Pareto Optimal Swimmers

Siddhartha Verma, Panagiotis Hadjidoukas, Philipp Wirth, Diego Rossinelli, Petros Koumoutsakos
Background: Intermittent locomotion

- Intermittent locomotion: A strategy used by many animals: Birds, dolphins, jellyfish…
  - “… animal pauses lasting from milliseconds to minutes…”
  - “… on average nearly 50% of locomotion time.”
    Kramer & McLaughlin (2001)

- Intermittent swimming patterns in zebrafish
  - ‘Burst-coast’ mode of swimming: quick flicks (burst) followed by unpowered glide (coast)
  - Lower **energy** expenditure
    (Weihs 1974, Wu et al. 2007)
  - Reduce **disturbances** created: avoid alerting predators/prey (Wieskotten et al., 2010)
  - Stabilized **sensory** field
    (Kramer & McLaughlin 2001)
  - Disadvantage: reduced average speed
Burst-Coast swimming

- Advantageous traits: *Evolutionary advantage* over rival species
- Majority of computational & theoretical studies: continuous, steady motion
  - Numerical study on Burst-Coast: *Chung 2009*, parameters specified a-priori
- **Goal**: Discover best intermittent swimming patterns for combination of:
  - High Speed
  - Low Energy expenditure
- Benefits of Multi-Objective Optimization:
  - Balance between speed and efficiency (two *conflicting* metrics)
  - Not just a single design point: Planners can adapt design according to mission requirements
  - Discover optima not limited by biological constraints

Credit: Prof. Hu, Univ. Essex
Numerical methods

- Naiver-Stokes: Remeshed vortex methods (2D)
  - Solve vorticity form of incompressible Navier-Stokes
  \[
  \frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \mathbf{u} \cdot \nabla \mathbf{u} + \nu \nabla^2 \omega + \lambda \nabla \times (\chi (\mathbf{u}_s - \mathbf{u}))
  \]
  - Brinkman penalization
    - Accounts for fluid-solid interaction
      Angot et al., Numerische Mathematik (1999)
  - 2D: Wavelet-based adaptive grid
    - Cost-effective compared to uniform grids
- 3D: Uniform grid Finite Volume solver
Swimming-efficiency & Cost of Transport

- Efficiency for self-propelled bodies:
  - Ratio of Thrust to Work done against fluid

\[ \eta = \frac{P_{\text{Thrust}}}{P_{\text{Thrust}} + \max(P_{\text{Def}}, 0)} \]

- Cost of Transport:
  - Energy spent for travelling unit distance

\[ CoT(t) = \frac{\int_{t-T_p}^{t} \max(P_{\text{Def}}, 0) dt}{\int_{t-T_p}^{t} \|u\| dt} \]
Parameterization of intermittent motion

- Undulation of midline modelled as travelling wave:
  \[ y_{midline}(s, t) = f(t) \alpha (s + bL) \sin \left[ (2\pi \left( \frac{s}{L} - \frac{t}{T} + \phi \right) \right] \]

- Defined over 4 stages of burst-and-coast
  1. \( \Delta t_{\text{decel}} \) (f from 1 to 0)
  2. \( \Delta t_{\text{coast}} \) (f = 0)
  3. \( \Delta t_{\text{accel}} \) (f from 0 to 1)
  4. \( \Delta t_{\text{steady}} \) (f = 1)

- 4 parameters to be optimized for 2 objectives:
  - Average speed
    \[ \text{Average speed} = \frac{1}{T} \int_{t-T}^{t} \|\mathbf{u}\|\,dt \]
  - Cost of Transport
    \[ \text{Cost of Transport} = \int_{t-T}^{t} \max(P_{def}, 0)\,dt \]
    \[ \int_{t-T}^{t} \|\mathbf{u}\|\,dt \]
Optimization: Parallelized NSGA-II

Basic structure of Genetic Algorithms:

- Population Initialization
- Fitness Evaluation
- Selection of the Fittest
- Mating & Mutation
- convergence?

Multi-objective selection of non-dominated individuals:

- Maintain a diverse approximation of the Pareto front
  - NSGA-II algorithm (Deb et al., 2002)
- Scheduling of evaluations to use comp. resources efficiently
  - TORC library (Hadjidoukas et al., 2012)

Pareto front

Fittest individuals: no other individuals present in this box
Parallelization

- Parallelization with \( \Pi 4U \): minimal effort
- No details of parallel machinery required in coding


Essentially, we took Deb et al’s C-code, and coupled it with \( \Pi 4U \)

```c
int main(int argc, char *argv[]) {
    double result[100];
    for (int i=0; i<100; i++) {
        double d[2] = {drand48(), drand48()};
        task(d, &result[i]);
    }
    return 0;
}
```

Function evaluation

```c
void task(double *x, double *y) {
    *y = x[0] + x[1];
}
```

Serial main-code

```c
int main(int argc, char *argv[]) {
    double result[100];
    for (int i=0; i<100; i++) {
        double d[2] = {drand48(), drand48()};
        task(d, &result[i]);
    }
    return 0;
}
```

OpenMP code

```c
int main(int argc, char *argv[]) {
    double result[100];
    #pragma omp parallel for
    for (int i=0; i<100; i++) {
        double d[2] = {drand48(), drand48()};
        task(d, &result[i]);
    }
    return 0;
}
```

TORC code

```c
int main(int argc, char *argv[]) {
    double result[100];
    torc_init(argc, argv, MODE_MW);
    for (int i=0; i<100; i++) {
        double d[2] = {drand48(), drand48()};
        torc_task(-1, task, 2,
            2, MPI_DOUBLE, CALL_BY_COP, // IN (COPY)
            1, MPI_DOUBLE, CALL_BY_RES, // OUT
d, &result[i]);
    }
    torc_waitall();
    return 0;
}
Task scheduling

- Each fitness evaluation is mapped to one asynchronous task
- Single worker thread per compute node
  - Launch MRAG-I2D with the fork-exec system calls
  - Perform data exchange through the local filesystem
- Correcting load imbalance introduced by large variance of simulation time
  - Population size larger than the number of workers
  - Task-stealing mechanism of the library
  - Schedule tasks with larger expected runtime first
- Estimation of expected runtime using ANN
  - Training set: input parameters and measured runtimes
  - Output: expected runtime for new individuals

Distribution and scheduling scheme
- Sort the tasks in descending order w.r.t. runtime
- Distribute first round cyclically to the workers
- Insert the rest into a single queue
Parallel performance on Piz Daint

Strong scaling of **MRAG-I2D** on a single node

- **Lower time-to-solution** minimizes the overall time for the optimization
- **Strong scaling for a single generation of NSGA-II** (population size 32)
  - Parallel efficiency $\geq 90\%$ on up to 16 nodes (2 simulations per worker)
  - 50\% if only one simulation is assigned to each worker (1 Sim/w)

Performance of NSGA-II for 27 generations

- **2 and 4 Sim/w**: Improved parallel efficiency
- **1 Sim/w**: 50.3\% parallel efficiency

- **4 Sim/w + ML-based scheduling**: Best performance
- **2 Sim/w + ML-based scheduling**: Comparable to the plain 4 Sim/w case

**Advantage** of run-time based scheduling
Evolution of the population

- Optimisation done for two Reynolds numbers (impacts burst-coast: Müller 2000)
  - Re = 400 (larval zebrafish)
  - Re = 4000 (adult zebrafish)

- Steps for optimisation
  - Initialise population randomly
  - Iterate using the optimisation algorithm
  - Converge to final population

- Consider 3 optimal individuals
  - Efficient swimmer - lazy, but efficient
  - Generalist - compromise between speed and efficiency
  - Fastest Swimmer
Optimal strategies for Larva (Re = 400)

<table>
<thead>
<tr>
<th>Swimmer</th>
<th>CoT</th>
<th>Avg. Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steady</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Efficient</td>
<td>0.62</td>
<td>0.53</td>
</tr>
<tr>
<td>Generalist</td>
<td>0.87</td>
<td>0.84</td>
</tr>
<tr>
<td>Fast</td>
<td>2.70</td>
<td>1.47</td>
</tr>
</tbody>
</table>
Optimal strategies for Adult (Re = 4000)

- Efficient
- Generalist
- Fast

<table>
<thead>
<tr>
<th>Swimmer</th>
<th>CoT</th>
<th>Avg. Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steady</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Efficient</td>
<td>0.48</td>
<td>0.63</td>
</tr>
<tr>
<td>Generalist</td>
<td>0.77</td>
<td>0.88</td>
</tr>
<tr>
<td>Fast</td>
<td>3.57</td>
<td>1.95</td>
</tr>
</tbody>
</table>
Observed behaviour

- The **efficient** swimmer spends a large amount of time coasting
  - Both Re=400, and Re=4000

- The **generalist** adopts a mixture of acceleration/deceleration, and a small coasting-time

- The **fast** swimmer - distinct difference
  - Re=400 (larva) : Prefers to swim steadily
  - Re=4000 (adult) : Short extreme acceleration burst, followed by long coasting duration
  - Expected, since viscous drag impedes larva to a much greater extent during coasting
Ongoing work with 3D simulations (similar to 2D)

- Starting point: optimal solutions from 2D
  - Reduce computational cost
- Body colour: thrust production
  - Gold => thrust  Black => drag
- Some distinct characteristics
  - Fast: steady, continuous swimming
  - Generalist: Large acceleration => large intermittent thrust generation (both midsection and tail)
  - Efficient: much longer duty cycle - longest coast period
- Coast phase: not completely still
  - Different from biological behaviour (rest to overcome fatigue)
  - Robotics: No need for fatigue recovery
Summary

- Used MO to discover optimal intermittent swimming modes
- Burst-Coast motion is an outcome of the optimisation
- Resembles behaviour observed in nature

- TORC/Π4U - a tool for effortless serial-code parallelisation

```c
int main(int argc, char *argv[]) {
    double result[100];
    torc_init(argc, argv, MODE_MW);
    for (int i=0; i<100; i++) {
        double d[2] = {drand48(), drand48()};
        torc_task(-1, task, 2, 2, MPI_DOUBLE, CALL_BY_COP, // IN (COPY)
                1, MPI_DOUBLE, CALL_BY_RES, // OUT
d, &result[i]);
    }
    torc_waitall();
    return 0;
}
```

TORC code
Backup
Vortices in the wake

- Depending on Re and swimming-kinematics, double row of vortex rings in the wake
- Our goal: **exploit** the flow generated by these vortices
Simulation Cost (2D)

- Wavelet-based adaptive grid
  - https://github.com/cselab/MRAG-I2D
  - Rossinelli et al., JCP (2015)

- Production runs (Re=5000)
  - Domain : [0,1] x [0,1]
  - Resolution : 8192 x 8192
  - 1600 points along fish midline
  - Running with 24 threads (12 hyper threaded cores - Piz Daint)
    - 10 tail-beat cycles : 27000 time steps
    - Approx. 96 core hours: 1 second/step

- Training simulations (lower resolution)
  - Resolution : 2048 x 2048
  - 10 tail-beat cycles : 36 core hours
  - Learning converges in : 150,000 tail-beats
  - 0.54 Million core hrs per learning episode
Simulation Cost (3D)

- Uniform grid Finite Volume solver

- Production runs (Re=5000)
  - Domain : \([0, 1] \times [0, 0.5] \times [0, 0.25]\)
  - Resolution : 6144 x 3072 x 768
  - 600 points along fish midline
  - Running on 128 nodes, 24 threads each (Hybrid MPI + OpenMP : Piz Daint)
  - 10 tail-beat cycles : 21000 time steps
  - Approx. 37,000 core hours: 3 seconds/timestep

- Training simulations (lower resolution)
  - Resolution : 2048 x 1024 x 512
  - Expected : 1.2 Million core hours per learning episode
Latest Results

- 2D simulations
Latest Results

- 3D simulations