Journal of the Faculty of Engineering and Architecture of Gazi University 39:3 (2024) 1733-1743

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

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

Hybrid simulated annealing-tabu search algorithms for solving U-shaped type-2 assembly line balancing problems with workload smoothing objective

Murat Arıkan*
Department of Industrial Engineering, Faculty of Engineering, Gazi University, 06570, Ankara, Türkiye

Highlights:

- U-shaped type-2 simple assembly line balancing problem with a workload balancing objective is considered
- The problem is dealt with two different hybrid simulated annealing-tabu search algorithms
- The hybrid TSSA algorithm achieved very successful results for both objectives considered

Keywords:

- U-shaped assembly lines
- type-2 assembly line balancing
- Simulated annealing
- Tabu search
- Hybrid meta-heuristics

Article Info:

Research Article
Received: 31.03.2023
Accepted: 30.08.2023

DOI:

10.17341/gazimmfd.1274474

Correspondence:

Author: Murat Arıkan
e-mail: marikan@gazi.edu.tr
phone: +90 312 582 3808

Graphical/Tabular Abstract

This study focuses on the U-shaped type-2 simple assembly line balancing problem with a workload smoothing objective. The problem is handled by two hybrid simulated annealing (SA)-tabu search (TS) algorithms with different working principles (SATS and TSSA, respectively) that have been proposed in the literature. This is the first time a U-shaped assembly line balancing problem has been addressed by hybrid simulated annealing-tabu search algorithms. The performances of hybrid algorithms are compared with each other and with pure versions of simulated annealing and tabu search algorithms on 128 instances of 9 problems taken from the literature.

	Algorithms			
	TSSA	SATS	TS	SA
best % deviation from cycle time lower bound	0.24	0.38	0.40	0.44
average % deviation from cycle time lower bound	0.25	0.54	0.52	0.67
worst % deviation from cycle time lower bound	0.26	0.71	0.59	1.00
# of optimal solutions with recpect to cycle time out of 128 instances	97	79	79	72
mean absolute deviation from average workload	2.56	10.03	7.00	12.30
average CPU s	32.87	20.66	7.84	17.90

Table A. Comparison of results obtained in test problems

Purpose:
The aim of the study is to develop efficient solution methods for solving the U-shaped type-2 assembly line balancing problem with a workload smoothing objective.

Theory and Methods:
The considered problem is solved by two distinct hybrid simulated annealing-tabu search algorithms, one of which is based on simulated annealing (Zolfaghari and Liang, 1999) and the other is based on tabu search (Zhang et al., 2008).

Results:
The performance of hybrid simulated annealing-tabu search algorithms is evaluated on 128 instances of 9 problems taken from literature, by comparing the results obtained for both objectives with each other and with those obtained from the pure versions of the simulated annealing and tabu search algorithms (Table A). The computational results show that the hybrid algorithm, the core of which is tabu search (TSSA), is superior to the others for both objectives.

Conclusion:
The best-performing TSSA algorithm can be safely used for the solution of the problem in terms of both the quality of the objective function values and the reasonableness of the solution time. In addition, secondary objective results, which do not have a result set in the literature, can provide comparison opportunities for researchers dealing with the same problem.

Journal of the Faculty of Engineering and Architecture of Gazi University 39:3 (2024) 1733-1743

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

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

U-şekilli hatlarda iş yükü dengelemeli tip-2 montaj hattı dengeleme probleminin çözümü için melez tavlama benzetimi-tabu arama algoritmaları

Murat Arıkan*
Gazi Üniversitesi, Mühendislik Fakültesi, Endüstri Mühendisliği Bölümü, 06570, Maltepe, Ankara, Türkiye

ÖNEÇIKANLAR

- U-şekilli tip-2 basit montaj hattı dengeleme problemi iş yükü dengeleme amacıyla ele alınmıştır
- Problem, iki farklı melez tavlama benzetimi-tabu arama algoritması ile çözülmüştür
- Melez TSSA algoritması, dikkate alınan her iki amaç için de çok başarılı sonuçlar elde etmiştir

Makale Bilgileri
Araştırma Makalesi
Geliş: 31.083.2023
Kabul: 30.08.2023

DOI:
10.17341/gazimmfd.1274474

Anahtar Kelimeler:
U-şekilli montaj hatları,
tip-2 montaj hattı dengeleme,
tavlama benzetimi,
tabu arama,
melez meta-sezgiseller

ÖZ

Son yıllarda, tam zamanında üretim felsefesini benimseyen işletmelerin artmasıyla birlikte U-şekilli montaj hatlarının kullanımı yaygınlaşmıştır. Dolayısıyla, U-şekilli montaj hattı dengeleme problemlerinin çözüm yöntemleri üzerine yapılan çalışmalar çoğalmaktadır. Bu çalışmada, iş yükü düzgünleştirme amacını ikincil amaç olarak dikkate alan, tip-2 U-şekilli basit montaj hattı dengeleme probleminin çözümü için, literatürde daha önce geliştirilmiş ve farklı problemler üzerinde etkinliği gösterilmiş olan, iki adet farklı yapıda melez tavlama benzetimi-tabu arama algoritması kullanılmıştır. Biri tavlama benzetimi, diğeri tabu arama üzerine inşa edilmiş bu algoritmalar ilk defa U-şekilli bir montaj hattı dengeleme problemine uygulanmaktadır. Bunun yanında, bu iki melez algoritma, performans karşılaştırma amacıyla, ilk defa bir problemin çözümünde beraberce kullanılmıştır. Melez algoritmaların performansları, literatürden alınmış test problemleri üzerinde, birbirleriyle ve tavlama benzetimi ve tabu arama algoritmalarının saf versiyonlarıyla karşılaştırılmıştır. Hesaplama sonuçları, çekirdeğini tabu aramanın oluşturduğu melez algoritmanın diğerlerine üstünlük sağladığını göstermektedir.

Hybrid simulated annealing-tabu search algorithms for solving U-shaped type-2 assembly line balancing problems with workload smoothing objective

HIGHLIGHTS

- U-shaped type-2 simple assembly line balancing problem with a workload balancing objective is considered
- The problem is dealt with two different hybrid simulated annealing-tabu search algorithms
- The hybrid TSSA algorithm achieved very successful results for both objectives considered

Article Info
Research Article
Received: 31.03.2023
Accepted: 30.08.2023

DOI:
10.17341/gazimmfd.1274474

Keywords:
U-shaped assembly lines,
type-2 assembly line balancing,
simulated annealing,
tabu search,
hybrid meta-heuristics

ABSTRACT

With more businesses adopting the just-in-time production philosophy in recent years, the popularity of U-shaped assembly lines has increased. As a result, research into the methods for solving U-shaped assembly line balancing problems is expanding. In this study, two different hybrid simulated annealing-tabu search algorithms, which have been previously developed in the literature and have been shown to be effective on different problems, are used to solve the type-2 U-shaped simple assembly line balancing problem, which considers workload smoothing as a secondary objective. These algorithms, one based on simulated annealing and the other on tabu search, are applied for the first time to an assembly line balancing problem. In addition, these two hybrid algorithms are used together for the first time to solve a problem with the aim of performance comparison. The performances of hybrid algorithms are compared with each other and with pure versions of simulated annealing and tabu search algorithms on test problems taken from the literature. The computational results show that the hybrid algorithm, the core of which is tabu search, is superior to the others.

*Sorumlu Yazar/Yazarlar / Corresponding Author/Authors : *marikan@gazi.edu.tr / Tel: +90 312 582 3808

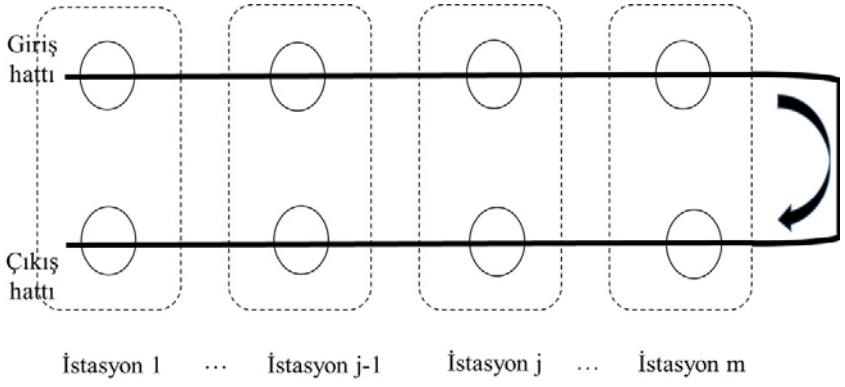
1. Giriş (Introduction)

Montaj hatları, benzer tip ürünleri yığın halinde üretmek için kurulmuş akış tipi üretim sistemleridir. Sıralı olarak dizilmiş, birbirlerine bir malzeme taşıma sistemiyle bağlanmış bir dizi istasyondan oluşurlar. Hat üzerindeki her istasyonda, ürünü oluşturan operasyonlar tekrarlı olarak sabit bir çevrim süresi içinde tamamlanacak şekilde yapılır. Çevrim süresi sonunda istasyonda işlemleri tamamlanan ürün bir sonraki istasyona iletilir [1]. Üretilen ürünün meydana geldiği operasyonlar, aralarındaki öncelik ilişkilerine uygun olarak, belli bir performans ölçütü dikkate alınarak istasyonlara atanır. Sözü edilen atama işi, montaj hattı dengeleme problemi olarak adlandırılır ve temel olarak ikiye ayrılabilir: Çevrim süresi sabitken istasyon sayısını en küçükleyen tip-1 ve istasyon sayısı sabitken çevrim süresini en küçükleyen tip-2. Yeni bir montaj hattı kurulmasına karar verildiğinde tip-1 problemin çözümüne ihtiyaç duyulurken, tip-2 problem var olan bir montaj hattı için üretim sürecinde ya da talep yapısında değişiklikler olduğunda ve üretim hızını arttırmak gerektiğinde çözülmelidir [2]. Problemin tipi, dikkate alınan birincil amaç fonksiyonunu tanımlar. Bunun yanında, problemin, görevler arasındaki öncelik ilişkilerine uyulması, bir istasyona atanan işlerin toplam süresinin çevrim süresini aşmaması ve her görevin sadece bir istasyona atanması gibi birtakım temel kısıtları vardır. Çevrim süresi, montaj hattının üretim hızını belirler ve ne kadar kısa olursa üretim hızı o kadar yüksek olur. Ayrıca, montaj hatlarında birincil amaçların yanında birtakım ikincil amaçların da dikkate alınması yaygın bir uygulamadır. En çok ilgilenilen ikincil amaç, iş yükü düzgünleştirme ya da istasyon iş yüklerinin dengelenmesidir. Bu sayede, işçiler arasında daha adaletli bir görev ataması hedeflenir.

Montaj hatları, üretilecek ürüne ya da hattın kurulacağı yerin fiziksel özelliklerine bağlı olarak farklı yerleşim şekillerinde tasarlanabilir. Düz hatlar ve U-şekilli hatlar en yaygın olarak kullanılan montaj hattı yerleşimleridir. Düz hatlarda istasyonlar düz bir hat üzerinde yerleştirilir, hattın girişi ve çıkışı karşılıklı uçlardadır. U-şekilli hatlarda ise, istasyonlar hattın girişi ve çıkışı aynı tarafta olacak şekilde, hattın çevresine konuşlandırılır. İşçiler ise hattın iç tarafında çalışırlar. Bu durumda, hattın giriş ve çıkış olarak ikiye ayrıldığı düşünülebilir (Şekil 1). İşçiler farklı operasyonları gerçekleştirebilecek şekilde eğitildikleri için farklı görevlere tahsis edilebilirler, bu da ortamdaki değişikliklere (makine arızaları, işçi devamsızlığı vb.) hızlı yanıt verilebilmesini sağlar. Gerektiğinde hat, işçi ekleme/çıkarma esnekliği ile yeni bir çevrim süresine göre yeniden dengelenebilir. Bu esneklik, tam zamanında üretim sistemlerinde şiddetle gereklidir, çünkü sonraki işlemler için üretilen parçaların talep oranları zaman zaman dinamik olarak değişmektedir [3]. Ayrıca, U-şekilli hatların ihtiyaç duyduğu istasyon sayısı en fazla geleneksel hatlar kadar olabilir [4] ve görevleri istasyonlara atarken düz hatlara göre daha fazla seçenek olduğundan istasyon iş yüklerinin daha dengeli bir şekilde dağıtılması mümkündür.

Bu çalışmada, iş yükü dengelemenin ikincil amaç olarak dikkate alındığı tek modelli U-şekilli tip-2 montaj hattı dengeleme problemi (BUMHDP-2) ile ilgilenilmiştir. U-şekilli montaj hatları ilk olarak Miltenburg ve Wijngaard [5] tarafından tanıtılmıştır. O zamandan beri, tek modelli basit U-şekilli montaj hattı dengeleme problemleri ve bunların daha karmaşık versiyonları üzerine birçok çalışma yapılmıştır. Literatür incelendiğinde, tip-2 problemi ele alan çalışmaların daha az olduğu görülmektedir. Scholl ve Klein [3], tip-1 basit düz montaj hattı dengeleme problemi (BMHDP-1) için dal-sınır yöntemine dayalı olarak geliştirdikleri SALOME-1'i ULINO adıyla tip-2 U-şekilli montaj hattı dengeleme problemine uyarlamışlardır. Çevrim süresi ve istasyon sayısı hedefinin de dikkate alındığı U-şekilli basit montaj hattı dengeleme problemi için Gökçen ve Ağpak [6] hedef programlama, Kara vd. [7] 0-1 bulanık hedef programlama yaklaşımları geliştirmişlerdir. Jonnalagedda ve Dabade [8], genetik algoritmaların tip-2 U-şekilli montaj hattı dengeleme problemine uygulanabilirliğini göstermişler ve algoritmanın literatürden alınan küçük ve orta ölçekli problemler üzerindeki performansını raporlamışlardır. Şahin ve Kellegöz [9], tip-2 BUMHDP için bir karma tam sayılı programlama modeli geliştirmiş, daha sonra da çözüm için bir gruplandırma genetik algoritması ve bir tavlama benzetimi algoritması önermişlerdir. Ayrıca literatürden bir parçacık sürü optimizasyonu (PSO) algoritmasını probleme uyarlamışlar ve tüm bu yöntemlerin performanslarını literatürden alınan test problemleri üzerinde karşılaştırmışlardır. Li vd. [10], BUMHDP-2 için çoklu kurallara ve bir tamsayı programlama modeline dayanan yeni bir sezgisel yaklaşım önermişlerdir. Li vd. [11], tip-1 BUMHDP'yi çözmek için güçlendirilmiş bir ışın arama (BS) yöntemi ve tip-2 BUMHDP'yi çözmek için yinelemeli bir ışın arama (IBS) yöntemi geliştirmişlerdir. Hesaplama sonuçları, önerilen yöntemlerin BUMHDP-1 ve BUMHDP-2 için en son teknolojiye sahip yöntemlerden daha iyi performans verdiğini göstermektedir.

Bu çalışmaların yanında, birçok araştırmacı problemin daha karmaşık versiyonlarına odaklanmıştır. Tip-2 U şekilli montaj hattı dengeleme probleminin Nakade vd. [12] stokastik görev zamanlı, Şahin ve Kellegöz [13] sıra bağımlı hazırlık zamanlı, Zhang vd. [14] işçi atamalı, Li vd. [15] robotik, Wang vd. [16] karışık akışlı, Pınarbaşı [17] atama kısıtlamalı türlerini ele almışlardır. Problem NP-zor bir yapıya sahip olduğundan çözüm için meta-sezgisel tekniklere sıkça başvurulmaktadır. U-şekilli tip-2 problemin ve çeşitli versiyonlarının ele alındığı çalışmalarda geliştirilen meta-sezgisel teknikler arasında genetik algoritmalar [8, 9, 13, 16], tavlama benzetimi [9, 13], tabu arama [18], göçmen kuşlar algoritması [14, 15] olduğu görülmektedir.



Şekil 1. U-şekilli hat örneği (A U-shaped line)

Meta-sezgisel yöntemlerin probleme uygun bir şekilde tasarlandıklarında optimale yakın sonuçlar verdikleri bilinmekle birlikte, birden fazla yöntemin avantajlarını birleştiren melez algoritmalar oluşturarak performanslarını arttırmak mümkündür. Literatürde, montaj hattı dengeleme problemleri de dahil olmak üzere, birçok problemin çözümü için farklı meta-sezgisel teknikler melezlenmiştir.

Literatürde düz hatlarda montaj hattı dengeleme problemlerinin çeşitli versiyonlarının çözümü için genetik algoritmaların yerel arama [19-21], dinamik programlama [22], değişken komşuluk arama [23], tabu arama [24, 25], tavlama benzetimi [23]; parçacık sürü optimizasyonunun tavlama benzetimi[26, 27], değişken komşuluk arama [28]; bal arısı evlilik optimizasyonunun tavlama benzetimi [29]; tavlama benzetiminin tabu arama [30], yerel arama [31]; yapay elektrik alan algoritmasının tavlama benzetimi [32]; değişken komşuluk arama algoritmasının da tavlama benzetimi [23, 33] ile melezlendiği çalışmalar bulunmaktadır. U-şekilli hatlarda ise Suwannarongsri ve Puangdownreong [34], tabu arama ve genetik algoritmanın melezlendiği bir yaklaşım, Nejad vd. [35] gruplama evrim stratejisine dayanan melez bir çözüm yöntemi, Khorram vd. [36] melez değişken komşuluk arama-yerel arama ve melez tavlama benzetimi-yerel arama algoritmaları önermişlerdir.

Bu çalışmada, iş yükü dengelemeli tip-2 U-şekilli montaj hattı dengeleme problemi, farklı yapıda, iki adet melez tavlama benzetimi-tabu arama algoritmasıyla çözülmüştür. Bizim bildiğimiz kadarıyla, ilgilenilen problemin çözümü için melez tavlama benzetimi-tabu arama algoritmaları ilk defa kullanılmaktadır. Daha önce literatürde önerilmiş olan bu algoritmaların birinin ana iskeletini tavlama benzetimi (TB) (Zolfaghari and Liang [37]) oluşturmaktadır. Algoritma, tavlama benzetimini bir yasaklı çözümler listesiyle destekleyerek arama performansını arttırmayı hedefler. Diğerinin ana iskeleti ise tabu arama (TA) (Zhang vd. [38]) üzerine inşa edilmiştir ve belli uzunlukta bir elit çözümler listesini tutarak, aramayı bu elit çözümler etrafında yoğunlaştırır. Elit çözümler, tavlama benzetiminde olduğu gibi, sıcaklık parametresine bağlı olarak hesaplanan bir olasılık değeri dikkate alınarak belirlenir. Bu melez algoritmaların önemi, yöntemlerin güçlü yönlerini birleştirerek daha etkin algoritmalar ortaya çıkarılmasından ileri gelmektedir. Melez algoritmalar ile ilgili daha fazla ayrıntı Bölüm 3'te verilmiştir.

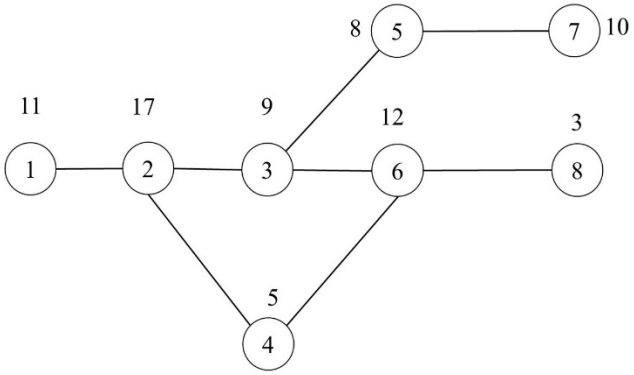
Algoritmaların performansı literatürden alınmış problemler üzerinde test edilmiş, elde edilen sonuçlar birbirleriyle ve algoritmaların temel sürümlerinden elde edilenlerle karşılaştırılmıştır. Ayrıca, çalışmada uygulanan farklı yapıdaki melez tabu arama-tavlama benzetimi algoritmaları problem özelliklerine uygun detaylar katılarak her probleme uygulanabilirler. Bu nedenle, bireysel meta-sezgisel tekniklerde olduğu gibi, performanslarının NP-zor problemler üzerinde karşılaştırılması potansiyel uygulayıcıları için ilgi çekici olabilir. Ayrıca, iş yükü düzgünleştirme montaj hattı dengeleme problemlerinde işçiler arasında adaletli görev atamasını gözeten önemli amaçlardan biridir. Daha önce, bu çalışmada ele alınan haliyle, iş yükü düzgünleştirme, düz hatlarda tip-2 basit montaj hattı dengeleme problemlerinde dikkate alınmış olmasına rağmen (Arıkan [39], [40]), yazarların bildiği kadarıyla, U-şekilli tip-2 basit montaj hattı dengeleme probleminde, Arıkan [18] ile birlikte, ilk kez ikincil bir amaç olarak kullanılmaktadır. Yani, dikkate alınan test problemleri için ikincil amaca ait bir sonuç seti literatürde bulunmamaktadır. Elde edilen iş yükü düzgünleştirme değerleri, aynı problemle uğraşan araştırmacılar için bir karşılaştırma imkânı sağlayabilir.

Çalışmanın ikinci bölümünde, problem tanımlanmıştır. Üçüncü bölümde, kullanılan yöntemler ve ele alınan probleme nasıl uygulandıkları anlatılmıştır. Dördüncü bölümde, yapılan deneysel çalışmanın sonuçları tartışılmıştır. Son bölümde ise çalışmanın özeti ve ileriki çalışmalar için öneriler yer almaktadır.

2. Problemin Tanımı (Problem Description)

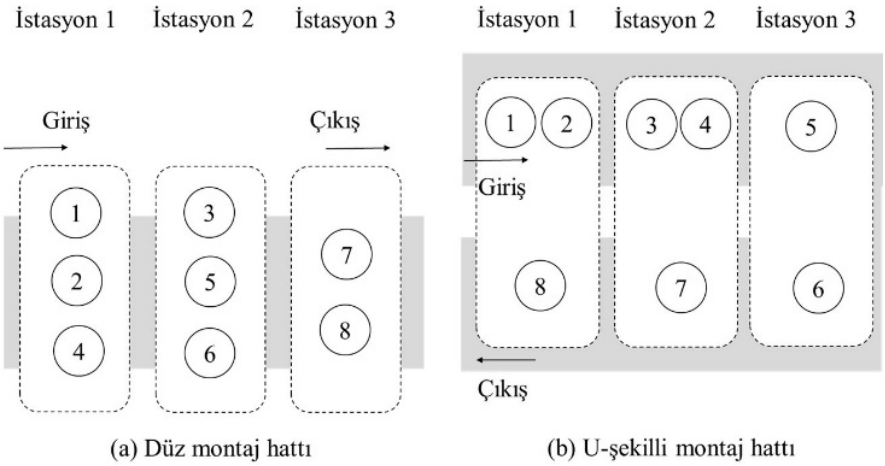
Tam zamanında üretim felsefesi, işletmelere sağladığı esneklik, israfın ve stokların azaltılması, daha iyi üretim kalitesi ve daha kısa ürün teslim zamanları gibi avantajları sayesinde imalat endüstrisinde giderek daha fazla kabul görmeye başlamıştır. Bu felsefeyi takip eden birçok firmada, klasik düz montaj hatları yerini söz konusu avantajlardan yararlanılmasını kolaylaştıran U-şekilli montaj hatlarına bırakmaktadır. U-şekilli montaj hattı dengeleme probleminin düz hatlı problemden en önemli farkı, görevler istasyonlara atanırken öncelik ilişkilerinin dikkate alınma şeklidir. Düz hatlarda, bir görevin bir istasyona atanabilmesi için tüm öncüllerinin mevcut istasyona ya da daha öncekilere atanmış olması gerekir. U-şekilli hatlarda ise bir görevin bir istasyona atanabilmesi için tüm öncülleri ya da tüm ardılları mevcut ya da önceki istasyonlardan birine atanmış olmalıdır. Şekil 2'de 8 görevden oluşan bir ürünün öncelik diyagramı görülmektedir. Düğümlerin içindeki sayılar görev numaralarını, dışındakiler görev zamanlarını ve oklar da görevler arasındaki öncelik ilişkilerini göstermektedir. Şekil 3a ve Şekil 3b, sırasıyla, 3 istasyona sahip düz hatlar ve U-şekilli hatlar için uygun görev atamalarını göstermektedir. Şekil 3a'da, mevcut ya da önceki istasyonlara, ilgili her bir görevin öncülleri atanmışken, 3b'de, ya öncülleri ya da ardılları atanmıştır.

Tip-2 U-şekilli montaj hattı dengeleme problemi, görevler arasındaki öncelik kurallarını ihlal etmeden çevrim süresini minimize etmek, başka bir deyişle üretim oranını maksimize etmek için görevlerin önceden belirlenmiş sayıda U-şeklindeki istasyonlara atanması olarak tanımlanabilir. Ele alınan problemin ikincil amacı, istasyonlar arası iş yükü düzgünlüğünü sağlamaktır. Literatürde iş yükü düzgünlüğü için tanımlanmış farklı fonksiyonlar [41, 42] bulunmaktadır ancak, bu çalışmada iş istasyonu yüklerinin ortalama iş yükünden mutlak farklarının toplamı (Eş. 3) kullanılmıştır (Rachamadugu ve Talbot [41]).



Şekil 2. Bowman (1960) probleminin öncelik diyagramı (Scholl [43]) (Precedence diagram of Bowman (1960) (Scholl [43]))

Ele alınan Tip-2 U-şekilli montaj hattı dengeleme probleminin varsayımları ve kısıtları şu şekildedir (Şahin ve Kellegöz, [9]): Tek bir ürün üretilir, her görev tek bir işçi tarafından yapılmalıdır, toplam istasyon sayısı sabittir ve sürecin başında bilinmektedir, görevler bölünemez ve herhangi bir işçi aynı anda sadece bir iş yapabilir, her istasyonda sadece bir işçi vardır, görev süreleri belirlidir ve sürecin başında bilinmektedir, iş parçalarının istasyonlar arası taşıma süreleri ve işçilerin istasyonlarda yürüme süreleri ihmal edilmiştir, öncelik kısıtları dışında herhangi bir atama kısıtlaması bulunmamaktadır ve gerekli tüm ekipman ve araç gereçler her istasyonda mevcuttur.



(a) Düz montaj hattı (b) U-şekilli montaj hattı

Şekil 3. Düz ve U-şekilli hat yerleşimleri için olurlu görev atamaları (Feasible task assignments for straight and U-shaped layouts)

Problem aşağıda matematiksel olarak tanımlanmıştır:

$$WSt_k = \sum_{i \in WS_k} t_i \ \ (k = 1, 2, \dots, m) \quad (1)$$

$$CT = \max_{k=1,2,\dots,m} \{WSt_k\} \quad (2)$$

$$Totdev = \sum_{k=1}^{m} \left| WSt_k - \frac{\sum_{i=1}^{n} t_i}{m} \right| \quad (3)$$

Diğer yerine getirilmesi gereken şartlar aşağıdaki gibidir:

$$\cup_{k=1}^{m} WS_k = E \quad (4)$$

$$WS_k \cap \ WS_l = \emptyset \ (k = 1, 2, \dots, m; l = 1, 2, \dots, m; k \neq l) \quad (5)$$

Her *j* görevi için:

$$Eğer\ (i,j) \in P, i \in WS_k, j \in WS_l\ ise\ k \leq l\ \forall i\ için;\ \vee$$
$$Eğer\ (j,r) \in P, j \in WS_l, r \in WS_h\ ise\ h \leq l\ \forall r\ için \quad (6)$$

$E = \{i | i = 1,2, \cdots, n\}$ ve $P = \{(i,j) | \text{i görevi j görevi başlamadan önce tamamlanmalı}\}$, sırasıyla, görevler kümesi ve öncelik kısıtları kümesi olsun. BUMHDP-2, *E* kümesinin *m* adet alt kümesini oluşturmayı hedeflemektedir. Bu alt kümeler istasyonlara atanan görevleri (WS_1, WS_2,…, WS_m) içerir. İstasyonların iş yükleri (WSt_k), Eş. 1 ile bulunur. Ele alınan BUMHDP-2 probleminin amacı, ilk önce, Eş. 2 ile belirlenen ve hattaki maksimum istasyon yüküne (WSt_{max}) eşit olan çevrim süresini (*CT*), sonrasında ise istasyon iş yüklerinin ortalama iş yükünden mutlak farklarının toplamını (*totdev*) minimize etmektir. İstasyon iş yüklerinin ortalama iş yükünden mutlak farklarının toplamı, Eş. 3 kullanılarak hesaplanır. Eş. 4 ve 5, her görevin tam olarak bir istasyona atanmasını garanti eder. Eş. 6, U-şekilli montaj hatları için öncelik kısıtlamalarının karşılanmasını sağlar. Sonuç olarak, çevrim süresi en aza indirilirken ya da başka bir deyişle üretim hızı en üst düzeye çıkarılırken en dengeli görev dağılımı elde edilir. Ele alınan problem NP-zor olduğundan, problemin çözümü için melez tavlama benzetimi-tabu arama algoritmaları kullanılmıştır. Sonraki bölümde tavlama benzetimi ve tabu arama yöntemleri hakkında bilgi verilmiş, melez algoritmalar ve uygulanmaları sırasında alınan kararlar açıklanmıştır.

3. Melez Tavlama Benzetimi-Tabu Arama Algoritmaları
(Hybrid Simulated Annealing-Tabu Search Algorithms)

Tavlama Benzetimi ve Tabu Arama, NP-zor problemlerin çözümünde yaygın olarak kullanılan meta-sezgisel arama yöntemleridir. İki yöntem de yerel optimumlara yakalanmamak için farklı kurallar kullanır. Bu yöntemlerin her biri sahip oldukları avantajların yanında, dezavantajlara da sahiptir ve arama performanslarını arttırmak için avantajlarını bir araya getiren melez algoritmalar üretilmiştir. Bu çalışmada, ele alınan problem, birinin çekirdeğini tavlama benzetiminin diğerininkini tabu aramanın oluşturduğu iki melez algoritma ile çözülmüştür. Tavlama benzetimi üzerine kurulmuş melez algoritma Zolfaghari ve Liang [37], tabu arama üzerine kurulmuş olan Zhang vd. [38] tarafından önerilmiştir.

3.1. Tavlama Benzetimi (Simulated Annealing)

Tavlama Benzetimi (TB), Kirkpatrick vd. [44] tarafından geliştirilmiş, katıların fiziksel tavlama sürecini taklit eden olasılıklı bir arama tekniğidir. Tavlama, katıların çok yüksek bir sıcaklık değerinden başlayarak düzgün kristal bir yapı elde edilene kadar yavaş yavaş soğutulduğu ısıl bir süreçtir. Bu sürecin kombinatoryal optimizasyon problemlerine uygulanmasında, her sıcaklık değerinde (*T*), mevcut çözümün (*S*) ve mevcut çözümden rassal olarak ulaşılan bir komşu çözümün (*S'*) amaç fonksiyonu değerleri karşılaştırılır. Mevcut bir çözümden komşu bir çözüme geçiş çözümün belli bir niteliğinin değiştirildiği bir hareketle gerçekleştirilir. Eğer komşu çözümün amaç fonksiyon değeri daha iyiyse, mevcut çözüm olarak kaydedilir. Daha kötüyse, komşu çözümün hala mevcut çözüm olarak kabul edilme ihtimali vardır. Amaç fonksiyonunun minimize edildiği durum için, söz konusu çözümlerin amaç fonksiyonları arasındaki farka ($\Delta f = f(S) - f(S')$) ve güncel sıcaklık değerine bağlı olarak hesaplanan ve metropolis kriteri adı verilen olasılık değeri ($exp^{-\Delta f/T}$), tekdüze dağılıma göre rassal olarak (0, 1) arasında üretilmiş bir olasılık değerinden büyükse komşu çözüm mevcut çözüm olarak kaydedilir ve aramaya yeni mevcut çözümden devam edilir. Aksi takdirde eski mevcut çözümün komşularından bir diğeri rassal olarak seçilir ve değerlendirilir. Kötüleşen çözümlerin kabul edilmesi, aramanın yerel optimumlardan kurtulmasına olanak sağlar. Sıcaklık değeri (*T*) yüksek olduğunda kötü çözümlerin kabul edilme olasılığı daha fazlayken sıcaklık azaldığında bu olasılık da azalır. Her sıcaklıkta belli sayıda komşu çözüm incelendikten sonra sıcaklık değeri belli bir fonksiyona göre düşürülür. Sıcaklık belli bir sınır değerinin altına düştüğünde de arama sonlandırılır. Tavlama benzetimi, her sıcaklıkta (iterasyonda) mevcut çözümün komşularının bir alt kümesini rassal olarak araştırır, iyi çözümlere ulaşmak için sıcaklık parametresinin çok ağır bir şekilde azaltılmasına ihtiyaç vardır. Tavlama Benzetiminin diğer bir dezavantajı, yakın bir geçmişte incelediği çözümleri, tekrar incelemesini engelleyen bir mekanizmaya sahip olmaması ve daha önce ziyaret ettiği çözümlere dönmesinin mümkün

olmasıdır. Bu nedenlerle, en iyi çözüme ulaşmak için çok fazla iterasyona ve hesaplama zamanına ihtiyaç vardır.

3.2. Tabu Arama (Tabu Search)

Tabu Arama (TA), kombinatoryal optimizasyon problemlerinin çözümünde çok etkili olduğu kanıtlanmış bir başka meta-sezgisel arama yöntemidir [45, 46]. TA'nın TB'ye göre en temel farkı, arama sürecinde topladığı geçmiş kayıtlardan faydalanmasıdır. Yöntem, bir başlangıç çözümünden başlayan ve daha iyi çözümler bulmaya çalışan, komşuluk arama tekniğine dayalı yinelemeli bir prosedürdür. TS, deterministik bir yöntemdir ve arama sırasında mevcut çözümün tüm komşu çözümleri arasından, iyileştirici olmasa bile en iyi komşuyu seçer ve mevcut çözüm olarak kaydeder. Bunun yanında, tabu listesi olarak adlandırılan bir listeye yasaklanmış hareketleri kaydeder ve listede kayıtlı hareketleri tekrar yapmaktan kaçınır. Bu özellikleri sayesinde de yerel optimumlara takılmaktan kurtulabilir. Tabu aramada kullanılan tabu listesi kısa süreli hafıza ile ilişkilendirilir ve tabu listesindeki hareketler tabu süresi olarak adlandırılan iterasyon sayısı boyunca gerçekleştirilmezler. Böylece, tabu süresine eşit veya daha az uzunluktaki çevrimlerin oluşması önlenir [47]. Ancak, tabu aramanın bir dezavantajı, daha önce ziyaret edilen bir çözüme ulaşırsa, tabu olan bir hareketle ulaşılan bir komşusu olmadığı sürece aynı yolu izleyerek döngüye girmesidir [37]. Tekniğin diğer bir özelliği, tabu listesindeki hareketlerin ne zaman yapılabileceğine karar veren tabu yıkma kriteridir ve uygun bir şekilde kullanıldığı takdirde algoritmanın performansını arttırabilir. Bazı uygulamalarda kısa dönemli hafıza yapısı kaliteli sonuçlar elde etmek için yeterli olsa da TA'ya, uzun dönem hafıza yapısı ve ilgili stratejiler dahil edildiğinde çok daha güçlü olmaktadır. Uzun dönemli hafıza fonksiyonları aramanın bölgesel olarak kuvvetlendirilmesi ve global olarak çeşitlendirilmesi amacıyla kullanılır (Glover ve Laguna, [48]).

3.3. Melez Tavlama Benzetimi-Tabu Arama (SATS) Algoritması (Hybrid Simulated Annealing-Tabu Search (SATS) Algorithm)

Tabu arama, yakın zamanda gerçekleştirilen hareketlerin kısa süreli bir hafızası olan tabu listesini kullanır, ancak deterministik bir yapıya sahiptir ve döngüden kaçınamaz. Öte yandan, SA stokastik bir özelliğe sahiptir ve bu nedenle döngüsel doğadan kaçınabilir ancak çözümün gelişme hızı çok yavaştır. Ayrıca, daha önce gerçekleştirilen hareketlerin kaydını tutmaması nedeniyle, aramanın son zamanlarda ziyaret edilen bir çözüme dönmesi her zaman mümkündür (Swarnkar ve Tiwari [49]). Zolfaghari ve Liang [37], bu dezavantajları ortadan kaldırmak için tavlama benzetiminin tabu listesiyle desteklendiği bir melez algoritma önermişlerdir. Bu sayede, aramanın daha fazla çözümü kapsaması ve tekrar ziyaret sayısının azaltılması amaçlanmaktadır. Algoritmanın çekirdeğini tavlama benzetimi oluşturmaktadır ve uygulanması için alınan kararlar aşağıda verilmiştir.

3.3.1. Çözüm temsili ve başlangıç çözümü (Solution representation and initial solution)

Bu çalışmada istasyona yönelik bir çözüm temsili kullanılmıştır. Buna göre bir çözüm, *n* (görev sayısı) elemanlı tek boyutlu bir dizi ile temsil edilir ve karşılık gelen görevin atandığı istasyon numarasını saklar. Şekil 2'deki öncelik diyagramı dikkate alındığında, 3 iş istasyonlu U şekilli bir montaj hattında, <1,1,2,2,3,3,2,1> çözümü, 1, 2 ve 8 görevlerinin 1 no'lu istasyona, 3, 4 ve 7 görevlerinin 2 no'lu istasyona, 5 ve 6 görevlerinin ise 3 no'lu istasyona atandığını gösterir. Her görevin atandığı istasyonlar bilindiği için her bir istasyonun iş yükü kolayca hesaplanabilir ve maksimum istasyon iş yükü, hattın çevrim süresi olarak belirlenir. Ortalama iş yükünün istasyon yüklerinden mutlak farklarının toplamı da iş yükü dengesizliğini verir. Başlangıç çözümü, çözümün olurluluğu garanti edilecek şekilde rastgele bulunur.

3.3.2. Amaç fonksiyonu (Objective function)

Çevrim süresinin minimizasyonu ve iş yükü dengesizliğinin (iş istasyonu yükleri ile ortalama iş yükü arasındaki mutlak farkların toplamı) minimizasyonu olmak üzere iki amaç dikkate alınmıştır. İlk olarak, amaçlar oransal ve Eş. 7'deki gibi bütünleşik olarak ifade edilmiştir.

$$Minimize\ \alpha(CT/CT_{min}) + \beta(totdev/t_{sum}) \quad (7)$$

Burada, *CT*, hattın çevrim süresi, CT_{min} mümkün olan minimum çevrim süresidir (teorik alt sınır). Bütünleşik amaç fonksiyonunun ilk parçası çevrim süresine, ikinci parçası iş yükü dengesizliğine karşılık gelmektedir. α ve β amaçların öncelik katsayılarını temsil eder. Ön denemeler, α=500 ve β=100 katsayıları için birincil amacın ikincil amaca göre önceliğinin iyi bir şekilde temsil edildiğini göstermektedir.

3.3.3. Komşu üretme mekanizması (Neighborhood generation mechanism)

Mevcut çözümden komşu bir çözüm üretmek için kaydırma ve yer değiştirme hareketleri kullanılır. Kaydırma hareketi, bir görevin bir istasyondan farklı bir istasyona taşınmasını ifade eder, yer değiştirme hareketi ise farklı istasyonlardaki iki görevin değiştirilmesine karşılık gelir. Yapılacak hareket rastgele belirlenir. Yer değiştirme ve kaydırma hareketlerinin olasılıkları, sırasıyla, 0,5 ve 0,5 olarak alınmıştır. Hareketin yapılacağı kaynak istasyon, maksimum iş yüküne sahip istasyondur. Mevcut çözüm her güncellendiğinde, çözümün olurluluğu korunacak şekilde hangi görevlerin giriş alt hattına, hangilerinin çıkış alt hattına atanması gerektiği ve olurluluk bozulmadan her görevin atanabileceği en erken ve en geç istasyonlar belirlenir. Aşağıdaki koşullar, her görevin en erken ve en geç istasyonlarını belirlemek için kullanılır (Şahin ve Kellegöz [9]).

Koşul 1 : Giriş alt hattındaki bir görev, giriş hattındaki öncüllerinin atandığı istasyonlardan daha erken bir istasyona atanamaz.
Koşul 2 : Giriş alt hattındaki bir görev, giriş hattındaki ardıllarının atandığı istasyonlardan sonraki bir istasyona atanamaz.
Koşul 3 : Çıkış alt hattındaki bir görev, çıkış hattındaki ardıllarının atandığı istasyonlardan daha önceki bir istasyona atanamaz.
Koşul 4 : Çıkış alt hattındaki bir görev, çıkış hattındaki öncüllerinin atandığı istasyonlardan sonraki bir istasyona atanamaz.

3.3.4. Komşuluk arama mekanizması (Neighborhood search mechanism)

Tavlama benzetimi tabanlı melez yöntemde, mevcut çözümün her bir farklı komşusunu elde etmek için yapılacak hareket rassal olarak belirlenir. Komşu çözüm kabul edilmezse, başka bir komşu için yeniden bir hareket belirlenir. Eğer kabul edilir ve mevcut çözüm olarak atanırsa, arama yeni mevcut çözümden devam ettirilir. Dolayısıyla, mevcut çözümün tüm komşularını araştırmak mümkün olmaz. İlk önce, maksimum yüklü istasyondan rastgele bir görev belirlenir. Eğer kaydırma hareketi yapılacaksa, seçilen görevin aktarılacağı istasyon, görevin atanabileceği en erken ve en geç istasyonlar arasından rassal olarak seçilir. Eğer yer değiştirme hareketi gerçekleştirilecekse, maksimum yüklü istasyondaki her görev için çözümün olurluluğu bozulmadan yer değiştirebileceği bir görevler listesi elde edilir. Bu liste, en erken/en geç istasyon ve görevin atandığı alt hat bilgilerinin birlikte kullanılmasıyla oluşturulur ve listeden rassal olarak belirlenen görevle daha önce maksimum yüklü istasyondan seçilen görevin yerleri değiştirilir.

Bazı durumlarda, çevrim süresinin teorik alt sınırı ($CT_{min} = max\{t_{max}, \lceil \sum_{i=1}^{n} t_i/m \rceil\}$) t_{max}'a (görev sürelerinin maksimumu) eşittir. Böyle bir durumda, maksimum yüklü istasyon tek bir maksimum zamanlı görev içerecektir, bu nedenle maksimum yüklü istasyonun hareketin kaynağı olması çok anlamlı olmayacaktır. Bu

takdirde, ortalama iş yükünden ($\lceil \sum_{i=1}^{n} t_i/m \rceil$) daha fazla yüke sahip bir istasyon kaynak istasyon olarak seçilir ve bu istasyondaki görevler kullanılarak hareket gerçekleştirilir.

Her sıcaklıkta araştırılacak komşu çözüm sayısı belirlenir ve o sayıda çözüm incelendikten sonra sıcaklık güncellenir. Bu çalışmada her sıcaklıkta incelenecek çözüm sayısı, görev sayısının %50'sine en yakın tam sayı (round(n*0,5)) olacak şekilde tespit edilmiştir.

3.3.5. Tabu listesi ve tabu süreleri (Tabu list and tabu tenures)

Bir hareketin tabu durumunu izlemek için, tabu durumu başlangıç iterasyonlarını saklayan $n \times m$ boyutlu bir tabu listesi kullanılır. Tabu süresi (tt), görev sayısının karekköküne en yakın tam sayıya ($round\sqrt{n}$) eşit olacak şekilde ayarlanır. Bir hareket gerçekleştirildikten sonra, hareket türüne göre, ilgili görev ve istasyonların tabu durumunun başlangıç iterasyonu aşağıdaki gibi güncellenir.

1. Kaydırma hareketi: Kaynak istasyonundaki (*source_st*) *task1* görevi *st* istasyonuna kaydırılırsa
   $Tabustart[task1, source\_st] = mevcut\ iterasyon$
2. Yer değiştirme hareketi: *source st* istasyonundaki *task1* görevi *st* istasyonundaki *task2* görevi ile yer değiştirirse
   $Tabustart[task1, source\_st] = mevcut\ iterasyon$
   $Tabustart[task2, st] = mevcut\ iterasyon$

Ayrıca, hareket türüne göre aşağıdaki koşullar sağlanıyorsa, bir hareket tabu olarak sınıflandırılır:

1. Kaydırma hareketi: *source st* istasyonundaki *task1* görevi *st* istasyonuna kaydırılırsa
   $mevcut\ iterasyon \leq Tabustart[task1, st] + tt$
2. Yer değiştirme hareketi: *source_st* istasyonundaki *task1* görevi *st* istasyonundaki *task2* görevi ile yer değiştirirse
   mevcut iterasyon$\leq Tabustart[task1, st] + tt$
   mevcut iterasyon$\leq Tabustart[task2, source\_st] + tt$

3.3.6. Tabu yıkma kriteri (Aspiration criterion)

Eğer komşu bir çözümün amaç fonksiyon değeri mevcut çözümünkinden daha iyi ise çözüm tabu olsa bile tabu durumu ihmal edilir.

3.3.7 Tavlama Planı (Cooling Schedule)

Tavlama planı, başlangıç sıcaklığı, bitiş sıcaklığı ve sıcaklık azaltma fonksiyonunun belirlenmesini içerir ve elde edilecek çözümün kalitesi üzerinde büyük etkisi vardır. Başlangıç sıcaklığının çok dikkatli bir şekilde belirlenmesi gerekir. Eğer aramaya çok yüksek bir sıcaklıktan başlanırsa algoritmanın koşum süresi gereksiz şekilde artar. Ayrıca, her problem örneği farklı bir başlangıç sıcaklığına ihtiyaç duyabilir. Bu çalışmada, Ben-Ameur [50] tarafından önerilen ve başlangıç sıcaklığını, kötüleşen çözümlerin kabul oranı belli bir olasılığa eşit olacak şekilde belirleyen bir algoritmadan yararlanılmıştır. Kötüleşen çözümlerin kabul oranı 90%, bitiş sıcaklığı (T_f) 2 olarak seçilmiştir. Sıcaklığın azaltılmasında ise Eş. 8'de verilen doğrusal çarpımsal soğutma fonksiyonu kullanılmıştır.

$$Temp_k = T_0/(1 + a * k) \quad (8)$$

Burada, k, iterasyon numarasını, $Temp_k$, k iterasyonunda kullanılan sıcaklığı, T_0, başlangıç sıcaklığını, a ise soğutma oranını göstermektedir.

3.3.8. SATS algoritmasının adımları (Steps of the SATS algorithm)

Algoritmanın adımları aşağıdaki gibidir.

Adım 0: Parametre başlangıç değerlerinin verilmesi (T_0, T_f , a), Tabu listesine başlangıç değerlerini ver.
Adım 1: Rassal bir başlangıç çözümü (S_{init}) bul, en iyi çözüm ve mevcut çözüm olarak kaydet ($S_{best}=S_{init}$, $f(S_{best})= f(S_{init})$, $S_{cur}=S_{init}$, $f(S_{cur})= f(S_{init})$); $Temp=T_0$, $iter$=1, her sıcaklıkta incelenecek komşu çözüm sayısını belirle.
Adım 2: Mevcut çözümdeki her bir görevin atanabileceği en erken/en geç istasyonları belirle; mevcut çözümden komşu çözümler üretmek için kullanılacak kaynak istasyonu belirle;
Adım 3: $Temp<T_f$ ise Adım 14'e git.
Adım 4: Mevcut çözümden komşu bir çözüm üretmek için yapılacak hareketi rassal olarak belirle. Hareket tipi yer değiştirme ise adım 9'a git.
Adım 5: Kaynak istasyondan bir görevi rassal olarak belirle, bu görevin çözümün olurluluğu bozulmadan atanabileceği en erken ve en geç istasyonlar arasından birini rassal olarak seç ve ilgili görevi ilgili istasyona atayarak komşu çözümü (S_{neigh}) oluştur.
Adım 6: Komşu çözüm tabu ise veya $f(S_{neigh})>=f(S_{cur})$ ise (tabu yıkma kriterini sağlamıyorsa) adım 5'e git.
Adım 7: Eğer $e^{-(f(S_{neigh})-f(S_{cur}))/Temp} \geq random(0,1)$ ise

- Mevcut çözümü güncelle
- Mevcut çözümdeki her bir görevin atanabileceği en erken/en geç istasyonları belirle
- Mevcut çözümden hareketin kaynağı olacak istasyonu belirle
- Eğer $f(S_{neigh})<f(S_{best})$ ise en iyi çözümü güncelle
- Tabu listesini güncelle

Aksi halde komşu çözümü eski haline getir (komşu çözüm belirleme sırasında istasyonu değiştirilen görevi eski istasyonuna ata)

Adım 8: Adım 13'e git.
Adım 9: Mevcut çözümde hareketin kaynağı olan istasyona atanmış görevlerin yer değiştirebileceği görevleri belirle.
Adım 10: Kaynak istasyondan bir görevi rassal olarak belirle, bu görevin çözümün olurluluğu bozulmadan, diğer istasyonlarda, yer değiştirebileceği bir görevi rassal olarak seç. İlgili görevlerin istasyonlarını değiştirerek komşu çözümü (S_{neigh}) oluştur.
Adım 11: Komşu çözüm tabu ise veya $f(S_{neigh})>=f(S_{cur})$ ise (tabu yıkma kriterini sağlamıyorsa) adım 10'a git.
Adım 12: Eğer $e^{-(f(S_{neigh})-f(S_{cur}))/Temp} \geq random(0,1)$ ise

- Mevcut çözümü güncelle
- Mevcut çözümdeki her bir görevin atanabileceği en erken/en geç istasyonları belirle
- Mevcut çözümden hareketin kaynağı olacak istasyonu belirle
- Eğer $f(S_{neigh})<f(S_{best})$ ise en iyi çözümü güncelle
- Tabu listesini güncelle

Aksi halde komşu çözümü eski haline getir (komşu çözüm belirleme sırasında istasyonları değiştirilen görevlere eski istasyonlarını ata).

Adım 13: Her sıcaklıkta incelenmesi gereken komşu çözüm sayısına ulaşıldı ise

- $iter=iter+1$
- $Temp = T_0/(1 + a * iter)$
- Adım 3'e git

Aksi halde Adım 4'e git.

Adım 14: En iyi çözümü (S_{best}, $f(S_{best})$) yazdır, DUR.

3.4. Melez Tabu Arama- Tavlama Benzetimi (TSSA) Algoritması (Hybrid Tabu Search-Simulated Annealing (TSSA) Algorithm)

Zhang vd. [38] melez algoritmalarında, tavlama benzetimini umut verici elit çözümleri bulmak için, tabu aramayı da bu elit çözümler etrafında aramayı yoğunlaştırmak için kullanmışlardır. Algoritmanın özü, TA yoğunlaştırma stratejisinin basit bir uygulamasıdır. Algoritma, arama sırasında bulduğu yeterince "iyi" çözümlerden oluşan *L* boyutlu bir elit çözümler listesi tutar. Bu çözümler, tavlama benzetiminin metropolis kriterine göre belirlenir. Seçilen elit çözümler, elit listenin en tepesine kaydedilir. Elit listenin mevcut ilk çözümü, TA algoritmasını sürdürmek için her zaman başlangıç çözümü olarak seçilir ve listeden silinir. Her yeni ve farklı başlangıç çözümü ile önceden belirlenmiş bir sayıda iterasyon gerçekleştirilir. Elit çözümler listesi, arama sırasında karşılaşılan yeni "iyi" çözümlerle güncellenir. Liste uzunluğunu sabit tutmak için de eski "iyi" çözümler listeden silinir. Metropolis kriterinde kullanılacak sıcaklık parametresinin başlangıç değeri (T_0), elit listedeki çözümlerin tükenmesi engellenecek şekilde verilmeli ve uygun bir şekilde güncellenmelidir. Algoritma, belli bir iterasyon sayısına ulaşıldığında sonlandırılır. Elit çözümlerin belirlenmesinde metropolis kriterinin kullanılması en iyi çözümler dışındaki çözümlerin de kabul edilmesine olanak tanır. Algoritmayı bu farklı çözümlerle yeniden başlatmak, aramayı çeşitlendirmeye yardımcı olurken, bu çözümlerin yeterince "iyi" olması da aramanın bu çözümler etrafında yoğunlaştırılmasını sağlar.

Kullanılan algoritmalar bazı ortak kararlar içermektedir. Komşuluk temsili ve başlangıç çözümü, amaç fonksiyonu, komşu üretme mekanizması, tabu listesi ve süreleri her iki melez algoritma için de aynıdır. Sadece, amaç fonksiyonunda çevrim süresi ve iş yükü düzgünleştirme amaçları için öncelik katsayıları α ve β, aralarındaki oransallık korunarak, sırasıyla, 5 ve 1 olarak alınmıştır. Algoritmanın uygulanmasında, SATS algoritmasından farklı olarak alınan kararlar ve ilgili parametreler için seçilen değerler aşağıda verilmiştir.

3.4.1. Komşuluk arama mekanizması (Neighborhood search mechanism)

Mevcut çözümden komşu bir çözüm üretmek için yapılacak hareket belirlendiğinde, aynı hareket tipi uygulanarak mevcut çözümün tüm komşuluğu incelenir. Kaydırma hareketinin yapılmasına karar verilirse, kaynak istasyondaki her görev, atanabileceği en erken ve en geç istasyonlar arasında, mevcut istasyonu hariç, sırayla her istasyona kaydırılır. Yer değiştirme hareketi seçildiğinde, kaynak istasyonuna atanan her bir görev, çözümün olurluluğu bozulmadan yer değiştirebileceği görevler listesindeki her görevle sırayla değiştirilerek, tüm komşuluk araştırılır.

3.4.2. Elit çözümleri kaydetme mekanizması (Elite solution saving mechanism)

Elit çözümleri bulmak için tavlama benzetimi tabanlı bir mekanizma kullanılmaktadır. Bu mekanizmaya göre, mevcut çözümün amaç fonksiyonu değeri ($f(S_{cur})$), o ana kadar bulunmuş en iyi çözüm değerinden ($f(S_{best})$) daha küçükse ($f(S_{cur}) < f(S_{best})$) ya da $exp^{-(f(S_{cur})-f(S_{best}))/Temp} > random(0,1)$ ise mevcut çözüm, elit listenin en tepesine kaydedilir. Algoritmanın özü (tabu arama), şimdiye kadar bulunmuş en iyi çözümde bir iyileşme olmadan önceden belirlenmiş bir iterasyon sayısı (*iterdiff*) kadar çalıştıktan sonra elit listenin en tepesinden başka bir çözümle tekrar başlatılır. TSSA'nın çalışması sırasında, bulunan yeni elit çözümler, elit çözümler listesine eklenir. Listenin boyutu, yani listede saklanan maksimum çözüm sayısı *elitesize* ile belirtilir. Liste dolduğunda, yeni bulunan çözümlere yer açmak için listedeki en eski çözüm silinir.

Başlangıç sıcaklığının da (T_0) algoritma ile elde edilen çözümlerin kalitesi üzerinde önemli bir etkisi vardır. Eğer başlangıç sıcaklığı çok düşükse, elit çözümlerin hızla tükenmesi nedeniyle algoritma beklenenden daha önce sonlandırılabilirken, sıcaklık çok yüksekse, "etkili" elit çözümlerin seçildiği garanti edilemez (Zhang vd. 2008). Sıcaklık, en iyi çözüm her güncellendiğinde, $Temp = f(S_b)/T_0$ fonksiyonuna göre yeniden hesaplanır. Yapılan ön denemeler sonucunda, problem boyutundan bağımsız olarak, T_0, *elitesize* ve *iterdiff*, sırasıyla, 10, 30 ve 5 olarak belirlenmiştir.

3.4.3. Tabu yıkma kriteri (Aspiration criterion)

Eğer komşu bir çözümün amaç fonksiyon değeri o ana kadar bulunmuş en iyi çözümden daha iyi ise, çözüm tabu olsa bile, tabu durumu ihmal edilir.

3.4.4. Durdurma koşulu (Termination criterion)

Algoritma, önceden belirlenen iterasyon sayısına (*iterlim*) ulaşıldığında veya elit listede çözüm kalmadığında sonlandırılır. Burada, *iterlim* problemin boyutuna bağlıdır ve (n×300) olarak belirlenir.

3.4.5. TSSA algoritmasının adımları (Steps of the TSSA algorithm)

Algoritmanın adımları aşağıdaki gibidir.

Adım 0: Parametre başlangıç değerlerinin verilmesi (T_0, *elitsize*, *iterdiff*, *iterlim*), tabu listesine başlangıç değerlerinin verilmesi.
Adım 1: Rassal bir başlangıç çözümü (S_{init}) bul, en iyi çözüm olarak kaydet ($S_{best}=S_{init}$, $f(S_{best})=f(S_{init})$); *totiter*=0; başlangıç çözümünü elit çözüm listesinin ilk sırasına kaydet (*elitçözümlistesi*[1]=S_{init}).
Adım 2: Elit listenin ilk sırasındaki çözümü mevcut çözüm olarak kaydet (S_{cur}= *elitçözümlistesi*[1]), iter=0;
Adım 3: Mevcut çözümdeki her bir görevin atanabileceği en erken/en geç istasyonları belirle; mevcut çözümden komşu çözümler üretmek için kullanılacak kaynak istasyonu belirle; *f(bestneighbour)*=1000, *iter*=*iter*+1, *totiter*=*totiter*+1;
Adım 4: Gerçekleştirilecek hareket tipini belirle. Seçilen hareket tipi yer değiştirme ise adım 8'e git.
Adım 5: Kaydırma hareketi ile mevcut çözümün tüm komşuluğunu araştır.

- Kaynak istasyonundaki her görev için aşağıdakileri tekrarla
  - Mevcut görevin, mevcut istasyonu hariç atanabileceği en erken ve en geç istasyonlar arasındaki her istasyon için aşağıdakileri tekrarla
    - Mevcut görevin istasyonunu değiştir (komşu çözüm (S') üret),
    - Yapılan hareketin tabu durumunu kontrol et
    - Eğer (tabu status=false) ∨ ($f(S')<f(S_{best})$) ise
    - Eğer $f(S')< f(bestneighbour)$ ise komşuluktaki en iyi çözümü güncelle ($S_{bestneigh}=S'$, $f(bestneighbour)=f(S')$)

Adım 6: Komşuluktaki en iyi çözüm elit bir çözüm ise elit listeyi güncelle:

- Eğer $exp^{-(f(bestneighbour)-f(S_{best}))/Temp} > random(0,1)$ ise $S_{bestneigh}$'ı elit listeye kaydet

Adım 7: Adım 10'a git
Adım 8: Yer değiştirme hareketi ile mevcut çözümün tüm komşuluğunu araştır.

- Kaynak istasyonundaki her görev için aşağıdakileri tekrarla
  - Mevcut görevin olurluluk bozulmadan diğer istasyonlardaki yer değiştirebileceği aday görevleri belirle ve her aday görev için aşağıdakileri tekrarla
    - Mevcut görevin istasyonuyla aday görevin istasyonlarını değiştir (komşu çözüm (S') üret),
    - Yapılan hareketin tabu durumunu kontrol et

- Eğer (tabu status=false) $\vee$ ($f(S') < f(S_{best})$) ise
- Eğer $f(S') < f(bestneighbour)$ ise komşuluktaki en iyi çözümü güncelle ($S_{bestneigh}=S'$, $f(bestneighbour)=f(S')$)

Adım 9: Komşuluktaki en iyi çözüm elit bir çözüm ise elit listeyi güncelle:

- Eğer $exp^{-(f(bestneighbour)-f(S_{best}))/Temp} > random(0,1)$ ise $S_{bestneigh}$ elit listeye kaydet

Adım 10: Eğer $f(bestneighbour) < f(S_{best})$ ise en iyi çözümü ve sıcaklığı güncelle ($S_{best}=S_{bestneigh}$, $f(S_{best})=f(bestneighbour)$, $Temp = f(S_{best})/T_0$), iter=0,

Adım 11: Komşuluktaki en iyi çözümü mevcut çözüm olarak kaydet ($S_{cur}=S_{bestneigh}$, $f(S_{cur})=f(bestneighbour)$)

Adım 12: Eğer $iter < iterdiff$ ise adım 3'e git

Adım 13: Eğer ($elitçözümlistesi[1]=\varnothing$) $\vee$ ($totiter > iterlim$) ise en iyi çözümü (S_{best}, $f(S_{best})$) yazdır, DUR, aksi halde adım 2'ye git.

4. Deneysel Çalışma (Experimental Study)

Melez tavlama benzetimi-tabu arama algoritmalarının performansı, https://assembly-line-balancing.de/salbp/benchmark-data-sets-1993/ adresinden alınmış 9 probleme ait 128 örnek üzerinde, her iki amaç için elde edilen sonuçların birbirleriyle ve tavlama benzetimi ve tabu arama algoritmalarının saf versiyonlarından elde edilenlerle karşılaştırılarak değerlendirilmiştir. Saf tabu arama algoritmasının sonuçları Arıkan (2022)'den alınmıştır. Saf tavlama benzetimi algoritmasında ise SATS algoritması için verilen tavlama planı kullanılmıştır. Çözülen problemler 29-111 arasında değişen görev sayılarına sahiptirler: Buxey (29, 8, 7, 14), Sawyer (30, 8, 7, 14), Lutz1 (32, 5, 8, 12), Gunther (35, 10, 6, 15), Kilbridge (45, 9, 3, 11), Tonge (70, 23, 3, 25), Arcus1 (83, 20, 3, 22), Lutz2 (89, 20, 9, 28), Arcus2 (111, 25, 3, 27). Parantez içindeki sayılar, sırasıyla, ilgili problemdeki görev, test örneği, test örneğinin minimum ve maksimum iş istasyonu sayılarını göstermektedir. Örneğin, 29 görevli Buxey problemi için, 7 ve 14 arasındaki istasyon sayılarıyla, toplam 8 adet test örneği çözdürülmüştür. Tüm algoritmalar TurboPascal programlama dilinde kodlanmıştır ve tüm denemeler 4,00 GB ön belleğe sahip Pentium çift çekirdekli 2,67 Ghz işlemcili kişisel bir bilgisayar üzerinde gerçekleştirilmiştir. Deneylerin tarafsızlığını sağlamak için algoritmalar, önceki bölümlerde bahsedilen parametre değerleri ile her test örneğinde farklı rassal sayılarla 10'ar kez çalıştırılmış ve her test probleminin tüm örnekleri üzerinden çözüm kalitesinin ortalaması alınmıştır. Yani her bir algoritma, toplamda (8+8+5+10+9+23+20+20+25) x 10 = 1280 kez çalıştırılmıştır.

Tablo 1'de melez ve saf algoritmaların ele alınan problem üzerindeki performansları karşılaştırılmıştır. Tablonun ilk iki sütununda problem

Tablo 1. Test problemlerinde elde edilen sonuçların karşılaştırılması (Comparison of results obtained in test problems)

Problem	n	Yöntem	best.c%dev	avg.c%dev	max.c%dev	MAD	Ort. CPU s
Buxey	29	SA	0,96	1,59	2,11	0,58	4,02
		TS	0,96	1,28	1,31	0,53	3,98
		SATS	0,96	1,28	1,69	0,55	4,27
		TSSA	*0,96*	*0,96*	*0,96*	*0,36*	4,23
Sawyer	30	SA	1,31	1,31	1,31	0,55	4,09
		TS	0,96	1,24	1,31	0,43	1,47
		SATS	0,96	1,28	1,31	0,50	4,29
		TSSA	*0,96*	*0,96*	*0,96*	*0,34*	4,65
Lutz1	32	SA	0,74	1,09	1,53	41,64	5,55
		TS	0,86	0,95	1,15	27,86	2,03
		SATS	0,67	0,91	1,15	36,82	5,90
		TSSA	*0,23*	*0,28*	*0,33*	*15,92*	5,23
Gunther	35	SA	0,23	0,49	1,21	1,31	7,51
		TS	0,23	0,23	0,23	0,84	3,77
		SATS	0,23	0,23	0,23	1,17	7,76
		TSSA	*0,00*	*0,00*	*0,00*	*0,72*	6,23
Kilbridge	45	SA	0,00	0,00	0,00	0,42	3,90
		TS	0,00	0,00	0,00	0,26	2,19
		SATS	0,00	0,00	0,00	0,39	6,50
		TSSA	*0,00*	*0,00*	*0,00*	*0,26*	10,42
Tonge	70	SA	0,14	0,39	0,89	1,41	10,50
		TS	0,16	0,22	0,28	0,71	7,42
		SATS	0,11	0,25	0,62	1,17	17,73
		TSSA	*0,00*	*0,00*	*0,00*	*0,41*	33,01
Arcus1	82	SA	0,52	0,90	1,37	49,50	9,63
		TS	0,41	0,54	0,69	26,06	9,98
		SATS	0,41	0,60	0,84	34,96	17,39
		TSSA	*0,03*	*0,04*	*0,05*	*3,31*	50,21
Lutz2	89	SA	0,00	0,11	0,28	0,35	14,08
		TS	0,00	0,14	0,28	0,35	7,92
		SATS	0,00	0,08	0,28	0,34	23,26
		TSSA	*0,00*	*0,00*	*0,00*	*0,34*	73,40
Arcus2	111	SA	0,09	0,17	0,32	14,90	101,82
		TS	0,05	0,07	0,09	5,93	31,76
		SATS	0,11	0,19	0,28	14,33	98,84
		TSSA	*0,01*	*0,02*	*0,02*	*1,36*	108,44
Average		SA	0,44	0,67	1,00	12,30	17,90
		TS	0,40	0,52	0,59	7,00	7,84
		SATS	0,38	0,54	0,71	10,03	20,66
		TSSA	*0,24*	*0,25*	*0,26*	*2,56*	32,87

isimleri ve boyutları verilmiştir. Diğer sütunlardaki bilgilerin açıklamaları, sırasıyla, aşağıdadır.

best.c%dev= çevrim zamanının alt sınır değerinden en iyi yüzde sapma, $((best.c - CT_{min})/CT_{min}) \times 100$ ile hesaplanır, burada CT_{min}çevrim zamanının teorik alt sınırı, *best.c* ilgili sezgiselle elde edilmiş en iyi çevrim zamanıdır.
avg.c%dev= çevrim zamanının alt sınır değerinden ortalama yüzde sapma, $((avg.c - CT_{min})/CT_{min}) \times 100$ ile hesaplanır, burada *avg.c* ilgili sezgiselle elde edilmiş ortalama çevrim zamanıdır.

max.c%dev= çevrim zamanının alt sınır değerinden maksimum yüzde sapma, $((max.c - CT_{min})/CT_{min}) \times 100$ ile hesaplanır, burada *max.c* ilgili sezgiselle elde edilmiş en kötü çevrim zamanıdır.

MAD= ortalama mutlak sapma, Eş. 3'ün istasyon sayısına bölünmesi ile hesaplanır ($MAD = \frac{1}{m}\sum_{k=1}^{m}\left|WSt_k - \frac{\sum_{i=1}^{n} t_i}{m}\right|$).

Ort. CPU s= Algoritmaların ele alınan problemler üzerindeki saniye cinsinden ortalama çözüm zamanları

Tablo 1'de her problem için algoritmaların verilen sapma değerleri, farklı istasyon sayılarıyla çözülen örneklerin ortalamasıdır. En iyi sapma değerleri italik puntolarla gösterilmiştir. Tüm algoritmaların ortalama çevrim zamanı amacı açısından ortalama yüzde sapmaları (*avg.c%dev*) incelendiğinde, SA algoritmasının 0,67%, TS algoritmasının 0,52%, SATS algoritmasının 0,54%, TSSA algoritmasının ise 0,25% sapma değerleri elde ettiği görülmektedir. Tüm algoritmalar, kabul edilebilir sapmalarla ele alınan problemi çözebilmelerine rağmen, TSSA diğerlerine büyük üstünlük sağlamıştır. Tavlama benzetimi üzerine inşa edilmiş melez algoritmanın (SATS) çözüm kalitesi açısından performansı, saf TS algoritmasına yakındır ancak, tabu aramanın çekirdeğini oluşturduğu melez algoritmanın (TSSA) oldukça gerisinde kalmıştır. TSSA algoritması ile farklı problemlerin çevrim zamanı amacı için elde edilen en kötü sapma değerleri bile en az diğer algoritmaların en iyi sapmaları kadar iyidir. Çözülen 128 örneğin her biri için yapılan 10'ar koşumda, teorik çevrim süresi alt sınırına en az bir kere ulaşıldıysa, ilgili algoritmanın o örnek için optimal sonucu elde ettiği kabul edilmiştir. Bu kabul ile 128 örneğin SA 72'sinde, TS ve SATS 79'unda, TSSA ise 97'sinde optimal çevrim zamanına ulaşmıştır. İş yükü düzgünleştirme amacı (*MAD*) dikkate alındığında da ortalama iş yükünden en küçük sapmaya ortalama 2,56 ile TSSA algoritmasının ulaştığı görülmektedir. TSSA algoritmasını sırasıyla TS, SATS ve SA algoritmaları izlemektedir. Bu amaç için de TSSA'nın SATS ile elde edilenden yaklaşık 74% daha iyi bir sonuç elde ettiği görülmektedir.

Sezgisel bir algoritmanın başarısını gösteren kriterlerden biri de, yapılan denemelerin ne kadar az değişken ve tutarlı sonuçlar verdiğidir. Bunun için, her örnek için yapılan koşumların içindeki en kötü sonuçlar da (*max.c%dev*) kaydedilmiştir. En iyi çözüm ve en kötü çözüm sonuçları arasındaki fark, algoritmayla elde edilen sonuçların değişkenliğini değerlendirmek için kullanılabilir. Çevrim zamanı çözümlerinin en az değişkenlik gösterdiği algoritma olarak TSSA öne çıkmaktadır. Tablo 1 incelendiğinde, *best.c%dev* ile *max.c%dev* arasındaki farkın TSSA algoritması için ortalama 0,02% olduğu görülmektedir. Söz konusu algoritma ile çözülen Gunther, Kilbridge, Tonge ve Lutz2 problemleri için en iyi ve en kötü sapma değerleri 0'dır. Bunun anlamı, ilgili problemlerin her örneği için yapılan tüm koşumlarda çevrim zamanı optimal değerlerine ulaşılmış olmasıdır. Bu kriter açısından ikinci en iyi algoritma, ortalama 0,19% ile TS olurken, 0,33% ile SATS ve 0,56% ile SA, TS'yi takip etmektedir.

Algoritmaların sonuca ulaşmak için ihtiyaç duydukları ortalama zamanlar da Tablo 1'den görülebilir. Bu kritere göre, TSSA algoritması, 32,87 CPU s'lik bir ortalamayla, diğerlerinden daha uzun koşum sürelerine sahiptir. Ancak, söz konusu algoritmanın ele alınan problemlerin en büyüğü, 111 göreve sahip, Arcus2 için ihtiyaç duyduğu ortalama çözüm zamanı 108,44 CPU s'dir. Bu sürenin de, ele alınan problemin kapsamı ve çözümüne ihtiyaç duyulabileceği sıklık göz önüne alındığında, oldukça makul olduğu söylenebilir.

5. Sonuçlar (Conclusions)

U-şekilli hatlar, işletmelere sağladıkları esneklik, verimlilik gibi avantajları sayesinde endüstride giderek daha fazla kabul görmeye başlamışlardır. Bu hatlarda çözülmesine sıklıkla ihtiyaç duyulan problemlerden biri, bir montaj hattı mevcutken, üretim sürecinde ya da talep yapısında değişiklikler olduğunda karşılaşılan ve üretim oranını maksimize etmeyi ya da, bir başka deyişle, çevrim süresini minimize etmeyi amaçlayan, tip-2 montaj hattı dengeleme problemidir. Bu çalışmada, U-şekilli hatlarda, iş yükü dengelemenin de dikkate alındığı tip-2 montaj hattı dengeleme problemi melez tavlama benzetimi-tabu arama algoritmalarıyla ele alınmıştır. Farklı çalışma prensiplerine sahip iki melez tavlama benzetimi-tabu arama algoritması birbirleriyle ve söz konusu algoritmaların saf halleriyle, literatürden alınan test problemleri üzerinde karşılaştırılmıştır. Ana iskeleti tabu arama üzerine kurulan melez algoritmanın ele alınan iki amaç ve elde edilen sonuçların tutarlılığı açısından diğer algoritmalara üstünlük sağladığı görülmüştür.

Bu çalışmada, U-şekilli tip-2 montaj hattı dengeleme problemiyle ilgilenilmiştir. Bu problem üzerinde en iyi performansı gösteren TSSA algoritması, ek destekleyici elemanlarla geliştirilerek, problemin daha karmaşık versiyonlarının çözümü için tasarlanabilir.

Kaynaklar (References)

1. Kılınççı, Ö., Assembly line balancing problem with resource and sequence-dependent setup times (ALBPRS), Journal of the Faculty of Engineering and Architecture of Gazi University, 38 (1), 557-570, 2023.
2. Scholl, A., Voß, S., Simple assembly line balancing—Heuristic approaches, J. Heuristics, 2, 217-244, 1996.
3. Scholl, A., Klein, R., ULINO: Optimally balancing U-shaped JIT assembly lines. Int. J. Prod. Res, 37 (4), 721-736, 1999.
4. Erel, E., Sabuncuoglu, I., Aksu, B.A., Balancing of U-type assembly systems using simulated annealing, Int. J. Prod. Res, 39 (13), 3003-3015, 2001.
5. Miltenburg, G.J., Wijngaard, J., The U-line line balancing problem, Manage. Sci., 40 (10), 1378-1388, 1994.
6. Gökçen, H., Agpak, K., A goal programming approach to simple U-line balancing problem, Eur. J. Oper. Res., 171 (2), 577-585, 2006.
7. Kara, Y., Paksoy, T., Chang, C.T., Binary fuzzy goal programming approach to single model straight and U-shaped assembly line balancing, Eur. J. Oper. Res., 195 (2), 335-347, 2009.
8. Jonnalagedda, V., Dabade, B., Application of simple genetic algorithm to U-shaped assembly line balancing problem of type II, IFAC proceedings volumes, 47 (3), 6168-6173, 2014.
9. Şahin, M., Kellegöz, T., An efficient grouping genetic algorithm for U-shaped assembly line balancing problems with maximizing production rate, Memet. Comput., 9, 213-229, 2017.
10. Li, M., Tang, Q., Zheng, Q., Xia, X., Floudas, C.A. Rules-based heuristic approach for the U-shaped assembly line balancing problem, Appl. Math. Modell., 48, 423-439, 2017.
11. Li, Z., Janardhanan, M.N. Rahman, H.F., Enhanced beam search heuristic for U-shaped assembly line balancing problems, Eng. Optim., 53 (4), 594-608, 2021.
12. Nakade, K., Ohno, K., Shanthikumar, J.G., Bounds and approximations for cycle times of a U-shaped production line, Oper. Res. Lett., 21, 191-200, 1997.
13. Şahin, M., Kellegöz, T., Increasing production rate in U-type assembly lines with sequence-dependent set-up times. Eng. Optim., 49 (8), 1401-1419, 2017b.

14. Zhang, Z., Tang, Q., Han, D., Li, Z., Enhanced migrating birds optimization algorithm for U-shaped assembly line balancing problems with workers assignment, Neural Comput. Appl., 31, 7501-7515, 2019.

15. Li, Z., Janardhanan, M.N., Ashour, A S., Dey, N., Mathematical models and migrating birds optimization for robotic U-shaped assembly line balancing problem, Neural Comput. Appl., 31, 9095-9111, 2019.

16. Wang, T., Fan, R., Peng, Y., Wang, X., Optimization on mixed-flow assembly u-line balancing problem, Cluster Comput., 22, 8249-8257, 2019.

17. Pınarbaşı, M., New mathematical and constraint programming models for U-type assembly line balancing problems with assignment restrictions, Eng. Optim., 54 (8), 1289-1304, 2022.

18. Arikan, M., A Tabu Search Algorithm for Type-2 U-Shaped Simple Assembly Line Balancing Problem, Sixteenth International Conference on Management Science and Engineering Management, Ankara-Türkiye, 435-449, 4-5 August, 2022.

19. Gonçalves, J. F., De Almeida, J. R., A hybrid genetic algorithm for assembly line balancing, J. Heuristics, 8, 629-642, 2002.

20. Triki, H., Mellouli, A., Hachicha, W., Masmoudi, F., A hybrid genetic algorithm approach for solving an extension of assembly line balancing problem, Int. J. Comput. Integr. Manuf., 29 (5), 504-519, 2016.

21. Zhang, J. H., Li, A.P., Liu, X.M., Hybrid genetic algorithm for a type-II robust mixed-model assembly line balancing problem with interval task times, Adv. Manuf., 7, 117-132, 2019.

22. Álvarez-Miranda, E., Pereira, J., Torrez-Meruvia, H., Vilà, M., A hybrid genetic algorithm for the simple assembly line balancing problem with a fixed number of workstations, Mathematics, 9 (17), 2157, 2021.

23. Salehi, M., Maleki, H. R., Niroomand, S., Solving a new cost-oriented assembly line balancing problem by classical and hybrid meta-heuristic algorithms, Neural Comput. Appl., 32, 8217-8243, 2020.

24. Tasan, S. Ö., Tunali, S., Improving the genetic algorithms performance in simple assembly line balancing, Computational Science and Its Applications-ICCSA 2006: International Conference, Springer Berlin Heidelberg, 78-87, 2006.

25. Suwannarongsri, S., Limnararat, S., Puangdownreong, D., A new hybrid intelligent method for assembly line balancing, 2007 IEEE International Conference on Industrial Engineering and Engineering Management, 1115-1119, December, 2007.

26. Jian-sha, L., Ling-ling, J., Xiu-lin, L., Hybrid particle swarm optimization algorithm for assembly line balancing problem-2, 2009 16th International Conference on Industrial Engineering and Engineering Management, 979-983, October, 2009.

27. Dong, J., Zhang, L., Xiao, T., A hybrid PSO/SA algorithm for bi-criteria stochastic line balancing with flexible task times and zoning constraints, J. Intell. Manuf., 29, 737-751, 2018.

28. Hamta, N., Ghomi, S. F., Jolai, F., Shirazi, M.A., A hybrid PSO algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect, Int. J. Prod. Econ., 141 (1), 99-111, 2013.

29. Yuan, B., Zhang, C., Shao, X., Jiang, Z., An effective hybrid honey bee mating optimization algorithm for balancing mixed-model two-sided assembly lines, Comput. Oper. Res., 53, 32-41, 2015.

30. Cakir, B., Altiparmak, F., Dengiz, B., Multi-objective optimization of a stochastic assembly line balancing: A hybrid simulated annealing algorithm, Comput. Ind. Eng., 60 (3), 376-384, 2011.

31. Yildiz, S. T., Yildiz, G., Okyay, R., Assembly line balancing problem with hierarchical worker assignment, positional constraints, task assignment restrictions and parallel workstations, Int. J. Ind. Eng. Theory Appl. Pract., 27 (3), 345-377, 2020.

32. Niroomand, S., Hybrid artificial electric field algorithm for assembly line balancing problem with equipment model selection possibility, Knowledge-Based Syst., 219, 106905, 2021.

33. Roshani, A., Paolucci, M., Giglio, D., Tonelli, F., A hybrid adaptive variable neighbourhood search approach for multi-sided assembly line balancing problem to minimise the cycle time, Int. J. Prod. Res, 59 (12), 3696-3721, 2021.

34. Suwannarongsri, S., Puangdownreong, D., Optimal balancing of multi-objective U-shaped assembly lines using the TSGA method, 2008 IEEE International Conference on Industrial Engineering and Engineering Management, 307-311, December, 2008.

35. Nejad, G. M., Husseinzadeh Kashan, A., Shavarani, S.M., A novel competitive hybrid approach based on grouping evolution strategy algorithm for solving U-shaped assembly line balancing problems, Prod. Eng., 12 (5), 555-566, 2018.

36. Khorram, M., Eghtesadifard, M., Niroomand, S., Hybrid meta-heuristic algorithms for U-shaped assembly line balancing problem with equipment and worker allocations, Soft Comput., 26 (5), 2241-2258, 2022.

37. Zolfaghari, S., Liang, M., Jointly solving the group scheduling and machining speed selection problems: a hybrid tabu search and simulated annealing approach, Int. J. Prod. Res, 37 (10), 2377-2397, 1999.

38. Zhang, C. Y., Li, P., Rao, Y., Guan, Z. A very fast TS/SA algorithm for the job shop scheduling problem, Comput. Oper. Res., 35 (1), 282-294, 2008.

39. Arıkan M., A tabu search algorithm for the simple assembly line balancing problem of type-2 with workload balancing objective, Journal of the Faculty of Engineering and Architecture of Gazi University 32 (4), 1169-1179, 2017.

40. Arıkan, M., Type-2 Assembly Line Balancing with Workload Smoothing Objective: A Reactive Tabu Search Algorithm, GU J Sci., 34 (1), 162-178, 2021.

41. Rachamadugu, R., Talbot, B., Improving the quality of workload assignments in assembly lines. Int. J. Prod. Res, 29 (3), 619-633, 1991.

42. Driscoll, J., Thilakawardana, D., The definition of assembly line balancing difficulty and evaluation of balance solution quality, Rob. Comput. Integr. Manuf., 17 (1-2), 81-86, 2001.

43. Scholl, A., Data of assembly line balancing problems. Techn. Hochsch., Inst. für Betriebswirtschaftslehre, 1995.

44. Kirkpatrick, S., Gelatt Jr, C. D., Vecchi, M. P., Optimization by simulated annealing. Science, 220 (4598), 671-680, 1983.

45. Glover, F., Tabu search—part I, ORSA Journal on computing, 1 (3), 190-206, 1989.

46. Glover, F., Tabu search—part II, ORSA Journal on computing, 2 (1), 4-32, 1990.

47. Glover, F., Taillard, E., Taillard, E., A user's guide to tabu search, Ann. Oper. Res., 41 (1), 1-28, 1993.

48. Glover F, Laguna M., Tabu search. Dordrecht: Kluwer Academic Publishers, 1997.

49. Swarnkar, R., Tiwari, M K., Modeling machine loading problem of FMSs and its solution methodology using a hybrid tabu search and simulated annealing-based heuristic approach, Rob. Comput. Integr. Manuf., 20 (3), 199-209, 2004.

50. Ben-Ameur, W., Computing the initial temperature of simulated annealing, Comput. Optim. Appl., 29, 369-385, 2004.