Hindawi
Modelling and Simulation in Engineering
Volume 2022, Article ID 9388146, 14 pages
https://doi.org/10.1155/2022/9388146

Research Article

Pathfinding for Mobile Robot Navigation by Exerting the Quarter-Sweep Modified Accelerated Overrelaxation (QSMAOR) Iterative Approach via the Laplacian Operator

A'qilah A. Dahalan,[1,2] **Azali Saudi**,[3] **and Jumat Sulaiman**[4]

[1]*Centre for Defence Foundation Studies, National Defence University of Malaysia, Kuala Lumpur, Malaysia*
[2]*CONFIRM Centre for SMART Manufacturing, University of Limerick, Limerick, Ireland*
[3]*Faculty of Computing and Informatics, Universiti Malaysia Sabah, Kota Kinabalu, Sabah, Malaysia*
[4]*Faculty of Science and Natural Resources, Universiti Malaysia Sabah, Kota Kinabalu, Sabah, Malaysia*

Correspondence should be addressed to A'qilah A. Dahalan; a.qilah@upnm.edu.my

Received 8 September 2021; Revised 28 January 2022; Accepted 16 February 2022; Published 12 March 2022

Academic Editor: Michele Calì

Mobile robots are often in a situation where they need to find a bump-free path or navigation in their environment from any starting to a specific target point. Within this study, improving the navigation problem of a mobile robot iteratively by using a numerical method based on the potential field method is one of the main aims. This potential field will lean on the use of Laplace's equation to restrain the formation of a potential function across regions within the mobile robot configuration area. The present paper proposed a Quarter-Sweep Modified Accelerated Overrelaxation (QSMAOR) approach to improve the pathfinding of mobile robots in a given environment. The experiment shows that, by using a finite difference method, it is capable of producing an optimal path and creating a smooth path between the starting and target point. The results of the simulation also show that this numerical approach works more rapidly and provides a smoother/clearer direction than the previous study.

1. Introduction

The problem of pathfinding or navigation plays a vitally important role in autonomous mobile robots to ensure accuracy, safety, and efficiency. The basic idea to construct a genuinely autonomous mobile robot is that it must be able to design a route effectively and efficiently from the initial to the final configuration, without interfering with any static obstacles or other agents present between them. Competent algorithms to solve these kinds of problems have substantial practices in fields like computer animation [1, 2], industrial robotics [3–5], automated surveillance [6], or drug design [7]. It is therefore not shocking that studies conducted in this area have gradually increased over the last few years.

In their groundbreaking work, Connolly and Gruppen [8] have shown that harmonic functions possess many useful properties in robotic applications. Meanwhile, Khatib [9] used potential functions for robot pathfinding, in which each obstacle generates repellent force while the targets exert an enticing force. However, the major shortcoming of potential fields was suffering from the cause of local minima. In the meantime, Connolly et al. [10] and Akishita et al. [11] individually industrialized a global approach through Laplace's path planning equations towards the construction of a smooth collision-free path. These two studies show that the harmonic functions provide a swift method of generating paths for a robot configuration region and prevent the unprompted formation of local minima. Sasaki [12] indicated the use of computational methods to address the issue of path planning. It says that by simulating complicated problems on the maze, the current computational approach to motion planning worked very successfully. Dijkstra's algorithm was implemented by Karonava et al. [13] using labyrinth-based image processing for mobile robot track

Begin
Step 1: Mapping the configuration spaces (known grid space containing the goal position, obstacles, and walls, which are based upon [15]).
Step 2: Formulation and modelling the finite difference method of proposed iterative schemes.
Step 3: Algorithm of the proposed iterative schemes.
Step 4: Numerical solution.
Step 5: Evaluation and analysis.
End

Algorithm 1

planning. The algorithm determines the shortest path to the destination and demonstrates that an object can be moved through a large-scale labyrinth for the least amount of time, whereas Hachour's analysis [14] employed the autonomous mobile technique of navigation in the form of a grid map of an unidentified region using an intelligent/smart hybrid with motionless anonymous obstacles. The crucial aspect of this is the use of the best approach to perform biological genetic theory in conjunction with networks in the role of fuzzy reasoning and inference through the use of human intelligence to take the finest avoidance course in obtaining excellent protection against obstacle risk.

This paper seeks to simulate a point-robot pathfinding in the configuration space, using numerical potential functions based on the thermal transmission theory. This heat conduction model generates an environment that is not only free from local minima but also advantageous for the robot navigation control. Laplace's equation is used to model the problem as a heat conduction process. The aim is to obtain Laplace's equation solutions, also known as harmonic functions, to be used in simulating the temperature distribution in the configuration space for path generation purposes. Different approaches have been used to gain the harmonic functions, but, due to the obtainability of fast processing machines and their elegance and competence in problem solving, the most commonly used method is numerical approaches. In this study, several experiments were carried out to investigate the efficiency of the accelerated iterative method used to generate a mobile robot path for multisize environments. In essence, the overall process of the pathfinding construction phase in this study consists of the following steps are shown in Algorithm 1.

2. Materials and Methods

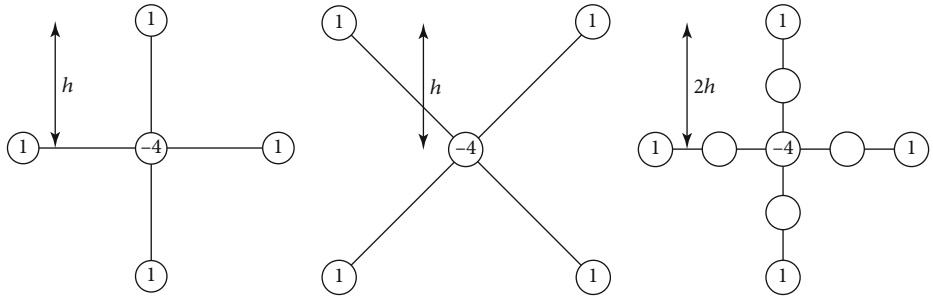
Instead of using the actual robot vehicle, we simulate the idea of moving the robot vehicle using a point that moves in a recognized space. The robot's pathfinding problem can be designated as a problem of steady-state heat conduction. In the analogy of heat conduction, the target is to be viewed as a sink heat-tugging in. Physical boundaries and obstacles are known as heat initiators that are set at constant temperatures. The temperature distribution evolves through the thermal conductivity process, and the thermal fluxes streaming into the sink fill the configuration space. This can be perceived as a means of communication between the target, robots, and obstacles. The field temperature distribution can then be used as a reference point for a mobile robot to travel from the departure point to the target point by monitoring the thermal flux from high-temperature sources to the lowest temperature point in the region. The temperature dispersal of the configuration region is figured by engaging the harmonic function to model the setup of the environment.

Mathematically, a harmonic function on a domain $\Omega \subset \mathbb{R}^n$ is a function that contains Laplace's equation, wherein x_i is the i-th Cartesian coordinates and n is the dimension. For the construction of the robot path, the domain Ω comprises the inner and outer boundary walls and altogether obstacles in the configuration space, starting points, and target point.

$$\nabla^2 \phi = \sum_{i=1}^{n} \frac{\partial^2 \phi}{\partial x_i^2} = 0. \tag{1}$$

The min–max principle holds for harmonic functions, implying that no local minima can arise spontaneously within the solution domain [8]. The Gauss Integral Theorem [16] states that there is a balance between inward and outward flow on the boundary of any volume within the solution domain (excluding obstacles/goals). As a result, there is always an escape path at any point or location. The gradient vector field of a harmonic function has zero curl, and the function itself follows the min–max principle. As a result, saddle points are the only critical points that can occur. A search in the area surrounding such a critical point can lead to the escape. Furthermore, any disruption of a path caused by such points produces a smooth path everywhere. The equation of Laplace can be efficiently solved with a numerical method. Jacobi and Gauss-Seidel are standard solutions to the problem, while in this paper, equation (1) has been solved using an accelerated iterative method for rapid computation.

The robot is defined in this model by a point in the configuration area. The design area is in the grid pattern. The value of the function for each node is then calculated by the numerical approach to fulfil equation (1) iteratively. The uppermost temperature is allocated to the starting point; however, the lowest is allocated to the target point. Various initial temperature values are given to the obstacles and wall boundaries. There is no requirement to allocate



(a) Standard/full-sweep (b) Half-sweep (c) Quarter-sweep iterations

Figure 1: Computational molecules of five-point approximation.

initial temperature values to the starting points. With the boundary conditions of Dirichlet, $\phi|\partial\Omega = c$, in which c is the constant, the solutions for Laplace's equation were examined. As soon as the potential values of the configuration area are gained, the smooth path is able to be created by outlining the temperature distribution via the steepest descent approach, where the algorithm trails the negative gradient from the beginning to the successive points with a lower temperature until/up to the lowest temperature target point.

2.1. Concept of Finite Difference Approximation. Consider the equation of 2-dimensional Laplace's as set out in equation (1) as

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 0. \tag{2}$$

The approximation of equation (2) can be streamlined over the five-point second-order standard finite difference method (FDM) as stated commonly as

$$U_{i-1,j} + U_{i+1,j} + U_{i,j-1} + U_{i,j+1} - 4U_{i,j} = 0. \tag{3}$$

Equation (3) is basically representing the full-sweep iteration, where the computation will consider all nodes in the mesh points. The $x - y$ axis can also be rotated clockwise to 45° to provide another form of approximation based on the cross-orientation operator, also known as half-sweep iteration [17]. The end result will have a rotated (skewed) approximation and only half of the total nodes are taken into account. The approximation of half-sweep concept can be written as

$$U_{i+1,j+1} + U_{i-1,j-1} + U_{i+1,j-1} + U_{i-1,j+1} - 4U_{i,j} = 0. \tag{4}$$

In addition, the following approximation can be obtained by considering the distance between two points and only one quarter of the total nodes, better known as quarter-sweep iteration [18], is considered:

$$U_{i-2,j} + U_{i+2,j} + U_{i,j-2} + U_{i,j+2} - 4U_{i,j} = 0. \tag{5}$$

To understand the concept of finite difference scheme, the computational molecules for the corresponding five-point approximation of full-, half-, and quarter-sweep iterations [18] are shown in Figure 1.

The illustration of the portion of the computational grid for these five-point approximations about point for all three concepts is detailed in Figure 2.

2.2. Modified Point Iterative Method. The basic concept of the red-black ordering technique is computing the iteration layer by layer. Thus, the formulation of full-, half-, and quarter-sweep cases will first be considered with the red nodes, followed by the computation of the black nodes in the mesh points of the configuration spaces. The computational grid for the modified variants, which involves the red-black ordering technique for full-, half-, and quarter-sweep, is presented in Figure 3.

2.3. Formulation of the Proposed Method. In the robotics literature, the iterative method of standard Gauss-Seidel (GS) [11] and Successive Overrelaxation (SOR) [19] was practiced to solve equation (1). The solution of Laplace's equation in this analysis is computed using an improved and faster numerical solver, namely, Accelerated Overrelaxation (AOR) iterative method and its variant, Modified Accelerated Overrelaxation (MAOR) iterative method.

2.4. Standard Accelerated Overrelaxation Iterative Method. From equations (3), (4), and (5), by adding the weighted parameter ω through SOR [20], the iterative scheme/formula for Full-Sweep SOR (FSSOR), Half-Sweep SOR (HSSOR), and Quarter-Sweep SOR (QSSOR), respectively, can be written as follows:

$$U_{i,j}^{(k+1)} = \frac{\omega}{4}\left[U_{i-1,j}^{(k+1)} + U_{i+1,j}^{(k)} + U_{i,j-1}^{(k+1)} + U_{i,j+1}^{(k)}\right] + (1-\omega)U_{i,j}^{(k)}, \tag{6}$$

$$U_{i,j}^{(k+1)} = \frac{\omega}{4}\left[U_{i-1,j-1}^{(k+1)} + U_{i+1,j-1}^{(k+1)} + U_{i-1,j+1}^{(k)} + U_{i+1,j+1}^{(k)}\right] + (1-\omega)U_{i,j}^{(k)}, \tag{7}$$

$$U_{i,j}^{(k+1)} = \frac{\omega}{4}\left[U_{i-2,j}^{(k+1)} + U_{i+2,j}^{(k)} + U_{i,j-2}^{(k+1)} + U_{i,j+2}^{(k)}\right] + (1-\omega)U_{i,j}^{(k)}. \tag{8}$$

In order to increase the convergence speed, the AOR

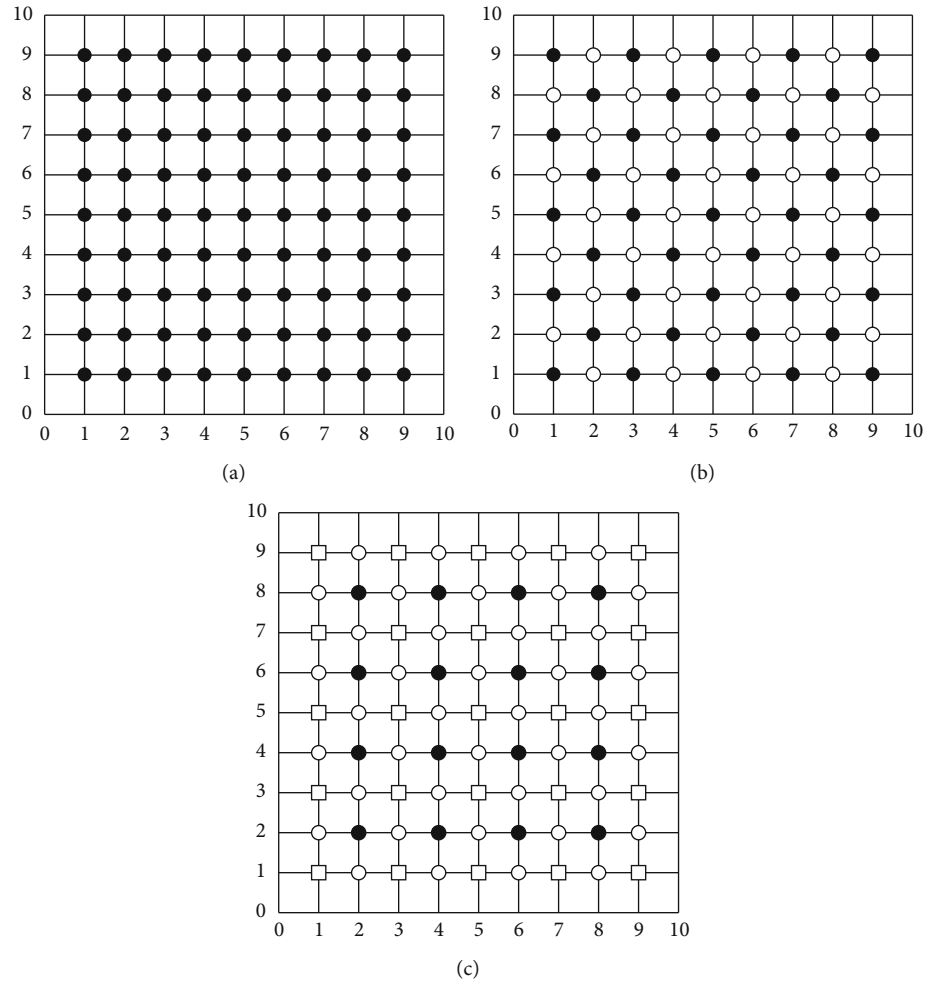


Figure 2: Computational nodes of the configuration space for (a) standard or full-sweep, (b) half-sweep, and (c) quarter-sweep iteration, respectively.

iterative scheme is implemented by dividing the weighted parameter ω from equation (6), (7), and (8) and adding another optimal relaxation parameter called r. The iterative scheme for the Full-Sweep AOR (FSAOR), Half-Sweep AOR (HSAOR), and Quarter-Sweep AOR (QSAOR) iterative method, respectively, is shown as follows:

$$U_{i,j}^{(k+1)} = \frac{r}{4}\left[U_{i-1,j}^{(k+1)} - U_{i-1,j}^{(k)} + U_{i,j-1}^{(k+1)} - U_{i,j-1}^{(k)}\right] + \frac{\omega}{4}\left[U_{i-1,j}^{(k)} + U_{i+1,j}^{(k)} + U_{i,j-1}^{(k)} + U_{i,j+1}^{(k)}\right] + (1-\omega)U_{i,j}^{(k)}, \tag{9}$$

$$U_{i,j}^{(k+1)} = \frac{r}{4}\left[U_{i-1,j-1}^{(k+1)} - U_{i-1,j-1}^{(k)} + U_{i+1,j-1}^{(k+1)} - U_{i+1,j-1}^{(k)}\right] + \frac{\omega}{4}\left[U_{i-1,j-1}^{(k)} + U_{i+1,j-1}^{(k)} + U_{i-1,j+1}^{(k)} + U_{i+1,j+1}^{(k)}\right] + (1-\omega)U_{i,j}^{(k)}, \tag{10}$$

$$U_{i,j}^{(k+1)} = \frac{r}{4}\left[U_{i-2,j}^{(k+1)} - U_{i-2,j}^{(k)} + U_{i,j-2}^{(k+1)} - U_{i,j-2}^{(k)}\right] + \frac{\omega}{4}\left[U_{i-2,j}^{(k)} + U_{i+2,j}^{(k)} + U_{i,j-2}^{(k)} + U_{i,j+2}^{(k)}\right] + (1-\omega)U_{i,j}^{(k)}. \tag{11}$$

2.5. Modified Accelerated Overrelaxation Iterative Method. An approach involving the application of the red-black ordering scheme strategy towards FSSOR, HSSOR, and QSSOR, namely, the Full-Sweep Modified SOR (FSMSOR), the Half-Sweep Modified SOR (HSMSOR), and the Quarter-Sweep Modified SOR (QSMSOR) methods, respectively, can eventually enhance the convergence speed. The formulation of FSMSOR can be expressed as

$$U_{i,j}^{(k+1)} = \frac{\omega}{4}\left[U_{i-1,j}^{(k)} + U_{i+1,j}^{(k)} + U_{i,j-1}^{(k)} + U_{i,j+1}^{(k)}\right] + (1-\omega)U_{i,j}^{(k)}, \tag{12a}$$

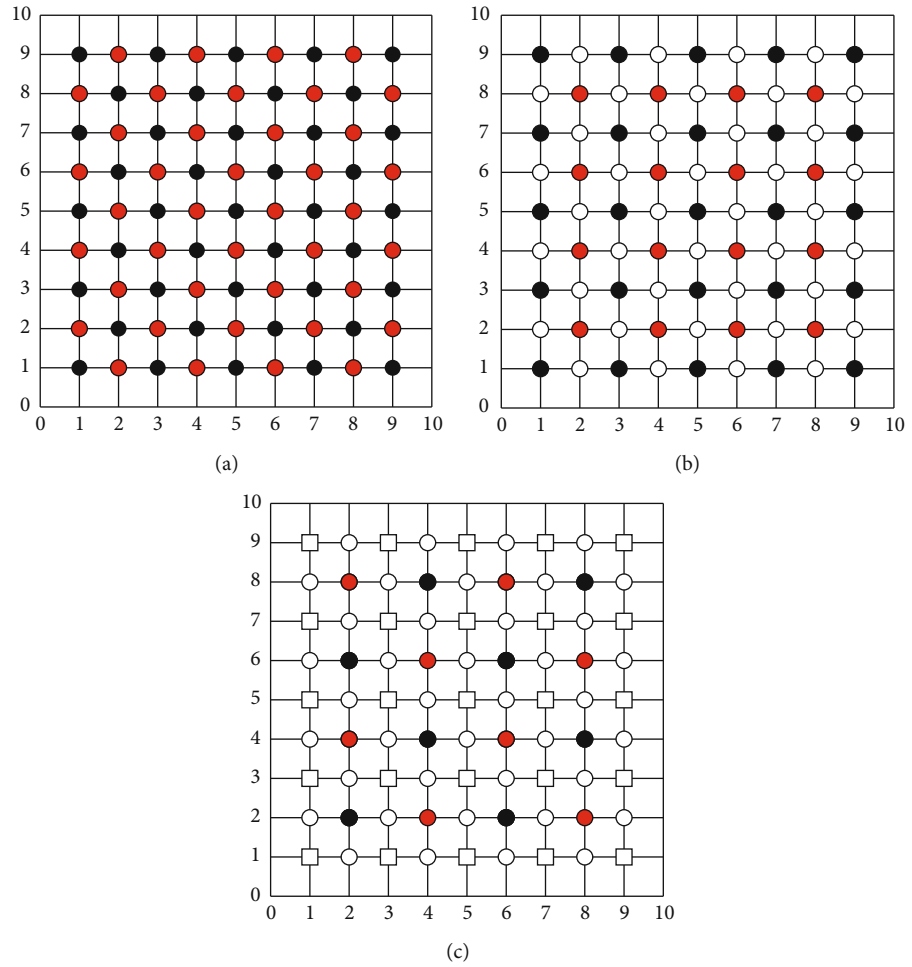


Figure 3: Computational nodes of the configuration space for (a) standard or full-sweep, (b) half-sweep, and (c) quarter-sweep iteration, respectively.

in red nodes, whereas in black nodes, it can be expressed as

$$U_{i,j}^{(k+1)} = \frac{\omega'}{4}\left[U_{i-1,j}^{(k+1)} + U_{i+1,j}^{(k+1)} + U_{i,j-1}^{(k+1)} + U_{i,j+1}^{(k+1)}\right] + \left(1-\omega'\right)U_{i,j}^{(k)}. \tag{12b}$$

Next, the formulation of HSMOR can be written in red nodes as

$$U_{i,j}^{(k+1)} = \frac{\omega}{4}\left[U_{i-1,j-1}^{(k)} + U_{i+1,j-1}^{(k)} + U_{i-1,j+1}^{(k)} + U_{i+1,j+1}^{(k)}\right] + (1-\omega)U_{i,j}^{(k)}, \tag{13a}$$

while in black nodes, it can be written as

$$U_{i,j}^{(k+1)} = \frac{\omega'}{4}\left[U_{i-1,j-1}^{(k+1)} + U_{i+1,j-1}^{(k+1)} + U_{i-1,j+1}^{(k+1)} + U_{i+1,j+1}^{(k+1)}\right] + \left(1-\omega'\right)U_{i,j}^{(k)}. \tag{13b}$$

The formulation of the QSMSOR method can be written

Table 1: Preliminary results of weighted parameters.

Weighted parameters	Methods	Iteration number	Execution time (in second)
$\omega = 1.86$	FSSOR	1728	8.13
	HSSOR	837	2.39
	QSSOR	351	0.39
$\omega = 1.86$	FSAOR	1591	8.61
$r = 1.87$	HSAOR	759	1.72
	QSAOR	348	0.56
$\omega = 1.86$	FSMSOR	1583	6.72
$\omega' = 1.88$	HSMSOR	747	2.13
	QSMSOR	374	0.34
$\omega = 1.86$	FSMAOR	1524	7.44
$r = 1.87$	HSMAOR	708	2.19
$\omega' = 1.85$	QSMAOR	264	0.39

1 Setup the configuration space with specified start and goal position.
2 Initialising starting point U, $\varepsilon \longleftarrow 10^{-15}$, $itera\ tion \longleftarrow 0$.
3 For all non-occupied red node points using Equation (17a), calculate
$U_{i,j}^{(k+1)} \longleftarrow (\omega/4)[U_{i-2,j}^{(k)} + U_{i+2,j}^{(k)} + U_{i,j-2}^{(k)} + U_{i,j+2}\ (k)] + (1-\omega)U_{i,j}^{(k)}$.
4 For all non-occupied black node points using Equation (17b), calculate
$U_{i,j}^{(k+1)} \longleftarrow (r/4)[U_{i-2,j}^{(k+1)} - U_{i-2,j}^{(k)} + U_{i,j-2}^{(k+1)} - U_{i,j-2}^{(\ k)}] + (r/4)[U_{i+2,j}^{(k+1)} - U_{i+2,j}^{(k)} + U_{i,j+2}^{(k+1)} - U_{i,j+2}^{(k)}] + (\omega'/4)[U_{i-2,j}^{(k)} + U_{i+2,j}^{(k)} + U_{i,j-2}^{(k)} + U_{i,j+2}^{(k)}] + (1-\omega')U_{i,j}^{(k)}$.
5 Compute the remaining non-occupied node points of type □ via direct method by using equation
$U_{i,j}^{(k+1)} \longleftarrow 1/4[U_{i-1,j-1}^{(k+1)} + U_{i+1,j-1}^{(k)} + U_{i-1,j+1}^{(k+1)} + U_{i+1,j+1}^{(k)}]$,
and node points of type ○ using equation.
$U_{i,j}^{(k+1)} \longleftarrow 1/4[U_{i-1,j}{}^{(k+1)} + U_{i+1,j}^{(k)} + U_{i,j-1}^{(k+1)} + U_{i,j+1}^{(k)}]$.
6 Check the convergence test for $\varepsilon \longleftarrow 10^{-15}$. If yes, go to next step. Else back to step (3).
7 Execute GDS to generate path from start to target position.

Algorithm 2: QSMAOR method.

Table 2: Number of arithmetic operations per iteration for SOR and its modified variants, MSOR methods.

Methods	ADD/SUB	MUL/DIV
FSSOR/FSMSOR	$4(N-1)^2$	$2(N-1)^2$
HSSOR/HSMSOR	$4\left[(N/2)^2 + (N/2-1)^2\right]$	$2[(N/2)^2 + (N/2-1)^2]$
QSSOR/QSMSOR	$4(N/2-1)^2$	$2(N/2-1)^2$

Table 3: Number of arithmetic operations per iteration for AOR and its modified variants, MAOR methods.

Methods	ADD/SUB	MUL/DIV
FSAOR/FSMAOR	$8(N-1)^2$	$3(N-1)^2$
HSAOR/HSMAOR	$8[(N/2)^2 + (N/2-1)^2]$	$3\left[(N/2)^2 + (N/2-1)^2\right]$
QSAOR/QSMAOR	$8(N/2-1)^2$	$3(N/2-1)^2$

Table 4: Number of additional arithmetic operations for the remaining points by using direct methods for all proposed iterative methods.

Methods	ADD/SUB	MUL/DIV
Half-sweep cases	$3\left(N^2/2 - N\right)$	$N^2/2 - N$
Quarter-sweep cases	$3(3N^2/4 - N)$	$3N^2/4 - N$

as

$$U_{i,j}^{(k+1)} = \frac{\omega}{4}\left[U_{i-2,j}^{(k)} + U_{i+2,j}^{(k)} + U_{i,j-2}^{(k)} + U_{i,j+2}^{(k)}\right] + (1-\omega)U_{i,j}^{(k)}, \tag{14a}$$

in red nodes, whereas in black nodes, it can be written as

$$U_{i,j}^{(k+1)} = \frac{\omega'}{4}\left[U_{i-2,j}^{(k+1)} + U_{i+2,j}^{(k+1)} + U_{i,j-2}^{(k+1)} + U_{i,j+2}^{(k+1)}\right] + \left(1-\omega'\right)U_{i,j}^{(k)}. \tag{14b}$$

In the meantime, the formulation of AOR variants, from now on known as FSMAOR, HSMAOR, and QSMAOR methods, which are generalized from equations (9), (10), and (11), respectively, can be specified as follows:

$$U_{i,j}^{(k+1)} = \frac{\omega}{4}\left[U_{i-1,j}^{(k)} + U_{i+1,j}^{(k)} + U_{i,j-1}^{(k)} + U_{i,j+1}^{(k)}\right] + (1-\omega)U_{i,j}^{(k)}, \tag{15a}$$

in red nodes, and in black nodes, it can be specified as

$$\begin{aligned} U_{i,j}^{(k+1)} = {} & \frac{r}{4}\left[U_{i-1,j}^{(k+1)} - U_{i-1,j}^{(k)} + U_{i,j-1}^{(k+1)} - U_{i,j-1}^{(k)}\right] \\ & + \frac{r}{4}\left[U_{i+1,j}^{(k+1)} - U_{i+1,j}^{(k)} + U_{i,j+1}^{(k+1)} - U_{i,j+1}^{(k)}\right] \\ & + \frac{\omega'}{4}\left[U_{i-1,j}^{(k)} + U_{i+1,j}^{(k)} + U_{i,j-1}^{(k)} + U_{i,j+1}^{(k)}\right] \\ & + \left(1-\omega'\right)U_{i,j}^{(k)}, \end{aligned} \tag{15b}$$

for FSMAOR. The formulation of the HSMAOR method is given as

$$U_{i,j}^{(k+1)} = \frac{\omega}{4}\left[U_{i-1,j-1}^{(k)} + U_{i+1,j-1}^{(k)} + U_{i-1,j+1}^{(k)} + U_{i+1,j+1}^{(k)}\right] + (1-\omega)U_{i,j}^{(k)}, \tag{16a}$$

Table 5: Performance of the methods considered in terms of the number of iterations.

N	Methods	300	600	900	1200	1500	1800
Case 1	FSSOR	1728	8117	17831	31346	47714	69063
	FSAOR	1591	7529	16594	28984	44292	64349
	FSMSOR	1583	7557	16697	29132	44800	63671
	FSMAOR	1524	7311	16069	28188	43396	61685
	HSSOR	837	4108	9086	15892	24286	34630
	HSAOR	759	3803	8420	14768	22569	32089
	HSMSOR	747	3812	8484	14845	22909	32583
	HSMAOR	708	3671	8190	14313	22163	31409
	QSSOR	351	2078	4632	8113	12483	17701
	QSAOR	348	1913	4280	7508	11567	16436
	QSMSOR	254	1910	4308	7571	11726	16650
	QSMAOR	264	1836	4161	7312	11303	16084
Case 2	FSSOR	2228	8776	19254	33558	51621	73346
	FSAOR	2006	7973	17538	30573	37595	40082
	FSMSOR	2097	8323	18307	31931	49131	69822
	FSMAOR	1872	7542	16617	28982	35351	37356
	HSSOR	1071	4438	9813	17149	26417	37562
	HSAOR	944	4023	8924	15614	19216	20483
	HSMSOR	988	4198	9314	16293	25116	35744
	HSMAOR	855	3787	8435	14782	18023	19014
	QSSOR	452	2229	5014	8771	13548	19254
	QSAOR	430	2007	4542	7976	9827	10433
	QSMSOR	363	2097	4747	8326	12874	18304
	QSMAOR	349	1876	4285	7537	9189	9634
Case 3	FSSOR	3624	14644	33004	57484	88366	125567
	FSAOR	3236	13165	29680	51738	79540	112979
	FSMSOR	3402	13814	31194	54363	83604	131946
	FSMAOR	3023	12395	28037	48890	75154	106841
	HSSOR	1780	7445	16856	29418	45264	64339
	HSAOR	1568	6681	15149	26456	40710	57897
	HSMSOR	1659	7006	15912	27802	42801	60851
	HSMAOR	1448	6271	14284	24970	38456	54714
	QSSOR	828	3769	8624	15061	23211	32999
	QSAOR	698	3366	7740	13545	20886	29680
	QSMSOR	754	3535	8122	14216	21933	31203
	QSMAOR	605	3142	7272	12757	19697	28031
Case 4	FSSOR	2507	9868	21654	37762	58054	82524
	FSAOR	2288	9025	19840	34601	53199	75634
	FSMSOR	2395	9411	20667	36037	55428	78781
	FSMAOR	2169	8623	18949	33056	50864	72308
	HSSOR	1212	5000	11036	19288	29683	42245
	HSAOR	1097	4555	10098	17670	27202	38693
	HSMSOR	1155	4769	10526	18412	28343	40319
	HSMAOR	1028	4351	9643	16877	26007	37012
	QSSOR	555	2502	5638	9873	15208	21647
	QSAOR	467	2287	5148	9030	13944	19850
	QSMSOR	496	2391	5364	9414	14526	20666
	QSMAOR	388	2174	4901	8619	13311	18950

in red nodes, and in black nodes, it is given as

$$\begin{aligned} U_{i,j}^{(k+1)} &= \frac{r}{4}\left[U_{i-1,j+1}^{(k+1)} - U_{i-1,j+1}^{(k)} + U_{i-1,j-1}^{(k+1)} - U_{i-1,j-1}^{(k)}\right] \\ &\quad + \frac{r}{4}\left[U_{i+1,j+1}^{(k+1)} - U_{i+1,j+1}^{(k)} + U_{i+1,j-1}^{(k+1)} - U_{i+1,j-1}^{(k)}\right] \\ &\quad + \frac{\omega'}{4}\left[U_{i-1,j-1}^{(k)} + U_{i+1,j-1}^{(k)} + U_{i-1,j+1}^{(k)} + U_{i+1,j+1}^{(k)}\right] \\ &\quad + \left(1-\omega'\right)U_{i,j}^{(k)}. \end{aligned} \tag{16b}$$

Finally, the formulation of the QSMAOR method can be stated as

$$U_{i,j}^{(k+1)} = \frac{\omega}{4}\left[U_{i-2,j}^{(k)} + U_{i+2,j}^{(k)} + U_{i,j-2}^{(k)} + U_{i,j+2}^{(k)}\right] + (1-\omega)U_{i,j}^{(k)}, \tag{17a}$$

in red nodes, and in black nodes, it can be stated as

$$\begin{aligned} U_{i,j}^{(k+1)} &= \frac{r}{4}\left[U_{i-2,j}^{(k+1)} - U_{i-2,j}^{(k)} + U_{i,j-2}^{(k+1)} - U_{i,j-2}^{(k)}\right] \\ &\quad + \frac{r}{4}\left[U_{i+2,j}^{(k+1)} - U_{i+2,j}^{(k)} + U_{i,j+2}^{(k+1)} - U_{i,j+2}^{(k)}\right] \\ &\quad + \frac{\omega'}{4}\left[U_{i-2,j}^{(k)} + U_{i+2,j}^{(k)} + U_{i,j-2}^{(k)} + U_{i,j+2}^{(k)}\right] + \left(1-\omega'\right)U_{i,j}^{(k)}. \end{aligned} \tag{17b}$$

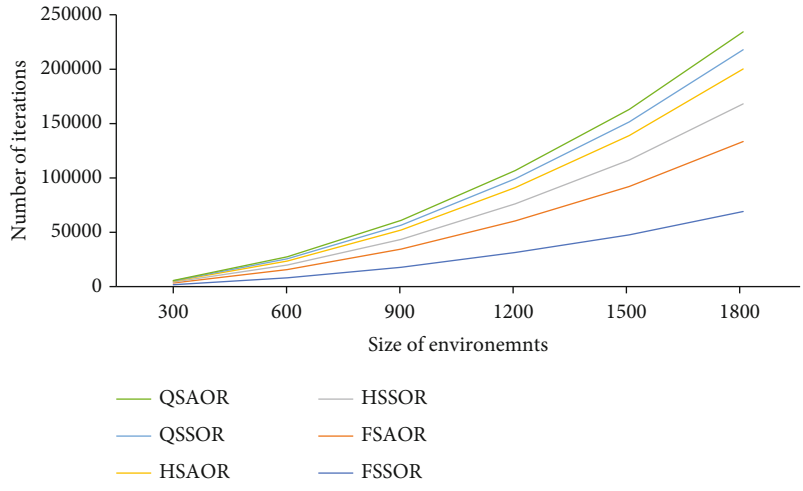
For all formulations of AOR variants, the r, ω, and ω' are indicated as the optimum relaxation parameters. The uncertain optimum values of r, ω, and ω' did not limit the minimum number of iterations. Hadjidimos [21] defined that the value of r and ω' is typically selected to be close to the value ω of the corresponding SOR, where $1 \le \omega < 2$.

In this study, all these weighted parameters are determined by the process of sensitivity analysis, otherwise called parameter tuning, by means of trial and error. In order to find the optimal value, the weighted parameter values are different for each of the half- and quarter-sweep cases, as some values do not converge in certain cases. Moreover, the effect of complexity on finding parameter value to overall computation does not alter since the value of each parameter is set before the execution/computation. It will indeed change if the ranges of parameter values are set in the algorithm computation. Table 1 shows some of the preliminary results (for environment size 300×300) with optimal values used throughout the experiments.

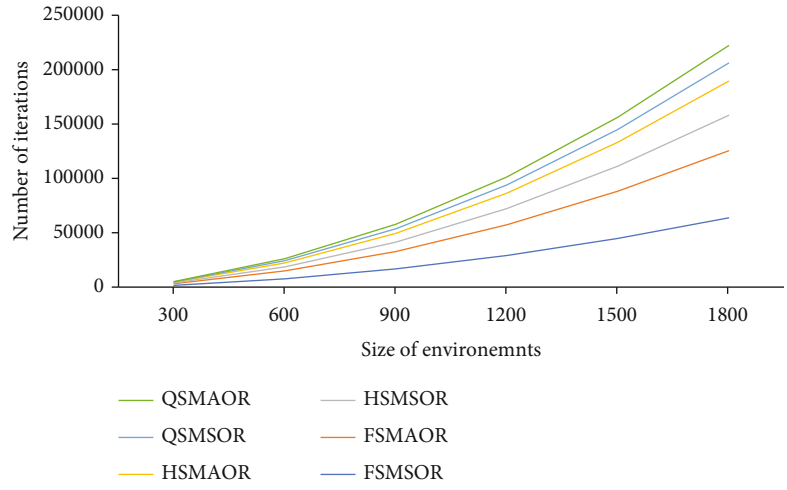
Thus, the implementation of the QSMAOR scheme based on equations (17a) and (17b) for solving 2-

Table 6: Performance of the methods considered in terms of the time of execution (in seconds).

N	Methods	300	600	900	1200	1500	1800
Case 1	FSSOR	8.13	227.95	1134.25	3728.92	8686.08	17147.79
	FSAOR	8.61	230.17	1148.87	3692.74	8660.15	16968.09
	FSMSOR	6.72	240.99	1227.39	4082.35	9588.90	19327.49
	FSMAOR	7.44	247.99	1295.65	4330.56	10208.13	19265.66
	HSSOR	2.39	81.24	404.15	1375.27	3255.63	6624.13
	HSAOR	1.72	73.76	369.91	1247.65	2990.33	6046.26
	HSMSOR	2.13	73.03	373.18	1295.14	3113.40	6234.52
	HSMAOR	2.19	81.73	431.96	1471.83	3580.86	6892.12
	QSSOR	0.39	14.99	81.55	293.92	718.55	1445.17
	QSAOR	0.56	15.83	84.47	292.46	709.11	1463.86
	QSMSOR	0.27	16.61	90.51	326.84	787.37	1571.21
	QSMAOR	0.39	18.18	96.43	349.09	848.99	1669.26
Case 2	FSSOR	10.69	251.72	1270.23	4077.22	8871.90	16374.18
	FSAOR	10.27	248.24	1226.66	3976.33	6346.31	9640.11
	FSMSOR	9.36	269.68	1355.34	4329.63	8601.34	18314.85
	FSMAOR	9.30	267.18	1360.64	4342.87	6977.82	11888.25
	HSSOR	2.95	86.77	445.70	1423.27	3356.36	6214.61
	HSAOR	2.75	76.79	403.25	1263.63	2200.89	3323.99
	HSMSOR	2.64	80.20	414.86	1338.13	2799.64	6024.91
	HSMAOR	2.34	86.67	450.83	1409.04	2397.26	3937.09
	QSSOR	0.64	16.69	90.03	313.44	738.74	1404.43
	QSAOR	0.56	16.68	89.98	314.14	547.43	829.91
	QSMSOR	0.50	18.90	99.75	341.75	751.32	1556.51
	QSMAOR	0.42	18.21	101.92	343.47	597.57	897.37
Case 3	FSSOR	16.22	427.27	2190.45	7432.68	14928.51	34024.02
	FSAOR	18.66	418.45	2073.25	7254.02	14726.63	34297.24
	FSMSOR	14.40	462.03	2361.08	7957.70	16036.74	41566.73
	FSMAOR	15.35	450.60	2420.88	7800.25	16291.28	38068.00
	HSSOR	5.16	154.79	783.72	2634.52	5571.93	12912.67
	HSAOR	4.80	137.18	721.94	2300.84	5044.60	11679.58
	HSMSOR	4.08	140.88	739.14	2443.07	5239.63	12545.53
	HSMAOR	4.66	151.04	803.10	2573.74	5642.28	12799.79
	QSSOR	0.92	30.04	166.12	567.28	1275.10	2803.43
	QSAOR	1.08	29.24	161.76	570.33	1255.65	2785.10
	QSMSOR	0.92	34.09	184.67	629.30	1367.24	3134.97
	QSMAOR	0.81	33.22	188.82	608.37	1369.75	3095.58
Case 4	FSSOR	11.02	281.85	1441.47	4853.57	10789.65	21088.78
	FSAOR	12.52	281.78	1423.54	4743.21	9182.12	20942.50
	FSMSOR	9.78	309.74	1576.44	5150.07	11768.10	23502.56
	FSMAOR	9.83	309.98	1581.29	5163.24	10231.80	23211.15
	HSSOR	3.58	102.16	510.22	1686.65	3969.00	8030.14
	HSAOR	3.08	92.44	471.17	1511.93	3173.82	7095.03
	HSMSOR	3.28	94.51	482.17	1578.80	3765.84	7712.18
	HSMAOR	3.27	100.31	533.66	1686.72	3559.83	7980.54
	QSSOR	0.75	19.85	106.87	369.38	883.07	1736.46
	QSAOR	0.73	19.97	108.78	364.51	790.14	1714.40
	QSMSOR	0.62	21.99	116.25	406.91	860.32	1925.26
	QSMAOR	0.45	22.54	123.71	402.04	876.54	1939.07



(a) Standard technique



(b) Modified technique

Figure 4: Number of iterations in varying environments.

dimensional Laplace's problem as expressed in equation (2) can be stated in Algorithm 2.

2.6. Computational Complexity. This section discusses the computational complexity analysis of all iterative techniques considered in this study. Each arithmetic operation (addition and multiplication) is expected to take one unit of computational time. The path tracing procedure of the GDS-DT algorithm and the arithmetic operations used in the convergence test are excluded. Tables 2 and 3 show the total number of arithmetic operations required by each of the approaches examined. Additional arithmetic operations are performed for the HS and QS algorithms to determine the remaining points after convergence using direct methods, as shown in Table 4.

Theoretically, as the computational complexity of the algorithm drops, the number of iterations decreases, reducing CPU time. Despite having more arithmetic operations than SOR method families, AOR method families converge faster because of additional accelerated parameter [22]. Meanwhile, the remaining points will be ignored from the total computation of computational complexity because they will have no significant impact because the computation of the remaining points is calculated in one iteration only.

3. Results

The experiments were carried out on the AMD A10 machine that was equipped with 8 GB memory running at 2.50 GHz. The iteration process to evaluate the temperature values numerically at all points continues up until the stopping criterion is encountered. If the temperature values are no longer showing changes, the loop would be terminated where the difference in the measurement values was extremely small, i.e., 10^{-15}. This high precision was needed in the solution to prevent the creation of a flat area, otherwise known as saddle points, from failing the generation of paths.

Tables 5 and 6 depict the number of iterations and the time of execution in seconds, respectively, required for all numerical techniques compared in the experiment to measure all temperature values in the region.

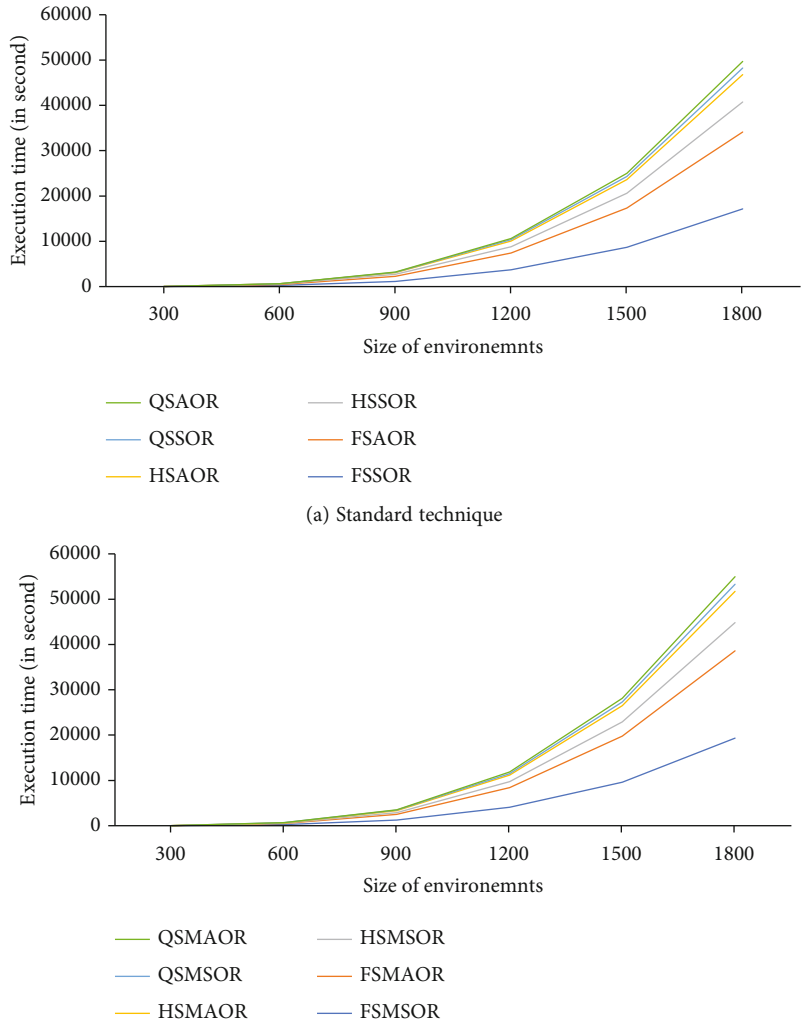


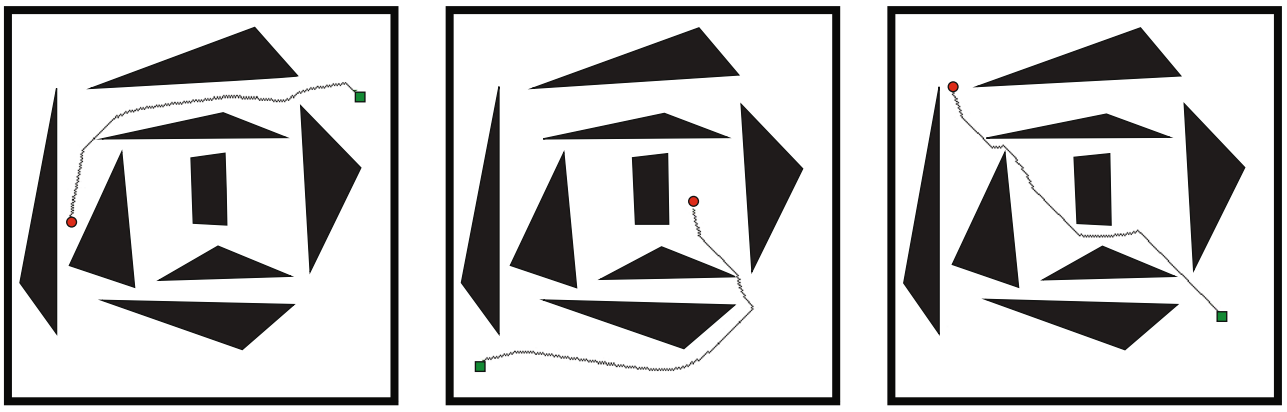
Figure 5: Performance (in seconds) in varying environments.

The graphs in Figures 4 (the number of iterations) and 5 (the execution time) indicate the output of the proposed methods based on Tables 5 and 6. It can be deduced from both figures that the higher the number of iterations, the longer each execution takes. By referring to both graphs, it can be seen that the QSMAOR and QSAOR have outperformed their corresponding suggested methods, in terms of either an iteration count or CPU time. This idea can also be clearly seen in Tables 5 and 6. As we can see from the table of results, the graphs for the number of iterations as well as execution time gave the same pattern. Apparently, the QSMAOR iterative scheme provides high efficiency in terms of iteration number compared to other proposed approaches, although the time required for modified families varied slightly from that for conventional methods, depending on the environments.

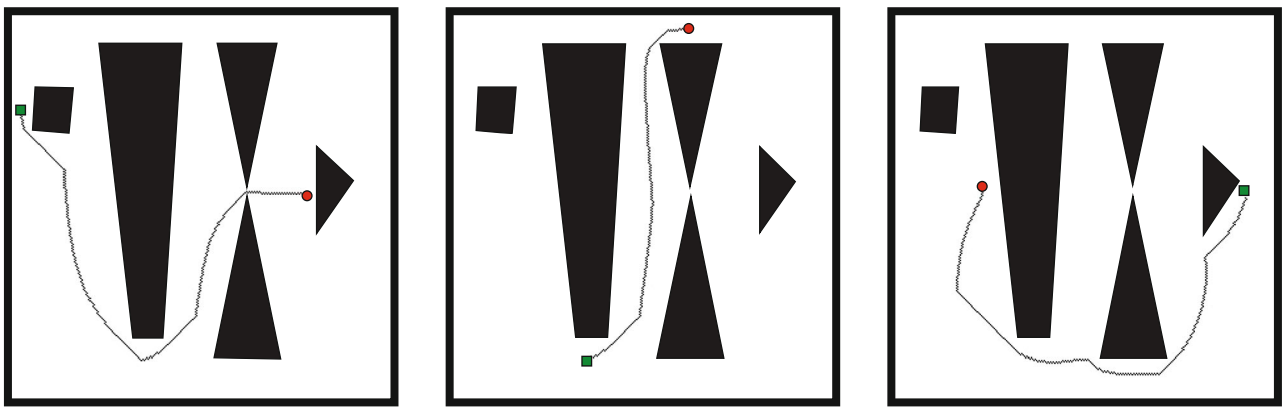
4. Discussion

In this analysis, the environment setup involves four different sizes: 300×300, 600×600, 900×900, and 1200×1200. From Figure 6, the target point was addressed at the fixed and lowest temperature values, while no particular temperature values are assigned to all three starting points. Various numbers of obstacles with different shapes have been placed in the environment. The Dirichlet boundary condition was implemented in the initial setting, in which the obstacles and walls were installed at high temperature values. In the region, every point was fixed to zero temperature value and then again for the target point at the lowest temperature values.

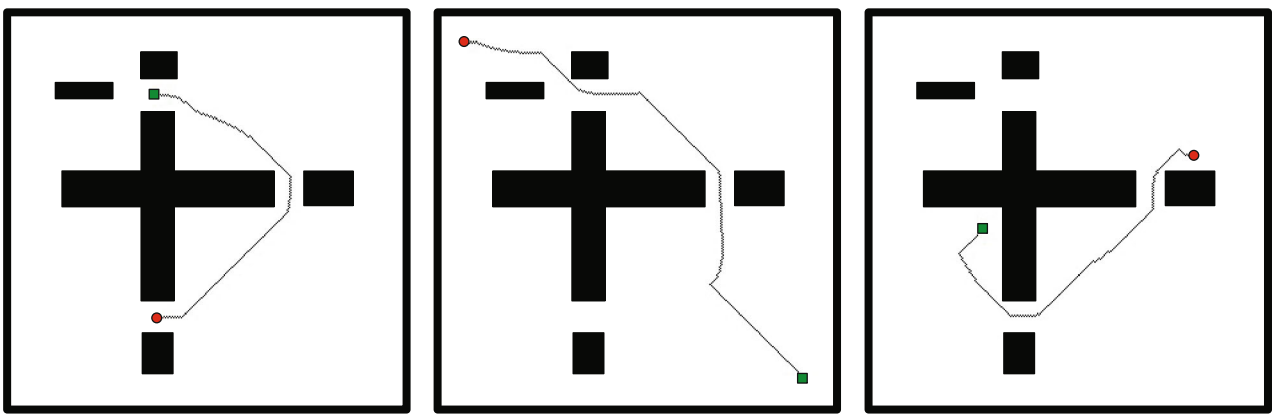
The trail was created by performing the steepest descent search from the start to the target point, once the temperature values have been obtained. The path generation process was very fast, in which the algorithm straightforwardly picks the lowest temperature value of its adjacent points from the current point. The cycle continues until the target point is reached. Figure 6 shows that in an obstacle environment, the paths were positively created on the basis of the temperature distribution outline gained by numerical computation. Each starting point (square/green point) has been successfully completed at the specified target point (round/red



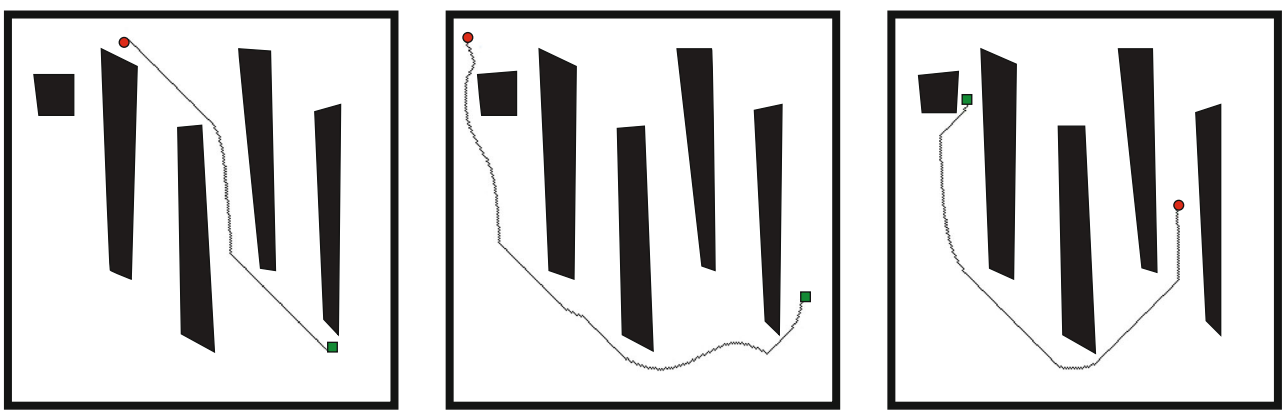
(a) Case 1



(b) Case 2



(c) Case 3



(d) Case 4

Figure 6: Creation of paths from different starting points (square/green point) and target positions (round/red point) for different environments.

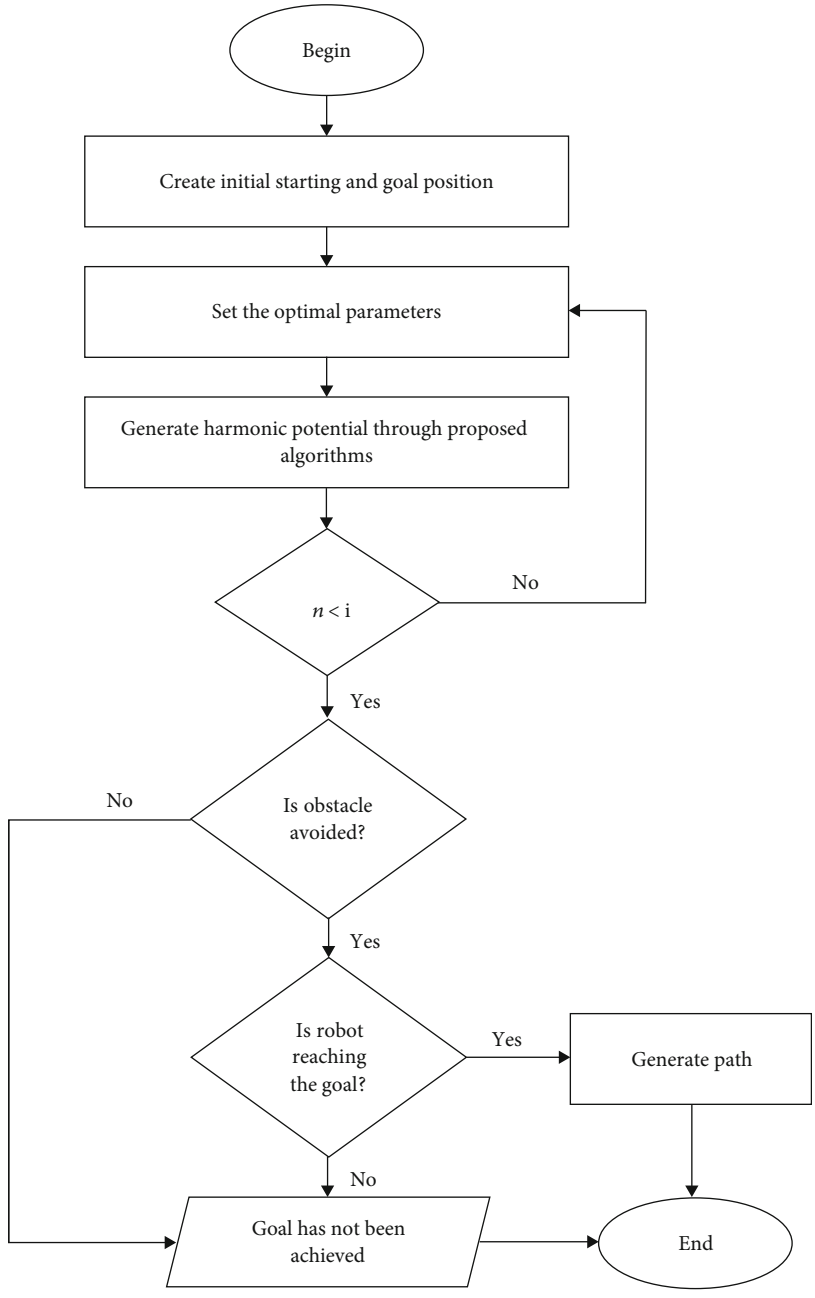


Figure 7: Flow diagram of the pathfinding technique.

point), escaping the many obstacles set in place. The flow diagram of the pathfinding technique of this study is shown in Figure 7.

5. Conclusions

The experiments show that the solution to the robot pathfinding problem can be solved using numerical approaches due to the availability of advanced algorithms and fast machines today. The QSMAOR iterative method has been shown to be very capable in obtaining the solution faster than standard SOR and AOR methods as shown in the results. In terms of actual computational time, the QSMSOR and QSMAOR methods provide the best performance. An increase in the number of obstacles in the environment does not affect the performance of the proposed algorithms, as the computing actually gets faster as the areas occupied by the obstacles are ignored during the calculation. The proposed algorithms provide a safe and smooth path from the start to the target regardless of obstacle shape and position since the generated path tends to move away from the obstacles. In the future work, the approach can be extended by employing a more advanced numerical technique that utilizes block iteration [23, 24], further speeding up the convergence rate of the iteration process.

Furthermore, this outcome has the potential to be utilized in real-world applications. For example, Amer et al. [25] discussed the development of an adaptive path tracking

controller with a knowledge-based supervisory algorithm for an autonomous heavy vehicle. They presented an adaptive algorithm that would determine the best controller parameter automatically based on the manoeuvring and vehicle conditions. The proposed adaptive controller worked effectively in guiding the vehicle along all trajectories, while Oskar et al. [26] developed a simulation model that allows researchers to investigate in-pipe machine features in terms of dependence on body size, bristle geometry, actuator, pipe deviations, etc. The experimental model was used to validate the simulation model, which yielded promising results. Following that, Virgala et al. [27] examined a snake robot motion in narrow spaces (i.e., a pipe) and developed a unique experimental snake robot with one revolute and one linear joint on each module and the capacity to execute in planar motion. The study presents a novel method for anchoring snake robot modules in a pipe using symmetrical curves during locomotion. Moreover, it is of high importance from a practical standpoint that the proposed new methods be able to cope with dynamic environments. Therefore, the findings of this study can also be extended/investigated in the future to include performing in a dynamic environment with moving obstacles. For example, consider [28], in which electrostatic potential field theory is applied to solve a robot's path planning problem. To make the decision, all obstacles' features are integrated into a scalar potential field. Practicing a scalar potential field simplifies mathematical computations and is thus realistically feasible to be applied in both static and dynamic contexts. It can be seen that the concept of potential field utilized for path planning in [28] has some similar key concepts with the current work.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This research was financially supported by National Defence University of Malaysia.

References

[1] Y. Li, W. Du, Z. Tang, and B. Zhang, "Computer animation of robot motion with collision dree path planning," in *Creating and Animating the Virtual World*, Computer Animation Series, N. M. Thalmann and D. Thalmann, Eds., pp. 149–161, Springer-Verlag, Tokyo, Japan, 1992.

[2] X. Sheng, "Motion planning for computer animation and virtual reality applications," in *CA '95: Proceedings of the Computer Animation; IEEE Computer Society*, NW Washington, DC, United States, 1992.

[3] L. Lars, J. Kim, K. Michael, and S. Alfons, "Automatic path planning of industrial robots comparing sampling-based and computational intelligence methods," *Procedia Manufacturing*, vol. 11, pp. 241–248, 2017.

[4] L. Lars, S. Alfons, J. Kim, and K. Michael, "Path planning of cooperating industrial robots using evolutionary algorithms," *Procedia Manufacturing*, vol. 17, pp. 286–293, 2018.

[5] H. Zhang, Y. Wang, J. Zheng, and J. Yu, "Path planning of industrial robot based on improved RRT algorithm in complex environments," *IEEE Access*, vol. 6, pp. 53296–53306, 2018.

[6] P. Federico, A. Mario, G. Derlis, G. R. Daniel, and T. Sergio, "A comparison of local path planning techniques of autonomous surface vehicles for monitoring applications: the Ypacarai Lake Case-study," *Sensors*, vol. 20, no. 5, article 1488, 2020.

[7] J. Byska, I. Kolingerova, B. Kozlikova, and J. Sochor, "Path-planning algorithm for transportation of molecules through protein tunnel bottleneck," *Proc. 31st Spring Conf. Comp.*, vol. - Graphics-SCCG'15, pp. 81–88, 2015.

[8] C. I. Connolly and R. Gruppen, "On the applications of harmonic functions to robotics," *Journal of Robotic Systems*, vol. 10, no. 7, pp. 931–946, 1993.

[9] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[10] C. I. Connolly, J. B. Burns, and R. Weiss, "Path planning using Laplace's equation," *Proc. IEEE Int. Conf. Robotics Automat.*, pp. 2102–2106, 1990.

[11] S. Akishita, T. Hisanobu, and S. Kawamura, "Fast path planning available for moving obstacle avoidance by use of Laplace potential," *Proc. IEEE Int. Conf. Intelligent Robots Syst.*, pp. 673–678, 1993.

[12] S. Sasaki, "A practical computational technique for mobile robot navigation," *Proc. IEEE Int. Conf. Control Appl.*, pp. 1323–1327, 1998.

[13] M. Karonava, D. Zhelyazkov, M. Todorova, I. Penev, V. Nikolov, and V. Petkov, "Path planning algorithm for mobile robot," *Recent Res. Appl. Comp. Sci.*, pp. 26–29, 2015.

[14] O. Hachour, "Path planning of autonomous mobile robot," *Int. J. Syst. Appl. Eng. Development*, vol. 4, no. 2, pp. 178–190, 2008.

[15] F. C. Shiang, *Collision-Free Path Planning; Retrospective Theses and Dissertations*, Iowa State University, 1997.

[16] R. Courant and D. Hilbert, *Methods of Mathematical Physics, Vol II*, John Wiley and Sons, New York, 1989.

[17] W. S. Yousif and D. J. Evans, "Explicit group over-relaxation methods for solving elliptic partial differential equations," *Mathematics and Computers in Simulation*, vol. 28, no. 6, pp. 453–466, 1986.

[18] M. Othman and A. R. Abdullah, "An efficient four points modified explicit group Poisson solver," *International Journal of Computer Mathematics*, vol. 76, no. 2, pp. 203–217, 2000.

[19] A. Saudi, J. Sulaiman, and M. H. A. Hijazi, "Fast robot path planning with Laplacian behaviour-based control via four-point explicit decoupled group SOR," *Res. J. Appl. Sci.*, vol. 9, no. 6, pp. 354–360, 2014.

[20] D. M. Young, *Iterative Methods for Solving Partial Difference Equations of Elliptic Type*, Harvard University, United States, 1950, PhD Thesis.

[21] A. Hadjidimos, "Accelerated overrelaxation method," *Mathematics of Computation*, vol. 32, no. 141, pp. 149–157, 1978.

[22] J. Kuang and J. Ji, "A survey of AOR and TOR methods," *Journal of Computational and Applied Mathematics*, vol. 24, no. 1-2, pp. 3–12, 1988.

[23] A. A. Dahalan, A. Saudi, J. Sulaiman, and W. R. W. Din, "Numerical evaluation of mobile robot navigation in static indoor environment via EGAOR iteration," *Journal of Physics Conference Series*, vol. 890, article 012064, 2017.

[24] A. Saudi, "Robot path planning with EGSOR iterative method using Laplacian Behaviour-Based Control (LBBC)," 5th Int. Conf. Intell. Syst. Model. Simul., IEEE Comp. Soc., 2014.

[25] N. H. Amer, K. Hudha, H. Zamzuri et al., "Knowledge-based controller optimised with particle swarm optimisation for adaptive path tracking control of an autonomous heavy vehicle," *Automation and Control, IntechOpen*, vol. 72446, 2020.

[26] O. Oskar, O. Eva, K. Michal, K. Tatiana, B. Jan, and V. Ivan, "Miniature mobile bristled in-pipe machine," *International Journal of Advanced Robotic Systems*, vol. 11, p. 189, 2014.

[27] I. Virgala, M. Kelemen, P. Božek et al., "Investigation of snake robot locomotion possibilities in a pipe," *Symmetry*, vol. 12, no. 6, p. 939, 2020.

[28] F. Bayat, S. Najafinia, and M. Aliyari, "Mobile robots path planning: electrostatic potential field approach," *Expert Systems with Applications*, vol. 100, pp. 68–78, 2018.