Imperial College London Department of Earth Science and Engineering MSc in Applied Computational Science and Engineering

Independent Research Project

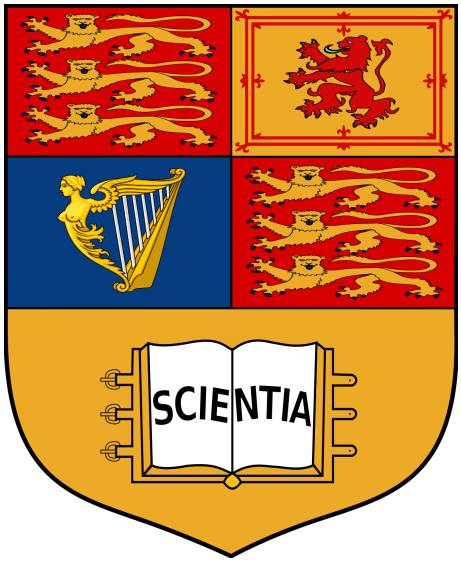
Optimal Drone Recharging Scheduling for Wireless Sensor Power Transfer and Data Collection

---

Qiuchen Qian

E-Mail: qiuchen.qian19@imperial.ac.uk
Github Login: acse-qq219

Supervisors: Dr. David Boyle, Dr. Adriana Paluszny



August 2020

ACKNOWLEDGEMENT

Firstly, I would like to appreciate my supervisors: Dr. David Boyle and Dr. Adriana Paluszny. Thanks to their patient guidance, I was able to complete the project and report successfully. I would like to show my gratitude to Ms. Akshayaa Pandiyan, who gave me patient and careful assistance in coding and project direction. Finally, I would like to thank all my friends and families. It is their support that provided me an excellent environment to implement this project.

ABSTRACT

Rechargeable power source is a popular option when considering power management of Internet-of-Things (IoT) projects. Automatic management of Wireless Rechargeable Sensor Networks (WRSN) can be feasible With combination of Wireless Power Transfer (WPT) and autonomous robotics technology. However, improving usage efficiency of vehicles can be challenging due to complex dynamic environment. For example, Drone Recharging Scheduling (DRS) problem, which is proposed as an optimization problem, often owns optimization objectives of minimizing energy cost of drones and maximizing total recharged energy. This report improves basic optimization algorithms (Genetic Algorithm (GA), Black Hole (BH) and Simulated Annealing (SA)) in solving DRS. To evaluate their performances under different scenarios, we designed 9 WRSNs with customized coverage areas and numbers of Sensor Nodes (SN). Through several tests, simulation results show that BH can have the best performance (i.e. charge most SNs, have shortest flight schedule, etc.) in most scenarios. Moreover, for all algorithms, a throughput of $\sim$85% can be achieved in large-scale networks.

Key words: Wireless Rechargeable Sensor Network, Optimization Algorithm, Recharge Scheduling

I. Introduction

Energy and data management of sensor nodes (SNs) can be tedious in most Internet-of-Things (IoT) projects, especially for dense systems. Some practical IoT projects harvest solar (Rao et al. 2017), heat (Stark 2006) and mechanical vibrations (Yang 2017) as power supply. However, the design of energy harvesting is often constrained to particular environment, which means flexibility and reliability cannot be guaranteed. Thus, Wireless Rechargeable Sensor Networks (WRSNs) is preferred in common situation. The commercial solution provided by (HEROTECH8 2020) demonstrates mature application of interactions between Power Delivery Vehicles (PDVs) and Facility Management. Many research have been carried out to prove that feasibility as well. In recent years, (Mitcheson et al. 2017) implemented Unmanned Aerial Vehicle (UAV)-based Inductive Power Transfer (IPT) and data collection through IEEE 802.15.4 2.4 GHz physical layer (PHY), which successfully obtained an IPT link efficiency of 90%. A more promising protocol was proposed by (Qin, Boyle, and Yeatman 2019), which builds a bespoke application layer upon augmented ContikiMAC over IEEE 802.15.4 PHY, which greatly increased reliability and network capacity. Though great possibilities of PDV-based WRSN solution can be found through their researches, drone positioning and flight path planning can be challenging under such scenario due to dynamic environmental factors and status of the PDV and SNs.

In the context of such WRSNs, Zorbas et al. introduced the Optimal Drone Positioning (ODP) problem for long-range wireless recharge process (Zorbas and Douligeris 2018). The main objective is to find optimal Power Delivery Vehicle (PDV) positions and minimize the total energy consumption of SNs. With different altitudes of the PDV, the number of covered SNs can vary. Two solutions were proposed: I. transform ODP to set-cover problem II. the Drone Positioning Heuristic (DPH) solution. While Cheng et al. converted the recharging process to Travelling Salesman Problem (TSP) with multiple PDVs (Cheng, Xu, and Wu 2019). The computational complexity of TSP is known as NP-complete (or NP-hard), which means the optimal solution can be found by traversing all possible solutions. But some local search algorithms can find an approximation optimal solution with fast convergence speed. Researchers adopted Genetic Algorithm (GA), which simulates the nature evolution process of chromosome with **Crossover, Mutation** and **Selection** stage, to schedule multiple PDVs charging routes. To ensure the population 'evolving' towards the desired direction (i.e. a solution within the trusted range), **Fitness function** is defined to evaluate each candidate in the whole population. Only candidates with higher metrics will survive for the next generation. To remain population diversity (i.e. avoid being stuck at local extreme value), some randomly initialized chromosome instances will be added as well. The simulation achieved good results with more than 96% SNs charged (Cheng, Xu, and Wu 2019) in a heavy traffic network scenario.

However, there are many limitations of the simulated system model in (Cheng, Xu, and Wu 2019). For example, they assume perfect charging process and the energy consumption of moving mobile charger is 0. To consider more practical scenario, the process of IPT should be simulated, as well as energy loss, recharging time, etc. (Boyle et al. 2019) proposed a more efficient Two-Stage Power Distribution System (2-PDS), which delivers energy to a cluster composed of one Centre Node (CN) and several End Nodes (ENs) with both IPT and Acoustic Power Transfer (APT). ENs can harvest acoustic energy through piezoelectric driver (see Fig. 1). Based on 2-PDS, (Pandiyan, M. E. Kiziroglou, et al. 2019) proposed a novel GA. They define the optimization objective as maximizing charged energy and minimizing PDV flight distance. According to simulation results, 2-PDS is proven to effectively reduce frequency of PDV visits and increase throughput (the percentage of successfully charged SNs with respect to the number of all SNs to be charged).

Except GA, there exists many other popular optimization algorithms. For instance, Simulated Annealing (SA), proposed by Pincus (Pincus 1970), can find an approximate global optimum in short time. Adewole et al. compared the performance of SA and GA and got the conclusion that they have unique advantages when solving classic TSP (Adewole, Otubamowo, and Egunjobi 2012). Moreover, according to experimental results of (Hatamlou 2018), Black Hole Algorithm (BH) can have a remarkable accuracy in finding the optimal solution. There is another related research integrated the idea from Particle Swarm Optimization (PSO) algorithm and **Mutation** of GA to solve 3D UAV optimal path planning problem (Chen et al. 2016). The modified algorithm showed a great improvement compared with basic GA, which mainly reflected in the accuracy of final solution and convergance speed. Also inspired by (Soto et al. 2018), which encodes each element in the solution with binary number 0 or 1 when solving Set Covering problem using BH, this study is intended to combine basic **Attraction** of BH and **Swap Mutation** of GA proposed in (Pandiyan, M. E. Kiziroglou, et al. 2019) (see section II).

The detail description of system model simulation and optimization algorithms will be delivered in Section. II. Section. III will describe metadata of code, including

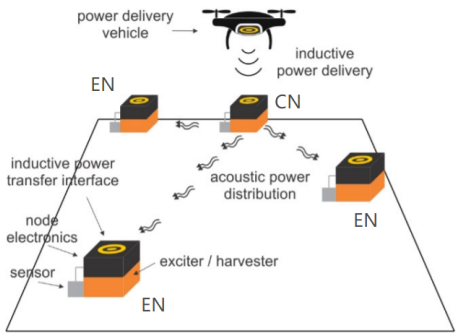


Fig. 1. Two-stage Power Distribution System (Boyle et al. 2019)

compilation dependencies. Code implementation and evaluation of experimental results with modified BH, GA and SA under different scales of the stated WRSN will be demonstrated as well in Section. IV. Finally, A conclusion of the project and future works will be presented in Section. V.

II. Software Description

This section summarizes the system model simulation, PDV flight simulation and optimization algorithms that were coded in *C++* std 17 with Microsoft Visual Studio 2019 for this project. This is because superior performance of *C++* in high computational complexity simulation. While for input data generation and result visualization, *Jupyter notebook* is selected due to its powerful ability of processing data I/O and visualization. The architecture of code design can be seen as Fig. 2. The concrete code logic will be discussed in this section using pseudo-code blocks.

A. System Model

The deployment of WRSN is considered to match the scenario of smart agriculture. The farm land is assumed as a square, including one Base Station (BS), N_{pdv} PDV(s) and N_{sn} randomly deployed SNs. This study selected DJI Matrice 100 (M100) quad-copter drone as PDV. (see Tab. I)

As for sensors, this study simulated LMT84 as temperature sensor and NPA300 as pressure sensor. To simplify model design, the specification of sensors is obtained from the research of Pandiyan et al. (Pandiyan, Boyle, et al. 2019) (see Tab. II). For optimization purpose, N_{pdv} is set to 1 initially, with BS located at origin (0, 0). Critical data stored in BS should be collected from previous recharging process, including coordinate, sensor type, current voltage V_{cur} and corresponding weight W (the ratio of V_{cur} to V_{max}) of each SN (DAT_{sn}), and position, flight distance d_{pdv}, flight time t_{pdv}

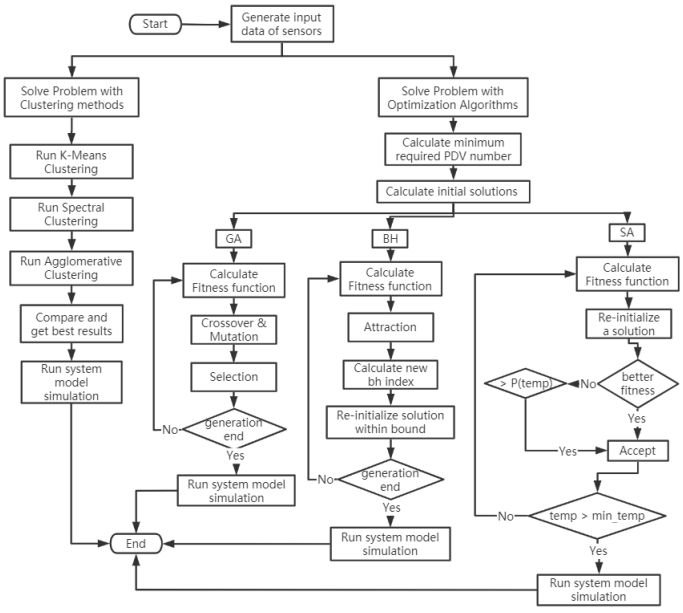


Fig. 2. Architecture of code design

TABLE I
DJI Matrice 100 Specifications (*Dji Matrice 100 User Manual* 2016)

	Parameter	Specifications
Performance	Hovering Time (2 TB48D batteries)	40 min
	Hovering Precision (GPS mode)	Horizontal: 2.5 m
	Maximum Speed of Ascent	5 m/s
	Maximum Speed of Descent	4 m/s
	Maximum Speed (GPS mode)	17 m/s
Battery	Model	TB48D
	Capacity	5700 mAh
	Energy	129.96 Wh
System Setting	Model	N1
	Flight Control	Programmable
	Navigation	Custom frame

and remain energy E_{pdv} of each PDV (DAT_{pdv}). Before single recharge cycle, PDV status will be reset (i.e. $E_{pdv} = E_{pdv}^{max}$ and $d_{pdv} = t_{pdv} = 0$). After calculation of sub flight path, the PDV will take off from the BS and visit each target coordinate until the end of waiting list obtained from the Shortest-Job-Next (SJN) strategy, and then execute Return-To-Home (RTH) instruction, which means the PDV will return to the BS immediately. Due to environmental and algorithmic limitations, the PDV may implement RTH in advance if E_{pdv} is not enough.

1) Power Delivery Vehicle: When M100 is approaching a coordinate through GPS mode, the installed module can only achieve accuracy of 2.5 m (*Dji Matrice 100 User Manual* 2016). According to (Mitcheson et al. 2017), when implementing IPT with two mid-size (20 cm diameter) coils, if the distance between them is longer than 2 m, the link efficiency will be reduced to 30%. Thus, Ultra-Wide-Band (UWB) positioning, which can shorten the distance to 1 m, will be applied to improve recharging and data collection efficiency. Considering possible error, a random offset (ranging from -1 to +1 m) will be added between GPS and UWB positioning. Moreover, there is an assurance mechanism, which checks needed energy of next behavior, to ensure the PDV has enough energy to implement IPT and RTH.

2) Recharge Wireless Sensor Nodes: In 2-PDS, one SN can be classified as CN or EN. With different sensor types, some attributes of the node can vary (i.e. capacitance, critical

TABLE II
SN Specifications (*PowerStor: Supercapacitors PHB Series Technical Sheet 4402* 2011) - (*NovaSensor NPA Surface Mount Pressure Sensors Datasheet* 2019)

Parameter	Temperature SN	Pressure SN
Sensing Unit		
Model	LMT84	NPA300
Voltage Typ.	1.5 V	3.3 V
Voltage Max.	2.5 V	5.0 V
Current Typ.	5.4 μA	1.2 mA
Sampling Freq.	0.01 Hz	0.1 Hz
On Time	2 ms	2 ms
Super Capacitor		
Model	HB 1030-2R5106	PHB-5R0
Capacitance	6 F	3 F
Leakage Current	20 μA	16 μA

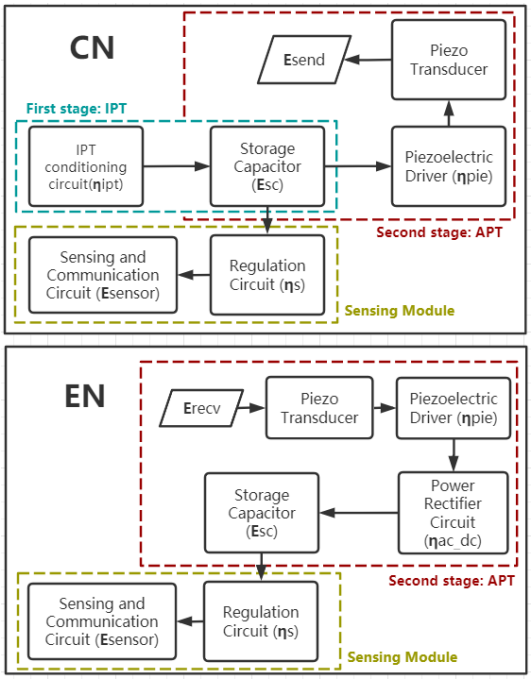


Fig. 3. Main process diagram of CN and EN (Pandiyan, Boyle, et al. 2019)

voltage, etc.). However, there is no hardware difference when defining a SN or CN (one EN can be CN in another cluster). The architecture of CN and EN can be seen as Fig. 3.

IPT occurs only at CNs whose $V_{cur} \leq V_{min}$. The difference between the PDV position and real coordinate will be ignored in simulation. To compensate this error, the link efficiency factor η_{ipt} is set to 50%. Besides, the energy level of super capacitor (E_{sc}) should be remain within 10-90% range for longer life time. Therefore, received energy 'packet' of the node can be calculated as (1).

$$E_{ipt} = \frac{1}{2} C(0.9 \cdot V_{max}^2 - V_{cur}^2) \cdot \eta_{ipt} \tag{1}$$

Acoustic Power Transfer (APT) will be performed after IPT. All adjacent nodes (defined as ENs) have distance (d) within acoustic wave propagation limit (0.7 m in this case) will be assigned to the same cluster, excluding nodes whose d is less than far field distance (0.215 m) (to avoid near field interference). In (Boyle et al. 2019), researchers successfully realized acoustic power distribution under 47.5 kHz signal (with 0.1% corresponding efficiency η_{aco}). Obtained from the experimental data (M. Kiziroglou et al. 2017), the received energy 'packet' from node m to n can be calculated as (2).

$$E_{apt} = \eta_{pie}^2 \cdot \eta_{ac\_dc} \cdot \exp(-2\pi f^{\eta_{aco}} \cdot d_{mn} \cdot \alpha_{mat}) \cdot E_{send} \tag{2}$$

where η_{pie} is the coupling efficiency of the piezoelectric transducer, which needs to be multiplied twice from transmitter to receiver. η_{ac_dc} is the efficiency of a Schottky bridge rectifier. Finally, considering energy consumption (E_{sn}) and storage leakage (μ) of the SN, the energy of CN and EN storage capacitor (E_{sc}) can be updated as (3). Note that E_{sc} cannot exceed 90% of E_{sc}^{max}.

$$E_{sc} = \begin{cases} (E_{cur} + E_{ipt} - \frac{E_{send}}{\eta_{pie}} - \frac{E_{sn}}{\eta_s}) \cdot \mu & \text{if CN} \\ (E_{cur} + E_{apt} - \frac{E_{sn}}{\eta_s}) \cdot \mu & \text{if SN} \end{cases} \tag{3}$$

B. Flight Simulation

The BS will implement pre-calculation (i.e. sensor node clustering and sub-path assignment) to set the minimum required N_{pdv} and corresponding flight path plans (list of target coordinates L_{tar}). Starting from BS, the flight strategy follows SJN, finding the nearest neighbour in charging list as next target coordinate.

This is implemented by updating the SN energy consumption (i.e. caused by sensing, communication, etc.) only after the recharging cycle. Moreover, all PDVs are assumed to perform recharging simultaneously, which means `line 16` in Alg. 1 depends on the longest flight time.

C. Optimization Algorithms

As a baseline, this study provide a naive solution with clustering methods (i.e. K-Means, Spectral and Agglomerative clustering provided by Scikit-learn library (Pedregosa et al. 2011)) and SJN strategy. According to (Zorbas and Douligeris 2018), initial solutions generated through different strategies (i.e. random, clustering, SJN, etc.), defined as target vectors V_{pdv}, can have different degrees of impacts on final results. Alg. 2 computes the minimum required N_{pdv} and randomly initializes other clustering approaches. According to N_{pdv} and the number of SNs requested to be recharged (N_{rec}), charging list (V_{rec}) can be equally divided into sub vectors. For situations where N_{rec} is not divisible by N_{pdv}, redundant SNs will be added to the sub path of last PDV.

Algorithm 1 Flight Simulation Process of single PDV

Input: N_{pdv}; L_{tar}; Initial DAT_{sn}
1: Calculate ascent and descent energy and time
2: Update ascent energy and time
3: **repeat**
4: Calculate distances between PDV and SNs $\in L_{tar}$
5: Find the index of the shortest distance
6: Calculate IPT, approaching and RTH energy and time
7: **if** $E_{rth} + E_{ipt} + E_{app} + 0.1 \cdot E_{pdv}^{max} \geq E_{pdv}$ **then**
8: **goto** line 15
9: **end if**
10: Update PDV position, distance, energy and time
11: Update CN voltage and energy
12: Update EN voltage and energy
13: Remove the CN coordinate from target list
14: **until** L_{tar} is empty
15: Implement RTH
16: Update energy consumption of all SNs
17: Reset DAT_{pdv}
18: **return** Updated DAT_{sn}

This study will implement Genetic, Black Hole and Simulated Annealing Computing for optimised recharge schedule and the searching efficiency and solution performance are compared. Total recharged energy (E_{rec}), PDV flight distance (d_{pdv}) and energy cost (ΔE_{pdv}) are used as evaluation criteria of the different algorithms. Because the 3 parameters have different scales and units, scale constraints and activation function ($tanh(x)$) are applied to make them dimensionless. Total recharged energy can be calculated as (4). In contrast to SJN, Longest-Job-Next (LJN) are defined as the 'worst' case, which finds the longest distance as next target. Note that d_{sjn} and d_{ljn} represent calculated distance using SJN or LJN.

$$E_{rec}^{cons} = \frac{\sum E_{rec}^{CN} + \sum E_{rec}^{EN}}{(\sum E_{rec}^{CN})_{\text{V=0}\,\rightarrow\,\text{V=}V_{max}} + \sum E_{rec}^{EN}} \tag{4}$$

$$d_{pdv}^{norm} = \frac{d_{bs\rightarrow sn_{(1)}} + \sum d_{i\rightarrow i+1} + d_{sn_{(N_{sp})}\rightarrow bs} - d_{sjn}}{d_{ljn} - d_{sjn}} \tag{5}$$

$$\Delta E_{pdv}^{norm} = \frac{E_{pdv}^{max} - E_{pdv}^{end}}{E_{pdv}^{max}} \tag{6}$$

Then the fitness metric (M) can be calculated as (7).

$$M = \begin{cases} \alpha \cdot tanh(E_{rec}^{norm}) \\ +\,\beta \cdot (1 - tanh(d_{pdv}^{norm})) & \text{if PDV can finish} \\ +\,\gamma \cdot (1 - tanh(\Delta E_{pdv}^{norm})) \\ \\ -100 & \text{otherwise} \end{cases} \tag{7}$$

where, α, β and γ are weight constants. Here we force M to be -100 if the PDV cannot finish the charging list in case of large scale network (i.e. 1500 SNs in a 9 km^2 size square). This will make fitness selection easier.

1) Genetic Algorithm: Classical GA takes chromosome evolution as its model. Several chromosomes (possible solutions) are initialized in a population of N_{pop} candidates to make a target vector. In the next step, **Crossover** takes two target vectors from the population to generate a trail vector through randomly exchanging their elements. This process is overlooked by a GA parameter called Crossover Ratio (cr). Sequentially, **Mutation** randomly changes elements in the trail vector. However, performing **Crossover** and **Mutation** will disrupt sub paths of other PDVs because the optimization objective is to recharge **all** SNs with low energy, the 'mutated' SN will be one in another sub path. Therefore, the Alg. 3 finds a n-th nearest SN p_n, where n is a random integer and p_n should be one member in the charging list (V_{rec}). Then, two SNs will be swapped to complete $C\&M$. Finally, **Selection** compares the fitness metric of target and trail vectors. Higher metric candidates will survive for next generation. Over N_{gen} generations, GA can find an approximately optimal solution (with the highest fitness metric).

Algorithm 2 Initialization Process

Input: Initial DAT$_{sn}$; pop
Initialize SNs vector to be recharged $V_{rec} = \{p_1, \ldots, p_n\}$
Initialize target vector of all PDVs $V_{pdvs} = \{p_1\}$
repeat Execute Alg.1. with **Input** N_{pdv}, V_{pdvs} and DAT$_{sn}$
  if line 8 in Alg.1. is **true then**
    $N_{pdv} \leftarrow N_{pdv}$ + 1 ▷ Can skip Alg.1. line 15-16
  else
    Push next p to V_{pdvs}
  end if
until $V_{pdvs} == V_{rec}$
Initialize DAT$_{pdv}$ and sub path length $N_{sp} = n/N_{pdv}$
for $i \in [1, pop]$ **do** ▷ Initialization
  for $j \in [1, N_{pdv}]$ **do**
    for $k \in [1, N_{sp}]$ **do**
      Calculate distances between PDV$_j$ and $p_n \in V_{rec}$
      For random m-th nearest p_m, let $V_{pdv_j}^i[k] = p_m$
      Remove p_m from V_{rec}
      Update PDV$_j$ pos
    end for
  end for
end for
return N_{pdv}; All initial guesses $V_{pdv_j}^i$

2) Black Hole Algorithm: Classic BH takes the process of black hole engulfing stars as the model. Similar as GA, several stars (solutions) are initialized until pop. The solution with the highest M will be set as **black hole** (bh). In each generation, other stars will be attracted to bh. Displacement of the star at the dimension i can be updated as (8).

Algorithm 3 Modified Genetic Algorithm

Input: Initial DAT$_{sn}$; pop; All guesses $V_{pdv_j}^i$; N_{pdv}; cr; gen
Calculate M_j^i of $V_{pdv_j}^i$ ▷ $i \in [1, pop]$, $j \in [1, N_{pdv}]$
repeat
  for $i \in [1, pop]$ **do** ▷ **Crossover**& **Mutation**
    for $j \in [1, N_{pdv}]$ **do**
      Initialize trail vector $TV_{pdv_j}^i = V_{pdv_j}^i$
      foreach $p_k \in TV_{pdv_j}^i$ **do** ▷ $k \in [1, N_{sp}]$
        if rand[1, 100] $\geq cr$ **then**
          For n-th nearest $p_n \in V_{pdvs}^i$, swap p_k and p_n
        end if
      end for
      Calculate M_j^i of $V_{pdv_j}^i$ and $M_j^{i'}$ of $TV_{pdv_j}^i$
    end for
    Get $M_{sum}^i = \sum_{j=1}^{N_{pdv}} M_j$ and $M_{sum}^{i'} = \sum_{j=1}^{N_{pdv}} M_j{}'$
    if $M_{sum}^{i'} \geq M_{sum}^i$ **then**
      Update $V_{pdv_j}^i \leftarrow TV_{pdv_j}^i$ and $M_{sum}^i \leftarrow M_{sum}^{i'}$
    end if
  end for
until gen reached
return The solution of all PDVs with $\{M_{sum}^i\}_{max}$

$$d_i = rand(0,1] \cdot \sqrt{X_i(\text{BH}) - X_i(\text{star})} \tag{8}$$

where, X represents location and $\sqrt{X_i(\text{BH}) - X_i(\text{star})}$ is the Euclidean distance. However, X is hard to calculate because each element in the candidate is obtained from requested SNs with fixed coordinates. When performing **Attraction**, the displacement will depend on the distance between two points with the same index. Then the point in $X_i(\text{star})$ at next generation will be fixed to one point in L_{rec} (best match of the d_i). After an iteration, if there is any solution with higher M, former bh will be replaced. The bound of bh can be calculated as (9) according to its fitness metric M_{bh}.

$$R_{bh} = \frac{M_{bh}}{\sum_{i=1}^{pop} M_i} \tag{9}$$

R_{bh} will be used as a selection criteria, which will be compared with a random number $\in (0, 1]$. Any candidates who owns smaller random number will be removed. To ensure there is no change on population size and remain population diversity, a random initialized star will be generated.

3) Simulated Annealing Algorithm: SA is a searching algorithm which shows good performance in solving TSP (may find the global optimal solution). Unlike Hill Climbing method, there exists a possibility for SA to accept another solution with 'worse' fitness metric. In this case, the new solution (trail vector) will be generated through randomly swapping two SNs in charging list once. This possibility depends on **temperature** (T), which gradually reduces by multiplying a cooling rate factor $r \in (0, 1)$ (normally close to 1) and fitness metric difference ΔM (see Alg. 5).

Algorithm 4 Modified Black Hole Algorithm

Input: Initial DAT$_{sn}$; pop; All guesses $V^i_{pdv_j}$; N_{pdv}; ar; gen
Calculate M^i_j of $V^i_{pdv_j}$ ▷ $i \in [1, pop]$, $j \in [1, N_{pdv}]$
Find bh index with $M^i_{j\,max}$
repeat
  for $i \in [1, pop]$ **do**
    for $j \in [1, N_{pdv}]$ **do**
      for $k \in [1, N_{sp}]$ **do**
        Calculate displacement d_k as radius
        For n-th nearest $p_n \in V^i_{pdvs}$, swap p_k and p_n
      end for
      Re-calculate $M^{i'}_j$
    end for
    Re-calculate $M^{i'}_{sum}$
  end for
  Find bh ($\{M^i_{sum}\}_{max}$) and calculate bound R of bh
  for $i \in [1, pop]$ **do**
    if $\frac{M^i}{\sum_{i=1}^{pop} M^i} \leq R$ **then**
      Generate a new V^i_{pdvs} of all PDVs
    end if
  end for
until gen reached
return The solution of all PDVs with $\{M^i_{sum}\}_{max}$

Algorithm 5 Modified Simulated Annealing Algorithm

Input: Initial DAT$_{sn}$; pop; All guesses $V^i_{pdv_j}$; N_{pdv}; gen
Calculate fitness of all guesses
for $i \in [1, pop]$ **do**
  repeat
    for $j \in [1, N_{pdv}]$ **do**
      Re-initialize new $TV^i_{pdv_j}$ with randomly swapping
      Calculate new $M^{'}_j$ of $TV^i_{pdv_j}$
    end for
    Calculate $M^{i'}_{sum} = \sum_{j=1}^{N_{pdv}} M_j{}'$
    if $M^{i'}_{sum} \geq M^i_{sum}$ **then**
      Update $V^i_{pdvs} \leftarrow TV^i_{pdvs}$ and $M^i_{sum} \leftarrow M^{i'}_{sum}$
    else if $e^{\frac{\Delta M_{sum}}{T}} > rand\ [0, 1)$ **then**
      Update $V^i_{pdvs} \leftarrow TV^i_{pdvs}$ and $M^i_{sum} \leftarrow M^{i'}_{sum}$
    end if
    Update $T \leftarrow r \cdot T$
  until $T \leq T_{min}$
end for
return The solution of all PDVs with $\{M^i_{sum}\}_{max}$

III. Code Metadata

This project was built under Windows environment with Microsoft Visual Studio 2019 and C++ 17 language standard (due to high requirement of file I/O). Library like *Boost.Filesystem* provides functionality of counting file number in the directory (Dawes 2020). For standard libraries, for example, *set*, *string*, and *vector* were used for specific data structures (i.e. unique set for deleting duplicates, file names, vectors to store index of solutions, etc.). Other libraries like *random*, *chrono*,and *math* were used for specific operations (i.e. generate a uniform distribution, apply mathematical functions, etc.). As for Python, libraries of *matplotlib* are used for visualization, *pandas, numpy* used for data processing and *Scikit-learn* for clustering algorithms (Pedregosa et al. 2011).

In the Github repository[1], only source codes are uploaded, users are required to create a new project and add those files to specific directory. We strongly recommend to deploy files following the folder structure of all *README* files. The source code is located at *src* directory, including unit tests files, *.ipynb* files, input and output folders. Documentation files can be found in *doc* directory, which are generated to *.html* files with *Doxywizard*. Experimental results can be found in *results* directory. The result visualization is placed in *fig* directory.

IV. Results and Discussion

This section will introduce results from simulations of proposed algorithms. The binary code was run on Intel Core i7 CPU @ 2.60 GHz and 8 GB RAM to get results. Moreover, functionalities of the system model and

[1]More information can be found in Github: https://github.com/acse-2019/irp-acse-qq219.git

optimization algorithms are verified through Microsoft Unit Test. For example, in sample test code[2], an extreme case (4 SNs in all and 3 of them are requested to be recharged) was tested through GA. The error between analytic solution was compared as well, which could be accepted in a reasonable range. Except unit test, we consider five metrics when evaluating the algorithm performance:

- *Flight Distance [m]*: The total flight distance of all PDVs.
- *Energy Cost [Wh]*: The total energy cost of all PDVs.
- *Charged Energy [J]*: All charged energy, including IPT and APT.
- *Throughput [%]*: The percentage of successfully charged SNs with respect to the number of all SNs which requested to be recharged.
- *Execution Time [s]*: The execution time of an algorithm. Timing begins after initialization and ends when iteration completed (i.e. find the 'best' solution).

Though the optimization objective is to minimize flight distance and maximize charged energy, we consider total charged energy with the highest priority when evaluating the solution.

A. Setting of System Parameters

GA, SA and BH own different concepts and hence different parameters were set. Table. III summarises the algorith parameters which were used in this work. Note that parameter names are same with those used in the code. Through experiments, we found these algorithms are sensitive to initial solutions. Thus, `pop` and `generation` of GA and BH are set to 250 and 50 respectively. `cr` and `ar` reflect the frequency of solution change. We set a binary probability (50%) to make the frequency moderate. There is no strict definitions of population and generation in SA. Considering the execution time of SA can be unnecessarily long for large population size, we set `pop` to 25, `init_temp` to 1.2×10^3, `min_temp` to 1×10^{-5} and `r` to 0.98.

B. Simulation Results

The network scale depends on the number of SNs (N_{sn}) and square area (S_{squ}). According to (Chen et al. 2016), the throughput can only achieve $\sim$10% when $S_{squ} \geq 20$ km^2. Thus, 9 scenarios are designed ($N_{sn} \in \{500, 1000, 1500\}$ and $S_{squ} \in \{1, 4, 9\}$ km^2). Initial attributes of SNs (i.e. coordinates, sensor types and voltages) are generated through uniform distribution. All presented results are run 3 times and averaged. In Tab. IV, V and VI, the ***bold and italics algorithm*** represents the algorithm with best performance. Only typical flight paths visualization will be shown in the paper.

1) $N_{sn} = 500$: For small network scales, the total charged energy will be similar because most SNs can be recharged. Therefore, we set the weight factor of `coeff_dist` to 60%, `coeff_wsn_eng` to 30% and `coeff_pdv_eng` to 10%, representing worst case scenario for the IPT systems. For the smallest scale case, all algorithms can find the optimal solution with similar performance (see Tab. IV). With larger S_{squ}, drones are less likely to complete the sub path. For **case II**, though with similar charged energy, the solution generated by GA has the shortest distance and least PDV energy cost.

2) $N_{sn} = 1000$: After case 4, we change the weight factor of `coeff_wsn_eng` to 50% and others to 25% because there is a high possibility that some PDVs cannot complete assigned path. When S_{squ} equals to 1 km^2, only subtle differences between algorithms can be observed. BH shows better performance than SA and GA (see Tab. V). In **case V**, BH has advantages in all aspects. Besides, in **case VI**, though GA owns similar charged energy, BH can finish tasks with shorter flight distance and less PDV energy cost.

3) $N_{sn} = 1500$: Similar to the previous discussed cases, solutions of three algorithms show related performance under small S_{squ}. In **case VIII**, GA has slight advantages in all aspects. But in the designed largest scale, BH can charge more SNs than GA and SA (see Tab. VI).

C. Algorithm Complexity Analysis

In all algorithms, 3-dimensional vectors will be used to record possible solutions. Thus the space complexity should be $O(N_{pop} \cdot N_{pdv} \cdot N_{sp})$, where $N_{pdv} \cdot N_{sp} = N_{rec}$.

[2]More information can be found in Github: https://github.com/acse-2019/irp-acse-qq219/tree/master/Code/unit_test

TABLE III
SYSTEM PARAMETERS

Algorithm	Parameter	Meaning
Common	`coeff_wsn_eng`	Weight coefficient of *Charged Energy*
	`coeff_pdv_eng`	Weight coefficient of *Energy Cost*
	`coeff_dist`	Weight coefficient of *Flight Distance*
	`max_num_r`	Random neighbour index selection range
GA & BH	`generation`	The number of iterations
	`pop`	The number of possible solutions
	`cr`	GA Crossover ratio (0, 1]
	`ar`	BH Attraction ratio (0, 1]
SA	`pop`	The number of possible solutions
	`init_temp`	Initial temperature
	`min_temp`	End temperature
	`r`	Factor to reduce temperature

TABLE IV
RECHARGING RESULTS WITH $N_{sn} = 500$

Case	S_{squ}	Alg.	PDV	d_{pdv}[m]	Throughput	ΔE_{pdv}[Wh]	E_{chr}[J]
I	1 km^2	BH	0	6141.72	100.00 %	106.36	1154.46
		GA	0	6141.72	100.00 %	106.35	1154.46
		SA	0	6141.72	100.00 %	106.29	1154.46
II	4 km^2	BH	0	7273.60	100.00 %	124.47	665.09
			1	8765.42	100.00 %	149.66	693.89
		GA	0	7139.26	100.00 %	121.96	667.51
			1	8753.68	100.00 %	149.20	691.47
		SA	0	8335.88	100.00 %	142.37	668.97
			1	8525.41	100.00 %	145.61	690.01
III	9 km^2	BH	0	6376.43	100.00 %	109.12	454.72
			1	8497.95	100.00 %	147.95	459.52
			2	9671.01	65.22 %	164.25	320.74
		GA	0	6323.64	100.00 %	108.23	453.82
			1	8562.81	100.00 %	145.96	458.89
			2	9448.99	69.57 %	160.57	342.38
		SA	0	8971.94	100.00 %	152.85	456.64
			1	8642.56	100.00 %	147.30	456.21
			2	9377.83	71.01 %	159.39	349.48

TABLE V
RECHARGING RESULTS WITH $N_{sn} = 1000$

Case	S_{squ}	Alg.	PDV	d_{pdv}[m]	Throughput	ΔE_{pdv}[Wh]	E_{chr}[J]
IV	1 km^2	BH	0	5040.07	100.00	88.90	1415.36
			1	6339.45	100.00	110.79	1410.00
		GA	0	4982.85	100.00	87.99	1413.98
			1	6290.64	100.00	110.02	1411.38
		SA	0	5423.62	100.00	95.34	1413.56
			1	5683.38	100.00	99.72	1411.80
V	4 km^2	***BH***	0	7050.54	100.00	121.49	1002.66
			1	8142.12	100.00	139.89	1021.31
			2	9542.21	86.53	163.16	882.22
		GA	0	6658.5	100.00	114.88	1002.63
			1	8599.32	100.00	147.59	1023.31
			2	9641.90	86.53	164.83	877.15
		SA	0	8486.43	100.00	145.64	1013.24
			1	8501.63	100.00	145.90	1012.82
			2	9692.85	84.40	165.61	857.08
VI	9 km^2	***BH***	0	5267.53	100.00	90.12	493.07
			1	8350.90	100.00	142.07	498.38
			2	9575.65	100.00	162.70	494.78
			3	9336.55	75.36	158.42	374.93
			4	9112.56	62.82	154.60	346.24
		GA	0	5479.73	100.00	93.70	494.44
			1	8164.91	100.00	138.94	497.79
			2	9589.07	100.00	162.93	494.89
			3	9672.37	79.71	164.12	394.78
			4	9237.58	58.97	156.66	326.61
		SA	0	6372.99	100.00	108.75	494.58
			1	8159.47	100.00	147.15	494.36
			2	9494.39	94.20	161.27	469.79
			3	9663.95	95.36	163.94	374.56
			4	8766.14	55.12	148.68	304.97

TABLE VI
RECHARGING RESULTS WITH $N_{sn} = 1500$

Case	S_{squ}	Alg.	PDV	d_{pdv}[m]	Throughput	ΔE_{pdv}[Wh]	E_{chr}[J]
VII	1 km^2	BH	0	6668.93	100.00	117.58	2216.12
			1	7396.88	100.00	129.87	2259.15
		GA	0	6604.19	100.00	116.55	2216.28
			1	7372.73	100.00	129.52	2258.99
		SA	0	7301.50	100.00	128.89	2217.69
			1	7101.04	100.00	125.53	2257.58
VIII	4 km^2	BH	0	9176.97	100.00	158.67	1307.65
			1	9261.45	100.00	160.09	1352.41
			2	9614.74	82.01	165.39	1130.68
		GA	0	9025.15	100.00	156.26	1369.53
			1	9254.37	100.00	160.12	1352.97
			2	9720.01	83.07	167.32	1140.75
		SA	0	9647.87	96.83	166.30	1331.07
			1	9502.01	100.00	163.94	1354.85
			2	9743.58	85.71	167.52	1172.11
IX	9 km^2	***BH***	0	6052.09	100.00	104.07	797.92
			1	9304.98	100.00	158.88	809.73
			2	9759.75	100.00	166.53	796.64
			3	9704.20	72.97	165.14	595.51
			4	8830.72	56.67	150.22	490.69
		GA	0	6160.56	100.00	105.90	796.49
			1	8996.20	100.00	153.17	812.15
			2	9688.32	100.00	165.33	796.82
			3	9572.76	72.97	162.92	592.88
			4	8942.47	49.17	151.96	425.96
		SA	0	7090.23	100.00	121.56	796.88
			1	8722.25	100.00	149.07	809.38
			2	9632.02	100.00	164.39	802.36
			3	9811.40	68.47	166.86	551.14
			4	8592.49	40.00	145.89	351.07

We consider the worst case when evaluating the time complexity. For GA, it takes $N_{pop} \cdot (N_{pdv} \cdot N_{sp})^2$ iterations to execute **Crossover**, and $N_{sp} \cdot N_{sn}$ iterations to execute **Fitness function**. Therefore, the time complexity of GA should be $O(N_{gen} \cdot N_{pop} \cdot N_{rec} \cdot N_{sn})$. In BH, performing **Attraction** takes $N_{pdv} \cdot N_{sp}^2$ iterations. Since BH has the same fitness function with GA, the time complexity should be $O(N_{gen} \cdot N_{pop} \cdot N_{rec} \cdot N_{sn})$ as well. Meanwhile for SA, the time complexity depends on the initial and minimum allowed temperature, which has more iterations compared with GA and BH. Therefore, this study will not elaborate it. See Tab. VII for detail execution time under 9 scenarios. Note that `pop` of SA is set to **50** at case 3, 6, 9, 12 while to **25** at other cases.

D. Flight Path Visualization

This section will include 3 typical scenarios (**III, VI** and **IX**). Their complete flight paths are visualized as Fig. 4, 5 and 6. Some edge overlap can be observed because the algorithm clusters SNs in the charging list and PDV flight follows SJN strategy. With more SNs, more PDVs are needed. Peripheral PDVs (i.e. PDV 3 and PDV 4 in Fig. 5 and 6) can be ineffective, which may lead to unnecessary energy consumption. Therefore, deploying two BSs (i.e. at (0, 750) and (0, -750)) can be an option to solve this problem.

E. Summary

Based on above results, BH demonstrates good performance in solving stated problem, especially for large scale networks.

TABLE VII
EXECUTION TIME SUMMARY

Case	S_{squ}	N_{sn}	Alg.	Min. [s]	Avg. [s]	Max. [s]
1	1 km^2	500	BH	167.75	170.49	175.27
			GA	167.97	176.66	185.48
			SA	397.68	415.00	424.60
2	1 km^2	1000	BH	637.32	646.09	653.33
			GA	643.98	646.99	650.68
			SA	854.37	882.88	905.52
3	1 km^2	1500	BH	1206.67	1222.82	1236.41
			GA	1161.18	1181.59	1203.70
			SA	1661.44	1703.37	1717.18
4	4 km^2	500	BH	179.92	187.85	192.61
			GA	182.03	186.94	195.05
			SA	203.27	206.91	212.02
5	4 km^2	1000	BH	601.50	607.61	613.94
			GA	620.11	626.84	631.40
			SA	640.92	661.12	688.21
6	4 km^2	1500	BH	909.54	915.28	922.12
			GA	922.97	927.70	930.58
			SA	481.43	482.08	482.60
7	9 km^2	500	BH	172.99	175.46	177.22
			GA	177.24	176.91	177.55
			SA	155.37	167.72	177.63
8	9 km^2	1000	BH	445.09	454.47	460.56
			GA	439.72	450.72	458.44
			SA	387.87	394.08	406.46
9	9 km^2	1500	BH	819.70	844.09	872.65
			GA	853.58	864.49	864.49
			SA	787.80	818.69	859.91

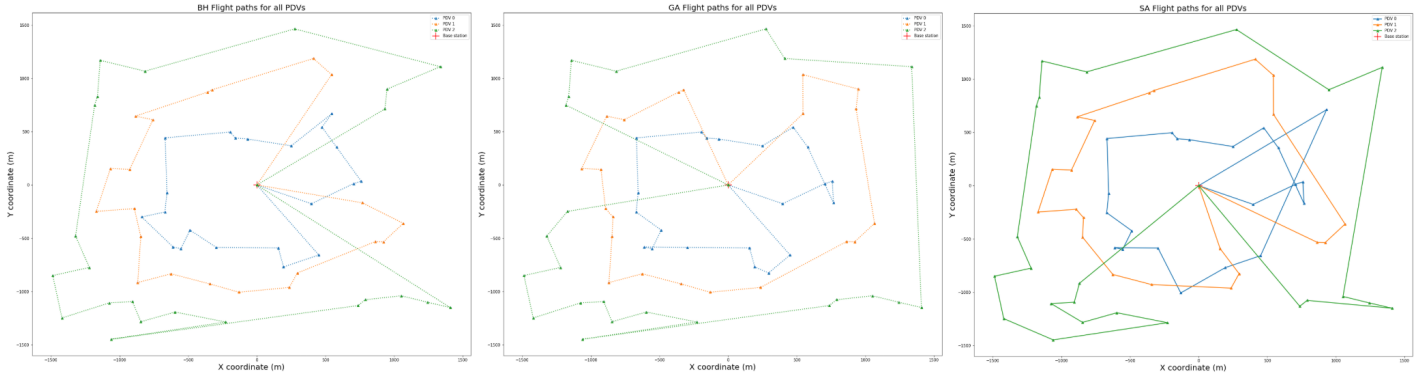


Fig. 4. Complete flight paths of algorithms (Case 3: $S_{squ} = 9km^2 \& N_{sn} = 500$)

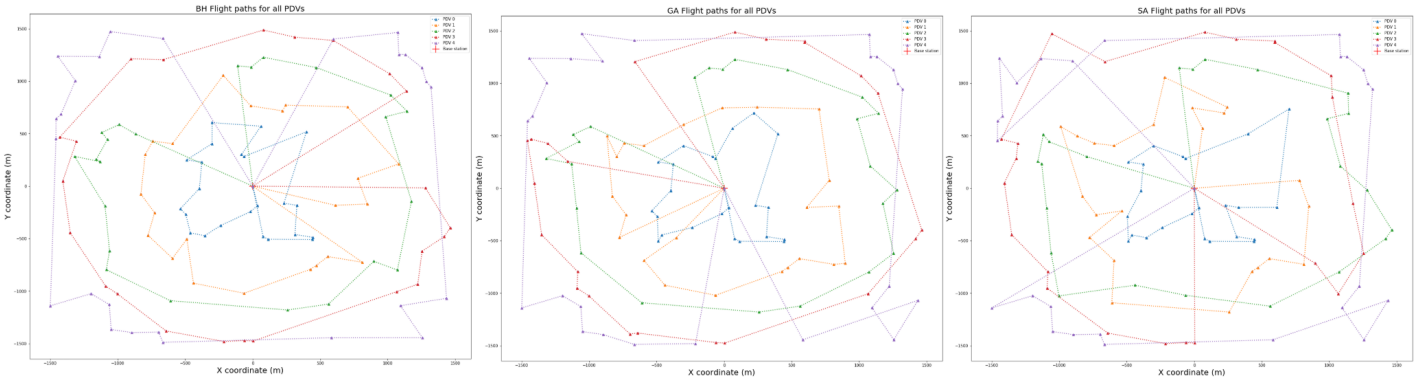


Fig. 5. Complete flight paths of algorithms (Case 6: $S_{squ} = 9km^2 \& N_{sn} = 1000$)

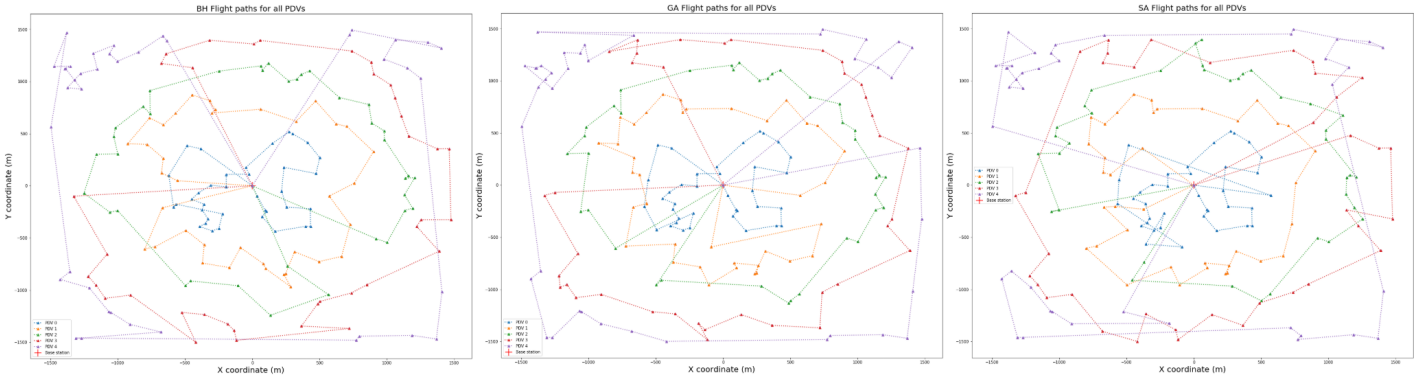


Fig. 6. Complete flight paths of algorithms (Case 9: $S_{squ} = 9km^2 \& N_{sn} = 1500$)

GA can have similar performance under some scenarios. The reasons may be due to, but not limited to:

- Appropriate initial solutions
- Requirement-oriented uniform fitness function
- Good global extreme value searching ability of BH

It is interesting to note that though SA works well in solving TSP, there is no strong evidence to prove its advantages of solving such complex optimization problem. Compared with other two algorithms, less modifications are applied to SA. For example, when generating the trail vector, only single random swap is performed. Such kind of simple modifications may lead to shallow searching, which may be the reason why SA shows poor performance in most tests.

V. Conclusion and Future Work

This paper provides an approach to evaluate optimization algorithm performance in solving UAV-based WRSN recharging scheduling problem, which is based on two-stage power distribution system. The 2-PDS can recharge SNs with both inductive and acoustic power transfer. The optimization objective is to maximize recharging energy and minimize PDV energy cost (or flight distance). Therefore,

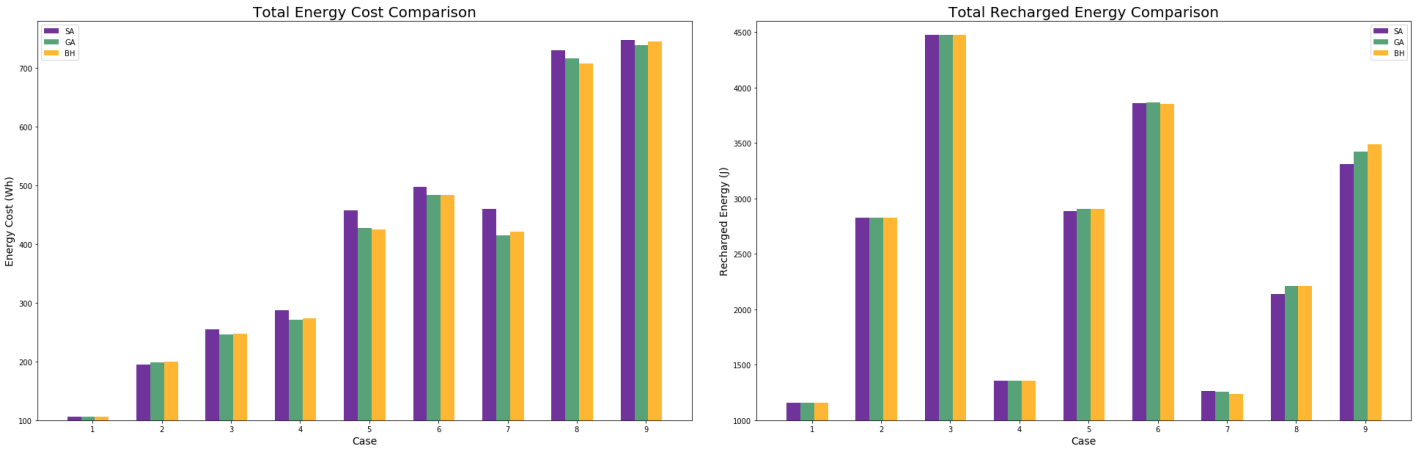


Fig. 7. Comparison of total PDV energy cost and total recharged energy

3 typical optimization algorithms (Genetic, Black Hole and Simulated Annealing) were implemented under same model to compare simulation results. As a result, this study successfully simulated the model of recharging scheduling of drones in WRSN under relatively ideal environments (i.e. no external interference, two-dimensional space, etc.). We tested algorithms under different scenarios, the summary of metrics with priority is shown as Fig. 7.

Through these tests, we found BH presenting better performance in most large-scale and some medium-scale networks. GA can have slight advantages in some tests. But considering randomness of evolutionary algorithms, those advantages cannot be seen as effective evidence. Moreover, it is proven that SA can generate meaningful solutions in some tests. But results of SA are not good in most cases, which may be caused by mismatched modifications.

Due to stated limitations, the system model can be improved further to simulate more realistic scenarios. For instance, simulating the model in three-dimensional space can accurate the calculation of energy change. Thus, additional parameter (i.e. altitude in Geographic Information System (GIS)) should be considered when performing IPT, flight, etc. Moreover, we assumed the PDV will reach exactly above the target SN (i.e. same coordinate) when performing UWB positioning. In 3D space, with consideration of latitude and altitude, calculation of the angle between UAV and the SN is needed to figure out the impact on IPT efficiency. The Super Capacitor (SC) of the PDV can be changed to other energy storage device (i.e. SC and Li-ion battery hybrid system (Peng, Shuhai, and Changjun 2017)) to enhance the energy density and power density simultaneously. But that will lead to different charge/discharge pattern, which will be a noteworthy experiment. In addition to the system model, the number of BSs and deployment of them can be optimized as well. In some specific SN deployments, flight distances of PDVs can be significantly reduced with linearly deployed BSs (i.e. PDV starts from BS 0 and returns at BS 1). Involving economic cost can further enhance contributions to practical projects. Improvements can be realized in software level as well. For example, all algorithms can be integrated as an ensemble system with user-defined parameter inputs. Users can access wanted data, visualization of results, PDV status, etc. online. As a solution of WRSN recharging scheduling problem, this study proposes the preliminary framework, which will be of future interest to improve.

Bibliography

Adewole, A. P., K. Otubamowo, and T. O. Egunjobi (2012). "A Comparative Study of Simulated Annealing and Genetic Algorithm for Solving the Travelling Salesman Problem". In: *International Journal of Applied Information Systems* 4, pp. 6–12.

Boyle, D. E. et al. (2019). "Inductive Power Delivery with Acoustic Distribution to Wireless Sensors". In: *2019 IEEE PELS Workshop on Emerging Technologies: Wireless Power Transfer (WoW)*, pp. 202–204.

Chen, Y. et al. (2016). "Modified central force optimization (MCFO) algorithm for 3D UAV path planning". In: *Neurocomputing* 171, pp. 878–888.

Cheng, R.-H., C.-J. Xu, and T.-K. Wu (2019). "A Genetic Approach to Solve the Emergent Charging Scheduling Problem Using Multiple Charging Vehicles for Wireless Rechargeable Sensor Networks". In: *Energies* 12, p. 287.

Dawes, B. (2020). *Boost Filesystem Library Version 3*. URL: https://www.boost.org/doc/libs/1_74_0/libs/filesystem/doc/index.htm.

Dji Matrice 100 User Manual (2016). V1.6. DJI Inc. URL: https://www.dji.com/hk/matrice100.

Hatamlou, A. (2018). "Solving travelling salesman problem using black hole algorithm". In: *Soft Computing* 22, pp. 8167–8175.

HEROTECH8 (2020). *Automated Drone Technology for Industry 4.0*. URL: https://www.herotech8.com/.

Kiziroglou, M.E. et al. (2017). "Acoustic power delivery to pipeline monitoring wireless sensors". In: *Ultrasonics* 77, pp. 54–60. ISSN: 0041-624X. DOI: https://doi.org/10.1016/j.ultras.2017.01.017. URL: http://www.sciencedirect.com/science/article/pii/S0041624X17300471.

Mitcheson, P. D. et al. (2017). "Energy-autonomous sensing systems using drones". In: *2017 IEEE SENSORS*, pp. 1–3.

NovaSensor NPA Surface Mount Pressure Sensors Datasheet (2019). Amphenol Advanced Sensors Inc. URL: https://www.amphenol-sensors.com/en/novasensor/pressuresensors/3161-npa-series.

Pandiyan, A. Y. S., D. E. Boyle, et al. (2019). "Optimal Dynamic Recharge Scheduling for Two Stage Wireless Power Transfer". In: *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS*. [Under Review].

Pandiyan, A. Y. S., M. E. Kiziroglou, et al. (2019). "Optimal Energy Management of Two Stage Energy Distribution Systems Using Clustering Algorithm". In: *2019 19th International Conference on Micro and Nanotechnology for Power Generation and Energy Conversion Applications (PowerMEMS)*, pp. 1–4.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Peng, X., Q. Shuhai, and X. Changjun (Feb. 2017). "A New Supercapacitor and Li-ion Battery Hybrid System for Electric Vehicle in ADVISOR". In: *Journal of Physics: Conference Series* 806, p. 012015. DOI: 10.1088/1742-6596/806/1/012015. URL: https://doi.org/10.1088%2F1742-6596%2F806%2F1%2F012015.

Pincus, M. (1970). "A Monte Carlo Method for the Approximate Solution of Certain Types of Constrained Optimization Problems". In: *Operations Research* 18.6, pp. 1225–1228. ISSN: 0030364X, 15265463. URL: http://www.jstor.org/stable/169420.

PowerStor: Supercapacitors PHB Series Technical Sheet 4402 (2011). Cooper Bussmman Inc.

Qin, Y., D. E. Boyle, and E. Yeatman (2019). "Efficient and Reliable Aerial Communication With Wireless Sensors". In: *IEEE Internet of Things Journal* 6.5, pp. 9000–9011.

Rao, Y. et al. (2017). "Practical deployment of an in-field wireless sensor network in date palm orchard". In: *Distributed Sensor Networks* 13, p. 5.

Soto, R. et al. (2018). "Adaptive Black Hole Algorithm for Solving the Set Covering Problem". In: *Mathematical Problems in Engineering* 2018.

Stark, I. (2006). "Invited Talk: Thermal Energy Harvesting with Thermo Life". In: *International Workshop on Wearable and Implantable Body Sensor Networks (BSN'06)*, pp. 19–22.

Yang, Y. (2017). "Massive IoT Monitoring System for the South-to-North Water Diversion Project in China". In: *IEEE PIMRC*.

Zorbas, D. and C. Douligeris (2018). "Computing optimal drone positions to wirelessly recharge IoT devices". In: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 628–633.