Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2018-292-AC3, 2019

Geoscientific Model Development Discussions

Interactive comment on "VISIR-I.b: waves and ocean currents for energy efficient navigation" *by* Gianandrea Mannarini and Lorenzo Carelli

Gianandrea Mannarini and Lorenzo Carelli

gianandrea.mannarini@cmcc.it

Received and published: 20 May 2019

General assessment

The paper is a through one. There are very few papers on ship weather routing covering so many aspects of this optimization problem and doing it with so much detail. I particularly appreciate:

- *the time interpolation - I agree that it may bring significant benefits for drastically changes in the subsequent weather forecasts,*
- *using bathymetric database with detailed real data,*
- *detailed results and analysis of time savings attributed to exploitation of waves*

and currents.

My specific comments are few – I provide them below.

–AUTHORS' RESPONSE:
We thank the Referee for his/her time and comments on our manuscript: They definitively contributed to improve it. In this document, we report Referee's text in italics and our replies as a normal text, distinguishing wherever needed our response from the manuscripts parts involved by changes. All references to sections, equations, figures, and tables are relative to the submitted gmd-2018-292 manuscript.

Specific comments

1 -*'We use throughout this manuscript the words "track" or "trajectory" for indicating a set of waypoints joining two given endpoints or harbours, in relation to departure on a given date, and the words "route" or "crossing" when there is no reference to a specific departure date.' While 'track' is perfectly acceptable here, I suggest replacing 'trajectory' with some other word (e.g. path). The word 'trajectory' is usually used in control and robotics with a different meaning: it involves greater accuracy (manoeuvrability and actuation is- sues), especially for obstacle avoidance or collision avoidance purposes. A "trajectory" between two harbours does not make sense.*

–AUTHORS' RESPONSE:
Agreed: occurrences of "trajectory" will be replaced by "path".

–MANUSCRIPT PARTS INVOLVED:
Whole manuscript.

2 - *Regarding section 2.3: an alternative approach would be to use varying resolution of a graph – the nodes can be placed with larger resolution in coastal areas and with lower resolution at open waters. I suggest commenting on the those two possible approaches to this problem and explaining why you choose the one with additional intersection check.*

–AUTHORS' RESPONSE:
In fact, we took into consideration the fact that the VISIR graph grid may deserve a redesign, e.g. reducing the density of gridpoints in open seas through the use of a nonuniform mesh. An adaptive refinement mesh (Berger and Colella, 1989) or unstructured mesh limiting the minimum angle (Shewchuk, 2002) could be another option. Their advantage would be to reduce the number of open-ocean edges, reducing RAM allocation and speeding up the computation of the shortest path.

However, we point out that, for the safety of navigation, a check on intersection between graph arcs and shoreline is in any case needed, no matter the grid resolution or structure. In fact, even if the mesh is built via a tessellation, intersection with islands and boundary elements smaller than mesh elements should be checked (Legrand et al., 2000). For a graph of higher order of connectivity ($\nu \gg 1$, cf. manuscript's Sect.2.3) this is even more challenging. Such a check on shoreline intersection can easily represent a significant computational cost (De Berg et al., 1997). In order to perform it effectively, it is crucial to be able to find indexes of graph elements next the shoreline. On a regular grid this operation can be carried out in $\mathcal{O}(M)$ time (M is the number of shoreline elements), no matter the size of the maritime domain (and we exploited this in the *i)* step of the algorithm described in Sect.2.3). Instead, on a random or not regular

mesh, a $\mathcal{O}(M \cdot n)$ time would be required by a linear search (n is here either the number of nodes or arcs of the graph). To speed up the search on a not regular mesh, a preliminary node indexing can be computed. With a *k-d* tree, an additional $\mathcal{O}(n \log(n))$ time for tree construction and, on average, $\mathcal{O}(M \cdot \log(n))$ for querying would be needed (Bentley, 1975). This is in excess of the $\mathcal{O}(M)$ estimate for corresponding step (cf. *i)* in Sect.2.3) in present VISIR graph creation algorithm.

Thus, at this stage we still preferred keeping a regular grid which enabled a relatively quick and easy graph computation at the cost of a longer path computing time. This is not critical, given the not operational functioning of VISIR for the present exercise. In future model versions, also depending on coding options, domain, and type of application, we may reconsider this choice.

–MANUSCRIPT PARTS INVOLVED:
Sect.2.3 will be expanded using response above.

3 - *Regarding section 2.5.2: 'Edges which, for a given EOT, violate stability are pruned before the shortest path algorithm is run. This way, it is ensured that the optimal track preserves vessel intact stability.' Based on the above description, I am not sure if this approach is correct. In presence of coastline, shallows etc. the exact time at which an edge will be transited cannot be know exactly prior to running the algorithm. Even for open ocean, avoiding a cyclone may cause a delay resulting in reaching a certain graph node much later, thus making all prior assumptions inaccurate. Therefore, in my opinion the edges' weights should be verified dynamically during the algorithm run instead of pruning the edges before the run.*

–AUTHORS' RESPONSE:

In VISIR, there is no prior assumption about the vessel time of sailing at the various spatial positions of the domain.

Following (Mannarini et al., 2016, Sect.2.2.2 & pseudocode in App.A), all vessel speeds at any location and direction (i.e. on each of the A edges) and any time (N_t time steps) are computed ahead of path optimization. RAM space allocation for storage of this information is discussed in Sect.3.2, Fig.3 and in our answer to Referee's comment *4a)* below. Then, the time-dependent Dijkstra's algorithm (Mannarini et al., 2016) can manage all this spatially and temporally dependent information for computing the time-optimal paths. Its correctness is demonstrated by comparison with the path resulting from the benchmark solution in a dynamic flow field by Techy (2011) (Sect.3.1.2, Fig.2, Tab.2). Thus, we can say that if cyclone avoidance causes a delay in reaching a specific location, vessel speed at that actually delayed time is used by VISIR for evaluating if sailing through that specific location at that specific time will still be part of the time-dependent optimal path.

For pruning of edges leading to loss of vessel intact stability, the algorithmic machinery works pretty much the same, with specific edges being labeled as unsafe at specific time steps only, cf. (Mannarini et al., 2016, Sect.2.2.2). If a vessel sails at that edge and time, it would experience stability loss, no matter the previous and subsequent path. Thus, that edge is pruned for just that time step ahead of path optimization.

–MANUSCRIPT PARTS INVOLVED:
The description provided in Sect.2.5.2 will be expanded making use of the response above.

4a - *While I appreciate the computational complexity analysis based on RAM allocation data, I would also hope for assessing computational time and space based on the algorithm itself. I agree that it is a hard task for complex algorithms, but still some*

analysis could be made, at least for the worst case.

–AUTHORS' RESPONSE:
Some deepenings concerning computational (CPU) time and memory space (RAM) of VISIR shortest path algorithm are provided in the following:

- **CPU time**
  Fig.3a (red markers) shows that the worst-case estimate of present VISIR implementation of Dijkstra's time-dependent algorithm scales nearly linearly with the number of degrees of freedom (DOF) of the problem. DOF is proportional to the product of the number A of graph edges and the number N_t of time steps of the dynamic environmental fields. N_t is roughly constant for a given route, as in Fig.3. It can be shown that, upon generalizing the graph arc arrangement of (Mannarini et al., 2016, Fig.1) to any order of connectivity ν of the graph (cf. Sect.2.3), A is given by

  $$A = 4\nu(\nu + 1)N \tag{1}$$

  with the number N of graph grid nodes (Mannarini et al., 2018, in review). In any two-dimensional regular mesh, N scales quadratically with the inverse mesh resolution, $N \sim (1/\Delta_g)^2$. For the series of experiments in Fig.3, we varied ν as $1/\Delta_g$. When taken together, these two effects result into:

  $$\text{DOF} = A \cdot N_t \sim \nu^2 N \sim (1/\Delta_g)^4 = \mathcal{O}(N^2) \tag{2}$$

  Thus, the empirically retrieved linearity of CPU time with DOF corresponds to a quadratic dependence in N. This is in fact the expected worst-case performance of a Dijkstra's algorithm (Bertsekas, 1998). As we stated in Sect 2.4, in presence of binary heaps, such estimate can be reduced to $N \log N$. This will come up in future VISIR versions.

- **RAM allocation**
  In oder to further clarify the memory space requirements of VISIR, with a focus on its shortest path algorithm, we collected and analyzed additional datasets as described below. They consist of:

  d_1*)* time series of RAM allocation of the VISIR Matlab job[1]

  d_2*)* stopwatch timer readings at specific VISIR processing phases[2]

  The d_2*)* dataset is then temporally offset by matching the end of the d_1*)* dataset. Finally, resulting d_2*)* data are smoothed by thinning and this results in the plots displayed in Fig. .e-f below.

  For each graph angular resolution (indexed by ν parameter) the timeseries exhibit different relative importance (both in terms of duration and RAM allocation) of the various processing phases. However, the d_1*)* and d_2*)* datasets confirm that, for $6 \leq \nu \leq 9$, the peak RAM is allocated during the edge weight computation. Furthermore, the shortest path algorithm is run twice: in its static version (Dijkstra, 1959) for the computation of the geodetic track, in a time-dependent version for the optimal track (Mannarini et al., 2016). The latter requires in input the edge delays at N_t time steps, and this justifies the uphill RAM step between these two phases.

–MANUSCRIPT PARTS INVOLVED:
The information already provided in Sect.3.2 and will be integrated with material above. In particular:
- Fig.3.a-d and Tab.3 will be updated for using performance data from the latest code version and for accounting for smoothing of the RAM timeseries;

[1]Using the shell command: `top | grep MATLAB >> RAM-timeseries.txt`
[2]Using the Matlab commands: `tic, toc`

- two panels *e)* and *f)* will be added to Fig.3 of the manuscript with following caption:
"a) CPU time for the total VISIR job (blue markers) and for just the computation of the time-dependent shortest path (red markers). Only the *cw* case is shown. Dashed lines are fits of the model in Tab.3. b) Peak RAM allocation during the jobs of a) panel, with a reference line at the total installed RAM. c) Ratio of CPU times of the *cw* to the *w* case and (just for optimal path) for with to without time-interpolation. d) Ratio of peak RAM allocation of the *cw* to *w* type jobs. For panels a,b,d) both cases with (filled) and without (empty markers) time-interpolation. The DOF (Sect.3.2) of the time-dependent shortest path problems is displayed on the horizontal axis. e,f) Time series of RAM memory allocation during VISIR execution for *w* and *cw* type jobs, respectively. Black circles (blue lines) refer to runs without (with) time-interpolation of edge weights. Vertical dashed lines separate the main phases of the processing. Both panels refer to the $\nu = 8$ case of a)-d). The processing phase labels are: *ew* (computation of edge-averaged fields); *ed* (edge delays); *gdt* (geodetic track); *opt* (optimal track)."

4b - *It would also be interesting to compare the computational time with that of a non-deterministic approach (there are multiple meta-heuristics available, including Evolutionary Multi-objective Optimization, Ant Colony Optimization etc.).*

–AUTHORS' RESPONSE:
We would like to note first that, being based on Dijkstra's algorithm, VISIR solution is not just guaranteed to be exact, but also its performance (for a given route and vessel departure date) is stable over different runs. This is a difference with evolutionary (EA) and, generally speaking, with heuristics-based algorithms. For that class of algorithms, both the quality and the computational cost of the solution may vary over subsequent runs, as they are driven by random effects. The issue of randomness can be mitigated by statistical averaging over many simulations. However, a more fundamental issue is

that, as clearly stated in Eiben et al. (2003), performance of an EA should be assessed in terms of both efficiency (CPU time) and effectiveness (quality of the solution). Furthermore, even for a specific EA and EA implementation, performance may vary with tuning. Tuning refers to specifying values for the algorithm parameters, such as the "mutation rate". Tuning may affect both EA performance and robustness (Eiben et al., 2003).

Apart from the EA peculiarities, performance comparison of VISIR with other ship routing systems is also hampered by the fact that:

i) there is usually little or no evidence that those models were preliminarily validated versus exact solutions;

ii) the input environmental fields are not always available for other published results;

iii) access to the source code for running on identical conditions would be necessary;

iv) the computational platforms employed are either different or not documented;

In a dedicated collaborative effort for evaluation of VISIR vs. a deterministic path planning model which was previously tested against an analytical benchmark, we were able to overcome most of these difficulties (Mannarini et al., 2018, in review). We are open to reply that approach for EA-based ship routing models, e.g., the multi-objective EA reported in (Szlapczynska, 2015) or the ant-colony algorithm described in Tsou and Cheng (2013).

–MANUSCRIPT PARTS INVOLVED:
The information already provided in Sect.3.2 and will be integrated with the discussion above.

5 - *I agree with the authors that the paper would further benefit from a more realistic modeling of speed loss in waves and wind. I encourage them to include such modelling in their research.*

–AUTHORS' RESPONSE:
Thanks for the comment. In fact such a more realistic modeling of speed loss in waves and wind is planned, at least for Ro-Pax vessels, in the frame of the newly started GUTTA project[3].

–MANUSCRIPT PARTS INVOLVED:
Reference to GUTTA project will be added to the Conclusions.

[3]http://bit.ly/guttaproject

Table 3. Fit parameters for the data displayed in Fig.3a. The fit model is $a \cdot x^b + c$. For the optimal path data, c parameter is not fitted.

		no T-interp		with T-interp	
	units	optimal path	total job	optimal path	total job
a	s	$9.9 \cdot 10^{-8}$	$4.7 \cdot 10^{-10}$	$2.6 \cdot 10^{-6}$	$1.2 \cdot 10^{-7}$
b	–	1.07	1.42	1.01	1.18
c	s	-	52	-	60
rmse	s	3.9	15.6	3.3	24.8

References

Bentley, J. L.: Multidimensional Binary Search Trees Used for Associative Searching, Commun. ACM, 18, 509–517, https://doi.org/10.1145/361002.361007, http://doi.acm.org/10.1145/361002.361007, 1975.

Berger, M. J. and Colella, P.: Local adaptive mesh refinement for shock hydrodynamics, Journal of computational Physics, 82, 64–84, 1989.

Bertsekas, D.: Network Optimization: Continuous and Discrete Models, Athena Scientific, Belmont, Mass. 02178-9998, USA, 1998.

De Berg, M., Van Kreveld, M., Overmars, M., and Schwarzkopf, O.: Computational geometry, in: Computational geometry, pp. 1–17, Springer, 1997.

Dijkstra, E. W.: A note on two problems in connexion with graphs, Numerische mathematik, 1.1, 269–271, 1959.

Eiben, A. E., Smith, J. E., et al.: Introduction to evolutionary computing, vol. 53, Springer, 2003.

Legrand, S., Legat, V., and Deleersnijder, E.: Delaunay mesh generation for an unstructured-grid ocean general circulation model, Ocean Modelling, 2, 17–28, 2000.

Mannarini, G., Pinardi, N., Coppini, G., Oddo, P., and Iafrati, A.: VISIR-I: small vessels – least-time nautical routes using wave forecasts, Geoscientific Model Development,

9, 1597–1625, https://doi.org/10.5194/gmd-9-1597-2016, http://www.geosci-model-dev.net/9/1597/2016/, 2016.

Mannarini, G., Subramani, D., Lermusiaux, P., and Pinardi, N.: Graph-Search and Differential Equations for Time-Optimal Vessel Route Planning in Dynamic Ocean Waves, IEEE Transactions on Intelligent Transportation Systems, 2018, in review.

Shewchuk, J. R.: Delaunay refinement algorithms for triangular mesh generation, Computational geometry, 22, 21–74, 2002.

Szlapczynska, J.: Multi-objective weather routing with customised criteria and constraints, The Journal of Navigation, 68, 338–354, 2015.

Techy, L.: Optimal navigation in planar time-varying flow: Zermelo's problem revisited, Intelligent Service Robotics, 4, 271–283, 2011.

Tsou, M.-C. and Cheng, H.-C.: An Ant Colony Algorithm for efficient ship routing, Polish Maritime Research, 20, 28–38, 2013.

Interactive comment on Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-292, 2019.

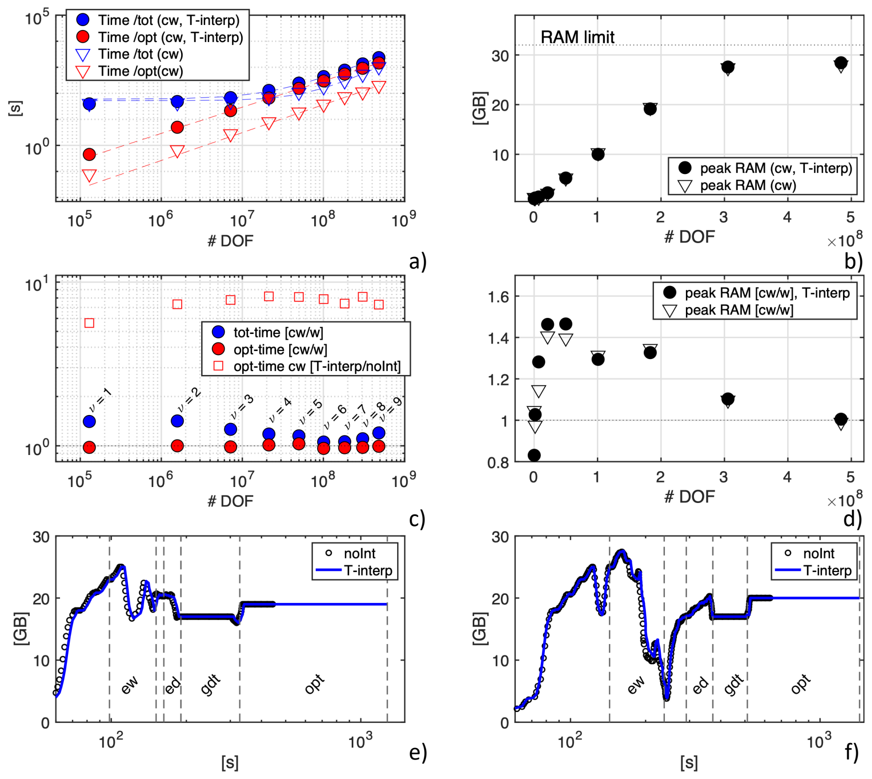

Fig. 3. e,f) Time series of RAM memory allocation during VISIR execution for w and cw type jobs, respectively. Black circles (blue lines) refer to runs without (with) time-interpolation of edge weights.