Evaluating Crossover and Mutation Techniques for Maximizing Sensor Lifespan in Rechargeable Sensor Networks

Yan-Heng Yu
Department of Computer Science National Yang Ming Chiao Tung University
HsinChu City 300, Taiwan
yanhengyu0719@gmail.com

Yi-Hsuan Chen
Department of Computer Science National Yang Ming Chiao Tung University
HsinChu City 300, Taiwan
yihsuanchen29@gmail.com

Hung-Hsiang Chen
Department of Computer Science National Yang Ming Chiao Tung University
HsinChu City 300, Taiwan
wellmagg@gmail.com

Bing-Shu Wu
Department of Computer Science National Yang Ming Chiao Tung University
HsinChu City 300, Taiwan
benson890924@gmail.com

ABSTRACT

This paper explores the application of genetic algorithms (GA) for efficient charging scheduling in wireless rechargeable sensor networks (WRSNs). While existing solutions for on-demand charging face challenges in mobility energy expenditure and response time, we evaluate alternative crossover and mutation techniques to enhance network survival rates. Our project builds upon the foundational work by Dudyala and Dash, introducing three novel techniques and conducting comparative analyses against the original methodology. Simulation results reveal improvements in survival rates and energy efficiency, providing a pathway for sustainable WRSN operation.

CCS CONCEPTS

• **Computer systems organization** → **Sensor networks**; • **Hardware** → *Battery management*.

KEYWORDS

Genetic Algorithm, Rechargeable Sensor Networks, Charging Schedule, Crossover Techniques, Mutation Techniques

1 INTRODUCTION

Wireless sensor networks (WSNs) are the foundation of numerous IoT applications, including health monitoring, environmental detection, and military operations. However, their functionality is often limited by energy constraints, as sensors rely on batteries that can deplete over time. To address this, researchers have explored various energy management strategies, such as energy harvesting, mobile chargers (MCs), and load balancing techniques [1]. Among these, the use of rechargeable sensor networks (RSNs) supported by mobile chargers equipped with wireless energy transfer (WET) technology has emerged as a promising solution.

1.1 Background

Wireless rechargeable sensor networks (WRSNs) enable sensors to be recharged wirelessly, ensuring continuous operation even in remote or harsh environments. Mobile charging vehicles (MCs) play a key role by visiting sensor nodes to provide energy on demand. Despite their advantages, MC-based WET systems face challenges such as high energy expenditure during movement, long service times, and slow charging rates. These limitations can lead to scenarios where sensors run out of energy before the MC arrives, reducing the survival ratio of the network.

Existing charging strategies can be broadly classified into periodic and on-demand methods. While periodic charging relies on predetermined schedules, it often fails to accommodate the dynamic energy requirements of large-scale WRSNs. On-demand charging, in contrast, reacts to real-time sensor requests but faces optimization challenges, such as determining charging thresholds, scheduling routes, and balancing energy consumption. Recent studies have highlighted partial charging as a more efficient alternative to full charging, enabling better survival ratios by dynamically distributing energy resources.

1.2 Motivation

The charging scheduling problem for WRSNs has a large solution space, numerous constraints, and a complex and nonlinear objective: maximizing the survivor rate of charging requests under limited resources. This problem requires balancing competing factors, such as energy availability, charging priorities, and the time required to reach sensors, which makes finding an optimal solution computationally challenging.

Genetic algorithms (GAs) are particularly well suited for addressing such problems due to their adaptability, ability to explore large solution spaces, and capacity to handle non-linearity. GAs can efficiently generate good enough solutions through iterative optimization.

During the classes, we learned about various crossover and mutation operators used in GAs. This inspired us to investigate the performance of these operators in solving the charging scheduling problem. Our goal is to identify the most effective strategies for maximizing the survivor rate under varying conditions, thereby contributing valuable insights into the application of evolutionary algorithms to real-world optimization challenges.

1.3 Research Challenges and Contributions

The primary goal of this work is to optimize the charging schedule for MCs in WRSNs to maximize the survival ratio of sensors. Achieving this requires addressing several key challenges:

1.3.1 Research Challenges.

- **Problem Complexity:** This TSP-like problem is classified as NP-hard, with its solution space growing exponentially as the number of sensors and charging constraints increase.
- **Solution Quality:** Various crossover and mutation strategies can significantly influence the depth and breadth of solution space exploration, affecting both convergence speed and solution quality.
- **Resource Constraints:** Balancing limited charging resources against the energy demands of sensor nodes is critical for effective network operation.

1.3.2 Research Contributions.

- **Dynamic Energy Thresholds:** Determining when sensors should request charging based on their residual energy and distance to the service station.
- **Efficient Scheduling:** Prioritizing charging requests to minimize the number of dead nodes while considering MC energy constraints.

2 PROBLEM DEFINITION

2.1 Network Entities

The charging network consists of three key entities: rechargeable sensors, mobile chargers (MC), and power stations.

The first entity is a collection of rechargeable sensors. Each sensor has distinct attributes, including its geographical location, current battery level, consumption rate, and maximum battery capacity. During the system's operation, these sensors may require recharging based on their battery levels to maintain functionality.

The second entity is the mobile chargers, which are mobile charging units with several critical initial attributes. These include the maximum energy capacity, the energy expenditure rate per unit distance traveled, a fixed charging rate that determines the speed at which the MC can recharge sensors, and the initial position of the MC within the network.

The third entity consists of power stations, which are fixed locations within the network. These stations serve as recharge points where the MC can replenish its own battery.

2.2 Problem Statement

Given a set of sensor requests, find out the best crossover and mutation strategy to generate an effective charging schedule that minimizes the number of sensors that deplete their battery to zero during the MC's journey.

2.3 Objective Scenario

This study focuses on the Fully Charging Schedule (FCS), where the mobile charger (MC) remains at each sensor node until it is fully charged before proceeding to the next. Each sensor node is defined by its x and y coordinates, with the MC starting from the origin, visiting all sensor nodes requiring charging, and ultimately returning to the origin.

We evaluate four crossover and four mutation methods, with one crossover and one mutation method adopted from [1]. It is important to note that we refer to the method from [1] as the "Baseline" for simplicity, and this usage is slightly abuse, as our scenario involves a Fully Charging Schedule (FCS), whereas the original paper addresses a Partial Charging Schedule (PCS). By analyzing the performance of these methods, we aim to identify the most effective crossover and mutation strategies for the FCS problem, offering valuable insights for future research on intelligent charging schedules.

3 METHODOLOGY

We adopted three crossover and mutation methods to evaluate their effectiveness in optimizing a rechargeable sensor network in our environment. One of which was inspired by the baseline crossover and mutation strategies proposed in [1].

3.1 Crossover

3.1.1 Baseline Crossover: Heuristic Crossover. This heuristic crossover improves the survival ratio by comparing sensor life spans. Let two parent chromosomes be Chr_i and Chr_j, each containing $(m+1)$ genes. The first gene represents the service station, while the remaining m genes correspond to specific requests.

(1) In the first iteration, the first gene (service station) from either Chr_i or Chr_j is randomly selected and passed to the offspring chromosome Chr_o.
(2) In the k-th iteration, the remaining sensor lives of genes $Chr_i(k)$ and $Chr_j(k)$ are compared. The gene with the shorter remaining life is copied to $Chr_o(k)$.
(3) To synchronize the parent chromosomes after each step, the remaining genes of the parent chromosome not selected are left-circumvolved until the two genes at position k match.
(4) This process repeats until all genes are assigned to the offspring.

Example:

- Parent 1 (Chr_i): $[S_2, S_1, S_3]$
- Parent 2 (Chr_j): $[S_1, S_3, S_2]$
- Sensor life spans: $Rl(S_1) = 30, Rl(S_2) = 18, Rl(S_3) = 25$
- Iteration details:
  (1) Iteration 1: The service station gene is randomly selected, e.g., $Chr_o(1) = Chr_j(1)$.
  (2) Iteration 2: Compare $Rl(S_2)$ and $Rl(S_1)$. $Chr_i(2)$ with S_2 is selected as it has a shorter remaining life.
  (3) Iteration 3: Both parents have S_1. Randomly assign it, e.g., $Chr_o(3) = Chr_i(3)$.
  (4) Iteration 4: Left shift the remain chromosomes.

3.1.2 Crossover 1: Order-Based Crossover. This type of crossover tries to preserve the parent gene order, which means that it maintains the charging sequence in this task.

(1) For each gene position in the offspring, determine whether it will be altered with a probability of 0.5.
(2) Gene positions that remain unchanged are directly inherited from the corresponding positions in the first parent.

(3) The positions marked for alteration are filled sequentially using the gene order of the second parent, ensuring that no duplicates are introduced.

Example:

- Parent 1 (*p1*): [0,5,6,7,3,2,4,1]
- Parent 2 (*p2*): [3,1,2,6,4,7,5,0]
- Suppose the following position are selected to be fixed:[1,2,4,7]
- The child first inherit the gene from parent 1:[-,5,6,-,3,-,-,1]
- Then filled in with the order in parent 2:[2,5,6,4,3,7,0,1]

3.1.3 Crossover 2: Priority-Based Crossover. This crossover strategy prioritizes genes based on the battery level of sensor nodes, with lower battery levels receiving higher priority. The process is detailed as follows:

(1) **Sort Parent 1:** Arrange the genes of Parent 1 in ascending order of battery level, giving higher priority to nodes with lower battery levels.
(2) **Select Top K Genes:** Use the Box-Muller transformation to generate a random number K. Copy the top K genes from Parent 1 to the child.
(3) **Fill Remaining Genes:** For the remaining positions in the child, randomly select genes from Parent 2. Only genes that have not yet been selected will be considered, ensuring uniqueness in the child's chromosome.

This strategy allows the child to inherit high-priority genes while maintaining genetic diversity through the contribution of genes from both parents.

3.1.4 Crossover 3: Edge-based Crossover. This crossover strategy is a slightly modified version of the original Edge Crossover. The process is detailed as follows:

(1) Construct an adjacency list from the two parents.
(2) Select a gene (sensor) with the lowest remaining life as the starting point.
(3) Repeatedly select the next node that has the lowest "available neighbor count."
(4) Replace any random selection required in the original Edge Crossover algorithm with a deterministic selection of the gene with the lowest remaining life instead.

Example:

- Parent 1 (*p1*): [0, 2, 1, 3, 4]
- Parent 2 (*p2*): [2, 0, 1, 4, 3]
- The adjacency list for each sensor is constructed as follows:
  - Sensor 0: {1, 2, 4}
  - Sensor 1: {0, 2, 3, 4}
  - Sensor 2: {0, 1, 3}
  - Sensor 3: {1, 2, 4}
  - Sensor 4: {0, 1, 3}
- Suppose the remaining life of sensors {0, 1, 2, 3, 4} is {27, 11, 2, 19, 30}. Based on the remaining life, the starting point is sensor 2 (lowest remaining life = 2). The child (*c*) path becomes [2, 3, 1, 4, 0].

3.2 Mutation

3.2.1 Baseline Mutation: Random Swap Mutation. The mutation operation involves altering a randomly selected chromosome to introduce genetic diversity. Let Chr_k be one of the two newly selected parent chromosomes Chr_i and Chr_j, chosen randomly for mutation.

(1) Randomly select two genes g_i and g_j within the chromosome Chr_k.
(2) Swap the positions of g_i and g_j in Chr_k to form a new chromosome Chr_o.
(3) Evaluate Chr_i, Chr_j, and Chr_o. Select the two best chromosomes to replace Chr_i and Chr_j.

The mutation operation is performed iteratively over G generations alongside the crossover operation:

- Crossover occurs with a probability Pr_c.
- Mutation occurs with a probability Pr_m.

3.2.2 Mutation 1: Split and Swap Mutation. This mutation method is to randomly split the charging route into two part and form a new path.

(1) A random position in the chromosome is selected as the split point, dividing it into two subparts: the left subpart and the right subpart.
(2) The two subparts are then swapped, exchanging their positions to form a new chromosome.

3.2.3 Mutation 2: Interval-Based Shuffle. This mutation strategy introduces genetic diversity by shuffling genes within a randomly selected interval. The procedure is as follows:

(1) **Select an Interval:** Randomly choose two positions within the chromosome to define the start and end of an interval.
(2) **Shuffle the Interval:** Randomly rearrange the genes within the selected interval, while keeping the rest of the chromosome unchanged.

By focusing variation within a localized region of the chromosome, this strategy maintains the overall structure of the solution while exploring new genetic configurations.

3.2.4 Mutation 3: Inverse Mutation. This mutation strategy aims to generate a reversed path. However, frequent reversals are unnecessary, as reversing the path twice would simply restore the original path. Thus, we propose the following approach.

(1) **Reverse Mutation (20% probability):** Simply reverse the path.
   - **Example:** Given parent (*p*) path [0, 5, 6, 7, 3, 2, 4, 1], the child (*c*) path becomes [1, 4, 2, 3, 7, 6, 5, 0].
(2) **Swap Mutation (80% probability):** Randomly select two positions in the path and swap their values.

4 EXPERIMENT

We employed two slightly different experimental designs to compare the performance differences of these various operators when applied to rechargeable sensor networks. By leveraging these designs, we aimed to evaluate how the distinct characteristics of each operator influence the number of dead nodes after the mobile charger (MC) make one complete trip. This approach provides a more comprehensive understanding of how we can apply evolutionary computation to the rechargeable sensor network.

4.1 Experiment 1: 4 Crossover × 4 Mutation Test

The objective of this experiment is to evaluate the performance of 16 different combinations of crossover and mutation strategies and identify the optimal configuration.

Settings:

- **Population Size:** 100
- **Number of Iterations:** 100
- **Number of sensors:** 200
- **Crossover Probability:** 0.9
- **Mutation Probability:** 0.04
- **Sensor Space:** All the sensors are placed in the bounded range of $[-50, 50] \times [-50, 50]$.

Evaluation Metric:
The primary metric for evaluation is the minimum dead nodes after completing the iterations. Among methods that achieve the same dead nodes, the approach that reaches this result in the fewest iterations is regarded the best one.

Result:
You can find the resulting dead nodes with respect to iterations in Figure1. For easy comparison, we also make a table (see Table1) to show the required iterations to reach the zero dead node. As we can see in the table, priority-based crossover achieves the best overall performance.

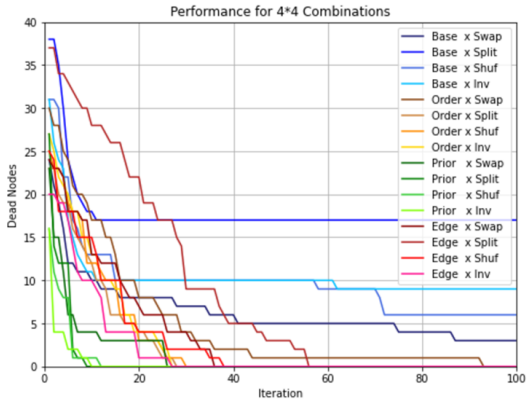

Figure 1: Performance Comparison of Crossover and Mutation Methods

Next, we conduct a comparative analysis of the mutation methods while keeping the crossover method fixed. For both order-based and priority-based crossovers, the choice of mutation method has minimal impact on their performance (see Figure2).

In contrast, mutation methods have a significant impact on the performance of both the baseline crossover and edge crossover. (see Figure3).

C \M	Swap	Split	Shuffle	Inverse
Baseline	x (3)	x (17)	x (6)	x (9)
Order-base	93	28	30	27
Edge-base	36	56	38	27
Priority-base	26	9	12	10

Table 1: The iterations to reach zero dead node for different combination. Note that the baseline crossover cannot reach zero dead node after 100 iterations. The number in the parenthesis represent the remaining dead nodes by using the baseline crossover.

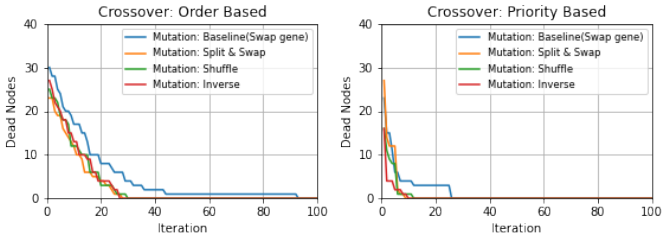

Figure 2: Mutation Impact on Crossover Strategies: Mutation method has minimal impact on the performance of Order-based and Priority-based crossovers.

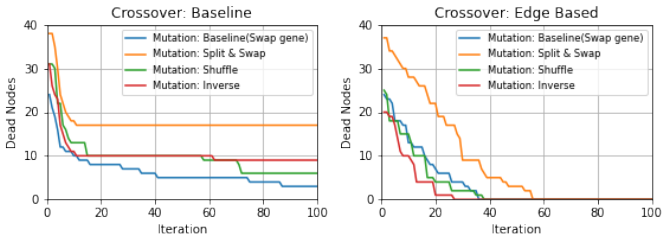

Figure 3: Mutation Impact on Crossover Strategies: Mutation method influences the performance of Original Strategy and Edge Recombination crossovers.

4.2 Experiment 2: Different Sensor Density

This experiment is designed to conduct an ablation study using three different sensor densities by varying the spatial distribution of the sensors. For simplicity, we focus solely on evaluating different crossover methods while using the baseline (swap) mutation strategy. Additionally, the number of iterations is adjusted based on the sensor density. The experimental settings are detailed below. (also see Table2).

Settings:

- **Population Size:** 100
- **Number of Iterations:** 100, 500, 1000
- **Number of sensors:** 200
- **Crossover Probability:** 0.9
- **Mutation Probability:** 0.04

- **Sensor Space:** All the sensors are placed in the bounded range of $[-k, k] \times [-k, k]$, where $k \in [50, 100, 200]$.

Density Level	Search Space	Iterations
High Density	$[-50, 50] \times [-50, 50]$	100
Medium Density	$[-100, 100] \times [-100, 100]$	500
Low Density	$[-200, 200] \times [-200, 200]$	1000

Table 2: Sensor Space and Iterations for Different Density Levels

Evaluation Metric:
The primary metric for evaluation is the minimum dead nodes after completing the iterations, which is the same as Experiment1.

Result:
From Figure 4, we observe that the priority-based crossover, which performed best in Experiment 1, nearly failed in the medium-density environment. In contrast, the order-based method, which was not particularly outstanding in Experiment 1, achieved the best results in both medium- and low-density scenarios. This discrepancy may be attributed to the greedy nature of the priority-based method.

Consider the following extreme scenario: if we always select the sensor with the lowest remaining battery life as the next stop, there is a risk of repeatedly circling around and arriving too late at each target sensor, resulting in battery depletion due to the sparse distribution of sensors. This issue is less likely to occur in high-density scenarios.

Therefore, it is advisable to select the most suitable crossover method based on the sensor density.

5 CONCLUSION

In addition to the original strategy, we explored three distinct crossover methods: order-based, edge-recombination, and priority-based.

Our findings indicate that the priority-based crossover method, which utilizes the remaining energy as the crossover indicator, performs well in high-density environments. However, it fails in medium- or low-density scenarios. This is because it disregards other critical factors, such as the time required to reach each sensor, leading to suboptimal performance and stagnation after a few iterations.

Unexpectedly, the uniform order-based crossover, which does not use the remaining lifetime as an indicator, demonstrated the most robust performance. It consistently outperformed the original method across all tested conditions, proving to be an effective approach for addressing the problem.

REFERENCES

[1] 2023. Genetic algorithm-based partial charging schedule of rechargeable sensor networks. *International Journal of Communication Systems* 36, 9 (2023), e5485 pages.

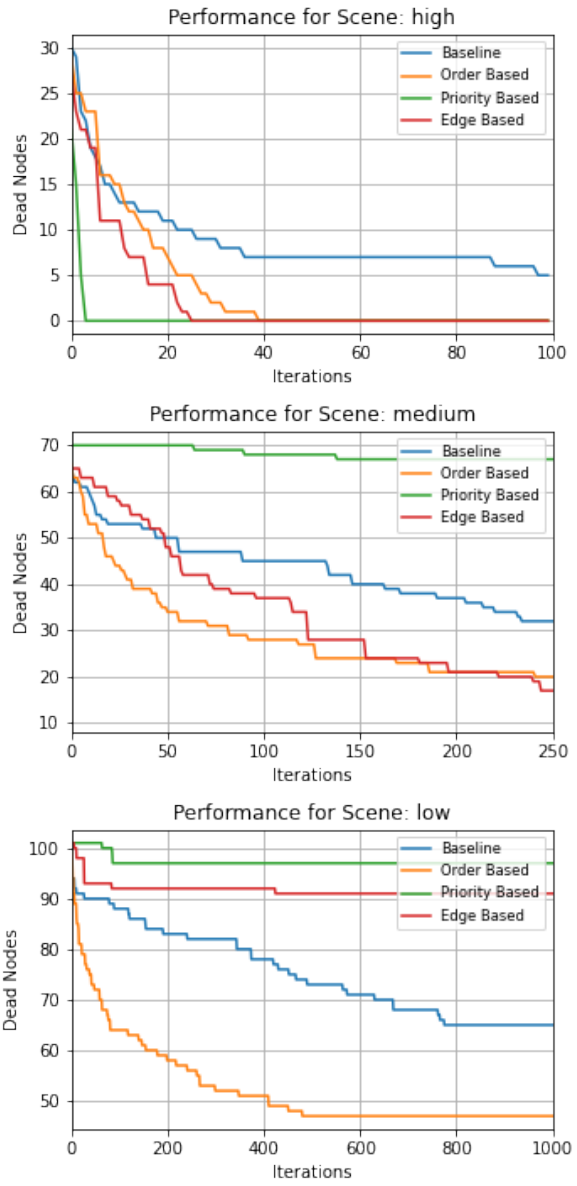

Figure 4: The performance of crossover methods under different sensor density.