Válasz dr. Szederkényi Gábor Professzor Úr

„*Nehéz, kombinatorikus optimalizálási feladatok számítógéppel segített megoldása*"

című doktori (PhD) disszertációmhoz megküldött bírálatára

1. kérdés/megjegyzés: Hogyan lehetne figyelembe venni a tárgyak méretének eloszlására vonatkozó előzetes információt a 3. fejezetben ismertetett eljárások teljesítményének hangolása érdekében, ha az eloszlás nem egyenletes?

1. kérdésre/megjegyzésre adott válasz: A bolognai egyetem operációkutatással kapcsolatos weboldalán (UNIBO, [1]) található egy nagyon átfogó ládapakolási benchmark készlet, amely általánosan elfogadott. A weboldal fenntartói között van Silvano Martello professzor, aki az egyik legkompetensebb a gyakorlati ládapakolási feladatok megoldásában [2–5]. A 9 benchmark osztályból a Falkenauer és a Schwerin két-két további, a Scholl pedig három további halmazra bomlik, így összesen 13 különböző feladathalmazt kapunk. Az alábbiakban egy összefoglaló táblázatban (1. táblázat) ismertetem ezeket a benchmark osztályokat. A táblázatban az n jelöli a tárgyak számát, a C pedig a ládák kapacitását.

Dolgozatomban a 13 osztályból három osztállyal foglalkoztam, ezek: Falkenauer_U valamint Schwerin 1 és Schwerin 2. Ezen osztályok esetében az inputokban szereplő tárgyméretek egyenletes eloszlásúak. Más osztályok is vannak, ahol a tárgyméretek egyenletes eloszlásúak. A benchmark osztályokat a következő csoportokba tudjuk sorolni:

- Az első csoportba azokat a benchmark osztályokat sorolom, ahol a tárgyméretek egyenletes eloszlásúak. A 13 osztályból 8 ilyen. Az ilyen feladatok megoldására alkalmasak a kidolgozott algoritmusaim.
- A második csoportba a Falkenauer osztály Falkenauer_T alosztályának feladatai tartoznak. Ezek az inputok a 3-partíciós feladat mintájára készültek. Ezek nehéz inputok, pakolásuk során már egy kis hiba (amikor 2 tárgyat közös ládába pakolunk, de optimális esetben ezek nincsenek egy ládában) esetén is előfordul, hogy nem érhető el optimális megoldás. Az általam kidolgozott algoritmusok ilyen feladatok megoldására nem alkalmasak.
- A harmadik csoportba a Hard28 osztály tartozik, ebben 28 darab input van. Ezen osztály feladatai is nagyon nehezek. Az ebbe a feladatosztályba tartozó

Feladatosztály	Halmaz	Feladatok száma	n	C	Eloszlás
Falkenauer	Falkenauer_U	80	120, 250, 500	150	egyenletes
	Falkenauer_T	80	60, 120, 250	1000	perfect packing
Scholl	Scholl 1	720	50	100	egyenletes
	Scholl 2	480	50	1000	egyenletes
	Scholl 3	10	200	100000	egyenletes
Wascher	-	17	[57, 239]	10000	ad-hoc
Schwerin	Schwerin 1	100	100	1000	egyenletes
	Schwerin 2	100	120	1000	egyenletes
Hard28	-	28	160, 180, 200	1000	ad-hoc
Random	-	3840	50, 100	50, 75	egyenletes
AI	-	250	202	≤ 2500	ad-hoc
ANI	-	250	201	≤ 2500	ad-hoc
GI	-	240	1227	500000	egyenletes
Összesen:		**6195**			

1. táblázat. Benchmark feladatosztályok tulajdonságai

problémák 160, 180 vagy 200 tárgyat tartalmaznak. A láda mérete egységesen 1000. A tárgyak méretei az $[1, 800]$ intervallumból kerülnek ki. A 28 darab feladatban átlagosan a tárgyak 30%-nak a mérete nagyobb, mint a láda kapacitásának a fele. Továbbá a páros és páratlan mérettel rendelkező tárgyak aránya átlagosan 50%-50% [6]. Mivel a tárgyméretek ezeknél az inputoknál nagyon "ravasz" módon vannak összeválogatva, ezek a feladatok mohó módszerrel nem megoldhatók, vagyis az általam kidolgozott algoritmusok ilyen feladatok megoldására sem alkalmasak.

- A negyedik csoportba az *ad-hoc* módon összeállított osztályok tartoznak, az *ad-hoc* eloszlással generált inputok nagyon "trükkösek", megoldásuk nehéz. Ezen feladatok esetében a tárgyak méretei meghatározott matematikai szabályok szerint kerültek generálásra, különböző feltételeket figyelembe véve. Ha ezeket valamilyen mohó algoritmussal akarjuk megoldani, az első lépés annak tesztelése lenne, hogy az algoritmusaim a jelenlegi állapotban hány inputot képesek ezek közül megoldani. A következő pedig, hogy az algoritmus működését ezekhez az *ad-hoc* módon generált inputok tulajdonságaihoz igazítsuk. Jelenleg vizsgálatok hiánya miatt nem tudom, hogy az algoritmusok milyen hatékonysággal oldanák meg ezeket a feladatokat. Összefoglalva: *ad-hoc* eloszlás esetén működhetnek az algoritmusaim vagy valamilyen más, de hasonló elven működő algoritmus, de ezt eddig semmilyen formában nem teszteltem.

Az előbbieket összefoglalva, a 13 feladatosztály közül 8 feladatosztály esetén algoritmusaim vagy más hasonló mohó algoritmus, alkalmas lehet a feladatok megoldására. Vagyis, összesen 6195 db feladat közül 5570 db olyan feladat van, amelyeknél a tárgyméretek eloszlása egyenletes, ezekre az általam kifejlesztett algoritmusok (vagy azokhoz hasonlók) alkalmasak lehetnek. Ez az ismertetett benchmark feladatok közel 90%-a.

Olyan benchmark osztállyal nem találkoztam az irodalomban, amelyeknél a tárgyméretek valamilyen eloszlással, de nem egyenletes eloszlással (pl. normális eloszlás) lettek generálva. Többek között a következő cikkek is az előbbi benchmark készletből vesznek példákat: [7], [8], [9], [10].

Abban az esetben, ha a tárgyak méretei valamilyen eloszlással, de nem egyenletes eloszlással vannak generálva, úgy további vizsgálatok elvégzése és tulajdonságok keresése válna szükségessé. Első lépésként vizsgálni kell azt, hogy az algoritmusok jelen állapotukban hány inputot tudnak optimálisan megoldani. Utána ezek az inputok kizárhatóak, és csak a maradék feladatokkal kell foglalkozni. Ha a feladatok karakterisztikájában vannak kiaknázható tulajdonságok, akkor az algoritmus működését ehhez kell igazítani.

Mivel az UNIBO honlapon található benchmark készletből indultam ki, ezért más eloszlások vizsgálatával nem foglalkoztam, de a kérdésnek van létjogosultsága. Vizsgálata további kutatást igényelne.

Feltételezésem szerint algoritmusaim normális eloszlás esetén is hatékonyan működnének, csak ott figyelni kellene arra, hogy ha a tárgyak méretei valamely $[a, b]$ intervallumból kerülnek ki, akkor a tárgy méretek zöme az intervallum "közepéből" származik és csak kevés az egyik vagy másik széléről.

2. kérdés/megjegyzés: Milyen elméleti és/vagy számításos vizsgálatokat javasolna, hogy további információkat nyerjen a 74. oldalon megfogalmazott 1. Sejtéssel kapcsolatban?

2. kérdésre/megjegyzésre adott válasz:

A nevezett sejtés a következő:

1. Sejtés. *Legyenek* $1 \leq a < b \leq C$ *rögzített egész számok, ahol* C *a láda mérete. Tételezzük fel, hogy a tárgyak mérete véletlenszerűen, egyenletes eloszlás mellett az* $a, a+1, \ldots, b$ *egészek közül kerülnek ki. Ekkor létezik egy olyan algoritmus, amelynek futási ideje alacsony rendű polinommal felülről becsülhető és az* $O(.)$ *kifejezés együtthatója is megfelelően kicsi, továbbá a feladat optimális megoldásának valószínűsége 1-hez közelít midőn* $n \to \infty$.

A. Számítással kapcsolatos további lehetséges vizsgálatok:

A számításos módszer a Schwerin és a Falkenauer_U feladatosztályokon, a kidolgozott algoritmusokkal elvégzett vizsgálatokat jelentik.

- A Schwerin feladatosztály két halmazra bontható: Schwerin 1 és Schwerin 2. Mindkét halmazban 100 darab feladat található. A Schwerin 1 esetében a tárgyak száma $n = 100$, a Schwerin 2 esetében pedig $n = 120$. A ládák kapacitása egységesen $C = 1000$. A tárgyak méretei a [150,200] intervallumból kerülnek ki egyenletes eloszlással. Vagyis itt $a = 150$ és $b = 200$.

- A Falkenauer_U benchmark osztályban 80 feladat található, amely további négy alosztályra bomlik, mindegyik alosztályban 20 feladat van. Az alosztályokban a tárgyak száma $n = 120$, $n = 250$, $n = 500$ és $n = 1000$. A tárgyak mérete (normalizálás után) a $[133, 666]$ intervallumból kerül ki egyenletes eloszlással, a ládák kapacitása $C = 1000$. Vagyis itt $a = 133$ és $b = 666$.

Azt mindenképpen érdemes lenne további vizsgálatoknak alávetni, hogy az $[a, b]$ intervallum szélesítése vagy szűkítése hogyan befolyásolja a feladat nehézségét. Ekkor tehát a feladatosztályokhoz tartozó intervallumokat (amelyekből a tárgyak méretei kerülnek ki) változtatjuk, azaz az alsó és felső határokat növeljük vagy csökkentjük. Majd megvizsgáljuk, hogy egyes esetekben a kidolgozott algoritmusok miként viselkednek. Feltételezhető, hogy minnél szűkebb ez az intervallum, annál könnyebb hatékony algoritmust kidolgozni (amelyik nagy százalékban optimális megoldást ad). Ezt a feltételezést alátámasztják az elvégzett vizsgálataink, hiszen a Schwerin benchmarkok esetében szűk ez az intervallum, és minden feladatot optimálisan meg tudtunk oldani, viszont a Falkenauer_U osztály esetén az intervallum tágabb volt, és a 80-ból csak 72-t tudtuk optimálisan megoldani.

Természetesen a Falkenauer_U osztályra kidolgozott algoritmusomat lehetne finomítani, vagyis nagyobb futási idő árán valószínűleg módosítható úgy, hogy több inputot oldjon meg optimálisan.

Nyilvánvalóan az sem mindegy, hogy az $[a, b]$ intervallum az $[1, C]$ intervallum melyik részén helyezkedik el. Legyen például a ládaméret $C = 1000$, legyen továbbá

$b - a = 99$. Ez esetben $[a, b] = [1, 100]$ esetén kicsi tárgyak vannak. Ezekre valószínűleg kidolgozható hatékony algoritmus. Másik végletként legyen $[a, b] = [501, 600]$ (vagy ennél is nagyobb tárgyak). Ez esetben nyilvánvalóan minden tárgy külön ládába kerül és a feladat triviálisan megoldható. Ha például $[a, b] = [401, 500]$, akkor minden ládába pontosan két tárgy kerül. Ha $[a, b] = [451, 550]$, akkor ismert, hogy az FFD algoritmus optimális megoldást ad. Röviden, más algoritmusra lehet szükség attól függően, hogy az $[a, b]$ intervallum hol helyezkedik el az $[1, C]$ intervallumon belül. Ilyen és ehhez hasonló vizsgálatokat nem végeztünk, de ez lenne a további számítógépes kutatások iránya.

B. Elméleti vizsgálatok:

Tudomásunk szerint korábban még nem történtek olyan elméleti vizsgálatok, amelyek pontosan ezt a kérdést vizsgálnák, mint ami a sejtésben szerepel. Hasonló vizsgálat történt viszont, ezt az alábbiakban részletezzük.

1. Tétel: *Tegyük fel, hogy adott a ládapakolási feladat egy inputja, ahol d különböző tárgyméret van, a különböző méretekből a tárgyak multiplicitása pedig $b_1, b_2, \ldots, b_d$. (A ládaméret legyen továbbra is C, a tárgyak mérete $[1, C]$ közötti egész szám.) Legyen továbbá Δ a maximuma a C ládaméretnek és a legnagyobb multiplicitásnak. Ez esetben a ládapakolási feladat (optimálisan) megoldható $(\log \Delta)^{2^{O(d)}}$ lépésszámú algoritmussal.*

A tétel bizonyítása a [11, 12]-ben szerepel, és Klaus Jansen is hivatkozik a tételre a [13] előadásában. A tételben megfogalmazott állítás jelenleg a legerősebb, ami a ládapakolási feladat optimális megoldásával kapcsolatos. Az idézett cikkek első szerzője Michel Goemans, az MIT professzora, Klaus Jansen is a téma kiváló szakértője.

Nagyjából ennyit lehet tudni elméleti szempontból a sejtéssel kapcsolatban. Megjegyzem, hogy a tételben szereplő lépésszám nem függ n-től, tehát a tárgyak számától, de a konstans nagyon nagy lehet, ha a C nagy. Vagyis praktikus szempontból ez a tétel nem igazán használható, mert a benne szereplő optimális algoritmus a mi algoritmusainkhoz képest nagyon lassú.

Példa: Tekintsük a Schwerin osztályt. Itt $C = 1000$, így $\Delta \approx 10$, $d \approx 50$, mivel nem ismerjük a nagy ordóbeli konstanst, vegyük úgy, hogy $O(d) \approx 100$. Ekkor $2^{100} = (2^{10})^{10} \approx 10^{30}$, vagyis a tételbeli konstans $10^{10^{30}}$, ami iszonyúan nagy szám: $10^{1000000000000000000000000000000}$.

Ha ezt összevetjük azzal, hogy a világegyetemben levő összes atom száma bizonyos becslések szerint nagyjából 10^{50} [14], más forrás szerint 10^{80} [15], az 1. tételbeli optimális algoritmus lépésszáma meglehetősen nagy.

Ehhez képest az algoritmusaim rendkívül gyorsak, futási idejük kevesebb, mint 1 másodperc. Ennek természetesen ára van, mégpedig az, hogy nincs arra garancia, hogy minden inputot optimálisan megoldanak. Ezzel együtt a Schwerin osztálynak mind a 200 feladatát, a Falkenauer_U osztály esetén pedig durván 90%-át optimálisan tudták megoldani.

3. kérdés/megjegyzés: Milyen megfontolások alapján határozta meg a 4.10. táblázatban szereplő paraméterbeállításokat?

3. kérdésre/megjegyzésre adott válasz:
Alább mellékelem a 4.10. táblázatot.

Beállítás	**K**	$\alpha_k, k = 1, \ldots, K$	β
M0	1	[200]	200
M1	4	[100; 200; 300; 400]	200
M2	5	[100; 200; 300; 400; 500]	200
M3	2	[100; 100]	200
M4	3	[100; 100; 100]	200
M5	4	[100; 100; 100; 100]	500
M6	4	[225; 225; 230; 230]	560
M7	4	[200; 200; 200; 200]	600
M8	4	[200; 200; 300; 400]	350

2. táblázat. MMask kézi paraméter-beállításai (4.10-es táblázat másolata)

A paraméterek beállításait többféleképpen kipróbáltam. Az egyes próbák a következők voltak:

- az α vektor komponensei egyenlőek, amit kisebb és nagyobb értékekre is megvizsgáltam (M3, M4, M5 és M7 beállítások),
- ha az előbbi beállítási stratégiát nem találtam elég hatékonynak, akkor lépcsőzetes beállítással próbálkoztam (M1 és M2), azaz az α komponensei egyenletes mértékben növekednek,
- ahol egyik előző beállítás sem volt megfelelően hatékony, úgy az előbbi kettő kombinációját választottam (M6 és M8), vagyis pl. M6 esetén az első két komponens egyenlő, a harmadik és negyedik komponens is egyenlő, de az előzőeknél kicsivel nagyobb. Az M8 esetén pedig az első két komponens egyenlő, de inenntől egyenletesen növekednek.

A fenti manuális beállítások valamelyike mindig segített, így részletesebb vizsgálatokat nem folytattam, mert az eredményekkel elégedett voltam. Természetesen a paraméter beállításokat lehetett volna tovább finomítani.

A másik oka annak, hogy megelégedtünk ezekkel a paraméter beállításokkal az az, hogy tudtuk, hogy ezeket még tovább fogjuk finomítani metaheurisztika alkalmazásával. Vagyis a paraméterek a végső értéküket egy két fázisú optimalizálás során kapják meg. Első fázis egy durva optimalizálás (ezt végeztem kézzel), a második fázis pedig az első fázis eredményének tovább javítása metaheurisztika által.

Összefoglalva a fenti paraméter beállításokkal elégedett voltam, mert a céljainknak megfelelő volt.

4. kérdés/megjegyzés: A szomszédságon alapuló keresés és a röviden felvázolt genetikus algoritmus mellett még milyen lehetőségeket lát az MMask algoritmus paramétereinek optimalizálására, javítására? Hogyan kezelné az algoritmus egész paramétereit olyan esetben, amikor magában az optimalizálásban folytonos változók szerepelnek?

4. kérdésre/megjegyzésre adott válasz: Az MMask algoritmusnak három paramétere van, amelyek az alábbiak:

- K - az egy időben nyitott ládák maximális számát meghatározó pozitív egész szám ($K > 0$),
- α - K-dimenziós nemnegatív vektor ($0 \leq \alpha_k \leq C$ $\forall k$-ra),
- β - egy pozitív egész szám ($\beta > 0$).

Az MMask működésének alapja egy elfogadó-elutasító politika. Az algoritmus elfogadja a soron következő tárgy pakolását, ha a pakolás után az adott láda töltöttsége az elfogadó tartományba esik. Ellenkező esetben elutasítja a pakolást. Az elfogadó és elutasító intervallumok az alábbiak szerint kerültek meghatározásra:

- Elfogadó tartomány a k. láda esetén ($1 \leq k \leq K$): $[0; C - \alpha_k] \cup [C; C + \beta]$
- Elutasító tartomány a k. láda esetén ($1 \leq k \leq K$): $(C - \alpha_k; C) \cup (C + \beta; \infty)$

A paraméterek optimalizálására az egyszerűbb megoldások közül a *grid search* és a *random search* eljárásokat lehetne kipróbálni. A *grid search* úgy gondolom, hogy kevésbé járható út, mert például $C = 1000$ ládaméret mellett, ha az *alpha* értékei a [100,600] intervallumból kerülnek ki (azaz se nem túl kicsik, se nem túl nagyok), akkor is 500^5 eset van. Ezeken kívül még meg lehet vizsgálni a *variable neighborhood search* eljárást is, amelynek a lényege a szomszédság szisztematikus megváltoztatása. Itt kipróbálható lenne az, hogy az α és β értékeket az optimalizálás során előre meghatározott stratégia mentén változtatjuk. Lehetne úgy, hogy előbb egy részletes keresést indítunk, ahol 100-asával változik az α és a β is, és utána 10-es léptékkel, végül 1-es léptékkel.

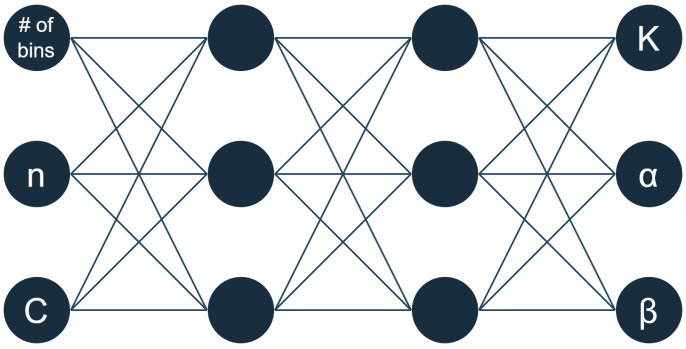
Egy további megoldás lehet egy neurális háló alkalmazása a szükséges paraméterek meghatározására. Konkrét gyakorlati feladatok esetében paraméterek meghatározására használtak neurális hálót például a [16, 17]-ben.

Az általam vizsgált esetben a neurális háló bemenete a felhasznált ládák száma, tárgyak száma (n) és a láda kapacitása (C). Első lépésben, kézi beállítással generálni kell egy nagy teszthalmazt (pl. 1000 elemű halmaz), ahol szerepelnek a paraméterek kézi beállításai, az eredményül kapott (felhasznált) ládaszám, a tárgyak száma és a láda kapacitása. A hálózat kimenete pedig a K, α és β paraméterek. A hálózat méretét tekintve egy bemeneti, két rejtett és egy kimeneti réteget tartalmazhatna (valószínűleg ez már elegendő lenne, de ez nem került kipróbálásra eddig). A két rejtett réteggel rendelkező hálózat már képes a bemenetnek bármilyen függvényét reprezentálni megfelelő rétegenkénti neuronszámmal. Természetesen ha

mégsem elegendő, akkor innen indulva tovább bővíthető a hálózat. A rétegeken belüli neuronszám ugyancsak tesztekkel beállítható. A neuronszám növelése manuális bővítést jelent. A hálózat hiba visszaterjesztéssel működne.

Mivel a neurális háló által kimeneti értékként kapott K, α és β valós számok, ezeket egészekre kell kerekíteni. Valószínűnek tartom, hogy amikor az α komponenseit és β értékét kerekítjük (pl. lefelé), azzal nem követünk el nagy hibát. Ugyanis ha feltételezzük, hogy például a ládaméret $C = 1000$ (amint a vizsgálataimban volt), akkor ha α valamelyik komponensére mondjuk 250 és 251 közötti számot ad az optimalizálás, akkor nagyjából mindegy, hogy ezt lefelé vagy fölfelé kerekítjük: ez nem fogja lényegesen befolyásolni az MMask algoritmus hatékonyságát.

Más a helyzet a ládaszámmal, vagyis a K értékével. K értéke ugyanis jellemzően kicsi. Ha például K értéke a neurális hálóval végzett optimalizálás végén például $K = 3,5$, akkor ezt lefelé vagy felfelé kell majd kerekíteni, de itt nem tudhatjuk, hogy melyik érték lehet igazán jó ($K = 3$ vagy $K = 4$). Ez esetben az algoritmus validációja során mindkét értéket kipróbálnám.

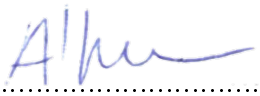


1. ábra. Alkalmasnak vélt neurális háló modellje

Fontos megjegyezni, hogy a mi esetünkben nem egy input van, hanem egy osztály több inputtal. Egy paraméter beállítással kell végigfuttatni az osztályban található minden feladatra az algoritmusokat.

Szeretném megköszönni dr. Szederkényi Gábor professzor úrnak, hogy elvállalta és elkészítette a bírálatot. A bírálatban megfogalmazott kérdések és észrevételek nagyon hasznosak voltak számomra.

Veszprém, 2023. január 9.



.....

Ábrahám Gyula
doktorandusz
Pannon Egyetem
Informatikai Tudományok Doktori Iskola

Hivatkozások

[1] Bin packing benchmarks of homepage unibo. `http://or.dei.unibo.it/library/bpplib`. Hozzáférés: 2023.01.10.

[2] A. Lodi, S. Martello, and M. Monaci. Two-dimensional packing problems: A survey. *European journal of operational research*, 141(2):241–252, 2002.

[3] S. Martello, D. Pisinger, and D. Vigo. The three-dimensional bin packing problem. *Operations research*, 48(2):256–267, 2000.

[4] E. G. Coffman, G. Galambos, S. Martello, and D. Vigo. Bin packing approximation algorithms: Combinatorial analysis. In *Handbook of combinatorial optimization*, pages 151–207. Springer, 1999.

[5] S. Martello and P. Toth. Lower bounds and reduction procedures for the bin packing problem. *Discrete applied mathematics*, 28(1):59–70, 1990.

[6] J. Pérez, R. de la Rosa, H. Castillo, and D. Vilariño. A storage pattern-based heuristic algorithm for solving instances of hard28 datasets for the bin packing problem. *Computación y Sistemas*, 22(1):235–244, 2018.

[7] I. Borgulya. A hybrid evolutionary algorithm for the offline bin packing problem. *Central European Journal of Operations Research*, 29(2):425–445, 2021.

[8] F. Brandão and J. P. Pedroso. Bin packing and related problems: General arc-flow formulation with graph compression. *Computers and Operations Research*, 69:56–67, 2016.

[9] K. Sim, E. Hart, and B. Paechter. A lifelong learning hyper-heuristic method for bin packing. *Evolutionary computation*, 23(1):37–67, 2015.

[10] Mohamed A. Basset, Gunasekaran Manogaran, Laila A. Fatah, and Seyedali Mirjalili. An improved nature inspired meta-heuristic algorithm for 1-d bin packing problems. *Personal and Ubiquitous Computing*, 22(5):1117–1132, 2018.

[11] Michel X Goemans and Thomas Rothvoß. Polynomiality for bin packing with a constant number of item types. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 830–839. SIAM, 2014.

[12] Michel X Goemans and Thomas Rothvoss. Polynomiality for bin packing with a constant number of item types. *Journal of the ACM (JACM)*, 67(6):1–21, 2020.

[13] Bin Packing Seminar Series, K. Jansen, New Algorithmic Results for Bin Packing and Scheduling, 17 March 2021. http://www.inf.u-szeged.hu/bpseminar/KlausJansenPresentation.pdf. Hozzáférés: 2023.01.19.

[14] How many atoms are in the world? https://sciencenotes.org/how-many-atoms-are-in-the-world/. Hozzáférés: 2023.01.20.

[15] Az atomok száma az univerzumban. https://hu.eferrit.com/az-atomok-szama-az-univerzumban/. Hozzáférés: 2023.01.20.

[16] D. Katherasan, Jiju V. Elias, P. Sathiya, and A. Noorul Haq. Simulation and parameter optimization of flux cored arc welding using artificial neural network and particle swarm optimization algorithm. *Journal of Intelligent Manufacturing*, 25(1):67–76, 2014.

[17] M. Mohammadi, M. Lakestani, and MH. Mohamed. Intelligent parameter optimization of savonius rotor using artificial neural network and genetic algorithm. *Energy*, 143:56–68, 2018.