Membrane Pseudo-Bacterial Potential Field for Mobile Robot Path Planning

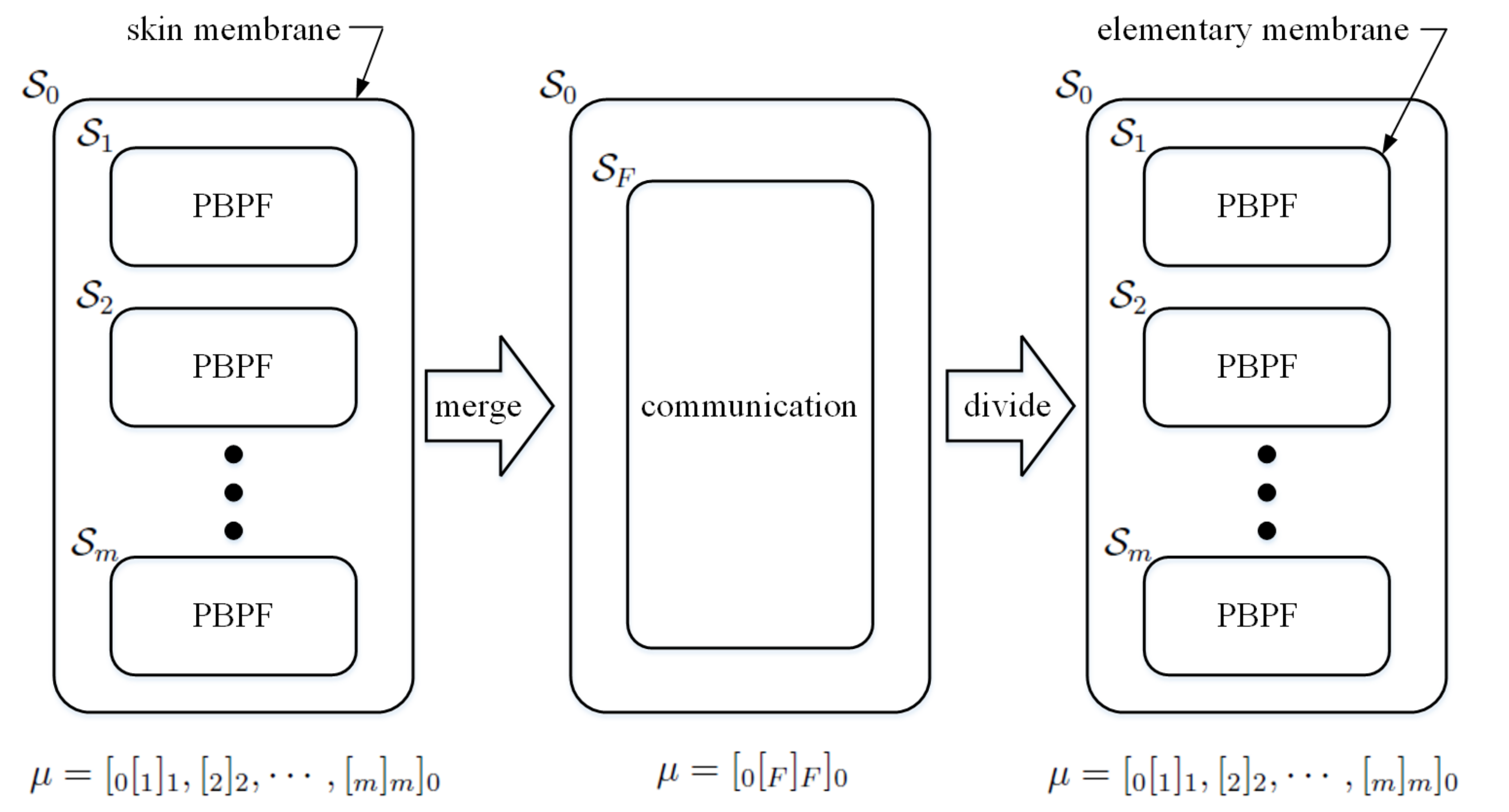
Ulises Orozco-Rosas, Kenia Picos, Antonio S. Montemayor

ulises.orozco@cetys.mx, kenia.picos@cetys.mx, antonio.sanz@urjc.es

MemPBPF algorithm

The membrane pseudo-bacterial potential field (MemPBPF) algorithm for path planning consists of a cell-like P system that evolves a set of parameters required for the artificial potential field (APF) method. These parameters are the attractive proportional gain k_a, the repulsive proportional gain k_r, and the step size η.

The MemPBPF algorithm employs a dynamic structure μ with active membranes and rules, such as membrane division and merger, as seen in the figure below. The membrane merger is helpful to enhance the information communication among individuals (set of parameters, k_a, k_r, and η) and the membrane division is beneficial to improve the search capability.



$\mu = [_0[_1]_1, [_2]_2, \cdots, [_m]_m]_0$ $\quad$ $\mu = [_0[_F]_F]_0$ $\quad$ $\mu = [_0[_1]_1, [_2]_2, \cdots, [_m]_m]_0$

The computational process outlined by the above figure consists of three steps: First, each elementary membrane S_i, composed of a pseudo-bacterial potential field (PBPF) evolves the individuals. Each individual is codified with a set of parameters. This first stage aims to find the best individual in each elementary membrane S_i.

Second, all the elementary membranes S_i, merge into one membrane S_F, containing all the individuals. Communication rules are applied, which first separate the best individual of each elementary membrane to find the global best individual (k_a, k_r, and η) and send out into the skin membrane S_0 a copy of this global best individual to preserve the global best solution.

The communication continues in the merged membrane S_F to exchange information among the elementary membranes S_i that will be formed in the next step. During the merge process, each subpopulation is maintained, and a copy of the best individuals will replace the worst individuals (a portion of the subpopulation) to improve the subpopulation in each elementary membrane.

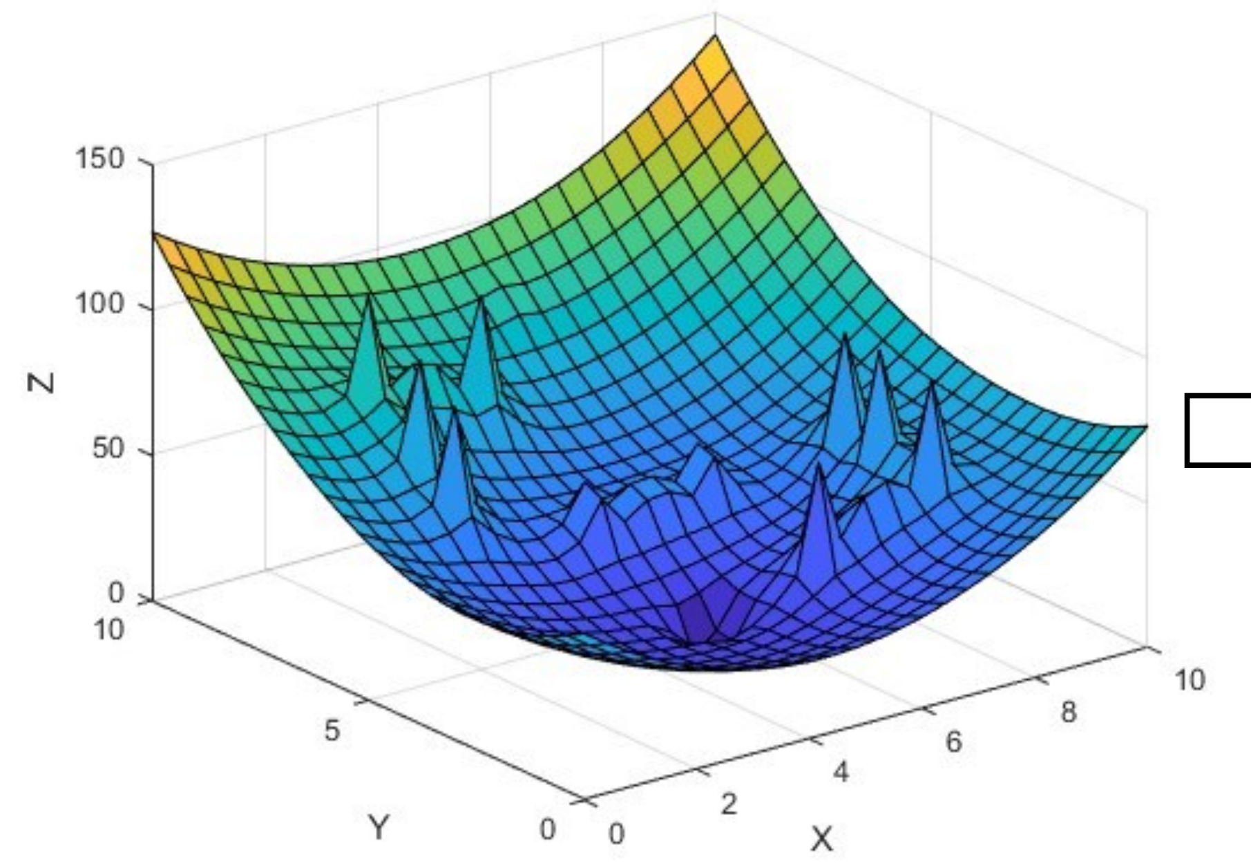
Third, the process is repeated to refine the sets of parameters to be able to perform optimal or close-to-optimal path planning. Inside each elementary membrane S_i, the PBPF performs the evolution process. The PBPF starts with the creation of a random population of individuals *P(t)*, each individual (solution) is codified with the values of the proportional gains, attraction k_a, repulsion k_r, and the step size η required to generate the path.

For the parallel evaluation on GPU, where each path is evaluated. We compute:

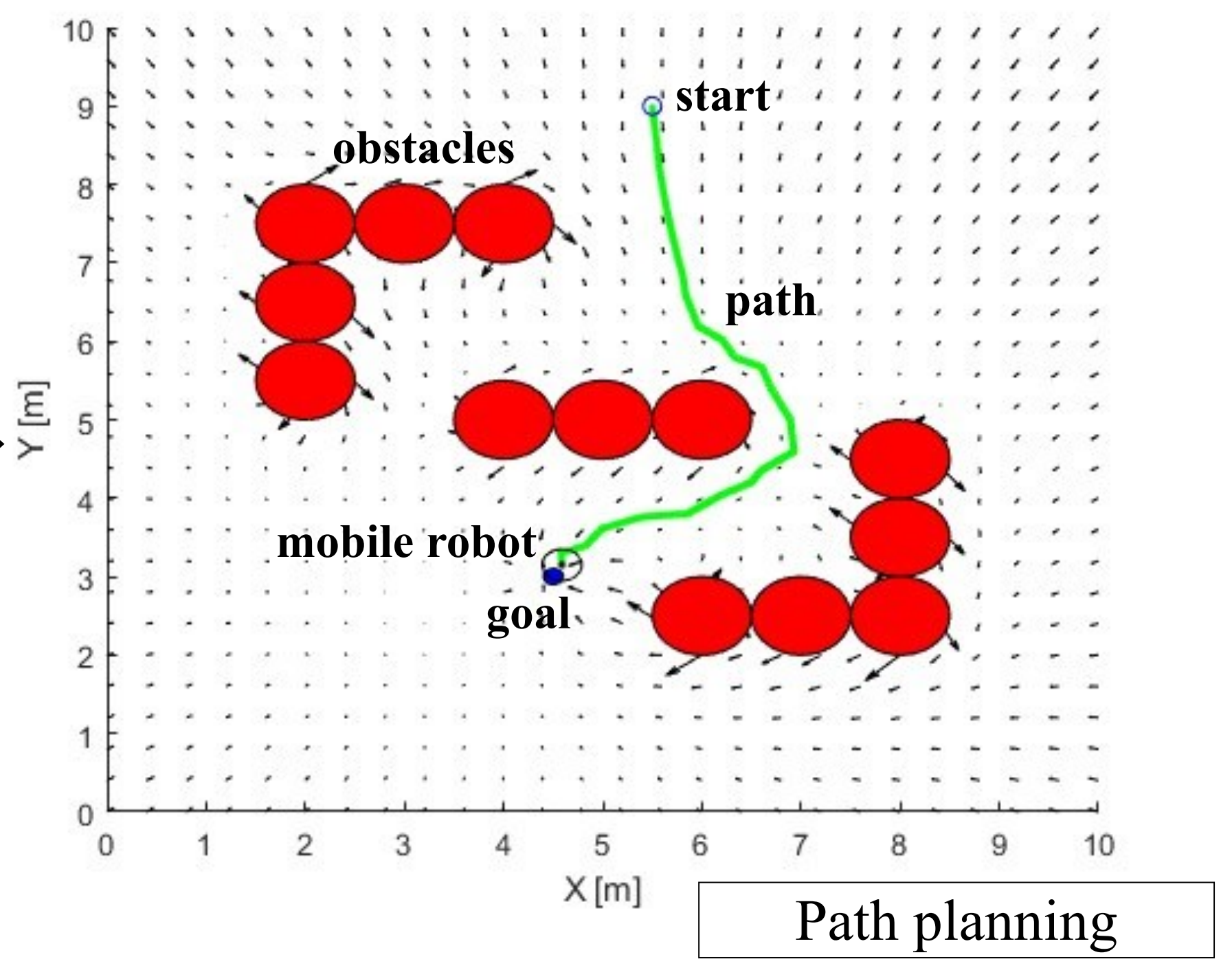
$$U_{total}(q) = \frac{1}{2}\left[k_a(q-q_f)^2 + k_r\left(\frac{1}{\rho}-\frac{1}{\rho_0}\right)^2\right] \qquad F_{total}(q) = -\nabla U_{total}(q) \qquad S = \sum_{i=0}^{m} \|q_{i+1}-q_i\|$$

After the parallel path evaluation on GPU, the selection, crossover, and mutation operators are applied to evolve the individuals in *P(t)*. All the path planning is enclosed in an iterative process until the maximum number of generations has been achieved.

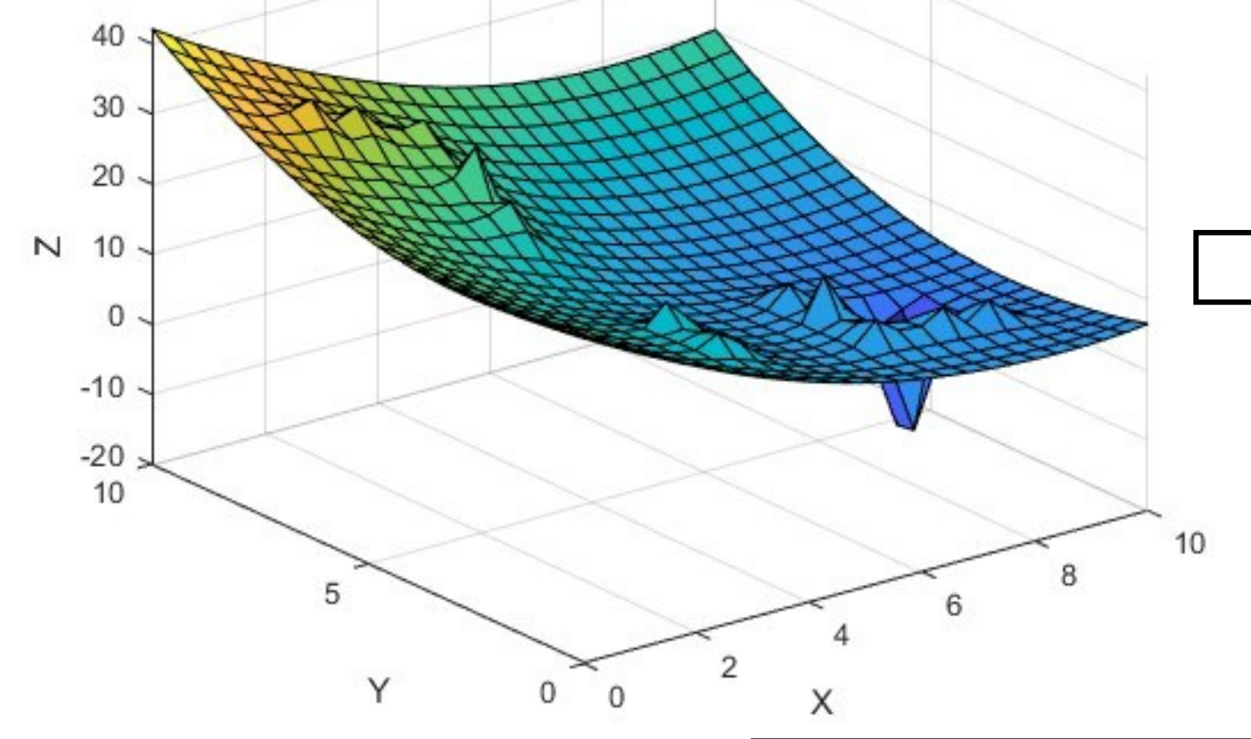
Path Planning Results



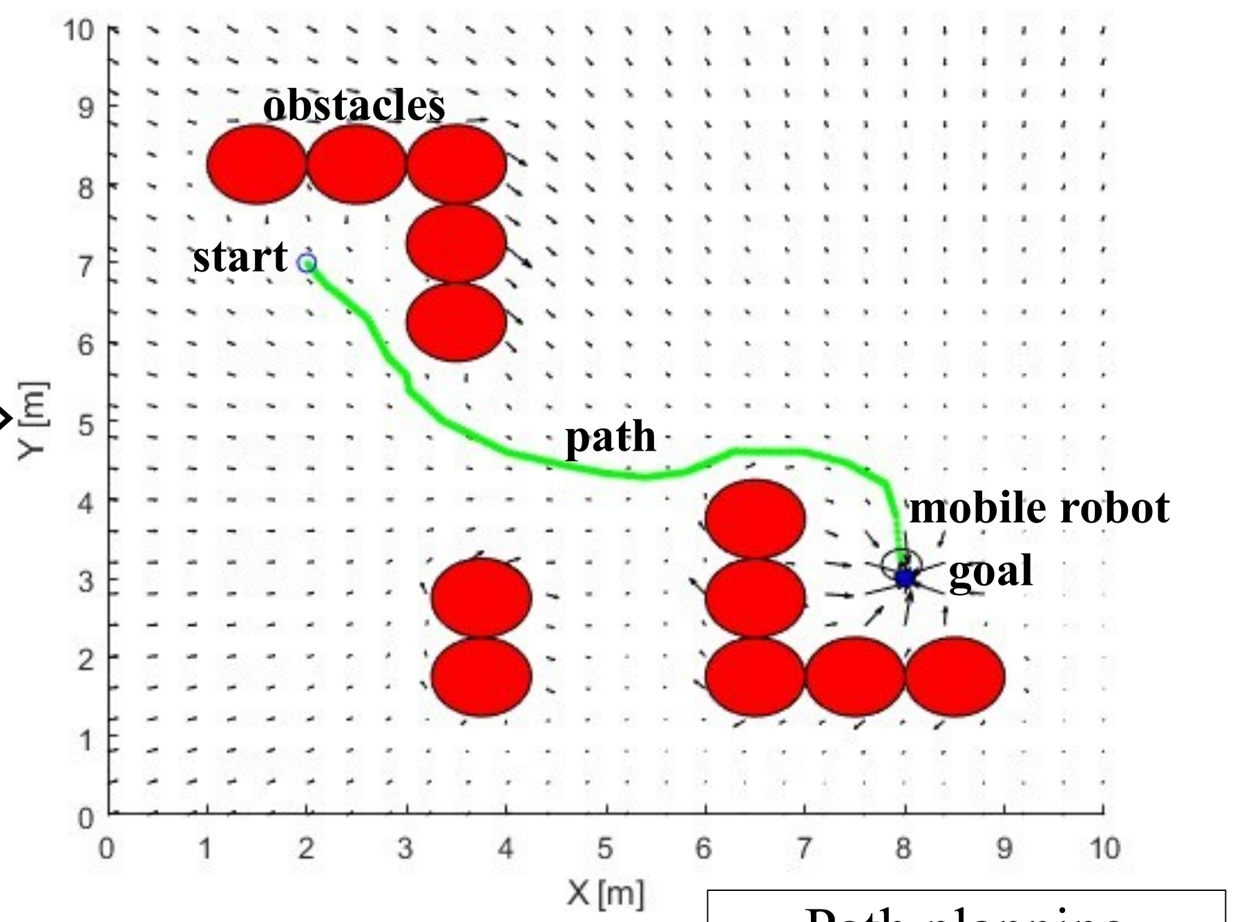
APF surface



Path planning



APF surface



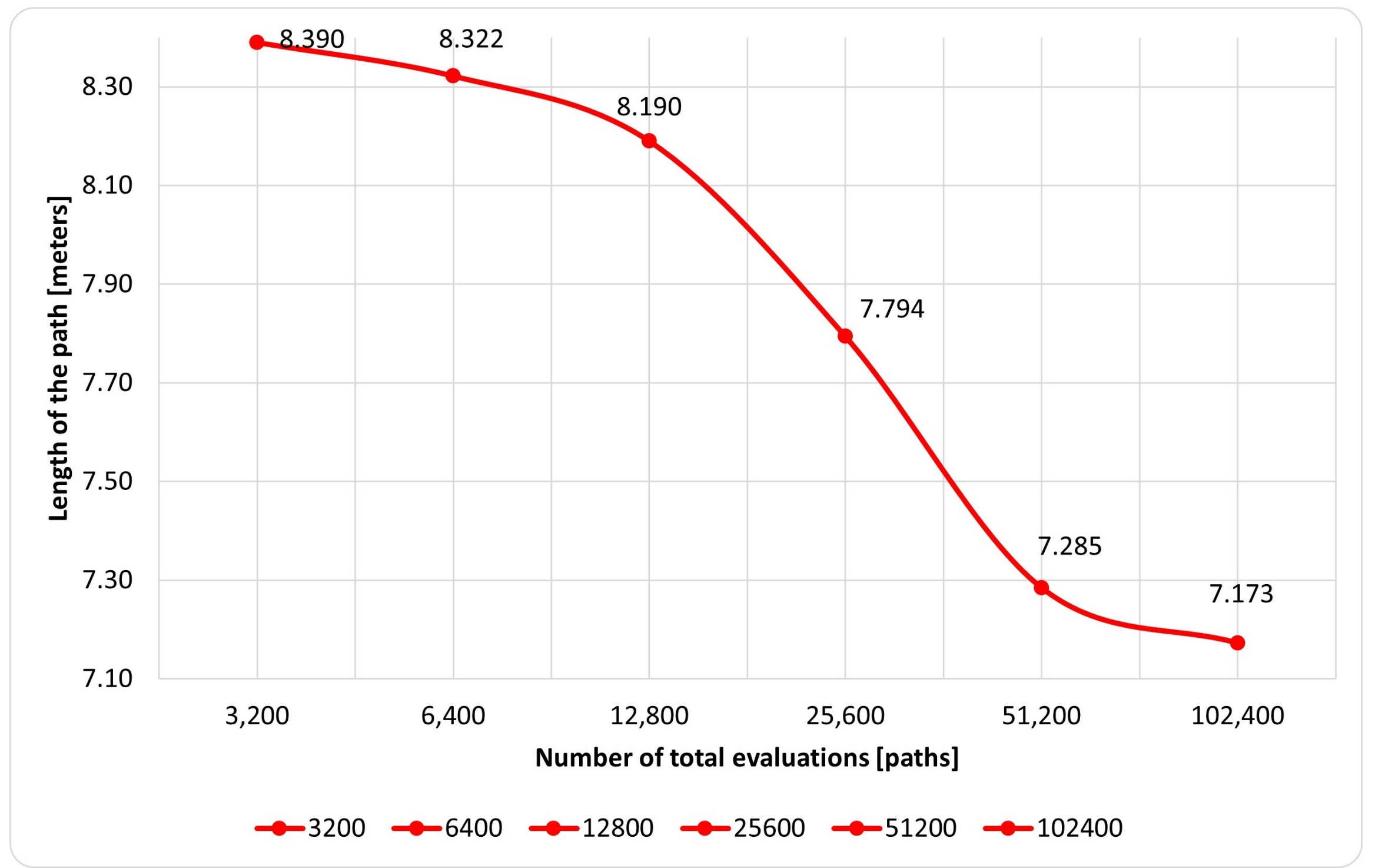
Path planning

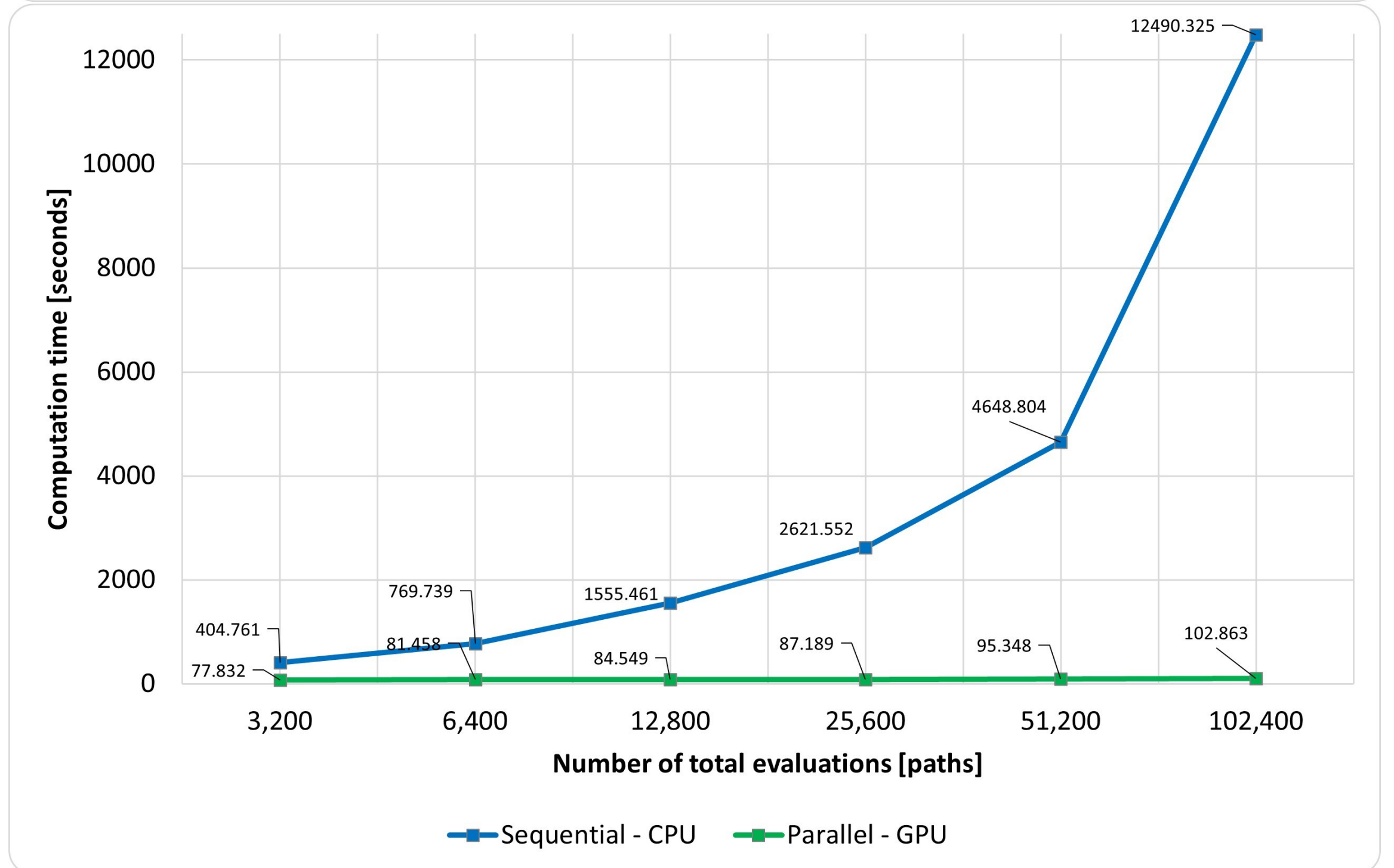
Performance Results

To evaluate the performance of the parallel path computation on GPU versus the sequential path computation on CPU, we carried out independently thirty tests for each number of total evaluations.

The convergence plot is presented in the right-upper figure. By increasing the number of individuals, we can observe how the MemPBPF algorithm improves the solution (a short path length is better).

In the right-lower figure, we can observe that the best speedup achieved for the parallel implementation on GPU, i.e., 12,490.325 divided by 102.863 is equal to **121.427** for 102,400 evaluations.





Conclusions

- In this work, we have presented the parallel path computation on GPU for mobile robot navigation using the MemPBPF programmed in C++/CUDA.
- The performance results show that the parallel path computation on GPU accelerates the process by a factor of **121x** for the biggest population tested in comparison with sequential path evaluation on CPU.
- We can see the advantage of using the parallel path computation on GPU, as well as we can see that the MemPBPF algorithm is highly scalable.
- The MemPBPF algorithm could be useful for many applications in mobile robots for global and local path planning, including industrial and domestic mobile robots, unmanned aerial vehicles, autonomous underwater vehicles, exploration vehicles, and self-driving cars.