Pruned simulation-based optimal sailboat path search using micro HPC systems

Roman Dębski[0000−0003−3283−6032] and Bartlomiej Sniezynski[0000−0002−4206−9052]

Institute of Computer Science
AGH University of Science and Technology
Al. Mickiewicza 30, 30-059 Kraków, Poland
{rdebski,sniezyn}@agh.edu.pl

Abstract. Simulation-based optimal path search algorithms are often solved using dynamic programming, which is typically computationally expensive. This can be an issue in a number of cases including near-real-time autonomous robot or sailboat path planners. We show the solution to this problem which is both effective and (energy) efficient. Its three key elements – an accurate and efficient estimator of the performance measure, two-level pruning (which augments the estimator-based search space reduction with smart simulation and estimation techniques), and an OpenCL-based SPMD-parallelisation of the algorithm – are presented in detail. The included numerical results show the high accuracy of the estimator (the medians of relative estimation errors smaller than 0.003), the high efficacy of the two-level pruning (search space and computing time reduction from seventeen to twenty times), and the high parallel speedup (its maximum observed value was almost 40). Combining these effects gives (up to) 782 times faster execution. The proposed approach can be applied to various domains. It can be considered as an optimal path planing framework parametrised by a problem specific performance measure heuristic/estimator.

Keywords: heterogeneous computing · SPMD-parallel processing · trajectory optimisation · dynamic programming.

1 Introduction

Optimal path search – an important issue in robotics, aerospace engineering, and optimal control – in a number of cases has to be simulation based since the corresponding mathematical model is too complex for analytical methods. In such situations algorithms based on dynamic programming are often used. This approach usually leads to accurate results but is computationally expensive, because of the search space size and cost of simulation [8].

High computational costs can be an issue in a number of cases including (hard) real-time embedded systems but also near-real-time autonomous robot or sailboat path planners. Some of the time constraints can be met (at least

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_13

partially) by parallelising the algorithms and adapting them to contemporary CPU-GPU heterogeneous mobile/onboard computers, which are massively parallel micro HPC platforms. Although effective, this approach is often inefficient – the computation is accelerated but with no reduction of the algorithm computational cost. This can be an issue if energy consumption is of primary importance, which is the case in onboard/mobile computers (especially when at sea).

The aim of this paper, which is a significant extension of [8], is to present a simulation-based optimal sailboat path planning algorithm which is both effective and (energy) efficient. Its main contributions are:

- an estimator of the performance measure (i.e., sailing duration) that reflects its variational character and is accurate without being computationally complex (section 4.2),
- the concept of two-level pruning, which augments the estimator-based search space reduction with smart simulation and estimation techniques (paragraph *External and internal pruning* in section 4.3),
- the SPMD-parallel[1] algorithm for simulation-based optimal sailboat path search, based on the above two elements and adapted to on-board heterogeneous micro HPC systems (section 4.3),
- numerical results which demonstrate three important aspects of the algorithm: the accuracy of the performance measure estimator, the efficacy of the two-level pruning, and the SPMD-parallelisation capabilities (section 5).

The remainder of this paper is organised as follows. The next section presents related research. Following that, the search problem under consideration is defined and proposed algorithm is described. After that, experimental results are presented and discussed. The last section contains the conclusion of the study.

2 Related research

The first scientific formulation of the problem of trajectory optimization[2] was proposed by Johan Bernoulli in 1696 as the brachistochrone problem (see [26] for discussion). For over two hundred years, the main approach for trajectory optimization was the calculus of variations (see, for instance [24]) based on Johan's brother Jakob's solution for the brachistochrone problem. This situation was changed when dynamic programming was introduced [1] after the development of the digital computer in the 1950's. Effective shortest path algorithms [2, 9], the Pontryagin Maximum Principle [19] and non-linear programming (NLP) are foundations for many trajectory optimization algorithms. They are classified either as *direct*, to construct the best path step by step like our algorithm, or *indirect* in which the best path is a solution of some set of equations [25, 15]. A subgroup of direct methods is a set of algorithms based on the shortest path algorithm [4, 21].

[1] SPMD - Single Program Multiple Data

[2] In this paper, the notions of *trajectory optimization* and *optimal path search* are used interchangeably.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_13

An important part of optimal search problems is those having *black-box represented* performance measures. In such cases the performance measure values are usually computed using computer simulation, and therefore classical optimization methods cannot be used directly. Algorithms for such problems are often based on heuristics, *soft-computing* and *AI* methods [28, 20, 27].

Another important direction of research is related to the parallelisation of both trajectory optimization and graph algorithms [6, 13] which then can be executed using GPU[3] acceleration [12, 22, 17, 10]. In our research we also utilized GPU. Although initially we applied machine learning (ML) algorithms with some success, even better results were obtained using a specially designed performance measure estimator allowing us to significantly limit the number of performance measure evaluations.

A different approach to speed up computations is pruning the search graph. In [11] uniform-cost grid environments are considered. They are simple but commonly used in robotics and video games. The authors propose an algorithm finding optimal paths by expansion of selected nodes only. Pruning rules are defined to decide if a node should be skipped or expanded. In [29] an algorithm for path planning of a differential drive mobile robot is proposed. It is an extension of a Bi-directional Rapidly-exploring Random Tree (RRT) method [14]. It improves the performance of path planning by incorporating kinematic constraints and efficient branch pruning. In [30] another extension of RTT is proposed. An initial tree covering the whole map is processed using branch pruning, reconnection, and regrowth operations. It allows for planning in a complex, dynamic environments in which obstacles and the destination are moving.

Trajectory optimization is often used in robotics, as mentioned above, and also in other various domains, for example aerospace engineering [5]. However, the sailboat domain is also common [23, 18, 7, 31].

3 Problem formulation

Consider a sailboat going from point $A(q_A, y_A)$ to $B(q_B, y_B)$, where (q_i, y_i) are the coordinates of the corresponding point in either the Cartesian or polar system. We assume that the true wind can be expressed by the following vector field (see Fig. 1)

$$\boldsymbol{v}_t(q, y, t) = M(q, y, t)\ \hat{\boldsymbol{q}} + N(q, y, t)\ \hat{\boldsymbol{y}}, \tag{1}$$

where: $M(q, y, t)$, $N(q, y, t)$ are scalar functions, and $\hat{\boldsymbol{q}}$, $\hat{\boldsymbol{y}}$ are the unit vectors representing the axes of the corresponding coordinate system.

The set of admissible $\widetilde{AB}$ paths (i.e. the problem domain) consists of C^1-continuous curves which cover the given sailing area S_A (see Fig. 1) and do not violate the constraints embedded in a sailboat model (these constraints can be related to the state and/or control variables). This model is used to evaluate each path ($y^{(i)}$) through simulation, therefore

$$J[y^{(i)}] = \text{PerformSimulation}[y^{(i)}, \text{cfg}\,(\boldsymbol{v}_t, \ldots)], \tag{2}$$

[3] Graphics Processing Unit

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_13

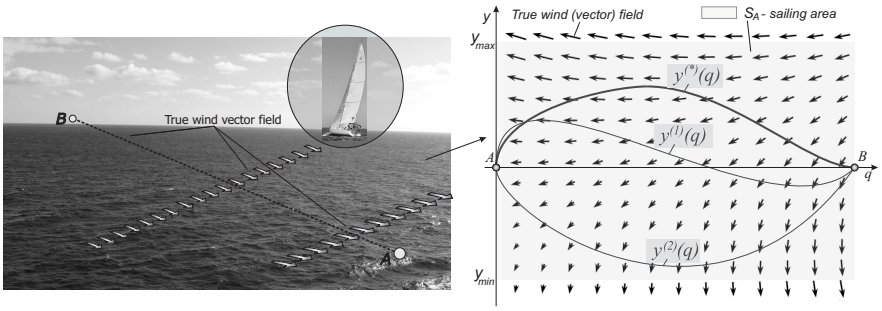

Fig. 1. Optimal sailboat path search problem: example admissible paths connecting points A and B, with $y^{(*)}(q)$ representing the optimal path.

where: J represents the given performance measure and $\mathrm{cfg}\,(\boldsymbol{v}_t, \ldots)$ – the simulator configuration.

Problem statement. The optimal sailboat path search problem under consideration can be defined as follows:

- find, among all admissible paths, the one with the best value of performance measure J;
- the values of J can be found only through simulation;
- only on-board, off-line computers can be used.

Remark 1. In the special case, when $J[y^{(i)}] = \Delta t[y^{(i)}]$, with Δt being the sailing duration, we get the *minimum-time problem*.

4 Solution

The approach we propose in this paper is a "pruning augmented" extension of the one introduced in [8]. It is based on the following two main steps:

1. transformation of the continuous optimisation problem into a (discrete) search problem over a specially constructed finite graph (*multi-spline*);
2. application of pruned dynamic programming to find the approximation of the optimal path represented as a C^1-continuous cubic Hermite spline.

These two steps repeated several times form an adaptive version of the algorithm in which subsequent grids are generated through mesh refinement making use of the best trajectory found so far. Key elements of the proposed algorithm are:

- multi-spline based solution space and the SPMD-parallel computational topology it generates;
- effective estimate of the performance measure (sail duration);
- two-level smart pruning that significantly reduces the time complexity of the reference algorithm.

They are discussed in the following sub-sections.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_13

4.1 Multi-spline as the solution space representation

A discretisation of the original continuous problem leads to a grid, G, which structure can be fitted into the problem domain. A simple example of such a grid is shown on the left of Fig. 2. The grid is based on equidistant nodes grouped in rows and columns: four regular rows plus two special ones – containing the start (A) and the end (B) points – and four columns. The number of nodes in such a grid is equal to

$$|G| = n_c\,(n_r - 2) + 2 \tag{3}$$

where n_c and n_r are the numbers of columns and rows (including the two special ones), respectively.

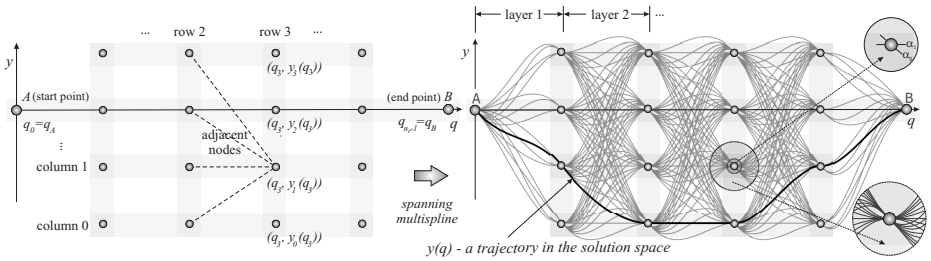

Fig. 2. Solution space representation: multi-spline built (spanned) on regular grid G_{ex}

Having assigned n_{ts} additional values (a vector of n_{ts} tangent slopes) to every node of grid G, we obtain a new structure, G_{ex}, that can store not only the coordinates of each node but also the n_{ts} slopes (angles) of path segments which start/end in that particular node (see the right part of Fig. 2).

Joining the nodes from subsequent rows of G_{ex} by using cubic Hermite spline segments, we get a *multi-spline* [8] which forms a discrete space of C^1-continuous functions (see Fig. 2). The properties of a multi-spline that are important from the point view of this paper are as follows[8]:

- when seen as a graph (knots are vertices, spline segments are edges) – it is directed (from A to B or vice-versa), acyclic and topologically sorted (i.e. the edges in layer l are followed by those from layer $l+1$ and the vertices in row r are followed by those from row $r+1$, see Fig. 2);
- it is built from $n_c n_{ts}^2\,[(n_r - 3)\,n_c + 2]$ different spline segments;
- the discrete search space it spans represents $n_{ts}^{n_r}\,n_c^{n_r-2}$ different trajectories connecting points A and B; this value corresponds to the "inter-row complete" graph (i.e. the one in which all vertices from subsequent rows are connected);
- each of its internal layers (again, in the "inter-row complete" graph) consists of $n_c^2 n_{ts}^2$ spline segments.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_13

Remark 2. Since the details of multi-spline auto-adaptation and formulae for Hermite spline segments are not of prime importance from the point of view of this paper, they have been omitted. If necessary, they can be found in [8].

4.2 Performance measure estimate

Simulation is the most complex and time-demanding phase of the optimal path search algorithm with the critical element being the computation of an instantaneous net-force[4], F, acting on a sailboat. Although the complete evaluation of a multi-spline spanned on a $G_{ex}\langle n_r, n_c, n_{ts}\rangle$ requires $n_c n_{ts}^2\,[(n_r-3)\,n_c+2]$ simulations (see Section 4.1), and a significant number of them need more than a thousand computations of F, in a typical scenario, simulations for more than 80% of the multi-spline segments are not really necessary as their performance measures are significantly worse. Unfortunately, we do not know them up-front (i.e. before the simulation), and hence the need for a good estimator.

After some experiments with several ML-algorithms (both off-line and on-line) we have found an estimate that reflects the variational character of the performance measure and is accurate enough without being computationally complex. The solution was not at all obvious because of the circular-dependent nature of the problem: the performance measure of a path segment ($t_E = t_S + dt$) obviously depends on the time of reaching its start point (t_S), the initial velocity (v_S), and the segment length (dl). It also depends on F which, in turn, depends on the current position of the sailboat (see Eq. 1) and its instantaneous velocity, which depends (circularly) on F. The pseudo-code of the estimator is presented as Algorithm 1.

The proposed estimator is inspired by *the work-energy theorem* that states that the net-work done by the forces on an object (here the sailboat), $W_{AB} = F\,l_{AB}$, equals the change in its kinetic energy, $0.5\,m_s(v_B^2 - v_A^2)$, where m_s stands for the sailboat mass. The estimate is calculated simultaneously (as a kind of side-effect) with the Gaussian quadrature based computation of the segment length.

Remark 3. The estimation of path segments is also pruned (see line 11 in Algorithm 1).

4.3 Pruned optimal sailboat path search

The graph G_{ex}, on which the solution space (multi-spline) is spanned, is directed, acyclic (DAG) and has a layered structure. These properties can be naturally utilised in a parallel version of the dynamic programming based optimal path search algorithm.

[4] i.e., the sum of all forces acting on a sailboat; this can be found taking into account the wind vector field and the characteristics of the particular sailboat, see [16]

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_13

Algorithm 1: Path segment performance measure estimate

Input:
- s: the segment to be estimated,
- v_S: the initial velocity of the sailboat (at the start point of s),
- t_S: the time of reaching the start point of s (starting from A),
- t_{min}: the best estimated performance measure found so far,
- Δ_t: the penalty value (e.g., 10^3 seconds) used when the sailboat stops,
- E_M: the "safety factor" (e.g., 1.2) for turning-on the rough estimation mode

Output: the estimated value of the performance measure of s

function `estimate`(s; v_S, t_S, t_{min}, Δ_t, E_M)**:**
  $v_1 \leftarrow v_S$; $dt \leftarrow 0$
  foreach x_i *in Gauss nodes for segment* s **do**
    $dl_i \leftarrow$ the length of the i-th sub-segment of s
    $F_i \leftarrow$ the net-force for the current position and velocity
    $s_v \leftarrow v_1^2 + 2\ F_i\ dl_i\ (m_s)^{-1}$
    if $s_v > 0$ **then** $v_2 \leftarrow \sqrt{s_v}$ **else** $v_2 \leftarrow 0$
    $\bar{v} \leftarrow 0.5\ (v_1 + v_2)$
    if $\bar{v} > 0$ **then** $dt \leftarrow dt + dl_i\ (\bar{v})^{-1}$ **else** $dt \leftarrow dt + \Delta_t$
    $v_2 \leftarrow v_1$
    if $(t_S + dt)\ t_{min}^{-1} > E_M$ **then return** (`length` (s) $(\sum_{k=0}^{i} dl_k)^{-1}\ dt$)
  return $t_S + dt$

Principle of Optimality as the algorithm foundation. At the beginning of the search process the *cost matrix* is unknown – the performance measure of each path segment in the graph will be received from simulation using the *Principle of Optimality* [3]. This principle can be expressed for an example path A-$N_{r,c,s}$ (see Fig. 3) in the following way:

$$\widetilde{J}_A^{N_{r,c,s}} = \min_{c_j, s_k} \left(\widetilde{J}_A^{N_{r-1,c_j,s_k}} + J_{N_{r-1,c_j,s_k}}^{N_{r,c,s}} \right) \tag{4}$$

where: $c_j = (0, \ldots, n_c - 1)$, $s_k = (0, \ldots, n_{ts} - 1)$, $J_{N_s}^{N_e}$ is the cost corresponding to the path N_s-N_e (N_s - start node, N_e - end node), $\widetilde{J}$ represents the optimal value of J and $N_{r,c,s}$ is the node of G_{ex} with "graph coordinates" $\langle row, column, tangent\_slope \rangle = \langle r, c, s \rangle$.

Fig. 3 (a visualization of Eq. 4) presents the computation state in which the optimal costs of reaching all nodes in row $r-1$ are known (they were calculated in previous stages of this multi-stage process). The optimal cost of path A-$N_{r,c,s}$ is calculated by performing simulations for all spline segments that join node $N_{r,c,s}$, which is located in row r, with nodes from the previous (i.e. $(r-1)^{\text{th}}$) row.

Multi-spline generated computational topology. The simulation-based multi-stage process can be visualized as a propagation of a "simulation-wave" presented in Fig. 4.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_13

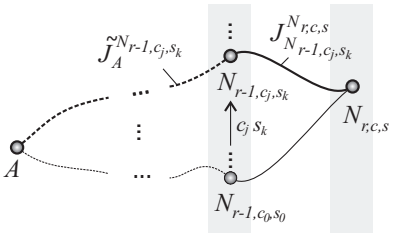

Fig. 3. Principle of Optimality in Dynamic Programming (see Eq.4).

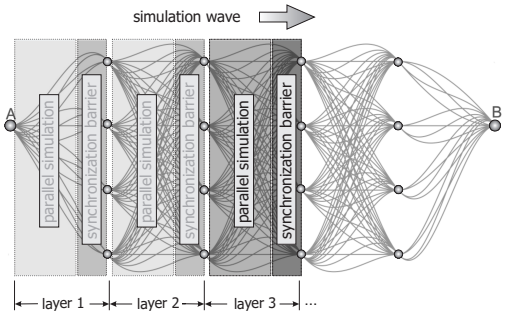

Fig. 4. A sequence of parallel simulations for multi-spline segments from the same layer.

The computation begins from point A in layer 1, taking into account the corresponding initial conditions, and is continued (layer by layer) for the nodes in subsequent rows. On the completion of the simulations for the last layer (i.e. reaching the end node B), we get the optimal path and the corresponding value of the performance measure. The sequential component of the computation – presented in Fig. 4 as a synchronization barrier – is the result of the layer-on-layer dependence (to start simulation for a segment we have to know the corresponding initial conditions, see Eq. 4).

The algorithm. The multi-spline generated computational topology is reflected in the SPMD structure of Algorithm 2. From its two sequential parts only the internal one – evaluating all segments which end in the same entry point (e_p) – needs some clarification. It has four important steps:

1. (lines 3-6) – estimation of all segments which end in entry point
2. (lines 7-10) – simulation for the k segments which have the best estimates (to verify the estimation accuracy; the value of k can be a constant or auto-adaptive variable);
3. (lines 11-13) – computation of the estimation accuracy measure Δt_{s_i};
4. (lines 14-17) – simulation for the rest of the "promising" segments (if there are any).

Pruning, discussed in the next paragraph, is applied in steps 1, 2, and 4.

External and internal pruning. Significant reduction of the average-case time complexity of the algorithm is important at least for two reasons. Firstly, we often need to know the solution as soon as possible (sometimes for safety reasons). Secondly, since we only use on-board computers, energy efficiency is critical while at sea. This is why the reference algorithm has been augmented with two-level pruning. The first pruning level (*external*) is related to the explicit reduction of the search space (lines 14-17 in Algorithm 2) using the performance measure estimate. The second level pruning mechanism (*internal*) is embedded both in

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_13

Algorithm 2: SPMD-parallel pruned search of the optimal path

Input:
- g_{AB}: initial (layered) grid with the start point, A, and the target point, B,
- $\boldsymbol{v}_t$: (true) wind vector field (see Eq.1),
- MODEL: sailboat model definition

Output: minimum time, t_{min}, sailboat path

```
1  foreach layer in grid g_AB do
2      @parallel foreach entry point e_p of its nodes do
3          sgms⟨e_p⟩ ← select segments ending in the entry point e_p
4          foreach s_i in sgms⟨e_p⟩ do
5              t_{s_i}^(est) ← estimate (s_i; v_0i, t_0i, t_min, ...)
6              if t_{s_i}^(est) < t_min then t_min ← t_{s_i}^(est)
7          k_best ← select k segments with best estimates
8          foreach s_i in k_best do
9              t_{s_i}^(sim) ← pruned adaptive simulation for (s_i)
10             if t_{s_i}^(sim) < t_best then t_best ← t_{s_i}^(sim)
11         foreach s_i in k_best do
12             Δt_{s_i} ← |t_{s_i}^(est) − t_best| t_best^{-1}
13             if Δt_{s_i} > Δt_max^(k) then Δt_max^(k) ← Δt_{s_i}
14         foreach s_i in { sgms⟨e_p⟩ \ k_best } do
15             if |t_{s_i}^(est) − t_best| t_best^{-1} < C_M Δt_max^(k) then
16                 t_{s_i}^(sim) ← pruned adaptive simulation for (s_i)
17                 if t_{s_i}^(sim) < t_best then t_best ← t_{s_i}^(sim)
18         save the best segment for e_p
```

the process of estimation (line 11 in Algorithm 1) and in simulation (lines 8-10 and 14-17). It terminates the computation from "inside" that cannot lead to a better solution than the reference one.

Complexity analysis. Algorithm 2 average-case *time complexity* is determined by the number of solution space refinements, n_i, the average number of net-force evaluations[5] for a single path segment, $\bar{n}_F$, and the number of such segments, $n_c n_{ts}^2 \left[(n_r - 3)\, n_c + 2 \right]$ (see Section 4.1). For the sequential version of the algorithm it can be expressed as:

$$T_s = \Theta \left(n_i \; \bar{n}_F \; n_r \; n_c^2 \; n_{ts}^2 \right). \tag{5}$$

[5] values of F are needed both in estimation and in simulation; Runge-Kutta-Fehlberg 4(5) method, used in the simulator, requires at each step six evaluations of F

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_13

In the SPMD-parallel version of the algorithm, the evaluations for all nodes in a given row can be performed in parallel (using p processing units), thus:

$$T_p = \Theta\left(n_i\ \bar{n}_F\ n_r\ n_c\ n_{ts}\left\lceil\frac{n_c\ n_{ts}}{p}\right\rceil\right). \tag{6}$$

As for the reference algorithm [8], Algorithm 2 *space complexity* formula, $\Theta\left(n_r n_c n_{ts}\right)$, arises from the solution space representation.

Remark 4. The aim of pruning is to significantly reduce $\bar{n}_F$.

5 Experimental verification

To demonstrate the effectiveness of the pruned optimal sailboat path search algorithm, a series of experiments was carried out using a MacBook Pro[6] with macOS 10.15.7 and OpenCL 1.2, having two (operational) OpenCL-capable devices: Intel Core i7-3740QM @ 2.7 GHz (the CPU) and Intel HD Graphic 4000 (the integrated GPU). The aim of the experiments was to investigate three important aspects of the algorithm: the accuracy of the performance measure estimator, the efficacy of different kinds of pruning, and the SPMD-parallelisation efficiency. The results are presented in the subsequent paragraphs.

Performance measure estimator accuracy. This element has an explicit impact on the search space reduction since the more accurate the estimator is, the more segments can be omitted in the simulation phase (see lines 14-17 in Algorithm 2). The results of its experimental evaluation are given in the form of a violin plot in Fig. 5. The plot shows the distributions of estimation relative errors (i.e., $|(t_{est} - t_{sim})/t_{sim}|$) for path segments from different search spaces.

Remark 5. In all cases the medians of relative estimation errors were smaller than 0.003 (i.e., 0.3%), which confirms the very high accuracy of the proposed estimator[7].

Pruning efficacy. The application of different types of pruning is one of the two ways of lowering the total computation time. Its efficacy was verified using the reduction of the number of net-force computations, $(\bar{n}_F^{(base)} - \bar{n}_F^{(prun)})/\bar{n}_F^{(base)}$, as the measure. The corresponding experimental results are given in Table 1 and Fig. 6. To verify the accuracy of $\bar{n}_F$ as the measure of the algorithm time complexity (see Eqns. 5 and 6), the duration of sequential computations, t_{sim}, was also measured. The results shown in Table 1 prove the very high accuracy since the Pearson correlation coefficient, $\rho(\bar{n}_F, t_{sim})$, is equal to 1 (up to 5 decimal places), which means (almost) perfect linear dependence of the two variables.

Remark 6. Different types of pruning working together reduced the sequential computation time from seventeen to twenty times (see Fig. 6).

[6] with 16GB of DDR3 1600 MHz RAM; the laptop manufactured in 2013

[7] the samples sizes (i.e., the number of segments) used to compute the distributions of errors were very large: from 29 760 for $n_c = 16$ to 950 528 for $n_c = 128$

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_13

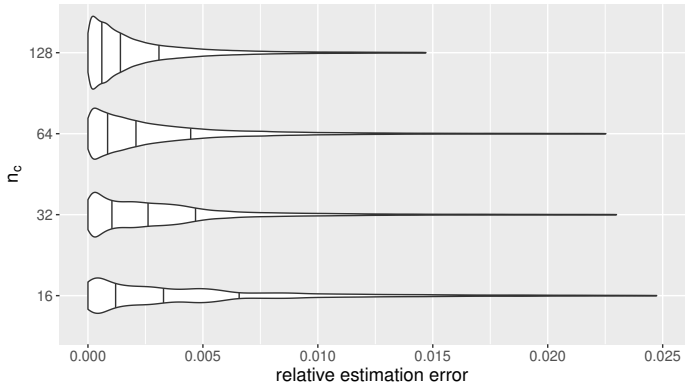

Fig. 5. Distributions of relative errors of the performance measure estimate, e, for grids with different n_c. In each case the calculation was based on the k best values (lines 7-10 in Algorithm 2) with outliers excluded (i.e., $e > \bar{e} + 3\ \sigma_e$).

Table 1. Efficacy of different kinds of pruning: average numbers of net-force evaluations, $\bar{n}_F$, and execution times, $\bar{t}_{sim}$ (in seconds), for different n_c. The solution space with $n_r = 32$, $n_{ts} = 8$; two refinements. *Auxiliary notation*: IPS - Internal Pruning in simulation, EP - External Pruning [+estimation], IPE - Internal Pruning in Estimation.

n_c	BASE		IPS		EP		IPS+EP		IPS+IPE+EP	
	$\bar{n}_F$	t_{sim}	$\bar{n}_F$	t_{sim}	$\bar{n}_F$	t_{sim}	$\bar{n}_F$	t_{sim}	$\bar{n}_F$	t_{sim}
16	748.8	206.0	165.5	44.3	172.1	15.0	53.7	14.1	44.1	12.2
32	735.8	812.4	142.8	154.9	170.5	59.9	51.6	55.0	42.6	47.3
64	731.4	3351.9	136.0	600.6	170.2	239.3	50.7	210.9	41.6	182.4
128	731.7	12950.2	137.6	2428.7	145.1	810.0	46.0	769.8	37.0	654.8

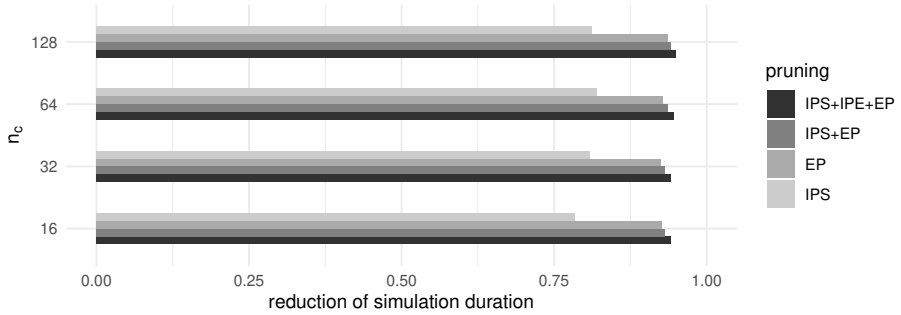

Fig. 6. Reduction of simulation duration for different kinds of pruning and different n_c. The auxiliary notation as in Table 1.

SPMD-*parallelisation efficiency.* Parallelisation is the second way of lowering the total computation time. Contemporary mobile/embedded computers are usually

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_13

equipped with more than one type of processor, typically one CPU and one or two GPUs. OpenCL makes it possible to use these heterogeneous platforms effectively since the same code can be executed on any OpenCL-capable processor. The performance comparison of the two available processors is shown[8] in Table 2 and Fig. 7. The results refer to the case of all three types of pruning being

Table 2. SPMD-parallelisation efficiency: average execution times, $\bar{t}_{sim}$ (in seconds), and standard deviations, σ, from ten runs of the parallel version of the algorithm for different n_c. The solution space with $n_r = 32$, $n_{ts} = 8$; two refinements.

n_c	i7-3740QM (seq)		i7-3740QM+OCL		HD4000+OCL	
	$\bar{t}_{sim}$	σ	$\bar{t}_{sim}$	σ	$\bar{t}_{sim}$	σ
16	12.2	0.05	2.53	0.06	1.85	0.00
32	47.3	0.88	5.74	0.10	4.34	0.05
64	182.4	3.31	22.16	0.27	8.89	0.01
128	654.8	19.04	72.87	0.21	16.55	0.12

active with the pair of columns annotated with "seq" corresponding to the reference (sequential) variant of the algorithm. The two platforms differ significantly (see Fig. 7), and therefore there is definitely room for processor allocation plan optimisation done before or during code execution (run-time).

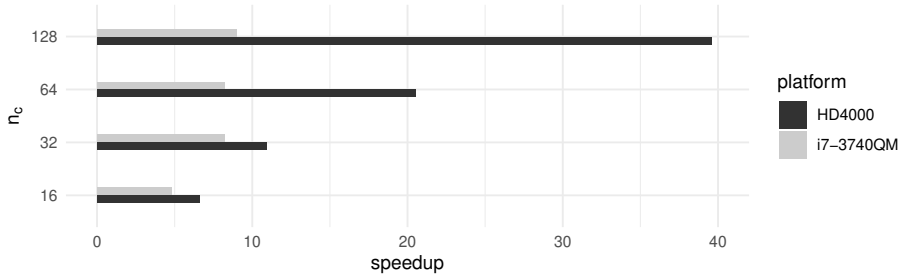

Fig. 7. SPMD-parallelisation efficiency: speed-ups, t_{seq}/t_{par}, for different Open-CL platforms and n_c ($t_{(.)}$ measured with $-Os$ flag). The remaining parameters as in Table 2.

Remark 7. The maximum observed speedup was 39.56 (see Fig. 7). Setting the reference to the case with no pruning gives in total $19.78 \times 39.56 \approx 782.5$ times faster execution.

[8] the results for $n_c < 16$ are omitted because the search spaces they define are too coarse-grained; it is worth noting, however, that for such cases the CPU was faster.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_13

6 Conclusions

It has been shown that the simulation-based optimal sailboat path planning algorithm can be both effective and (energy) efficient. The three key elements in achieving this have been an accurate and efficient estimator of the performance measure (sailing duration), the two-level pruning (which augments the estimator-based search space reduction with smart simulation and estimation techniques), and the OpenCL-based SPMD-parallelisation of the algorithm.

The numerical results show the high accuracy of the estimator (the medians of relative estimation errors were smaller than 0.003, see Fig. 5), the high efficacy of the two-level pruning (search space and computing time reduction from seventeen to twenty times, see Table 1 and Fig. 6), and the high parallel speedup (its maximum observed value was 39.56, see Fig. 7). Combining these effects has given (up to) 782.5 times faster execution.

The proposed approach can be applied to various domains. It can be considered as an optimal path planing framework parametrised by a problem specific performance measure heuristic/estimator. Further exploration of this idea could be the first possible future research direction. Another could be the algorithm space complexity reduction.

Acknowledgement. Research presented in this paper was supported by the funds assigned to AGH University of Science and Technology by the Polish Ministry of Science and Higher Education.

References

1. Bellman, R.: The theory of dynamic programming. Buletin of the American Mathematical Society **60**, 503–515 (1954)
2. Bellman, R.: On a Routing Problem. Quarterly of Applied Mathematics **16**, 87–90 (1958)
3. Bellman, R., Dreyfus, S.: Applied Dynamic Programming. Princeton University Press, Princeton, New Jersey (1962)
4. Bertsekas, D.P.: Dynamic Programming and Optimal Control. Belmont, Mass, Athena Scientific, 2nd edn. (2000)
5. Ceriotti, M., Vasile, M.: MGA trajectory planning with an ACO-inspired algorithm. Acta Astronautica **67**(9–10), 1202–1217 (2010)
6. Crauser, A., Mehlhorn, K., Meyer, U., Sanders, P.: A parallelization of Dijkstra's shortest path algorithm. In: Brim, L., et al. (eds.) Mathematical Foundations of Computer Science 1998, LNCS, vol. 1450, pp. 722–731. Springer (1998)
7. Dalang, R.C., Dumas, F., Sardy, S., Morgenthaler, S., Vila, J.: Stochastic optimization of sailing trajectories in an upwind regatta. Journal of the Operational Research Society (2014)
8. Dębski, R.: An adaptive multi-spline refinement algorithm in simulation based sailboat trajectory optimization using onboard multi-core computer systems. Int. J. Appl. Math. Comput. Sci. **26**(2), 351–365 (2016)
9. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik **1**, 269–271 (1959)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_13

10. Dębski, R.: High-performance simulation-based algorithms for alpine ski racer's trajectory optimization in heterogeneous computer systems. Int. J. Appl. Math. Comput. Sci. **24**(3), 551–566 (2014)
11. Harabor, D., Grastien, A.: Online graph pruning for pathfinding on grid maps. vol. 2 (01 2011)
12. Harish, P., Narayanan, P.: Accelerating large graph algorithms on the gpu using cuda. In: High performance computing–HiPC 2007, pp. 197–208. Springer (2007)
13. Jasika, N., Alispahic, N., Elma, A., Ilvana, K., Elma, L., Nosovic, N.: Dijkstra's shortest path algorithm serial and parallel execution performance analysis. In: MIPRO, 2012 Proc. of the 35th Int. Convention. pp. 1811–1815. IEEE (2012)
14. Kuffner, J.J., LaValle, S.M.: Rrt-connect: An efficient approach to single-query path planning. In: Proc. 2000 IEEE Int'l Conf. on Robotics and Automation. vol. 2, pp. 995–1001. IEEE (2000)
15. Lewis, R.M., Torczon, V., Trosset, M.W.: Direct search methods: Then and now. Journal of Computational and Applied Mathematics **124**, 191–207 (2000)
16. Marchaj, C.: Aero-hydrodynamics of Sailing. Adlard Coles Nautical (2000)
17. Park, C., Pan, J., Manocha, D.: Real-time optimization-based planning in dynamic environments using GPUs. In: Robotics and Automation (ICRA), 2013 IEEE International Conference on. pp. 4090–4097. IEEE (2013)
18. Pêtres, C., Romero-Ramirez, M.A., Plumet, F.: Reactive path planning for autonomous sailboat. In: Advanced Robotics (ICAR), 2011 15th International Conference on. pp. 112–117. IEEE (2011)
19. Pontryagin, L.S., Boltyanski, V.G., Gamkrelidze, R.V., Mischenko, E.F.: The Mathematical Theory of Optimal Processes. Interscience, NY (1962)
20. Pošík, P., Huyer, W., Pál, L.: A comparison of global search algorithms for continuous black box optimization. Evolutionary Computation pp. 1–32 (2012)
21. Rippel, E., Bar-Gill, A., Shimkin, N.: Fast graph-search algorithms for general-aviation flight trajectory generation. Journal of Guidance, Control, and Dynamics **28**(4), 801–811 (2005)
22. Singla, G., Tiwari, A., Singh, D.P.: New approach for graph algorithms on gpu using cuda. International Journal of Computer Applications **72**(18), 38–42 (June 2013), published by Foundation of Computer Science, New York, USA
23. Stelzer, R., Pröll, T.: Autonomous sailboat navigation for short course racing. Robotics and autonomous systems **56**(7), 604–614 (2008)
24. Stillwell, J.: Mathematics and its History. Springer, New York, third edn. (2010)
25. von Stryk, O., Bulirsch, R.: Direct and indirect methods for trajectory optimization. Annals of Operations Research **37**(1), 357–373 (1992)
26. Sussmann, H.J., Willems, J.C.: 300 years of optimal control: from the brachystochrone to the maximum principle. Control Systems, IEEE **17**(3), 32–44 (1997)
27. Szłapczyński: Customized crossover in evolutionary sets of safe ship trajectories. Int. J. Appl. Math. Comput. Sci **22**(4), 999–1009 (2012)
28. Vasile, M., Locatelli, M.: A hybrid multiagent approach for global trajectory optimization. Journal of Global Optimization **44**(4), 461–479 (aug 2009)
29. Wang, J., Li, B., Meng, M.Q.H.: Kinematic constrained bi-directional RRT with efficient branch pruning for robot path planning. Expert Systems with Applications **170**, 114541 (2021)
30. Zhang, C., Zhou, L., Li, Y., Fan, Y.: A dynamic path planning method for social robots in the home environment. Electronics **9**, 1173 (07 2020)
31. Życzkowski, M.: Sailing route planning method considering various user categories. Polish Maritime Research **27** (10 2020)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_13