Ruteplanlegging i skjærgård — problemnotat fra praksis

> **Til:** forsker innen ruteplanlegging / beregningsgeometri
> **Fra:** BoatNav (Fellag AS) — maritim navigasjons-app
> **Dato:** 2026-07-26
> **Formål:** beskrive de problemene som faktisk oppsto ved implementasjon, med målte tall, slik at de kan vurderes mot literaturen. Dette er ikke en løsningsskisse.

---

1. Systemet og hva det skal gjøre

En fritidsbåt-app som skal foreslå en rute fra der båten ligger til et sted brukeren har søkt opp, i norsk skjærgård. Ruta skal ikke gå gjennom land, tørrfall eller områder grunnere enn brukerens oppgitte sikkerhetsdybde.

Tre trekk skiller dette fra klassisk robot-/spillnavigasjon og former alle valgene:

Forslaget er rådgivende, ikke autonomt. Et menneske skal lese ruta, forstå hvorfor hvert knekkpunkt ligger der det ligger, justere den for hånd på land før avreise, og deretter styre etter den. Det gir to harde krav som sjelden stilles i literaturen:

- *Få knekkpunkter.* En optimal sti med 62 punkter langs cellevegger er ubrukelig å redigere. Vi trenger en sti som er kort **og** har få vertekser — nærmere min-link-path enn shortest-path.
- *Hvert legg må kunne begrunnes.* Systemet sier aldri «trygt». Det sier «ingen funn innenfor 50 m i dataene», og lister hva som ble funnet. En algoritme hvis svar ikke kan brytes ned i pr.-legg-begrunnelser er vanskelig å bruke her, uansett hvor optimal den er.

Feilene er asymmetriske. En falsk «klar» kan sette en båt på et skjær. En falsk «sperret» er bare irriterende. All diskretisering må derfor være konservativ — men konservatisme har en pris som viste seg å være mye høyere enn ventet (§4.1).

Kjøremiljøet er en telefon. JavaScript (Hermes), én tråd, ingen native kode, ingen forhåndsberegning i sky. Interaksjonsbudsjettet er ~1 s med spinner, absolutt tak noen få sekunder. Kartdata hentes over mobilnett og skal helst caches.

---

2. Datagrunnlaget

Kartverkets sjøkartdata, forhåndsprosessert til vektorfliser (PMTiles, z14) og lest direkte på klienten. Per utsnitt får vi:

- **Flater:** `land`, `torrfall`, `grunt-omraade` (sistnevnte med dybdeattributt). Rene ringer — ingen topologi, ingen naboskap, ingen navigasjonsnett.
- **Punkter:** `skjaer`, `grunne`, `dybdepunkt`, dybde ofte, men ikke alltid, satt.

Målt i et havneområde (Langesund, boks ±2,5 km): **2 804 punkter og 3 352 flater fra 49 fliser.** Ett enkelt landpolygon hadde **841 noder**. Tettheten er svært ujevn: åpen fjord er tom, skjærgård er tett, og et fastlandspolygon strekker seg over hele utsnittet.

Viktige egenskaper ved dataene:

- Ringer er **klippet per flis**. Samme øy blir flere polygoner. Vi mistet aldri hull/øy-semantikk i praksis (verifisert), men topologi må antas fraværende.
- **Manglende dybde er ikke lik trygg dybde.** Et dybdepunkt uten verdi må telles som farlig; ellers blir «ingen data» til «ingen fare», som er den farligste feilslutningen i domenet.
- **Utstrekningen av hentede data må inn i planleggeren.** En celle uten kjente farer ser ut som åpent vann. Uten en eksplisitt datagrense planla søket gjennom kartløst område og «fant en klar vei» rundt nordsiden av en øy vi ikke hadde lest.

---

3. Nåværende metode (for kontekst, ikke som forslag)

1. Romlig indeks over utsnittet (uniformt 500 m rutenett, hash-basert).
2. Uniformt boolsk rutenett over søkeboksen. Celle sperret hvis senter eller noe av de fire hjørnene ligger i sperret flate, eller en punktfare ligger nærmere enn halve celldiagonalen.
3. A*, 8-naboer, diagonal koster √2 og krever at begge ortogonale naboer er frie. Euklidsk heuristikk i celleenheter. Åpen liste med lineært minimumsuttak.
4. **Etterforenkling:** grådig «hopp så langt fram som mulig», der hvert kandidathopp måles med eksakt segment-mot-polygon-sjekk mot de faktiske dataene.
5. Endelig verifisering av hvert gjenværende legg, med per-legg-begrunnelse.

Arbeidsdelingen er bevisst: **rutenettet avgjør OM det finnes vei, den eksakte linjesjekken avgjør hvordan ruta ser ut.** Det gir korte ruter med få punkter uten at nettets diskretiseringsfeil arves inn i sluttsvaret.

En tidligere variant — rekursiv halvering med informerte sidesprang rundt hindringen — ble forkastet. Den løser én hindring elegant og konvergerer aldri på en rekke av dem: tvers over en øygruppe fant den ingen vei i det hele tatt. Skjærgård er nettopp en rekke hindringer.

---

4. De praktiske problemene, med målinger

4.1 Konservativ diskretisering forsegler smale løp

Hjørneregelen (celle sperret hvis noe hjørne er i land) er nødvendig: med bare senter-testing smatt diagonalsteg mellom to «farbare» celler tvers gjennom et nes, og A* leverte stier som den etterfølgende eksakte sjekken avviste — altså forslag som ikke holdt sitt eget løfte.

Men samme regel sperrer et løp som er smalere enn omtrent to celler. Målt i Langesund havn, samme boks, samme data:

Cellestørrelse	Nett	Farbare celler	Resultat
120 m	46 × 42 = 1 932	1 019 (53 %)	**Ingen sti.** A* åpnet 4 celler — startcellen lå i en isolert lomme
40 m	138 × 125 = 17 250	10 794 (63 %)	Sti funnet: 62 rutenettpunkter, 326 celler åpnet

Renna er ~250 m bred og trafikkeres av hurtigbåt. Med 120 m celler eksisterte den ikke. Merk at nettet med 120 m celler brukte **1 932 av 60 000 tillatte celler** — det var ikke budsjettet som tvang grovheten, det var en dårlig valgt konstant.

Spørsmål: finnes det en diskretisering som er samtidig konservativ (aldri klassifiserer en sperret passasje som fri) og konnektivitetsbevarende (aldri lukker en passasje som faktisk er farbar for et fartøy med gitt bredde)? Vi mistenker at det riktige svaret er å forlate den boolske celle-abstraksjonen til fordel for en klaringsbasert representasjon (§4.5), men vil gjerne vite om problemet har et navn.

4.2 Ett uniformt nett kan ikke dekke begge skalaene

Dette er hovedproblemet, og det vi ikke har løst.

Cellestørrelsen bestemmes nå av søkeboksens areal og et celletak på 60 000:

Strekning	Boks	Cellestørrelse	Celler
0,3 nm (ut av havn)	±2,5 km	40 m	17 250
8,4 nm (kystoverfart)	±12,6 km	**223 m**	60 180 (90 % farbare)

På 223 m celler er problemet fra §4.1 tilbake i full styrke: sund smalere enn ~450 m forsvinner. Altså virker ruteplanleggeren i havneområdet man starter i, og degraderer til «åpen sjø»-oppløsning nettopp når ruta blir lang nok til å passere gjennom skjærgård underveis.

Å heve celletaket løser det ikke innenfor kjøretidsbudsjettet: klassifiseringskostnaden er lineær i celleantall, og en 40 km boks med 40 m celler er $\sim 10^6$ celler.

Spørsmål: hierarkisk/multiresolusjon-planlegging er det åpenbare svaret, men vi finner lite som håndterer *korrekthetsgarantien* vi trenger: et grovt nett som svarer «ingen vei» kan ta feil (§4.1), så en grov-til-fin-strategi kan ikke bruke det grove nettet til å beskjære søkerommet uten å risikere å miste den eneste farbare veien. Hvordan strukturerer man grov-til-fin når det grove nivået er *pessimistisk* framfor optimistisk? En optimistisk grov abstraksjon (celle fri hvis *noe* av den er fri) ville vært beskjærings-trygg, men gir korridorer som kanskje ikke er farbare.

4.3 Endepunkter inne i hindringer er normaltilfellet

En båt ved kai ligger nesten alltid innenfor havnas kystkontur i kartdataene. Verifisert: den faktiske GPS-posisjonen ved kai i Langesund er inne i et 841-noders landpolygon.

Første implementasjon returnerte «startpunktet ligger på land» og ga ingen rute. Det vil si at funksjonen feilet i 100 % av tilfellene der den er mest nyttig — når man planlegger turen fra plassen sin. Samme gjelder målet: mange havner ligger innenfor kystkonturen.

Nåværende håndtering: A* snapper begge ender til nærmeste farbare celle (ringsøk, 600 m rekkevidde), ruta legges derfra, og strekningen ut fra plassen merkes eksplisitt som **uverifisert** framfor å skjules. Ingen kartfil kan planlegge en avgang fra en båtplass.

Spørsmål: dette virker som et generelt, underbeskrevet problem i path-planning-literaturen — start/mål i det formelt ikke-farbare området, der snapping må være både geometrisk fornuftig og *semantisk ærlig* i det som rapporteres til brukeren. Finnes det formuleringer vi kan låne?

4.4 To orakler som er uenige

Systemet har to farevurderinger:

- **grov** (celleklassifisering i rutenettet), som avgjør om en vei finnes,
- **eksakt** (segment-mot-polygon med 50 m korridor), som avgjør om et legg godkjennes.

Vi oppdaget at den grove var **strengere** enn den eksakte: punktfarer sperret celler uansett, mens den eksakte sjekken med gjeldende policy godtar å passere 40 m fra en grunne (uten det ville forslaget blitt en slalåmløype mellom hver eneste grunne). Resultatet var at planleggeren svarte «ingen vei» på strekninger verifikatoren gladelig ville godkjent — en stille inkonsistens, ikke en synlig feil.

Spørsmål: dette er generelt for to-nivå-systemer med en billig feasibility-orakel og en dyr verifikator. Er det formulert noe sted som et designkrav (f.eks. «det grove oraklet må være en relaksasjon av det eksakte»), og hva er konsekvensene når kravet brytes i hver retning?

4.5 Binær farbarhet er feil modell

En båt vil ikke gå så nær et skjær som geometrien tillater; den vil ha klaring, og hvor mye klaring avhenger av sjøgang, mørke, fartøyets dypgang og skipperens nerver. Nåværende modell er boolsk, med én global 50 m korridor.

Dette gir to observerte utslag:

- Ruta legges gjerne rett langs kanten av det farbare, fordi kostnaden er ren geometrisk lengde.
- Terskelen for hva som sperrer en celle blir et hardt valg der virkeligheten er glidende.

Målt, og verre enn ventet. En bruker rapporterte at ruta la seg mellom skjær og grunner i stedet for i det dype vannet rett ved siden av. Sammenligning av mellomresultatet med sluttsvaret på den ruta (9 nm, Langesund–Fluer):

	Rå A*-sti	Etter etterforenkling
Punkter	206	7
Lengde	9,47 nm	8,98 nm
Minste klaring til noe grunnere enn 2 m	**41 m**	**1 m**
Farlige punkter nærmere enn 50 m	1	**21**

Rutenettet fant altså en vei med 41 m klaring, og etterforenklingen rettet den ut igjen forbi de samme grunnene — for å spare 0,49 nm, 5 %. Klaringen fantes i mellomresultatet og ble kastet, fordi forenklingens eneste akseptkriterium var «krysser linja noe», og en grunne 1 m til siden krysser ingenting.

Rettet med et eksplisitt klaringskrav på snarveier (50 m, klemt mot korridorbredden): minste klaring 1 m → 52 m, grunner innenfor 50 m 21 → 0, for prisen av tre knekkpunkter (7 → 10) og 0,05 nm. At det ble både tryggere *og* kortere enn 30 m-varianten viser hvor vilkårlig det gamle svaret var.

Generaliseringen er verdt å merke seg: en pipeline der et konservativt søk etterfølges av en aggressiv forenkling kan miste hele sikkerhetsmarginen i siste steg, uten at noe feiler. Forenklingen så ut som ren kosmetikk og var i praksis en ny — og mye dårligere — planlegger.

Spørsmål: en klaringsbasert kostnad (avstandstransform over det farbare området, kostnad som funksjon av klaring) er nærliggende, men gjør problemet flerkriterie: korteste vei og tryggeste vei er ikke samme rute, og brukeren må kunne forstå avveiningen. Hvordan presenteres et Pareto-sett for en skipper som skal ta ett valg på en telefonskjerm før avreise?

4.6 «Urimelig omvei» lar seg ikke uttrykke som forholdstall

Vi trenger å forkaste svar som er teknisk korrekte og åpenbart gale — målt på en kunstig vegg gikk et tidlig forslag 280 km rundt enden av den.

Et rent forholdstak (forkast hvis lengde > 2,5 × rett linje) straffer korte strekninger urimelig: veien ut av havna i Langesund er 1,35 nm mot 0,28 nm rett linje (+387 %) og er den helt normale veien rundt pirene. Nåværende hack: forkast bare hvis omveien er lang både i forhold (2,5 ×) **og** i meter (3 km).

Spørsmål: finnes det en prinsipiell formulering av «denne løsningen er optimal, men spørsmålet var galt stilt»? Det virker beslektet med å oppdage at start og mål ligger i ulike, nesten-adskilte komponenter av det farbare rommet, der forbindelsen går gjennom en omvei ingen ville valgt — altså en egenskap ved konnektiviteten, ikke ved stien.

4.7 Planlegging over delvis ukjent kart

Kartdata hentes flisvis over nett. Søkeboksen må klippes til det som faktisk er lest, ellers planlegges det gjennom kartløst område (§2). Men søkeområdet som trengs er ikke kjent før søket er gjort: en omvei kan kreve data langt utenfor den rette linja. På en 8,4 nm strekning slo flistaket inn og deler av strekningen ble **ikke lest**.

Dette produserte den alvorligste feilen vi har hatt. Grensen planleggeren fikk oppgitt var boksen vi *ba om*, ikke den vi *fikk*. Når flistaket kappet, ble store deler aldri lest mens grensen fortsatt påsto dekning. Observert på 8,2 nm fra Langesund: appen tegnet én rett strek tvers over Langesund og Stathelle og merket den **«ingen funn i dataene» — i grønt** — ved siden av sin egen oransje advarsel om at ruta ikke var vurdert. Fravær av data hadde blitt til fravær av fare, som er den ene feilen dette systemet ikke har lov til å gjøre.

Rettet slik:

- Dekning uttrykkes som **region, ikke boks** («er dette punktet i en flis vi faktisk leste?»). En boks kan ikke uttrykke det hentede uten enten å love for mye eller kaste bort lest farvann.
- Kjernen velger flisene, plattformen henter dem. Dekningen bygges av *nøyaktig* de samme flisene, så «hva ble lest» og «hvor har vi data» ikke kan komme i utakt.
- Ulest celle = **sperret** celle. Legger som ikke er heldekket får eget verdikt («utenfor data»), som rangeres over «ikke vurdert» fordi det ikke lar seg fikse ved å lese på nytt.

- Henting skjer i en **korridor** langs kurslinja, ikke i den omsluttende boksen. Boksen vokser kvadratisk: 8,2 nm ba om over tusen fliser og fikk 400 kolonnevis fra vest. Korridoren på ±3,5 km er 128 fliser og er sammenhengende langs ruta.

En ikke-åpenbar konsekvens er verdt å merke seg: siden rutenettet dimensjoneres etter **dekket** areal, gjør en bredere korridor cellene grovere. Målt på samme strekning fant søket *ingen* vei med ±7 km korridor, men fire punkter med ±3,5 km — på nøyaktig samme data og samme kode. **Mer kart gjorde svaret dårligere.** Vi har ikke sett den avveiningen beskrevet noe sted, og den er kontraintuitiv nok til at vi antar andre implementasjoner går i samme fella.

Spørsmål: dette er planlegging over et delvis kjent kart der «ukjent» må behandles som **ufarbart** (motsatt av det vanlige optimistiske frontier-baserte oppsettet), og der å utvide det kjente området koster både nettverk *og* oppløsning. Er det formulert som noe annet enn en anytime-/D*-variant? Vi er særlig interessert i strategier som lar søket etterspørre data ved fronten uten ubegrenset nedlasting — og i om det finnes en prinsipiell måte å velge datamengde når mer data forringer diskretiseringen.

4.8 Ytelse og hvor den ligger

Etter optimalisering: **~750 ms** for hele forslaget (nettbygging + A* + forenkling + verifisering) på havneområdet, målt i Node på en Mac. Ikke målt på fysisk telefon; Hermes forventes å være vesentlig tregere.

Kostnadsfordelingen er kanskje det mest interessante for en forsker: **nesten alt går med til å bygge nettet, ikke til å søke i det.** A* åpnet 326 av 17 250 celler. Klassifiseringen dominerer fordi hver celletest kjører punkt-i-polygon mot lokale flater, og et landpolygon med 841 noder gjør hver test dyr.

Én enkel gevinst er tatt: hjørnene i rutenettet deles mellom fire naboceller, så hjørnegitteret klassifiseres én gang framfor at hver celle tester fem punkter (1 162 ms → 749 ms).

Den åpenbare neste: **rasterisering framfor punkt-testing.** Å scanline-fylle hvert polygon inn i nettet er O(kanter + fylte celler) i stedet for O(celler × kanter). Vi har ikke gjort det ennå.

Merk at søkedelen er tilsvarende uoptimalisert — den åpne lista er et lineært minimumsuttak over et `Set`, ikke en binærheap — og at det **ikke** har vist seg å bety noe ved dagens cellemengder. Det peker på at klassisk profilering av A*-implementasjonen er feil sted å lete i dette domenet.

4.9 Rutas kvalitet som redigerbart objekt

Etterforenklingen gir få punkter, men den er grådig og *post hoc*: den kan bare kollapse den stien A *alt har valgt, og A* valgte den ut fra rutenettavstand, ikke sjøavstand. Målt omvei rundt en øy: **+140 %** i ett tilfelle, **+55 %** i et annet der ruta er geografisk riktig.

Any-angle-søk (Theta*/Lazy Theta*) er det nærliggende svaret fordi det påvirker *hvilke noder som utvides*, ikke bare hvordan resultatet ser ut etterpå.

Spørsmål: hvor stor er faktisk forskjellen mellom (a) A* på nett + eksakt etterforenkling og (b) any-angle-søk, når etterforenklingen får bruke eksakte geometriske sjekker mot originaldataene og ikke bare line-of-sight i nettet? Vi finner sammenligninger av any-angle mot naiv strengpulling, men ikke mot etterforenkling som har tilgang til den underliggende geometrien.

Og det underliggende: vi vil egentlig ikke ha korteste sti. Vi vil ha **korteste sti med få nok vertekser til at et menneske kan redigere den**, hvilket er en flerkriterie-avveining mot min-link-path.

4.10 Sikkerhetsmarginen forsvinner mellom stegene

Dette er den observasjonen vi tror har bredest gyldighet utenfor vårt eget domene, og den kom fra en bruker på vannet, ikke fra en test.

Pipelinen har fire steg som hver håndhever sin egen forestilling om «trygt nok»:

Steg	Hva det håndhever
1. Celleklassifisering	Sperr cellen hvis noe grunnere enn sikkerhetsdybden er innenfor halve diagonalen
2. A*	Bruk bare frie celler
3. Etterforenkling	Godta en snarvei hvis den ikke *krysser* noe
4. Endelig verifisering	Rapporter funn innenfor 50 m korridor

Steg 1–2 produserte en sti med 41 m klaring. Steg 3 leverte 1 m — og steg 4 kalte det klart, fordi «krysser» og «er nær» er ulike predikater. **Ingen av stegene feilet.** Hvert av dem gjorde nøyaktig det det var spesifisert til. Marginen forsvant i sammensetningen.

Vi har nå truffet den samme klassen feil tre ganger, i tre forkledninger:

- **§4.4** — det billige feasibility-oraklet ble *strengere* enn den dyre verifikatoren, så planleggeren avviste ruter verifikatoren ville godkjent.
- **§4.7** — dekningsgrensen søket fikk oppgitt var mer *optimistisk* enn dataene som faktisk var lest, så «ingen funn» ble sagt om områder uten data.
- **§4.10** (denne) — forenklingen var mer *aggressiv* enn søket, så klaringen søket fant ble kastet.

Alle tre er brudd på samme invariant: **hvert steg nedstrøms må være minst like konservativt som steget oppstrøms, målt på det samme predikatet.** Ingen av dem ga noe feilsignal — systemet svarte selvsikkert og galt hver gang, og alle tre ble funnet ved å måle mellomresultatet mot sluttresultatet på ekte data.

Spørsmål: finnes det en formulering av dette som designkrav — «monoton konservatisme gjennom en planleggings-pipeline» eller lignende — og finnes det teknikker for å *verifisere* egenskapen framfor å oppdage bruddene i felt? Vi ser for oss noe i retning av at hvert steg må kunne uttrykke sin sikkerhetsvurdering som samme skalar (klaring i meter), slik at komposisjon blir sjekkbar. I dag er de fire stegene uttrykt i fire ulike vokabularer — celler, naboskap, kryssing, korridorfunn — og det er nettopp derfor uoverensstemmelsene er usynlige.

---

5. Metoder vi har vurdert, og hvorfor de ikke er prøvd

Metode	Attraktivt fordi	Uavklart
Synlighetsgraf + Dijkstra	Eksakt korteste vei i polygonalt domene, ingen diskretiseringsfeil, får naturlig få vertekser	Tusenvis av polygonhjørner per utsnitt; $O(n^2 \log n)$ konstruksjon på telefon. Hvor mye hjelper tangent-beskjæring og konveks-innhylling-reduksjon i ekte sjøkartdata?
Constrained Delaunay + navmesh + funnel	Eksakt sti innen korridor, gjenbrukbart nett, håndterer trange sund presist	Byggekostnad på klienten; inkrementell/flisvis konstruksjon; hvordan representere punktfarer som klaringskrav framfor hindringer; og hvordan produsere per-legg-begrunnelser fra en korridor
Theta* / any-angle	Liten endring fra dagens A*, fikser sikksakk ved rota	Se §4.9
Konveks innhylling av øyer som forbehandling	Kutter hjørnetall dramatisk	Stenger bukter — feil når målet ligger i en bukt

---

6. Det vi helst vil ha svar på

1. **Konservativ og konnektivitetsbevarende diskretisering** (§4.1) — finnes det, eller er boolske celler feil abstraksjon for smale løp?
2. **Hierarkisk planlegging med pessimistisk grovnivå** (§4.2) — hvordan beskjære trygt når det grove nivået kan si «ingen vei» feilaktig?
3. **Planlegging der ukjent = ufarbart, og der mer data gir grovere celler** (§4.7) — hvordan velges datamengden når den er en avveining mot oppløsning?
4. **Korteste sti med få vertekser** som eksplisitt mål, ikke som etterbehandling (§4.9).
5. **Klaringsbasert flerkriteriekostnad presentert forståelig** for en beslutningstaker (§4.5).
6. **Monoton konservatisme gjennom en pipeline** (§4.10) — kan egenskapen formuleres og verifiseres, framfor å oppdages i felt? Dette er det vi tror er mest overførbart.
7. Om noen av disse er velkjente problemer under navn vi ikke har lett etter.

---

7. Presiseringer

«Lovlig farvann» dekker i dag kun land, tørrfall og dybde. Fartsgrenser, verneområder med ferdselsforbud, trafikkseparering og ledlinjer/farleder er egne datakilder som ikke er lastet. Når de kommer, blir begrensningene ikke-binære (en farled er *foretrukket*, ikke påkrevd), og problemet flytter seg fra hindringsunngåelse mot regelbasert kostnad.

Korteste vei er dessuten ikke nødvendigvis riktig vei i sjøgang eller mørke. Algoritmen kan gi korteste lovlige vei; den kan ikke gi *riktig* vei. Alt systemet sier til brukeren er formulert deretter — det finnes ingen «trygg»-verdi noe sted i koden.