı

- *Problem verileri*

JobNum 25
StageNum 2
Epsilon 6
Delta 6

MachineNum
1
2

ProcTime

81	126
128	187
0	79
91	136
79	149
0	22
135	178
0	189
47	109
161	192
26	124
71	142
120	163
28	153
0	52
0	43
171	173
66	83
45	199
0	104
35	129
0	32
52	79
44	120
0	169

- *Çözüm Çıktısı*

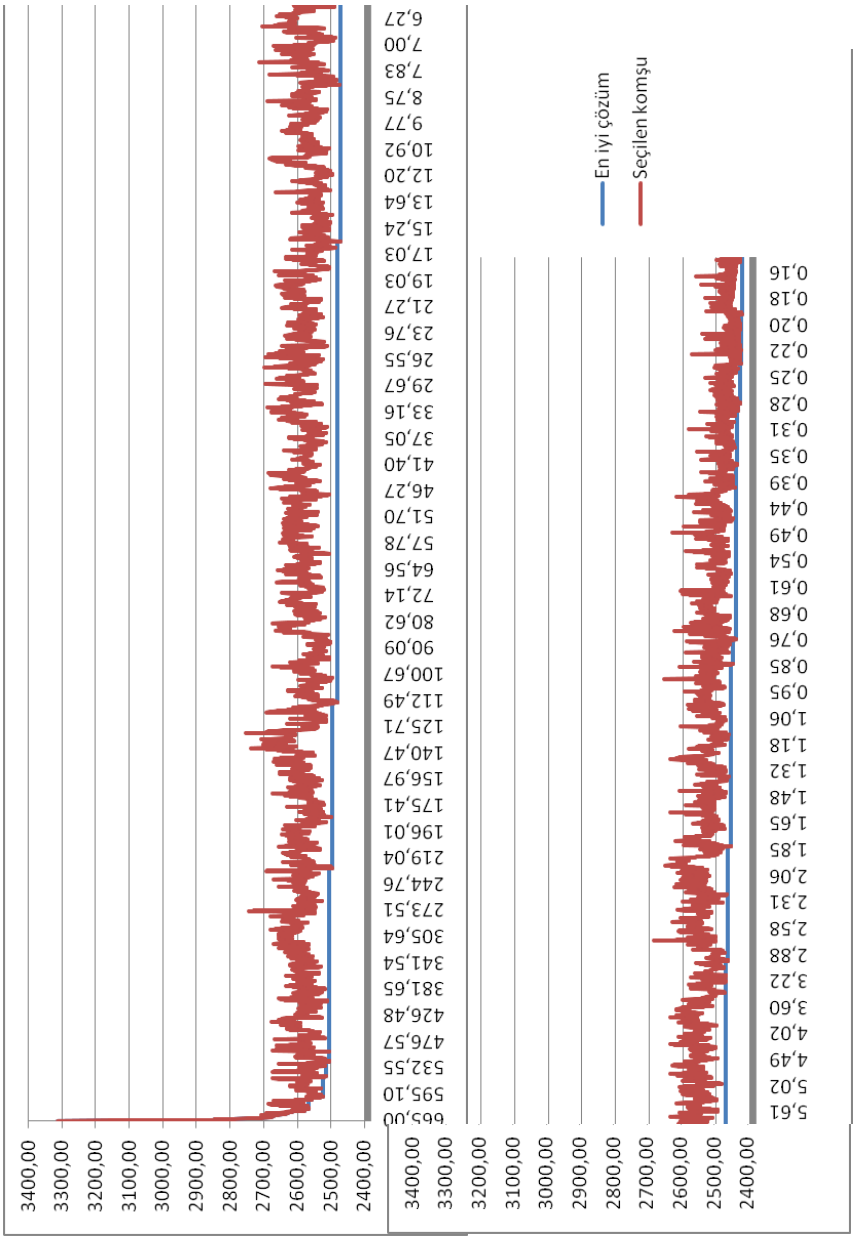
Lower Bound = 2094
Cycle Time = 2180

Job	Stg_1	Stg_2
19	1	2
9	1	1
17	1	1
20	1	2
14	1	2
1	1	1
21	1	2
23	1	1

EK-4.(devam) Sezgisel algoritmaya ait örnek problem verileri ve çözüm çıktısı

2	1	2
15	1	2
3	1	1
13	1	1
16	1	1
12	1	2
11	1	1
24	1	2
7	1	2
8	1	1
5	1	1
18	1	2
10	1	2
25	1	1
22	1	2
4	1	1
6	1	1

EK-5. T_i=665 için iterasyon bazında en iyi çevrim zamanı ve komşu çözümler



Şekil 1.1. İterasyon bazında en iyi çevrim zamanı ve komşu çözümler

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : BATUR SİR, Gül Didem

Doğum tarihi ve yeri : 09/06/1984 Samsun

E-posta : dbatur@gazi.edu.tr

Eğitim Derecesi	**Okul / Program**	**Mezuniyet Yılı**
Doktora	Gazi Üniv. / Endüstri Müh. Böl.	2014
Yüksek lisans	Bilkent Üniv. / Endüstri Müh. Böl.	2009
Lisans	Gazi Üniv. / Endüstri Müh. Böl.	2006
Lise	Samsun Atatürk Anadolu Lisesi	2002

İş Deneyimi, Yıl	**Çalıştığı Yer**	**Görev**
2009 - devam ediyor	Gazi Üniversitesi	Araştırma Görevlisi
2006 - 2009	Bilkent Üniversitesi	Araştırma Görevlisi

Yabancı Dili

İngilizce

Yayınlar

- Uluslararası hakemli dergilerde yayınlanan makaleler

1. **Batur, G.D.**, Karasan, O. and Akturk, M.S. (2012) Multiple Part-Type Scheduling in Flexible Robotic Cells. International Journal of Production Economics, 135(2):726-740.

- Uluslararası bilimsel toplantılarda sunulan bildiriler

1. **Batur, G.D.**, Erol, S. (2010). Multiple Part-Type, 3 Parallel Machine Scheduling In Robotic Cells, Proceedings of 24th European Conference on Operational Research, p221.
2. **Batur, G.D.**, Erol, S. (2011). A Solution Approach for Multiple Part-Type Scheduling in Robotic Cells, Proceedings of INFORMS Annual Meeting, p408.
3. **Batur, G.D.**, Erol, S. (2012). Robotic Cell Scheduling in Hybrid Flexible Flowshops, Proceedings of INFORMS Annual Meeting, p458.
4. **Batur Sir, G.D.**, Erol, S. (2013). Robotic Cell Scheduling in Hybrid Flexible Flowshops, Proceedings of 26th European Conference on Operational Research, p233.

- Ulusal bilimsel toplantılarda sunulan bildiriler

1. **Batur, G.D.**, Karasan, O.E. ve Akturk, M.S. (2009). Esnek Robotik Hücrelerde Çoklu Parça Tipi Çizelgelemesi", YA/EM'2009, Yöneylem Araştırması / Endüstri Mühendisliği 29. Ulusal Kongresi, Bildiri Özetleri Kitabı, s134.

2. **Batur, G.D.**, Erol S. ve Akcayol, M.A. (2011). İki Makineli Robotik Hücrelerde Çoklu Parça Tipi Çizelgemesi İçin Bir Tavlama Benzetimi Algoritması. ÜAS'2011 XI. Ulusal Üretim Araştırmaları Sempozyumu, Bildiriler Kitabı, s71-80.

3. Rouyendegh, B.D., **Batur, G.D.**, Sir, E., Erol, S. (2013). The Dea And Fuzzy Ahp Approach To Health-Care Organızatıons' Performances, Proceedings of the 11th International Conference of DEA, p351-357.

GAZİ GELECEKTİR..