FLSF (Felsefe ve Sosyal Bilimler Dergisi)
FLSF (Journal of Philosophy and Social Sciences)
https://dergipark.org.tr/tr/pub/flsf

2024 Bahar / Spring 2024
sayı 37 / issue 37
s. 325-344 / pp. 325-344
ISSN 2618-5784

ARAŞTIRMA MAKALESİ / RESEARCH ARTICLE
DOI Number: 10.53844/flsf.1437990

TURİNG TESTİNİN SINIRLARI ÜZERİNE FELSEFİ BİR İNCELEME

Ümit TAŞTAN*

ÖZ

Bir makineye zekâ atfetmenin gerek ve yeter koşulları konusunda herkesçe uzlaşılan bir zemin yoktur. Ancak Turing testi, makine zekâsının yeteneklerini değerlendirmek için çok önemli bir kilometre taşı ve ölçüt olarak durmaya devam etmektedir. Bu makalede, Turing testinin geçilmesinin önündeki bizce muhtemel engellerden ikisini inceliyoruz. Birincisi, programlamanın temelinde yer alan mantıksal çıkarıma dayalı engellerdir. İkincisi ise hesap karmaşıklığı alanındaki bazı problemlerle ilgili etkili çözümler bulunamamasından kaynaklanan engeldir. Bu iki engeli ortaya koyarken mantık disiplinine ve teorik bilgisayar bilimine dair literatürü felsefi bir yöntemle sentezlemeye çalışıyoruz. Çalışmamız, yukarıda sözü edilen iki engelden hareketle ve bu engeller var olduğu sürece Turing testinin geçilemez olduğu sonucuna varmaktadır.

Anahtar Kelimeler: *Alan Turing, Turing testi, yapay zekâ, hesap karmaşıklığı, monoton olmayan düşünme*

A PHILOSOPHICAL INVESTIGATION ON THE LIMITS OF THE TURING TEST

ABSTRACT

There is no consensus on the necessary and sufficient conditions for attributing intelligence to a machine. However, the Turing test remains a crucial milestone and benchmark for assessing the capabilities of machine intelligence. In this paper, we examine two of what we see as possible obstacles to passing the Turing test. The first is in the area of logical inference, which is at the core of programming. The second is the lack of effective solutions to some problems in the area of computational complexity. In presenting these two obstacles, we try to combine the literature on the discipline of logic and theoretical computer science in a philosophical way. Our paper concludes that the Turing test is impassable as long as these two obstacles exist.

Keywords: *Alan Turing, Turing test, artificial intelligence, computational complexity, non monotonic reasoning*

* Araştırma Görevlisi, Aksaray Üniversitesi Fen-Edebiyat Fakültesi Felsefe Bölümü. umit16tastan@gmail.com, ORCID: 0000-0002-5121-0943

Makalenin geliş tarihi: 15.02.2024 ***Submission Date: 15 February 2024***
Makalenin kabul tarihi: 29.04.2024 ***Approval Date: 29 April 2024***

Giriş

Alan Turing 1950 yılında *Mind* dergisinde yayınlamış olduğu *Hesaplama Makineleri ve Zekâ* (İng. *Computing Machinery and İntelligence*)[1] isimli makalesinde bir taklit oyunundan (İng. *imitation game*) bahsederek yapay zekâ tartışmalarını başlatır. Turing'in *taklit oyunu* olarak duyurduğu bu test, daha sonraki çeşitlemelerinde genellikle *Turing testi* olarak anılmıştır. Turing testinde bir sorgulayıcı, bir insan ve bir bilgisayar yer alır. Sorgulayıcı, bilgisayar ve insanla yazılı olarak gerçekleştirdiği sohbet sonucunda bilgisayarın insan gibi davranıp davranmadığını belirlemeye çalışır. Makinenin amacı, sorgulayıcının yanlışlıkla makinenin diğer kişi olduğu sonucuna varmasına neden olmaya çalışmaktır; diğer kişinin amacı ise sorgulayıcının makineyi tanımasını sağlamaktır**.** Turing, *taklit oyunu* adını verdiği bu testi, makinelerin düşünüp düşünemeyeceği sorusunu ele almanın bir yolu olarak görmektedir. Turing'in yukarıda anılan makalesinde sözünü ettiği makinelerin bugünkü anlamda bir makine ya da bilgisayar olmadığını belirtmeliyiz. Ancak, Turing tarafından belli bir algoritma uyarınca bir şerit üzerinde işlemler yapan, hesaplama yapan bir matematikçi gibi düşünülen Turing makinesi günümüzde kullandığımız bilgisayarların ilham kaynağıdır.

Çalışmamız, daha ziyade Turing'in 1950 tarihli makalesinde ileri sürdüğü test hakkında olacaktır. Ancak adı geçen yazıda *hesaplama makinesi* olarak adlandırılan makinelerin nasıl tanımlandığını görmek amacıyla Turing'in 1936 tarihinde yayınladığı *On Computable Numbers, with an Application to the Entscheidungsproblem* [*Hesaplanabilir Sayılar Üzerine: Karar Probleminin Bir Uygulaması*] adlı çalışmasına değineceğiz. Bu yazısında Turing, eğer bir makinenin şerit üzerindeki hareketi tamamen bir yapılandırma (İng. *configuration*) tarafından belirleniyorsa o makinenin *otomatik makine* olduğunu öne sürmektedir.[2] Turing'in otomatik makinesi hiçbir sorunun tüketemeyeceği kadar büyük bir belleğe, bu bellekten okuma ve belleğe yazma araçlarına sahip olan teorik bir cihazdır. Bu makinenin, cevabı evet ya da hayır olan karar

[1] Alan Turing, "Computing Machinery and Intelligence," *Mind* LIX, no. 236 (1950).
[2] A. M. TURING, "On Computable Numbers, with an Application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society* s2-42, no. 1 (1937): 232.

problemlerini çözmek ve belirli bir işaret dizisinin (İng. *string*) makinenin diline ait olup olmadığını belirlemek gibi amaçları vardır.[3]

Turing otomatik makine fikrini tanıttıktan sonra 1950 tarihli makalesinde makineler düşünebilir mi sorusunu sormuş ve ardından da makinelerin düşünmesine karşı olası bazı itirazları ele almıştır. Ayrıca ileride görüleceği gibi, bilgisayarlara yeterli işlem hızı ve bellek kapasitesi kazandırılırsa bu taklit oyununu geçebileceği öngörüsünde bulunmuştur.[4] Peki, Turing, makinelerin düşünmesi ile tam olarak neyi kasteder? Turing'in buradaki ölçütü, makinelerin tamamen insan benzeri davranışlar sergileyebilmesidir. Bu davranışlara; araç sürmek, pilotluk yapmak, hasta muayene etmek, hâkim gibi karar vermek gibi örnekler verilebilir. Turing'in henüz bilgisayarlar ortaya çıkmadan önce ortaya atmış olduğu "hesap yapan makine" fikrinin artık çok ilkel kaldığı akla gelebilir. Ancak birazdan görüleceği üzere, bugün pek çoğumuzun kullandığı modern bilgisayarın hala Turing'in hesaplama modeline dayandığını söylemeliyiz. Bilgisayarların programlanma konusunda ne kadar girift bir hal aldığı ve her geçen gün farklı görevleri yerine getirecek gelişmeler kaydettiği konusunda şüphe yoktur. Ancak, ortaya çıkan bütün bu gelişimden geriye dönüp bakıldığında sorulması gereken soru şudur: Turing makinesi esasına bağlı olan modern bilgisayarlar niçin Turing testini geçme konusunda hâlâ başarısız olmaktadır? Çalışmamız bu soruya iki temel gerekçe üzerinden bir yaklaşım sunmaktadır. Birincisi, *geriçıkarım* (İng. *abduction*) gibi *monoton olmayan düşünme* biçimlerinin biçimsel sistemlerde modellenmesi konusunda yaşanan zorluklarla ilgilidir. İkincisi ise, etkili bir çözümü bulunamayan bazı problemlerin bilgisayarları maruz bıraktığı *hesap karmaşıklığı* sorunu ile ilgilidir. Bu iki sorunun Turing testinin geçilemiyor oluşunda pay sahibi olduğunu düşünüyoruz.

Turing Makinesi ve Hesaplama

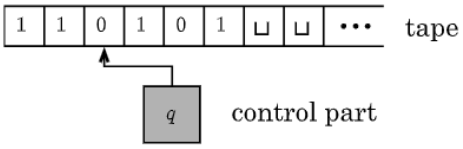
Turing her ne kadar karmaşık matematiksel hesaplamalar yapmayı sağlayan modeller üzerine çalışsa da bazı hesaplanamayan problemler olduğunun da farkındaydı. Bu sebeple 1936 yılındaki makalesini *karar problemi*

[3] Edna E. Reiter and Clayton M. Johnson, *Limits of Computation: An Introduction to the Undecidable and the Intractable* (Boca Raton, FL: CRC Press, Taylor & Francis Group, 2013), 61,62.
[4] Alan Turing, "Computing Machinery and Intelligence," *Mind* LIX, no. 236 (1950): 442.

olarak ifade edilen probleme cevap verecek hiçbir makine olamayacağını göstermek için kaleme almıştır.[5] *Karar problemi* (Alm. *entscheidungsproblem*) birinci dereceden mantığa (İng. *first order logic*) ait her ifadenin bu mantıkta türetilebilir olup olmadığına karar verme problemidir.[6] Bir *karar problemi*ne verilmesi gereken cevap evet ya da hayır şeklindedir. Örneğin "*3* sayısı asal sayı mıdır?" ya da "5 sayısı 3'e tam bölünebilir mi?" gibi sorular evet ya da hayır şeklinde cevaplanabildiği için birer karar problemi sorusudur. Turing makinesine bir *karar problemi* girdi olarak verildiğinde makinenin algoritması evet ya da hayır çıktılarından birisini üretirse bu problem karar verilebilir (İng. *decidable*) bir problemdir.[7]

Bir Turing makinesinin nasıl hesaplama yaptığına geçmeden önce makineyi oluşturan üç farklı bileşeni de kısaca tanıtmak istiyoruz. Makinenin çalışma aşamalarını tasvir eden Turing makinesi *yapılandırması*na ait üç bileşen şunlardır: işlemleri yürüten bir okuma/yazma kafası (İng. *head*), işlemlerin üzerinde yürütüldüğü bir şerit (İng. *tape*) ve makinenin belirli bir zamanda belirli bir durumda bulunduğunu belirten sonlu durum kümesi.[8] Turing'in tasarladığı makine, her biri bir "sembol" taşıyabilen kutucuklara ya da karelere ayrılmış, her iki yöne de sonsuzca uzayabilen kâğıt benzeri doğrusal bir şeride sahiptir. Her bir kutucuğa tek bir işaret yazılabilmekte ve boş kutucuklar kendilerinin boş olduğunu gösteren özel bir işarete sahip olabilmektedir. Şerit üzerinde işlemler yürüten makine yürüttüğü işlemlerin her bir adımında sonlu sayıda durumdan birinde bulunacak biçimde tasarlanmıştır. Bu sebeple, makine herhangi bir anda sadece bir kutucuğu tarayabilir ve Turing bu kutucuğa *taranmış kare* (İng. *scanned square*) adını verir.[9]

[5] Liesbeth de Mol, *Turing Machines* (2018), https://plato.stanford.edu/entries/turing-machine/ (Erişim: 2 Şubat, 2023).
[6] Age.
[7] Carlos Martin-Vide, Victor Mitrana and Gheorghe Paun, eds., *Formal Languages and Applications*, 2004 edition, Springer eBook Collection 148 (Berlin: Springer, 2004), 5.
[8] Reiter, E.E., & Johnson, C.M. (2012). Limits of Computation: An Introduction to the Undecidable and the Intractable, s.66.
[9] A. Turing, "On Computable Numbers, s. 231.



Şekil 1: Turing makinesinin şeması

Yukarıdaki şekilde görüldüğü üzere, bir Turing makinesi işlem yaptığı her bir adımda; taranan kutucukta hangi işaret olduğunu belirleyebilmekte, bir işareti silebilmekte ya da yazabilmekte, bir kutucuk sağa ya da bir kutucuk sola hareket edebilmektedir.[10] Şekilde makinenin mevcut durumu *q* sembolü ile gösterilmiştir. Şeridin içeriği *110101* dizisini (İng. *string*), makinenin kafası ise en soldaki kareden itibaren üçüncü karenin üzerinde olduğu durumu gösterir. Ayrıca şeritteki *110101* dizisini içeren kareler dışındaki tüm karelerin boş olduğunu gösteren bir sembol de bulunmaktadır. Böylelikle şeridin en solundaki ardışık kareler bir girdi dizisini oluştururken dizinin sağ tarafında ise sonsuz sayıda boş sembol olduğu varsayılır. Makinenin okuma/yazma kafası başlangıç durumundayken şeridin en solundaki karenin üzerinde yer alır. Hesaplama başladığında makine, geçiş fonksiyonu (İng. *transition function*) tarafından belirtilen kurallara göre şeridin içeriğini, durumu ve kafanın konumunu değiştirme işlemini tekrarlar. Makine şeritteki girdi dizisinin tamamını okuduğunda ise üç farklı durum ortaya çıkar: kabul veya red durumunda hesaplama sonlanırken üçüncü durumda makine sonsuza kadar hesaplamaya devam eder.[11] Buradan hareketle, bir Turing makinesinin çok genel bir açıdan yukarıda tasvir edildiği gibi hesaplama yaptığını söyleyebiliriz. Teorik bilgisayar bilimi açısından düşünecek olursak, Turing'in hesaplama modelinin bugünkü bilgisayarlar için de çok temel bir yerde durduğunu belirtmeliyiz.

Massachusetts Institute of Technology'de (MİT) matematik profesörü olan Michael Sipser, *Hesap Kuramına Giriş*[12] isimli kitabında, genel amaçlı bilgisayarları simüle edebilecek yetenekte olan modelin *Turing makinesi* modeli

[10] Çitil, Ahmet Ayhan. *Matematik ve Metafizik Kitap 1: Sayı ve Nesne*. Alfa Yayınları, 2012, s. 166
[11] Akira Maruoka, *Concise Guide to Computation Theory* (London: Springer, 2011), 133.
[12] Michael Sipser, *Introduction to the Theory of Computation* (Florence: Cengage Learning, Inc, 2012).

olduğunu ifade eder. Bu yüzden teorik açıdan bakıldığında en güçlü hesaplama modeli Turing makinesidir. Sipser'ın bir Turing makinesi modelinin güçlü olmasıyla kastettiği şey ise gerçek bilgisayarların yaptığı her şeyi yapabilmesidir.[13] Bununla birlikte, bir Turing makinesinin bile belli başlı problemleri çözemediği bilinmektedir. Böylece bu problemlerin varlığı, hesaplamanın teorik sınırları olduğunu bize göstermiş olur.

Sipser'ın sözlerinden hareketle, günümüz bilgisayarlarının hesap yapma kabiliyeti bakımından Turing makinesinden niteliksel bir fark taşımadığını tespit etmek önemlidir. İleride açıklayacağımız gibi, günümüzdeki yüksek işlem hızına sahip bilgisayarların hala Turing testini geçemiyor oluşu da bu tespitimizle örtüşmektedir. 2014 yılında, *Eugene Goostman* adlı bilgisayar programının Turing testi yarışmasında jürinin %33'ünü kandırdığı için "Turing Testini geçtiği" iddiaları ortaya atılmıştır. Ancak benzer sonuçların elde edildiği başka tek seferlik yarışmalar da olmuştur. 1991 yılında *PC Therapist* jürinin %50'sini kandırmıştı. Ayrıca 2011'deki bir gösteride *Cleverbot* daha da yüksek bir başarı oranına sahipti. Bu vakaların üçünde de sorgulamanın boyutu çok küçüktü ve sonuç güvenilir bir şekilde tahmin edilebilir değildi: hiçbir vakada, ortalama bir sorgulayıcının beş dakikalık sorgulama sonrasında ilgili program hakkında doğru tespiti yapma şansının %70'ten fazla olmadığını söylemek için güçlü gerekçeler yoktu.[14]

Turing, *hesaplama*dan bahsederken, (matematiksel ya da mantıksal) semboller üzerinde insan ya da mekanik cihaz tarafından sonlu sayıda kurala göre gerçekleştirilen işlem dizilerini kastetmektedir. Turing'e göre bu işlem dizileri, sezgi, buluş ya da tahmin gerektirmeyen ve yürütülmesi her zaman doğru çözümü üreten işlem dizileri olmalıdır.[15] Turing'in yapmayı amaçladığı şey, bir girdinin hesaplanabilmesi için etkili bir yordam (İng. *procedure*) ya da karar sürecinin tanımını ortaya koymaktır. Bu karar süreci öylesine etkin bir işleyişe sahip olmalıdır ki herhangi bir girdi üzerinde *kabul* ya da *red* cevabını vermeyi mümkün kılacak kesin sonuçları garanti altına alabilsin. Burada dikkat edilmesi gereken şey, hesaplamanın bir başlangıç noktasının olması ve diğer bütün işlemlerin sırasıyla bu başlangıç noktasını takip etmesidir. Örneğin;

---

[13] Sipser, "Introduction to the Theory of Computation", 165.

[14] Oppy, Graham and David Dowe, *The Turing Test*, (2021), *The Stanford Encyclopedia of Philosophy,* https://plato.stanford.edu/archives/win2021/entries/turing-test/>/ (Erişim: 25 Nisan, 2024).

[15] Gualtiero Piccinini, "Alan Turing and the Mathematical Objection," *Minds and Machines* 13, no. 1 (2003): 26, https://link.springer.com/article/10.1023/A:1021348629167.

"karesi dokuz olan doğal sayıyı bulunuz" şeklinde bir sorunun Turing makinesine sorulduğunu düşünelim. Bu durumda makine, söz gelimi "0"dan başlamak üzere sırasıyla her doğal sayının karesini alarak hesaplama yapmaktadır. İlk önce "0"ın karesini alır ve işleminin sonucu "9" sayısıyla eşleşmediği için işleme devam eder. Daha sonra "1"in karesini alır, sonra "2"nin ve son olarak "3"ün karesini alır. "3"ün karesini alınca dokuz rakamını hesapladığı için yönergenin "DUR" komutu devreye girer ve "3" rakamı çıktı olarak şerit üzerinde gösterilir.

Yukarıda tarif ettiğimiz hesaplama işleminin sonlu işlem adımında gerçekleşmesi, makinenin bir yerde durması gerektiğini göstermek içindir. Verili bir girdi üzerinde hesap yapan bir Turing makinesinin bu girdi üzerinde durup durmayacağının saptanamaması problemine *durma problemi* (İng. *halting problem*) denilmektedir.[16] Turing, *durma problemi* olarak ifade etmese de Turing makinelerinin cevap veremeyeceği ya da yanlış cevap vereceği birtakım soruların olduğunu 1950 tarihli makalesinde ifade eder. Bu makalede makinelerin düşünüp düşünemeyeceği fikrine karşı gelebilecek eleştirilerden bahsederken *matematiksel itiraz* (İng. *mathematical objection*) isimli bir eleştiriyi ele alır. Bu eleştiri Gödel'in *tamamlanamazlık teoremleri*nde (İng. *incompleteness theorems*) ortaya koyduğu sonuçlara dayanmaktadır. Kurt Gödel, programlamanın da temelinde yer alan biçimsel sistemlerde, doğru olduğu insan zihni tarafından bir şekilde fark edilebilen ama matematiksel olarak ispatı verilemeyen önermeler olduğunu göstermiştir.[17] Dolayısıyla biçimsel sistemde ispatı verilemeyen bazı önermeler bilgisayarlar için de karar verilemez bir önerme anlamına gelmektedir. Turing bu eleştiriyi ele alırken makineleri başarısız kılacak soruların mümkün olduğunu ve makinelerin insan aklında bulunmayan bir engeli olduğunu kabul eder. Hatta makineye bu tarz sorular sorulduğunda makine net bir yanıt verse bile bu yanıtın yanlış olması gerektiğini bileceğimizi ve bunun bize kesin bir üstünlük duygusu verdiğini de ekler. Ancak Turing'e göre herhangi bir makine böyle bir engele sahip olsa da insan aklının da bazı engellere sahip olmadığı iddia edilemez.[18]

[16] Sipser, "Introduction to the Theory of Computation", 216.
[17] Ümit Taştan, "Gödel'in Tamamlanamazlık Teoremleri Bakımından Biçimsel Dillerde İspatlanabilirlik-Doğruluk İlişkisi," *MetaZihin: Yapay Zeka ve Zihin Felsefesi Dergisi* (2022): 58.
[18] A. M. Turing, Computing Machinery and Intelligence, Mind, New Series, Sayı. 59, Published by: Oxford University Press, s. 445

Turing'in 1936 yılındaki çalışmasının üzerinden neredeyse bir yüzyıl geçti ve Turing'in projesinde ciddi bir ilerleme kaydedildi. Günümüzde saniyede milyarlarca işlem yapabilen bilgisayarlar olduğu göz önüne alınırsa, Turing'in işlem hızı ve bellek taleplerinin fazlasıyla karşılandığını söyleyebiliriz. Ancak bilgisayarların işlem hızlarında devasa bir gelişme yaşanmış olsa da Turing testinin geçilmesi konusundaki beklentiler hala karşılanamamaktadır. Çalışmanın başında belirttiğimiz gibi, Turing testinin geçilmesine dair ileri sürdüğümüz iki engeli incelemeye çalışacağız. Birincisi, mantık literatüründe farklı çıkarım türlerinden birisi olarak kabul edilen monoton olmayan akıl yürütme türü ile ilgilidir.

Monoton Olmayan Akıl Yürütme ve Geriçıkarım

Modern mantık çalışmaları sonucunda artık biliyoruz ki insan zihninin en az üç farklı akıl yürütme biçimi vardır: tümdengelimli (İng. *deductive*), tümevarımlı (İng. *inductive*) ve geriçıkarımlı (İng. *abductive*) akıl yürütme. Standart mantıkta akıl yürütme, bazı öncüllerden sonuca gidilmesini gerektiren çıkarım tarzıdır. Eğer öncüllerin doğruluğu sonucun doğruluğunu kesin olarak gerektiriyorsa buna tümdengelimli akıl yürütme denir. Eğer öncüllerin doğruluğu sonucun doğruluğunu kesin olarak değil de yüksek bir olasılıkla gerektiriyorsa tümevarımlı akıl yürütme (İng. *inductive*) yapıldığı anlamına gelir.[19]

Tümevarımlı akıl yürütmenin bilgisayarlarda modellenmesi konusunda pek çok ilerleme olduğunu belirtmeliyiz. Özellikle büyük veri kümelerine dayanan geniş dil modelleri sayesinde bilgisayarların insana benzer çıkarımlar yapmasının önü açılmıştır. Ancak tümevarımlı akıl yürütmeden çok özel bir biçimde farklılaşan geriçıkarımın modellenmesi konusunda hala ciddi zorluklar vardır. Dolayısıyla çalışmamızın başında belirttiğimiz mantıksal çıkarım engelinden kastımızın geriçıkarıma dayalı akıl yürütme olduğunu belirtmeliyiz. Geriçıkarıma dayalı akıl yürütme, Amerikalı filozof Charles Sanders Pierce tarafından 19. yy.'ın ikinci yarısında mantıksal bir forma kavuşturulmuştur.[20] Geriçıkarım, olgulardan yola çıkarak bu olguları en iyi şekilde açıklayan

[19] Keith J. Holyoak and Robert G. Morrison, *The Oxford Handbook of Thinking and Reasoning,* Oxford library of psychology (Oxford: Oxford University Press, 2012), 2.
[20] A. Aliseda, *Abductive Reasoning: Logical Investigations into Discovery and Explanation / by Atocha Aliseda,* Synthese library 330 (Dordrecht, London: Springer, 2006), 35.

nedenleri bulmaya çalışan akıl yürütme türüdür. İçinde yaşadığımız dünyayı daha iyi anlamamıza yardımcı olan bu akıl yürütme türü, şaşırtıcı gözlemlere anlam vermek amacıyla kullanılır.[21] Eksik bilgilere dayanarak birçok farklı durum hakkında açıklama yapmayı mümkün kılan geriçıkarımın çok geniş bir kullanım alanı vardır.

Akşamleyin işten eve döndüğünüzde oturma odasındaki pencerenin açık olduğunu fark ettiniz. Evden çıkarken kapalı olduğuna emin olduğunuz için açık unutulma seçeneğini doğrudan eliyorsunuz. Ayrıca pencerenin önündeki cam şişenin devrilip kırıldığını, buzdolabı kapağının açık vaziyette olduğunu ve bazı yiyeceklerin de eksik olduğunu fark ettiniz. Bu kanıtları birleştirince aklınızda bir soru belirir: Eve hırsız mı girdi? Eğer Tayland gibi bir yerde yaşıyorsanız bu hırsızın bir maymun olması da pekâlâ mümkündür. Bu tür sorulara karşı insan zihni, bulunduğu duruma bağlı olarak yaşadığı olayı en iyi açıklayan çıkarımları hızlı bir şekilde yapma yeteneğine sahiptir.

Bu tarz akıl yürütmeleri yapabiliyor olmamız bize ilginç gelmiyor olsa da bunu nasıl yaptığımız konusu mantık disiplini açısından oldukça ilginçtir. Çünkü bu tip durumlarda sadece yaşadığımız olayı en iyi açıklayacak seçenekleri hızlı bir şekilde bir araya getirmekle kalmıyor aynı zamanda birbiri ile doğrudan ilişkili olmayan kanıtlardan anlamlı bütünler elde ediyoruz. Dolayısıyla, nedensellik zincirinin çok farklı bir şekilde kurulduğu ve çoğu kez sağduyu, eksik bilgi, monoton olmayan düşünme gibi unsurların işin içine karıştığı bir akıl yürütme türünden bahsediyoruz. Peki, bilgisayarlar böyle bir düşünmeyi modelleme konusunda ne kadar başarılı?

OpenAI şirketi tarafından sırasıyla 2022 ve 2023 yıllarında piyasaya sürülen *GPT-3.5* ve *GPT-4* isimli büyük dil modelleri insan-bilgisayar etkileşimini yeni bir boyuta taşımıştır. Bu modeller üzerinden alınan yanıtların insan çıkarımlarına benzerliği karşılaştırılmaktadır. Yapılan bir çalışma, benzerlik (İng. *similarity*), türdeşlik (İng. *typicality*), çeşitlilik (İng. *diversity*) ve monoton olmama (İng. *non-monotonicity*) dahil olmak üzere 11 parametreden oluşan bir karşılaştırma yapmıştır. Sonuç olarak neredeyse insan benzeri çıkarımlar yapan

[21] Aliseda, "Abductive Reasoning", 28.

modellerin sadece monoton olmama konusunda insan yanıtlarını yakalayamadığı görülmüştür.[22]

Yukarıdaki çalışmadan hareketle, bilgisayarın *geriçıkarım* yapabilmesini mümkün kılacak şekilde programlanmasının bugüne kadar mümkün olmadığını söyleyebiliriz. Bunun muhtemel nedeni, akıl yürütme zincirinin tümdengelimde olduğu gibi monoton olarak ilerlemiyor olmasıdır. Bu sebeple geriçıkarım gibi monoton olmayan akıl yürütmelerin programlanması konusunda ciddi bir zorluk vardır. Bilgisayar biliminin erken dönemlerinden beri bu zorluk boyut değiştirerek çeşitli kavramsallaştırmalar altında devam etmiştir. Yapay zekâ kavramının mûcidi olarak görülen John McCarthy, sağduyunun programlanması ve formüle edilmesi konusunda pek çok çalışma ortaya koymuştur. 1959 yılında yazmış olduğu bir makalede insanların kendi deneyimlerinden öğrenmesi gibi makinelerin de kendi deneyiminden öğrenmesini sağlayacak programlar yazılabileceğini belirtir.[23] McCarthy, insanın fiili olarak eylediği eylemlerin bir otomat sistemindeki belli çıktı dizilerinden çok da farklı olmadığını düşünür. Belli girdi ve çıktı dizilerine sahip olan bir otomat elbette ki ne yapacağını önceden bilmeyecektir (çünkü hesaplamamıştır). Ancak, bu otomatın bazı çıktı dizileriyle elde edilebilecek her şeyi yapabileceğini düşünmek makuldür. Buradan hareketle McCarthy, sağ duyusal akıl yürütmenin de aynı şekilde işlediği sonucuna varır.[24]

Sağ duyusal akıl yürütmenin bir otomat sisteminde modellenmesinin mümkün olduğunu düşünen McCarthy'nin ana fikri, insan eyleminin nihayetinde bir çıktı dizisi olarak görülebilmesine dayanır. McCarthy bu konuyla ilgili olarak çokça bilinen "havalimanı" örneğini verir. Bir insanın havalimanına gitme eylemi, belli girdilere bağlı olarak belli çıktıların üretilmesine benzer bir eylemdir. Dolayısıyla eylemin ortaya çıkmasını mümkün kılacak bütün veriler programa girdi olarak verildiğinde eylemin üretilmesi de mümkün olacaktır.[25] Ancak programlamanın gelişmesiyle artık biliyoruz ki, belli bir eylemi gerektiren girdilerin belirlenmesi ve bilgisayar programında bir araya getirilmesi, hesap karmaşıklığı gibi pek çok soruna yol açmaktadır.

---

[22] Simon J. Han et al., "Inductive Reasoning in Humans and Large Language Models," *Cognitive Systems Research* 83 (2024): 12, https://www.sciencedirect.com/science/article/pii/S1389041723000839.
[23] John McCarthy, "Programs with Common Sense," (1960): 2.
[24] J. McCarthy and P. J. Hayes, "Some Philosophical Problems from the Standpoint of Artificial Intelligence," in *Readings in Artificial Intelligence* (Elsevier, 1981), 436.
[25] John McCarthy, "Programs With Common Sense,": 6,7

Bu sorunun ortaya çıkmasındaki en önemli etkenlerden birisi, bilgisayarların her türden düşünme tarzını *tümdengelim*e indirgemeye çalışmasıdır. Halbuki, gündelik hayatın akıl yürütme tarzlarına ilişkin vermiş olduğumuz yukarıdaki örnekler belli öncüllerin belli sonuçları gerektirdiği standart *tümdengelimli* mantık içerisinde ele alınamamaktadır. Çünkü gündelik akıl yürütmeler genellikle monoton ya da tekdüze bir şekilde ilerlememektedir. Hatta günlük hayatta eksik bilgilerden yola çıkarak çıkarım yapmak da oldukça yaygındır. Bu düşünüş biçiminde öncüller belli bir sonucu gerektirmese bile bir sonuca varılabilir. Bu durum standart mantığın "geçerlilik" ilkesinin de yer yer ihlal edilmesi anlamına gelir. Çünkü standart *tümdengelimli* çıkarımın aksine, geçerli bir çıkarımın öncül kümesine yeni bir öncül eklendiğinde çıkarım geçersiz hale gelebilmektedir. Gündelik hayatta çok kez yapılan bu tarz düşünme biçimleri için monoton olmayan mantık geliştirilmiştir.

Monoton olmayan akıl yürütmeyle ilgili çokça bilinen bir örneği, "kuşlar uçar" önermesini ele alalım. "Kuşlar uçar" önermesi monoton olmayan akıl yürütme içerisinde düşünüldüğünde "tüm" niceleyicisi önermenin başına eklenmiş halde "tüm kuşlar uçar" şeklinde anlaşılmaz. Kuşların çoğunluğu uçar şeklinde anlaşılan bu önerme daha nitelikli ve spesifik bir bilgi karşısında vazgeçmeye hazır olduğumuz genel bir bilgiyi içerir. Böylece, Tweety'nin bir kuş ve penguen olduğunu bilmek "kuşlar uçar" genel bilgisinin geçersiz olmasına imkân sağlar.[26] Örnekten de anlaşılacağı gibi, monoton olmayan mantık ile kastedilen şey, öncül ve sonuç ilişkisinin monoton ya da tekdüze olmayışıdır. Öncüllere yeni öncül eklendiğinde sonuç da değişeceği için önceki çıkarım geçersiz hale gelmektedir. Bu sebeple monoton olmayan akıl yürütmede kullanılan bilginin varlığı kadar yokluğu da önem kazanır. Turing makinesi esasına göre çalışan modern bilgisayarlar *tümdengelimli* mantık temelinde programlandığı için *tümdengelimli* olmayan düşünme biçimlerini modelleme konusunda pek çok zorluk yaşamaktadır. Buradan hareketle, Turing testini geçmek üzere programlanmış bir algoritmanın da benzer zorluklara maruz kalacağını düşünüyoruz.

Turing'in bilgisayarlara dair meşhur bir öngörüsünü aktardıktan sonra Turing testine yönelik ikinci engele geçmek istiyoruz. Turing, yaklaşık elli yıl içinde, 10^9'a yakın bir depolama kapasitesi olan ve ortalama bir sorgulayanın beş

[26] Dov M. Gabbay, John Woods and Akihiro Kanamori, *Handbook of the History of Logic*, 1st ed. (Amsterdam, Boston: Elsevier, 2004-2012), 451.

dakikadan uzun bir sorguda %70 üstü doğrulukla tahmin yürütemediği bir başarıyla taklit oyununu oynayabilecek bilgisayarların mümkün olacağını öngörmektedir. Ayrıca Turing, 20. asrın sonunda kelimelerin kullanımı ve eğitimlilerin genel görüşünde yaşanan dramatik değişim sayesinde hiçbir ihtilaf olmadan düşünen makinelerden söz edilebileceğine inanmaktadır.[27]

Hesap Karmaşıklığı ve Turing Testi

Turing testinin geçilmesi yolundaki ikinci engel hesap karmaşıklığı problemleridir. Hesap karmaşıklığında yaşanan temel sorun, bilgisayara sorulan sorudaki girdi sayısı arttıkça bilgisayarın üreteceği çıktı sayısının üstel (İng. *exponential*) bir şekilde artmasıdır. Bu konuya genellikle *Gezgin Satıcı Problemi* ya da *Satranç*, *Sudoku* gibi oyunlar örnek verilir. Hesap karmaşıklığı kuramında incelenen problemler algoritmalar aracılığıyla çözülebilen problemlerdir ve bir algoritma bir Turing makinesi tarafından hesaplanabildiğinden, karmaşıklığı ölçmek için temel hesaplama çerçevesi olarak gerekirci (İng. *deterministic*) Turing makineleri kullanılır.[28] Bir algoritmanın herhangi bir problemi etkili bir şekilde çözüp çözmediğini tespit etmek için belli ölçütler belirlenmiştir. Hesap karmaşıklığı alanının görevi, etkili bir çözümü olmayan bir problemin ne kadar maliyetle çözülebildiğini belirlemektir. Burada maliyetten kasıt, problemi hesaplamaya başlayan bilgisayarın kullanacağı bellek ve işlem zamanıdır. Bu işlem zamanı, hesaplamadaki geçişlerin ya da işlem adımlarının sayısı sayılarak belirlenir.[29]

Bilgisayar biliminde, bir problemi çözmeye çalışan bilgisayarın kullandığı kaynağın problemin büyüklüğü ile doğru orantılı olması beklenmektedir. Böyle bir doğru orantının olmadığı problemler hesap karmaşıklığına yol açmaktadır. Hesap karmaşıklığı konusuna temel karakteristiğini veren problem *P vs NP* problemidir. *P vs NP* problemi, teorik bilgisayar biliminin en önemli açık uçlu problemlerinden birisidir. Kısaca ifade etmek gerekirse *P* (İng. *polynomial*); gerekirci bir algoritma tarafından polinom zamanda çözümü bulunan problemlere verilen isimdir. *NP* (İng. *non-deterministic polynomial-time*) ise; polinom zamanda çözümü bulunamayan

[27] Turing, "Computing Machinery and Intelligence,": 442.
[28] Bill Marion, "Turing Machines and Computational Complexity," *The American Mathematical Monthly* 101, no. 1 (1994): 63.
[29] Bill Marion, "Turing Machines and Computational Complexity", 63.

ancak önerilen bir çözümün doğruluğunun polinom zamanda kontrol edilebildiği problemlerdir.[30]

Np problemlerini kısa bir örnekle açıklayalım. Bir bilgisayar için herhangi bir *Sudoku* problemini çözmek oldukça maliyetli bir işlemdir. Ancak bilgisayara bir *Sudoku* probleminin -doğru ya da yanlış- bir çözümü verildiğinde bu çözümün doğruluğunu çok kısa sürede tespit edebilir. Belli başlı ölçütlere göre zor sayılabilecek bir *Sudoku*[31] probleminin ortalama çözüm süresi 2000 saniye iken verilen bir çözümün kontrolü ise 1 saniyeden daha azdır.[32] Bu örnekten de anlaşılacağı üzere, çözümü bulunamayan ama verilmiş bir çözümün doğruluğunun hızlıca yapılabildiği problemler *Np* kümesinde yer almaktadır.

Stephen Cook ve Leonid Levin'in çalışmaları sayesinde, *NP* sınıfındaki problemlerden herhangi birisi için polinom zamanlı bir algoritma mevcutsa *NP*'deki diğer problemler için de böyle bir çözümün mevcut olduğu fark edilmiştir. Böylece birbiriyle ilişkili olduğu görülen bu problem kümesi de *NP-tam* ismini almıştır.[33] Ancak birbirlerine indirgenebilen *NP-tam* problemler için polinom zamanlı çalışan gerekirci algoritmaların şimdiye kadar bulunamadığını belirtmemiz gerekir. Bunun yerine, polinom zamanda çalışan ama bazı *höristik* (İng. *heuristic*) yöntemlere dayanan algoritmalar geliştirilmektedir ki bunların etkili bir çözüm olması mümkün değildir. *Höristik* algoritmaların niçin etkili çözüm sağlamadığı konusunu ileride açıklıyoruz.

Turing testinin hesap karmaşıklığı ile olan ilişkisi özellikle zaman karmaşıklığı (İng. *time complexity*) bağlamında ortaya çıkmaktadır. Turing, 1950 tarihli yazısını yayımladıktan sonra pek çok eleştiri almıştır. Ancak 1952 yılında *BBC* tarafından kayda alınan bir tartışma programında Max Newman'ın Turing testine dair sorduğu bir soru vardır ki bu soru bizim konumuzu ilgilendiren *zaman faktörü*ne (İng. *time factor*) ilişkindir. Turing'in varsayımsal testine yönelik olarak Max Newman Turing'e şu soruyu sorar:

---

[30] Phillip A. Laplante, *Dictionary of Computer Science, Engineering, and Technology* (Boca Raton, FL: CRC Press, 2001), 336.
[31] Burada sözünü ettiğimiz *Sudoku*, 9x9 ölçüsündeki standart versiyondur. Bu ebat 10x10 ya da 11x11 olduğunda problemin zorluk düzeyi de üstel [exponential] bir biçimde artmaktadır.
[32] Ricardo Soto et al., "A Prefiltered Cuckoo Search Algorithm with Geometric Operators for Solving Sudoku Problems," *TheScientificWorldJournal* (2014).
[33] Sipser, *Introduction to the Theory of Computation*, 299.

> Hala tartışmalarımızın çoğunun gelecekteki varsayımsal makinelerin ne yapacağıyla ilgili olduğunu düşünüyorum. Bir makinenin şunu ya da bunu kolayca yapabileceğini söylemek çok iyi, ama sadece pratik bir noktayı ele alırsak, bunu yapmak için gereken zaman ne olacak?[34]

Newman'ın sorusu bir hesaplama işleminin teorik olarak yapılabilmesi ile pratik olarak ne kadar sürede yapılacağı arasındaki farka ilişkindir. Newman, *Manchester makinesi*nin[35], satranç oyununun tüm olası hamlelerini analiz ederek en iyi hamleyi bulan bir rutin oluşturmasının sadece bir saat süreceğini belirtir. Ancak bir saatte oluşturulan rutini çalıştırmanın binlerce milyon yıl almasına aldırış edilmediği takdirde bu bir saate sevinilebileceğini belirtir. Çünkü Newman'a göre makinede bir problemi çözmek, bunu şimdi ile sonsuzluk arasında değil, makul (İng. *reasonable*) bir süre içinde yapmanın bir yolunu bulmak demektir. Üstelik, bu sadece gelecekteki iyileştirmelerle halledilecek teknik bir ayrıntı da değildir. Newman'a göre mevcut makinelerde binlerce, milyonlarca yıl sürecek çalışmaların geleceğin makinelerinde bir çırpıda yapılacağını varsaymak, bilim kurgu alemine geçmektir.[36] Turing, Newman'ın bu soruda ifade ettiği zorluğun farkında olacak ki, zaman faktörüne dair sorunun tüm gerçek teknik zorlukları içerdiğini belirtmiştir. Yine de Turing, satrancın bir insan beyni tarafından makul bir sürede oynanabilmesi gerçeğine dayanarak makineler konusunda da ümitsiz olunmaması gerektiğini belirtir. Çünkü bir beynin bunu yapabildiği gerçeği, zorlukların gerçekten de göründükleri kadar kötü olmayabileceğini düşündürmektedir.[37]

Proudfoot'a göre, Turing'in *zaman faktörü* ile ilgili soruna dair yukarıdaki yaklaşımı, onun "zekâ" anlayışının matematiksel olmaktan ziyade

---

[34] Turing, Alan M. and Jack Copeland. "The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life Plus the Secrets of Enigma." (Oxford: Clarendon Press, 2010), 503.

[35] *Manchester makinesi*, Turing'in fikrinin o günlerde donanımsal olarak hayata geçirilmiş olan bir ürünüdür. Turing'in sınırsız bellekli hesaplama makinesi fikri ABD'de von Neumann ve İngiltere'de Newman tarafından yayılmıştır; bu iki matematikçi Turing'in soyut evrensel makinesinin elektronik mühendislerinin eline geçmesinden Turing'in kendisiyle birlikte büyük ölçüde sorumluydular. 1946 yılına gelindiğinde her iki ülkedeki çeşitli gruplar evrensel bir Turing makinesini donanımsal olarak oluşturmaya girişmişlerdi. İlk elektronik depolanmış program bilgisayarını hazırlama ve çalıştırma yarışını Manchester Üniversitesi kazandı ve Newman'ın Hesaplama Makinesi Laboratuvarı'ndaki 'Manchester Bebeği' 21 Haziran 1948'de ilk programını çalıştırdı. Bkz. Turing and Copeland, "The Essential Turing", 16.

[36] Turing and Copeland, "The Essential Turing", 503.

[37] Turing and Copeland, "The Essential Turing", 503, 504.

duygusal olması ile ilgilidir. Proudfoot, Turing'in sorgulayıcıya cevap verme üzerinden bir test tasarlamış olmasını zekâya yanıt-bağımlı (İng. *response dependent*) yaklaşım ile ilişkilendirir. Yanıt-bağımlı bir açıklama şu şekildedir: Gerçek dünyada, kısıtlanmamış bir bilgisayarın insanı taklit ettiği bir oyunda, *x* ortalama bir sorgulayıcıya zeki görünüyorsa *x* zekidir (ya da düşünür). Turing'in bu zekâ tanımlaması, "makineler düşünebilir mi?" sorusunu niçin "Taklit oyununda başarılı olabilecek dijital bilgisayarlar var mıdır?" sorusuyla değiştirdiğini açıklamaktadır. Aynı zamanda taklit oyununun neden makinenin davranışını ya da içsel işlemlerini değil de sorgulayıcının yanıtını test eden bir deney olduğunu da açıklamaktadır.[38]

Amerikalı bilgisayar bilimci Scott Aaronson'un 2013 yılında yazdığı bir makalenin başlığı şöyledir: Filozoflar Niçin Hesap Karmaşıklığıyla İlgilenmeli?[39] Bu makalede Aaronson, bir şeyin hesaplanabilir olduğunu bildiğimizde, bu hesaplamanın ne kadar verimli ya da etkili olduğu sorusunun felsefi açıdan çok önemli bir soru olduğunu belirtmektedir. Aaronson'a göre, hesaplanabilir olan ile hesaplama zamanı arasındaki uçurum bazen o kadar büyüktür ki, bunları sadece niceliksel değil, niteliksel uçurumlar olarak da düşünmek gerekir. Bunu hayal edebilmek için, bin basamaklı bir sayıyı yazmak ile bu sayıya kadar saymak arasındaki farkı düşünün.[40] Aaronson'un bin basamaklı sayıyı sayma eylemi üzerinden ifade ettiği niteliksel uçurum, bilgisayarların hesaplama yetisindeki niteliksel bir eksikliğe işaret etmektedir. Bu eksiklik, daha önce belirttiğimiz gibi, Newman'ın Turing'e yönelttiği sorunun daha güncel bir ifadesidir.

Michael Sipser, bir konuşmasında *Np* problem kümesindeki *Asal Çarpanlara Ayırm*a problemine çözüm aramayı "samanlıkta iğne aramaya" benzetmiştir. Sipser bu örneği verirken şu soruyu sorar: samanlıktaki iğneyi aramanın, tek tek saman saplarının altına bakmaktan daha kolay ve hızlı bir yolu var mıdır? Sipser'a göre, bir iğneyi samanlıkta aramak ne kadar zor ve zaman alıcı bir işlemse, asal çarpanlara ayırma işlemi de en az o kadar zordur. Ancak, iğneyi hızlıca bulabilmek için kullanılabilecek bir mıknatıs gibi, acaba asal çarpanlara ayırmayı kolaylaştıracak matematiksel bir yöntem veya algoritma

[38] Diane Proudfoot, "Rethinking Turing's Test and the Philosophical Implications," *Minds and Machines* 30, no. 4 (2020): 495, https://link.springer.com/article/10.1007/s11023-020-09534-7.
[39] Scott Aaronson, "Why Philosophers Should Care About Computational Complexity" içinde *Computability*, ed. B. J. Copeland, Carl J. Posy and Oron Shagrir, 261–328 (The MIT Press, 2013).
[40] Aaronson, "Why Philosophers Should Care About Computational Complexity", 264.

bulunabilir mi?[41] Sipser'ın mıknatıs örneği üzerinden anlatmaya çalıştığı matematiksel yöntem arayışı, bir problemin çözüm yolunu bize veren yordam (İng. *procedure*) ile ilgilidir. Polinom zamanda çalışan bir algoritmadan bahsedildiğinde böyle bir yordama sahip olunduğu anlaşılmalıdır. Ancak *Asal Çarpanlara Ayırma* ya da *Sudoku* problemi çözme konusunda böyle bir yordamdan yoksun oluşumuz hesap karmaşıklığına yol açmaktadır.

Peki, böyle bir yordama ya da algoritmaya sahip olunmadığında bilgisayarlar mevcut problemin çözümü konusunda nasıl bir yol izler? Birazdan açıklayacağımız üzere, etkili bir çözüm yoluna sahip olmayan bilgisayarlar için tek çıkış kapısı *kaba kuvvetle arama* (İng. *brute force search*) yöntemine başvurmaktır. Sipser, bilgisayarların geniş olasılıklar uzayındaki bir nesneyi arama işini çok hızlı bir şekilde yapabildiğini belirtir. Ancak bazı durumlarda bu arama uzayı öyle boyutlara ulaşır ki hayal edilebilecek en hızlı makineler bile arama yapmak için devasa zamana ihtiyaç duyarlar. Bu tür durumlarda pratik bir çözüme ulaşabilmek için kaba kuvvetle arama yapmaktan kaçınan bir yöntem bulmak gerekmektedir. Kısaca ifade etmek gerekirse, *P vs NP* sorusu genel olarak böyle bir yöntemin var olup olmadığını sorar.[42]

Bir bilgisayarın belirli bir problemi çözmesi istendiğinde eğer o probleme has (İng. *specific*) bir algoritma yoksa eldeki tüm çözüm yolları sırayla denenmektedir. Elinizdeki defterde karmaşık bir şekilde sıralanmış binlerce isimden oluşan listedeki bir ismi aradığınızı düşünün. Yapabileceğiniz tek şey, en baştan sona kadar tek tek listeyi taramaktır. Kaba kuvvet araması adı verilen bu yöntem bazı durumlarda işe yarasa da pek çok durumda bilgisayarların devasa bir kaynak enflasyonuna maruz kalmasına sebep olur. Çünkü bilgisayar, çözüm ile ilgili olmayan tüm diğer seçeneklere de göz atmak durumunda kalmaktır.

Hesap karmaşıklığı konusu ile ilgili olarak literatürde çokça atıf yapılan *Gezgin Satıcı Problemi*ni (İng. *traveling salesman problem*) göz önüne alalım. Problemin tarifi kısaca şöyledir: tüm şehir çiftleri arasındaki mesafelerin belli olduğu n farklı şehirden oluşan bir küme verilsin. Tüm bu şehirleri ziyaret etmek isteyen bir gezgin satıcı için en kısa rota nedir? Rota belirlenirken satıcının aynı şehirde başlayıp yine aynı şehirde turunu sonlandırdığı varsayılmaktadır.

---

[41] Michael Sipser, "Beyond Computation: The P Vs NP Problem", https://www.youtube.com/watch?v=msp2y_Y5MLE (Erişim Tarihi: Eylül 25, 2023).
[42] Michael Sipser, "The History and Status of the P Versus NP Question," içinde *Theory of Computing: Twenty-Fourth Annual ACM Symposium: Proceedings*, ed. Rao Kosaraju et al. (New York: Association for Computing Machinery, 1992), 603.

Böylece problemin çözümü, kalan n-1 şehrin tüm sıraları üzerinde örtük arama (İng. *implicit search*) yapılmasına dayanır ki bu da (n-1)! boyutunda bir arama uzayına yol açar.[43] Bu arama uzayının niçin (n-1)! boyutunda olduğunun matematiksel bir izahı vardır. Belirlenecek tur için n farklı şehir arasından başlangıç şehri olarak herhangi bir şehir seçilir. Bu şehirden başlayan bir gezgin satıcının gideceği ikinci şehir için n-1 seçenek, üçüncü şehir için n-2 seçenek vardır ve bu böyle devam eder. Bunları çarparak toplam tur maliyetine şu hesaplama üzerinden ulaşılır: (n-1)! = (n-1) . (n-2) . (n-3) . . . 3 . 2 . 1.[44] Örneğin, Türkiye'nin herhangi bir şehrinden başlayıp tüm şehirleri dolaşan en kısa rotayı dikkate alalım. Türkiye'deki 81 il için arama yapan bir bilgisayar, 119 haneli bir arama uzayına karşılık gelen 80! büyüklüğündeki hesaplama maliyetine sahiptir. Bu sebeple *Gezgin Satıcı Problemi* için (n-1)! çalışma süresini gerektiren bir algoritma kullanışlı değildir. Çünkü, 50 ya da daha fazla şehir içeren soruların çözümü, dünyadaki tüm makinelerin birleşik hesaplama gücünün ulaşamayacağı bir zaman dilimini gerektirir.[45]

Gezgin Satıcı Problemi gibi problemler ile ilgili daha verimli ve hızlı çözümler elde etmek için bilgisayar bilimciler yöntemlerini sürekli olarak geliştirmektedir. *Eniyileme* (İng. *optimization*) algoritması olarak anılan bu yöntemler belirli bir hedefi maksimize etmek veya minimize etmek için en iyi çözümü bulmaya çalışan hesaplama tekniklerdir. Bu algoritmalar, herhangi bir problemin çözümünde belli bir kriter ya da kısıtlamaya giderek en iyi sonucu elde etmeyi amaçlar.[46] Oysa esas amaçlanan şey, en iyi çözüme yakınsamak yerine, Gezgin *Satıcı Problemi* için etkili bir çözüm ya da algoritma oluşturmaktır. Bilgisayarların karar verme yöntemlerindeki bunca gelişmeye rağmen hesap karmaşıklığının giderilmesi konusundaki engelin devam ettiğini belirtmeliyiz. Bu sebeple bilgisayarlar hesap karmaşıklığı problemlerine çözüm ararken devasa hesaplama maliyetleri ile karşılaşmaya devam etmektedir. Bu maliyetlerin, Turing testindeki bilgisayarın cevaplarının gecikmesine ve testte başarısız olmasına neden olacağını düşünüyoruz. Ayrıca bilgisayarın zaman

[43] Jon Kleinberg and Éva Tardos, *Algorithm Design* (Boston, Mass., London: Pearson/Addison-Wesley, 2006), 55, 56.
[44] David L. Applegate, *The Traveling Salesman Problem: A Computational Study / David L. Applegate ... [Et Al.],* Princeton series in applied mathematics (Princeton, Oxford: Princeton University Press, 2006), 45.
[45] David L. Applegate, *The Traveling Salesman Problem: A Computational Study*, 46.
[46] J. Hromkovic, *Algorithmics for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation and Heuristics* (Berlin: Springer-Verlag Berlin and Heidelberg GmbH & Co., 2001), 96.

maliyeti konusunda sorun yaşamasının tek sebebinin hesap karmaşıklığı problemleri olmadığını da belirtelim. Turing testi gerçek zamanlı bir diyaloğa dayandığı için doğal dilin anlaşılması konusunda bilgisayarın yaşayacağı sorunlar da cevaplarda gecikmelere sebep olabilmektedir. Bu durumu daha önce monoton olmayan düşünmenin bilgisayarlar tarafından modellenemeyişi örneği üzerinden ele aldık.

Sonuç

Çalışmamızda Turing testinin geçilemeyeceği argümanı gerekçelendirilmeye çalışılmıştır. Bu gerekçenin ilk ayağını geriçıkarım gibi monoton olmayan düşünmenin bilgisayarlarda modellenmesi sorunu oluşturmaktadır. Bu sorun, Turing makinesi esasına göre çalışan modern bilgisayarların tümdengelimli mantık temelinde programlanması ile yakından ilişkilidir. Bu mantıksal temele dayanan bilgisayarlar monoton olmayan akıl yürütmelerin olduğu gündelik dili modellemekte zorluk yaşamaktadır. Dolayısıyla sorgulayıcının soracağı soruların monoton olmayan düşünme içermesi durumunda Turing testinin geçilemeyeceği gibi bir sonuç elde edilmiştir.

Turing testinin geçilemeyeceği yönündeki argümanımızın ikinci ayağını ise hesap karmaşıklığı problemleri oluşturmuştur. Kaba arama yönteminden kaçınamayan bilgisayarların maruz kaldığı hesap karmaşıklığı problemleri ile Turing testinin geçilemiyor oluşu arasında bir bağlantı olduğunu düşünüyoruz. Çünkü Turing testine tabi tutulan bilgisayara Gezgin *Satıcı Problemi* gibi bir soru sorulduğunda bilgisayarın cevabı gecikecek ve kendisini ele verecektir. Öte yandan aynı sorular bir insana sorulduğunda ise insan bu soruları bilmediğini ifade edebilecektir. Bunun en temel sebebi, bilgisayarların bazı problemlerin etkili çözümüne sahip olmadığı durumlarda kaba aramaya başvurmak zorunda kalmasıdır. Kaba aramaya maruz kalmayı gerektiren bu sorun, Sipser'ın mıknatıs örneğiyle dile getirdiği, belli bir yordamdan yoksun oluştur. Böyle bir yordam olmadığında bilgisayar doğal olarak kaba arama yöntemine başvurmakta ve hesap karmaşıklığı yaşamaktadır. Buradan, monoton olmayan düşünmeyi modelleyemeyen ve kaba arama yönteminden kaçınamayan bir bilgisayarın Turing testini geçemeyeceği sonucuna varılabilir.

KAYNAKÇA

Aaronson, Scott. "Why Philosophers Should Care About Computational Complexity." In *Computability*. Edited by B. J. Copeland, Carl J. Posy and Oron Shagrir, 261–328. The MIT Press, 2013.

Aliseda, A. *Abductive Reasoning: Logical Investigations into Discovery and Explanation / by Atocha Aliseda.* Synthese library 330. Dordrecht, London: Springer, 2006.

Applegate, David L. *The Traveling Salesman Problem: A Computational Study / David L. Applegate ... [Et Al.].* Princeton series in applied mathematics. Princeton, Oxford: Princeton University Press, 2006.

Copeland, B. J., Carl J. Posy, and Oron Shagrir, eds. Computability. The MIT Press, 2013.

De Mol, Liesbeth. *Turing Machines,* (2018), *The Stanford Encyclopedia of Philosophy,* https://plato.stanford.edu/entries/turing-machine/ (Erişim Tarihi: 2 Şubat, 2023).

Gabbay, Dov M., John Woods, and Akihiro Kanamori. *Handbook of the History of Logic*. 1st ed. Amsterdam, Boston: Elsevier, 2004-2012.

Han, Simon J., Keith J. Ransom, Andrew Perfors, and Charles Kemp. "Inductive Reasoning in Humans and Large Language Models." *Cognitive Systems Research* 83 (2024): 101155.

Holyoak, Keith J., and Robert G. Morrison. *The Oxford Handbook of Thinking and Reasoning.* Oxford library of psychology. Oxford: Oxford University Press, 2012.

Hromkovic, J. *Algorithmics for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation and Heuristics*. Berlin: Springer-Verlag Berlin and Heidelberg GmbH & Co., 2001.

John McCarthy. "Programs with Common Sense." (1960).

Kleinberg, Jon, and Éva Tardos. *Algorithm Design*. Boston, Mass., London: Pearson/Addison-Wesley, 2006.

Kosaraju, Rao, Mike Fellows, Avi Wigderson, and John Ellis, eds. *Theory of Computing: Twenty-Fourth Annual ACM Symposium: Proceedings*. New York: Association for Computing Machinery, 1992.

Laplante, Phillip A. *Dictionary of Computer Science, Engineering, and Technology*. Boca Raton, FL: CRC Press, 2001.

Marion, Bill. "Turing Machines and Computational Complexity." *The American Mathematical Monthly* 101, no. 1 (1994): 61.

McCarthy, J., and P. J. Hayes. "Some Philosophical Problems from the Standpoint of Artificial Intelligence." In *Readings in Artificial Intelligence.* Elsevier, 1981.

Michael Sipser. "Beyond Computation: The P Vs NP Problem", https://www.youtube.com/watch?v=msp2y_Y5MLE (Erişim Tarihi: Eylül 25, 2023).

Oppy, Graham and David Dowe, *The Turing Test*, (2021), *The Stanford Encyclopedia of Philosophy,* https://plato.stanford.edu/archives/win2021/entries/turing-test/>/ (Erişim Tarihi: 25 Nisan, 2024).

Piccinini, Gualtiero. "Alan Turing and the Mathematical Objection." *Minds and Machines* 13, no. 1 (2003): 23–48.

Proudfoot, Diane. "Rethinking Turing's Test and the Philosophical Implications." *Minds and Machines* 30, no. 4 (2020): 487–512.

Readings in Artificial Intelligence. Elsevier, 1981.

Sipser, Michael. "The History and Status of the P Versus NP Question." içinde *Theory of Computing: Twenty-Fourth Annual ACM Symposium: Proceedings*. Edited by Rao Kosaraju et al. New York: Association for Computing Machinery, 1992.

——— . *Introduction to the Theory of Computation*. Florence: Cengage Learning, Inc, 2012.

Soto, Ricardo, Broderick Crawford, Cristian Galleguillos, Eric Monfroy, and Fernando Paredes. "A Prefiltered Cuckoo Search Algorithm with Geometric Operators for Solving Sudoku Problems." *TheScientificWorldJournal* (2014): 465359.

Taştan, Ümit. "Gödel'in Tamamlanamazlık Teoremleri Bakımından Biçimsel Dillerde İspatlanabilirlik-Doğruluk İlişkisi." *MetaZihin: Yapay Zeka ve Zihin Felsefesi Dergisi* (2022).

Turing, Alan. "Computing Machinery and Intelligence." *Mind* LIX, no. 236 (1950): 433–460.

Turing, Alan M., and B. J. Copeland, eds. *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life Plus the Secrets of Enigma*. Oxford: Clarendon Press, 2010.