# Gručenje v hadronske curke

*Praktikum strojnega učenja v fiziki*

Domen Vaupotič

## 1 Hadronski curki

V nalogi obravnavamo trke med visokoenergijskimi curki protonov, ki jim po začetni interakciji sledi proces hadronizacije. Pri tem nastane ogromna količina hadronov, ki v kolimiranih curkih letijo v okoliški prostor ter jih zaznamo z detektorji. Namen naloge je z algoritmi strojnega učenja obravnavati izmerjene delce ter jih smiselno združiti v manjše število curkov. Algoritme za združevanje delcev v grobem delimo v dve skupini: algoritmi *zaporednega združevanja* (rekombinacije) curkov in *stožčasti* algoritmi. Pri zaporednem združevanju identificiramo bližnje pare delcev in jih združimo v curek, kar iterativno ponavljamo do nekega končnega nabora curkov. Pri stožčastih algoritmih združujemo delce, ki ležijo znotraj stožca. Interakcija QCD in proces hadronizacije namreč le malo spremenita smer energijskega toka, zato so stožci smiselna reprezentacija prvotnih produktov trka.

Delce (in curke) bomo obravnavali relativistično, torej s četvercem energije in gibalne količine:

$$p^\mu = (E, \boldsymbol{p}) = (E, p_x, p_y, p_z),$$

pri čemer velja zveza $E = \sqrt{m^2 + |\boldsymbol{p}|^2}$. Zaradi geometrije procesov je smiselno uvesti drugačen cilindrično-sferični koordinatni sistem in nekoliko preoblikovati četverec. Za primarno os $z$ proglasimo smer prvotnih dveh žarkov protonov, pravokotno na njo (in stikajoč se z mestom trka) pa leži transverzalna ravnina. Uvedemo polarni kot $\theta$, merjen od os $z$, ter azimutni kot $\phi$. Sedaj vektor gibalne količine projiciramo na transverzalno ravnino in velikost projekcije imenujemo transverzalna gibalna količina, $p_T$. Uvedemo še količino, imenovano *rapidnost*, ki v splošnem podaja hitrost delca ($w = \operatorname{Artanh} v = \operatorname{Artanh}(|\boldsymbol{p}|/E)$), v kontekstu eksperimentalne fizike delcev pa jo izračunamo v transverzalni smeri: $y = \frac{1}{2} \log \frac{E+p_z}{E-p_z}$. Kinematične podatke o delcu oz. curku sedaj združimo v vektor $(E, y, \phi, m)$. Zaradi velikih hitrosti delcev in posledično energij si lahko v eksperimentalni fiziki delcev privoščimo približek majhnih mirovnih mas, torej

$$E = \sqrt{m^2 + |\boldsymbol{p}|^2} \approx |p|.$$

Posledično se nam poenostavi še rapidnost, tako da iz nje pridelamo *psevdorapidnost*:

$$\eta = -\ln \tan \frac{\theta}{2} = \frac{1}{2} \frac{|\boldsymbol{p}| + p_T}{|\boldsymbol{p}| - p_T} = \operatorname{Artanh} \frac{p_T}{|\boldsymbol{p}|}.$$

Gibalne količine tako končno zapišemo z vektorjem $(p_T, \eta, \phi)$. Zapišimo še transformacijo

koordinat:

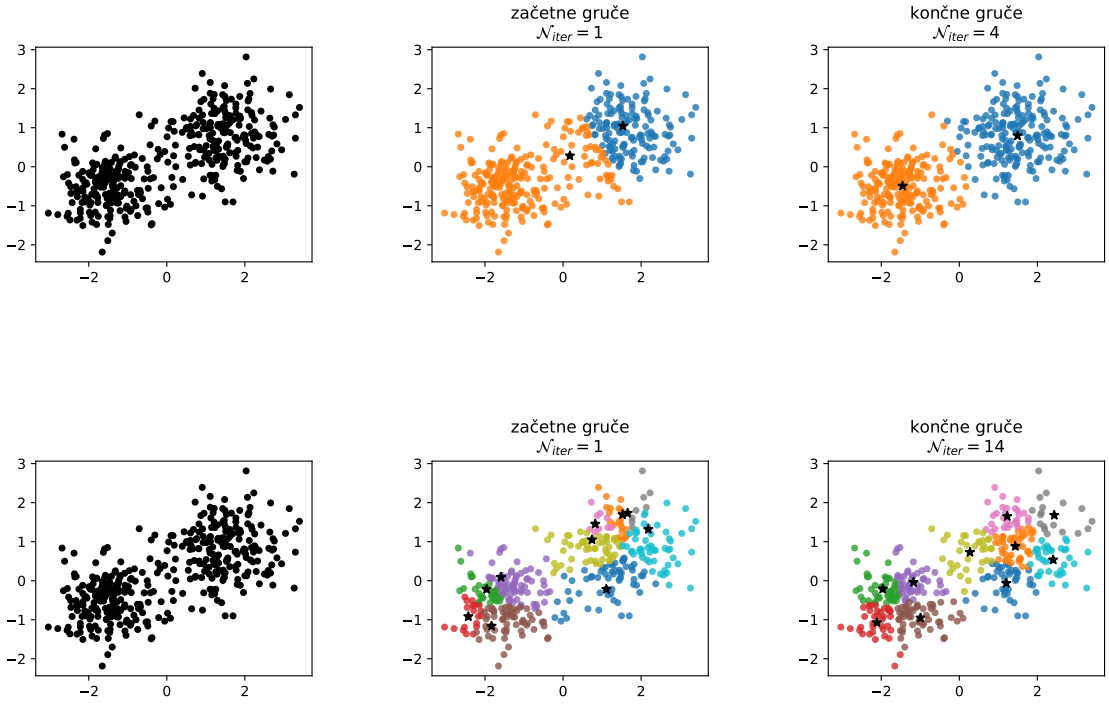
$$
\begin{aligned}
p_x &= p_T \cos\phi, \\
p_y &= p_T \sin\phi, \\
p_z &= p_T \sinh\eta, \\
|\boldsymbol{p}| &= p_T \cosh\eta.
\end{aligned}
$$

# 2 Gručenje

## 2.1 Gručenje $k$-voditeljev

Najpreprostejši algoritem za gručenje je metoda $k$-voditeljev. Algoritem je iterativen in konvergira k rešitvi, pri kateri je vsota razdalj (po neki izbrani metriki) točk od centroidov skupin (lokalno) minimalna. Ker algoritem začnemo z naključno izbiro začetnih položajev centroidov gruč, algoritem ne konvergira zmeraj v isto rešitev.

Na sliki 1 je prikazan primer gručenja naključno žrebanih točk v ravnini.




Slika 1: Gručenje podatkov, žrebanih po vsoti dveh gaussovskih porazdelitev, v 2 gruči (zgoraj) in 10 gruč (spodaj) z metodo $k$-voditeljev.

## 2.2 Hierarhično gručenje

Medtem ko pri metodi $k$-voditeljev dobimo $k$ končnih disjunktnih gruč, si pri hierarhičnem gručenju želimo točke med seboj povezati v hierarhične gruče. V splošnem so algoritmi aglomerativni (*bottom-up*), pri katerih sprva vsaka točka leži v svoji gruči, te pa nato združujemo v večje gruče; ali razdelilni (*top-down*), pri katerih vse podatke sprva obravnavamo kot eno gručo, to pa nato delimo na manjše gruče. V splošnem imajo algoritmi

časovno zahtevnost $\mathcal{O}(N^3)$, za gručenje pa moramo izbrati tudi ustrezno metriko med točkami in vezavno metriko (*linkage*) med gručami.

V nalogi bomo obravnavali primer aglomerativnega algoritma, in sicer algoritem invariatne transverzalne gibalne količine (*longitudinally-invariant $k_t$ algorithm*). Pri tem algoritmu definiramo simetrično razdaljo med parom delcev $i$ in $j$:

$$d_{ij} = d_{ji} = \min(p_{ti}^2, p_{tj}^2)\frac{\Delta R_{ij}^2}{R^2},$$

pri čemer je $\Delta R_{ij}$ evklidska razdalja med točkama v ravnini $y - \phi$ (oziroma $\eta - \phi$): $\Delta R_{ij}^2 = (\eta_i - \eta_j)^2 + (\phi_i - \phi_j)^2$, količina $R \sim \mathcal{O}(1)$ pa je parameter, imenovan *polmer curka*, in določa kotno razširjenost curka. Dodatno definiramo še razdaljo vsakega delca $i$ od glavnega žarka:

$$d_i = p_{Ti}^2.$$

Algoritem lahko izvajamo v dveh različicah, in sicer *izključujoči* (*exclusive*) ali *vključujoči* (*inclusive*). Pri izključujoči različici moramo definirati mejno vrednost razdalje $d_{cut}$, pri kateri zaključimo združevanje delcev. V nalogi pa se bomo ukvarjali z vključujočo različico, pri kateri ni treba definirati mejne vrednosti, saj delce iterativno združujemo v curke, vse dokler nam ne zmanjka prostih delcev.

Algoritem lahko sedaj nekoliko posplošimo, tako da v metriko uvedemo splošen parameter $p$:

$$\begin{aligned} d_{ij} &= \min(p_{ti}^{2p}, p_{tj}^{2p})\frac{\Delta R_{ij}^2}{R^2}, \\ d_i &= p_{ti}^{2p}. \end{aligned}$$

Izbira $p = 1$ nas povrne v prvotni algoritem invariantne transverzalne gibalne količine (imenovan tudi algoritem $k_t$), izbiro $p = 0$ imenujemo algoritem Cambgridge/Aachen, izbiro $p = -1$ pa algoritem anti-$k_t$. Izkaže se, da algoritem anti-$k_t$ pravzaprav daje podobne rezultate kot stožčasti algoritmi. To lahko pokažemo na sledeči način. Zamislimo si, da imamo v sistemu nekaj trdih delcev (takšnih, ki imajo velik $p_T$) in veliko mehkih delcev (takih z majhnim $p_T$). Razdalja med trdim in mehkim delcem je (zaradi predfaktorja min) odvisna le od transverzalne gibalne količine trdega delca in razdalje $\Delta R$ med obema delcema. Nasprotno pa bo razdalja med dvema mehkima delcema zelo velika (majhen $p_T$, torej velik $1/p_T^2$). Opazimo torej, da se bodo mehki delci začeli mnogo prej združevati z okoliškimi trdimi delci kot z drugimi mehkimi delci. Trd delec, ki na razdalji $2R$ nima nobenega drugega sosednjega trdega delca, bo tako v svoj curek zbral vse okoliške mehke delce in nastal bo stožčasti curek s polmerom $R$. Bistveno je torej, da na obliko curka vplivajo le trdi delci, medtem ko mehki nimajo veliko vpliva (infrardeča varnost).

## 2.3 Rekombinacija curkov

Ničesar še nismo povedali o tem, kako dva delca združimo v en curek. Izbrati moramo torej način, kako dva štirivektorja združiti v en štirivektor. Izkaže se, da to lahko storimo na različne načine.

Najenostavnejša je shema $E$ (`E_scheme`), pri kateri enostavno seštejemo vse komponente obeh štirivektorjev:

$$p^\mu = (E_1 + E_2, \boldsymbol{p}_1 + \boldsymbol{p}_2).$$

Uporaba takšne sheme je zelo pogosta in tudi standardni način pri eksperimentih na LHC. Za trke protonov lahko izberemo tudi eno izmed nabora shem, pri kateri najprej vse štirivektorje naredimo brezmasne (reskaliramo energijo ali gibalno količino, da postaneta enaki), nato pa naredimo uteženo aritmetično sredino med komponentami obeh vektorjev:

$$\begin{aligned} p_{Tk} &= p_{Ti} + p_{Tj} \\ y_k &= (w_i y_i + w_j y_j)/(w_i + w_j) \\ \phi_k &= (w_i \phi_i + w_j \phi_j)/(w_i + w_j). \end{aligned}$$

Pri tem lahko izberemo količino, ki jo predstavljata uteži $w_i$ in $w_j$. Imenujemo jih torej:

- shema $p_T \implies w_i = p_{Ti}$ (reskaliramo energijo),
- shema $p_T^2 \implies w_i = p_{Ti}^2$ (reskaliramo energijo),
- shema $E_T \implies w_i = E_{Ti}$ (reskaliramo gibalno količino),
- shema $E_T^2 \implies w_i = E_{Ti}^2$ (reskaliramo gibalno količino).

Obstajajo še različice, pri katerih izpustimo prvotno reskaliranje, s čimer dobimo shemi (`BIpt_scheme` in `BIpt2_scheme`), ki sta invariantni na longitudinalne potiske. Podobno lahko posplošimo shemi $p_T$ in $p_T^2$ v splošno shemo $w_i = p_T^n$, ki se v limiti $n \to \infty$ izrodi v shemo WTA (*winner-takes-all*).

# 3 Implementacija

Gručenje sem izvajal v programskem jeziku Python. Implementiral sem lastni algoritem za gručenje $k$-voditeljev (`my_kmeans`) ter lastni algoritem za hierarhično gručenje (`my_hca`), ki je omogočal izbiro poljubnih parametrov $p$ in $R$ ter rekombinacijski shemi $E$ in $p_T$. Dodatno sem rezultate primerjal z algoritmom `kmeans` iz knjižnice `scikit-learn`.

## 3.1 Knjižnica `pyjet`

Čeprav lastne implementacije dajejo pravilne rezultate, imajo zaradi naivnega implementiranega algoritma zelo neugodno časovno zahtevnost, tipično reda $\mathcal{O}(N^3)$. Za hitrejše izračune je bila razvita knjižnica `fastjet`, ki ima boljšo časovno zahtevnost $\mathcal{O}(N \ln N)$. To doseže s trikom spremembe simetrične metrike v nesimetrično ($d_{ij} \neq d_{ij}$), pri čemer pa se najmanjša izmed razdalj ne spremeni. Z uvedbo nesimetrične metrike se zmanjša število razdalj, ki jih je treba izračunati v vsaki iteraciji. Dodatne pohitritve doseže knjižnica z geometrijskimi triki, s čimer ponovno zmanjša število potrebnih izračunov razdalj.

Knjižnica je zapisana v programskem jeziku C++, a obstaja povezljiva knjižnica za uporabo v Pythonu, imenovana `pyjet`. Ker sem med testiranjem svojega algoritma naletel na težave in neskladja (v navodilih vaje je bila namreč podana rekombinacijska shema $p_T$, medtem ko je v knjižnici `fastjet` privzeta rekombinacijska shema $E$), sem želel preveriti delovanje obeh rekombinacijskih shem. Žal knjižnica `pyjet` ni omogočala izbire rekombinacijske sheme, zato sem to možnost dodatno implementiral. Implementacijo sem naložil (*pull request*) tudi v izvirni repozitorij GitHub[1], za katero upam, da jo bodo razvijalci sprejeli.

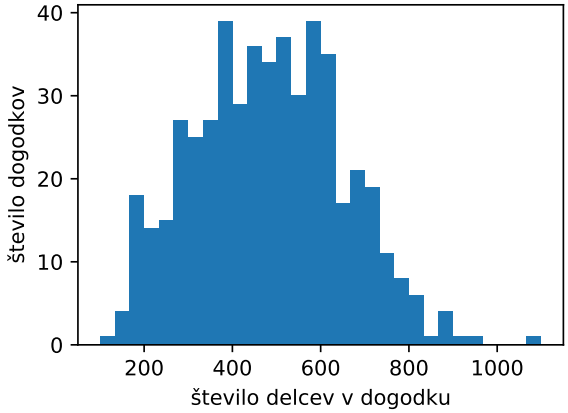
[1] `https://github.com/scikit-hep/pyjet/pull/40`

# 4 Nabor podatkov

V nalogi bomo obravnavali 500 simuliranih trkov dveh protonov, ki razpadeta v Higgsov bozon, ta pa v dva $b$-kvarka. V vsakem izmed trkov imamo nekaj sto delcev (povprečno 482), za katere poznamo kinematični vektor $(p_T, \eta, \phi, m)$. Histogram števila delcev po dogodkih prikazuje slika 2.

Namen gručenja je torej izluščiti curka, ki predstavljata prvotna $b$-kvarka, nato pa rekonstruirati maso Higgsovega bozona, $m_H = 125\,\mathrm{GeV}$. Maso rekonstruiramo iz relativistične invariante, ki jo dobimo s seštevanjem dveh curkov z največjo transverzalno gibalno količino:

$$m_H^2 = (p_1 + p_2)^2$$
$$m_H = \sqrt{p_{T1}^2 + p_{T2}^2 - 2\boldsymbol{p}_1 \cdot \boldsymbol{p}_2}$$




Slika 2: Histogram števila delcev v simuliranih dogodkih.

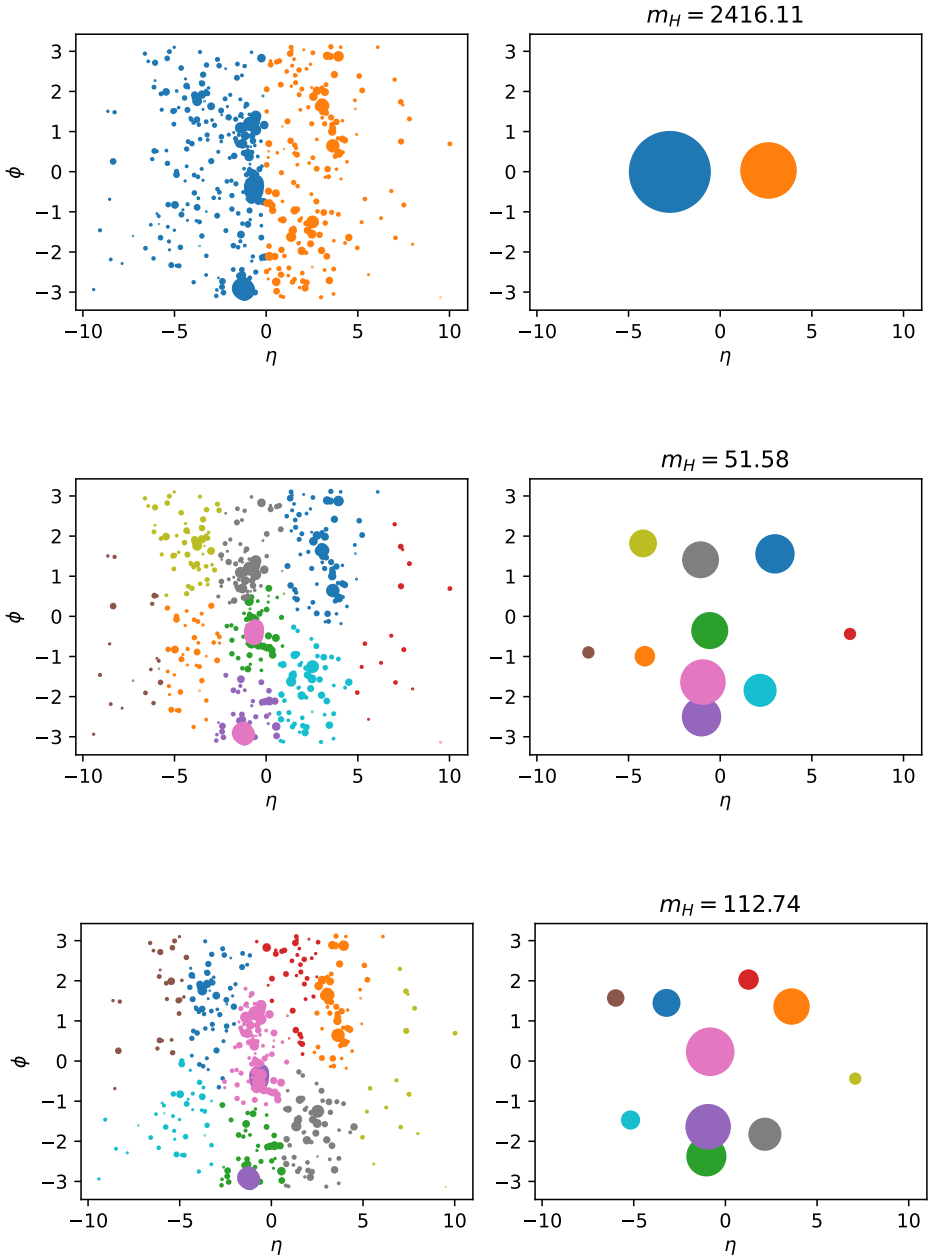
# 5 Rezultati

## 5.1 Gručenje $k$-voditeljev

Oglejmo si za začetek rezultate gručenja s $k$-voditelji. Na sliki 3 so prikazani rezultati gručenja prvega simuliranega dogodka v 2 in v 10 gruč. Opazimo, da gručenje v 2 curka ne da smiselnega rezultata, medtem ko je gručenje v 10 curkov bližje pravi vrednosti. Lepo je razvidna nedeterminičnost algoritma, saj dobimo pri različnih ponovitvah povsem drugačne rezultate. Naj omenim, da sem gručenje izvajal v vseh treh dimenzijah (torej $p_T, \eta, \phi$), čeprav bi morda bilo smiselno meriti razdaljo med curki zgolj v ravnini $\eta - \phi$.

## 5.2 Infrardeča varnost

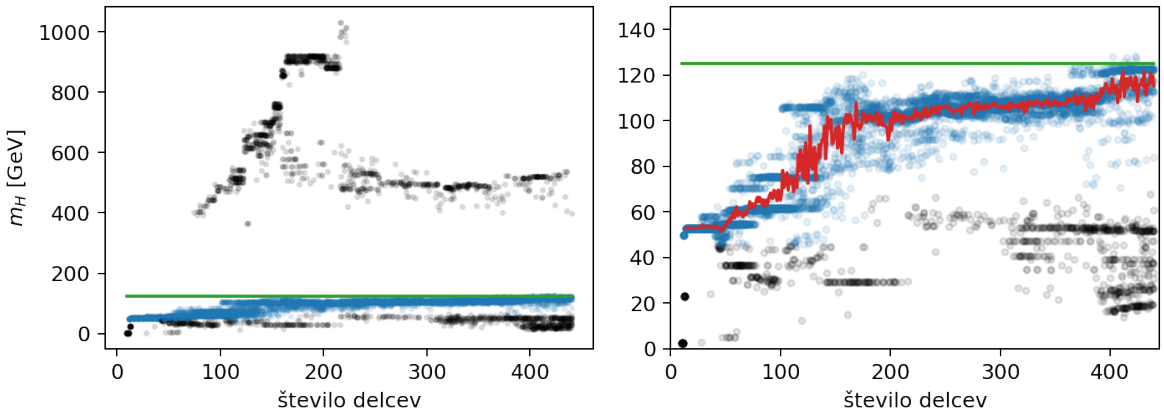
Od algoritma za gručenje curkov pričakujemo, da bo neobčutljiv na dodajanje mehkih delcev ($p_T \to 0$). Tej lastnosti pravimo *infrardeča varnost*. Na sliki 4 so prikazani rezultati testiranja infrardeče varnosti za algoritem $k$-voditeljev, ki je bilo izvedeno tako, da smo iz podatkov postopoma odstranjevali delce z najmanjšim $p_T$. Za vsak odstranjen delec smo nato pognali algoritem $k$-voditeljev s $k = 10$ (iz knjižnice `scikit-learn`), in



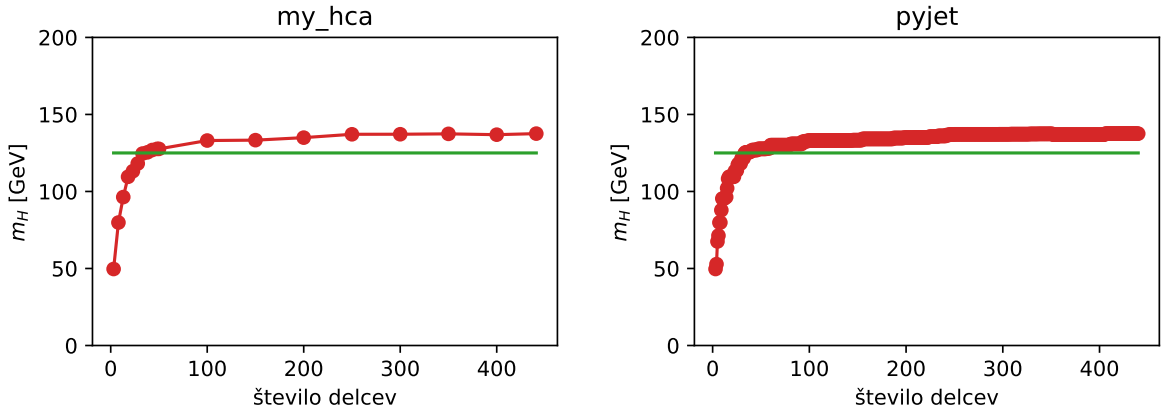

Slika 3: Rezultati gručenja delcev iz prvega simuliranega dogodka z metodo $k$-voditeljev v 2 (zgoraj) in 10 curkov (sredina, spodaj). Na levih slikah so prikazani delci v ravnini $\eta - \phi$, velikost delca pa je sorazmerna z njihovim $p_T$. Desne slike prikazujejo združene končne curke. Iz curkov z največjima $p_T$ je izračunana masa Higgsovega bozona, $m_H$.

sicer 20-krat. Opazimo, da so izračunane mase v splošnem zelo raztresene; zanimivo so združene v dve različni veji.

Na sliki 5 je prikazana infrardeča varnost algoritmov hierarhičnega gručenja (lastna implementacija in `pyjet`). Opazimo, da se rezultata ujemata, s čimer je potrjena tudi pravilnost implementacije algoritma. Pri celotnem naboru delcev je masa Higgsovega bozona nekoliko precenjena, z odstranjevanjem mehkih delcev pa se približuje pravi vrednosti. Ko odstranimo preveč delcev, rekonstruirana masa znatno pade, saj smo iz meritev odstranili tudi delce, ki so bili del hadronizacije $b$-kvarkov.




Slika 4: Infrardeča varnost algoritma $k$-voditeljev za $k = 10$. Rdeča črta prikazuje povprečje modrih točk, ki so bile izbrane po občutku.



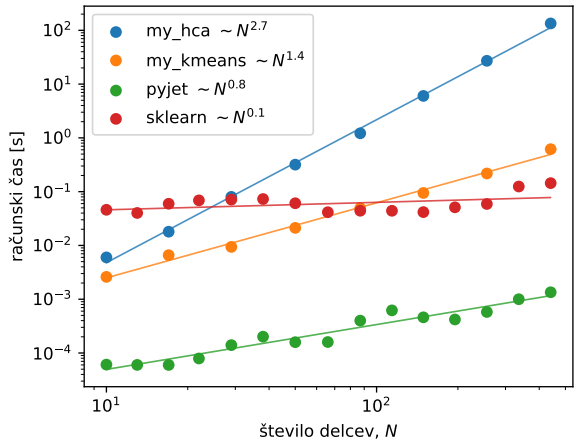

Slika 5: Infrardeča varnost algoritmov hierarhičnega gručenja. Lastna implementacija hierarhičnega gručenja (levo) in algoritem iz knjižnice `pyjet` (desno), oba pri $p = 1$ in $R = 0{,}4$.

## 5.3 Časovna zahtevnost

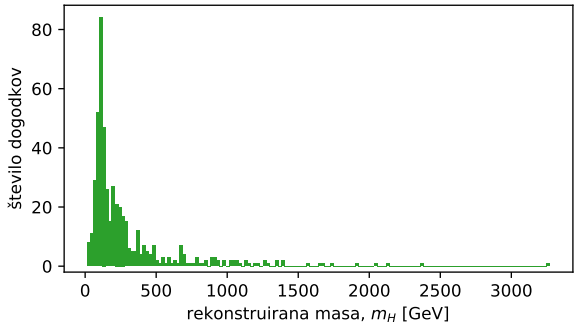
Oglejmo si še časovno računsko zahtevnost iplementiranih algoritmov, ki je prikazana na sliki 6. Algoritem $k$-voditeljev v implementaciji *sklearn* ima v primerjavi z lastno imlpementacijo boljšo časovno odvisnost, saj uporablja izboljšane načine računanja (npr. Lloydov ali Elkanov algoritem, pri katerem upoštevamo trikotniško neenakost). Kot je bilo omenjeno, opazimo, da ima lastna implementacija hierarhičnega gručenja približno kubično časovno odvisnost, medtem ko je implementacija v `pyjet` več redov velikosti hitrejša in ima tudi boljšo časovno odvisnost.

## 5.4 Vsi podatki

V prejšnjih rezultatih smo uporabljali zgolj podatke iz enega dogodka. Z uporabo hitre knjižnice `pyjet` pa si lahko privoščimo gručenje vseh 500 dogodkov. Rezultati gručenja so prikazani na sliki 7 pri parametrih $p = 1$, $R = 0{,}4$ in rekombinacijski shemi $E$. Opazimo visok vrh v okolici $m_H \approx 130\,\mathrm{GeV}$, a tudi zelo dolg rep, torej dogodke, pri katerih je rekonstruirana masa presegala $3000\,\mathrm{GeV}$.




Slika 6: Primerjava časovne zahtevnosti algoritmov v odvisnosti od števila delcev v dogodku.




Slika 7: Histogram rekonstruirane mase pri parametrih $p = 1$ (algoritem $k_t$) in $R = 0{,}4$ ter rekombinacijski shemi $E$.

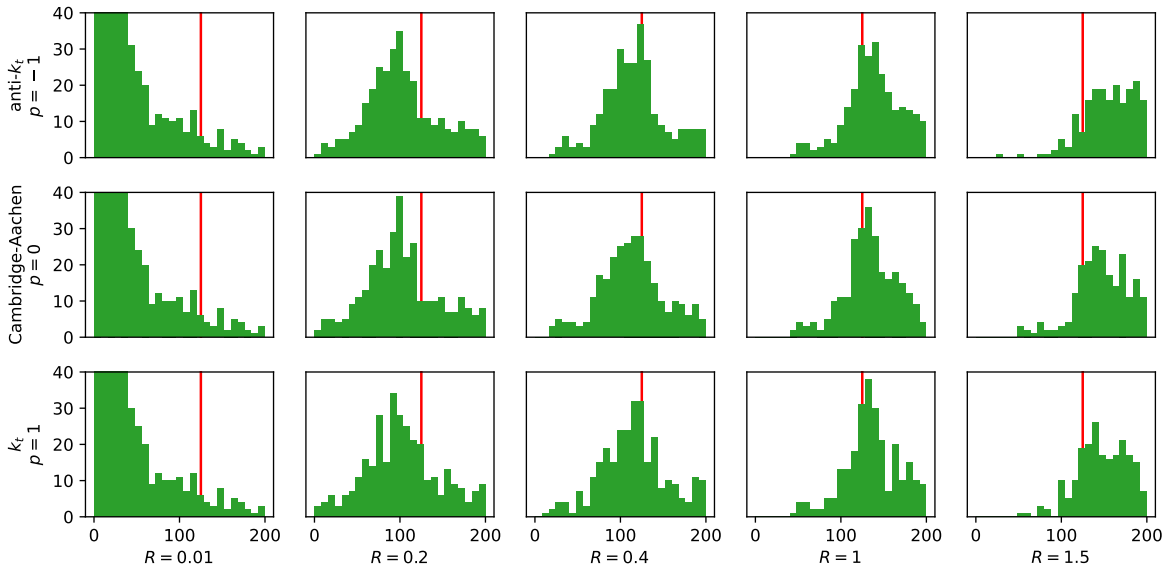
Podrobneje so prikazani histogrami na sliki 8, in sicer na intervalu od 0 do 200 GeV za vse tri različne algoritme in 5 različnih izbranih parametrov $R$. Priročnik za `fastjet` naaja, da je optimalna izbira parametra $R$, s katerim reguliramo delež ozadja v curkih, med 0,4 in 0,6, kar se potrjuje tudi na naših histogramih, saj prav pri tej izbiri $R$ maksimum histograma sovpada s točno vrednostjo. Pri manjšem $R$ dobivamo manjše mase, saj je efektivni premer curkov manjši, s tem pa v curek ne zajamemo vseh produktov hadronizacije. Obratno pri večjih $R$ dobivamo večjo maso Higgsovega bozona, saj so curki širši in posledično zajemajo preveč delcev, tudi tiste, ki so nastali v procesih ozadja.

Rezultati za algoritme anti-$k_t$, Cambridge-Aachen in $k_t$ so v splošnem primerljivi in podobni brez večjih opaznih razlik.

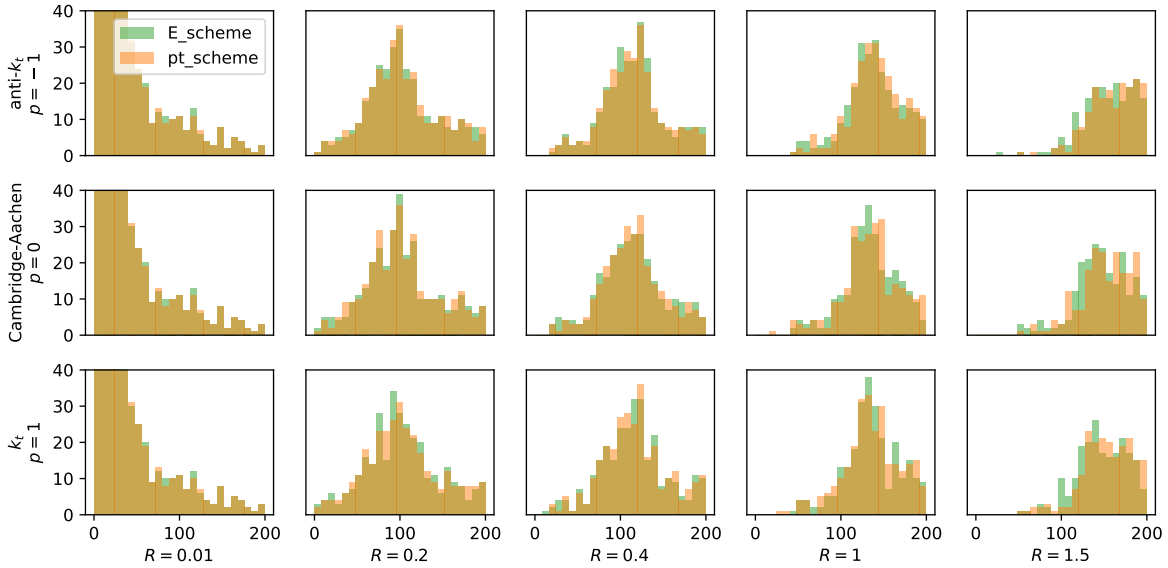
Ker sem se veliko ukvarjal tudi z rekombinacijsko shemo, me je zanimalo, ali njena izbira vpliva na rezultate. Slika 9 prikazuje histograme za shemo $E$ in shemo $p_t$. Opazimo, da shemi ne vrneta povsem identičnih rezultatov, a je groba oblika histogramov primerljiva.

## 5.5 Vizualizacija curkov

Za zaključek si lahko ogledamo še vizualizacijo curkov, rekonstruiranih z algoritmom `fastjet`. Vizualizacija poteka tako, da k prvotnim delcem dodamo veliko izjemno meh-




Slika 8: Histogrami izračunanih mas Higgsovega bozona pri različnih $p$ in $R$.

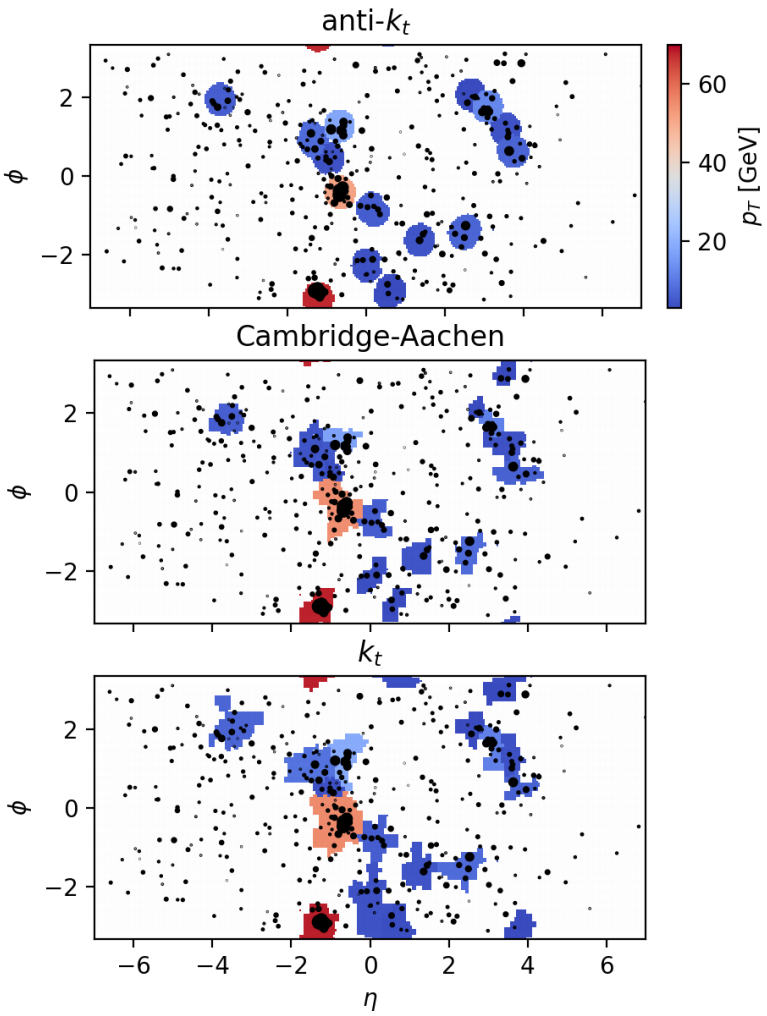



Slika 9: Primerjava rekombinacijskih shem $E$ in $p_T$.

kih delcev (*ghost particles*, $p_T \ll 1$), ki jih enakomerno razporedimo po mreži v ravnini $\eta - \phi$. Po združevanju lahko nato obarvamo območje (torej množico pravih delcev in delcev duhov), ki pripada vsakemu curku. Rezultati so prikazani na sliki 10 za tri različne izbire $p$. Opazimo, da ima večino curkov razmeroma majhno gibalno količino, medtem ko izstopata dva curka, ki predstavljata hadronizacijo $b$-kvarkov. Slike si moramo predstavljati kot valje, torje z zlepljenima zgornjo in spodnjo stranico. Prav tukaj vidimo, kako pomembna je periodičnost v kotu $\phi$, ki je nismo upoštevali pri naivnih združevanja s $k$-voditelji, zaradi česar bi lahko zgrešili dobro definiran curek, ki bi bil razdeljen na robovih $\phi = \pm\pi$.

Na vizualiciji curkov tudi vidimo lastnost algoritma anti-$k_t$, da deluje kot stožčasti algoritem – curki imajo namreč pravilno elipsasto/krožno obliko z enakimi površinami,

medtem ko so curki pri ostalih metrikah nepravilnih oblik in različnih velikosti.




Slika 10: Vizualizacija curkov po združevanju z algoritmom `fastjet`. Pred vsakim združevanjem je bilo dodanih 37 240 delcev duhov s $p_T = 10^{-10}\,\mathrm{GeV}$. Prikazani so zgolj curki s $p_T \geq 3\,\mathrm{GeV}$.

# 6 Zaključek

V nalogi smo se ukvarjali z analizo trkov visokoenergijskih curkov protonov, pri kateri smo želeli z ustreznim gručenjem zaznanih delcev v curke določiti maso Higgsovega bozona, ki je razpadel v dva kvarka $b$. Ogledali smo si enostavnejši algoritem združevanja s $k$-voditelji in ga tudi sami implementirali, nato pa se posvetili hierarhičnemu gručenju, ki je dalo veliko točnejše rezultate. Pozornost smo posvetili tudi razlagi različnih rekombinacijskih shem in v knjižnico `pyjet` dodali možnost izbire sheme. Za konec smo si ogledali še vizualizacijo curkov in pokazali stožčastost algoritma anti-$k_t$.