OPTIMIZATION OF ELECTRIC VEHICLE CHARGE SCHEDULING IN MULTIPLE PARKING LOTS

A Method Based on Metaheuristics and High Performance Computing

OPTIMISATION DE LA PLANIFICATION DE LA RECHARGE DES VÉHICULES ÉLECTRIQUES DANS PLUSIEURS STATIONNEMENTS

Une méthode basée sur les métaheuristiques et le calcul haute performance

A Thesis Submitted to the Division of Graduate Studies
of the Royal Military College of Canada
by

Katerina Olive Jean Brooks, BEng, rmc, P.Eng.
Major

In Partial Fulfillment of the Requirements for the Degree of
Master of Applied Science in Electrical Engineering

March 2024

Acknowledgments

I would like to express my gratitude to my supervisors, Dr. Mohammed Tarbouchi and Dr. Vincent Roberge, for their mentorship and guidance throughout my graduate studies. Their unwavering patience, continuous assistance, and insightful feedback on this thesis work has been crucial to its success.

I would also like to thank everyone in the Electrical and Computer Engineering Department Technical Support Staff, for their constant support both technical and moral.

Finally, I am especially grateful to my spouse Jess for his constant support and understanding throughout this entire two-year degree process, his assistance with troubleshooting, and for listening to every rant I ever had about testing runtimes. None of this would have been possible without him.

Abstract

The increase in global energy demand has led to the expansion, modernization, and digitization of power systems to incorporate new technologies like Electric Vehicles (EVs), which has created the potential for optimization of different aspects of the grid. The growing number of EVs requires system operators to balance the grid power supply against the electrical demand while reducing costs, which can be done through optimizing the charging schedule of EVs to manage their charging load. The EV charge scheduling optimization problem is complex and nondeterministic polynomial time-hard, generally involving a large amount of uncertainty in the variables and constraints. Because conventional mathematical and heuristic optimization techniques are not always appropriate for use with this problem, it may be considered a good candidate for use with metaheuristic optimization methods. Metaheuristics are a type of non-deterministic optimization algorithm, which have become a popular approach in EV optimization problems because they have been shown to provide improved performance with complex problems with many possible solutions and local optima, to escape local optima convergence, and to handle discrete variables. However, they require significant computational resources to compute the optimization for large problem sizes, and may not be calculated in an acceptable amount of time. Used in combination with a centralized optimization framework, they may produce more optimized results than decentralized models, but become unscalable with large power networks or number of EVs.

The aim of this thesis is to develop a solution for the problem of centralized EV charge scheduling with multiple parking lots using metaheuristics, to prove that it will find a more optimized solution to the charge scheduling problem, compared to decentralized optimization. High Performance Computing (HPC) techniques are used to combat the high resource requirements and long simulation time associated with the problem size, to increase the scalability of the solution, and to complete the optimization in an acceptable real-time interval. A parallelized centralized optimization model using two-level particle swarm optimization is designed and validated against its sequential version and a decentralized model on a HPC cluster, where it provides more optimal solutions than the decentralized model, computes solutions for scenarios with up to 27 parking lots in less than 15 minutes, and provides average computation speedups of up to 139 times faster. This research contributes to the development of centralized optimization solutions to EV charge scheduling optimization problems for multiple parking lots, which may have previously been considered intractable.

Keywords: Electric vehicle, charge scheduling, multiple parking lots, centralized optimization, particle swarm optimization, metaheuristics, two-level optimization, high performance computing, parallelization, multicore computing, distributed computing

Résumé

L'augmentation de la demande mondiale d'énergie a conduit à l'expansion, à la modernisation et à la numérisation des systèmes électriques pour intégrer de nouvelles technologies telles que les véhicules électriques (VE), ce qui a permis d'optimiser les différents aspects du réseau. Le nombre croissant de VE oblige les opérateurs de système à équilibrer l'alimentation électrique du réseau par rapport à la demande électrique tout en réduisant les coûts, ce qui peut être fait en optimisant l'horaire de recharge des VE pour gérer leur taux de recharge. Le problème d'optimisation de la planification des charges des VE est complexe et non déterministe en temps polynomial, impliquant généralement une grande quantité d'incertitude dans les variables et les contraintes. Les techniques d'optimisation mathématiques et heuristiques conventionnelles ne sont pas toujours appropriées pour une utilisation avec ce problème, ce qui en fait un bon candidat pour une utilisation avec les méthodes d'optimisation métaheuristiques. Les métaheuristiques sont un type d'algorithme d'optimisation non déterministe, qui sont devenues des approches populaires dans les problèmes d'optimisation des VE, car il a été démontré qu'elles offrent des performances améliorées pour des problèmes complexes avec de nombreuses solutions possibles et optima locaux, permettant d'échapper à la convergence vers des optima locaux et permettant aussi de manipuler des variables discrètes. Cependant, elles nécessitent des ressources de calcul importantes pour l'optimisation de problèmes de grande taille et peuvent ne pas être calculées dans un délai acceptable. Utilisées en combinaison avec un cadre d'optimisation centralisé, elles peuvent produire des résultats plus optimisés que les modèles décentralisés, mais deviennent non évolutives avec de grands réseaux électriques ou un grand nombre de VE.

L'objectif de cette thèse est de développer une solution au problème de planification centralisée de la recharge des VE avec plusieurs parcs de stationnement en utilisant des métaheuristiques, pour prouver que la méthode centralisée trouvera une solution plus optimisée au problème de planification de la recharge, par rapport à l'optimisation décentralisée. Les techniques de calcul haute performance (CHP) sont utilisées pour lutter contre les besoins élevés en ressources et le long temps de simulation associés à la taille du problème, pour augmenter l'évolutivité de la solution et pour terminer l'optimisation dans un intervalle en temps réel acceptable. Un modèle d'optimisation centralisé parallélisé utilisant l'optimisation par essaim de particules à deux niveaux est conçu et validé par rapport à sa version séquentielle et à un modèle décentralisé sur un cluster CHP. Ce modèle centralisé fournit des solutions plus optimales que le modèle décentralisé, calcule des solutions pour des scénarios avec jusqu'à 27 parcs de stationnement en moins de 15 minutes et offre des accélérations de calcul moyennes jusqu'à 139 fois plus rapides. Cette recherche contribue au développement de solutions d'optimisation centralisées pour résoudre les problèmes d'optimisation de la planification de la recharge des véhicules électriques pour plusieurs parcs de stationnement qui pouvaient auparavant être considérés comme insolubles.

Mots clés : Véhicule électrique, planification de la recharge, plusieurs parcs de stationnement, optimisation centralisée, optimisation par essaim de particules, métaheuristiques, optimisation à deux niveaux, calcul haute performance, parallélisation, calcul multicœur, calcul distribué

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviation	Meaning
Approx.	Approximately
Min.	Minutes
Max	Maximum
Min	Minimum
No.	Number
Ref.	Reference

List of Acronyms

Acronym	Meaning
A	Ampere
AC	Alternating Current
AIMMS	Advanced Interactive Multidimensional Modeling System
BEV	Battery Electric Vehicle
CHP	Calcul haute performance
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DC	Direct Current
EV	Electric Vehicle
G2V	Grid-to-Vehicle
GA	Genetic Algorithm
Gbps	Gigabits per second
GPU	Graphics Processing Unit
HPC	High Performance Computing
IL	Inner level
kW(h)	Kilowatt(-hour)
MPI	Message Passing Interface
MWh	Megawatt-hour
NP	Nondeterministic Polynomial Time
OL	Outer level
OpenMP	Open Multi-Processing
OPF	Optimal Power Flow
PHEV	Plug-in Hybrid Electric Vehicle
PSO	Particle Swarm Optimization
p.u.	Per unit
pWh	Picowatt hour
SAE	Society of Automotive Engineers
SPMD	Single Program, Multiple Data
TWh	Terawatt-hour

V	Voltage (in volts)
VE	Véhicule électrique
V2G	Vehicle-to-Grid

List of Symbols

Power Flow

Symbol	Meaning
b	Bus identifier
h	Branch identifier
i	Generator identifier
k	Bus identifier
$numB$	Number of buses
B_{bk}	Susceptance of the branch connecting bus b to bus k
G_{bk}	Conductance of the branch connecting bus b to bus k
P_{Db}	Real power demand at bus b
P_{Gb}	Real power generation at bus b
P_{Gi}	Real power of generator i
$P_{Gi,max}$	Maximum real power of generator i
$P_{Gi,min}$	Minimum real power of generator i
Q_{Db}	Reactive power demand at bus b
Q_{Gb}	Reactive power generation at bus b
Q_{Gi}	Reactive power of generator i
$Q_{Gi,max}$	Maximum reactive power of generator i
$Q_{Gi,min}$	Minimum reactive power of generator i
S_h	Apparent power in branch h
$S_{h,max}$	Maximum apparent power in branch h
$V_b\angle\delta_b$	Per-unit complex voltage at bus b
$V_{b,max}$	Maximum voltage magnitude at bus b
$V_{b,min}$	Minimum voltage magnitude at bus b
δ_b	Voltage angle at bus b
$\delta_{b,max}$	Maximum voltage angle at bus b
$\delta_{b,min}$	Minimum voltage angle at bus b

Particle Swarm Optimization

Symbol	Meaning
$\boldsymbol{b}$	Vector for best position previously occupied by PSO particle
c_1	Personal influence parameter
c_2	Social influence parameter
$\boldsymbol{g}$	Vector for best position previously occupied by any particle of a PSO swarm
n_{itr}	Number of PSO iterations
$n_{itr(IL)}$	Number of PSO iterations of the inner level PSO
$n_{itr(OL)}$	Number of PSO iterations of the outer level PSO
n_p	Number of PSO particles
$n_{p(IL)}$	Number of PSO particles of the inner level PSO
$n_{p(OL)}$	Number of PSO particles of the outer level PSO
$\boldsymbol{r_1}$	Vector of random values between 0 and 1
$\boldsymbol{r_2}$	Vector of random values between 0 and 1
$\boldsymbol{v}$	Velocity vector of a PSO particle
v_{max}	Maximum velocity
$\boldsymbol{x}$	Position vector of a PSO particle
x_i	Position of i^{th} PSO particle
ω	Inertia

High Performance Computing

Symbol	Meaning
f	proportion of the algorithm which can be parallelized
n	Number of processors
s	Speedup
s_n	Speedup using n processors
s'	Scaled speedup
T_{par}	Parallel algorithm runtime
T_{par_n}	Parallel algorithm runtime using n processors

T_{seq}	Sequential algorithm runtime
T_{seq_n}	Sequential algorithm runtime using n processors

EV Charge Scheduling

Symbol	Meaning
arr_i	EV i arrival time to the parking lot
b	Bus from set of buses B
d	Penalty value
d_{dem}	Unmet demand (penalty)
d_{limChr}	Charging limit violation (penalty)
d_{limTf}	Parking lot transformer limit violation (penalty)
$d_{normalized}$	Normalized penalty value
d_v	Voltage magnitude limit violation (penalty)
d_ρ	Multi-parking lot penalty term
${dd_{limChr}}_i^t$	One EV's demand in one time interval exceeding the charging rate limit (penalty)
${dd_{limTf}}^t$	All EVs demand in one time interval exceeding the transformer limit (penalty)
dd_{v_b}	One bus's voltage magnitude exceeding or subceeding the voltage magnitude limit (penalty)
${dd_\rho}^p$	One parking lot's penalty term
dem_i	EV i total charging demand (desired consumption) (kWh)
$demu_p$	Parking lot p total unmet charging demand (kWh)
dep_i	EV i departure time from the parking lot
i	Index of EV_i in set EV
j	Algorithm pseudocode iteration index
k	Algorithm pseudocode iteration index
$limChr$	Charging station port charging rate limit (kW)
$\boldsymbol{limTf}$	Vector of solution variables for the set of all transformer limits across all parking lots (kW)
$limTf_{base}$	Base transformer limit for one parking lot (kW)
$limTf_p$	Transformer limit of parking lot p (kW)

n_{pl}	Number of parking lots in set P_l
$numB$	Number of buses
$numEV$	Number of EVs
$numT$	Number of time intervals
$numX$	Number of solution variables
p	Parking lot in set P_l
t	Time interval in set T
x_i^t	proportional charging demand quantity from $\boldsymbol{X}$
${x_d}_i^t$	the decoded proportional charging demand quantities x_i^t (kWh)
A_i^t	availability to charge of EV_i in a parking lot at time t
B	Set of distribution system buses
C	Cost (\$)
C_{OL}	Cost (outer PSO) (\$)
C_p	Cost of electricity for parking lot p (\$)
$ElecPrice$	Set of Electricity Prices
$ElecPrice_t$	Electricity Price at time interval t (\$)
EV	Set of EVs
$F(\boldsymbol{limTf})$	Fitness function of solution vector $\boldsymbol{limTf}$
$F(\boldsymbol{X})$	Fitness function of solution vector $\boldsymbol{X}$
P_l	Set of parking lots
P_{total}	Total power available to aggregator
T	Set of time intervals
$\boldsymbol{X}$	Vector of solution variables
V_b	Voltage magnitude at bus b
α	Normalization range minimum
β	Normalization range maximum
ρ	Penalty term
ρ_{OL}	Penalty term (outer PSO)
ρ_p	Penalty term of parking lot p
Δt	Duration of time interval t
$\forall$	For all

$\in$	Element of
$\wedge$	And
$\vee$	Or

Other

Symbol	Meaning
N	MATLAB std function number of observations for standard deviation normalization [1]

Section 1

Introduction

There is an increasing demand for energy and electricity across the globe, which requires the modernization, digitization, and expansion of current power systems, along with the development of new technologies [2], [3]. Within the transportation sector, electric vehicles (EVs) drive a large portion of the current and future demand for electricity, currently consuming approximately 100 terawatt-hours (TWh) per year [2]. Due to governmental policies and vehicle manufacturer initiatives [2], [4], [5], EVs are beginning to dominate the market [6], [7], with the number of EVs in active use increasing every year [8] and expecting to reach a global electricity demand of over 380 TWh by 2032 [2], [7].

The electricity grid has been required to modernize and automate in order to adapt to the increasing demands for electricity from the increasing numbers of EVs, as well as new technologies like high levels of renewable energy sources, distributed energy resources, and distributed generation [9]. While the modernization may include physical upgrades to grid infrastructure in some cases, the optimization of its current operations is a more cost-effective solution to integrate the new technologies and shift the peak power demand [10], to find the most effective, optimal way to operate the system for the lowest cost [11], [12], [13], [14]. The minimization of costs and balancing of the grid's power supply with the electrical demand posed by EVs can be done through management of the charging load of the EVs [15], by optimizing the charging schedule of EVs.

This thesis is concerned with that optimization of EV charge scheduling, specifically optimizing the charging schedules of EVs in multiple parking lots. Since the EV charge scheduling optimization problem is complex and nondeterministic polynomial time (NP)-hard, generally involving a large amount of uncertainty in the variables and constraints, it is a good candidate for use with metaheuristic optimization methods [16].

Metaheuristics are non-deterministic optimization methods, which work by iteratively improving one or several candidate solutions to a problem to converge towards the global optimal solution [17], [18]. They are problem-independent, and can be applied to complex non-linear, non-convex and non-differentiable problems, which is the case for power optimization problems involving EVs [7], [18], [19], [20]. While metaheuristics have become a popular approach in EV and power systems optimization literature [7], because of their "ability to produce near-optimal results in a computationally efficient manner" [20], the high complexity of the problems, long simulation times, and requirement for significant computational resources to compute the problem output [21] have generally limited their use in power systems optimization, either to smaller networks or to problems which have longer time horizons [7], [18], [21], [22], [23]. This has led to the use of High Performance Computing (HPC) techniques to reduce computation times [24], [25], [26].

HPC refers to the use of powerful processors in parallel to solve complex problems and process large datasets at high speeds. HPC has been used in many EV-related optimization problems [26], [27], [28], [29], [30], and while the problem of EV charge scheduling has been identified as a "major research problem" [15] within power systems optimization, and has been approached using metaheuristics many times [7], [15], the literature at present is limited in the integration of both metaheuristics and HPC for optimal charge scheduling, with few papers having used any form of parallelization to improve the execution time of their simulations [23], [24], [25]. The development and use of an EV charge scheduling optimization method using both metaheuristics and HPC within this thesis will be shown to provide a more optimized solution while maintaining an acceptable execution time.

1.1 Motivation

The motivation to pursue the subject of the optimization of EV charge scheduling for EVs in multiple parking lots is twofold: the increasing number of EVs requires a framework to balance their demand against the grid power supply, and the optimization of that charge scheduling framework can lead to cost savings for utilities and consumers, with even a small percentage improvement in cost reduction equating to a large amount of savings.

As the number of EVs in use around the world increases, their load demand on the power system will increase [2], [5], [7], due to their requirement to charge and re-charge continuously during their lifetime. These loads impact the local grid by affecting the electricity generation (i.e., is the supply adequate), transformers (potentially accelerating their aging and overloading), distribution system power and voltage quality, and potentially increasing the peak load demand with uncontrolled charging [6], [10], [20]. Uncontrolled charging introduces voltage instability to the grid [31], so one of the main goals of EV optimization is to determine their optimal coordinated charge scheduling and management to reduce peak power draw on the distribution system and balance that demand with the supply [15], [26], [31].

The management of EV charging can be done through charge scheduling, which refers to the assignment of specific time slots to EVs for charging [20]. The optimal scheduling [7] of EV charging can be used to "control charging activity at charging stations to minimize unexpected spikes in peak load demand" [15], respect local grid constraints, and prevent blackouts [20], [32]. Optimization of charge scheduling, when performed taking local power system loading and other requirements and constraints into consideration, can ensure that the burden on local systems is minimized (for example, in the case of fast charging, degrading the power quality in a local system when large amounts of power are drawn over a short period of time [7]), and can also provide “a smoother and hassle-free transition from conventional transportation methods to the electrical realm” [7], a state desired by many governments and agencies [5]. This means that the problem of optimization of EV charge scheduling is relevant to the current state of the industry.

The potential for cost savings is another major motivator of optimization, with the optimization of power systems having long been focused on finding the most effective, optimal way to operate the system for the lowest cost [11], [12], [13], [14]. Optimizing the operation of EVs through their charge scheduling will reduce energy consumption and improve energy management [7], ultimately balancing the demand on the grid and reducing costs for operators and consumers. In Ontario Canada, where EVs are expected to add approximately 6.81 TWh of annual energy demand by 2030 [33], a 1% reduction in demand to 6.129 TWh would provide total annual consumer savings of $32,224,630 based on an hourly energy price of $47.32 per MWh [33].

1.2 Statement of Deficiency

In literature on the EV charge scheduling optimization problem, there are two main approaches to developing a coordinated EV charge scheduling solution: centralized and decentralized approaches [7], [15], [34], [35]. Centralized approaches have been recognized as providing more optimal solutions [10], [34], [36], but lacking scalability and becoming computationally intractable to compute in real-time as the problem size increases [15], [34], [35], [37]. Decentralized approaches have therefore attracted interest due to their scalability, being better able to address larger problem sizes with lower computational complexity and in a shorter time [10], [34], however they do not always provide the more optimal charging solution which respects all power system constraints [34], [35], [38]. The computational resource requirement and longer execution time for larger optimization problems has made centralized solutions less attractive for the EV charge scheduling optimization problem.

In terms of the specific optimization methods used with the EV charge scheduling problem, metaheuristics have become a popular and fitting method, due to their ability to tackle complex, “non-convex and non-linear” [15] EV objective functions which have a large number of possible solutions and local optima [7], [15]. However, metaheuristics also have the drawbacks of requiring larger computational resources and longer computational times [7], [20].

A solution to the EV charge scheduling optimization problem which addresses the higher computational resource requirements of both the centralized and metaheuristic optimization methods, would likely result in a solution that is more optimized, more scalable, and importantly, able to be computed in an acceptable amount of time. The use of HPC techniques like parallelization has been recognized as a potential method to address these deficiencies or limitations [39], though few papers have used such techniques at this time [24], [25], [26]. The metaheuristic Particle Swarm Optimization (PSO) [25] and PSO with Mixed-Integer Linear Programming [24] were used with distributed and multicore computing for day-ahead resource and EV scheduling, and the Alternating Direction Method of Multipliers algorithm [26] was used with multicore computing for an energy management system with EV charge schedule optimization.

1.3 Aim

The aim of this thesis is to address the deficiencies in current literature and to prove the following research hypotheses:

1. Centralized EV charge scheduling optimization for EVs in multiple parking lots will find a more optimized solution to the charge scheduling problem, compared to decentralized optimization.
2. Calculations for the centralized EV charge scheduling optimization algorithm can be parallelized on a HPC system to allow for real-time optimization.

A “more optimized solution” in this case refers to a reduced (lower) cost. “Real-time” in this case refers to a 15 minute time limit for the optimization calculations.

1.3.2 Success Criteria

The following success criteria will be used to determine if the aim has been met:

1. Comparison of the results of at least three scenarios of different sizes (i.e., different network sizes, number of EVs, and parking lot locations), by fitness function metrics (i.e., total cost), between the centralized (parallelized) and decentralized algorithm (parallelized) models or methods. This is to determine if the algorithms are performing as intended, and to determine which method can produce the more optimal solution. The single parking lot scenario will be compared against the originating single parking lot model found in [40] to confirm that it can find results similar to those found by the authors of that paper. The centralized and decentralized methods will be compared against each other.
2. Comparison of the results of the single parking lot scenario, between the centralized sequential and parallelized methods. This is to determine the accuracy, or correctness, of the parallelization.
3. Comparison of the runtimes of the multi-parking lot scenarios, between the centralized sequential and parallelized models, for different problem sizes. This is to determine the speedup between the models for a variety of problem sizes, confirm that the model can continue to perform as intended as the problem size

increases, and to determine the potential maximum problem size that can be handled by the model in the real-time limit. Full results for the multi-parking lot scenarios will not be compared due to the long runtimes required for the sequential versions.

1.4 Research Activities

The following activities were conducted in order to prove the research hypotheses stated above and compare against the success criteria:

1. Development of the single parking lot optimization model. This consisted of the selection of a suitable metaheuristic-based single parking lot EV charge scheduling optimization algorithm and associated simulation parameters from literature, the development of a single metaheuristic-based algorithm to optimize a single parking lot based on this method from literature, and verification to ensure that the developed model performed as expected, i.e. was able to achieve similar results to that model from literature when using the same simulation parameters.

2. Development of the multiple parking lot optimization models. This consisted of the development of the centralized and decentralized optimization models, as well as suitable simulation parameters, for the optimization of multiple parking lots, based on the single parking lot optimization model developed in the previous activity.

3. Parallelization of the single and multiple parking lot optimization models. This consisted of taking the algorithms developed in the previous activities, and parallelizing the code to enable them to run on a HPC cluster using multicore and distributed computing, in MATLAB®.

4. Validation of the single and multiple parking lot optimization models. This final phase consisted of evaluating the simulation results of all the algorithms and associated models (single or multiple parking lot, centralized or decentralized, sequential or parallelized) against the criteria listed in Section 1.3.2. Five different parking lot scenarios with four different distribution system test cases were used to demonstrate the behaviour of the algorithms and associated models in a variety of situations, and produce the metrics necessary to assess the performance of the algorithms against the criteria and determine if the aim of this thesis has been achieved.

1.5 Summary of Results

Four main tests with over fourteen individual sub-tests were conducted using MATLAB® in order to determine the optimized costs, optimized parking lot charge schedules, and speedup (if relevant) for the parallelized and sequential single parking lot optimization algorithm, the parallelized and sequential centralized multiple parking lot optimization algorithm, and the parallelized decentralized multiple parking lot optimization algorithm.

The results indicated relationships between the number of parking lots, the metaheuristic parameters, the cost, the feasibility of solutions, the total runtime, and the speedup gained through parallelization. The single parking lot optimization method was able to produce results as good as those using the same single metaheuristic in the paper its method was based on, by Wu et al. [40], and nearly as good as the best cost result produced by that paper's proposed model. The parallelized single parking lot optimization method was able to produce an average speedup of 1.18 times when using multicore computing. On average, the centralized multiple parking lot optimization method was able to provide more optimal solutions with lower costs than the decentralized method for all parking lot scenarios. In each scenario, a single parking lot contained charging ports for up to 20 EVs. The parallelized centralized multiple parking lot optimization method was able to produce average speedups of 73.95 to 139.44 times when using distributed and multicore computing. The maximum problem size that can be handled by the model in the real-time limit of 15 minutes depends on the number of inner PSO iterations: the parallelized centralized multiple parking lot optimization method with 500 iterations was able to optimize 9 parking lot charging schedules, though it can be extrapolated that it should be able to optimize the charge schedules for up to 15 parking lots in under 15 minutes. The centralized algorithm with 250 inner PSO iterations was able to optimize the 27 parking lot scenario, and should be able to optimize the schedules for up to 32 parking lots in under 15 minutes.

These results provided four main contributions, to add new knowledge to the field of real-time centralized EV charge scheduling optimization, to reinforce the assumption of the more optimal outcomes of centralized algorithms, and to provide a foundation for future work in centralized EV charge scheduling optimization and algorithm parallelization to enable real-time optimization. These contributions are:

1. Development of a complete centralized solution to the problem of EV charge scheduling optimization in multiple parking lots, which included the use of a two-level PSO and verification of power flow constraints in the system.
2. Verification that centralized optimization provides a more optimal solution than decentralized optimization for the optimization problem, with direct comparisons being made between the two multiple-parking lot optimization models for a variety of problem sizes.

3. Provision of a method to parallelize the two-level PSO centralized optimization model.

4. Demonstration of the scalability of the solution, and verification that using parallel computing on HPC systems can allow for real-time optimization of the problem for a variety of problem sizes.

1.6 Organization

The remaining chapters will provide further details about this area of research and the conduct of the research activities for this thesis. Section 2 provides a literature review and background on the area of research, providing more information on the topic of power systems, EVs, charging, optimization, metaheuristics including PSO, and HPC. Section 3 describes the methodology and design of the optimization algorithms, including the development of the single parking lot optimization algorithm, the centralized and decentralized multiple parking lot optimization algorithms, their parallelization, selection of the simulation parameters, creation of the parking lot profiles, and selection of the distribution system test cases used to make the testing scenarios. Section 4 describes the results of the tests or validation activities, comparing the results against the success criteria. Section 5 outlines the contributions of this thesis, potential areas for future work in the field, and provides recommendations for that future work. Section 6 provides a general conclusion of this thesis document.

Section 2
Background and Literature Survey

This chapter provides a literature review as well as background information on the topics related to this thesis, including power systems and optimization, EVs and EV charging problems, optimization methods, metaheuristics, and HPC in Sections 2.1-2.5. Five papers related to this thesis are summarized and their influence on this thesis described in Section 2.6. Additionally, information on the parallelization and power flow software tools used in this thesis is described in Section 2.7 and Section 2.8.

2.1 Power Systems

As EVs increase in popularity and number, they result in an increasing significant variable load, and resulting power demand, on the grid [20], and therefore an increasing need for research related to the topic of EV integration into these power systems.

When the word "grid" is used in this context and throughout this document, also referred to as the power or electricity grid, network, or system, it refers to a system which is generally defined as a network of components and their associated equipment, which generates electrical energy and transports it from its sources to its loads or users via power transmission lines [41], [42]. There are three main components which form the basis of all modern power systems or power grids: the generation system, the transmission system, and the distribution system, which provides the power to consumers and industries [42], [43]. These components may form systems or sub-systems unto themselves, depending on the size of the power system or network.

Historically, transmission and distribution systems were only designed to serve peak power demand requirements, and so did not require any real-time management of the system. However, the emergence of EVs and general electrification of the transportation sector has brought to light the importance of the modernization, automation, and improvement of the grid [9], in order to expand capacity, improve system operational security limits, and ensure sustained power reliability [44]. Per the authors of [32], "uncontrolled or un-coordinated charging of EVs will not only degrade the power grid capability but will also affect the distribution facility of the overall power structure. If the electrification of transportation is not properly optimized, this will ultimately lead to the collapse of [the] existing power structure." While physical upgrades to grid infrastructure may be required in some cases, the total upgrade of the power grid is generally considered to be unrealistic [10], and the elimination or postponement of "expensive investments on grid infrastructure" [44] through investigating into EV charging control, charging station placement, demand scheduling, and optimization, has been favoured by system operators and utilities [44]. In order to understand how EV charging and optimization can affect the grid, one must be familiar with its three main components, as well as emerging topics in literature like smart grids and microgrids.

2.1.1 Power System Concepts

Generation systems are sources of electrical energy or power, which supply the energy via the use of conventional or renewable energy to turn a turbine rotor or generate current, or via the supply of power from energy storage systems or other types of fuel cells [43], [45]. Each power system will have one or more generating units, which provide direct current (DC) or alternating current (AC) power depending on their power source [43].

Transmission systems transport electrical energy or power from the generation system sources to the system loads, often at a high voltage level [41], [43], though often the transmission system of a power system is considered to end at the distribution electrical substation, where power is transferred from the transmission system to the distribution system. Transmission systems provide users with the ability to draw power from multiple different generating units [46].

Distribution systems distribute the power transmitted to the distribution substations to commercial and residential customers. Distribution systems contain a number of sub- and related topics which play an important role in the system, and are also emerging and popular topics within power system research, such as distributed generation, and EVs [47]. Distributed generation is generation that is connected directly to the distribution system, rather than the transmission system, and usually consists of one or more smaller-scale power generation systems which are located much closer to the load, reducing the demand on conventional power grid generation and transmission systems [46]. One of the largest-growing types of distributed generation is renewable energy generation, specifically wind and solar power generation [48]. There has been a large amount of research with has looked at the integration of EVs with these topics, such as the integration of solar panels at EV charging facilities [49], [50], and the treatment of EVs as large energy storage systems within a smart grid [20], [24], [25], [51].

Smart Grids and microgrids are major topics or concepts in recent power systems research, which also have some overlap with the topic of distribution systems due to their integration with distributed generation technologies. In general terms, a smart grid is a modernized, digitized version of the conventional power grid, which uses digital technology and computer processing to facilitate two-way communication and power flow between the power utility and the consumers [14], as well as the integration of automated sensors to provide updated, real-time information on the status of the system [43], [52]. The aim of a smart grid is to gather and act on its collected information to improve the reliability and efficiency of production, transmission, and distribution of power, to detect and react quickly to disturbances, to reduce costs for utilities and consumers, to integrate renewable energy sources, and to improve the overall security of the power system [14], [43], [52]. A noted benefit of smart grids is their ability to support the increasing deployment of EVs, especially those capable of supporting Vehicle-to-Grid (V2G) charging and discharging of EVs when functioning as variable loads and energy storage systems [24], [53]. The authors of [14] state that a smart grid may be considered to be composed of integrated smart microgrids.

Microgrids are a collection of distributed energy resources, loads, and energy storage systems which operate together as a "single controllable entity" which can operate in grid-connected or disconnected (islanded) mode [54], and which can be used to improve integration and effective use of distributed generation at the distribution level [14].

2.1.2 Power System Optimization

According to [55], economic dispatch, power flow, and optimal power flow (OPF) are three main types of problems encountered in power systems literature and research, and while they have been used and solved since the 1930s, they have evolved over time to match the constant evolution of mathematical optimization techniques and computing power since the mid-twentieth century [11].

The Economic Dispatch problem aims to find the lowest possible cost of generation dispatch to serve a specified load demand [43], [55]. The Power Flow (also called load flow) problem refers to the equations used to determine the current, voltages, and real and reactive power flows at each bus in a system, and therefore across the generation, load, and transmission networks, in response to specified set of generator power output and load demands [43], [55].

The OPF problem was the first to be fully formulated by Carpentier in 1962 [14], [55], [56]. The aim of OPF is to find the optimal solution to a given non-linear objective function, which has been developed to find optimal control variable values (such as real and/or reactive power generation level, generation cost, phase shifter angles, load shedding values, voltage settings, transformer tap settings, etc.) or objectives like active power cost or line loss minimization, and is constrained by any number of characteristics, such as minimum or maximum voltage levels, switching limits, generator minimum outputs, etc. [11], [14], [43], [55].

The OPF problem is one of the most well-known optimization problems in generation and transmission systems, with a wide variety of approaches proposed to solve the problem over the years [11], [18]. While the basic problem has not necessarily changed since the 1960s and it remains mathematically complex, its applications have expanded and it has become required to be computed at increasing speeds, which have necessitated the development of further algorithms and advanced computing methods to increase the speed of its computations [11], [55].

In general, there are a number of optimization problems found in all aspects of power systems, smart grids, and microgrids. Some optimization problems are related to those discussed above, like the minimization of line loss within the system, as found in in [57] and [58]. Increasingly, problems involve elements of distributed generation and other new technologies, and are mostly based on the configuration of the system and the positioning of components, with problems objectives including the optimal reconfiguration of distribution networks, the optimal placement, sizing, and control of electrical units, and the optimal management of EV charging.

The solutions found in literature are generally concerned with ensuring optimal power delivery and minimal power loss within a distribution system while compensating for the intermittent power generation from renewable energy sources, improving power quality, ensuring voltage stability, and in the case of EVs, reducing the voltage fluctuations and impact to the grid caused by their uncontrolled charging through determining optimal charging schedules, controlling charging costs to users, and through the management of V2G services [20], [43], [59].

Optimization tasks within the topic of smart grid are generally the same as those used when optimizing the conventional power grid: OPF-based optimization of the operation of the grid, scheduling to ensure the grid's current resources meet the demand, and planning to ensure the future state of the grid will meet expected power demands. The optimization aims are also generally similar to those discussed above, including the minimization of active power generation and operation costs, minimization of power losses, and improvement of grid resiliency [14], [22].

EVs have been involved with numerous aspects of power systems optimization, and will continue to be important as power system operators and governments stive to meet the growing electrification demands of the world [2].

2.2 EV Optimization Problems

The specific optimization problems concerned with EV operation and integration into the power grid are numerous. However, before discussing these optimization problems in detail, the term EV must be defined. According to [60], an EV is any vehicle that is or can be powered by an electric motor which draws electricity from a battery, and can be charged from an external source (i.e., through plugging it into an electrical outlet). This definition encompasses both Battery Electric Vehicles (BEV), which are only powered by batteries, and Plug-in Hybrid Electric Vehicles (PHEV), which can be powered by batteries or the vehicle's internal combustion engine [20], [60]. Both EV types have the capability to connect (plug-in) to the power grid.

The optimization problems concerned with EV operation and integration into the power grid generally involve a large amount of uncertainty in their variables and constraints due to their behaviour as variable loads. EVs present a large, stochastic load demand to the grid, with stochastic charging locations and behaviours impacting where that load is present on the grid. With V2G technology, EVs can stabilise grid voltage and act as mobile energy storage systems, providing similar benefits. The large-scale penetration of EVs will impact the function and reliability of the grid based on a multitude of factors, including driving characteristics, charging characteristics, charge timing, battery capacities, battery age, charging processes, and penetration level [27], [61].

According to the authors of [7], [15], there are six major optimization problems concerning EVs, with some overlap between the topics depending on the author: EV routing, EV charge scheduling, EV charging station placement and sizing, energy/load management, EV design and manufacture, and EV control. According to [7], the EV energy management and charge scheduling problems are the two most popular topics in recent literature, with hundreds of papers published in those topics in the last five years. The six major optimization problems are as follows:

1. EV routing: Since EVs have a range limited by their need to charge, the EV routing problem is concerned with selecting a least-cost route that would serve a set of customers, taking into account numerous factors like vehicle load capacity, customer time windows, working hours, the time needed to travel to and charge at a charging station, and recharging times [7], [15], [62].

2. EV charge scheduling: this problem is concerned with generating an optimized charging schedule for all EVs within a system, generally managing the schedules to achieve a goal, like reducing cost, reducing the peak power draw on the distribution system, and balancing demand with supply [15], [26], [31]. This can be done through the use of electricity pricing signals and other demand response techniques, however the large amount of uncertainties and unknown system parameters like EV driving profiles, RES DG production, and electricity price, along with the requirement for a secure communications infrastructure for utilities, vehicles, and customers, and high computational resources make the estimation or prediction of power demand, and therefore the optimization problems, very difficult [26], [31], [61].

3. EV charging station placement and sizing: these problems are concerned with identifying a location for a charging station for each EV in a region to minimize the total distance traveled to charge and reduce its impact on the local grid, and determining the number and size of those charging stations based on the number of EVs in a region and their average demand [7], [15].

4. Energy or load management problem: this problem has a large amount of overlap with the EV charge scheduling problem, as it is defined in [15] as being

concerned with the balancing of EV electrical demand with the power supplied by the grid. Whereas the EV charge scheduling problem is mostly focused on the scheduling of EV charging with respect to grid constraints, the energy management problem involves the integration of distributed generation and use of EVs as energy storage systems with bidirectional V2G charging and discharging to maximize "strategic utilization of energy sources in a smart grid" [20].

5. EV design and manufacture: these problems are generally concerned with optimizing different mechanical or electrical systems within an EV, such as engine, motor, battery, and wheel component sizing, motor design and optimization, and transmission system optimization, and often aims to influence energy management and lower fuel consumption of hybrid vehicles [7].

6. EV control optimization: this problem is concerned with improving the performance of EV through its control systems, specifically with its control parameters, control strategy, transmission, torque, braking, and other components [7].

2.3 EV Charge Scheduling

As stated in Section 2.2, the EV charge scheduling problem is concerned with generating an optimized charging schedule for all EVs within a system. Coordinated EV charging, through optimal charge scheduling, has been recognized as an "effective and cost-efficient way to mitigate the charging stress on power systems" [63], and remains a popular and relevant topic within EV and power system optimization research. The problem has been identified as being NP-hard [63], with "no known accurate algorithms for finding optimal solutions in polynomial time" [15], and non-convex due to the high number of potential variables like number of EVs and random arrival and departure times [37], however optimization solutions in literature have been able to achieve optimality or near-optimality [34].

The following subsections will provide more detail about this and related sub-topics, including the technologies involved with EV charging, the use of centralized and decentralized approaches to optimization, the use of metaheuristics and other optimization methods, integration with power flow, inclusion of multiple parking lots, and problem abstraction.

2.3.1 EV Charging Technologies

2.3.1.1 *Vehicle-to-Grid*

Within the literature, the charging of plug-in EVs is generally studied in the context of controlled or uncontrolled unidirectional charging (sometimes referred to as Grid-to-Vehicle (G2V), unidirectional V2G charging, or one-way smart charging), and bi-directional charging, including V2G technologies [64]. While some sources specify that V2G can refer to the unidirectional or bidirectional control and management of EVs [20], [65], it is most commonly used to refer to bidirectional power flow technologies where the EV functions as a distributed and variable energy storage system in the smart grid [25], [53], [64], and can be used to provide ancillary services like frequency regulation, voltage regulation, reactive power compensation, congestion management, and improvement of power quality [20], [64].

2.3.1.2 *EV Charging Stations*

An EV charging station is defined by [60] as a location with one or more EV charging ports (each port can supply charge to one EV) at the same location. This definition encompasses many locations, including utility and user-installed public, commercial, and private parking and charging facilities, including parking lots [6], [44]. In this thesis, a parking lot is the same as an EV charging station.

There are three charging levels which can be supported by a charging port: a summary of the three charging levels is shown in Table 2.1 [6], [35], [60], [64], [66], [67]. Since Levels 2 and 3 are the fastest, most commercial public charging stations having ports which support Level 2 and 3 charging [60], [66], [68], [69]. These levels have the highest power draw, and therefore have a larger impact on the local grid. The more EVs in operation, the more charging stations that will be required in public parking lots and other locations, the higher the power draw, and therefore the larger the impact from EV charging at these levels. A framework to manage the charging of EVs at these locations through optimal charge scheduling would assist in mitigating the negative effects that these charging stations may have on the grid.

Table 2.1 Summary of EV Charging Levels (North American Standards)

Level	Charging Power (kW)	Voltage Supply (V)	Current Rating (A)	Connectors	Speed	Use	Advantages	Drawbacks
1	1.44-1.9	120 V AC	12-16	SAE J1772	3-8 km per hour of charge	Domestic	Home charging. Lowest impact on distribution systems.	Slowest charging time.
2	Up to 19.2	208/240 V AC	80	SAE J1772 Tesla	16-97 km per hour of charge	Domestic Public	Shorter charging time.	May require dedicated supply equipment.
3	20-120+	480 V DC	80-200	Combined Charging System CHAdeMO Tesla	Full charge in approx. 30 mins-1 hour	Public	Shortest charging time. Most energy efficient.	Extremely high currents and voltages. High impact on utility. May trigger extreme surges in grid demand. High installation cost.

2.3.2 Centralized and Decentralized Approaches

While there are numerous coordinated charging (also called smart charging) algorithms developed by researchers since at least the mid-2000s [34], most current researchers agree that there are two main approaches to developing a coordinated EV charge scheduling framework: centralized, and decentralized approaches (or architectures) [7], [15], [34], [35].

In both approaches, the EV charging activities are coordinated by a central aggregator, which is the link between the distribution network operator and the EV customers that ensures that the EV charging demand is met while respecting network limitations and constraints [15], [35]. Depending on the objective of the aggregator, it may act to obtain a desired grid (utility or operator) or customer objective, as well as provide ancillary support to the grid [6], [35]. In a real scenario, the aggregator of a network may be a network operator, utility, EV charging facility, EV fleet operator, or a communication device located on a distribution network transformer [34].

In the centralized approach, the aggregator alone is responsible to collect all parameter information and perform the required calculations to determine the charging schedule of each EV. It collects a large amount of information about the network, EV charge requirements, and more, processes it at a single point, and communicates the optimized charging schedules back to each EV. In the decentralized approach, also called the distributed approach, the information and calculations are performed by several different entities which obtain different parameters from each other through communication. In the classic decentralized approach, the computational load of the optimization scheme is shared across EVs, and each EV acts as an independent decision-maker or optimizer for its own charge scheduling problem. In the hierarchical decentralized approach, the computational load is shared between EVs and one or more aggregators through a "tree-like communication topology" [34], where each aggregator coordinates the schedules for its group of EVs, and may also provide parameter information to the other aggregators.

Both approaches require a number of smart grid infrastructure elements to be present in order to be conducted, namely the availability of bidirectional communication between EV customers and EV aggregators so they can share their charging demand information in advance [10]. As well, centralized charging requires that EV customers have relinquished control of their EV charging process and schedule to the aggregator, which can remotely control the charging infrastructure [10], [34].

While it does require this infrastructure to be in place, the centralized approach offers one main benefit compared to the decentralized approach: it can provide better solutions for optimizing the energy management of grids and EVs, having a higher possibility of finding the globally optimal solution to the problem, because it collects complete information about the system it is connected to and is able to consider more system states and constraints in its problem formulation [10], [34], [36], [70]. However, it has a number of drawbacks, including that it generates a single point of failure at the aggregator, requires a sophisticated communications infrastructure with redundancies, raises privacy and security concerns with respect to the amount of information, has a large control overhead, is computationally complex, and has a higher computational cost due to its use of a larger amount of data to provide a better solution [15], [32], [34], [35], [36], [71]. As stated by [34], the key challenge or drawback for centralized approaches is the lack of scalability: as the size of the optimization problem increases, in terms of planning time horizon, number of EVs, size of the power network, or number of control variables and constraints, the complexity increases, and it becomes "computationally intractable with respect to the implementation time" [34], able to only handle a limited amount of EVs in real-time [15], [35], [37]. As the numbers of EVs connected to the system increase, the centralized approach becomes "impractical" [34].

In contrast, decentralized approaches have become popular in the literature because they are highly scalable, flexible, resistant to communication failure, more practical to implement in the field, and offer more control to the individual EV owners [34], [35], [38]. They are more scalable in terms of computational complexity, and have often been developed with the aim to be more computationally efficient and reduce the computational burden [10], [34]. However, their largest drawback is that decentralized approaches do not guarantee an optimal charging solution [34], [35], [38], and are generally inferior to centralized approaches in terms of "the stability of the entire power network" [38].

Ultimately, according to the literature, the decentralized approach is more computationally efficient, but does not give the global optimal or optimized solution compared to the centralized approach, which requires more computational resources, but has the potential to provide a more optimal solution. While this statement is generally agreed upon in recent literature, there are only a few works which compare directly their own centralized and decentralized solutions against each other: [44], [63], [71], [72], [73], [74], [75], [76]. Since optimization for such complex problems as EV charge scheduling will likely not find the ultimate single optimal solution with current methods, the generation of more research which directly compares centralized and decentralized model results from the same problem will help strengthen the arguments of one approach's superiority over the other. These limitations are part of what this thesis work will attempt to address: it will provide direct comparisons between centralized and decentralized models, and will address the computational burden of the centralized approach through HPC techniques.

2.3.3 Optimization Methods

There is a large variety in the optimization approaches viewed across the literature to approach the EV charge scheduling optimization problem. The most common methods include conventional mathematical approaches and heuristics, specifically metaheuristics. It has also been approached with solutions based on machine learning [15], [31], [44] and game theory [34], [77], [78]. The authors of [36] note that most centralized approaches utilize mathematical optimization like linear and quadratic programming, though many others use "game theoretic approaches, heuristics, scheduling algorithms, and medium access methods inspired from telecommunication literature." More details on conventional optimization methods, heuristics, and metaheuristics, are found in Section 2.4.

Conventional mathematical techniques such as linear and mixed-integer linear programming have been frequently used as seen in the number of papers surveyed by [15], [34], [35], however it was noted by [15] that these approaches are somewhat unsuitable, as they "cannot handle [EV charge scheduling] difficulties" due to the long computation times and the inability to add necessary problem components into the model. Some authors like

[37] have noted that it is difficult to solve the problem in real-time as the complexity increases as the number of EVs considered increases, and so have used simplified, combined, or hybrid programming techniques to approach the problem.

Heuristic and other custom algorithms like “Graph Search Algorithm” [79], “First Come First Served” [80], and “Interval Graph Coloring”[80] have also been found in the literature, often in combination with other techniques [10], [34], [35], [40], [63]. However, as with conventional methods, [15] has noted that “the solutions obtained using greedy or conventional algorithms for [EV charge scheduling] are usually sub-optimal, especially when the problem size becomes larger” [15].

Metaheuristics are popular methods for the EV charge scheduling problem, with the authors of [7] stating “the most prominent area of research in the optimization of EVs is the adoption of meta-heuristic based optimization algorithms and machine learning strategies.” The authors of [15] notes that metaheuristics are “outperform mathematical programming and heuristic methods” when problems like EV charge scheduling are complex and have a large number of possible solutions. There are a large number of metaheuristics used throughout the literature, used individually and in combination with other metaheuristics and optimization methods, so that authors can customize the metaheuristics to suit their problem and parameter sizes [7], [15], [20]. The more popular metaheuristics used with this problem seem to be PSO and Genetic Algorithm (GA) [35]. The drawbacks associated with the use of metaheuristics is their higher computational expense and related higher computational time [7], [20]. Centralized metaheuristic optimization methods have generally been found to be slower than decentralized heuristic or conventionally-based methods [70].

2.3.3.1 *Objective Functions*

Regardless of the optimization method used, most literature in the area of EV charge scheduling share similarities in terms of the specific objective function they are optimizing, because ultimately most are trying to reduce costs to achieve economic efficiency, whether from the point of view of the aggregator or the EV user [7], [34], [35]. As well, many researchers identify a large number of their equality constraints as coming from the power flow model, to ensure that their “system-level voltage and branch flow limits are met” [44]. Power flow equations will be discussed in Section 2.3.3.2.

Looking through the literature, it can be seen that there is a huge variety in objective functions, being single or multi-objective, and having a large number of constraints and other variables considered. An objective function describes what is being optimized, and a fitness function includes the objective function and the constraints or penalties of the problem [25], [43]. The fitness function is used by the optimization algorithm to “evaluate the quality of the candidate functions and to allow the metaheuristic to steer the search toward higher quality solutions” [81]. Since there is no standardized version of a single objective function for the EV charge scheduling problem, authors are free to make their problems as complex or simple as they like, depending on their computational resources. When resources are increased, the problem can be made more complex and more realistic. Multi-objective functions may be made of a combination of two or more objective functions, often added together with a weight applied to each function [82].

2.3.3.2 *Power Flow Integration*

The use of power flow equations to verify network conditions is an important element of EV charge schedule optimization problems, when an entire distribution system is taken into consideration in the problem formulation. Based on a proposed objective function charging or scheduling solution and its associated load, the power flow is run to verify if that solution meets the given equality and inequality constraints.

The authors of [25] describe how the objective functions and power flow-based constraints interact to determine the fitness function of their problem. The fitness function includes the two objective functions of their problem (income and operational cost) each multiplied by a Pareto weight added together, plus the penalty values. In their solution, the penalty values are based on the power flow results of the time period: if the generation demand does not meet the power flow load demand, if the bus voltages are over or under the power flow voltage limits, and if the current capacity of the network lines is violated. In their solution flowchart, the AC power flow is conduced as part of the PSO's evaluation phase (similar to Step 2 of Figure 2.1in Section 2.4.3.2), where the solution's parameters are used to calculate the power flow, those results are checked against the constraints and used to assign penalties, and then the fitness function is evaluated. A similar process by the same authors is shown in Figure 2.5 in Section 2.6.4.

The power flow equations model the electrical properties of a transmission or distribution network, giving the current, voltage, and real and reactive power flows at every bus in the network [43]. The notation used in each paper varies, but the general forms of the power flow equations (equations (2.1)-(2.2) copied from [21]), or equality constraints, are:

$$P_{Gb} - P_{Db} - \sum_{k=1}^{numB} |V_b||V_k|(G_{bk}\cos(\delta_b - \delta_k) + B_{bk}\sin(\delta_b - \delta_k)) = 0 \tag{2.1}$$

$$Q_{Gb} - Q_{Db} - \sum_{k=1}^{numB} |V_b||V_k|(G_{bk}\sin(\delta_b - \delta_k) + B_{bk}\cos(\delta_b - \delta_k)) = 0 \tag{2.2}$$

where P_{Gb} and P_{Db} are the real power generation and demand at bus b, Q_{Gb} and Q_{Db} are the reactive power generation and demand at bus b, $V_b\angle\delta_b$ is the per-unit complex voltage at bus b, and G_{bk} and B_{bk} are the conductance and susceptance of the branch connecting bus b to bus k.

These equations are often accompanied by the following inequality constraints ((2.3)-(2.7), from [21], [44]), modeling the physical limitations of the transmission network:

$$|V_{b,min}| \leq |V_b| \leq |V_{b,max}| \tag{2.3}$$

$$\delta_{b,min} \leq \delta_b \leq \delta_{b,max} \tag{2.4}$$

$$P_{Gi,min} \leq P_{Gi} \leq P_{Gi,max} \tag{2.5}$$

$$Q_{Gi,min} \leq Q_{Gi} \leq Q_{Gi,max} \tag{2.6}$$

$$|S_h| \leq |S_{h,max}| \tag{2.7}$$

where $|V_b|$ and δ_b are the voltage magnitude and voltage angle at bus b, P_{Gi} and Q_{Gi} are the real and reactive power for generator i, and $|S_h|$ is the apparent power in branch h.

In other sources like [25], other physical limitation-based inequality constraints may be included, to model things like the line thermal limits, power transformer limits, maximum distributed generation limits, minimum reserve [24], etc.

The inclusion of the power flow model and constraints within problem formulation is important, as without it, it is uncertain if a given optimization solution violates system voltage, current, and power flow limitations [44].

2.3.4 Multiple Parking Lots

Within the EV charge scheduling problem, there are a variety of different problem formulations in terms of the physical setup or representation of the EVs within a system. Some look at EVs within a system individually at nodes, some look at multiple EVs within a single parking lot, and some look at multiple EVs within multiple parking lots.

The problem of performing EV charge scheduling optimization on multiple EVs in multiple parking lots is particularly relevant in that most large communities currently have one or more parking lots with one or more EV charging ports [69], [83]. The authors of [5] point out the relevancy of this problem formulation to the modern world: "while most of the charging demand is currently met by home charging, publicly accessible chargers are increasingly needed in order to provide the same level of convenience and accessibility as for refueling conventional vehicles. In dense urban areas, in particular, where access to home charging is more limited, public charging infrastructure is a key enabler for EV adoption" [5].

This problem formulation is also relevant to study because it is more realistic when comparing it to the real-life operations of a city: EVs do not exist in a vacuum, operating in isolation from the grid and each other. The actions of EV charging in one parking lot in one area affect the grid, and thus the EVs in another area: “multiple [EV charging stations] connected to the same distribution network are coupled by the network constraints” [44]. The load at a parking lot increases with every EV that charges, which can create an imbalance between parking lots which change over time. The optimization of the EV charging schedules is therefore impacted by the functioning of the local parking lot, which depends on the functioning of the grid (which changes over time due to factors like generation, time of day, time of year, etc.), which depends on the functioning of other parking lots.

The problem involving multiple parking lots is more complex than that with no parking lots or just looking at a single lot, because coordination must be done between the parking lots, which may require the solving of more than one optimization problem. According to [44], there must be coordination between the system operators and the EV aggregators to ensure that the system is operated within its physical limitations, and that all objectives are satisfied “simultaneously” [44]. The centralized method would be well-suited for coordination between the different organizations or problem levels, but is accompanied by a high computational burden because of the increased amount of data and constraints that must be considered. With multiple parking lots, the power flow equations must be considered in the problem formulation to ensure that the limits of the power network which connects the parking lots are respected. This may also be the case when looking at a network with no parking lots and individual EVs charging per bus or node, however when the size of the parking lots (number of EV charging ports) increases, the load demand may vary more dramatically over time, and the size of the scheduling optimization problem at that node (number of EVs and associated parameters) increases, making the problem overall more complex.

There may also be the opportunity for direct coordination between parking lots or EV aggregators to redirect EV charging demands between parking lots. Not much literature could be found on the subject, but one paper [84] described the use of an inter-aggregator collaboration system to have aggregators communicate the charging requirements of EVs under their control to other aggregators if they do not have the capacity to charge the EV with their own equipment. This paper did not consider power flow within a specific distribution system, only using the energy capacity available to the aggregator as an aggregator constraint, however it demonstrated the requirement for information about the EVs’ origin and destination information so that an EV, if it cannot be charged by its controlling aggregator, can be redirected to another charging location within the same zone. The authors commented that centralized control would not scale well for the charge scheduling problem, however it is possible that a centralized system which contains all the information about system aggregators would be able to produce a more profitable response than the solution where aggregators only receive partial information about the other aggregators in the system.

While the authors of [85] do not tackle the problem of multiple parking lots, their paper is related in that it also involves directing EVs to a specific parking location, directing the EV to a specific port within a parking lot. The paper does not provide details on the behaviour of the distribution network during the V2G charge scheduling within the two simulated parking lots.

2.3.5 Problem Abstraction

As mentioned previously, researchers have approached the EV charge scheduling optimization problem in many ways. Many papers seem to abstract the problem in different ways, which is understandable due to the requirement to linearize or otherwise simplify a problem down to an approximate form that can be optimized.

In a general survey of EV charge scheduling optimization literature, it was found that the problem seems to be approached in one of four main ways: Approach 1) scheduling EV charging in a distribution system while abstracting distribution system power flow limitations [10], [16], [71], [84], [86], [87], [88]; Approach 2) scheduling EV charging in one or more parking lots while abstracting distribution system power flow limitations [36], [37], [40], [63], [68], [85], [89], [90], [91]; Approach 3) scheduling EV charging in parking lots in a distribution system while abstracting optimization within the parking lots [92]; Approach 4) scheduling EV charging within a distribution system while considering changes in the distribution system (i.e. verification of the network conditions during optimization algorithm evolution), with [31], [38], [44], [82], [93], [94] or without [24], [25], [26], [39], [95], [96] multiple parking lots.

In Approaches 1-3, where the power flow calculations seem to only be calculated once or are simplified (i.e., only a node power upper limit is considered), this indicates that the authors are assuming a given maximum load in a parking lot, and that the grid will not necessarily change in behaviour over time. The lack of detail provided in some papers makes it hard to confirm what level of abstraction was used in each problem formulation. As mentioned in Section 2.3.3.2, the inclusion of the power flow model and constraints are important to ensure that a solution is viable on a given distribution system, and prevents the need to validate the solution after the optimization has been performed [25].

A centralized control method with Approach 4 may be a good candidate when attempting to formulate the charge scheduling optimization problem to optimize the scheduling of EVs in multiple parking lots in a system, because the central aggregator would be able to see all details of the distribution system and EVs in the parking lots, and could attempt to combat any power imbalances by directing EVs to different charging locations (as mentioned above) or delaying their charging times. Due to the potential complexity and number of variables associated with this problem, the use of metaheuristics and HPC techniques to combat the long runtime and computational resource requirement may be a potential solution, as they were with papers [24], [25].

2.3.6 Offline versus Online Calculations

In some literature, the words “online” and “offline” have been used to refer to the strategies or calculations used in their optimization solutions. Offline strategies (one-time open-loop control strategies [34]) assume perfect predicted knowledge of a system’s operations in advance of EV scheduling, and are calculated only once [34]. Online strategies (recursive closed-loop control strategies [34]) are calculated multiple times, updating their calculated based on the feedback received from the system [34]. Online strategies are therefore more realistic, as they can respond to the uncertain EV arrival times and other uncertainties [34], [89]. As an example, in [44] the critical power is calculated in an online manner, being updated with every new EV arrival to ensure the network power constraints “are always satisfied even when there are multiple [EV charging stations] across the distribution system.” Online computation strategies are required in order to solve problems in real-time [63].

2.4 Optimization Methods and Metaheuristics

As explained in Section 2.3.3, it is the complexity in their optimization that makes EV optimization problems like charge scheduling a good candidate for use with metaheuristic optimization methods. However, as mentioned previously, they are not the only types of optimization methods that have been used with the problem. This section will provide a description of the optimization methods used in power systems and EV charge scheduling, with a focus on metaheuristics.

Optimization, as a mathematical concept, is the process of finding the maximum or minimum solution of some function compared against other potential feasible solutions, and usually subject to some specified constraints [97], [98]. Optimization problems are generally composed of objective functions (the item to be optimized), variable parameters called control (or decision) variables, and constraints on the parameter values, sometimes referred to as equality or inequality constraints [13], [21]. The goal is to find a set of variables which achieve the best, or optimum, value of the problem’s objective function within a given number of constraints [99]. Optimization problems may have a single or multi-objective function, depending on the problem to be solved.

Power systems are subject to a variety of constraints, both technical (e.g., equipment operation limits) and non-technical (e.g., costs to operate a system over a given time), so there must be a way to find the most effective way to operate the system, given the constraints. According to [11], the concepts and algorithms of optimization were introduced to power systems in the mid-twentieth century, to formalize decisions and processes around power system problems in a mathematical way. Since the early twenty-first century, there has been a large increase in the amount of research and publications concerning the applications of optimization to power engineering, as researchers and engineers attempt to find the most effective, optimal way to operate their systems for the lowest cost [11], [12], [13], [14]. Multi-objective functions are also becoming more important to study, as most real problems in power engineering involve more than one optimization objective, especially as power systems become more modern and more complex and are further integrated with new technologies like EVs [13], [14], [20], [22]. As well, as it becomes necessary to perform power system-related optimization in real-time [63], [100], research into how to adapt optimization so that solutions can be calculated faster or within a given time limit is becoming more important, especially for EV-related optimization [7], [70].

2.4.1 Conventional Optimization Methods

Conventional optimization methods are those deterministic methods based on precise mathematical formulas, often based on calculus and enumeration techniques [13]. A selected objective function is subject to given constraint functions, and the best possible choice or result from the control variable is selected from a set of potential candidate choices [19], [98].

The benefits of these types of methods is that they can often work very well, if the type chosen is well-suited to the problem [13]. As well, depending on the method, the computation time may be relatively lower (when compared to a metaheuristic method), and they may not require an initial solution before finding the optimal solution [101]. Types of conventional optimization methods include: discrete and continuous optimization methods, Unconstrained Optimization Approaches (Lagrange multiplier, Newton-Raphson, etc.), Linear Programming, Nonlinear Programming, Quadratic Programming, Interior Point, Gradient search methods, and Mixed-Integer Linear Programming (MILP) [13], [43].

There are also many drawbacks associated with these methods. They can have high algorithm complexity, which results in them being difficult to solve and computationally expensive for large problem sizes. They also may not be able to process discrete variables, and can often fail to converge to the global optimum, instead becoming trapped in a local optima [13], [19], [21], [98]. As well, they are not well suited to complex, realistic, and non-linear problems, because they require detailed and complete information on all aspects of the problem's objective function and its variables, which is often not realistic for most real engineering problems [13]. According to [24], "deterministic optimization techniques do not cope well with uncertain variables and require increasing computational resources to deal with the large-scale real-world problems." When a problem becomes very large and complex, these methods may not be able to find a convergence solution in an acceptable amount of time.

2.4.2 Heuristic Optimization Methods

Heuristics and metaheuristics were developed to improve the speed of problem solutions, and to combat the drawbacks associated with conventional optimization techniques, like the lack of guarantee of finding the global optimum, difficulties with non-linear problems, solving multiple objectives, and addressing problem uncertainty [102].

Heuristic optimization methods are problem-specific optimization methods that look to develop optimal solution procedures for their given problem [103]. They are generally classified into two types: construction heuristics, which perform iterative calculation steps to develop a solution, and improvement heuristics, which iteratively apply search operators to a provided, initial complete solution to search through the problem-specific search space and find a more optimal, improved solution [19], [103].

Types of heuristics include: nearest-neighbour, nearest insertion, cheapest insertion, furthest insertion, k-opt, and approximation algorithms like the polynomial-time approximation scheme [103].

Heuristics can improve the speed of an optimization problem when compared with conventional methods [102], [103]. They are better suited for large, non-linear search spaces [13]. If the specifics of the structure of chosen problem are known very well, a heuristic can outperform a metaheuristic. They may be easier to implement when compared to other methods, and are more flexible in addressing problem uncertainty. However, because they are problem and problem-structure specific, they are very difficult to define, and their design and application is very demanding [17], [19], [103].

2.4.3 Metaheuristic Optimization Methods

Metaheuristics are non-deterministic, problem-independent optimization methods, which work by iteratively improving one or more candidate solutions to a problem to find an optimized solution [17], [18], [103]. They are considered a type of improvement heuristics which are not problem specific [103]. They can more effectively solve a variety of problems without requiring extensive detail about the problem formulation.

Their main benefits are that they can be applied to non-linear and complex problems, which are those often found in power optimization problems, they can more easily escape local minima to find the global optimal solution, and can consider continuous and discrete variables within their problem formulation [18], [19]. They can find generally acceptable solutions in an acceptable amount of time, depending on the parameters chosen by the user [20].

Metaheuristics are often inspired by natural behaviours [19], with numerous types and subtypes. Some of the more recognized metaheuristic algorithms include: population-based algorithms like evolutionary (GA, differential evolution, evolution strategy, etc.) and swarm-based algorithms (ant colony optimization, bee colony, PSO, grey wolf optimizer, whale hunting, etc.), and non-population-based algorithms like physics-inspired algorithms (gravitational search, simulated annealing, harmony search), Tabu search, teacher-learner, artificial immune system, and more [12], [20], [22], [43].

Evolutionary algorithms are optimization methods which simulate the evolution of a population of possible solutions to a problem, based on Darwinian theories of evolution and natural selection [20], [22]. The algorithms use three main operator types in their execution: selection operators to select the parent solutions, a crossover operator to generate one or more child solutions, and a mutation operator to modify the child solutions [13], [20], [21]. GA, developed in 1975, is one of the most popular evolutionary algorithms.

Swarm intelligence methods use a population of solutions which interact with each other to generate a more "intelligent" global behaviour and achieve a common goal within the solution space, and are based on the swarming behaviour of animals and other biological organisms [20], [22]. The output of swarm optimization methods depend heavily on the tuning of the parameters – while they can generally converge faster than evolutionary algorithms, they may converge prematurely and result in a non-optimal solution [20]. PSO, developed in 1995 [104], is one of the most popular swarm intelligence algorithms. It is described in detail in Section 2.4.3.2 below.

Non-population based algorithms have a variety of different behaviours, depending on their specific solution inspiration. Physics-inspired methods imitate the process of physical phenomena, observed in chemical, human, or other natural interactions [20], [103]. The main difference between them and population-based algorithms is that they often use a single search agent or solution to search the solution space [20]. Simulated annealing, developed in 1983, is one of the most popular physics-based algorithms.

Hybrid metaheuristic techniques combine two or more metaheuristics in order to combine their strengths and limit their limitations to find high-quality global optimum solutions with fewer iterations [20], [105].

Combined metaheuristic optimization techniques combine two different optimization techniques together, i.e. a discrete method and a metaheuristic [20]. In some literature, this term is used interchangeably with hybrid optimization techniques. These techniques may be used to reduce the complexity of a problem, break it into related sub-problems each tackled by a different optimization method, and/or to increase its efficiency. Examples include GA- and PSO-linear programming [21], [105]. For example, according to [24], PSO and GA are suitable for many different types of power system optimization problems, but both suffer from "premature convergence and stagnation" [24], which can negatively affect the quality of the solution. The combination of one or both metaheuristics with a conventional optimization method can "overcome the shortcoming of both [methods]" [24].

The drawbacks associated with metaheuristics are that depending on the complexity of the problem, metaheuristics may require significant computational resources to compute the optimization problem output [21]. Correct problem-specific assumptions are still required in order to get a good output from their use, and the basic problem formulation must still be well-understood [103].

According to the authors of [12], [13], [20], [22], metaheuristics can be generally categorized into three main types: evolutionary algorithms, swarm intelligence algorithms, and non-population or physics-inspired algorithms. These classifications are helpful to explain the inspirations behind and behaviour of various algorithms, however they are arbitrary and based on the inspiration of the authors.

The effectiveness of using one metaheuristic over another for an optimization problem will depend heavily on the problem being optimized, the researcher's understanding of it, the method of solution encoding, clear identification of the objective function and control variables, the researcher's understanding of the chosen metaheuristic, a good general awareness of how their system reacts when using existing optimization techniques, and the understanding of the parameters used in the problem implementation [13]. Many algorithms are parameter dependant: a lack of well-selected and well-tuned parameters greatly impacts an algorithm's speed, efficiency, and ability to find the optimal solution [106].

2.4.3.1 *Metaheuristics for Charge Scheduling Optimization*

As mentioned above, metaheuristics are gaining popularity with EV optimization problems [7]. There are a variety of metaheuristics used throughout the EV optimization literature, often customized by the authors to suit their problem and parameter sizes [7], [15], [20], with PSO and GA seeming to be the most popular metaheuristics used with the EV charge scheduling optimization problem [35]. Papers [25], [82], [92], [96], [107] used PSO and GA (though often with slight algorithm modifications or different weights applied as in [25]), with [38], [92] comparing the performance of PSO and GA. The authors of [16] developed a "Mixed-Variable Differential Evolution" algorithm, [80] compared Simulated Annealing with other heuristics, [108] compared PSO, Imperialist Competitive, and Training-Learning, and [109] used a "Multi-Objective Advanced Grey Wolf Optimization" algorithm in their solutions. Other authors develop (and often compare against a "base" metaheuristic like PSO) hybrid and combined metaheuristics: [38] developed and compared a hybrid PSO-GA and an "improved-hybrid" PSO-GA algorithm, [40] used a hybrid "heuristic fuzzy" PSO, [86] developed PSO-Firefly algorithms with Lévy flight search strategy, [95] developed and compared a hybrid Tabu Search-Greedy Randomized Adaptive Search Procedure against the original algorithms, and [24] combined PSO with Mixed-Integer Linear Programming. Due to PSO's popularity with EV charge scheduling optimization solutions and its use within this thesis, it is described in detail in the next section.

Metaheuristics have gained their popularity with EV optimization due to the advantages they hold over other optimization methods. As stated above, they can "escape" local optima to find the global optimum of a solution, and are able to consider both continuous and discrete variables within their problem formulation [18], [21], [22], [23]. Specifically for the EV charge scheduling optimization problem, the authors of [15] state that these "inherent" advantages of metaheuristics over conventional (mathematical) and heuristic methods make them better able to tackling complex, "non-convex and non-linear" EV objective functions, particularly when problems have a large number of possible solutions and local optima, especially as techniques like linear and non-linear programming are "incapable of solving [the EV charge scheduling problem] in a reasonable amount of time" [15]. The authors of [16] also noted metaheuristics' suitability for the problem, noting that since the problem "has been proven as NP-hard, [it] is suitable for evolutionary computation algorithms such as the ant colony optimization to solve."

Even with their suitability for the problem, metaheuristics do have a few disadvantages that must be taken into account when applying them to the EV charge scheduling problem: mainly the higher computational burden, and the need for parameter tuning [22]. Metaheuristics are recognized across the literature as requiring significant computational resources to compute the optimization problem output, since they usually "require many more function evaluations than standard mathematical optimization approaches" [22], which often translates to a long computation time for very complex problems [7], [15], [20], [21]. Since EV charge scheduling optimization computational complexity "increases exponentially" [71] with the number of EVs and length of the planning horizon [63], some metaheuristic-based solutions involving higher numbers of EVs may be not be implementable for practical solutions [71], and must be limited to smaller problem sizes. For PSO specifically, the authors of [96] explain that this metaheuristic suffers from the "curse of dimensionality" and cannot handle a large amount of EVs or high number of dimensions in the EV charge scheduling problem without requiring significant computational resources: "[since] the number of particles must increase with the number of problem dimensions, and the computational time increases exponentially with the number of particles […] simultaneous optimization of practical-size distribution system areas with thousands or even tens of thousands of EVs and distribution system buses may well be intractable" [96].

The requirement for complex parameter tuning is a drawback for metaheuristic use with EV charge scheduling optimization, because tuning impacts the speed of convergence, and often requires multiple trial-and-error simulations to find an optimized parameter set [22], [106]. When there is a large amount of EVs or dimension size of the problem, imprecise tuning may greatly impact the optimization solution's ability to be computed in a reasonable amount of time.

This emphasis on computational complexity and computation time has led some authors to combine their metaheuristic-based solutions with HPC techniques, to overcome the disadvantages by reducing computation time [24], [25]. As well, the authors of [22] noted that “most metaheuristic search methods are inherently capable of parallelization,” so further research into possible parallelization methods for population and non-population-based metaheuristic algorithms used in power systems optimization problems could allow their use in real-time problems like OPF [22] and EV charge scheduling optimization, and could allow for the study of larger problem sizes. The authors of [110] note that the parallel implementation of evolutionary algorithms have become “an effective strategy to reduce the [computation] time cost.”

2.4.3.2 *Particle Swarm Optimization*

PSO is a population-based, swarm intelligence algorithm, where a large population or swarm of particles (candidate solutions) move within a multidimensional problem space [21]. Each particle has an initial randomized directional velocity and position which are updated at every iteration based on its previous best position and the swarm’s best position, and they work together to find the best solutions to the given problem based on their experiences [12], [13], [21]. PSO is similar to another algorithms like GA in that it is population-based and iterative, with its initial population composed of a set of random solutions, however it differs in its general behaviour, its fewer parameters, and in its ease of use due to its very simple encoding, [13], [21], [22], [24]. Other advantages of PSO include its simplicity, efficiency, general effectiveness in solving a large variety of engineering problems, and its relatively faster execution time, which contribute to it being an ideal choice for those optimization problems which are constrained by computational resources [13], [22]. Its main disadvantage is that it is prone to premature convergence, as with other swarm-type methods, and has been noted to suffer from “slow convergence speed and sometimes local optima” [111].

The general steps followed are adapted from [112] and shown in Figure 2.1, where the initial candidate solutions (particles) are randomly initiated, best personal and global positions are updated, new particle position and velocities are updated, and so on, until the process is terminated.

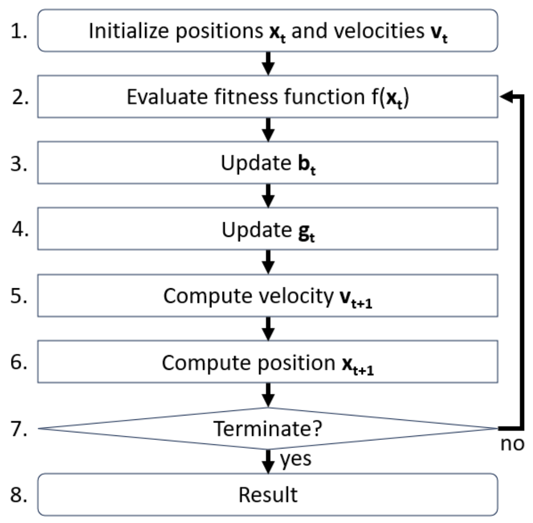

1.
Initialize positions $\mathbf{x_t}$ and velocities $\mathbf{v_t}$
2.
Evaluate fitness function f($\mathbf{x_t}$)
3.
Update $\mathbf{b_t}$
4.
Update $\mathbf{g_t}$
5.
Compute velocity $\mathbf{v_{t+1}}$
6.
Compute position $\mathbf{x_{t+1}}$
7.
Terminate?
no
yes
8.
Result

Figure 2.1 PSO Algorithm Flowchart

As explained in [112] the standard [113] PSO equations used to calculate the velocity and position of a single particle, at iteration t, in steps 5 and 6 in Figure 2.1 are:

$$\boldsymbol{v}_{t+1} = \omega \boldsymbol{v}_t + c_1 \boldsymbol{r}_1 (\boldsymbol{b}_t - \boldsymbol{x}_t) + c_2 \boldsymbol{r}_2 (\boldsymbol{g}_t - \boldsymbol{x}_t) \tag{2.8}$$

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{v}_{t+1} \tag{2.9}$$

where: all bold variables are vectors, v is the velocity of the particle; x is its position; b is the best position previously occupied by the particle; g is the best position previously occupied by any particle of the swarm; r_1 and r_2 are vectors of random values between 0 and 1; and ω, c_1 and c_2 are the inertia, the personal influence and the social influence parameters. The dimension d of each vector is the number of independent variables being optimized [114].

Since PSO has been used in multiple EV charge scheduling optimization papers in both standard and hybrid forms [7], [15], [24], [25], [38], [40], [82], [92], [96], [107], [108], there are a number of ways that the metaheuristic has been adapted to the problem. In [40], the dimension of the vectors is "derived by the attributed of the EVs," being the number of EVs (20-150) multiplied by the number of timeslots (10), giving up to total 1500 variables to be optimized. The authors of [25] uses 46848 decision variables related to the "generators active and reactive power and [EV] charge and discharge." In [24], the decision variables are "the scheduling of the energy resources, such as the EVs' charging schedule and the network variables (such as power flow, voltages and losses)." Papers [38] and [96] optimize scheduling of n_{EV} number of EVs with dimension $d = n_{EV}$ for the vectors, 50000 and 57 EVs respectively. In the other solutions, the size or basis for the dimension or number of variables is not explicitly stated.

2.5 High Performance Computing

HPC refers to the use of powerful processors in parallel to solve complex problems and process large datasets at high speeds [23], [115]. In some literature it also overlaps with the term Parallel Heterogeneous Computing, which refers specifically to systems where the different types of processors (primarily Central Processing Units (CPU) and Graphics Processing Units (GPU)) in a computer system execute tasks in parallel [23], [116]. There are two main goals or uses of HPC: to perform calculations at a faster rate than conventional computing [117]; and to solve large and complex problems that may have been considered intractable or unapproachable [117]. Combining these uses leads to a third goal that can be achieved [118]: to solve a more complex version of a problem (i.e., of the same size but considering more constraints or other parameters) within the same amount of time as conventional computing, increasing the computations done within that same amount of time (per the Gustafson-Barsis Observation, or Gustafson's Law [119], [120], see Section 2.5.5).

For EV-related problems, HPC has been used to optimize the design of vehicle subsystems, simulate charging loads and energy consumption to identify and minimize power losses, determine driving routes to minimize power consumption, and to minimize transmission dispatch costs [26], [27], [28], [29], [30]. The authors of [26] note that the use of HPC is becoming very important for EV research and other problems which model a large number of EVs or distributed generation items as variables, because this makes them "a computationally-intensive optimization problem." Research in HPC and EVs has not seemed to have advanced largely into the area of charge schedule optimization, at least not in the area of combining metaheuristic-based solutions with HPC techniques: only two papers were found during the course of this review with used both metaheuristics and HPC [24], [25]. In both papers, which share the same first author, parallel computing (distributed and multicore) was used through MATLAB, with the aim to reduce the computation time of the PSO-based optimizations. In the first paper, two machines with 24 total cores were used to reduce the computation time from 42 hours to three hours and 48 minutes (a speedup of 11 times) [25], and in the second, a HPC cluster with six machines and 42 cores was used to reduce the computation time of their two-step solution from 12 hours to 30 minutes (a speedup of 24 times) [24]. Neither paper appeared to use of heterogenous computing. The growth of the HPC market and its increasing accessibility to researchers and other users due to advances in technology is only expected to rise [121], [122], and may appear in more works as research continues over time.

This research aims to achieve all three "goals" of HPC, because it uses or tests three main techniques in the EV charge scheduling optimization problem, that have traditionally been deemed time consuming and computationally expensive:

1. Centralized EV charge scheduling optimization, for EVs in multiple parking lots: a conventionally unscalable approach which requires analysis and calculation on a large amount of data, and a more complex problem which requires the inclusion of more constraints like power flow (see Sections 2.3.2, 2.3.4). The literature reviewed which looked at multiple parking lot optimization was a mix of decentralized [44], [93] and centralized [31], [38], [82], [94] methods, but this research will aim to use the centralized method to provide a more optimized solution for multi-parking lot charge scheduling optimization, and provide a method to parallelize it to reduce its computation time.

2. Metaheuristic-based optimization: computationally resource-heavy and requiring long computation times when tackling a large problem sizes and suffering from the curse of dimensionality (see Section 2.4.3.1).

3. Perform the optimization in real-time: the online charge scheduling optimization calculations for the EVs will be completed within 15 minute intervals (justification for this interval length is provided in Section 2.5.4). Scalability of the solution will be tested to determine the maximum problem size that can be optimized in the 15 minute window.

These three aspect of this thesis connect to all three goals of HPC: with the centralization, multiple parking lots, and metaheuristics, HPC is needed to solve a more complex problem [15]. With the need for real-time optimization, HPC is needed to solve that more complex problem within a much faster timeframe. To test the scalability of the problem, HPC is needed to increase the computations completed within the real-time limit since the computational cost or runtime increases as the number of EVs, system size, or other parameters increase [15].

Using a centralized optimization approach, a better, more precise and realistic (more constraints) solution is expected to be achieved with the HPC techniques, as with HPC the computational cost of a centralized solution is not a concern [44]. As well, the development of centralized and decentralized optimization models in this work will enable direct comparison of optimization and timing results. Papers like [44], [63], [71], [72], [73], [74], [75], [76] do compare both, often demonstrating that centralized results are slightly more optimized, but do not necessarily dictate the specific limits that are reached before the centralized versions have too much "computational complexity, control overhead" to deal with [71].

The use of HPC in this thesis is expected to result in the finding of a more optimal solution to the EV charge scheduling in multiple parking lots problem, using centralized optimization, and which is able to be computed in an appropriate real-time interval.

2.5.2 Definitions

This section provides short definitions of the HPC-related terms used within this thesis document: Single Program, Multiple Data (SPMD), multicore computing, distributed computing, GPU parallelization, and computing cluster:

1. Single Program, Multiple Data (SPMD). In the SPMD computational model, multiple threads or processors run the same program code, but each thread applies the code logic to a different set of data [123].

2. Multicore computing. Multicore processors are computer chips which have two or more processing cores (CPUs, also called compute nodes, cores, processors, or workers) placed on the same chip [121], [124]. These processors can be used to simultaneously execute multiple different instructions or to divide the workload and execute the same instructions on different data in parallel, called multicore or parallel computing [121], [124], [125]. In this thesis, multicore computing refers to the use of multiple processor cores within one computer to complete a computing task [126]. With a SPMD-model program, each core executes the same code on different portions of a dataset in parallel. A simplified visual representation of multicore computing is shown in Figure 2.2 (a).

3. Distributed computing. This term refers to the use of two or more separate computers (or hosts) connected in a network to act together as a single, more powerful computer [125], [126]. Each processor within each computer can be used to perform different tasks simultaneously, or the same task in parallel [23]. A simplified visual representation of this is shown in Figure 2.2 (b). In this thesis, distributed computing refers to a combination of multicore and distributed computing, using multiple processor cores within multiple computers within a cluster to complete a computing task. With a SPMD-model program, each core of each computer executes the same code on different portions of a dataset in parallel. A simplified visual representation of this computing method is shown in Figure 2.2 (c).

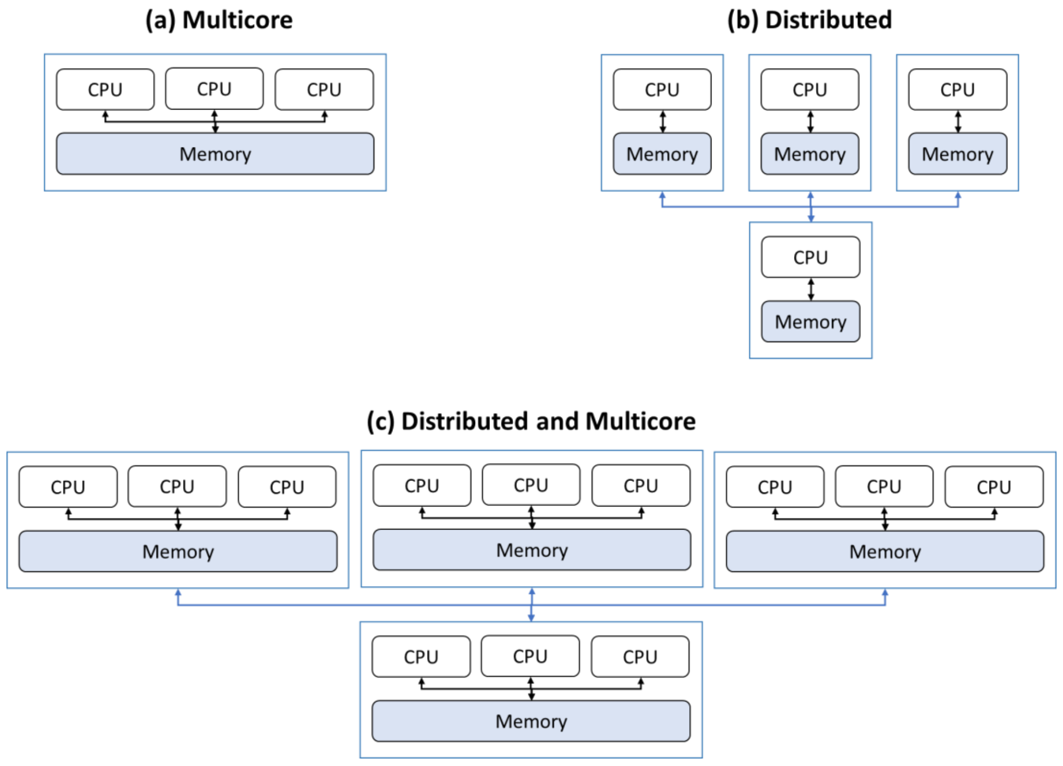


Figure 2.2 Multicore, Distributed, and Distributed and Multicore Computing Systems (adapted from [126])

4. GPU parallelization. This is another parallelization method which is often encountered in literature on HPC, sometimes referred to as GPU or CUDA parallelization. Parallelization is possible on GPUs with the use of specialized languages such as the NVIDIA® CUDA™ programming language [18], [110], [121]. GPUs contain many more cores than CPUs, and through the use of a kernel function a large group of threads "in a single multi-processor block" [112] can be made to execute a set of instructions on different data in parallel in the SPMD method [112], [121]. Further information on this parallelization method can be found in online and text sources [121], [127], [128]. CUDA C++ programming was not used with this thesis work.

5. Computing cluster. A computing cluster, sometimes called a HPC cluster in this thesis, is a group of computer servers networked together to compute the parallel computing task. Most computers in clusters use multi-core CPUs and/or GPUs to perform the computing tasks [115]. Clusters can be used to perform multicore computing when using all cores within a single computer, distributed computing

when using multiple computers in the cluster, and a combination of the two when using multiple cores within multiple computers within the cluster.

2.5.3 Parallelization of the Particle Swarm Optimization Algorithm

PSO has been identified as a metaheuristic with inherent parallelism [110]. This has led to a number of works presenting methods to parallelize the PSO algorithm, using different algorithm parallelization methods and different programming methods. Some methods are presented here.

Many methods focus on the fitness function evaluation of each particle, since they can be performed in parallel because the evaluation of each candidate solution does not depend on the evaluation or outcome of any other candidate solution. This method has been referred to as coarse-grain parallelization [129]. A coarse-grain synchronous parallelization method is presented in [130], where each particle's fitness evaluation is completed in parallel, and all particles evenly distributed between the available compute nodes by using an MPI-based "primary-secondary" implementation. Barrier synchronization is used to force the program to wait for all secondary nodes to complete their computations before moving onto the evaluation of the best personal and global positions.

To improve parallel efficiency, the authors of [129] present an asynchronous method, where the particle fitness functions evaluations and all the best personal position update iterations are completed in parallel for all individual particles, and then the global best position is updated at the end of each main PSO iteration. While the particles are updating their designs, they only have access to the best possible information available at the moment, not the best overall for each iteration [129], [131]. This method also used the MPI master-slave implementation.

The authors of [110] present a combined coarse-grained and fine-grained method using heterogenous computing with CPU and GPUs working together. The fitness function evaluations and position updates are completed in parallel by the GPUs. Coarse-grained parallelization is used to evaluate the fitness function of each particle, with one thread computing the fitness function result for one particle. Fine-grained parallelism is used to update the best personal positions of each particle, with a group of threads (one per each dimension of the particle) computing the personal position update for one particle, and each thread performs the same operations or instructions.

Other parallel PSO methods are described in [131], including multi-swarm approaches, map and reduce, and GPU-based methods. The authors mention that CPU-based PSO parallelization are less complicated than GPU-based methods, and that while the synchronous implementation method is easy to implement, it may result in poorer parallel efficiency than other methods [129], [131]. In all cases, the number of CPU and/or GPU cores affects the potential performance (speedup) of the parallelized algorithms.

Other PSO parallelization examples can be found in literature, where some cases parallelize the PSO algorithm using multicore and/or distributed computing [21], [112], [132], [133], [134], [135], [136], [137] or GPUs [18], [112], [133], [138], [139], [140], [141], and others parallelize the execution of multiple PSOs [24], [25].

2.5.4 Real-time limits

The real-time aspect of the aim of this thesis comes from the literature, which has indicated that while optimization time intervals should generally be as short as possible in order to reflect the most recent changes of the system and therefore generate the most accurate charge scheduling optimization [10], a 15-minute period of time is the commonly-used "time slot length" or "time interval duration" used for power system load measurements and EV charge scheduling research [71], [142]. Per [16], the length of the timeslot determines the frequency of calling an online scheduling algorithm to complete the optimization calculations, and "has direct impacts on both the system response time and the consumed computational resources" [16], so papers like [16] have chosen the 15 minute time length to balance the time and the computational cost. The optimization calculations must be complete in 15 minutes or less, so that the results may be applied and the next schedule optimization or update can be calculated.

While some papers have used time intervals as short as 10 minutes [37] to as long one hour [68], 15 minutes seems to be a very widely-accepted time interval [10], [16], [44], [63], [94], per the limited literature review shown in Table 2.2. Therefore, in this work, the optimization calculations for the charge scheduling for each EV must be completed within 15 minute intervals, in order to be considered completed in real-time.

While the use of HPC might enable the solving of a specific problem size in as small of a time interval as possible, in this thesis, part of the work will be done to determine the scalability of the solution to find what maximum problem size with the highest precision can be solved in an acceptable real-time interval of 15 minutes.

Table 2.2 Selected Real-Time Intervals in Literature

Time interval (min)	Optimization problem	Optimization method	Problem size	Ref.
10	Maximize revenue and minimize power consumption cost for a smart parking lot	Linear programming and simplified convex relaxation approach	1 parking lot, 500 EV	[37]
15	Maximize use of off-peak power (valley filling)	Valley-filling algorithm	1 000 000 EV	[10]
15	Minimize time cost, charging expense, maximize recharging	Mixed-Variable Differentiate Evolution	10 parking lots (charging station) with 50 EVs, 20 parking lots with 100 EV	[16]
15	Maximize utility of service provider	Load-shifting heuristic	480 EV in 5 parking decks	[63]
15	Minimize network losses and user inconvenience	Mixed Integer Quadratic Programming with GUROBI solver	500 EV, 5 parking lots, 33+ node network	[94]
15	Minimize demand, cost, maximize recharging	Cooperative hierarchical multiagent system	300 and 900 EVs, 3 and 6 parking lots on 33 and 118-bus system	[44]
30	Maximize revenue and num. recharged EVs	AIMMS software	2 parking lots, 7800 EVs	[36]
30	Minimize electricity cost for parking lot	Heuristic fuzzy PSO	1 parking lot, 20-150 EVs	[40]
30	Charging cost and waiting time minimization	Two-stage PSO and GA	496-992 EVs in 22 residential and 5 commercial charging platforms	[143]
60	Maximize recharged EVs	MATLAB software (nonlinear, convex programming)	1 parking lot, 1-40 EVs arriving per hour	[68]

It is worth noting that the real-time interval length which determines the frequency of calling an online scheduling algorithm and therefore its calculation completion time limit, may not be the same as the timeslot duration (scheduling interval Δt, or time horizon) used within the EV charging schedule problem [70]. For example, the scheduling interval from the algorithms developed in [24], [25] is dependant on the cost function time intervals, and since both algorithms are for day-ahead scheduling, it does not affect the real-time interval or time limit required for the optimization solution to be calculated.

2.5.5 Speedup

When comparing the success of the parallelization of an algorithm or program implementation, in terms of meeting the HPC aim of performing calculations faster and therefore solving more complex problems at high speeds [23], [115], one of the metrics used is called the speedup. Per equation (2.10), in the simplest terms, the speedup is the ratio of the computational runtime of the parallelized algorithm to the computational runtime of the sequential algorithm:

$$s = \frac{T_{seq}}{T_{par}} \tag{2.10}$$

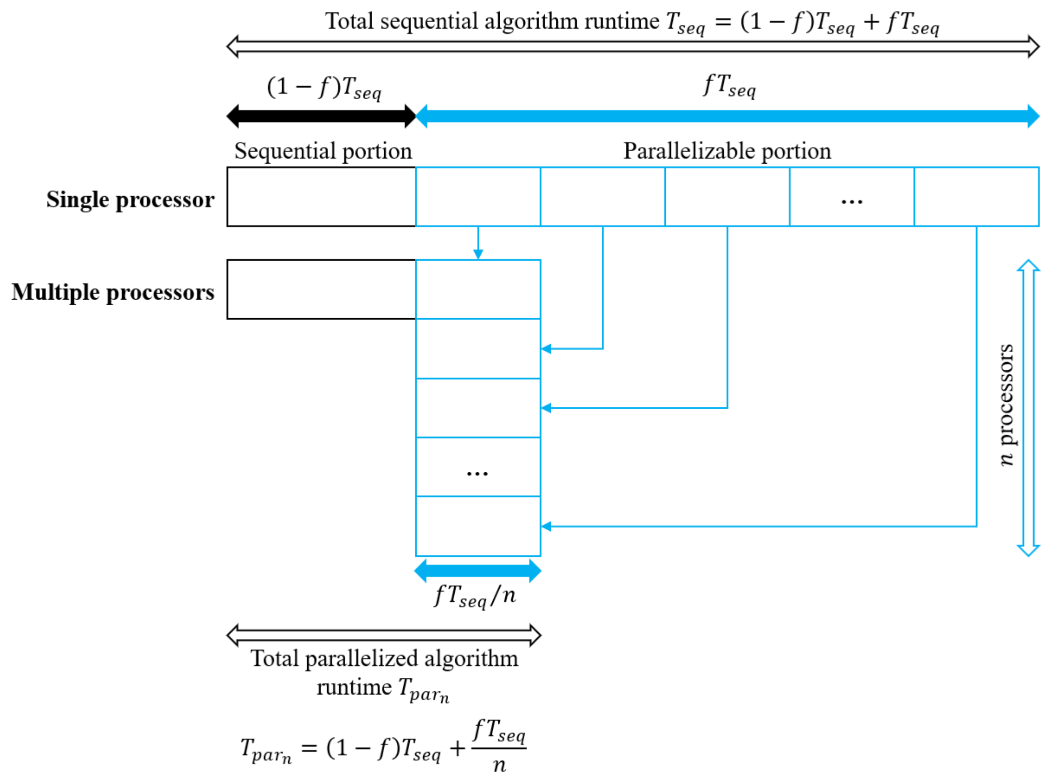
where s is the speedup, T_{seq} is the computational runtime of the sequential algorithm, and T_{par} is the computational runtime of the parallelized algorithm. For a speedup of value s, the parallelization is said to have a speedup of "s times" or an s-fold speedup [144]. As the runtime of a parallelized algorithm improves, i.e., becomes shorter, the speedup improves. For an algorithm to be considered successfully sped-up, the speedup $s > 1$.

However, the speedup of an algorithm is limited by the part of the algorithm which is "inherently sequential" [119]. Per Amdahl's Law [119], [145], the speedup using n processors is a function of the proportion of the algorithm which can be parallelized, per equation (2.11):

$$s_n = \frac{T_{seq_{n=1}}}{T_{par_n}} = \frac{1}{\frac{f}{n} + (1 - f)} \tag{2.11}$$

where s_n is the speedup using n processors, f is the proportion of the algorithm which can be parallelized, $T_{seq_{n=1}}$ is the computational runtime of the sequential algorithm using one processor, and T_{par_n} is the computational runtime of the parallelized algorithm using n processors.

Since the runtime of the inherently sequential portion of the algorithm $1 - f$ cannot be reduced by adding more processors or improving the speed of the parallelized portion, it dominates the runtime of the total algorithm. This is illustrated in Figure 2.3. The total runtime of the algorithm is separated into the inherently sequential and parallelizable portions. The runtime of the sequential algorithm (the algorithm if run by only one processor, $n = 1$) T_{seq} is made up of the runtime of the inherently sequential portion plus the total runtime of all the parallelizable portions of the algorithm, run sequentially one after another. The runtime of the parallelized algorithm, with its parallelized portion divided up over n processors T_{par_n} is made up of the runtime of the inherently sequential portion, plus the runtime of the portion of the parallelizable work done by n processors in parallel.



Total sequential algorithm runtime $T_{seq} = (1-f)T_{seq} + fT_{seq}$
$(1-f)T_{seq}$
fT_{seq}
Sequential portion
Parallelizable portion
Single processor
…
Multiple processors
…
n processors
fT_{seq}/n
Total parallelized algorithm runtime T_{par_n}
$T_{par_n} = (1-f)T_{seq} + \frac{fT_{seq}}{n}$

Figure 2.3 Amdahl's Law (adapted from [118], [120])

Per [119], the larger the parallelized portion f, the more the speedup can be improved by increasing the number of processors. The larger the inherently sequential portion, the faster a speedup plateau is reached, meaning that after a certain point, the addition of extra processors will not improve the speedup, and may detrimentally affect the runtime if more overhead communication is required by the parallelization method. When the inherently sequential proportion of the algorithm is known, Amdahl's Law can be used to provide the maximum possible speedup that could be gained as the number of processors is increased, per equation (2.12) (from [119]).

$$f = \left(\frac{n}{n-1}\right)\left(1 - \frac{1}{s_n}\right) \tag{2.12}$$

While Amdahl's Law is related mostly to the HPC aim of performing calculations faster, there is another perspective which is related to the aim of HPC of solving more complex versions of a problem within a given timeframe, increasing the number of computations done in that time: the Gustafson-Barsis Observation, or Gustafson's Law [119], [120]. This observation posits that the problem size scales with the number of processors, and that a much more complex version of a problem can be tackled with HPC techniques in order to have it run within a set timeframe. The more processors one has, the larger and more complex the problem to be solved can become. In this case, the amount of work that can be done in parallel f varies linearly with the number of processors n [120], and so the scaled speedup is proportional to n or $1 - n$, per equation (2.13):

$$s' = (1 - f) + fn = n + (1 - n)(1 - f) \tag{2.13}$$

where s' is the scaled speedup. As shown in [119], when the parallelizable portion of the work f is larger, and the amount of data to process is so many times larger, the maximum potential speedup increases in a linear fashion.

Knowing these concepts, it is clear that the speed and final speedup achievable through a parallelized EV charge scheduling algorithm will be largely dictated by the length of the inherently sequential portion of the algorithm $1 - f$. If that portion consists of the part of the algorithm which optimizes a group of parking lots, the parallelized algorithm will only run as fast as the single slowest processor or worker in a HPC system can compute the optimization of those lots. If the inherently sequential portion is made smaller, to consist of the parallelization of a single parking lot, or of a single EV, then the runtime and speedup can be expected to improve. Considering the Gustafson-Barsis observation, if the size or dimension of the problem increases, then the runtime is expected to increase, but the potential speedup is expected to improve as well.

2.6 Related Works

The following sections provide a brief summary of five papers which include concepts, design methodology, or other research closely related to and/or used in this thesis. While none share the exact methodology of the full design of the centralized multiple parking lot optimization model (see Section 3.3), elements of these papers greatly influenced or provided the basis for different aspects of the model's design. The works present an algorithm for EV charge scheduling in a single parking lot [40], two-level optimization algorithms using PSO-PSO [111] and PSO-GA [143] for EV charge scheduling, and the integration of PSO-based EV charge scheduling with HPC [24], [25].

2.6.1 Dynamic resource allocation for parking lot electric vehicle recharging using heuristic fuzzy particle swarm optimization algorithm (2018) [40]

This 2018 paper influenced this thesis by providing an example of a single-objective, PSO-based optimization algorithm for charge scheduling in a single parking lot, which formed the basis for the single parking lot optimization method developed in this thesis.

This paper presents a "dynamic resource allocation system" [40] for the optimization of charging of EVs in a single parking lot, using a developed heuristic fuzzy

PSO algorithm to minimize the cost of purchasing electricity to charge the EVs. The paper presents the results of its algorithm for simulations of a single parking lot with 20, 40, 100, and 150 EVs with transformer limits of 60, 110, 500, and 460-540 kW respectively. The simulations were computed with MATLAB (R2016b) on a workstation with an Intel® Core i5-4570 CPU.

Using the presented algorithm, the parking lot (the local aggregator or optimizer and executor of the schedule) finds an optimal charging schedule for all EVs which arrive at its location. The parking lot can consider the demands of EVs with and without appointments (without their arrival time, departure time, and charging demand i.e. desired consumption known in advance), as it calculates (and executes) the optimized schedule at the beginning of each timeslot, for that current timeslot as well as all subsequent timeslots until the end of the total charging period. At every new timeslot, the schedule is re-calculated and executed. The algorithm does not identify the specific mechanism used to deal with unacceptable charging requests or infeasible solutions.

The authors of the paper considered only unidirectional (non-V2G) charging for their algorithm, due to a number of factors including the potential drawbacks of battery damage, lack of charging flexibility for the EVs, and the potential for demand to be unmet due to changes in departure time [40]. Their model did not include renewable energy resources like distributed generation or energy storage systems, abstracting the behaviour of the power system the parking lot is connected to and only considering the transformer limit of the parking lot as the maximum power supply from the grid to the parking lot. In their algorithm, the transformer limit is the largest constraint on how many EVs can be charged at a time [40]. Only one EV type was used as a model, and only one Level 2 charging rate limit was used in the simulations.

The objective of the model in this paper is to minimize the cost of purchasing electricity by the parking lot to charge the EVs, taking into consideration the cost of electricity varies with every timeslot. The problem formulation uses continuous variables to represent the EVs charging decisions, from 0 (no charging) to the charging rate limit (maximum possible charging rate in one timeslot). The solution vector or decision is therefore to have each EV assigned a charging decision (proportion of its total charging demand) to each timeslot it is present at the parking lot. The objective function result is constrained by the charging rate limit, transformer capacity limit, and requirements of each EV to charge to 100% of their demand.

The heuristic fuzzy PSO algorithm was developed by the authors to provide a method which could provide a more optimized result than other PSO-based algorithms. With this algorithm, a proportion-based assignment method was used to generate a more efficient initial population of solutions for the PSO. The PSO then executes one iteration, followed by the proposed fuzzy system determining the new inertia, personal influence, and social influence parameters (called the personal and global acceleration coefficients

[40]), and the proposed heuristic reduces the cost of the current iteration's solution. The algorithm repeats the PSO-fuzzy-heuristic process until the maximum number of iterations has been met, and then provides the optimal solution.

The authors compared the results of their heuristic fuzzy PSO algorithm with first-in-first-serve and earliest-deadline-first mechanisms, random search, original PSO, inertia-weight PSO, construction PSO, "original fuzzy" PSO, and proposed fuzzy PSO results [40]. In all cases, the heuristic fuzzy PSO algorithm was able to produce a more optimal solution, though it required a longer computational time [40].

While the objective value and optimization methodology used within this paper was single-objective and relatively simple when compared to some multi-objective papers which considered EVs in different locations or multiple parking lots, it provided the basis for the single parking lot optimization method used in this thesis, and its use and comparison with other PSO-based algorithms makes it ideal to compare results against when using the same simulation parameters.

A limitation of this work is that it did not consider the impact of the various transformer limits of the parking lot on the physical grid. The transformer limit varied between 60-540 kW, but there was no consideration on what the various maximum limit values, representing the maximum power supply to the parking lot or maximum demand from the grid, would have in terms of impacting the limitations of the distribution system it was connected to. To improve upon this, this thesis will expand the basic PSO-based optimization method presented in this work to integrate power flow calculations, to ensure that the transformer limit set for the parking lot does not exceed the limits of the distribution system. As well, this this thesis will extend the basic single parking lot optimization method to be used for the optimization of multiple parking lots, to better reflect the current state of public charging infrastructure, where one community often has many different public parking lots with a smaller amount of EV charging ports, rather than one large parking lot which can charge a very large number of EVs [69].

2.6.2 Multi-Objective Scheduling of Electric Vehicles in Smart Distribution Network (2016) [111]

This 2016 paper also influenced this thesis, by proposing a method to integrate the optimization of the EV charging with the optimization and consideration of the distribution network, through the use of a two-level, PSO-based algorithm.

This paper presents a two-level ("bi-level" [111]) EV charge scheduling algorithm which coordinates the charging of EVs controlled by aggregators using improved PSO. The algorithm coordinates between two levels of optimization objectives, to coordinate the charging and discharging of EVs in aggregator-controlled parking lots with the distribution network loads of the network they are connected to, and the price of electricity. The algorithm considers V2G EVs, capable of charging and discharging power to the grid. Simulations were performed to coordinate the charging of 1100-2000 EVs in a six aggregator (parking lot), 69-bus system, on a single workstation.

The algorithm uses a hierarchical approach in order to provide the optimization required to the EVs but also reduce the communication burden on the network, by using EV aggregators to manage the EVs in a region: the network operator communicates with the aggregators, and each aggregator communicates with its group of EVs. The upper level of the model considers the demands of the EV aggregators and distributed generators (wind turbines, photovoltaic generators), and minimizes the network operating cost [111]. The upper level minimizes three objective functions, which minimize: the cost of purchasing electricity from the main grid and the distributed generation sources, the cost of network losses calculated through power flow, and the cost of the EV charging by the aggregators. The functions are constrained by the power flow power balance equations, grid active and reactive power limits, aggregator apparent power limits, aggregator EV state of charge limits, and distributed generation power limits.

The lower level of the model uses the results of the upper level optimization to determine the charging schedules for the EVs. The objective of the lower level was to minimize the deviation between the total aggregated charging power of the EVs, and the charging power amount optimized by the upper level model. This single objective function was subject to EV state of charge (security) constraints, charging equality limits, charging power limits, and final state of charge limits.

The authors used the “improved” PSO in their algorithm by combining PSO with the interior-point method in order to improve the algorithm’s ability to locate and avoid local optima [111]. According to the flowchart included in the paper, copied below in Figure 2.4, the two-level algorithm worked as follows: at the beginning of one iteration of the algorithm, the upper level model used a complete improved PSO to find its optimal solution outputs. These outputs, or upper level decision variables, were integrated as the lower-level inputs, and the lower level used another complete improved PSO to find its optimal solution, and calculated the total charging power of all EV aggregators. At the end of the iteration, the lower level results were checked against the constraints of the upper level, and if they did not meet the constraints, then the upper-level constraints were altered to fit the lower-level results, then a new iteration of the whole algorithm started, with the lower level decision variables integrated into the upper level inputs. Based on the flowchart, the lower level improved PSO-based algorithm was not contained within the upper level improved PSO-based algorithm, but was executed following the completion of the upper level algorithm.

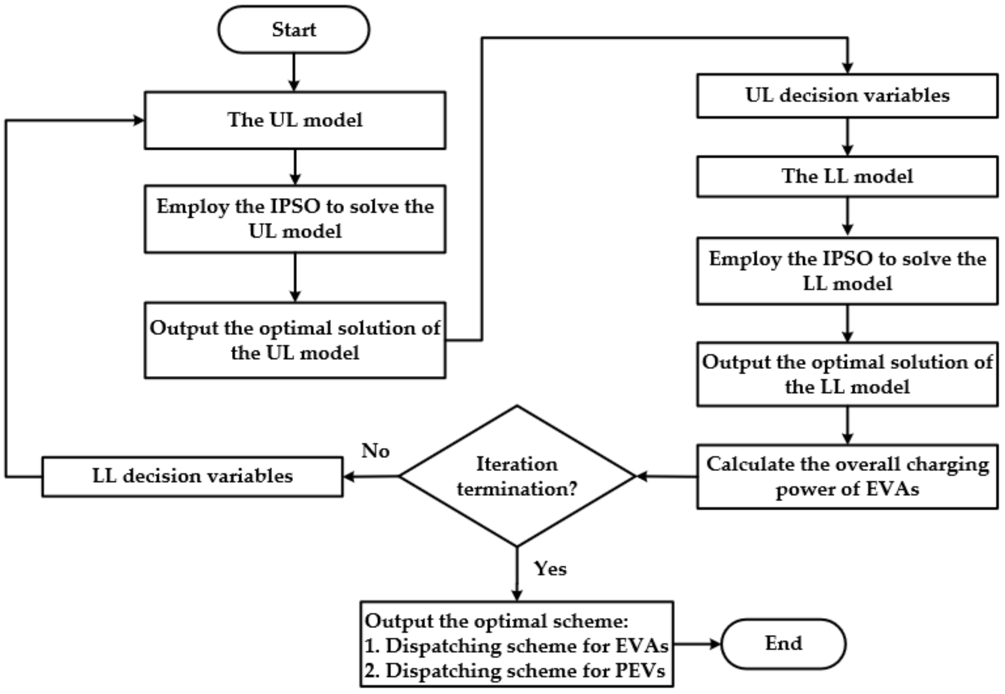

Start
The UL model
Employ the IPSO to solve the UL model
Output the optimal solution of the UL model
UL decision variables
The LL model
Employ the IPSO to solve the LL model
Output the optimal solution of the LL model
Calculate the overall charging power of EVAs
Iteration termination?
No
LL decision variables
Yes
Output the optimal scheme:
1. Dispatching scheme for EVAs
2. Dispatching scheme for PEVs
End

Figure 2.4 Dispatch model flowchart (copied from Figure 2 of [111])

The authors compared the results of their model against an uncoordinated charging scheme, and used different case studies with varying percentages of dispatchable EVs. Their proposed model was able to provide more optimal solutions than the uncoordinated charging model, minimizing the distribution system losses and cost of energy, and was able to improve the results when using higher amounts of dispatchable EVs.

This paper provided an example of the integration of two-levels of optimization together, with each level optimizing different objective functions with different constraints, while relying on the outputs of the other to influence its own optimization. A limitation of this work is that the algorithm must wait for the completion of the full lower level improved PSO algorithm, before it determines if the constraints have been violated and if another full iteration of both upper and lower level algorithms is required to find a feasible optimal solution – while not addressed in the paper, this algorithm behaviour is likely to require a longer execution time (the authors did not analyze the performance of the model in terms of behaviour in a real-time environment, and did not provide computation runtime information [111]). This thesis proposes a method influenced by this paper, which uses two-levels of PSO-based optimization, with one level concerned with the optimization of the EV charge scheduling, and the other concerned with the optimization of the parking lot transformer limits and behaviour of the distribution network. However, in order to consider a real-time environment and address the likely longer computational runtimes required of the two-level algorithm proposed in this paper, this thesis will propose a two-level PSO-based optimization algorithm where the inner (lower) level algorithm is executed within the outer (upper) level algorithm instead of after its completion. This will allow for the checking of the outer level algorithm's constraints against the inner level's results throughout its iterative process, decreasing the computational runtime required and allowing the model to perform its calculations within a real-time limit. This thesis will also demonstrate the performance and scalability of the model on more than one distribution system size, one of which is the 69-bus distribution system (though the simulation parameters, problem setup, and number of EVs are different).

2.6.3 A Two-Stage Multi-Agent EV Charging Coordination Scheme for Maximizing Grid Performance and Customer Satisfaction (2023) [143]

This 2023 paper influenced this thesis in a similar way to [111], also proposing a method to integrate the optimization of the EV charging with the optimization and consideration of the distribution network, through the use of a two-stage, metaheuristic-based algorithm.

This paper presents a two-stage, sequential, multi-agent-based algorithm to optimize and coordinate the charging schedules of EVs in aggregator-controlled charging platforms, using PSO and GA, considering both grid performance and customer satisfaction. Simulations were conducted to coordinate the charging of 496-992 EVs in a 33-bus distribution system with 22 residential and five commercial EV charging aggregators, using MATLAB (R2015a) on a workstation with an Intel® Core i5-5200 CPU.

The algorithm uses a hierarchical two-stage method, where the first stage, the distribution network operator, aims to divide power between EV aggregators while minimize power losses and voltage deviations to maximize grid performance. In the second stage, the aggregators determine the charging schedules of all EVs in their charging platforms while maximizing customer satisfaction through minimal cost and minimal waiting time. These optimizations are done sequentially, with the second stage requiring completion of the optimization of the first stage before it can begin its optimization process. The authors explained that the choice of diving the algorithm into two stages was to reduce the computation time and resources required for its calculations.

In the first stage of the algorithm, the network operator receives the charging requests from all aggregators. Its objective function is to minimize the power loss index and voltage deviation index of the network, constrained by the voltage limits, power allocation limitations to serve only active aggregators, power distribution limits to each aggregator, and maximum demand limits. The output of the optimization using PSO is the distribution of power among the aggregators.

In the second stage of the algorithm, each aggregator receives their distribution of power from the output of the first stage. Each aggregator has the objective of minimizing the cost index and waiting time index, while constrained by the time required to achieve the requested state of charge. The output of this GA optimization is each EV's charging power and connection time. Once all aggregators in the second stage have completed their optimization process, the network operator in the first stage calculates the total system demand and verifies adherence to the network constraints.

The two-stage algorithm is unlike the one presented in [111], where the two-level optimization process may be repeated for a number of iterations until all constraints are met. In this paper, the algorithm is executed only once: once the first stage optimization is complete, the second stage optimization completes, and then the algorithm ends.

The authors tested the two-stage algorithm against uncoordinated EV charging, and coordinated charging with waiting time and/or charging cost minimization. Charging was considered for home and workplace profiles, and simulations used time of use and real time pricing schemes. The proposed algorithm was able to provide more optimal results, in terms of a higher level of customer satisfaction and network performance, minimizing network power loss, voltage deviation, and power consumption.

This paper provided another example of the integration of two-levels or stages of optimization, with each stage optimizing different objective functions with different constraints, and ultimately having local aggregators developing optimized EV charging schedules which respect the constraints of the distribution system they are attached to. A limitation of this work is that following the end of the second stage optimization process, there is no opportunity for the algorithm to attempt to re-optimize the schedules or power distribution if the network constraints were not adhered to by the EV aggregator agents. Due to this, there is potentially the requirement to have the system operator manually re-start the algorithm until a feasible solution is found and therefore increasing the computation time (actions to be taken in the event of an infeasible final solution being found are not described in the paper). This thesis proposes a method influenced by this paper, which, as described in Section 2.6.2, uses two-levels of optimization to schedule the charging of EVs in multiple parking lots while ensuring the distribution network constraints are respected. However, the method proposed in this thesis includes a process for the algorithm to re-optimize the EV charging schedules and power distribution between the parking lots if the network constraints are violated, by integrating the two levels of the optimization algorithm together: placing one level inside of the other, instead of having their completions be sequential. As described in Section 2.6.2, this will allow for the verification of the network constraints as part of the upper-level optimization process since the constraints will be integrated into that optimization process. While this thesis does not consider residential EV charging, it will demonstrate its scalability of the model on more than one distribution system size, one of which is a 33-bus distribution system (though the simulation parameters, problem setup, and number of EVs are different).

2.6.4 A multi-objective model for the day-ahead energy resource scheduling of a smart grid with high penetration of sensitive loads (2016) [24] and Multi-objective parallel particle swarm optimization for day-ahead Vehicle-to-Grid scheduling (2013) [25]

These two papers, with the same first author, influenced this thesis by proposing methods which integrated resource scheduling problems involving EV charge scheduling with metaheuristics and HPC. At the time of research for this thesis, these were the only papers found by the author which explicitly combined all three topics.

These two papers present multi-objective PSO-based algorithms which perform day-ahead resource scheduling for grids with V2G-capable EVs and distributed generation. The key feature of both of these papers is the use of parallel (distributed and multicore) computing to reduce the computation times of the algorithms.

In the 2013 paper [25], the authors develop an algorithm for an aggregator to schedule the charging and discharging of EVs to support day-ahead energy management, which has two objective functions: one to minimize operational costs for the energy resources, and the second to maximize profits of the EV owners when they provide energy to the grid. The objective functions are subject to the constraints of power flow active and reactive power balance, bus voltage magnitude and angle, line thermal limits, transformer limits, generation and supply limits, EV technical limits, and battery limits. In the fitness function, Pareto weights were added to the two objectives, and a penalty term of value 0-1000 was used to identify solutions with constraint violations. The authors also used the direct repair method to correct solutions which violates constraints [25].

PSO was used as the optimization algorithm, with Gaussian mutation weighted parameters. The authors had 501 total sets of Pareto weights for the fitness function, and a PSO using each set of weights was run to find the most Pareto optimal solution [25]. These 501 PSOs were executed in parallel: using MATLAB (R2012a) on a workstation with two six-core Intel® Xeon® X5650 processors and on an Apple® Mac Pro® workstation with two six-core Intel® Xeon® processors, each of the 24 cores evaluated one or more 10 particle, 500 iteration PSOs with a different set of weights. The parallel execution provided an approximate speedup of 11 times (per Section 2.5). The case study simulation used a 33-bus distribution system with 66 distributed generation sources, 10 energy sources, and 1800 EVs. The authors demonstrate that their algorithm is able to produce the desired optimization results for the day-ahead V2G scheduling, minimizing cost and maximizing income simultaneously depending on the chosen Pareto weights.

The 2016 paper [24] is somewhat similar to the first, where the authors developed an algorithm for a Virtual Power Player (similar to the network operator) to schedule the day-ahead activities in the smart grid, with two objectives: to minimize operational costs of the system, and to maximize the minimum system reserve. The functions are subject to the same constraints as [25], plus system net reserve and storage system battery limits. A two-stage optimization method was developed, where the first stage uses Mixed Integer Linear Programming to solve the objective functions, and the second stage uses PSO with different Gaussian weighted parameters to solve the full problem including direct repair of any solutions.

Similar to [25], in the 2016 paper [24] the authors had 100 total sets of Pareto weights for the fitness function, and a MILP-PSO algorithms using each set of weights was run to find the most Pareto optimal solution [24]. These 100 MILP-PSO algorithms were executed in parallel: using MATLAB (R2014a) and TOMLAB on a HPC cluster with six machines for a total of 42-cores, each of the 42 cores evaluated one or more 10 particle, 1000 iteration PSOs with a different set of weights. The parallel execution provided an approximate speedup of 24 times (per Section 2.5). This parallel process is shown in Figure 2.5. The case study simulation used a 180-bus distribution system with 116 distributed generation sources, 7 energy storage sources, and 1000 EVs. The authors demonstrate that their algorithm is able to produce the desired optimization results, and performed more optimally than a MILP or PSO-only based algorithm to solve the same problem.

Both papers abstracted the specific behaviour of individual EVs in their results, showing the charging or discharging behaviour of all EVs as part of their total contribution to the algorithm objectives.

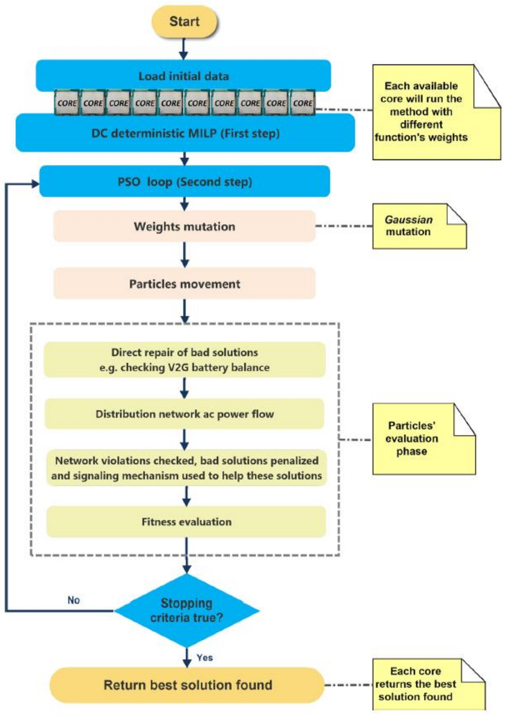

Start
Load initial data
CORE CORE CORE CORE CORE CORE CORE CORE CORE CORE
Each available core will run the method with different function's weights
DC deterministic MILP (First step)
PSO loop (Second step)
Weights mutation
Gaussian mutation
Particles movement
Direct repair of bad solutions e.g. checking V2G battery balance
Distribution network ac power flow
Particles' evaluation phase
Network violations checked, bad solutions penalized and signaling mechanism used to help these solutions
Fitness evaluation
No
Stopping criteria true?
Yes
Return best solution found
Each core returns the best solution found

Figure 2.5 Flowchart of the weighted sum parallel PSO method (copied from Figure 2 of [24])

Both papers provided examples of the parallelization of a PSO-based process involving EV charge scheduling, where PSOs with different parameters were evaluated in parallel to reduce the computation time. While the day-ahead resource scheduling problems addressed in both papers are quite different from the EV charge scheduling problem addressed in this thesis, the parallelization of individual PSOs influenced the parallelization method used in this thesis, where individual (inner level) PSOs are evaluated in parallel as part of the fitness function evaluation of another (outer level) PSO.

As well, while not necessarily a limitation to the effectiveness of the overall algorithms proposed in the two papers, the speedups of the parallelized algorithms were likely limited by the number of processor cores used in each case. Per Amdahl's Law, the speedup of a parallelized algorithm is impacted by the number of processors used to compute the parallelized portion of an algorithm. The two papers experienced 11 and 24 times speedups, using 12 and 42 processor cores, and both parallelized algorithms took 30 minutes or more to compute. For this thesis, in order to complete the EV charge scheduling algorithm calculations in under 15 minutes, a large speedup was required, and therefore a large number of HPC cluster processor cores (192) were used to provide that speedup.

2.7 MATLAB® Parallel Computing Toolbox™

MATLAB® (referred to throughout this document as MATLAB) is a programming and computing environment used across many engineering and other technical fields [146]. One of the toolboxes it offers is the Parallel Computing Toolbox™ (referred to throughout this document as the Parallel Computing Toolbox), which enables the use of multicore, GPU, and distributed computing with MATLAB applications [147], [148], [149]. It contains functions for specific task and worker communication and control through SPMD functions like `spmdSend`, which works somewhat similarly to MPI message passing functions, and the function `parfor`, which is similar to the OpenMP parallel for loop [146]. MATLAB also offers the Parallel Server™, which enables the scaling of programs like those made with the Parallel Computing Toolbox to computer clusters and clouds [150]. This thesis work used a HPC cluster equipped with MATLAB, the Parallel Computing Toolbox, and MATLAB Parallel Server.

2.7.1 MATLAB parfor

The MATLAB `parfor` function is a function from the Parallel Computing Toolbox which executes the code within its loop body, in an order-independent and parallel fashion, on the workers available in the current computing cluster or multi-core computer (the parallel pool) [146], [151]. According to the 2009 paper by Sharma and Martin [146], the loop iterations are assigned dynamically to the workers after dividing the iteration range into "pieces," enabling a potentially shorter runtime than a typical static assignment and a more even distribution of the workload across the workers in the parallel pool. However, the current Parallel Computing Toolbox online documentation does not specify the type of scheduling or method it uses to distribute the iteration range, so it is not apparent if the work distribution is done statically (with a fixed iteration range being assigned to each worker, as is the case with the static scheduling scheme of OpenMP [123]), dynamically, or with another method [151], [152].

More specific details about the implementation and use of the `parfor` function within MATLAB can be found within MATLAB online documentation and the paper by Sharma and Martin [146], [151], [152], but for this thesis it is critical to understand that the function enables task parallelism by having all available workers in the cluster work independently, simultaneously, on different iterations of the code contained within the specified loop. Identification of specific cluster workers to act as the master or worker(s), and to send and receive specific data, is not required as it would be with the use of a `spmd` block: the division of work, transportation of data to and from workers, communication between workers, and other details are hidden from the user by MATLAB. According to [21], when the first instance of the `parfor` is encountered within the code, the MATLAB Parallel Server job scheduler takes the source file code and distributes it to the configured cluster (parallel pool) nodes. This enables complex code, which uses `for` loops to complete complex calculations requiring long iteration times, to be simply parallelized with the use of the single `parfor` function.

2.8 MATPOWER

MATPOWER is an open-source package of MATLAB files and functions which allow users to simulate and solve power flow and optimal power flow problems [153], [154], [155]. The software tool allows the user to select a test case, whether included within the MATPOWER package or with defined parameters of their own, to set or modify any parameters within the case and the simulation function, and to access the solved results of their simulation after execution. Modifications to data in the provided transmission and distribution system test cases can be made easily, like changing the real and reactive power demands to reflect an increased load or the addition of a distributed generator.

The function `runpf`, in its default state, is the standard Newton-Raphson method-based AC power flow solver. Using the “traditional formulation” of the AC power flow problem as outlined in [155], the function solves for the unknown voltage quantities and generator real and reactive power injections, and does not consider any generator, branch flow, or voltage magnitude limits. By solving a selected test case with the function, a user can use the function to determine if the solution is initially feasible (i.e., does the solution converge), if any voltage, branch, power, or other limits have been breached, and its total power loss across all branches.

Full details on the use and properties of MATPOWER, including all available functions, data file formatting and function options, can be found within the user's manual [155]. This thesis worked used the MATPOWER `runpf` function to run power flow simulations, and four of the included test cases to develop the distribution system test cases used in this work (see Section 3.6.2).

2.9 Summary

This chapter provided a literature review as well as background information on the major topics related to this thesis. Section 2.1 discussed power systems and related concepts including distribution systems, smart grids, and power system optimization problems. Section 2.2 outlined the six major EV optimization problems including EV charge scheduling optimization. Section 2.3 discussed the topic of EV charge scheduling in more detail, including charging technologies, centralized and decentralized optimization, optimization objective functions, the inclusion of power flow in the problem, the inclusion of multiple parking lots in the problem, and the problem abstraction found in literature. Section 2.4 discussed the three main optimization methods, the use of metaheuristics with EV charge scheduling, and the PSO method. Section 2.5 discussed the main concepts of HPC, its application to this thesis, and PSO parallelization methods found in literature. Section 2.6 provides a short review or summary of five papers which influenced the design methodology used in this thesis. Section 2.7 provides a short description of the MATLAB Parallel Computing Toolbox and `parfor` function, and Section 2.8 provides a short description of the MATPOWER software package.

Section 3

Methodology and Design

This chapter outlines the research methodology and design used to develop the multiple parking lot EV charge scheduling optimization model. The first section discusses the assumptions and basic decisions made to determine the scope and limitations of the models used in this research. The second section describes the adaptation of the single parking lot optimization model from [40] and its parallelization as Algorithm 1. The third section describes the design of the centralized multiple-parking lot optimization model and its parallelization as Algorithm 2 (and Algorithm 3). The fourth section describes the design of the decentralized multiple-parking lot optimization model and its parallelization as Algorithm 4 (and Algorithm 3). The next sections describe the PSO parameters chosen for the models, and the adaptation and creation of the parking lot profiles and chosen distribution system test cases.

3.1 Scope and Limitations

This section describes the decisions, assumptions, and limitations made about how the EV charge scheduling problem is approached in this thesis.

3.1.1 EVs and Charging

In a general sense, the two EV sub-types of PHEV and BEVs are included in the scope of this problem, as they both have the capability to connect (plug-in) to the power grid, though they are not identified or treated differently when considered as loads in the problem. There is no difference in their identification within in the Parking Lot Profiles in Section 3.6.1.

For this thesis, only unidirectional V2G (also known as G2V) charging will be considered, where EVs function as stochastic and variable loads within the grid. This limitation is imposed in order to decrease the complexity of the proposed research objectives, and bring the research more directly in-line with related literature [10], [36], [40], [84] and the current state of EV charging. V2G technologies are not widely adopted, with only a limited number of car manufactures have developed V2G charging technologies [156], [157], and the potential for delay of the widespread adoption of bidirectional V2G technologies due to a reduction in EV user satisfaction due to the risk of accelerated battery degradation from continuous charging and discharging, frequent and expensive battery replacement [40], [64] and the hesitance to “give up” their power in case of unplanned departures [65]. Therefore, the problem of unidirectional charge scheduling is more relevant to the current state of EV charging.

A number of parameters were selected due to the use of the single parking lot optimization model presented in [40] as the basis for the single parking lot optimization model developed in this thesis. Each parking lot (or EV charging station as described in Section 2.3.1.2) has the same number of EVs, and the same base transformer limit. Each parking lot has as many charging ports as there are EVs: only as many EVs as can be supported by each parking lot are scheduled for charging within it. Each parking lot charging port supported only one type of charger, or charging level: Level 2 AC charging with the SAE J1772 connector, which is the most common type of charger currently available to public use in Canada [66], [69].

3.1.2 Distribution System

For this thesis, the scope of the power grid considered in the problem is limited to a distribution system network, which represents a commercial area, i.e. the "downtown" area of a city which contains one or more parking lots within it, and does not include any residential neighbourhood nodes. No distributed generation or renewable energy sources, like solar or wind turbine generation, or energy storage resources have been included in the system. The four distribution system scenarios developed for this thesis are described in Section 3.6.

As described in Section 2.3.3.2, power flow equations are used to verify network conditions and confirm that solutions will not violate system limitations. For this thesis, only voltage magnitude limitations are used to indicate if a solution is realistic, i.e. if it will operate within the physical or other limitations of the distribution system.

3.1.3 Centralization with a Single Aggregator

For centralized optimization within this thesis, a single central aggregator or distribution network operator will be performing all the optimization required for the centralized optimization of one to n_{pl} parking lots. For decentralized optimization, a central aggregator provides information to local aggregators at each parking lot, and these local aggregators perform their optimizations independently and provide limited information back to the central aggregator.

To perform the centralized optimization, the aggregator is assumed to have prior perfect knowledge of all optimization parameters: number of EVs, number of parking lots, the charging profile for each EV (from the parking lot profiles), charging rate limit, transformer limits for each parking lot, and location of parking lots within the distribution system. The distribution system on which the aggregator operates and controls is assumed to be a smart grid, with the requisite communications and other control architecture required to enable the aggregator to communicate with all parking lots and EVs, and to dictate their charging.

The aggregator considers a set of parking lots P_l with n_{pl} number of parking lots, each with $numEV$ number of EVs with index i (EV_i), a $numT$ total number of time intervals t of a given duration Δt from the set T, a charging station port charging rate limit $limChr$ in kW, and the price of purchasing electricity $ElecPrice$ at every timeslot t ($ElecPrice_t$). Each parking lot p is located at a bus b from the set of all system buses B. The aggregator also considers the transformer limit $limTf_p$ for each parking lot p, from the set of all transformer limits $limTf$. These transformer limits are set to a specific value in advance, or changed during the optimization process.

The charging demands and basic behaviour of each EV in each parking lot are known in advance to the aggregator, being provided in the associated Parking Lot Profile. Each profile lists each EV's arrival time to the parking lot arr_i, departure time dep_i from the parking lot, and its total charging demand (desired consumption) in kWh dem_i. For this research, the duration of timeslot t, or Δt, is 15 minutes (0.25 hours), though this value is ultimately arbitrary to the optimization model itself, and only the total number of timeslots $numT$ that the EVs are present in the parking lots is relevant for the optimization. The time intervals are the same across the parking lots: they each have the same total number of intervals, and t_1 in one parking lot is the same for all other lots.

3.1.4 Parallelization

The parallelization of each model will be completed in MATLAB, and the programs executed on either a single multicore workstation (multicore computing) or a computer cluster with multiple multicore workstations (multicore and distributed computing).

For the centralized models which have a single aggregator conducting all computations, this single aggregator is considered to encompass the entire multicore workstation or computing cluster. All compute nodes (or processors, or workers) of the HPC system being used in the model's execution are controlled by the aggregator.

For the decentralized models, which have a single central aggregator collecting and providing information to multiple local aggregators, each aggregator is considered to be one compute node in the cluster, with the central aggregator and a single local aggregator of one parking lot p sharing the same compute node. Different portions of the HPC system represents the different independent aggregators.

3.2 Single Parking Lot Optimization

The single parking lot optimization model was based on the single parking lot optimization model presented in [40], and its general problem formulation, objective function, constraints, and EV and parking lot parameters were reproduced for use in this thesis. As discussed in Section 2.6.1, the original paper used a hybrid heuristic fuzzy PSO, with a proportion-based assignment method to generate the initial population, and a "dynamic resource allocation system" to consider EVs with and without charging appointments. For this research, a much simpler model was developed: the PSO was used without any hybridization or other algorithm combinations, and all EV parameters were considered to be known in advance by the central aggregator.

When provided the charging profile behaviour for all EVs EV for all timeslots t in the single parking lot ($n_{pl} = 1$), the single parking lot optimization model will determine a single optimized charging schedule for all EVs in the entire parking lot, across all timeslots.

Per the problem formulation in Section 4 of [40], each individual EV EV_i in a given parking lot arrives just before its arrival time arr_i, departs immediately following dep_i, and will be fully charged to its desired demand (desired consumption) dem_i when it leaves the parking lot. Therefore, EV_i can only charge between the arrival and departure times arr_i and dep_i. This behaviour is shown in equation (3.1):

$$A_i^t = \begin{cases} 1, & arr_i \leq t \leq dep_i \\ 0, & otherwise \end{cases} \quad \forall i \in EV \tag{3.1}$$

where the availability to charge A_i^t of EV_i in a given parking lot at time t is defined as 1 (available to charge and therefore have its charging rate set through optimization) or 0 (unavailable to charge).

As will be discussed in Section 3.2.1, the single parking lot optimization model will be solving for the solution vector $\boldsymbol{X}$, which is denoted per equation (3.2):

$$\boldsymbol{X} = \left\{x_1^{arr_1}, \dots, x_1^{dep_1}, x_2^{arr_2}, \dots, x_2^{dep_2}, \dots, x_i^t, \dots, x_{numEV}^{arr_{numEV}}, \dots, x_{numEV}^{dep_{numEV}}\right\} \tag{3.2}$$

where x_i^t represents a proportional quantity of EV_i's charging demand (consumption) being charged during, or assigned to, timeslot t. For all timeslots where the EV_i is available to charge ($A_i^t = 1$) an electric quantity of value x is assigned. For use with the PSO algorithm, the solution vector $\boldsymbol{X}$ is equivalent to the PSO particle position vector term $\boldsymbol{x}$ from equations (2.8) and (2.9).

Per [40], the single parking lot optimization model finds $numX$ number of solution variables:

$$numX = \sum_{t=1}^{numT} \sum_{i=1}^{numEV} A_i^t \tag{3.3}$$

where $numT$ is the total number of time intervals t and $numEV$ is the number of EVs in the parking lot. In a case with one parking lot, ten timeslots and 20 EVs, a maximum of 200 solution variables must be solved.

3.2.1 Objective and Fitness Function

As was the case for the objective function presented in [40], the objective function of this single parking lot optimization model was to minimize the cost of electricity purchased from the utility to meet the charging demands of all EVs in the parking lot, considering the variation in the electricity price in each timeslot:

$$\text{minimize } C = \sum_{i=1}^{numEV} \sum_{t=arr_i}^{dep_i} x_{d}{}_{i}^{t} ElecPrice_t \tag{3.4}$$

where $x_{d}{}_{i}^{t}$ is the "decoded" version of the proportional charging demand quantities x_i^t, which sum to 100% for each individual EV i (equations (3.5) and (3.6) below), and $ElecPrice_t$ is the electricity price at time interval t. Each term x_i^t has a value from $[0,1]$, and therefore must be "decoded" to its proper kWh value for the final solution.

Similar to the constraints in the source paper [40], the constraints for this objective function were as follows:

$$x_{d}{}_{i}^{t} = \left(\frac{x_i^t}{\sum_{t=arr_i}^{dep_i} x_i^t} \right) dem_i \quad \forall i \in EV, \forall t \in T \tag{3.5}$$

$$\sum_{t=arri_i}^{dep_i} x_{d}{}_{i}^{t} = dem_i \quad \forall i \in EV \tag{3.6}$$

$$x_{d}{}_{i}^{t} \leq limChr \quad \forall i \in EV, \forall t \in T \tag{3.7}$$

$$\sum_{i=1}^{numEV} x_{d}{}_{i}^{t} \leq limTf_1 \quad \forall t \in T \tag{3.8}$$

$$0.9 \leq |V_b| \leq 1.1 \ \forall b \tag{3.9}$$

where $limChr$ is the charging rate limit, $limTf_1$ is the transformer capacity limit ($limTf$) at parking lot 1, and $|V_b|$ is the bus voltages magnitude at bus b. The constraints are that the charging demand at each charging port could not exceed the charging rate limit $limChr$ (equation (3.7)), and the total charging demand across all EVs (and thus all charging ports) at time t could not exceed the given transformer capacity limit $limTf$ (equation (3.8)).

In addition to the constraints on the charging within the parking lot, the feasibility of the solution's transformer limit was tested through the conduction of a power flow simulation on a selected distribution system test case, and analysis of the voltage magnitudes at all buses. The constraint therefore imposed was that all distribution system bus voltages magnitudes $|V_b|$ across all buses b had to remain within the minimum and maximum voltage magnitude of 0.9 and 1.1 per unit (p.u.), equation (3.9). The power flow simulation was conducted by selecting a bus for the parking lot to be located at, adding the maximum possible additional active power demand at any time interval ($limTf_1$, in kW) to that bus, and then running the power flow simulation to find the resulting bus voltage magnitudes.

In order to incorporate the constraints with the objective function and provide a "relative measure of the optimality of feasible solutions," [81] a fitness function is required. The fitness function created for this model was composed of a penalty term and a cost term, similar to that used in [81].

3.2.1.1 *Penalty Term*

For this single parking lot optimization model, the penalty term ρ is used to penalize the candidate solutions which violated one or more of the constraints listed in equations (3.5) to (3.8), i.e. the infeasible solutions. The penalty term equations are:

$$\rho = d_{dem} + d_{limChr} + d_{limTf} \tag{3.10}$$

$$d_{dem} = \sum_{i=1}^{numEV} \left(dem_i - \sum_{t=arri_i}^{dep_i} x_{d}{}_i^t \right)^2 \tag{3.11}$$

$$dd_{limChr}{}_i^t = \begin{cases} x_{d}{}_i^t - limChr, & x_{d}{}_i^t > limChr \\ 0, & x_{d}{}_i^t < limChr \end{cases} \quad \forall i \in EV, \forall t \in T \tag{3.12}$$

$$d_{limChr} = \sum_{i=1}^{numEV} \sum_{t=arr_i}^{dep_i} \left(dd_{limChr}{}_i^t \right)^2 \tag{3.13}$$

$$dd_{limTf}{}^t = \begin{cases} \displaystyle\sum_{i=1}^{numEV} x_{d}{}_i^t - limTf_1, & \displaystyle\sum_{i=1}^{numEV} x_{d}{}_i^t > limTf_1 \\ 0, & \displaystyle\sum_{i=1}^{numEV} x_{d}{}_i^t < limTf_1 \end{cases} \quad \forall t \in T \tag{3.14}$$

$$d_{limTf} = \sum_{t=1}^{numT} \left(dd_{limTf}{}^t \right)^2 \tag{3.15}$$

where ρ is the penalty term, d_{dem} is the unmet demand (desired consumption) term, d_{limChr} is the charging limit violation term, d_{limTf} is the parking lot transformer limit violation term, $dd_{limChr}{}_i^t$ is the term for when EV i's demand in time interval t exceeds the charging rate limit $limChr$, and $dd_{limTf}{}^t$ is the term for when the collective demand for all EVs charging in one time interval t exceeds the transformer limit $limTf_1$.

The penalty term in equation (3.10) is calculated by summing the squares of the constraint violations together. The unmet demand (desired consumption) term d_{dem} in equation (3.11) calculates the difference between the desired demand or total energy consumption of EV_i and the actual demand provided by the solution, squares the term, and then finds the sum of the squared unmet demand across all EVs in the parking lot. The charging limit violation term d_{limChr} in equation (3.13) finds all instances of an EV's demand in a time interval exceeding the charging rate limit (equation (3.12)), finds the difference between the exceeded rate and the limit, squares the result, and then sums the values across all EVs in all time intervals together. The parking lot transformer limit violation term d_{limTf} in equation (3.15) finds all instances of all EVs in a time interval exceeding the transformer limit (equation (3.14)), squares the difference between the exceeded rate and the transformer limit, and then sums the values across all time intervals together.

These penalty terms were all squared in order to ensure that the values were always positive, and to ensure they would remain fairly proportionate in their purpose as penalty terms. If the difference was very large, i.e. 25, the squared total would be 625, and if small, i.e. 0.25, the squared total would be 0.0625: the larger the constraint violation value, the larger the penalty term, and the higher its importance in the outcome of the overall fitness function.

One note in this design is that the penalty terms are not normalized to the same units, so there is no prevention of bias where penalties with potentially large ranges of values would outweigh penalty terms with a smaller range of values [158]. While the terms of equations (3.12) and (3.14) could have been multiplied by Δt to convert all terms to kWh, with interval value of $\Delta t = 15$ mins (0.25 hours), it was found that the lack of inclusion did not prevent the proper working of this model. See Section 3.3.2.1 for more information.

The total unmet demand across all EVs in the parking lot is found using:

$$demu_p = \sum_{i=i}^{numEV} \left| dem_i - \sum_{t=arri_i}^{dep_i} x_{d_i}^{t} \right| \tag{3.16}$$

where $demu_p$ is the total unmet demand across all EVs in parking lot p. Due to the behaviour of the PSO, finding candidate solutions which had no violation of any constraints (i.e. $\rho = 0$) was extremely difficult. Therefore, in order for a solution to be considered feasible, the total unmet demand across all EVs in the parking lot p (from equation (3.16)) had to be less than one watt, and the total penalty term ρ less than 1×10^{-8} kWh^2, allowing for some level of tolerance. This is illustrated in the combined fitness function.

3.2.1.2 *Cost Term*

The cost term C of the fitness function represents the optimization objective of the objective function equation (3.4), the cost of electricity purchased from the utility. It is repeated in equation (3.17) below for clarity:

$$C = \sum_{i=1}^{numEV} \sum_{t=arr_i}^{dep_i} x_{d_i}^{t} ElecPrice_t \tag{3.17}$$

3.2.1.3 *Combined Fitness Function*

Following the calculation of the penalty and cost terms for the candidate solution $\boldsymbol{X}$, the fitness function $F(\boldsymbol{X})$ of that candidate solution can be calculated per equation (3.18):

$$F(\boldsymbol{X}) = \begin{cases} 0 + \dfrac{1}{1+\rho}, & \rho \geq 1 \times 10^{-8} \vee demu_p \geq 1 \times 10^{-3} \\ 1 + \dfrac{1}{1+C}, & \rho < 1 \times 10^{-8} \wedge demu_p < 1 \times 10^{-3} \end{cases} \tag{3.18}$$

where ρ is the penalty term from equation (3.10), C is the cost term from equations (3.4) and (3.17), and $demu_p$ is the total unmet demand across all EVs in parking lot p from equation (3.16). This fitness function equation has been defined so that all infeasible candidate solutions (where $\rho \geq 1 \times 10^{-8}$, and/or $demu_p \geq 1 \times 10^{-3}$) always have a fitness in the range $(0, 1)$, and all feasible candidate solutions will have a fitness above 1, in the range of $(1, 2)$. This provides an easy way for the PSO to evaluate the quality of the solution (is it feasible or not) [81], and an easy visual representation of the behaviour of the PSO as it moves from infeasible to feasible solutions over its iterations. As the solution becomes more optimal, i.e. the cost decreases or is minimized, the fitness function value will increase, or is maximized.

3.2.2 Metaheuristic Choice

The only metaheuristic used within the models for this research is the PSO. The PSO version used is the "standard" [113], [131] PSO discussed in Section 2.4.3.2, with a constant inertia ω weight included in the velocity update equation of the algorithm. The decision variables used within the objective functions of this research are all continuous variables.

3.2.3 Algorithm Steps

This model used the basic PSO algorithm as described in Section 2.4.3.2 to compute feasible EV charging schedules for the single parking lot. The algorithm steps and flowchart for the parallelized version of this are shown in Figure 3.1 and Algorithm 1.

For the non-parallelized model described in Sections 3.2 to 3.2.1.3, the flowchart of the model follows the standard PSO flowchart in Figure 2.1. Firstly, after receiving all the requisite inputs to the algorithm like the EV charging profiles, the aggregator initializes the PSO position and velocity vectors, with n_p number of particles with $numX$ variables each. Then, the fitness function in equation (3.18) is evaluated for each candidate solution. The best personal and global positions are updated, new particle position and velocities are updated, and so on, until the process is terminated once the maximum number of PSO iterations n_{itr} has been reached. The best global position, in this case the most optimal charging schedule from a candidate solution vector $\boldsymbol{X}$, is taken as the most optimal final result for this model.

However, instead of the finding of this result denoting the end of the algorithm, the second-last step in the process is the checking of the feasibility of the solution's transformer limit through the conduction of a power flow simulation. The power flow was conducted with the chosen distribution system test case, to find if the constraints in equation (3.9) were breached. If this was the case, the final optimized solution was determined to be not feasible. If not, the solution was determined to be feasible. After this check, the algorithm ended.

3.2.4 Parallelization

When metaheuristics like the PSO algorithm are used to solve the EV charge scheduling problem with a large number of particles and large number of iterations, the computation time may increase to the point of being impractical [21], [96]. In order to reduce this runtime, the problem algorithm can be parallelized in order to benefit from the use of HPC.

In the PSO algorithm, the fitness function evaluation of each particle can be performed in parallel, because the evaluation of each candidate solution does not depend on the evaluation or outcome of any other candidate solution. Since this is true of this single parking lot optimization algorithm, the algorithm can be parallelized in that portion of the algorithm: step 2 of the PSO (Figure 2.1), the evaluation of the fitness function for each particle. This parallelization method is known as coarse-grained or synchronous parallelization, and is most similar to the method presented in [130]. This parallelized algorithm process is shown in Figure 3.1 below.

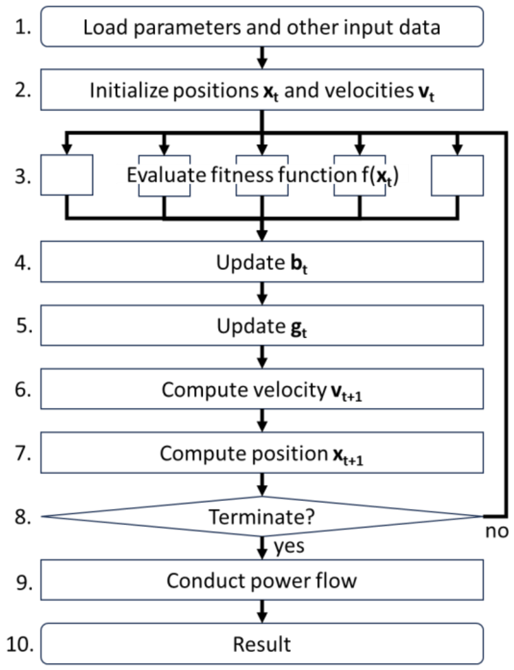

1. Load parameters and other input data
2. Initialize positions $\mathbf{x}_t$ and velocities $\mathbf{v}_t$
3. Evaluate fitness function f($\mathbf{x}_t$)
4. Update $\mathbf{b}_t$
5. Update $\mathbf{g}_t$
6. Compute velocity $\mathbf{v}_{t+1}$
7. Compute position $\mathbf{x}_{t+1}$
8. Terminate?
no
yes
9. Conduct power flow
10. Result

Figure 3.1 Single Parking Lot Optimization Algorithm Flowchart

Figure 3.1 shows that steps 1 to 2 and 4 to 10 are performed sequentially, by the same single process, and that step 3 of the process, the evaluation of the fitness function, is performed in parallel for each particle of the PSO. In step 3, all processes which perform the parallel fitness function evaluations receive the same parameters and initial velocity and position data. The fitness value for each particle is then evaluated independently, simultaneously with the other particles, by the parallel processes. Following this, the fitness function results are all passed back to the single main process, which then performs the updates for the best personal and global positions, and updates the velocity and positions for the next iteration. If the maximum number of PSO iterations has been completed, the algorithm terminates and presents the final result; if not, steps 3 to 8 are repeated, with step 3 always occurring in parallel.

The method used for this parallelization was the SPMD method, where a single program is executed (i.e. one program runs the algorithm from Figure 3.1) but each processor in the HPC cluster executes the same instructions on their own assigned data, i.e. their candidate solution(s) [21]. The program was coded in MATLAB, and the parallelization was performed through the use of a `parfor` loop around the function which evaluated the fitness of a single particle. Per Section 2.7, the `parfor` function or loop enables task parallelism by having all available worker processors in the cluster work independently, simultaneously, on different iterations of the code contained within the specified loop.

The `parfor` loop was coded to perform the evaluation of the fitness function of each particle (from 1 to n_p), and therefore executed each iteration (for particle or iteration $j = 1, 2, \dots, n_p$) in parallel on the workers in the HPC cluster (the parallel pool [151]). The function which evaluates the fitness of a single particle or iteration intakes all data needed to perform the fitness function evaluation, and outputs only the particle's fitness value $F(\boldsymbol{X})$, cost C, unmet demand $demu_p$, and penalty value ρ, in order to limit the data sent between workers and therefore the overhead. The pseudocode of the parallelized algorithm with the `parfor` function is shown in Algorithm 1.

Algorithm 1. Single Parking Lot Optimization Algorithm Pseudocode (Parallelized)

```
1:   generate EV parking lot data from parking lot profile
2:   generate problem and PSO parameter data
3:   generate population of $n_p$ initial candidate solutions (particles)
4:   initialize PSO position and velocity vectors
5:   while termination criteria is false (iteration < $n_{itr}$)
6:       parfor each candidate solution ($j = 1, 2, \dots, n_p$)
7:           evaluate the fitness of the candidate solution
8:       end parfor
9:       for each candidate solution ($j = 1, 2, \dots, n_p$)
10:          update the local best position and fitness
11:          update the global best position and fitness
12:      end for
13:      for each candidate solution ($j = 1, 2, \dots, n_p$)
14:          compute the new particle velocity and enforce velocity limitations
15:          compute the new particle position and enforce velocity limitations
16:      end for
17:  run power flow and check voltage feasibility constraints of best solution
18:  return best solution
```

The sequential and parallelized portions of the single parking lot optimization model are clearly shown with Figure 3.1 and Algorithm 1: only step 3 in the flowchart, or lines 6 to 8 in the algorithm, are parallelized, with all other parts of the model or algorithm remaining sequential. Per Section 2.5.5, this means that if there are n worker processors in the HPC cluster, one worker performs all sequential portions of the algorithm, including sending and receiving the required data to and from the other worker processors, and all n worker processors perform the parallelized portion together in parallel. Per [146], since MATLAB `parfor` provides generally even distribution of the workload across the n workers in the parallel pool, it is expected that each worker process performs approximately ${}^{n_p}/_{n}$ fitness function evaluations, and that a speedup in line with Amdahl's Law (see Section 2.5.5) should be provided.

While the `for` loops in lines 9-12 and 13-16 of Algorithm 1 could also be parallelized with `parfor`, the calculations performed in these sections are very simple and quick to be performed by a single worker in MATLAB. They are not parallelized in order to reduce the communication overhead created when using `parfor`, as the trade-off between work performed in parallel and runtime is not worth the use of the function.

3.3 Multi-Parking Lot Optimization

The optimization of multiple parking lots is a more complex problem, requiring the coordination of the parking lots by system operators and other aggregator entities to ensure that the power distribution system is operated within its physical limitations, while simultaneously optimizing one or more objectives [44].

For this thesis, in order to optimize multiple parking lots with multiple EVs in each parking lot in a centralized manner, the element that was selected to be coordinated between the parking lots by the single central aggregator was the transformer capacity limit of each parking lot. The explanation is as follows: for a set of parking lots P_l with n_{pl} number of parking lots, there is a defined amount of power, or generator capacity, P_{total} that is available to the aggregator to divide among the parking lots (similar to [143]). This amount of power has been set to be linearly proportional to the number of parking lots in the problem – it is the base transformer limit for one parking lot multiplied by the number of parking lots:

$$P_{total} = limTf_{base}n_{pl} \tag{3.19}$$

where P_{total} is the total power capacity available to the aggregator, $limTf_{base}$ is the base transformer limit for one parking lot, and n_{pl} is the number of parking lots. The power is available during all time intervals t, and does not change with time.

This total power capacity P_{total} is divided up between the parking lots by the aggregator, and for each parking lot p their portion represents the maximum possible total power capacity available at every time interval. This portion therefore represents the new transformer capacity limit for that parking lot: $limTf_p$, from the set of transformer limits for all parking lots $\boldsymbol{limTf}$:

$$\boldsymbol{limTf} = \left\{limTf_1\,, limTf_2, \dots, limTf_p, \dots, limTf_{n_{pl}}\right\} \tag{3.20}$$

where $limTf_p$ represents the maximum transformer capacity of parking lot p (across all time intervals t), in kW. The aggregator is therefore dividing the total capacity P_{total} into the transformer limits for all parking lots. The aggregator then takes the new transformer limits per each parking lot, and optimizes the charging schedule for all EVs in each parking lot. The optimization for each individual parking lot with the new transformer limit is based on the single parking lot optimization method developed in Section 3.2.

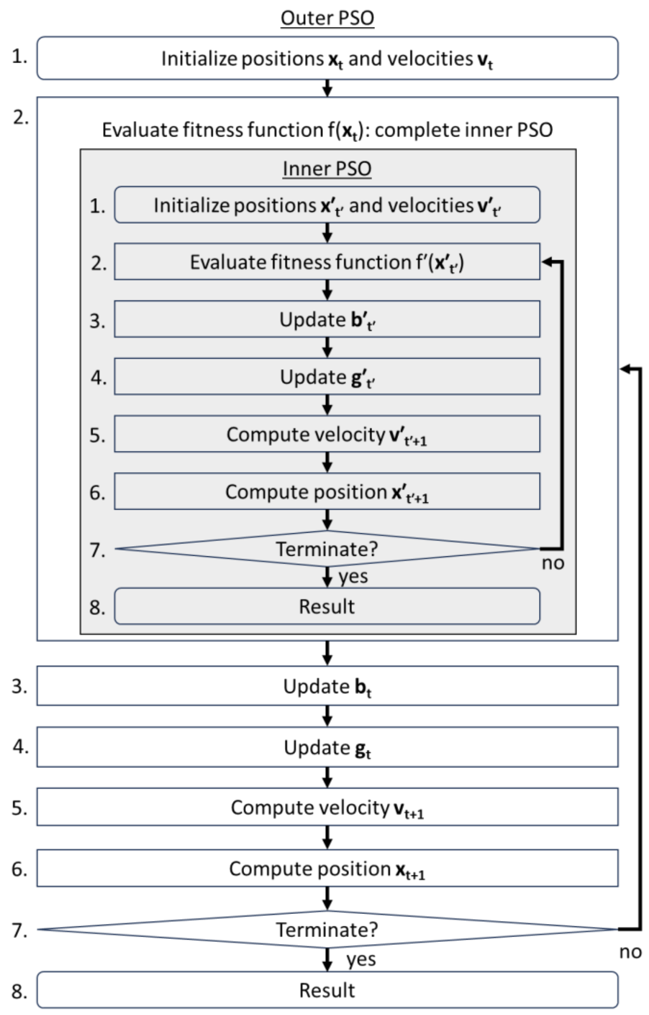
The general problem formulation for this multiple parking lot optimization model is as follows: when provided the charging profile behaviour for all EVs EV for all timeslots t in n_{pl} parking lots, the multiple parking lot optimization model will determine an optimized transformer limit for all parking lots in the problem, and an optimized charging schedule for all EVs in all parking lots with their new optimized transformer limits, across all timeslots.

As will be discussed in Section 3.3.2, the multiple parking lot optimization model will be solving two objectives, and therefore has a more complex problem formulation than the single parking lot optimization model. Using the model from Section 3.2 as the basis, the multiple parking lot optimization model will be solving for the solution vector $\boldsymbol{X}$ from equation (3.2) for each parking lot, to determine an optimized charging schedule for each EV in each parking lot, and will also be solving for the solution vector $\boldsymbol{limTf}$ (equation (3.20), for the set of all transformer limits across all parking lots.

The multiple parking lot optimization model finds $n_{pl}numX$ number of solution variables for all solution vectors $\boldsymbol{X}$ (from equation (3.3)), and n_{pl} number of solution variables for the solution vector $\boldsymbol{limTf}$. In a case with three parking lots, ten timeslots and 20 EVs per parking lot, a maximum of total 603 solution variables must be solved.

3.3.1 Two-level PSO

To optimize two objectives in a nearly simultaneous way, a two-level PSO algorithm was developed for this model. This model uses the PSO-based single parking lot optimization model as the "inner PSO" level of the algorithm, to optimize individual parking lots, with an "outer PSO" level developed to optimize the transformer limits for all parking lots in the model. A basic flowchart for this two-level PSO is shown in Figure 3.2.



Outer PSO
1. Initialize positions $\mathbf{x}_t$ and velocities $\mathbf{v}_t$
2. Evaluate fitness function $f(\mathbf{x}_t)$: complete inner PSO
Inner PSO
1. Initialize positions $\mathbf{x'}_{t'}$ and velocities $\mathbf{v'}_{t'}$
2. Evaluate fitness function $f'(\mathbf{x'}_{t'})$
3. Update $\mathbf{b'}_{t'}$
4. Update $\mathbf{g'}_{t'}$
5. Compute velocity $\mathbf{v'}_{t'+1}$
6. Compute position $\mathbf{x'}_{t'+1}$
7. Terminate?
no
yes
8. Result
3. Update $\mathbf{b}_t$
4. Update $\mathbf{g}_t$
5. Compute velocity $\mathbf{v}_{t+1}$
6. Compute position $\mathbf{x}_{t+1}$
7. Terminate?
no
yes
8. Result

Figure 3.2 Two-level PSO Algorithm Flowchart

In Figure 3.2, the inner PSO is contained in the grey box within step 2 of the outer PSO. The completion of the inner PSO is contained within the fitness function evaluation of each candidate solution of the outer PSO: for each candidate solution or particle of the outer PSO, a complete inner PSO is completed as part of the evaluation of its fitness.

As opposed to the bi-level PSO model in [111] which has the upper level PSO's optimal solution computed before the lower level PSO begins, or the two-stage model in [143] which has the first-stage PSO's optimal solution computed before beginning the second stage GA optimization, this algorithm: begins the outer PSO, completes $n_{itr(IL)}$ total iterations of the inner ("inner level (IL)") PSO, uses the results of the inner PSO in the outer PSO's fitness function calculations, and then continues on to complete its $n_{itr(OL)}$

total iterations of the outer (“outer level (OL)”) PSO. The outer PSO has $n_{p(OL)}$ total particles. In every single iteration of the outer PSO, $n_{p(OL)}$ full inner PSOs with $n_{itr(IL)}$ iterations and $n_{p(IL)}$ particles each are completed.

For the inner PSO, as stated in Section 3.2, the solution vector $\boldsymbol{X}$ is equivalent to the PSO particle position vector term $\boldsymbol{x}$ from equations (2.8) and (2.9) ($\boldsymbol{x'}$ in Figure 3.2). For the outer PSO, an “undecoded” version of the solution vector $\boldsymbol{limTf}$ is equivalent to the PSO particle position vector term $\boldsymbol{x}$ from equations (2.8) and (2.9) (see Appendix D).

3.3.2 Objective and Fitness Function

For this model, each PSO level has a different objective function and fitness function. However, both PSO levels have a similar overall objective: the minimization of costs. Per equation (3.4), the objective of the inner PSO is to minimize the cost of electricity C required to meet the charging demands (desired consumption) of all EVs in the parking lot, and results in an EV charging schedule for all EVs in one parking lot. For the outer PSO, the objective is to minimize the total cost of electricity required to meet the charging demands of all EVs in all parking lots, by adjusting the transformer limits of all parking lots. This adjustment allows for the transformer limit to be set higher or lower depending on the demands of the parking lot’s EVs, which allows for the inner PSO to minimize the cost using the new transformer limit parameter.

The objective function of the outer PSO is:

$$\text{minimize } C_{OL} = \sum_{p=1}^{n_{pl}} C_p \tag{3.21}$$

where the total cost of electricity for all parking lots C_{OL} is the sum of the cost of electricity for each parking lot p, C_p. The C_p term is equivalent to the cost term C from equation (3.4). This objective function therefore relies on the result of the inner PSO algorithm of Section 3.2 in order to perform its own optimization.

For this objective function, there were fewer additional constraints, since the majority of the constraints were addressed by the objective and fitness function of the inner PSO as described in Section 3.2.1. The additional constraint for this outer PSO, based on equation (3.19), is:

$$\sum_{p=1}^{n_{pl}} limTf_p = P_{total} \tag{3.22}$$

where the total of the parking lot transformer limits for all parking lots p ($limTf_p$) must equal the total power capacity P_{total}.

As with the fitness function of the single parking lot optimization model, the basis for the inner PSO, the fitness function of the outer PSO for this model was composed of a penalty term and a cost term.

3.3.2.1 *Penalty Term*

For this outer PSO of the multiple parking lot optimization model, the penalty term ρ_{OL} is used to penalize the infeasible candidate solutions. The penalty term of this outer PSO fitness function relies on the results and penalty terms of the inner PSO fitness function from equation (3.18). The penalty term equations are:

$$\rho_{OL} = d_\rho + d_v \tag{3.23}$$

$$dd_\rho{}^p = \begin{cases} \rho_p, & F(X) < 1 \\ 0, & F(X) \geq 1 \end{cases} \quad \forall p \in P_l \tag{3.24}$$

$$d_\rho = \sum_{p=1}^{n_{pl}} d_\rho{}^p \tag{3.25}$$

$$dd_{v_b} = \begin{cases} 0.9 - |V_b|, & |V_b| < 0.9 \\ |V_b| - 1.1, & |V_b| > 1.1 \\ 0, & otherwise \end{cases} \quad \forall b \in B \tag{3.26}$$

$$d_v = \sum_{b=1}^{numB} dd_{v_b} \tag{3.27}$$

where ρ_{OL} is the outer PSO penalty term, d_ρ is the total multi-parking lot penalty term, d_v is the voltage magnitude limit violation term, $dd_\rho{}^p$ is the penalty term for parking lot p, ρ_p is the single parking lot penalty term for parking lot p, and dd_{v_b} is the term for when one bus's voltage magnitude exceeds or subceeds the voltage magnitude limit.

The penalty term in equation (3.23) is calculated by summing the penalty terms for all n_{pl} parking lots (from the inner PSO, equation (3.10)) with the penalty or constraint violation term for the power flow voltage magnitudes.

The outer PSO has $n_{p(OL)}$ particles, which each contain the n_{pl} candidate solutions for the transformer limits of n_{pl} parking lots. If the inner PSO's final results were feasible, i.e. $F(\boldsymbol{X}) \geq 1$ for the parking lot p term of the candidate solution vector $limTf_p$, then the penalty term for the outer PSO is 0 (for that parking lot $dd_\rho{}^p = 0$). A feasible result means that the single parking lot optimization method of Section 3.2 was able to find a feasible EV charging schedule solution for parking lot p when using the candidate transformer limit value for that parking lot $limTf_p$ as its transformer limit $limTf_1$ in equations (3.8) and (3.14).

If the results were not feasible, i.e. a feasible EV charging schedule solution could not be found with transformer limit $limTf_p$ for that parking lot and $F(\boldsymbol{X}) < 1$, then the penalty term for that parking lot $dd_\rho{}^p$ is equal to the single parking lot penalty term ρ from equation (3.10), denoted as ρ_p. The total multi-parking lot penalty term d_ρ in equation (3.25) is therefore found by taking the penalty terms for all parking lots of that solution (equation (3.24)) and summing them together. If all n_{pl} results for the solution vector $\boldsymbol{limTf}$ were feasible, then the penalty term $d_\rho = 0$.

The voltage magnitude limit violation term d_v in equation (3.27) finds all instances where the voltage magnitude at a bus was above or below the voltage magnitude limits in equation (3.9), finds the difference between the actual values and the violated limit (equation (3.26)), and sums all constraint violations across all $numB$ buses together. The voltage magnitudes come from the conduction of a power flow simulation. The power flow simulation is conducted by selecting the buses for each parking lot to be located at, adding the maximum possible additional active power demand of the parking lots ($limTf_p$) to those buses, and then running the power flow simulation to find the resulting bus voltage magnitudes.

The constraint listed in equation (3.22) was not checked as part of the PSO's fitness function evaluation process, but was applied during the solution decoding process.

A value of more than 0 for the multi-parking lot penalty term d_ρ meant that one or more of the individual parking lot EV charge scheduling solutions was infeasible. A value of more than 0 for the voltage magnitude limit violation term d_v meant that one or more of the distribution system bus voltages magnitudes $|V_b|$ was outside the limits, therefore violating the distribution system network constraint. In order for a solution to be considered feasible, penalty term ρ_{OL} had to be equal to 0, i.e. with a feasible EV charge scheduling solution for all parking lots and no distribution system limit violations. This is illustrated in the combined fitness function.

As was the case with the single parking lot model penalty terms described in Section 3.2.1.1, these penalty terms are also not normalized to the same units, so there is no prevention of bias or attempt to give the terms equal weight [158]. Min-max normalization feature scaling was identified as a potential method for normalization of all penalty values in a range $[\min(d), \max(d)]$ to the range $[\alpha, \beta]$, but was found to be impractical as all penalty terms have no identifiable real number possible maximum value $\max(d)$. Min-max normalization feature scaling is given by the following equation:

$$d_{normalized} = (\beta - \alpha)\frac{d - \min(d)}{\max(d) - \min(d)} + \alpha \tag{3.28}$$

where $d_{normalized}$ is the normalized penalty value, d is the original penalty value, and α and β are the minimum and maximum of the normalisation range.

Ultimately, the biases inherent in the penalty term ρ_{OL} allow solutions with larger single parking lot penalty terms (ρ_p) to be more heavily penalized than solutions with large voltage magnitude violations. This characteristic supports the model well, as finding transformer limits which provide feasible solutions for their parking lots is the key focus of the model, and penalization of solutions which violate the voltage magnitude limits becomes important after these feasible solutions are found and the penalty term d_ρ is minimized.

3.3.2.2 *Cost Term*

The cost term C_{OL} of the fitness function represents the optimization objective of the objective function equation (3.21), the cost of electricity purchased from the utility for all parking lots. It is repeated in equation (3.29) below for clarity:

$$C_{OL} = \sum_{p=1}^{n_{pl}} C_p \tag{3.29}$$

3.3.2.3 *Combined Fitness Function*

Following the calculation of the penalty and cost terms for the candidate solution $\boldsymbol{limTf}$, the fitness function $F(\boldsymbol{limTf})$ of that candidate solution can be calculated per equation (3.30):

$$F(\boldsymbol{limTf}) = \begin{cases} 0 + \dfrac{1}{1+\rho_{OL}}, & \rho_{OL} > 0 \\ 1 + \dfrac{1}{1+C_{OL}}, & \rho_{OL} = 0 \end{cases} \tag{3.30}$$

where ρ_{OL} is the penalty term from equation (3.23), C_{OL} is the cost term from equations (3.21) and (3.29). As with combined fitness function in equation (3.18), this equation has been defined so that all infeasible candidate solutions (where $\rho_{OL} > 0$) always have a fitness in the range $(0, 1)$, and all feasible candidate solutions will have a fitness above 1, in the range of $(1, 2)$. This provides an easy way for the PSO to evaluate the feasibility of the solution as well as an easy visual representation of the behaviour of the PSO as it moves from infeasible to feasible solutions over its iterations. As the solution becomes more optimal and the cost is minimized, the fitness function value is maximized.

3.3.3 Algorithm Steps

As described in Section 3.3.1, a two-level PSO algorithm with an inner and an outer PSO was developed for this method. This model used the standard PSO algorithm as the basis for both the inner and outer PSOs. The outer PSO algorithm computed feasible parking lot transformer limits for multiple parking lots. The inner PSO algorithm computed EV charging schedules for each parking lot, with each transformer limit computed by the outer PSO. The algorithm steps and flowchart for the parallelized version of this are shown in Figure 3.4 and Algorithm 2.

For the non-parallelized model described in Sections 3.3 to 3.3.2.3, the flowchart of the model follows the general two-level PSO flowchart in Figure 3.2. Firstly, after receiving all the requisite inputs to the algorithm like the number of parking lots, base transformer limit, and EV charging profiles for each parking lot, the aggregator initializes the outer PSO position and velocity vectors, with $n_{p(OL)}$ number of particles with n_{pl} variables (transformer limit candidate solution vector $\boldsymbol{limTf}$) each. Then, the fitness function in equation (3.30) is evaluated for each candidate solution.

For each candidate solution or particle, the fitness function evaluation requires the evaluation of a complete inner PSO for each parking lot. In every iteration of step 2 of the outer PSO, $n_{p(OL)}n_{pl}$ inner PSOs are required to be evaluated. Each complete inner PSO follows the process described in Section 3.2.2, evaluating $n_{itr(IL)}$ PSO iterations with $n_{p(IL)}$ number of particles with $numX$ variables each. Once all inner PSOs have been completed, the next part of the fitness function evaluation requires the conduction of the power flow simulation. Once this has been completed, the fitness function evaluation is completed for the outer PSO.

Next, the best personal and global positions are updated, new particle position and velocities are updated, and so on, until the process is terminated once the maximum number of PSO iterations $n_{itr(OL)}$ has been reached. The best global position, in this case the most optimal transformer limits for the n_{pl} parking lots (from a candidate solution vector $\boldsymbol{limTf}$) with the associated optimal charging schedule for each parking lot (from a candidate solution vector $\boldsymbol{X}$ per each parking lot), is taken as the most optimal final result for this model. Once the final result is selected, the algorithm ends.

3.3.4 Parallelization

With the development of this model to optimize n_{pl} parking lot charging schedules, requiring the evaluation of $n_{p(OL)}n_{pl}$ inner PSOs within the outer PSO, the problem has become more complex, and features an increased computational burden. As with the single parking lot optimization model, this model was parallelized in order to reduce the computational runtime and benefit from the use of HPC.

In this two-level PSO algorithm, the fitness function evaluation of each outer PSO particle can be performed in parallel (as can the fitness function evaluations of each inner PSO particle per Section 3.2.4), because the evaluation of each candidate solution does not depend on the evaluation or outcome of any other candidate solution. As with the single parking lot optimization algorithm, the two-level algorithm can be parallelized in that portion of the algorithm: step 2 of the PSO, per Figure 3.2.

Following the general steps of Algorithm 1, step 2 of the outer PSO would require the evaluation of a `for` loop for each candidate solution (iterations $j = 1, 2, \ldots, n_{p(OL)}$), which would have within each loop iteration the evaluation of a `for` loop for each inner PSO candidate solution (iterations $k = 1, 2, \ldots, n_{p(IL)}$) (the sequential version of Algorithm 1 lines 6-8). However, the parallelization of the outer PSO process prevents the nested parallelization of any portion of the inner PSO, since in MATLAB, `parfor` loops cannot be nested inside one another [159]. While some other programming languages like OpenMP have the possibility to support nested parallelism (though nested parallelism is currently disabled by default and the related routines are now deprecated [160], [161]), the use of all threads (worker processes) at the most outer-level of the computation algorithm and/or the collapsing of multiple inner and outer `for` loops into one outer loop is generally seen as the most efficient and correct way to perform the parallelization of nested loops [160], [162]. This method uses the threads at the most outer-level of the algorithm.

In order to efficiently parallelize the two-level PSO algorithm on a HPC with a large number of workers, while maintaining the effective operation of the inner PSO single parking lot optimization algorithm, the fitness function evaluation of the outer PSO is parallelized, using a row-major layout [121] to linearize the array of candidate solutions. Per the first image within Figure 3.3 below, each outer PSO candidate solution (in this example, $n_{p(OL)} = 5$) has $n_{pl} = 4$ parking lots which need to be optimized using the inner PSO algorithm. These solutions are represented by a conventional 20-element $n_{p(OL)}n_{pl}$ two-dimensional array, where each HPC worker evaluates the fitness of one candidate solution (one row): one worker performs n_{pl} sequential inner PSO optimizations. In Figure 3.3, each candidate solution is denoted by a different colour (red, blue, yellow, green, and grey).

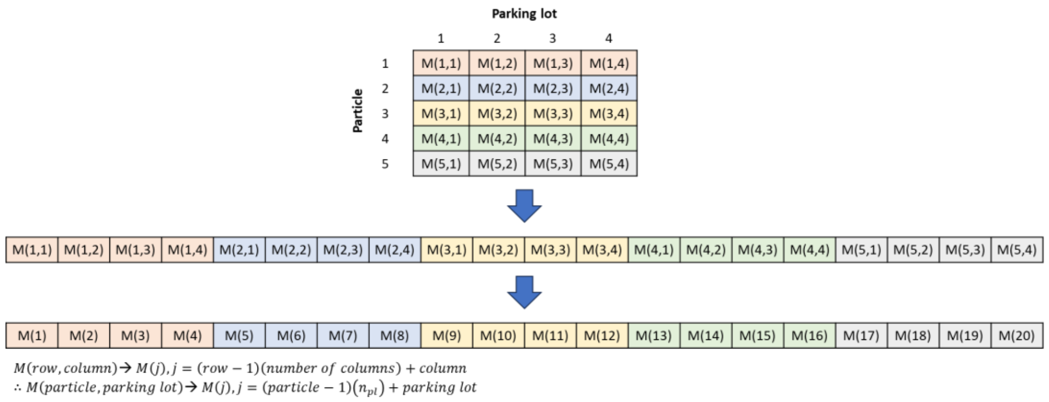

Parking lot
1
2
3
4
Particle
1
2
3
4
5
M(1,1) M(1,2) M(1,3) M(1,4)
M(2,1) M(2,2) M(2,3) M(2,4)
M(3,1) M(3,2) M(3,3) M(3,4)
M(4,1) M(4,2) M(4,3) M(4,4)
M(5,1) M(5,2) M(5,3) M(5,4)
M(1) M(2) M(3) M(4) M(5) M(6) M(7) M(8) M(9) M(10) M(11) M(12) M(13) M(14) M(15) M(16) M(17) M(18) M(19) M(20)
M(row, column)→ M(j), j = (row − 1)(number of columns) + column
∴ M(particle, parking lot)→ M(j), j = (particle − 1)(n_pl) + parking lot

Figure 3.3 Row-major linearization of a two-dimensional array of outer PSO candidate solutions (particles) and parking lots (adapted from [121])

If the number of HPC workers is greater than the number of candidate solutions, then a number of workers within the cluster may go unused. However, transforming the array to a one-dimensional row-major layout with $n_{p(OL)}n_{pl} = 20$ elements (the second and third images in Figure 3.3) enables more efficient use of the HPC cluster, with assigning one worker per column, and every column containing just one parking lot to be optimized. The reduction of individual sequential inner PSO optimizations performed by each HPC worker from n_{pl} to one reduces the potential control divergence and differences in computational runtimes between workers, and uses the HPC workers more effectively when the number of workers is more than the number of candidate solutions $n_{p(OL)}$.

Using this linearized array, step 2 of the outer PSO therefore consists of the evaluation of a `for` loop for element of the linearized array (elements $j = 1, 2, \ldots, n_{p(OL)}n_{pl}$). This algorithm process flowchart is shown in Figure 3.4 below.

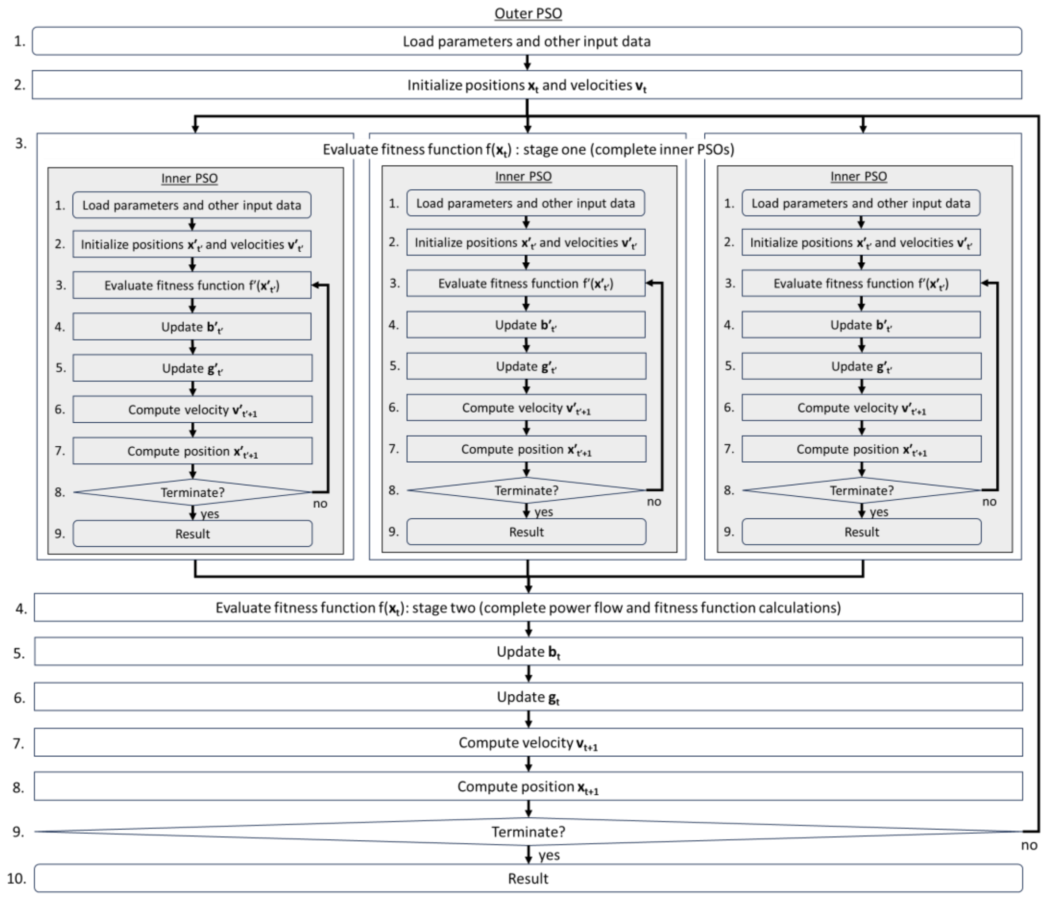

Outer PSO
1. Load parameters and other input data
2. Initialize positions $\mathbf{x}_t$ and velocities $\mathbf{v}_t$
3. Evaluate fitness function f($\mathbf{x}_t$) : stage one (complete inner PSOs)
Inner PSO
1. Load parameters and other input data
2. Initialize positions $\mathbf{x}'_{t'}$ and velocities $\mathbf{v}'_{t'}$
3. Evaluate fitness function f'($\mathbf{x}'_{t'}$)
4. Update $\mathbf{b}'_{t'}$
5. Update $\mathbf{g}'_{t'}$
6. Compute velocity $\mathbf{v}'_{t'+1}$
7. Compute position $\mathbf{x}'_{t'+1}$
8. Terminate?
no
yes
9. Result
4. Evaluate fitness function f($\mathbf{x}_t$): stage two (complete power flow and fitness function calculations)
5. Update $\mathbf{b}_t$
6. Update $\mathbf{g}_t$
7. Compute velocity $\mathbf{v}_{t+1}$
8. Compute position $\mathbf{x}_{t+1}$
9. Terminate?
no
yes
10. Result

Figure 3.4 Multiple Parking Lot Optimization Algorithm Flowchart

Figure 3.4 shows that steps 1 to 2 and 4 to 10 of the outer PSO are performed sequentially, by the same single process, and that step 3 of the process, the completion of the PSOs as part of the evaluation of the fitness function, is performed in parallel for each parking lot in each particle of the outer PSO.

In step 3, all processes which perform the parallel fitness function evaluations receive the same parameters, except for the candidate solution transformer limits per each parking lot. The single parking lot optimization PSO algorithm for each parking lot in each particle is completed independently, simultaneously with the other particle parking lots, by the parallel processes. Following this, the PSO results (inner PSO fitness value, cost, unmet demand, penalties, and the optimized schedule for the parking lot) are passed back to the single main process, which then performs the power flow simulation and computes the final penalty and fitness results per equation (3.30) for all parking lots. The key inputs to the inner PSOs in step 3 are the candidate transformer limit values (equation (3.20)), and their key outputs to step 4 are the fitness function values per parking lot (equations (3.18), (3.24)), cost per parking lot (equations (3.4), (3.21), (3.29)), penalties per parking lot (equations (3.10), (3.24), (3.25)), and resulting schedule per parking lot (from the best candidate solution vector, equation (3.2)).

The single worker process then performs the updates for the best personal and global positions, and updates the velocity and positions for the next iteration. If the maximum number of PSO iterations has been completed, the algorithm terminates and presents the final result (the most optimal transformer limits for the n_{pl} parking lots with the associated optimal charging schedule for each parking lot); if not, steps 3 to 9 are repeated, with step 3 always occurring in parallel.

The method used for this parallelization was the same SPMD method described in Section 3.2.4. The parallelization was performed through the use of a `parfor` loop around the function which performed the optimization of a single parking lot in a single particle, and therefore executed each iteration of the linearized candidate solution array (iterations $j = 1, 2, \dots, n_{p(OL)} n_{pl}$) in parallel on the workers in the HPC cluster. The pseudocode of the parallelized algorithm with the `parfor` function is shown in Algorithm 2. Algorithm 1 has been modified to become the sequential version of the inner PSO algorithm used for this method, and is shown in Algorithm 3.

Algorithm 2. Multiple Parking Lot Optimization Algorithm Pseudocode (Parallelized)

```
1:  generate EV parking lot data for $n_{pl}$ parking lots from parking lot profiles
2:  generate problem and PSO parameter data
3:  generate population of $n_{p(OL)}n_{pl}$ initial candidate solutions (linearized array)
4:  initialize PSO position and velocity vectors
5:  while termination criteria is false (iteration < $n_{itr(OL)}$)
6:      parfor each element of the linearized candidate solution array ($j = 1, 2, \ldots, n_{p(OL)}n_{pl}$)
7:          complete Algorithm 3 (fitness function evaluation stage one)
8:      end parfor
9:      for each candidate solution ($j = 1, 2, \ldots, n_p$)
10:         run power flow and check voltage feasibility constraints of each solution
11:         evaluate the fitness of the candidate solution (fitness function evaluation stage two)
12:     end for
13:     for each candidate solution ($j = 1, 2, \ldots, n_p$)
14:         update the local best position and fitness
15:         update the global best position and fitness
16:     end for
17:     for each candidate solution ($j = 1, 2, \ldots, n_p$)
18:         compute the new particle velocity and enforce velocity limitations
19:         compute the new particle position and enforce velocity limitations
20:     end for
21: return best solution
```

Algorithm 3. Single Parking Lot Optimization Algorithm Pseudocode (Sequential Inner PSO Version)

```
1:  generate and intake PSO parameter data
2:  generate population of $n_{p(IL)}$ initial candidate solutions (particles)
3:  initialize PSO position and velocity vectors
4:  while termination criteria is false (iteration < $n_{itr(IL)}$)
5:      for each candidate solution ($j = 1, 2, \ldots, n_{p(IL)}$)
6:          evaluate the fitness of the candidate solution
7:      end for
8:      for each candidate solution ($j = 1, 2, \ldots, n_{p(IL)}$)
9:          update the local best position and fitness
10:         update the global best position and fitness
11:     end for
12:     for each candidate solution ($j = 1, 2, \ldots, n_{p(IL)}$)
13:         compute the new particle velocity and enforce velocity limitations
14:         compute the new particle position and enforce velocity limitations
15:     end for
16: return best solution
```

The sequential and parallelized portions of the multiple parking lot optimization model are clearly shown with Algorithm 2 and Figure 3.4: only step 3 of the outer PSO, or lines 6 to 8 in the algorithm, are parallelized, with all other parts of the model or algorithm remaining sequential. As was the case with Algorithm 1, the `for` loops in lines 9-12, 13-16, and 17-20 of Algorithm 2 are not parallelized, in order to reduce the communication overhead and resulting increase in runtime, because the calculations performed in these sections are quick to be performed by a single worker in MATLAB. Since MATLAB `parfor` provides generally even distribution of the workload across the n workers in the parallel pool, it is expected that each worker process performs approximately $(n_{p(OL)} n_{pl})/n$ `parfor` iterations or inner PSOs.

The inner PSO, Algorithm 3, is the most computationally complex part of the algorithm, being very time consuming to execute. Per Amdahl's Law, by limiting the sequential part of the algorithm, the speedup of an algorithm can be improved. Therefore, by parallelizing the execution of a large number of the computationally complex inner PSOs, a large gain in performance (decrease in algorithm runtime, and increase in speedup) is expected.

3.4 Decentralized Multi-Parking Lot Optimization

The method developed for the decentralized optimization of the EV charge scheduling within multiple parking lots is based on the centralized algorithm described in Section 3.3: it is a decentralized version of that algorithm.

The concept for the decentralized version is as follows: instead of the computational load being processed by only one aggregator, it is shared across each parking lot. Each parking lot has their own local aggregator, which acts as an independent optimizer for the EV charging schedules within its location. Parameter or other optimization information is not shared between the local parking lot aggregators, and only limited optimization results are provided back to the central aggregator. This method attempts to provide the benefits of the decentralized approach as discussed in Section 2.3.2: to reduce communication overhead, divide the computational load, and provide a much faster runtime.

The information known to the central aggregator which flows "downwards" to the local aggregators consists of the EV parking lot profiles, the charging rate limit, transformer limit per each parking lot, the price of purchasing electricity during all time intervals, and the set amount of power P_{total} that is available to the central aggregator to divide among the parking lots. The only information provided back to the central aggregator from each local parking lot, which is required for the functioning of the algorithm, is the feasibility of that lot's optimization results, the cost, and the resulting EV charging schedule. This limited information is only required by the central aggregator to determine the total cost among parking lots, and to determine if the optimization solution for the total n_{pl} parking lots is feasible or not, with regards to both individual optimization results and power flow within the distribution system.

The main difference in this method from the centralized method is that the transformer limits are not optimized: the central aggregator does not perform any optimization itself. Instead, each parking lot is assigned the same transformer limit by the central aggregator: $limTf_{base}$. The power is divided equally among the parking lots, and each lot performs their own independent single parking lot optimization algorithm. The removal of the optimization of the transformer limits for each parking lot (and therefore the removal of the outer PSO) is expected to result in a less optimal result in terms of the total cost of electricity for all parking lots, but is also expected to result in a much more computationally efficient (faster) result.

Figure 3.5 shows a basic comparison between the two methods. In the centralized method, all computations for the two-objective, two-level optimization are performed by the central aggregator. In the decentralized method, all computations are divided amongst the local aggregators at each parking lot, where each aggregator performs its own independent single-level optimization.

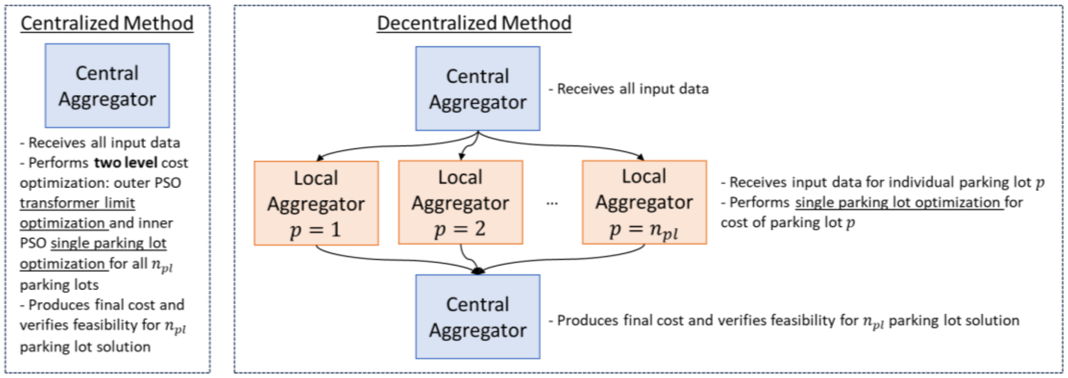

Centralized Method
Central Aggregator
- Receives all input data
- Performs **two level** cost optimization: outer PSO transformer limit optimization and inner PSO single parking lot optimization for all n_{pl} parking lots
- Produces final cost and verifies feasibility for n_{pl} parking lot solution
Decentralized Method
Central Aggregator
- Receives all input data
Local Aggregator $p = 1$
Local Aggregator $p = 2$
…
Local Aggregator $p = n_{pl}$
- Receives input data for individual parking lot p
- Performs single parking lot optimization for cost of parking lot p
Central Aggregator
- Produces final cost and verifies feasibility for n_{pl} parking lot solution

Figure 3.5 Centralized and Decentralized Method Comparison

3.4.1 Objective and Fitness Function

The overall objective of this method is the same as the others: to minimize cost of electricity. In this case, it is the electricity required to meet the charging demands (desired consumption) of all EVs in all parking lots, resulting in EV charging schedules for all EVs in all parking lots. However, since the central aggregator is not performing any optimization of the total cost, the optimization objective for this model is met by each local aggregator independently optimizing the cost of their single parking lot charging schedule.

Therefore, the objective function is the same as that for the single parking lot optimization model: to minimize the cost of electricity purchased from the utility to meet the charging demands of all EVs in a parking lot, considering the variation in the electricity price in each timeslot, using equation (3.4). The optimization function is subject to the same constraints from equations (3.5) to (3.8) (where $limTf_1 = limTf_{base}$), and the fitness function which is being evaluated during the optimization process by each local aggregator is the same function from equation (3.18).

This model is also subject to the voltage magnitude constraint from equation (3.9), though it is not included in the fitness function like it is in the centralized model. Instead, it is used by the central aggregator to check the feasibility of the overall solution, as it was for the single parking lot optimization model. The central aggregator receives the fitness values $F(\boldsymbol{X})$ from each local aggregator, and runs the power flow simulation using the transformer limits $limTf_{base}$ for each parking lot as maximum possible additional active power demand at each selected bus. Since a penalty value is not required from the power flow simulation as it was from the centralized model (equations (3.26), (3.27)), if the constraint was found to be violated at any bus, the solution was determined to be infeasible. As well, if any parking lot's returned fitness value indicated an infeasible solution for that parking lot, $F(\boldsymbol{X}) < 1$, then the total solution was also determined to be infeasible.

To have a feasible final solution for this model, the power flow simulation had to result in no violations of equation (3.9), and all local aggregator solutions for all n_{pl} parking lots had to result in feasible solutions, $F(\boldsymbol{X}) \geq 1$ for each parking lot.

Equation (3.31) is used by the central aggregator to produce the final cost:

$$C_{total} = \sum_{p=1}^{n_{pl}} C_p \tag{3.31}$$

where C_{total} is the final or total cost of the solution, found by summing the cost per each optimized parking lot C_p.

3.4.2 Algorithm Steps

This model used the PSO-based single parking lot optimization algorithm developed in Section 3.2 in combination with summation and feasibility actions to compute feasible EV charging schedules for multiple parking lots. The algorithm steps and flowchart for the parallelized version of this are shown in Figure 3.6 and Algorithm 4.

For the non-parallelized model described in Sections 3.4 and 3.4.1, the flowchart of the model follows and extended version of the standard PSO flowchart from Figure 2.1, or the non-parallelized version of Figure 3.1. Firstly, the central aggregator receives all the inputs to the algorithm like the number of parking lots, base transformer limit, and EV charging profiles for each parking lot, and passes the relevant information to each local aggregator.

Next, each local aggregator performs the single parking lot optimization process for their lot. Each aggregator initializes the PSO position and velocity vectors, with n_p number of particles with $numX$ variables each. Then, the fitness function in equation

(3.18) is evaluated for each candidate solution. The best personal and global positions are updated, new particle position and velocities are updated, and so on, until the process is terminated once the maximum number of PSO iterations n_{itr} has been reached. The best global position, in this case the most optimal charging schedule from a candidate solution vector X, is taken as the most optimal final result for that aggregator's optimization process.

Next, each local aggregator provides their fitness and cost result to the central aggregator. The central aggregator checks the feasibility of the entire solution by checking the individual fitness results and by conducting a power flow simulation. Once the feasibility has been checked and the total cost calculated, the algorithm is complete. The optimal charging schedule for each parking lot is taken as the final result for this model.

3.4.3 Parallelization

This model was developed to optimize n_{pl} parking lot charging schedules, requiring the evaluation of n_{pl} independent PSOs by the local aggregators. As with the other optimization models, this model was parallelized in order to reduce the computational runtime. However, in this case the parallelization of the algorithm is also necessary to demonstrate how the model would perform on a system with local aggregators which are physically or otherwise separate from a central aggregator, and would be expected to perform their computations independently and in parallel with each other. As opposed to a single central aggregator which uses a computational cluster to perform its calculations for the centralized multiple parking lot optimization model, this model would function by having the central aggregator using a single workstation or compute node, and the local aggregators each represented by a single workstation or compute node, which can all communicate with the central aggregator (see Section 3.1.4).

In this algorithm, the full PSO optimization of each parking lot by the local aggregators can be performed in parallel since no part of their optimization process depends on the outcome or actions of another parking lot's optimization. Therefore, the algorithm can be parallelized in step 2 of the figure below, where each local aggregator completes a full single parking lot optimization. This algorithm process is shown in Figure 3.6 below.

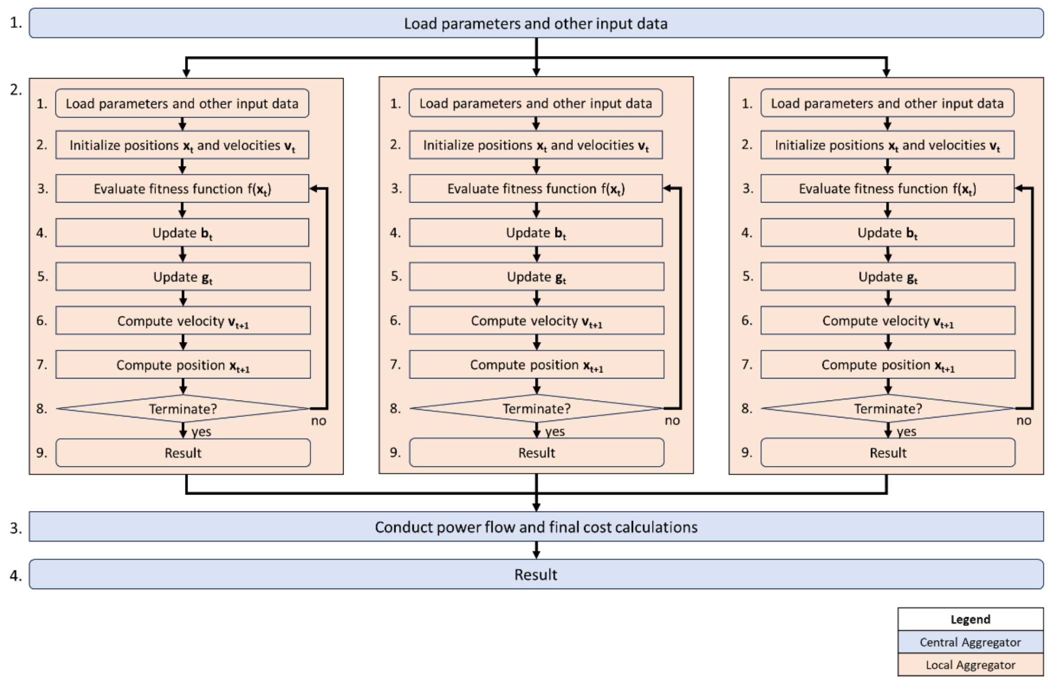

1.
Load parameters and other input data
2.
1. Load parameters and other input data
2. Initialize positions x_t and velocities v_t
3. Evaluate fitness function $f(x_t)$
4. Update b_t
5. Update g_t
6. Compute velocity v_{t+1}
7. Compute position x_{t+1}
8. Terminate?
no
yes
9. Result
3.
Conduct power flow and final cost calculations
4.
Result
Legend
Central Aggregator
Local Aggregator

Figure 3.6 Decentralized Multiple Parking Lot Optimization Algorithm Flowchart

Figure 3.6 shows that steps 1, 3, and 4 of the algorithm, shown in blue, are performed by the central aggregator sequentially, and that step 2, which encompasses all 9 steps of the single parking lot PSO optimization process, has each PSO optimization performed in parallel by the local aggregators.

In step 2, all local aggregators receive the same parameters, except for the parking lot profile to be optimized. The single parking lot optimization PSO algorithm for each parking lot is completed independently, simultaneously with the other particle parking lots, by the parallel aggregators. Following this, optimization results for fitness, cost, and the optimized schedule for the parking lot are passed back to the single main process, which then performs the power flow simulation, determines the feasibility of the solution, and computes the final costs per equation (3.31) for all parking lots.

The method used for this parallelization was the same SPMD method described in Sections 3.2.4 and 3.3.4. The parallelization was performed through the use of a `parfor` loop around the function which performed the optimization of a single parking lot, and therefore executed each iteration, or each parking lot's optimization (iterations $j = 1, 2, \dots, n_{pl}$), in parallel on the workers in the HPC cluster. The pseudocode of the parallelized algorithm with the `parfor` function is shown in Algorithm 4. For this algorithm, the single parking lot optimization performed by the local aggregators is the same as that shown in Algorithm 3.

Algorithm 4. Decentralized Multiple Parking Lot Optimization Algorithm Pseudocode (Parallelized)

```
1:  generate EV parking lot data for $n_{pl}$ parking lots from parking lot profiles
2:  generate problem data
3:  **parfor** each parking lot ($j = 1, 2, \dots, n_{pl}$)
4:      complete Algorithm 3
5:  **end parfor**
6:  run power flow and check voltage feasibility constraints of the solution
7:  check overall feasibility of the solution (each parking lot's best solution) and compute total cost
8:  **return** solution
```

The sequential and parallelized portions of the decentralized multiple parking lot optimization model are clearly shown with Figure 3.6 and Algorithm 4: only step 2 of the flowchart, or lines 3 to 5 in the algorithm, are parallelized, with all other parts of the model or algorithm remaining sequential.

It is expected that each worker process performs approximately $^{n_{pl}}/_{n}$ `parfor` iterations. In a case where there are fewer parking lots than processes, only $n = n_{pl}$ processes are required to execute the algorithm.

3.5 PSO Parameters

For all instances of the PSO being used (inner and outer PSOs of the centralized parallelized and sequential functions, the single PSO in the decentralized function), the parameters in Table 3.1 were used. The inertia and influence values came recommended from Table 2 of [132] and [21], and are close to the values recommended in [163].

Table 3.1 PSO Parameters

Parameter	Value
Inertia (ω)	0.7298
Personal Influence (c_1)	1.4960
Social Influence (c_2)	1.4960
Particle (x_i) / search space boundaries	$0 \leq x_i \leq 1$
Maximum velocity (v_{max})	0.25

The particle boundaries ensure that the particle remains within the search space of 0 to 1. The maximum velocity value ensures that each single particle cannot travel more than a quarter of the distance of the search space (from 0 to 1 for particle x_i as $x \in [0, 1]$) in each iteration, assisting to "reduce the convergence speed and encourage exploration" [21].

3.6 Distribution System Scenarios

Four distinct scenarios, or combinations of parking lots and distribution system test cases, were created for the tests in Section 4.

3.6.1 Parking Lot Profiles

Nine parking lot profiles were developed, based on the single parking lot, 20 EV charging request profile from Table 5 in [40]. Parking lot profiles 2-9 can be found in Table 6.1 to Table 6.8 in Appendix A. Parking Lot Profile 1, seen in Table 3.2 below, is identical to the 20 EV charging request profile from Table 5 in [40], in order to allow for direct comparison and validation (see Sections 4.3 and 4.7.1.1). All other parking lot profiles were made by combining EVs from Profile 1 in various ways to change their charging timeslots, the number of EVs with different demands, and the overall total demand (desired consumption). The base transformer limit for each parking lot was kept as 60 kW.

Table 3.2 Parking Lot Profile 1 (reproduced from [40])

EV Number	Arrival Timeslot	Departure Timeslot	Charging Demand (Desired Consumption) (kWh)
1	1	3	18
2	3	5	15
3	1	5	25
4	2	4	18
5	2	3	15
6	6	10	22
7	7	8	14
8	8	10	16
9	7	8	10
10	6	9	20
11	3	8	26
12	2	6	17
13	5	8	15
14	4	8	16
15	3	7	14
16	5	8	12
17	4	7	16
18	5	9	19
19	1	10	28
20	1	5	16

3.6.2 MATPOWER Distribution Systems

Four distribution system test cases included in the MATPOWER package were selected as the test case scenarios to be used in combination with the different parking lot scenarios (1, 3, 9, 18, and 27 parking lots): 18-bus [164], 33-bus [165], 69-bus [166], and 141-bus [167] distribution systems. The locations of each parking lot within each distribution system model were selected after some experimentation. The symbol for the EV parking lot in this thesis is shown in Figure 3.7, and represents a parking lot with 20 EV charging ports, which can contain and/or charge a maximum of 20 EVs at one time.

For the scenario of one parking lot in an 18-bus system, the parking lot was connected to bus 2. For the scenario of three parking lots in an 18-bus system, the parking lots were connected to buses 2, 22, and 26. For the scenario of nine parking lots in the same system, the parking lots were connected to buses 2, 4, 6, 7, 8, 22, 23, 24, and 26, as shown in Figure 3.8.

Figure 3.7 EV Parking Lot Symbol

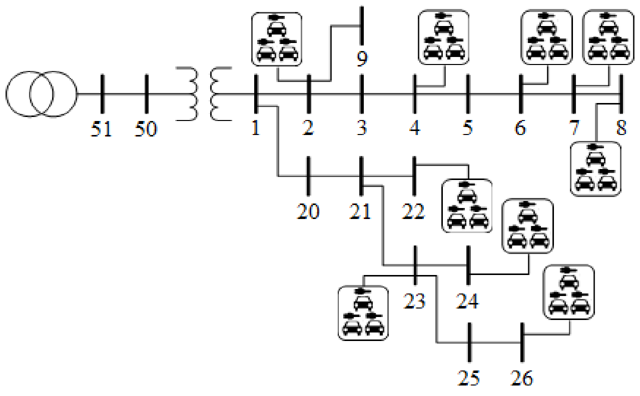


Figure 3.8 18-bus system with 9 Parking Lots (adapted from [153], [164])

For the scenario of nine parking lots in a 33-bus system, the parking lots were connected to buses 7, 8, 14, 24, 25, 29, 30, 31, and 32, as shown in Figure 3.9.

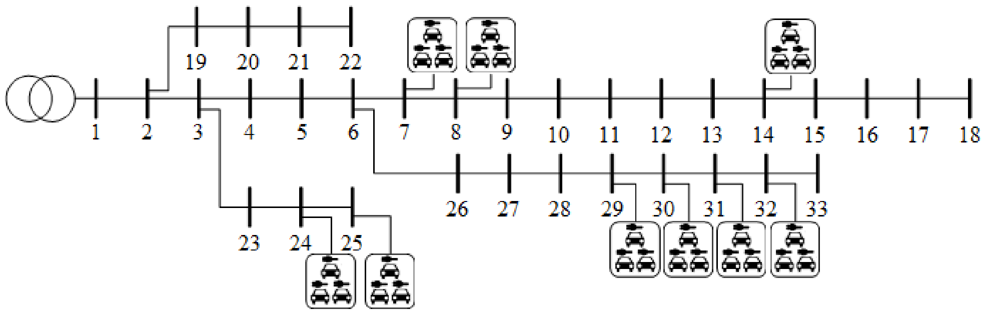


Figure 3.9 33-bus system with 9 Parking Lots (adapted from [153], [165])

For the test scenario with 18 parking lots in the 69-bus system, the parking lots were connected to buses 2, 3, 4, 5, 15, 19, 23, 25, 30, 31, 32, 38, 42, 44, 47, 56, 57, and 58 of the system, as shown in Figure 3.10.

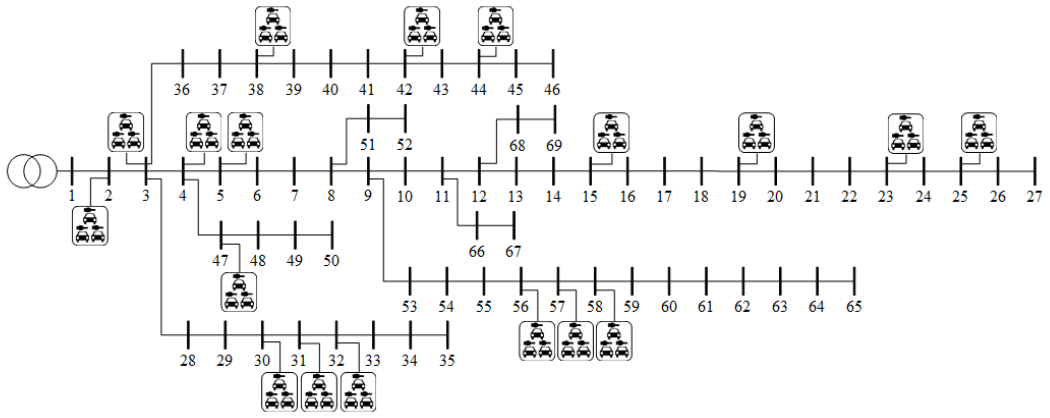


Figure 3.10 69-bus system with 18 Parking Lots (adapted from [153], [166])

For the test scenario with 18 parking lots in the 141-bus system, the parking lots were connected to buses 2, 4, 7, 10, 22, 38, 46, 63, 70, 78, 81, 95, 99, 102, 114, 118, 120, and 131 of the 141-bus system. For the test scenario with 27 parking lots in the 141-bus system, the parking lots were connected to buses 2, 4, 6, 7, 10, 16, 22, 28, 31, 38, 46, 54, 63, 70, 78, 81, 85, 91, 95, 99, 102, 108, 114, 118, 120, 125, and 131, as shown in Figure 3.11.

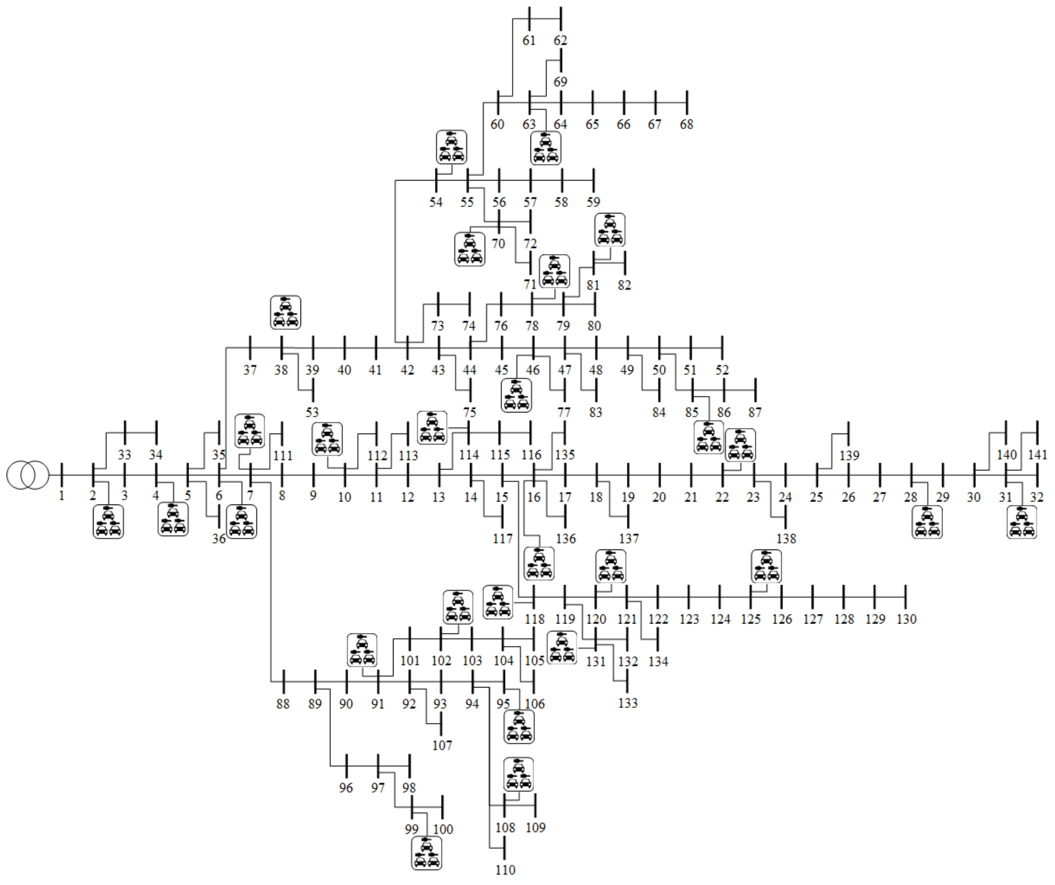


Figure 3.11 141-bus system with 27 Parking Lots (adapted from [153], [167])

Since each parking lot contains 20 EVs, the scenarios with 1, 3, 9, 18, and 27 parking lots contain a total of 20, 60, 180, 360, and 540 EVs.

3.7 Summary

This chapter presented the design of the models used in this research, detailing their objective and fitness functions, algorithm workflow, and parallelization. The assumptions and limitations described in Section 3.1 determined the scope and behaviour of the three models. The single parking lot optimization method described in Section 3.2 was adapted from [40] and parallelized as its own model, but also served as the basis for the sub-algorithm (Algorithm 3) used by the two multiple-parking lot models. Section 3.3 described the centralized multiple-parking lot optimization model, including its use of a two-level PSO and the linearization of its candidate solution array to be more effectively parallelized. The development of the decentralized version of the multiple-parking lot optimization model is described in Section 3.4. Section 3.5 outlined the PSO parameters used for all models, and Section 3.6 described the parking lot profiles and distribution system test cases developed for this research.

Section 4

Results

This section describes the results obtained from the different experimental tests run in order to validate the performance of the developed methods or algorithms, for different scenarios. The summarized results for each test are presented in their relevant sections, which include results for the algorithm outputs and performance in terms of cost, fitness, and runtime. The results for each test are discussed in order to address the aim, or research hypothesis, of this thesis.

The experimental setup is discussed in Section 4.1, and the success criteria in Section 4.2. The four tests and their results are shown and discussed in Sections 4.3, 4.4, 4.5, and 4.6. The test results in terms of meeting the success criteria are discussed in Section 4.7, and the overall section is summarized in Section 4.8.

The aim of this thesis is to develop a solution for EV charge scheduling optimization for multiple parking lots, using centralized and metaheuristic optimization methods to provide more optimal solutions, and using parallel computing to increase the scalability of the solutions and to complete the optimization in an acceptable real-time interval. The results show that the centralized method was able to provide more optimal solutions compared to the decentralised method for scenarios with more than one parking lot, and that parallel computing enabled the optimization of scenarios with up to 27 parking lots to be computed in less than 15 minutes, depending on their PSO parameters.

It must be noted that any use of the words “optimal solution,” “optimized solution,” or other similar wording does not mean that the solution is the best possible global or final solution to the problem [40], but means that it is the best solution found by the algorithm out of all its completed tests.

4.1 Experimental Setup

The tests to optimize a single parking lot used a single, standalone workstation with an Intel® Xeon® Silver 4210 CPU with 20 cores, using the Microsoft Windows operating system. All other tests were conducted on the RMC Taurus HPC Cluster [147], configured using MATLAB Parallel Server to use all 192 cores of the eight Linux compute nodes. Each node contains dual Intel Xeon Silver 4214R CPUs with 12 hyper-threaded cores, for a total of 24 cores or workers each. The eight compute nodes are connected through a 10 Gbps gigabit network [21]. The setup of the parallel job schedule is the same as described in [21]: the first node in the cluster, `leo1.local.net`, runs the scheduler, receives the parallel jobs from the client workstation (also `leo1.local.net`), and distributes that parallel workload among the 192 compute nodes. Code which is not parallelized is run on the client workstation CPU [21], [147].

All tests were conducted using MATLAB version 2023a, the MATLAB Parallel Computing Toolbox, and the MATLAB Parallel Server [149], [168], [169]. All tests used the standard Newton-Raphson method-based AC power flow solver from the MATPOWER software tool [153], [154] using different distribution system test cases included in the tool: 18-bus [164], 33-bus [165], 69-bus [166], and 141-bus [167] distribution systems, as described in Section 3.6.2.

The simulation parameters listed in Table 4.1 and Table 4.2 were consistent through all tests.

Table 4.1 Simulation Parameters for All Tests

Parameter	Value
Number of Charging Time Intervals $numT$	10
EV Charging Rate Limit in One Time Interval $limChr$	9.6 kW
Base transformer limit value for a single parking lot $limTf_{base}$	60 kW

Table 4.2 Electricity Prices Per Timeslot (from [40])

Time Interval	1	2	3	4	5	6	7	8	9	10
Price ($/kWh)	0.1	0.2	0.4	0.2	0.1	0.1	0.2	0.4	0.2	0.1

Since the PSO metaheuristic is non-deterministic [19], and therefore expected to produce different results each time, the average value over a large number of samples needed to be taken to produce a result more representative of the method's capabilities. As well, since simulation computation times may vary due to any number of background activities happening on the machines, the average computation time was taken to provide a better representation of the model's behaviour. All timing results found in the next sections include the parallel pool startup time for 192 workers.

In order to obtain statistical results that more accurately represented the method's capabilities, Tests 1 and 3 were was run 100 times independently, and Test 2 run 10 times independently (similar to the method used by [38]). For Test 4, the item being tested (iteration time) was only trialed five times, due to the long and impractical time required to run full versions of the speedup tests (for example, it would require over 750 hours or over 31 days to run 10 trials of the full centralized, sequential 27 parking lot model).

The optimum (best out of all test results), worst, median, mean, and standard deviation (std.) results for each test are provided in the sections below. The standard deviation being used or calculated for all tests used the default MATLAB normalization of $N - 1$, where N is the number of observations [1]. The average time or runtime required to run one independent test of each type is also included in the relevant results sections.

All results are provided with an accuracy of two to four decimal places and copied from the MATLAB results. There may be slight discrepancies between results in MATLAB and those printed here due to rounding.

4.2 Success Criteria

The aim of this thesis is to develop a solution for EV charge scheduling optimization for multiple parking lots, using centralized and metaheuristic optimization methods to provide more optimal solutions, and using parallel computing to increase the scalability of the solutions and to complete the optimization in an acceptable real-time interval.

As per Section 1.3.2, the success criteria for this thesis are:

1. Comparison of the results of at least three scenarios of different sizes (i.e., different network sizes, number of EVs, and parking lot locations), by fitness function metrics (i.e., total cost), between the centralized (parallelized) and decentralized algorithm (parallelized) models or methods. This is to determine if the algorithms are performing as intended, and to determine which method can produce the more optimal solution. The single parking lot scenario will be compared against the originating single parking lot model found in [40] to confirm that it can find results similar to those found by the authors of that paper. The centralized and decentralized methods will be compared against each other.

2. Comparison of the results of the single parking lot scenario, between the centralized sequential and parallelized methods. This is to determine the accuracy, or correctness, of the parallelization.

3. Comparison of the runtimes of the multi-parking lot scenarios, between the centralized sequential and parallelized models, for different problem sizes. This is to determine the speedup between the models for a variety of problem sizes, confirm that the model can continue to perform as intended as the problem size increases, and to determine the potential maximum problem size that can be handled by the model in the real-time limit. Full results for the multi-parking lot scenarios will not be compared due to the long runtimes required for the sequential versions.

These criteria will be evaluated through four main tests, testing the centralized algorithm's ability to optimize a single parking lot (Test 1, accuracy of the implementation, criterion #1), ability to optimize multiple parking lots (Test 2, accuracy of the implementation, criteria #1 and #3), the accuracy of its parallelization and its speedup when compared to sequential version of the single and multi-parking lot algorithms (Test 3 and 4, criteria #2 and #3), and the final cost outcome of the algorithm when compared to the decentralized version of the parallelized algorithm (Test 2, accuracy of the implementation of the decentralized version, criterion #1).

4.2.2 Accuracy of Implementation: Centralized Single and Multi-Parking Lot Optimization

For success criteria #1: to determine the accuracy, or correctness, of the centralized model for a single parking lot, a test must be conducted which runs the centralized multiple parking lot optimization model (Algorithm 2, which contains Algorithm 3), for a single parking lot ($n_{pl} = 1$). With the specifications for certain parameters to suite the testing of a single parking lot, the use of a separate program to run Algorithm 1 is not required. The test will use a modified parallelized version of Algorithm 3. The test results will be used for Test 1 (Section 4.3) and Test 3 (Section 4.5).

The cost solution output of the test will be compared to the cost or objective value solution of the 20 EV single parking lot optimization tests discussed in Section 6.1 of [40], the paper its objective function and general parameters were reproduced from. The results will be used to verify that the average, lowest, and highest electricity costs produced by the model are comparable to the range of results listed in [40]: \$73.69 for the Earliest Deadline First scheduling algorithm, to \$59.45 (59.4461) for the "original PSO" method, to \$53.77 (53.7660) for the proposed heuristic fuzzy PSO method. An averaged and a maximum result within the range of \$53.7660 to \$73.69 would show that the centralized model is performing as intended, providing an accurate result. A minimum or optimized result within the range of \$53.7660 to \$59.4461 would show that the centralized model is performing accurately and well.

For success criteria #1 and #3: to determine the accuracy of the centralized model for a multiple parking lots, and to confirm that the model can continue to perform as intended as the problem size increases, tests must be conducted which run the centralized multiple parking lot optimization model for a minimum of three different scenarios. Since there are no equivalent results to compare against in literature, the accuracy of the model implementation will be based on the cost results. With a base transformer limit of 60 kW, and likely cost output of \$54-\$74 for a single parking lot (an approximate average cost of \$63.73) [40], a solution for n_{pl} parking lots should be approximately $\$\, n_{pl} \times \$\, 64$, scaling linearly with the number of parking lots. Values close to this (below it, or within $\$\, n_{pl} \times \$\, 5$ above it) for the different scenario results will show that the centralized model is performing accurately. The test will use Algorithm 2, which contains Algorithm 3. The test results will be used for Test 2 (Section 4.4) and Test 4 (Section 4.6).

For success criteria #1: to determine which model is able to provide the more optimal result, the cost results for the centralized and decentralized models will be compared for a minimum of three different scenarios. The model which is able to produce the lower minimal and lower average result is the model able to produce the more optimal result.

4.2.3 Accuracy of Parallelization: Parallel Implementation of the Centralized Single and Multi-Parking Lot Algorithms

Accuracy of parallelization in this case refers to both the accuracy, or correctness, of the implementation of the sequential code into the parallelized version (does it provide the same result [125]) and the impact of the parallelization on the computational speed (does the code run faster i.e. provide a speedup). Per [123], to be considered successfully parallelized, the code must "not only run faster, but also produce identical results as its sequential counterpart" [123].

For success criteria #2: to determine the accuracy of the parallelization of the single parking lot models, tests must be run using both the parallelized and the sequential versions of the model, to test the accuracy of the implementation of the sequential code into the parallelized version. The expectation is that the parallelized and sequential versions will have very similar if not the same results (due to the non-deterministic PSO), so the accuracy of the parallelization will be based on the cost results. Costs for the sequential and parallelized results within \$5 of each other (minimum, maximum, median, and average) would show that the parallelization has been completed correctly. The other aspect of parallelization, the speedup provided by the parallelized version, will be measured be recording the code runtimes for both versions. An average speedup of more than 1 times ($s > 1$, see Section 2.5.5 and 4.2.5) will indicate that the model had been successfully parallelized. This test will use a modified parallelized version of Algorithm 3, and the non-parallelized version, with results used for Test 3 (Section 4.5).

Full cost results for the multi-parking lot scenarios will not be compared due to the long runtimes required for the sequential versions which were determined to be impractical to measure for this thesis. Instead, only average program runtimes for a single iteration of the outer PSO of Algorithm 2 were used to determine speedup and therefore accuracy of parallelization. The other aspect of the accuracy of the parallelization, in terms of it being able to produce the correct or expected cost result, will be confirmed by the tests discussed in Section 4.2.2 which determine the accuracy of the models' implementation.

For success criteria #3: to determine the speedup between the centralized sequential and parallelized models, for different problem sizes, tests must be run which measure the runtime of at least three different multi-parking lot scenarios. An average speedup of more than 1 times will indicate that the parallelization was done accurately. The runtimes must also be compared between the parallelized models to determine which models were able to complete their optimizations within the 15 minute time limit imposed in this research (see Section 2.5.4): this will find maximum problem size that can be handled by the model in the real-time limit. The tests will use the parallelized and sequential version of Algorithm 2. The test results will be used for Test 2 (Section 4.4) and Test 4 (Section 4.6).

4.2.4 Accuracy of Implementation: Parallel Implementation of the Decentralized Multi-Parking Lot Algorithm

For success criteria #1: to determine the accuracy, or correctness, of the implementation of the decentralized model for a multiple parking lots, tests must be conducted which run the decentralized multiple parking lot optimization model for a minimum of three different scenarios. The cost results for this model will be compared against the cost results for the centralized model. The same cost values to determine accuracy will be used for the results of this model: a solution for n_{pl} parking lots should be approximately $\$\, n_{pl} \times \$\, 64$, scaling linearly with the number of parking lots. Cost results below this values, or within $\$\, n_{pl} \times \$\, 5$ if above it for the different scenario results will show that the decentralized model is performing accurately, as intended. The test will use Algorithm 4, which contains Algorithm 3. The test results will be used for Test 2 (Section 4.4).

In terms of the accuracy of the parallelization of the decentralized model, no separate tests will be run which calculate the speedup of the model versus a sequential version. However, the runtime results of the decentralized model tests can be compared to the runtime results of the sequential single parking lot optimization model. Since Algorithm 4 has only one instance of evaluating its parking lots in parallel, for any scenarios where the number of parking lots is less than or equal to the number of HPC cluster workers, the scenario runtime should be equal to the runtime required to sequentially optimize a single parking lot. Therefore, if the minimum, maximum, and average runtime for the decentralized model scenarios are within the same general range of the minimum, maximum, and average runtime of the sequential single parking lot results (for Test 3 in Section 4.5), then the decentralized model can be considered to be accurately parallelized.

4.2.5 Speedup of the Parallelized Algorithms

To compare the success of the parallelization of the algorithms per success criteria #2 and #3, one of the metrics to be used the speedup. The simplified version of the calculation from equation (2.10) will be used, with the speedup being the ratio of the computational runtime of the parallelized algorithm to the computational runtime of the sequential algorithm.

As discussed in Section 4.1, the non-deterministic nature of metaheuristic-based algorithms requires an average value of a selected item, taken over a large number of samples, to produce a result more representative of the method's capabilities. Therefore, to calculate a speedup more representative of the algorithm's capabilities, the speedup will be calculated using the average algorithm runtimes:

$$s = \frac{\frac{1}{n}\sum_{i=1}^{n} T_{seq_i}}{\frac{1}{m}\sum_{j=1}^{m} T_{par_j}} \tag{4.1}$$

where the speedup s is the ratio of the parallelized algorithm computation runtime T_{par} averaged over $i = 1, 2, 3, \dots n$ rounds of tests, to the computational runtime of the sequential algorithm T_{seq}, averaged over j = 1, 2, 3, … m rounds of tests. For all tests in this section, $m = n$.

4.3 Test 1: Single Parking Lot Optimization

This test was conducted to verify the performance of the PSO optimization of a single parking lot, by comparing its results to the results in Section 6.1 of the paper its method was based on: [40]. The aim of the test was to meet success criteria #1, and verify that the average, lowest, and highest electricity costs produced by the algorithm were comparable to the range of results listed in [40]: from to \$53.77 (53.7660) for the proposed heuristic fuzzy PSO method, to \$73.69 for the Earliest Deadline First scheduling algorithm. This test demonstrated the performance of Algorithm 1, by using the parallelized version of Algorithm 3 (the inner PSO of Algorithm 2), optimizing the charging of a single parking lot with a given number of EVs and their charging request profiles, to obtain the lowest cost.

This test used the simulation parameters listed in Table 4.3, in addition to those listed in Table 4.1 and Table 4.2. This test was run on the single, multicore workstation, using 9 of the 20 available cores.

Table 4.3 Simulation Parameters – Test 1

Parameter	Value
Number of Parking Lots	1
Number of EVs	20
Number of Particles (Outer PSO)	1
Number of Iterations (Outer PSO)	1
Number of Particles (Inner PSO)	100
Number of Iterations (Inner PSO)	500
Distribution System Test Case	18-bus

The test used the EV charging request profile of EV Parking Lot 1, listed in Table 3.2. The parking lot was connected to bus 2 of the 18-bus system, as described in Section 3.6.2.

The test description is as follows: the modified version of the centralized algorithm (Algorithm 2) was run with the specified parameters, 100 times independently. The final cost, fitness, and algorithm runtime were recorded, along with graphical representations of the fitness and cost for the PSOs.

The same test was also run with the sequential version of the algorithm in order to obtain speedup results: this test and the subsequent results are described for Test 3 in Section 4.5 below. The results from Test 1 were used in Test 3 as the parallelized results.

4.3.1 Results

A summary of the results of this test are listed in Table 4.4 to Table 4.7, and the test's optimum results shown in Figure 4.1 and Figure 4.2. A segment of the full results can be found in Appendix B. Table 4.5 has been coloured to better visualize the charging schedules of the EVs in the parking lot, where green indicates the EV is charging and yellow indicates no charging by that EV in that time interval even though it remains plugged into the grid.

In Table 4.4, for the cost and test runtime, the optimum value is the minimum value out of all tests. For the fitness, the optimum value is the maximum value out of all tests.

Table 4.4 Single Parking Lot Results: Cost and Time

	Cost ($)	**Fitness**	**Time required to run 1 test (inner PSO parallelized) (s)**
Optimum	53.77	1.0183	14.91
Worst	61.36	1.0160	19.25
Median	57.16	1.0172	15.32
Mean	57.09	1.0172	15.30
Std.	1.83	0.00	0.44

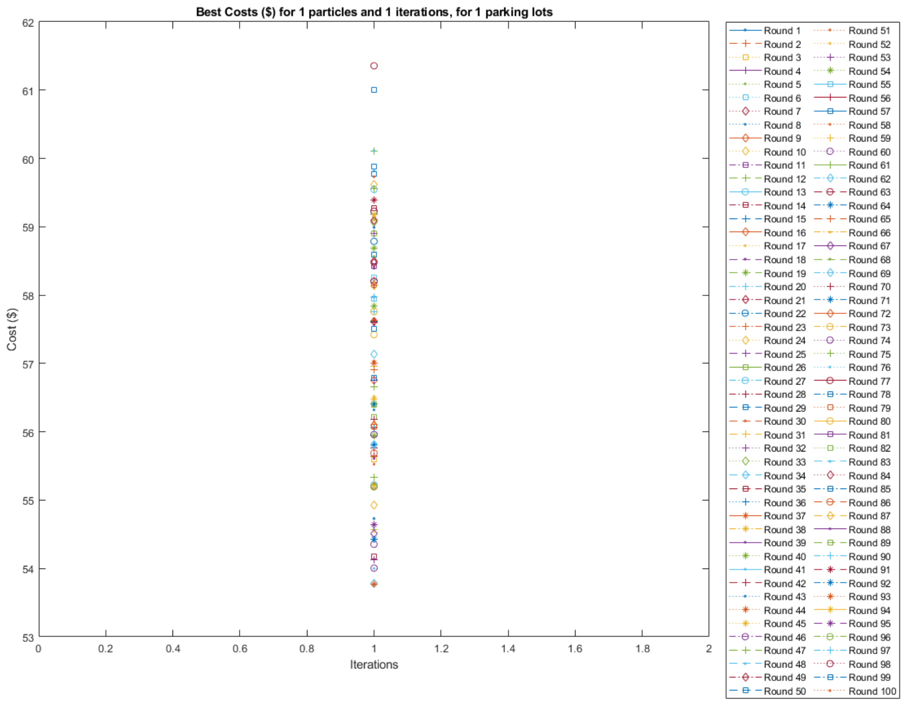


Figure 4.1 Final Cost for all 100 Test Rounds

The results in Table 4.4 and Figure 4.1 show that Algorithm 1 is able to produce results within the range of objective values produced in [40]: the optimum result was within \$0.004 of the optimal result produced in that paper, and the mean, median, and worst (or highest) results were comparable to the results produced by the construction, inertia-weight, and original PSOs in [40], within \$1.92. These results show that the algorithm was able to function accurately, producing an optimized charging schedule for a set of EVs with known arrival and departure times and demands, when provided the cost of electricity for all time intervals, the maximum rate of charging for one time interval, and the maximum transformer capacity or limit for one time interval. The feasible results produced by this algorithm also demonstrate that the 60 kW transformer limit, and therefore the maximum real power demand at any time interval, is a permissible demand on bus two of the 18-bus system, as the voltage limits are respected at all buses within the system.

The closeness of the objective values to those produced by the improved and other PSO-based algorithms in [40] may be attributed in part to the increased number of particles and iterations used in these tests: 100 particles and 500 iterations versus the 28 particles and 50 iterations used in [40], a 3.5 and 10-fold increase. This increase in particles and iterations is reflected in the average runtime of 15.30 seconds for one test, a 2.5-fold increase compared to the 6.2 second runtime for the original PSO in [40].

Table 4.5 shows the optimum schedule determined by the algorithm in 10 rounds, with a total objective value (electricity cost) of \$53.77 (53.76934, see Table 4.7), total power demand (consumption) of 352 kWh (see Table 4.6), and unmet demand of approximately 0 kWh, or no unmet demand (see Table 4.8). Table 4.6 and Table 4.7 show that the total demand (consumption) and cost at each time interval follow the electricity price pattern from Table 4.2 (and therefore from [40]); most EVs are present during intervals 3 to 8, and the highest total demands occur when the price is the lowest at intervals 5 and 6, and the lowest demands occur when the price is highest, at intervals 3 and 8. Table 4.8 shows that the algorithm was able to optimize the schedule and meet the demand (desired consumption) of each EV to the kWh, only over or undercharging EVs 5, 9, 10, 13, 14, and 19 by approximately two to four pWh.

Table 4.5 Optimal Schedule for Parking Lot 1 (kWh)

Time Interval	1	2	3	4	5	6	7	8	9	10
EV 1	9.5592	8.4408	0	-	-	-	-	-	-	-
EV 2	-	-	0	7.5000	7.5000	-	-	-	-	-
EV 3	9.5239	3.4930	1.7954 $\times 10^{-8}$	7.9677	4.0155	-	-	-	-	-
EV 4	-	9.1548	0.0059525	8.8992	-	-	-	-	-	-
EV 5	-	9.5989	5.4011	-	-	-	-	-	-	-
EV 6	-	-	-	-	-	7.7530	4.6644	0	0	9.5826
EV 7	-	-	-	-	-	-	9.5119	4.4881	-	-
EV 8	-	-	-	-	-	-	-	0	6.4584	9.5416
EV 9	-	-	-	-	-	-	9.5828	0.41722	-	-
EV 10	-	-	-	-	-	9.2887	3.6175	0	7.0938	-
EV 11	-	-	0	1.3405	8.2198	8.2198	8.2198	4.2514 $\times 10^{-8}$	-	-
EV 12	-	4.7788 $\times 10^{-5}$	0	6.6806	4.6476	5.6717	-	-	-	-
EV 13	-	-	-	-	7.3372	7.6628	0	0	-	-
EV 14	-	-	-	9.0084	0.009424	3.4839	3.4984	0.00034914	-	-
EV 15	-	-	0.0013075	1.5723	4.9773	5.1414	2.3076	-	-	-
EV 16	-	-	-	-	5.4180	2.8946	3.6874	0	-	-
EV 17	-	-	-	1.6007	5.4646	3.0561	5.8786	-	-	-
EV 18	-	-	-	-	9.2136	4.8019	2.4750	0	2.5095	-
EV 19	9.5822	1.01481	0.0049156	3.7445	0.11984	2.0031	2.9279	0.0030483	0	9.5996
EV 20	9.5884	0.38607	1.5919 $\times 10^{-6}$	2.9520	3.0735	-	-	-	-	-

Table 4.6 Total Power Demand (Consumption) for Optimal Schedule Solution (kWh)

Time Interval	1	2	3	4	5	6	7	8	9	10	Total Demand
Power Demand	38.254	31.088	5.4132	51.206	59.996	59.977	56.371	4.9087	16.062	28.724	352

Table 4.7 Cost for Optimal Schedule Solution ($)

Time Interval	1	2	3	4	5	6	7	8	9	10	Total Cost
Cost	3.8254	6.2177	2.1653	10.241	5.9996	5.9977	11.274	1.9635	3.2123	2.8724	53.769

Table 4.8 EV Demand (Consumption) for Optimal Schedule for Parking Lot 1 (kWh)

Time Interval	Desired Demand (consumption)	Met Demand* (consumption)	Unmet Demand (consumption)
EV 1	18	18	0
EV 2	15	15	0
EV 3	25	25	0
EV 4	18	18	0
EV 5	15	15	-1.7764×10^{-15}
EV 6	22	22	0
EV 7	14	14	0
EV 8	16	16	0
EV 9	10	10	-1.7764×10^{-15}
EV 10	20	20	3.5527×10^{-15}
EV 11	26	26	0
EV 12	17	17	0
EV 13	15	15	-1.7764×10^{-15}
EV 14	16	16	-1.7764×10^{-15}
EV 15	14	14	0
EV 16	12	12	0
EV 17	16	16	0
EV 18	19	19	0
EV 19	28	28	-3.5527×10^{-15}
EV 20	16	16	0

*value rounded to nearest kWh

Figure 4.2 shows the behaviour of the objective function's value (cost) over the iterations, showing the expected steady improvement over all iterations, with the most dramatic improvement occurring in the first 150 iterations. This figure shows that the behaviour of the algorithm follows the typical PSO convergence pattern, similar to those shown in Figure 6 of [40].

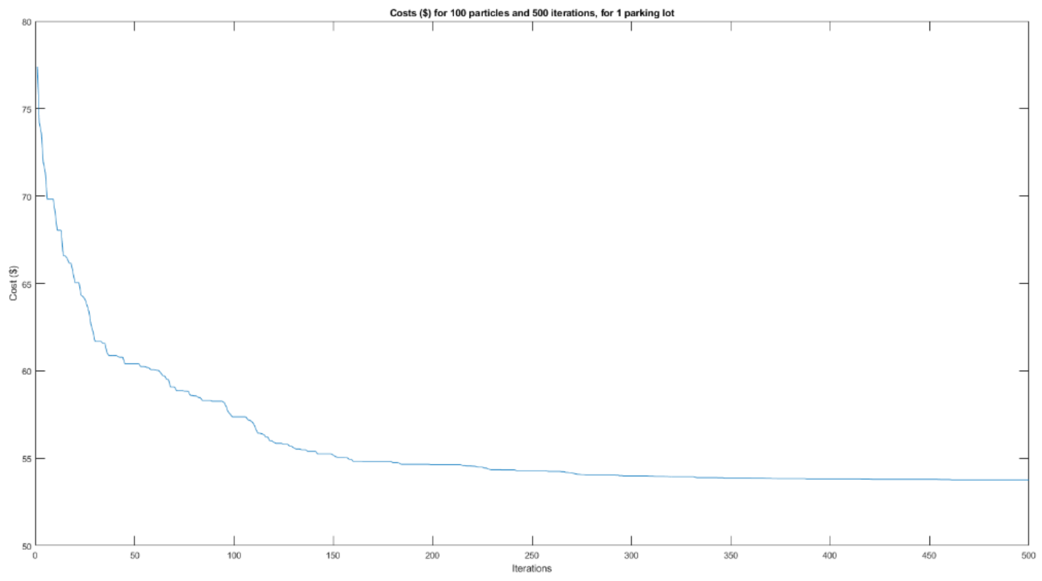


Figure 4.2 Change in Cost over Iterations for Optimum Test Round (Round 93)

Overall, the results of this test show that the algorithm and its selected PSO parameters were able to perform as expected and produce good results, providing an optimized EV charging schedule for a 20-EV parking lot, with final costs in the same range as the results provided by the paper the algorithm was based on. The lowest cost was very close to the lowest cost provided in the paper, demonstrating that the algorithm could be expected to provide very good results when required to optimize the charging schedule of a single parking lot with a given number of EVs and their charging request profiles, to obtain the lowest cost.

The results of this test led to the conclusion that this algorithm would be able to optimize different parking lot charge profiles using different transformer limits, enabling the use of the same algorithm to independently optimize different parking lots with different parameters.

4.4 Test 2: Multiple Parking Lot Optimization: Centralized versus Decentralized Methods

This test was conducted to compare the results of the centralized optimization method (Algorithm 2) with the decentralized optimization method (Algorithm 4). The aim of this test was twofold: 1) to determine if the decentralized model performed as expected (i.e., produced comparable results to the centralized model in terms of algorithm behaviour to optimize the individual parking lots); and 2) to observe if one hypothesis was correct: determining if the costs results obtained from the centralized model were lower on average (i.e., "better") than those of the decentralized model.

The results of these tests were also used for two main areas of analysis and discussion: 1) to compare feasibility of the results of the centralized optimization method as the number of parking lots increased; and 2) to compare algorithm runtime of the centralized optimization method, for different inner PSO iteration totals. The aim of this analysis was to determine what potential combinations of PSO parameters had an average algorithm runtime, over of all sub-tests conducted, under the 15-minute real-time limit set in Section 2.5.4 above, for simulations with 3, 9, 18, and 27 parking lots. This test also analyzed the optimum costs versus the general feasibility and average runtime of that test case, to determine the potential trade-off between the potential cost savings versus time required to calculate that result and the risk of calculating an infeasible result.

The test results of the centralized model demonstrated the performance of Algorithm 2, optimizing the power division between multiple 20 EV parking lot loads (the outer PSO), and the charge scheduling of EVs within the parking lot (inner PSO) with a given number of EVs and their charging request profiles, to obtain the lowest cost. The test results of the decentralized model demonstrated the performance of Algorithm 4, optimizing the charge scheduling of the same EVs within the parking lots with a set transformer limit of 60 kW each, to obtain the lowest cost.

This test used the simulation parameters listed in Table 4.9, in addition to those listed in Table 4.1 and Table 4.2. This test consisted of six main sub-tests, which used different combinations of parking lots and distribution system test cases. The test used different combinations of the EV charging request profiles of EV Parking Lots 1-9, listed in Appendix A. Since there are almost unlimited number of possible combinations of population and iteration sizes for the outer and inner PSOs, the main test cases (2.1-2.6) use the selected base case of 50 and 100 particles with 10 and 500 iterations for the outer and inner PSOs. The additional cases used with the 3, 9, 18, and 27 parking lot scenarios, which used 250 iterations of the inner PSO, were selected from experimental trials to provide results to compare against the base case in terms of cost, feasibility, and runtime (tests 2.7-2.10).

Table 4.9 Simulation Parameters – Test 2

Parameter	Value					
Sub-test no.	**2.1**	**2.2**	**2.3**	**2.4**	**2.5**	**2.6**
Number of Parking Lots	3	9	9	18	18	27
Number of EVs	60	180	180	360	360	540
Number of Particles (Outer PSO)	50	50	50	50	50	50
Number of Iterations (Outer PSO)	10	10	10	10	10	10
Number of Particles (Inner PSO)	100	100	100	100	100	100
Number of Iterations (Inner PSO)	500	500	500	500	500	500
Distribution System Test Case	18-bus	18-bus	33-bus	69-bus	141-bus	141-bus
Sub-test no.	**2.7**	**2.8**	**2.9**	**2.10**		
Number of Parking Lots	3	9	18	27		
Number of EVs	60	180	360	540		
Number of Particles (Outer PSO)	50	50	50	50		
Number of Iterations (Outer PSO)	10	10	10	10		
Number of Particles (Inner PSO)	100	100	100	100		
Number of Iterations (Inner PSO)	250	250	250	250		
Distribution System Test Case	18-bus	33-bus	69-bus	141-bus		

For Test 2.1 and 2.7, the profiles of Parking Lots 1-3 were used, and the parking lots were connected to buses 2, 12, and 16 of the 18-bus system.

For Test 2.2, the profiles of Parking Lots 1-9 were used, and the parking lots were connected to buses 2, 4, 6, 7, 8, 22, 23, 24, and 26 of the 18-bus system, as shown in Figure 3.8.

For Test 2.3 and 2.8, the profiles of Parking Lots 1-9 were used, and the parking lots were connected to buses 7, 8, 14, 24, 25, 29, 30, 31, and 32 of the 33-bus system, as shown in Figure 3.9.

For Test 2.4 and 2.9, the profiles of Parking Lots 1-9 were duplicated to make 18 parking lots. The parking lots were connected to buses 2, 3, 4, 5, 15, 19, 23, 25, 30, 31, 32, 38, 42, 44, 47, 56, 57, and 58 of the 69-bus system, as shown in Figure 3.10.

For Test 2.5, the profiles of Parking Lots 1-9 were duplicated to make 18 parking lots. The parking lots were connected to buses 2, 7, 22, 46, 70, 118, 4, 10, 38, 63, 78, 81, 95, 99, 102, 114, 120, and 131 of the 141-bus system.

For Test 2.6 and 2.10, the profiles of Parking Lots 1-9 were tripled to make 27 parking lots. The parking lots were connected to buses 2, 4, 6, 7, 10, 16, 22, 28, 31, 38, 46, 54, 63, 70, 78, 81, 85, 91, 95, 99, 102, 108, 114, 118, 120, 125, and 131 of the 141-bus system, as shown in Figure 3.11.

The test description is as follows: for each sub-test, the centralized parallelized algorithm (Algorithm 2) was run with the specified parameters, 10 times independently. Then, within the same sub-test, the parallelized decentralized algorithm (Algorithm 4) was run with the same parameters, 10 times independently. The final cost, fitness, feasibility, and algorithm runtime were recorded, along with graphical representations of the fitness and cost, for both versions of the algorithm, for the inner and outer PSOs.

4.4.1 Results

A summary of the results of sub-tests 2.1-2.6 are listed in Table 4.10 to Table 4.11, and shown in Figure 4.3 to Figure 4.13 (showing results from the 9 parking lot scenario from test 2.3 only). The summarized results for all tests, including 2.7-2.10, can be found in Table 6.9 and Table 6.10 of Appendix C, and have been included in Figure 4.11 to Figure 4.13 for comparison purposes.

As will be described in Section 4.4.3, some of the results for the tests with larger numbers of parking lots were not feasible. All results are included in Table 4.10, but for the results and comparison in Section 4.4.2 and Table 4.11, these infeasible results have been omitted.

In Table 4.10, the cost is the total cost required to charge all parking lots in that scenario. The fitness values for the decentralized results are taken from all fitness values across all parking lots, across all test rounds. The time is for the time required to run one test. The feasibility columns list the total number of feasible test results out of 10 (i.e., 8), followed by its conversion into percentage (i.e., 80%). For the cost and runtime, the optimum value is the minimum value out of all tests. For the fitness, the optimum value is the maximum value out of all tests.

Table 4.10 Centralized and Decentralized Multiple Parking Lot Results: Cost, Fitness, Time, and Feasibility

Sub -test		Centralized				Decentralized			
		Cost ($)	**Fitness**	**Time (s)**	**Feasibility**	**Cost ($)**	**Fitness**	**Time (s)**	**Feasibility**
2.1	Optimum	182.17	1.0055	253.78	10	189.53	1.0180	17.43	10
(3	Worst	187.39	1.0053	307.63	(100%)	197.49	1.0136	21.54	(100%)
park	Median	183.49	1.0054	264.17		193.00	1.0148	17.48	
ing	Mean	184.31	1.0054	267.40		193.81	1.0154	17.90	
lots)	Std.	1.80	0.00	14.81		2.78	0.00	1.28	
2.2	Optimum	527.75	1.0019	531.90	10	556.56	1.0268	19.17	10
(9	Worst	535.29	1.0019	574.48	(100%)	578.57	1.0106	36.07	(100%)
park	Median	531.15	1.0019	537.29		565.98	1.0167	19.38	
ing	Mean	531.65	1.0019	541.09		566.08	1.0168	21.02	
lots)	Std.	2.43	0.00	12.60		6.47	0.00	5.29	
2.3	Optimum	527.45	1.0019	525.51	10	550.41	1.0264	17.99	10
(9	Worst	541.25	1.0018	576.56	(100%)	573.96	1.0107	46.13	(100%)
park	Median	532.15	1.0019	536.89		564.51	1.0169	18.20	
ing	Mean	532.82	1.0019	538.76		563.94	1.0168	21.92	
lots)	Std.	3.93	0.00	14.56		6.89	0.00	8.73	
2.4	Optimum	1082.49	1.0009	943.63	10	1117.15	1.0271	19.09	8
(18	Worst	1107.87	1.0009	1018.75	(100%)	*1156.95*	*0.2033*	37.58	(80%)
park	Median	1091.19	1.0009	967.31		1128.38	1.1068	25.59	
ing	Mean	1094.43	1.0009	960.92		1130.15	1.0079	25.78	
lots)	Std.	9.98	0.00	22.05		11.93	0.09	4.72	
2.5	Optimum	1079.51	1.0009	973.00	10	1116.44	1.0267	18.37	9
(18	Worst	1117.53	1.0009	1021.54	(100%)	1142.61	*0.2033*	37.58	(90%)
park	Median	1087.05	1.0009	988.55		1131.05	1.0168	18.46	
ing	Mean	1091.42	1.0009	991.39		1130.32	1.0124	20.41	
lots)	Std.	12.79	0.00	13.33		9.31	0.06	6.03	
2.6	Optimum	1624.80	1.0006	1455.90	9	1687.61	1.0272	18.98	10
(27	Worst	*1747.82*	*0.7895*	1628.01	(90%)	1724.23	1.0105	43.34	(100%)
park	Median	1657.37	1.0006	1483.13		1692.60	1.0166	21.60	
ing	Mean	1667.83	*0.9795*	1497.78		1695.77	1.0169	23.70	
lots)	Std.	38.37	0.07	49.21		11.20	0.00	7.4	

Results in *italic* denote an infeasible result

The results of Table 4.10 show that the centralized algorithm was able to provide a more optimal result (i.e., a lower cost) than the decentralized algorithm, for all scenarios. As the number of parking lots increased, the likelihood that the algorithms would produce an infeasible final test result increased. As the number of parking lots increased, the runtime for both algorithms increased, with the centralized algorithm taking a much longer time to complete a single test due to it completing 10 iterations of the outer PSO in order to optimize the transformer limits or maximum real power demand for each individual parking lot. These results are discussed in more detail in Sections 4.4.2 to 4.4.4 below.

A feasible solution was one where which did not violate any constraints: where the inner PSO solution, for the optimization of the charge schedule of a specific parking lot with a specific transformer limit, did not exceed the transformer limit or the charging rate, and was within 1 W of the demand (desired consumption) across all EVs (per equations (3.7)-(3.8), (3.10)-(3.15), (3.18)), and the outer PSO solution, for the optimization of the individual parking lot transformer limits, did not exceed the voltage magnitudes at all the buses where the parking lots were located (equations (3.23)-(3.27), (3.30)). The transformer limits, i.e., the maximum additional active power demands at the assigned buses, optimized by the outer PSO were tested for compliance with the system voltage limits through the power flow simulation.

The following figures show the cost and fitness results for all 10 test rounds for the centralized (Figure 4.4 to Figure 4.6) and decentralized algorithms (Figure 4.7 to Figure 4.10) of sub-test 2.3, in order to show PSO convergence behaviour for the inner and outer PSOs of the centralized algorithm, and the PSOs of the decentralized algorithm.

Figure 4.3 shows all test cost results for the centralized and decentralized algorithms of sub-test 2.3, clearly showing the improvement in the optimization objective of total cost of the centralized algorithm when compared to the decentralized algorithm. Even between the highest cost of the centralized algorithm ($541.25) and the lowest of the decentralized algorithm ($550.41), there was still a different of $9.16. The figure shows that the improvement in results remained consistent across all tests, demonstrating the overall better performance (in terms of cost) of the centralized algorithm.

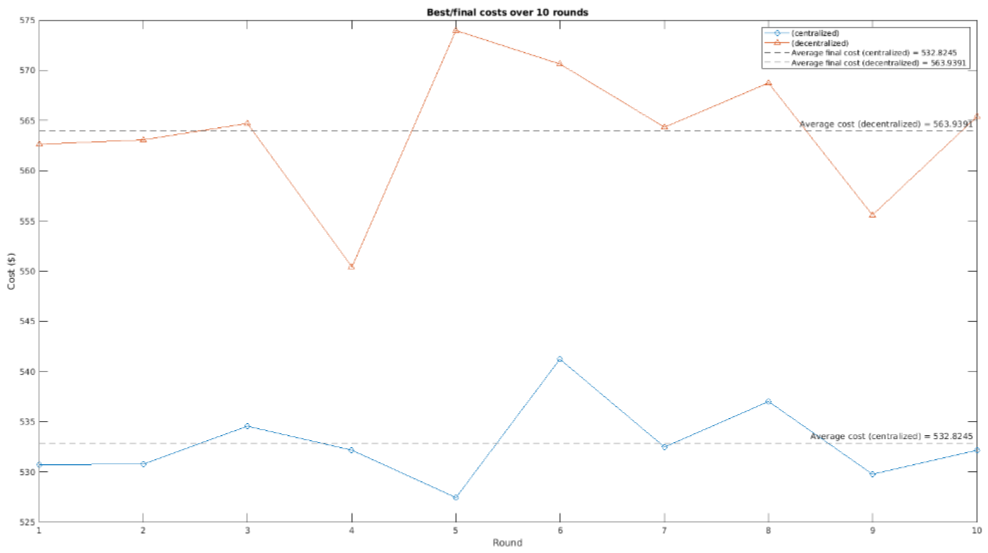


Figure 4.3 Final Costs for Centralized and Decentralized Test Rounds of Test 2.3

Figure 4.4 shows the convergence behaviour of the objective value of the outer PSO of the algorithm, showing fairly steady improvement over all iterations for each test round.

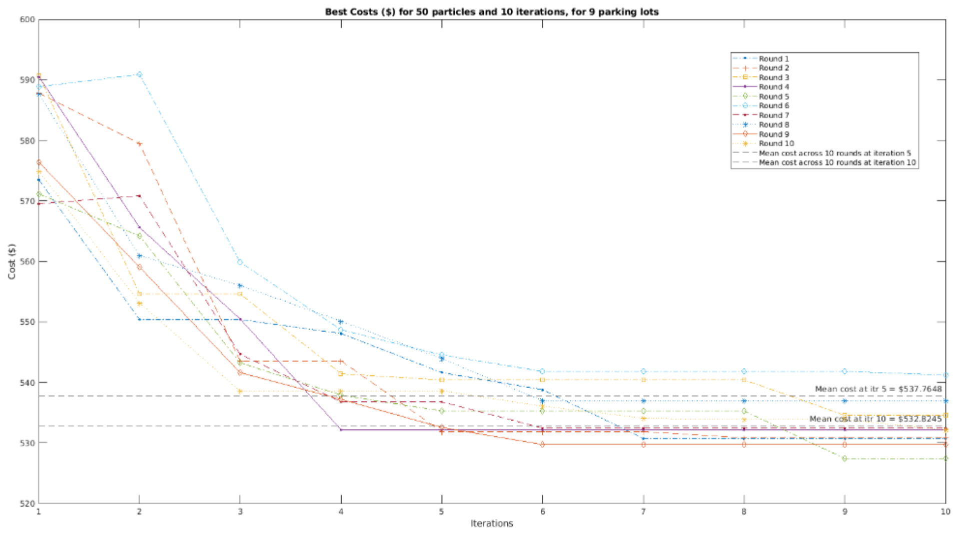


Figure 4.4 Change in Cost over Iterations for 10 rounds of Centralized Algorithm of test 2.3 (outer PSO)

Figure 4.5 and Figure 4.6 show the convergence behaviour of the fitness function value of the outer PSO. They show that while only one test round (round 4) had a feasible result at the end of its first iteration, by the third iteration all test rounds had generated a feasible solution, and continued to improve their objective values over each subsequent iteration. These figures also clearly show the behaviour of the PSO in terms of how the fitness function values change depending on the feasibility of the results (as per equation (3.30)): all infeasible results have a fitness below 1, but as soon as the result is feasible, the value is instantly equal to 1 or more, and the fitness value increases in much smaller increments after this point, as it increases in relation to the improvement in cost only. In order to see the improvement after reaching feasibility, the results must be enlarged as they are in Figure 4.6.

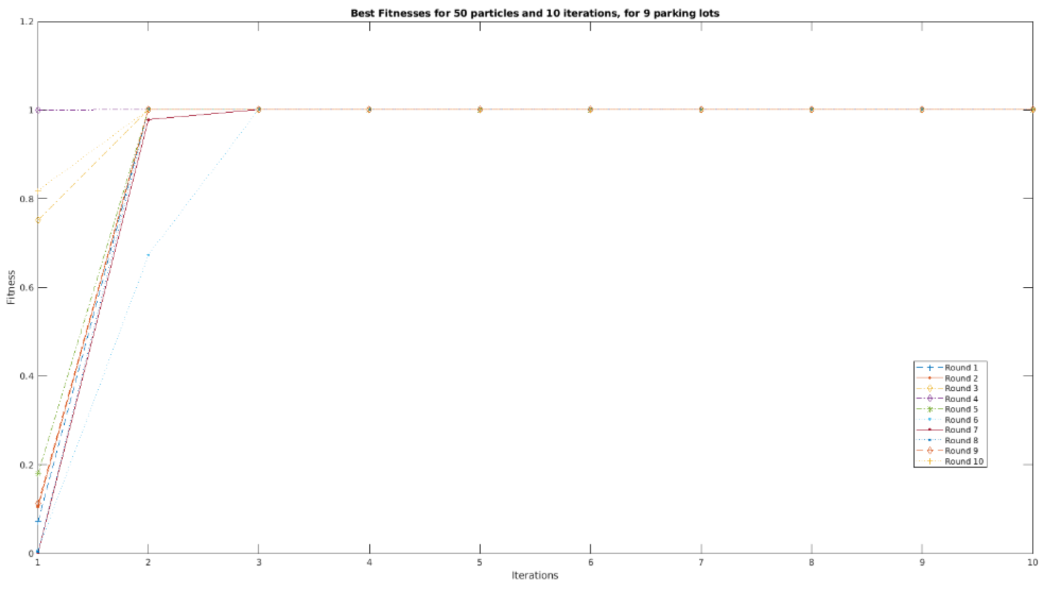


Figure 4.5 Change in Fitness over Iterations for 10 rounds of Centralized Algorithm of test 2.3 (outer PSO)

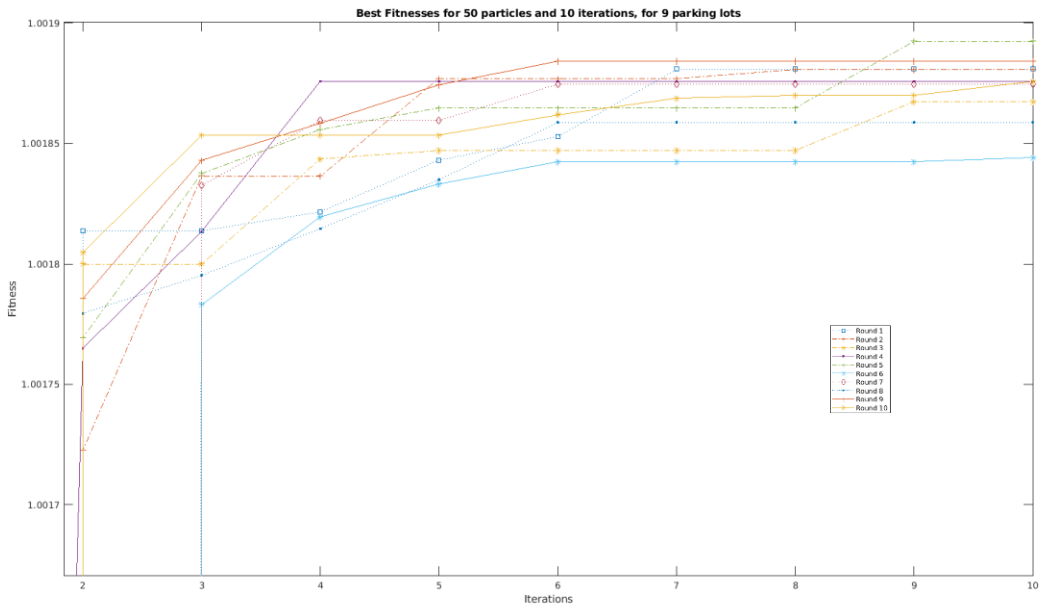


Figure 4.6 Enlarged Segment of Figure 4.5

Figure 4.7 shows the general convergence behaviour of the objective value of the single PSO of the decentralized algorithm, showing the total cost of all nine parking lots summed together, at each iteration. It shows the same general behaviour as that for the single parking lot optimization in Figure 4.2, with fairly steady improvement over all iterations for each test round, with the most dramatic improvement occurring in the first 150 iterations.

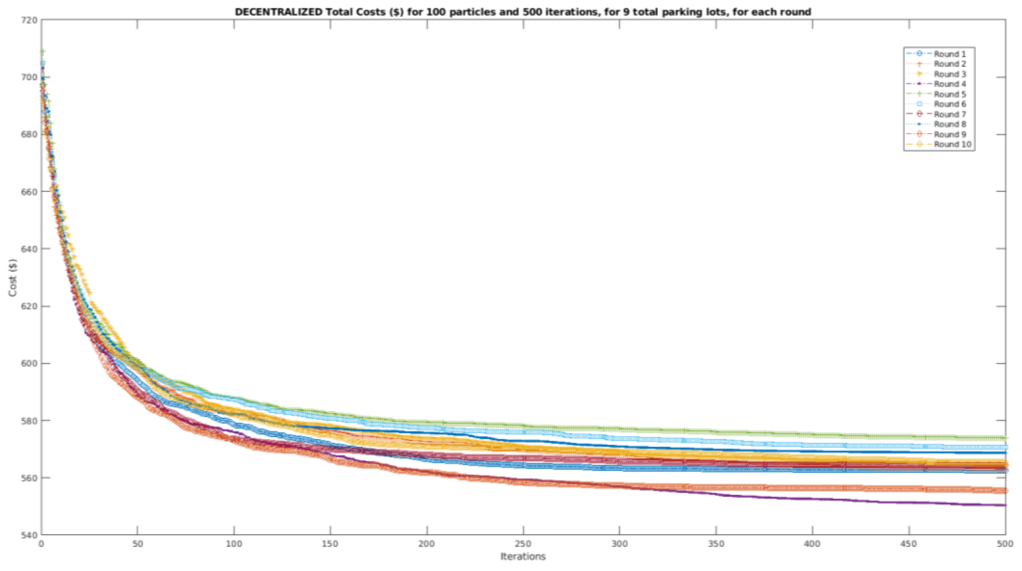


Figure 4.7 Change in Cost over Iterations for 10 rounds of Decentralized Algorithm of test 2.3

Figure 4.8, Figure 4.9, and Figure 4.10 show the convergence behaviour of the fitness function value of the PSO for each individual parking lot of each test round (a total of 90 objects in each graph). They show that while many tests began with infeasible parking lot solutions, by iteration 35 all test rounds had generated feasible solution for all parking lots, and thus for their round. They show the continued improvement of the objective values over each subsequent iteration. As with the results for the centralized algorithm, these figures also clearly show the behaviour of the PSO in terms of how the fitness function values change depending on the feasibility of the results (as per equation (3.18)): all infeasible results have a fitness below 1, but as soon as the result reach feasibility, the value is instantly equal to 1 or more, and increases in much smaller increments after that point.

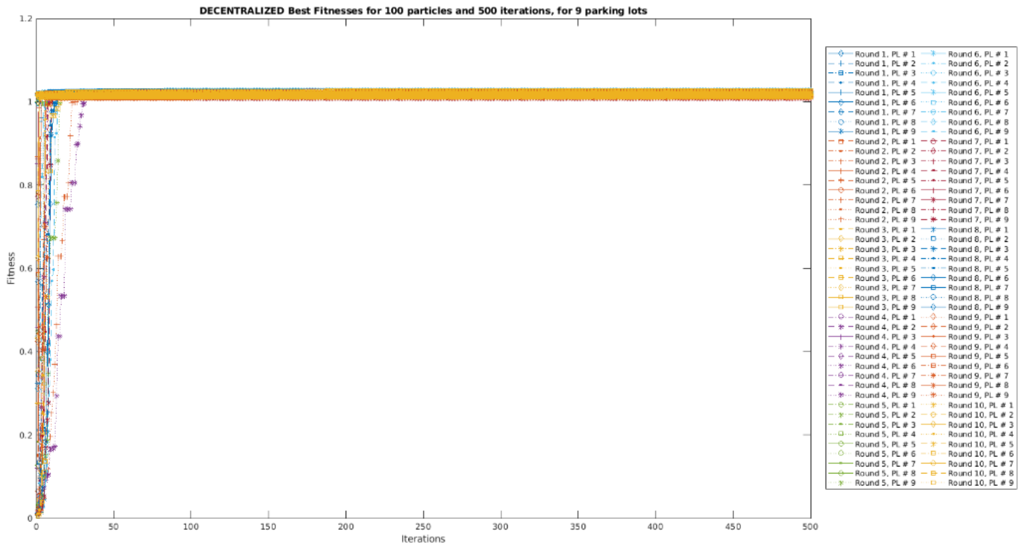


Figure 4.8 Change in Fitness over Iterations for 10 rounds of Decentralized Algorithm of test 2.3

Figure 4.9 shows the fitness values before all parking lots reached feasible solutions by iteration 35, and Figure 4.10 shows the improvement in fitness results after each solution reached feasibility, showing the expected gradual improvement and general PSO convergence behaviour.

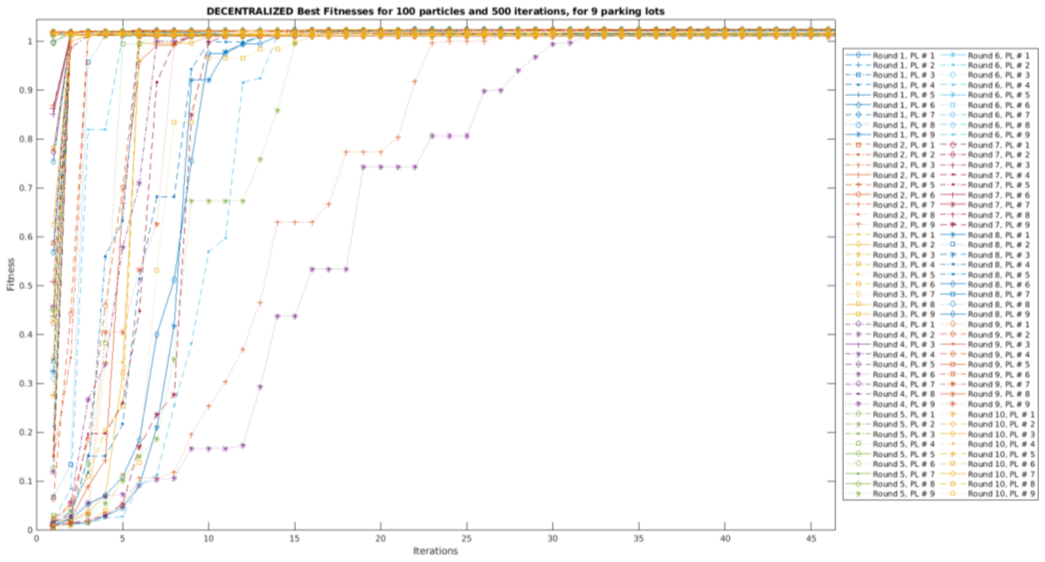


Figure 4.9 Enlarged Segment of Figure 4.8

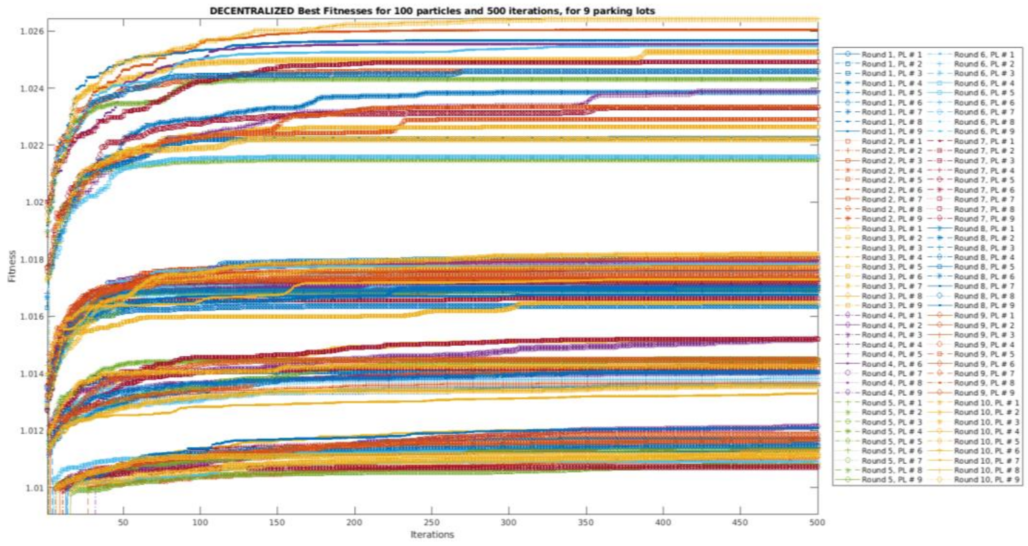


Figure 4.10 Enlarged Segment of Figure 4.8

The results of sub-test 2.3 displayed in the above tables and figures clearly illustrate the overall successful behaviour of the centralized and decentralized algorithms, in being able to produce comparable results, and in optimizing the total cost of a nine parking lot EV charge scheduling scenario, while ensuring the voltage magnitudes of the associated 33-bus system remained within the specified voltage limits.

4.4.2 Discussion: Cost

This section discusses the results of Test 2 as they pertain to the costs for the two algorithms.

Table 4.11 lists the same cost results as Table 4.10, but includes a column listing the difference between the centralized and decentralized costs for each result type. As with Table 4.10, the optimum value is the minimum value out of all tests, and the worst value is the maximum value.

Table 4.11 Cost Comparison for all Sub-Tests ($)

Sub-test		Centralized	Decentralized	Difference
2.1 (3 parking lots)	Optimum	182.17	189.53	7.36
	Worst	187.39	197.49	10.10
	Median	183.49	193.00	9.51
	Mean	184.31	193.81	9.50
	Std.	1.80	2.78	
2.2 (9 parking lots)	Optimum	527.75	556.56	28.81
	Worst	535.29	578.57	43.28
	Median	531.15	565.98	34.83
	Mean	531.65	566.08	34.43
	Std.	2.43	6.47	
2.3 (9 parking lots)	Optimum	527.45	550.41	22.96
	Worst	541.25	573.96	32.71
	Median	532.15	564.51	32.36
	Mean	532.82	563.94	31.12
	Std.	3.93	6.89	
2.4 (18 parking lots)	Optimum	1082.49	1117.15	34.66
	Worst	1107.87	1143.08	35.21
	Median	1091.19	1125.87	34.68
	Mean	1094.43	1126.94	32.51
	Std.	9.98	8.28	
2.5 (18 parking lots)	Optimum	1079.51	1116.44	36.93
	Worst	1117.53	1142.61	25.08
	Median	1087.05	1130.51	43.46
	Mean	1091.42	1126.61	35.19
	Std.	12.79	9.58	
2.6 (27 parking lots)	Optimum	1624.80	1687.61	62.81
	Worst	1720.65	1724.23	3.58
	Median	1656.28	1692.60	36.32
	Mean	1658.94	1695.77	36.83
	Std.	27.71	11.20	

Table 6.10 in Appendix C shows the difference in cost between the centralized and decentralized versions for all inner PSO iteration variations.

These tables show that there is an increase in cost as the number of parking lots increases, and in all cases, the centralized algorithm produced a lower cost solution, and therefore more optimal solution. For both the centralized and decentralized algorithms, the cases using more iterations of the inner PSO produced lower costs than those using fewer, as would be expected: double the number of iterations provides a much larger chance of the PSO finding a more optimal result.

The increase in cost as the number of parking lots increased followed a nearly linear relationship, likely due to the way the parking lot profiles were developed (all algorithms used the same combinations of the 20 EV parking lot profiles, the base transformer limit used by the decentralized algorithm was 60 kW, and the cost per parking lot tended to range between \$35 to \$95 depending on the specific profile). Based on the trendline for each dataset generated by Microsoft Excel (see Appendix C Table 6.11), the cost per parking lot for the centralized algorithm with 500 iterations was approximately \$61.72. The centralized algorithm with 250 iterations cost approximately \$62.45, the decentralized algorithm with 500 iterations cost approximately \$62.57, and the decentralized algorithm with 250 iterations cost approximately \$63.11. These values indicate that as the number of parking lots increases, the centralized algorithm will be expected to produce a lower, more optimum cost solution for all cases.

4.4.3 Discussion: Feasibility

This section discusses the results of Test 2 as they pertain to the feasibility of the two algorithms. In this case, feasibility refers to the ability of the algorithm to generate results which did not violate any constraints, such as the transformer limit, charging rate, demand, or voltage magnitudes from equations (3.7)-(3.15), (3.18), (3.23)-(3.27), and (3.30). Figure 4.11 shows the feasibility results from Table 4.10 in graphical form.

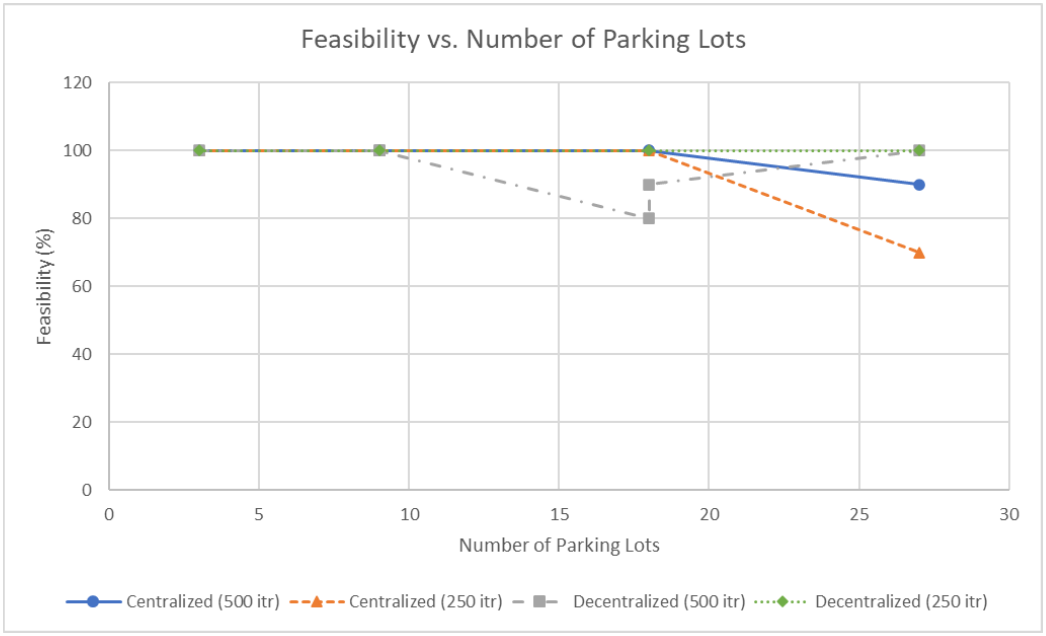


Figure 4.11 Feasibility Comparison for Test 2 Results

The results in Figure 4.11 show that generally, as the number of parking lots increased, the total amount of feasible solutions generated by the algorithm in the ten test rounds decreased. However, there are some unexpected behaviours in the results which are likely to due to the nondeterministic nature of the metaheuristic, and the PSO's susceptibility to premature convergence to a local minima [170].

The centralized algorithm was able to produce 100% feasible results for both iteration test cases, until the tests with 27 parking lots. This behaviour was expected, as due to the nondeterministic behaviour of the PSO, a higher proportion of infeasible results was expected for the 18 and 27 parking lot cases. A higher number of parking lots means that the algorithm has a higher number of inner PSOs to solve at every outer PSO iteration (number of outer PSO `parfor` loops is equal to the number of particles n_p multiplied by the number of parking lots n_{pl}), and therefore a higher chance of prematurely converged PSOs to occur. The results do indicate that the algorithm is more likely to produce a feasible solution than an infeasible one for all tested number of parking lots, but if the number of parking lots increases, it is expected that there would be a higher number of infeasible solutions and therefore a downward trend in percentage of feasible results, if the PSO parameters remain unchanged.

The test results for the centralized algorithm also demonstrate the expected behaviour in terms of inner PSO iterations: the case with half the number of iterations (250 versus 500) had a higher number of infeasible results with the 27 parking lots test case. This is most likely because with fewer iterations, the inner PSO had fewer chances to escape local minima. These results indicate that if the number of PSO iterations were increased, on both the inner and outer PSOs, there would be a higher chance of producing more feasible final solutions. As has been seen so far in this section and will also be discussed in Section 4.4.4, there is a trade-off that occurs when the PSO parameters are affected: more iterations lead to better and more feasible solutions, but also requires a longer runtime.

The decentralized algorithm was able to produce 100% feasible results except for both 18 parking lot cases. It was expected that the results would follow the same pattern as the centralized algorithm, and have more infeasible results for the 27 parking lot cases, but this did not occur. For sub-test 2.4, two single parking lot PSOs in two different test rounds (round 4, parking lot 18 and round 6, parking lot 9) were unable to converge to a feasible solution, becoming stuck in a local minima within the first 25 iterations, and therefore resulted in infeasible solutions for those test rounds.

Since the test cases with 27 parking lots required the solving of nine more PSOs than the 18 parking lot cases, it was expected that the decentralized algorithm would have behaved the same as the centralized version, and produced a few infeasible results in the 10 test rounds, particularly for the 250 iteration test case. However, due to the centralized test rounds with 250 iterations resulting in 100% feasible results for all test rounds, it is possible that the results for the 18 parking lot test cases were anomalous, and totally coincidental that the cases where the PSO failed to act in the ideal manner occurred in both sub-tests 2.4 and 2.5. It is possible that instead, the more likely behaviour of the decentralized algorithm is to produce more feasible results for higher numbers of parking lots. Further testing with more than 27 parking lots could be completed to confirm if this conclusion is correct: that the decentralized results for sub-tests 2.4 and 2.5 were exceptions to the general behaviour of the algorithm.

Comparing the feasibility results of the centralized and decentralized algorithms, it can be seen that it is more likely for the centralized algorithm to result in infeasible solutions, particularly for higher numbers of parking lots. This is most likely due to the centralized algorithm having to run 50 times more PSOs than the decentralized algorithm in a single PSO iteration, which increases the risk of one of the PSOs becoming trapped in an infeasible local minima. It is also likely due to the centralized algorithm's outer PSO behaviour producing many infeasible transformer limit results, which require additional PSO iterations to correct. Due to the way the EV parking lot profiles were created, a feasible solution for all 9 profiles can be found with a base transformer limit of 60 kW. Since the decentralized algorithm uses this limit for all parking lots, it is more likely that each parking lot's PSO can find a feasible solution. Since the centralized algorithm's outer PSO may generate infeasible solutions in each of its 50 outer PSO particles, it is possible that the outer PSO cannot converge to a feasible final solution.

4.4.4 Discussion: Runtime

This section discusses the results of Test 2 as they pertain to the runtime of the two algorithms. Table 4.13 shows the runtime results from Table 4.10 in seconds and minutes for ease of comparison. Figure 4.12 and Figure 4.13 show the runtime results from Table 4.10 and Table 4.13 in graphical form.

Table 4.12 Centralized and Decentralized Multiple Parking Lot Results: Runtime

Sub-test		Centralized		Decentralized	
		Seconds (s)	**Minutes (min.)**	**Seconds (s)**	**Minutes (min.)**
2.1	Optimum	253.78	4.23	17.43	0.29
(3 parking	Worst	307.63	5.13	21.54	0.36
lots)	Median	264.17	4.40	17.48	0.29
	Mean	267.40	4.46	17.90	0.30
	Std.	14.81		1.28	
2.2	Optimum	531.90	8.87	19.17	0.32
(9 parking	Worst	574.48	9.57	36.07	0.60
lots)	Median	537.29	8.95	19.38	0.32
	Mean	541.09	9.02	21.02	0.35
	Std.	12.60		5.29	
2.3	Optimum	525.51	8.76	17.99	0.30
(9 parking	Worst	576.56	9.61	46.13	0.77
lots)	Median	536.89	8.95	18.20	0.30
	Mean	538.76	8.98	21.92	0.37
	Std.	14.56		8.73	
2.4	Optimum	943.63	15.73	19.09	0.32
(18 parking	Worst	1018.75	16.98	37.58	0.63
lots)	Median	967.31	16.12	25.59	0.43
	Mean	960.92	16.02	25.78	0.43
	Std.	22.05		4.72	
2.5	Optimum	973.00	16.22	18.37	0.31
(18 parking	Worst	1021.54	17.03	37.58	0.63
lots)	Median	988.55	16.48	18.46	0.31
	Mean	991.39	16.52	20.41	0.34
	Std.	13.33		6.03	
2.6	Optimum	1455.90	24.27	18.98	0.32
(27 parking	Worst	1628.01	27.13	43.34	0.72
lots)	Median	1483.13	24.72	21.60	0.36
	Mean	1497.78	24.96	23.70	0.40
	Std.	49.21		7.4	

Figure 4.12 includes a yellow line along the y-axis at 900 seconds, to represent the 15-minute limit. Figure 4.13 is an enlargement of Figure 4.12, showing the average runtime of the decentralized algorithm.

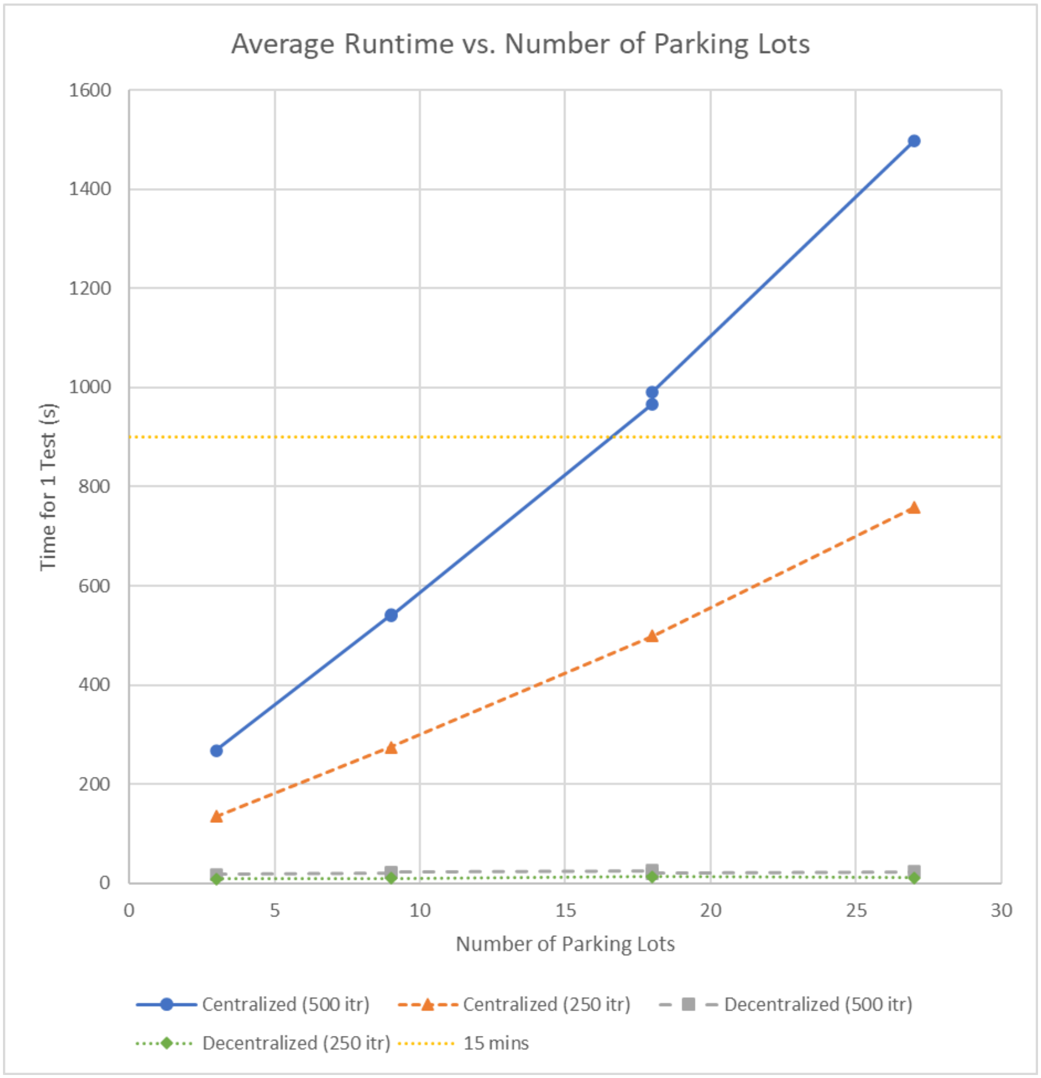


Figure 4.12 Average Runtime Comparison for Test 2 Results

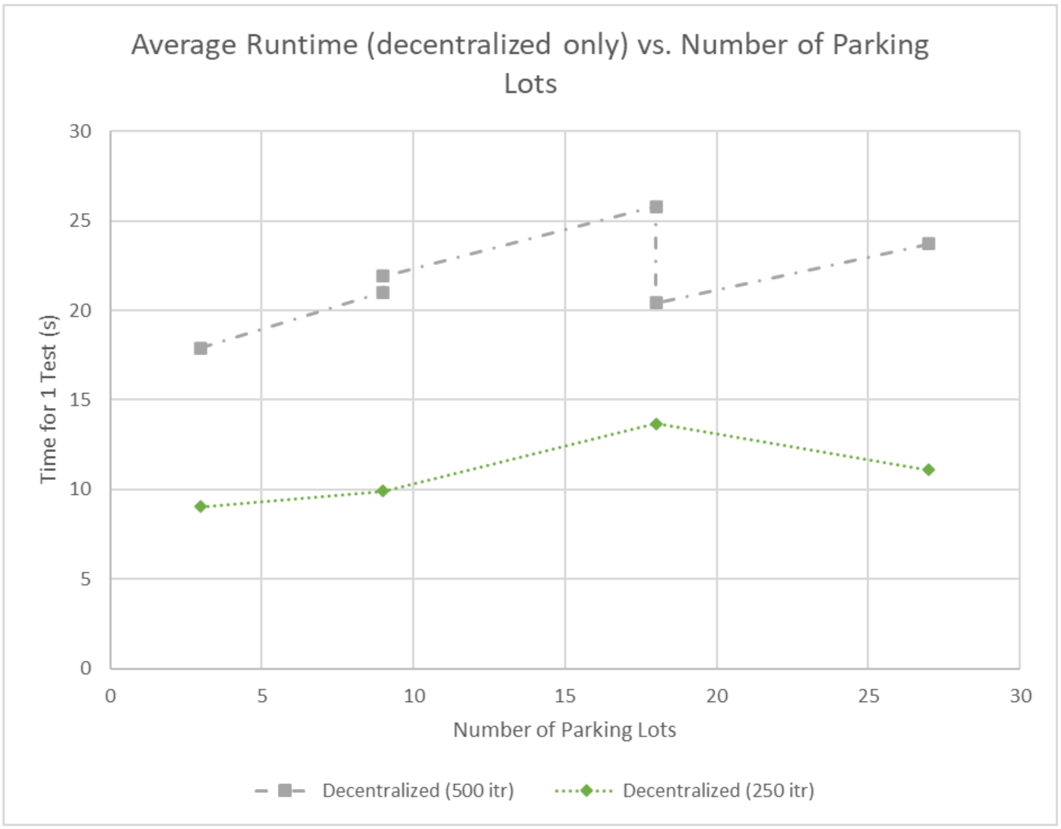


Figure 4.13 Average Runtime Comparison for Test 2 Decentralized Results

Figure 4.12 and Figure 4.13 show the runtime for each number of parking lot charge schedules to be optimized, for the two cases of different inner PSO iteration parameters. Of all the tests completed, only the centralized algorithm with 500 inner PSO iterations was unable to complete its optimizations within the 15-minute timeframe: requiring approximately 16 and 25 minutes to complete the optimizations for 18 and 27 parking lots. All centralized algorithm optimizations using 250 inner PSO iterations were completed in under 14 minutes, and all decentralized optimizations were completed in under one minute. In all cases, the algorithm test cases with half the number of iterations had a runtime approximately half the length of that with more iterations. As was seen in Sections 4.4.2 and 4.4.3, more iterations results in more optimal (lower cost) and more feasible solutions, but also requires a longer runtime.

For the centralized algorithm, the table and figures show a linear relationship between the runtime and the number of parking lot charging schedules to optimize. As was the case with feasibility as described in Section 4.4.3, this is most likely attributable to the relationship between the amount of `parfor` loops (and therefore inner PSOs) to be completed and the number of parking lots in Algorithm 2: since the number of outer PSO `parfor` loops is equal to the number of particles n_p multiplied by the number of parking lots n_{pl}, as the parking lots increase, the number of loops and PSOs to be executed also increases. In the case of 27 parking lots, the 192 workers in the Taurus Cluster are required to execute 1350 PSOs in parallel every outer PSO iteration, which is approximately seven PSOs per worker if split evenly. Knowing that it takes approximately 20 seconds for a single PSO to be optimized and approximately 160 seconds or 2.67 minutes for one iteration of the outer PSO to optimize 1350 PSOs (per Test 4 in Section 4.6), it therefore makes sense that the execution of 10 iterations of the outer PSO for this case would require approximately 26.67 minutes, as shown in Table 4.12.

Based on the trendlines produced by Microsoft Excel (see Appendix C Table 6.11) and Figure 4.12, it can be extrapolated that the centralized algorithm with 500 inner PSO iterations should be able to optimize the charge schedules for up to 15 parking lots in under 15 minutes, and the centralized algorithm with 250 inner PSO iterations should be able to optimized the schedules for up to 32 parking lots in under 15 minutes. Further testing with different combinations of inner and outer PSO iterations and numbers of parking lots could confirm the total number of parking lots which can be optimized in 15 minutes or less.

For the decentralized algorithm, the table and figures show a different relationship between the runtime and the number of parking lot charge schedules to be optimized. It shows that the algorithm with 500 inner PSO iterations requires less than 30 seconds to complete the optimizations, and the algorithm with 250 iterations requires less than 15 seconds. This relationship, as well as the variation seen in Figure 4.13 (a decrease in runtime as the number of parking lots increases in four cases, which is not present for the centralized runtime results) is most likely due to the number of `parfor` iterations, and therefore workers in the Taurus cluster, required to complete the work. In the decentralized algorithm, Algorithm 4, the number of `parfor` loops (and therefore individual PSO optimizations) is equal to the number of parking lots n_{pl}. For less than 24 parking lots, only one workstation in the Taurus Cluster was required to be activated, so there was likely less communication overhead between the 3, 9, or 18 workers of the cluster completing the work. For the 27 parking lot case, only 27 workers on the first and second workstations of the Taurus Cluster were required to complete the work. The decrease in runtime from the 18 to the 27 parking lot optimization cases may be due to communication overhead between the workers changing between tests due to other work being completed, or changes in how the workers communicated within the same or different workstations.

It would be expected that up to 192 parking lots could be optimized by both decentralized algorithm versions in similar 15 and 30 second intervals, since that is the maximum size of the Taurus Cluster. Further testing would be required to confirm the behaviour of the decentralized algorithm, but it can be theorized that up to approximately 5760 and 11520 parking lots could be optimized within 15 minutes with the decentralized algorithm, using 500 and 250 inner PSO iterations respectively.

4.4.5 Summary

The results of this test provided ample comparison between the centralized and decentralized algorithms, in order to compare their results in terms of cost, feasibility, and runtime. The results showed that in terms of cost, the decentralized model was able to function accurately and produce comparable results to the centralized model, and that most importantly, that the centralized algorithm was able to provide a more optimal result (i.e., a lower cost) than the decentralized algorithm, for all scenarios. In terms of feasibility, the likelihood that both algorithms would produce an infeasible final test result increased as the number of parking lots increased, with the centralized result slightly more likely to produce an infeasible result. In terms of runtime, the results showed that as the number of parking lots increased, the runtime for both algorithms increased, with the centralized algorithm taking a much longer time to complete a single test due to it completing 10 iterations of the outer PSO in order to optimize the transformer limits or maximum real power demand for each individual parking lot. The decentralized model's runtimes were very short for all cases, whereas the centralized model's runtimes increased dramatically and linearly as the number of parking lot charging schedules to be optimized increased.

Related to the 15-minute time limit for the optimization, it was theorized that the centralized algorithm with 500 inner PSO iterations should be able to optimize the charge schedules for up to 15 parking lots in under 15 minutes, and the centralized algorithm with 250 inner PSO iterations should be able to optimize the schedules for up to 32 parking lots in under 15 minutes. It was also theorized that the decentralized algorithm with 500 inner PSO iterations should be able to optimize the charge schedules for approximately 5760 parking lots in under 15 minutes, and the decentralized algorithm with 250 inner PSO iterations should be able to optimize the charge schedules for approximately 11520 parking lots in under 15 minutes.

In terms of weighing the optimum costs versus general feasibility versus average runtime of the test cases, the results of the test showed that these are all related to the parameters of the PSO. When the number of PSO iterations was increased from 250 to 500, the costs were lower and more optimal, and there was a greater number of feasible final solutions, but the runtime was doubled. For larger numbers of parking lots, having more inner PSO iterations caused the algorithm runtimes to exceed the 15-minute time limit. Further testing is required to determine how different inner PSO population sizes, and outer PSO population and iteration size combinations could be made to enable the centralized algorithm to meet the 15-minute runtime limit but also provide good, low cost solutions. For a distribution system where there are many different EV parking lots and locations, the potential cost savings offered by the centralized method may provide enough incentive for the operator to use larger inner and outer PSO populations and/or iterations in order to lower costs and reduce the risk of an infeasible final solution, but for those less concerned with cost, using fewer inner or outer PSO iterations may provide a solution that is "good enough" for their purposes, and within their required time limit.

4.5 Test 3: Speedup - Sequential versus Parallelized Centralized Methods for a Single Parking Lot

The purpose of this test was to determine the speedup of a parallelized version of the PSO optimization of a single parking lot, as described in Algorithm 1, by comparing the results of a sequential version of the algorithm with the results of a parallelized version (from Test 1). The aim of the test was to verify the speedup, or reduction in computation time, using the standalone multicore workstation, and also to verify that the parallelized version performed as well as the sequential version. This test demonstrated the speedup of the parallelized version of Algorithm 3, the inner PSO of Algorithm 2 and modified version of Algorithm 1, optimizing the charging of a single parking lot with a given number of EVs and their charging request profiles.

This test used the same simulation parameters as Test 1, and had the same 60 kW parking lot transformer limit.

The test description is as follows: the parallelized, centralized algorithm described in Section 3.3 above was run with the specified parameters, 100 times independently, and then the sequential algorithm was run with the same parameters, 100 times independently. The final cost, fitness, and algorithm runtime were recorded, along with graphical representations of the fitness and cost, for both versions of the algorithm. The parallelized test was run using 9 of the 20 available cores or processors on the workstation. This number was selected based on speedup tests run as part of this overall test, as described in Section 4.5.2.

4.5.1 Results

A summary of the results of this test are listed in Table 4.13, and shown in Figure 4.16 to Figure 4.15. A segment of the full results can be found in Appendix B. The results from Table 4.4 of Test 1 are repeated in Table 4.13 for ease of comparison. The speedup is calculated by dividing the time required to run one sequential test, by the time required to run one parallelized test.

Table 4.13 Parallelized and Sequential Single Parking Lot Results: Cost, Time, and Speedup – using 9 workers

	Sequential			Parallelized			
	Cost ($)	Fitness	Time required to run 1 test (s)	Cost ($)	Fitness	Time required to run 1 test (s)	Speedup (x times)
Optimum	53.78	1.0183	17.47	53.77	1.0183	14.91	1.17
Worst	62.14	1.0158	36.69	61.36	1.0160	19.25	1.91
Median	56.69	1.0173	18.10	57.09	1.0172	15.30	1.18
Mean	57.04	1.0173	18.75	57.16	1.0172	15.32	1.22
Std.	2.19	0.00	2.22	1.83	0.00	0.44	0.14

The results listed in Table 4.13 can be used to answer both aims of this test: the standalone multicore workstation was able to provide an average speedup of 1.22 times with the parallelized version of the test, and the parallelized version was able to perform as well as the sequential version, with less than a $0.80 difference between the best, worst, median, and mean results.

Figure 4.14 shows the variation in runtime over all parallelized and sequential tests, and Figure 4.15 shows a visual representation of the speedup results.

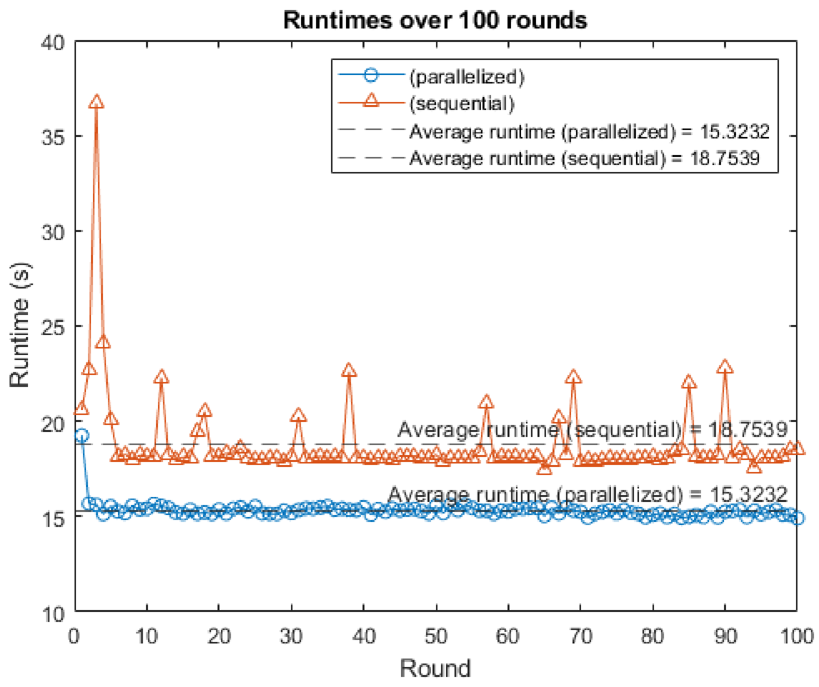


Figure 4.14 Runtime and Average Runtime for all Parallelized and Sequential Test Rounds

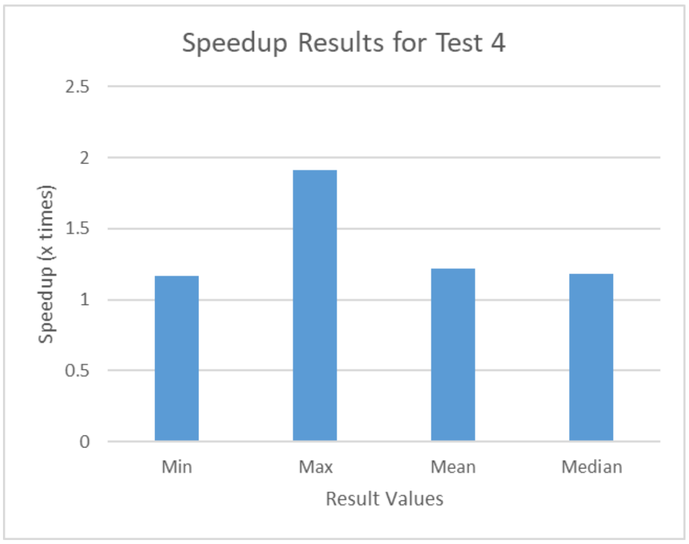
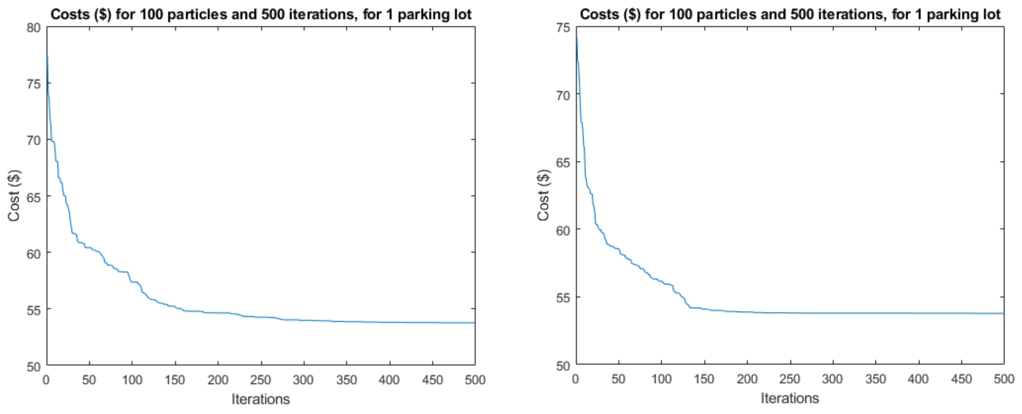


Figure 4.15 Optimum (min), Worst (max), Mean, and Median Speedup Results for Test 3

These figures illustrate that in all test cases except for the first one, the parallelized runtime was shorter than the sequential runtime by approximately two seconds (an averaged parallelized runtime of 15.30 to the minimum sequential runtime of 17.47 seconds), and therefore was always able to provide a speedup just over 1.1 times. A potential maximum speedup of almost 2 times was observed when comparing the maximum runtimes between the two algorithms (36.69 to 19.25 seconds), though the mean and median speedups provide a more accurate runtime of approximately 1.2 times since the vast majority of the test runtimes were must closer to the calculated average runtimes of 15.32 and 18.75 seconds.

It is evident in these results that the speedup is incredibly limited. Using equation (2.12) where $n = 9$ and $s_n = 1.22$, the parallelizable portion of the algorithm f is found to be approximately 0.2029, approximately two tenths of the overall algorithm. This aligns with the results in [119], where an algorithm with a parallelizable portion f of 0.2 has a maximum speedup of 1.25 times. The limited speedup may also be attributable to communication overhead of the `parfor` function, but the limited parallelizable proportion of the algorithm is likely to be the most important factor in the relatively small speedup. Per the Gustafson-Barsis Observation, is likely that a higher speedup would be seen if the number of PSO particles were increased and therefore the amount of parallel work increased by that factor, but this was not tested formally as part of this thesis.

The following figures demonstrate visually how the parallelized version performed in comparison the sequential version: Figure 4.16 shows the performance of both PSO versions, and Figure 4.17 shows the variation in final cost for both versions over all 100 test rounds.



(a) **(b)**

Figure 4.16 Change in Cost over Iterations for the Optimum Test Rounds: Parallelized – round 93 (a), and Sequential – round 188 (b)

Figure 4.16, with Figure 4.16(a) being a reproduction of Figure 4.2, shows the very similar behaviour of the objective function's cost values over the iterations, for the optimum test round of each algorithm. Both images show that the PSOs followed the expected convergence pattern, and therefore were able to produce very similar results.

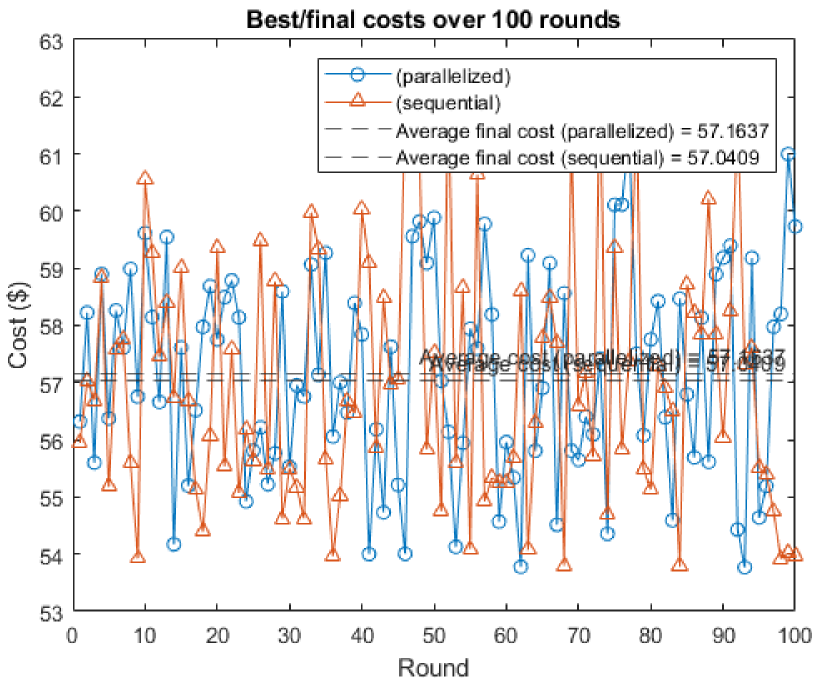


Figure 4.17 Final and Average Final Costs for all Parallelized and Sequential Test Rounds

Figure 4.17 confirms the similarity of final results between the two algorithms, and shows that the parallelized version was able to perform as well as the sequential version, if not slightly better, over the 100 test rounds.

Overall, these results show that the parallelization of the algorithm was done accurately, or correctly, able to produce results similar to and slightly better than to those of the sequential version, and that the parallelization resulted in a speedup of approximately 1.2 times.

4.5.2 Selection of Nine Worker Processes

When the standalone workstation with 20 cores was selected to run the single parking lot optimization tests, initial tests were run to determine the number of cores that should be used in the MATLAB parallel pool, to provide the best speedup before the final official 100-round simulations would be run. To determine the approximate speedup available on the multicore workstation, the same centralized parallelized and sequential algorithms were run with the specified parameters in Table 4.3 10 times independently. The test runtimes and speedup were recorded. Figure 4.18 shows the results, for the average (mean), minimum, maximum, and median speedup values for all tested number of cores.

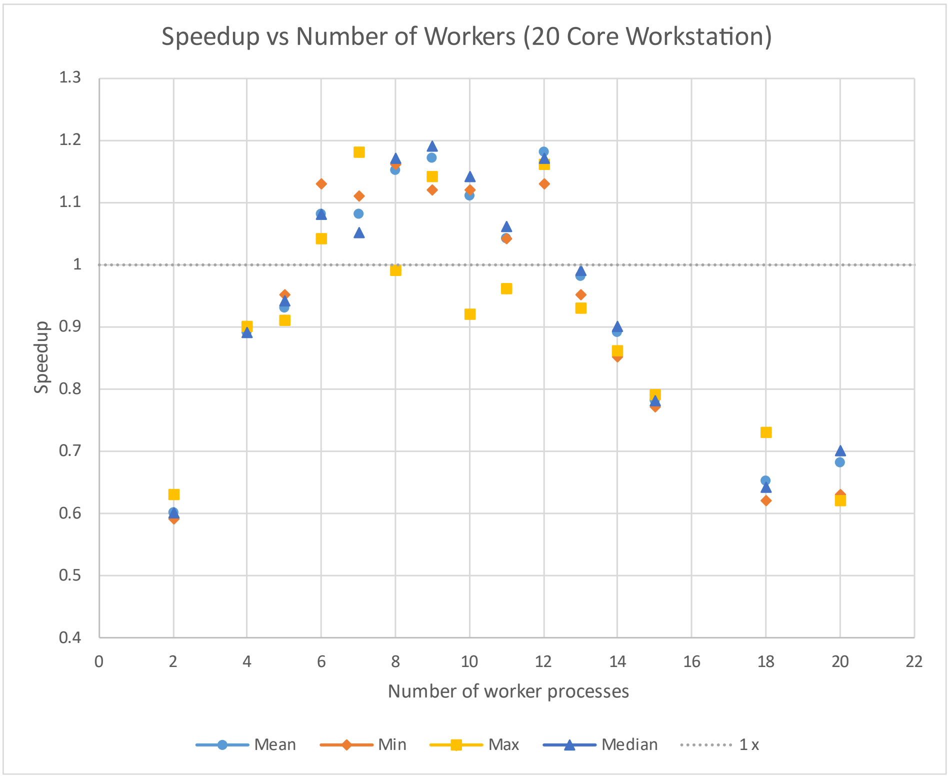


Figure 4.18 Single 20 Core Workstation Speedup Test Results

Figure 4.18 shows that a speedup of more than 1 (i.e. achieved any speedup) was found when using six to 12 cores of the available 20. With five or fewer cores, or 13 or more cores, the parallelized algorithm took as long, if not longer, than the sequential algorithm to run, likely due to the communication overhead required. The use of nine cores provided the highest potential speedup of up to 1.19 times, so that number of workers was selected for use in the test described in Sections 4.3.1 and 4.5.1.

4.6 Test 4: Speedup - Sequential versus Parallelized Centralized Methods for Multiple Parking Lots

The purpose of this test was to determine the speedup of a parallelized version of the PSO optimization of multiple parking lots, as described in Algorithm 2, by comparing the results of a sequential version of the algorithm with the results of a parallelized version. The aim of the test was to verify the speedup, or reduction in computation time, using the 192-worker HPC cluster. This test demonstrated the speedup of the parallelized Algorithm 2, optimizing the charging of multiple parking lots with a given number of EVs and their charging request profiles.

This test used most of the same simulation parameters and tests results (for the parallelized version) as Test 2. However, due to the long algorithm runtimes required by the sequential versions, this test used a modified set of parameters and only one scenario for each number of parking lots, as listed in Table 4.14.

Table 4.14 Simulation Parameters – Test 4

Parameter	Value			
Sub-test no.	**4.1**	**4.2**	**4.3**	**4.4**
Number of Parking Lots	3	9	18	27
Number of EVs	60	180	360	540
Number of Particles (Outer PSO)	50	50	50	50
Number of Iterations (Outer PSO)	5	5	5	5
Number of Particles (Inner PSO)	100	100	100	100
Number of Iterations (Inner PSO)	500	500	500	500
Distribution System Test Case	18-bus	33-bus	69-bus	141-bus

As described previously, running 10 test trials of the full version of these tests requires an exceptionally long and impractical amount of time (for example, it would require over 750 hours or 31 days to run 10 trials of the full centralized, sequential 27 parking lot model). Therefore, for this test, the individual item being tested was a single test's outer PSO iteration time, and the final cost and fitness outcomes were ignored. The iteration time was therefore trialed five times in one test, to keep the test runtimes under

48 hours. To determine the approximate or estimated time required to run one full 10-iteration version of the test, the average or mean iteration runtime was multiplied by 10.

The test description is as follows: the parallelized, centralized algorithm described in Section 3.3 was run for each sub-test with the specified parameters, one time independently, and then the sequential algorithm was run with the same parameters, one time independently. The five iteration runtimes were recorded for both versions of the algorithm, and their speedup results compared. In order to determine the speedup as if a full 10 iteration version of the parallelized and sequential algorithms were run, the average single iteration times were multiplied by 10 and the speedup results for this case were also recorded.

4.6.1 Results

A summary of the results of this test are listed in Table 4.15 and Figure 4.19. The speedup is calculated by dividing the time required to run one sequential test by the time required to run one parallelized test, per equation (2.10).

Table 4.15 Parallelized and Sequential Multiple Parking Lot Results: Time and Speedup

Sub -test		**Sequential**		**Parallelized**			
		Time required to run 1 iteration (s)	**Time required to run 1 test* (s)**	**Time required to run 1 iteration (s)**	**Time required to run 1 test* (s)**	**Speedup for 1 iteration (x times)**	**Speedup for 1 test* (x times)**
4.1	Optimum	2650.69		25.34		104.59	
(3	Worst	2768.03		75.67		36.01	
park	Median	2664.67		25.69		103.72	
ing	Mean	2685.64	26856.4	36.32	363.20	73.95	73.95
lots)	Std.	48.31		22.7		29.18	
4.2	Optimum	7079.96		52.71		134.31	
(9	Worst	7499.52		107.58		69.71	
park	Median	7247.30		55.09		131.55	
ing	Mean	7278.86	72788.60	65.67	656.70	110.84	110.84
lots)	Std.	183.90		23.54		30.96	
4.3	Optimum	14424.00		90.10		160.09	
(18	Worst	14553.74		149.83		97.13	
park	Median	14461.90		92.28		156.71	
ing	Mean	14485.84	144858.40	103.88	1038.80	139.44	139.44
lots)	Std.	57.46		25.81		27.53	
4.4	Optimum	20758.22		138.92		149.42	
(27	Worst	21254.14		205.73		103.31	
park	Median	20945.81		147.25		142.25	
ing	Mean	20933.40	209334.0	157.34	1573.40	133.05	133.05
lots)	Std.	202.47		27.34		19.33	

* estimated times, based on multiplication of average runtime of one iteration by 10 to make 10 iterations (1 full test)

The results in Table 4.15 are visually illustrated in Figure 4.19. They show that in all test cases, there was a runtime speedup of at least 27 times, due to the variations in runtime across all tests. The lowest potential speedups for all cases were seen when comparing the maximum runtimes for the algorithm versions, and the highest seen when comparing the minimum runtimes observed across the five test iterations. The estimated times for the parallelized tests are in-line with the real results seen in Test 2 (runtimes ranging from 253.78 to 307.63 seconds for three parking lots, 525.51 to 576.56 seconds for nine parking lots, 943.63 to 1018.75 seconds for 18 parking lots, and 1455.90 to 1628.01 seconds for 27 parking lots), showing that these can be considered good estimated total runtimes.

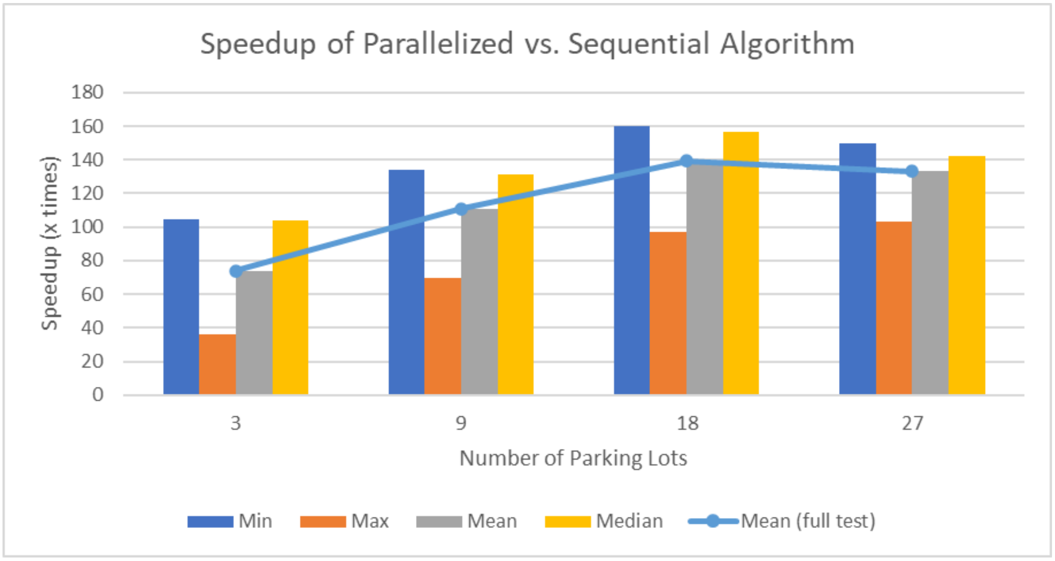


Figure 4.19 Single Iteration Optimum (min), Worst (max), Mean, and Median Speedup of the Parallelized versus the Sequential Algorithm, and estimated Mean Speedup for a Full Test, for Various Numbers of Parking Lots

The line in Figure 4.19 shows the general behaviour of the average speedup value as the number of parking lots, and therefore size of the problem, increases. The speedup appears to increase in a mostly linear fashion from three to 18 parking lots, but plateaus or slightly decreases from 18 to 27 parking lots. This is mostly consistent with the expected linear increase in speedup as the amount of work increases. Further testing with different numbers of parking lots above and below 27 may produce a clearer relationship between the number of parking lots and algorithm speedup.

Using equation (2.12) and the average speedup for each number of parking lots, the parallelizable portions of the algorithm f are found to be approximately 1, or 100% (0.9916, 0.9962, 0.9980, 0.9977). Based on the speedup rules from Amdahl's Law, this parallelizable proportion of nearly 100% implies that the addition of more processors should always provide a speedup with nearly no upper bound, and from and the Gustafson-Barsis Observation, the speedup should be proportional to the number of processors n. Further testing with different numbers of processors may produce a clearer relationship between the number of processors, amount of work, and algorithm speedup.

The results of Test 3 showed that the parallelization of the fitness function evaluation of the single parking lot PSO, i.e. the parallel evaluation of the 100 inner PSO particles, reflected an algorithm with only a 20% parallelizable proportion. These test results showed that the parallelization of the fitness function evaluation of the outer PSO, i.e., the parallelization of 50 particles multiplied by n_{pl} number of parking lots full 100 particle, 500 iteration PSO evaluation, reflected an algorithm with a nearly 100%

parallelizable proportion. This shows that the algorithm with the parallelized outer PSO (Algorithm 2) was more effectively parallelized in terms of it resulting in a much greater speedup, however with the chosen inner and outer PSO parameters, the minimum runtime for a test remains 20 seconds (for one parking lot optimized with one outer PSO iteration), and that time cannot be further reduced without changes to the parallelization method of the algorithm since `parfor` loops cannot be nested inside one another [159].

Overall, the results of Test 4 show using all 192 worker processors in a HPC cluster can result in a speedup of approximately 74 to 139 times for the optimization of EV charge scheduling for three to 27 parking lots.

4.7 Validation

To validate this thesis, the centralized, parallelized algorithms adapted (Algorithm 1 and Algorithm 3) and developed (Algorithm 2) to optimize one or more parking lot charging schedules were compared against: the original results of the single parking lot optimization method developed in [40] to confirm that individual parking lots were being optimized properly, compared against the developed decentralized algorithm (Algorithm 4) to compare optimization results as the problem size increased, and compared against the sequential version of the algorithms to compare runtime results.

The following sections compare the results obtained in Sections 4.3, 4.4, 4.5, and 4.6 with the success criteria established in Section 4.2, to verify the accuracy of the implementation and the accuracy of the parallelization of the model algorithms.

4.7.1 Accuracy of Implementation: Centralized Single and Multi-Parking Lot Optimization

The results used to determine the accuracy of the implementation of the centralized model, from success criteria #1 and #3, come from Test 1 (Section 4.3), Test 2 (Section 4.4), and Test 3 (Section 4.5).

4.7.1.1 *Single Parking Lot Optimization*

As was discussed in Section 4.3.1, the Test 1 results showed that the implementation of the centralized multiple parking lot optimization model, when modified to run for one parking lot, was able to perform accurately, as intended. The average, lowest, and highest electricity costs produced by the model were within the range of \$53.77 to \$61.36, well within the range of \$53.77 to \$73.69 from [40]. The \$61.36 maximum result is slightly higher than the \$59.45 “original PSO” result, but since no standard deviation or other results for the same 20 EV, 60 kW transformer limit result were presented in the paper, it is unsure if that result was for the best original PSO result, or the average result from an unknown number of trials.

4.7.1.2 *Multiple Parking Lot Optimization*

The Test 2 results show that the centralized multiple parking lot optimization model was implemented accurately, and could continue to perform as intended as the problem size increased. The average cost output for each problem size or scenario scaled linearly with the number of parking lots (see Table 6.11 in Appendix C), and were below and within the desired $\$\, n_{pl} \times \$\, 5$ margin value. This is shown in Table 4.16.

Table 4.16 Cost Comparison for Expected and Test 2 Centralized Results ($)

Sub-test	n_{pl}	$n_{pl} \times 5$	Average Cost	$n_{pl} \times 64$	Difference
2.1	3	15	184.31	192	7.69
2.2	9	45	531.65	576	44.35
2.3	9	45	532.82	576	43.18
2.4	18	90	1094.43	1152	57.57
2.5	18	90	1091.42	1152	60.58
2.6	27	135	1658.94	1728	60.17

The average cost results are from Table 4.11. The fifth column shows the result of the expected approximate cost output equation from Section 4.2.2, $\$\, n_{pl} \times \$\, 64$, and the third column shows the result of the value margin equation $\$\, n_{pl} \times \$\, 5$. The table shows that in all cases, the average cost was close to or below the expected value, and never exceeded the margin value.

For success criteria #1, determining which model is able to provide the more optimal result, the cost results for the centralized and decentralized models were compared in Section 4.4.2, and clearly showed that in all cases, the centralized algorithm produced a lower cost therefore more optimal solution.

4.7.2 Accuracy of Parallelization and Speedup: Parallel Implementation of the Centralized Single and Multi-Parking Lot Algorithms

The results used to determine the accuracy, or correctness, of the parallelization of the centralized model, from success criteria #2 and #3, come from Test 1 (Section 4.3), Test 2 (Section 4.4), Test 3 (Section 4.5), and Test 4 (Section 4.6).

4.7.2.1 *Single Parking Lot Optimization*

The results of Test 3 show that the sequential and parallelized version of the centralized multiple parking lot optimization model, when modified to run for one parking lot, provided results within \$0.01-0.78 of the minimum and maximum results. These very similar results show that the model was parallelized accurately, in terms of implementation.

The speedup results of Test 3 also show that the model was parallelized accurately in terms of speedup: all speedup values were greater than 1 times, with an average speedup of 1.22 times. Though this was not a large decrease in computational runtime it does demonstrate that the model was parallelized accurately.

4.7.2.2 *Multiple Parking Lot Optimization*

The results of Test 2 show that the centralized multiple parking lot optimization model was parallelized accurately, in terms of implementation, being able to produce the accurate expected cost results (see Section 4.7.1.2),

The results of Test 4 showed that the model was parallelized accurately, in terms of speedup. All speedup values for all tested scenarios were greater than 1, with average speedups ranging from 74 to 139 times.

The problem sizes (number of parking lots with 20 EVs each) which were able to complete their optimizations within the 15 minute time limit depended on the PSO iteration parameters chosen for the test. The centralized algorithm using 500 inner PSO iterations was able to compute the optimization for a maximum of 9 parking lots in under 15 minutes, though it may be possible to optimize up to 15 parking lots. The centralized algorithm using 250 inner PSO iterations was able to optimize up to 27 parking lots in under 15 minutes, though it may be able to optimize up to 32 parking lots.

4.7.3 Accuracy of Implementation: Parallel Implementation of the Decentralized Multi-Parking Lot Algorithm

The results used to determine the accuracy of the implementation (and parallelization) of the decentralized model, from success criteria #1, Test 2 (Section 4.4) and Test 3 (Section 4.5).

The Test 2 results show that the decentralized multiple parking lot optimization model was implemented accurately, and could continue to perform as expected as the problem size increased. The average cost output for each problem size or scenario scaled linearly with the number of parking lots (see Table 6.11 in Appendix C), and were below or within the desired $\$\, n_{pl} \times \$\, 5$ margin value. This is shown in Table 4.17.

Table 4.17 Cost Comparison for Expected and Test 2 Decentralized Results ($)

Sub-test	n_{pl}	$n_{pl} \times 5$	Average Cost	$n_{pl} \times 64$	Difference
2.1	3	15	193.81	192	1.81
2.2	9	45	566.08	576	9.92
2.3	9	45	563.94	576	12.06
2.4	18	90	1126.94	1152	25.06
2.5	18	90	1126.61	1152	25.39
2.6	27	135	1695.77	1728	32.23

The average cost results are from Table 4.11. The fifth column shows the result of the expected approximate cost output equation from Section 4.2.2, $\$\,n_{pl} \times \$\,64$, and the third column shows the result of the value margin equation $\$\,n_{pl} \times \$\,5$. The table shows that in all cases, the average cost was close to or below the expected value, and never exceeded the margin value.

Since the single parking lot optimization tests used a different computer and testing configuration than those of Test 2, the runtime results do not align perfectly, so there cannot be true comparison between the minimum, maximum, and average runtime for the decentralized model from Table 4.12 and the minimum, maximum, and average runtime for the sequential single parking lot optimization model from Table 4.13. However, as was discussed in Section 4.4.4, all decentralized model average test runtimes were within a 17.9-23.7 second range, indicating that each decentralized model test runtime was approximately equal to the runtime required to sequentially optimize a single parking lot.

4.8 Summary

This chapter described the results obtained from the different tests run in order to validate the performance of the developed methods for different scenarios and numbers of parking lots. The results showed how the models developed support the aim of this thesis, as the centralized method was able to provide more optimal solutions compared to the decentralised method for scenarios with more than one parking lot, and that parallel computing enabled the optimization of scenarios with up to 27 parking lots to be computed in less than 15 minutes, depending on the chosen PSO parameters. The centralized model with 500 inner PSO iterations was only able to compute up to 9 parking lots, but could compute up to the maximum test amount of 27 parking lots within 15 minutes when using 250 inner PSO iterations. Section 4.1 discussed the experimental setup of the multicore and distributed computing cluster testing systems. The three success criteria and their comparison metrics were explained in Section 4.2. Sections 4.3, 4.4, 4.5, and 4.6 showed the results for all the completed tests, and Section 4.7 explained how the results met the success criteria for this thesis.

Section 5

Contributions and Future Work

This chapter describes the contributions of this thesis research, and provides a number of recommendations for future work to build on the research as well as areas to improve and expand on the models designed and presented in this thesis.

5.1 Contributions

As mentioned in Section 1.5, this research provided four main contributions to the field of EV charge scheduling optimization, with a focus on real-time centralized optimization. The contributions from this thesis research are as follows:

1. Development of a complete centralized solution to the problem of EV charge scheduling optimization in multiple parking lots. A two-level PSO algorithm was designed and implemented, building on the single parking lot optimization model developed by Wu et al. in 2018 [40]. This algorithm included the definition of two fitness functions, and included the verification of power flow voltage constraints in the system. This solution enabled the optimization of charging schedules for individual EVs within parking lots, coordinating the transformer limits or maximum charging capacity between all parking lots in a system to minimize costs while remaining within system power flow constraints.

2. Verification that centralized optimization provides a more optimal solution than decentralized optimization. A decentralized version of the centralized algorithm was created for the optimization problem, and direct comparisons for cost, feasibility, and algorithm runtime, were made for ten different scenarios. In all cases, the centralized algorithm was able to provide a more optimal result (i.e., a lower cost) than the decentralized algorithm. These results support the assertions made in literature that the centralized approach provides a more optimal result [10], [34], [36], [70]. The results presented in this thesis also support assertions made about the decentralized approach: that while it may not be able to provide a more optimal solution, it is more computationally efficient. These results and their comparison contribute to the body of literature which directly compare centralized and decentralized EV charge scheduling solutions together, providing more examples of benefits and drawbacks which may support the use of one method over another.

3. Provision of a method to parallelize the two-level PSO centralized optimization model. The two-level PSO centralized multiple-parking lot optimization model was parallelized with the SPMD method, using the MATLAB `parfor` function to evaluate the fitness of and perform the optimization of a specified amount of PSO candidate EV parking lot charging schedules in parallel. This parallelization reduced the computational burden of the centralized optimization model, enabling all tested scenarios of three to 27 parking lots to be completed in a much shorter amount of time, providing speedups of approximately 74 to 139 times using the experimental setup.

4. Demonstration of the scalability of the solution. The parallelized two-level PSO centralized optimization model was tested against its sequential version, and the results verified that using parallel computing on HPC systems can allow for real-time optimization of the EV charge scheduling problem for a variety of problem sizes. Depending on the PSO parameters used, the parallelized model was able to compute the EV charging schedules for up to 180 EVs in 9 parking lots (500 inner PSO iterations) or 540 EVs in 27 parking lots (250 inner PSO iterations) in under 15 minutes. Further experimentation could verify the exact number of parking lots which could be optimized with this algorithm with a specific set of PSO parameters in 15 minutes or less using the same experimental setup.

5.2 Future Work and Recommendations

The research conducted for this thesis can provide a foundation for future work in centralized EV charge scheduling optimization and algorithm parallelization to enable real-time optimization.

Firstly, the two-level PSO algorithm can be integrated with future optimization problems which consider multiple objectives, multiple EVs, or multiple parking lots, when considering a centralized optimization method. As well, the two-level model could be adapted to include other metaheuristic or optimization methods as the outer or inner method, to tailor the optimization to the specific problem and combat some of the less desirable aspects of PSO like its tendency to premature convergence. A combination of PSO and GA, hybrid methods, or the use of techniques like islanded PSO could be trialed to determine if a method could be found which can offer more consistent feasible results for a larger number of parking lots. Further experimentation with the centralized multiple parking lot optimization model developed in this thesis could be conducted in order to determine which different inner and outer PSO population and iteration size combinations could enable the algorithm to meet the 15 minute runtime limit while also providing good, low cost, and feasible solutions.

Secondly, the use of parallelization with other approaches should be investigated. The parallelization of the two-level PSO algorithm within this thesis was successful, since it was able to greatly reduce the runtimes and provide large speedups. The use of parallelization to reduce the computational burden and improve algorithm runtimes should be investigated for use with other approaches to the problem of EV charge scheduling, especially for those which have larger problem sizes (using the centralized method, optimizing multiple parking lots, etc.). As well, investigation on how to use GPU-based parallelization should be considered, as it may allow for even greater reductions in algorithm runtime as the problem size increases.

There are also a number of recommendations for future work which may build on this thesis, to improve the optimization process, increase the realism of the scenarios and model, and overall improve the quality of solutions produced by the centralized, two-level PSO multiple-parking lot EV charge scheduling optimization model.

Firstly, to increase the realism of the scenarios to better reflect current power systems, the distribution system can have distributed generation and energy storage systems integrated into the scenarios. As well, with the expected implementation of V2G technologies in the future, the modification of the model to support charging and discharging of EVs and their behaviour as mobile energy storage systems would allow for a wider variety of scenarios and EV charging behaviours to be investigated.

Another way to increase the realism of the EV charging would be to generate more distinct EV parking lots and EV charging profiles based on different types of EVs with different charging demands. These parking lots could have differing numbers of charging ports, and include Level 2 and 3 charging ports. The EV charging profiles could be extended to include more than 10 time intervals to better model behaviour over a full day or a week. Lots could be identified as serving commercial, industrial, workplace, or other areas, and have parking profiles generated from real EV charging data to better model the behaviour and impact on different nodes of a distribution system.

The modification of the electricity costs to reflect a real city's utility and power demand would also increase the realism of the model, and better support the assertion that EV charging schedule optimization can be used to "control charging activity at charging stations to minimize unexpected spikes in peak load demand" [15]. If the price reflects the demand, i.e. a higher price reflects a period where demand needs to be shaved or minimized, then the model developed in this thesis can reduce the peak power demand by optimizing the schedule to reduce charging during those periods. However, this claim could be better supported by expanding the simulation data and model to use a longer timeframe and demand and pricing information from a city where the demand is known for each time, and the costs reflect real time pricing. Using this expanded information, it could be demonstrated whether the model can be truly used to minimize spikes in load demand on the system.

5.3 Summary

This chapter described the four main contributions of this thesis research, which included the development of a complete centralized solution to the EV charging schedule optimization problem, verification that it provided a more optimal solution than a decentralized solution, the provision of a method to parallelize it, and demonstrations of its scalability. It also described a number of number of recommendations for future work to build on the research, as well as areas to improve and expand on the models designed and presented in this thesis in order to make the scenarios and resulting solutions more reflective of realistic scenarios.

Section 6

Conclusion

The research conducted for this thesis has successfully produced an optimization model for the EV charge scheduling problem, which used a parallelized two-level PSO algorithm to optimize the charging schedules of EVs in multiple parking lots. The model was validated using four main tests with over fourteen individual sub-tests and four distribution system scenarios, and its results compared against its sequential version and a decentralized model in order to prove the research hypothesis. The results showed that a centralized EV charge scheduling optimization model for EVs in multiple parking lots will find a more optimized solution to the charge scheduling problem, compared to a decentralized optimization model, and that parallelization of the calculations on a HPC system like a multicore workstation or HPC cluster will allow for real-time optimization.

As the number of EVs in operation around the world continues to rise, the management and optimization of EV charge scheduling is becoming even more necessary in order to minimize the burden on existing power infrastructure while reducing costs for utilities and users. This work contributes to the body of literature providing potential solutions to the problem of optimizing EV charging schedules in multiple parking lots. It provides a method of parallelizing a two-level metaheuristic optimization algorithm to enable centralized optimization of the problem, which is both scalable and computationally viable in a practical amount of time, and considers power flow limitations to respect grid constraints. With further research and improvements to the model to include more parameters to simulate even more realistic scenarios, it has the potential to provide solutions to problems that were not previously investigated with a centralized model due to computational limits.

References

[1] "Standard deviation - MATLAB std," MathWorks Help Center. Accessed: Nov. 27, 2023. [Online]. Available: https://www.mathworks.com/help/matlab/ref/std.html

[2] "World Energy Outlook 2022," International Energy Agency (IEA), Paris, France, Nov. 2022. Accessed: Mar. 01, 2023. [Online]. Available: https://www.iea.org/reports/world-energy-outlook-2022

[3] "Global Energy Review 2021," International Energy Agency (IEA), Paris, France, 2021. Accessed: Mar. 01, 2023. [Online]. Available: https://www.iea.org/reports/global-energy-review-2021

[4] J. S. Bartlett and B. Preston, "Automakers Are Adding Electric Vehicles to Lineups. Here's What's Coming.," Consumer Reports. Accessed: Jun. 28, 2023. [Online]. Available: https://www.consumerreports.org/cars/hybrids-evs/why-electric-cars-may-soon-flood-the-us-market-a9006292675/

[5] "Global EV Outlook 2023," International Energy Agency (IEA), Paris, France, Apr. 2023. Accessed: Jun. 28, 2023. [Online]. Available: https://www.iea.org/reports/global-ev-outlook-2023

[6] A. Dubey and S. Santoso, "Electric Vehicle Charging on Residential Distribution Systems: Impacts and Mitigations," *IEEE Access*, vol. 3, pp. 1871–1893, 2015, doi: 10.1109/ACCESS.2015.2476996.

[7] A. K. Vamsi Krishna Reddy and K. Venkata Lakshmi Narayana, "Meta-heuristics optimization in electric vehicles -an extensive review," *Renew. Sustain. Energy Rev.*, vol. 160, p. 112285, May 2022, doi: 10.1016/j.rser.2022.112285.

[8] "Market Snapshot: Record-high electric vehicle sales in Canada," Canada Energy Regulator. Accessed: Jul. 14, 2023. [Online]. Available: https://www.cer-rec.gc.ca/en/data-analysis/energy-markets/market-snapshots/2022/market-snapshot-record-high-electric-vehicle-sales-canada.html

[9] J. R. Aguero, E. Takayesu, D. Novosel, and R. Masiello, "Modernizing the Grid: Challenges and Opportunities for a Sustainable Future," *IEEE Power Energy Mag.*, vol. 15, no. 3, pp. 74–83, May 2017, doi: 10.1109/MPE.2017.2660819.

[10] L. Jian, Y. Zheng, and Z. Shao, "High efficient valley-filling strategy for centralized coordinated charging of large-scale electric vehicles," *Appl. Energy*, vol. 186, pp. 46–55, Jan. 2017, doi: 10.1016/j.apenergy.2016.10.117.

[11] J. A. Momoh, *Electric Power System Applications of Optimization*, Second Edition. Boca Raton, FL: CRC Press, 2009.

[12] B. Khan and P. Singh, "Selecting a Meta-Heuristic Technique for Smart Micro-Grid Optimization Problem: A Comprehensive Analysis," *IEEE Access*, vol. 5, pp. 13951–13977, 2017, doi: 10.1109/ACCESS.2017.2728683.

[13] S. A.-H. Soliman and A.-A. H. Mantawy, *Modern Optimization Techniques with Applications in Electric Power Systems*. in Energy Systems. New York, New York: Springer, 2012.

[14] H. Abdi, S. D. Beigvand, and M. L. Scala, “A review of optimal power flow studies applied to smart grids and microgrids,” *Renew. Sustain. Energy Rev.*, vol. 71, pp. 742–766, May 2017, doi: 10.1016/j.rser.2016.12.102.

[15] A. K. Kalakanti and S. Rao, “Computational Challenges and Approaches for Electric Vehicles,” *ACM Comput. Surv.*, Jan. 2023, doi: 10.1145/3582076.

[16] W.-L. Liu, Y.-J. Gong, W.-N. Chen, Z. Liu, H. Wang, and J. Zhang, “Coordinated Charging Scheduling of Electric Vehicles: A Mixed-Variable Differential Evolution Approach,” *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 12, pp. 5094–5109, Dec. 2020, doi: 10.1109/TITS.2019.2948596.

[17] “Heuristics and meta-heuristics,” EURODECISION. Accessed: Feb. 22, 2023. [Online]. Available: https://www.eurodecision.eu/algorithms/operations-research-optimization/heuristics-and-meta-heuristics

[18] V. Roberge, M. Tarbouchi, and F. Okou, “Optimal power flow based on parallel metaheuristics for graphics processing units,” *Electr. Power Syst. Res.*, vol. 140, pp. 344–353, Nov. 2016, doi: 10.1016/j.epsr.2016.06.006.

[19] O. Ivanov, B.-C. Neagu, N.-C. Toma, G. Grigoras, P.-D. Ghilan, and M. Gavrilas, “Metaheuristic Approaches for Distributed Generation Placement Optimization in Electrical Grids : A comparison between PSO, Tiki-Taka and Archimedes Optimzation Algorithms,” in *2022 International Conference and Exposition on Electrical And Power Engineering (EPE)*, Oct. 2022, pp. 208–212. doi: 10.1109/EPE56121.2022.9959077.

[20] K. E. Adetunji, I. W. Hofsajer, A. M. Abu-Mahfouz, and L. Cheng, “A Review of Metaheuristic Techniques for Optimal Integration of Electrical Units in Distribution Networks,” *IEEE Access*, vol. 9, pp. 5046–5068, 2021, doi: 10.1109/ACCESS.2020.3048438.

[21] V. Roberge and M. Tarbouchi, “Hybrid Method Based on Metaheuristics and Interior Point for Optimal Power Flow,” in *2021 Fifth International Conference On Intelligent Computing in Data Sciences (ICDS)*, Oct. 2021, pp. 1–8. doi: 10.1109/ICDS53782.2021.9626722.

[22] M. Papadimitrakis, N. Giamarelos, M. Stogiannos, E. N. Zois, N. A.-I. Livanos, and A. Alexandridis, “Metaheuristic search in smart grid: A review with emphasis on planning, scheduling and power flow optimization applications,” *Renew. Sustain. Energy Rev.*, vol. 145, p. 111072, Jul. 2021, doi: 10.1016/j.rser.2021.111072.

[23] D. Rodriguez, D. Gomez, D. Alvarez, and S. Rivera, “A Review of Parallel Heterogeneous Computing Algorithms in Power Systems,” *Algorithms*, vol. 14, no. 10, p. 275, Oct. 2021, doi: 10.3390/a14100275.

[24] J. Soares, M. A. Fotouhi Ghazvini, Z. Vale, and P. B. de Moura Oliveira, “A multi-objective model for the day-ahead energy resource scheduling of a smart grid with high penetration of sensitive loads,” *Appl. Energy*, vol. 162, pp. 1074–1088, Jan. 2016, doi: 10.1016/j.apenergy.2015.10.181.

[25] J. Soares, Z. Vale, B. Canizes, and H. Morais, “Multi-objective parallel particle swarm optimization for day-ahead Vehicle-to-Grid scheduling,” in *2013 IEEE Computational Intelligence Applications in Smart Grid (CIASG)*, Apr. 2013, pp. 138–145. doi: 10.1109/CIASG.2013.6611510.

[26] B. Wang, P. Dehghanian, and D. Zhao, “Chance-Constrained Energy Management System for Power Grids With High Proliferation of Renewables and Electric Vehicles,” *IEEE Trans. Smart Grid,* vol. 11, no. 3, pp. 2324–2336, May 2020, doi: 10.1109/TSG.2019.2951797.

[27] H. Bian, Z. Guo, C. Zhou, and S. Peng, “Multi-time scale electric vehicle charging load forecasting considering constant current charging and parallel computing,” *Energy Rep.*, vol. 8, pp. 722–732, Nov. 2022, doi: 10.1016/j.egyr.2022.08.034.

[28] V. Schwarzer and R. Ghorbani, “New opportunities for large-scale design optimization of electric vehicles using GPU technology,” in *2011 IEEE Vehicle Power and Propulsion Conference,* Sep. 2011, pp. 1–6. doi: 10.1109/VPPC.2011.6043051.

[29] A. Di Martino, S. M. Miraftabzadeh, and M. Longo, “Strategies for the Modelisation of Electric Vehicle Energy Consumption: A Review.,” *Energ. 19961073,* vol. 15, no. 21, p. 8115, Nov. 2022, doi: 10.3390/en15218115.

[30] A. Schambers, M. Eavis-O’Quinn, V. Roberge, and M. Tarbouchi, “Route planning for electric vehicle efficiency using the Bellman-Ford algorithm on an embedded GPU,” in *2018 4th International Conference on Optimization and Applications (ICOA),* Apr. 2018, pp. 1–6. doi: 10.1109/ICOA.2018.8370584.

[31] D. Liu, P. Zeng, S. Cui, and C. Song, “Deep Reinforcement Learning for Charging Scheduling of Electric Vehicles Considering Distribution Network Voltage Stability.,” *Sens. 14248220,* vol. 23, no. 3, p. 1618, Feb. 2023, doi: 10.3390/s23031618.

[32] M. Amjad, A. Ahmad, M. H. Rehmani, and T. Umer, “A review of EVs charging: From the perspective of energy optimization, optimization approaches, and charging techniques,” *Transp. Res. Part Transp. Environ.*, vol. 62, pp. 386–417, Jul. 2018, doi: 10.1016/j.trd.2018.03.006.

[33] “2022 Annual Planning Outlook,” Independent Electricity System Operator (IESO), Toronto, Dec. 2022. [Online]. Available: https://www.ieso.ca/en/Sector-Participants/Planning-and-Forecasting/Annual-Planning-Outlook

[34] N. I. Nimalsiri, C. P. Mediwaththe, E. L. Ratnam, M. Shaw, D. B. Smith, and S. K. Halgamuge, “A Survey of Algorithms for Distributed Charging Control of Electric Vehicles in Smart Grid,” *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4497–4515, Nov. 2020, doi: 10.1109/TITS.2019.2943620.

[35] A. Amin *et al.*, “A Review of Optimal Charging Strategy for Electric Vehicles under Dynamic Pricing Schemes in the Distribution Charging Network,” *Sustainability*, vol. 12, no. 23, Art. no. 23, Jan. 2020, doi: 10.3390/su122310160.

[36] M. Ş. Kuran, A. Carneiro Viana, L. Iannone, D. Kofman, G. Mermoud, and J. P. Vasseur, “A Smart Parking Lot Management System for Scheduling the Recharging of Electric Vehicles,” *IEEE Trans. Smart Grid*, vol. 6, no. 6, pp. 2942–2953, Nov. 2015, doi: 10.1109/TSG.2015.2403287.

[37] M. Jawad *et al.*, “A Cost-Effective Electric Vehicle Intelligent Charge Scheduling Method for Commercial Smart Parking Lots Using a Simplified Convex Relaxation Technique,” *Sensors*, vol. 20, no. 17, p. 4842, Aug. 2020, doi: 10.3390/s20174842.

[38] Q. Kang, J. Wang, M. Zhou, and A. C. Ammari, “Centralized Charging Strategy and Scheduling Algorithm for Electric Vehicles Under a Battery Swapping Scenario,”

IEEE Trans. Intell. Transp. Syst., vol. 17, no. 3, pp. 659–669, Mar. 2016, doi: 10.1109/TITS.2015.2487323.

[39] N. Bañol Arias, C. Sabillón, J. F. Franco, J. Quirós-Tortós, and M. J. Rider, "Hierarchical Optimization for User-Satisfaction-Driven Electric Vehicles Charging Coordination in Integrated MV/LV Networks," *IEEE Syst. J.*, vol. 17, no. 1, pp. 1247–1258, Mar. 2023, doi: 10.1109/JSYST.2022.3188220.

[40] H. Wu, G. K.-H. Pang, K. L. Choy, and H. Y. Lam, "Dynamic resource allocation for parking lot electric vehicle recharging using heuristic fuzzy particle swarm optimization algorithm," *Appl. Soft Comput.*, vol. 71, pp. 538–552, Oct. 2018, doi: 10.1016/j.asoc.2018.07.008.

[41] S.-M. Hsu, "Power Systems - Basic Concepts and Applications - Part I," PDH Online, Fairfax, VA, 2012. Accessed: Feb. 21, 2023. [Online]. Available: https://pdhonline.com/courses/e104a/Module1.pdf

[42] "Proposed Terms & Definitions for Power System Stability," *IEEE Trans. Power Appar. Syst.*, vol. PAS-101, no. 7, pp. 1894–1898, Jul. 1982, doi: 10.1109/TPAS.1982.317476.

[43] J. Zhu, *Optimization of Power System Operation*, Second Edition. in IEEE Press Series on Power Engineering. Hoboken, New Jersey: John Wiley & Sons, Inc., 2015.

[44] C. B. Saner, A. Trivedi, and D. Srinivasan, "A Cooperative Hierarchical Multi-Agent System for EV Charging Scheduling in Presence of Multiple Charging Stations," *IEEE Trans. Smart Grid*, vol. 13, no. 3, pp. 2218–2233, May 2022, doi: 10.1109/TSG.2022.3140927.

[45] "How does solar power work?," National Grid Group. Accessed: Feb. 21, 2023. [Online]. Available: https://www.nationalgrid.com/stories/energy-explained/how-does-solar-power-work

[46] M. H. Brown and R. P. Sedano, "Electricity Transmission, A Primer," National Council on Electric Policy, Jun. 2004. [Online]. Available: https://www.energy.gov/sites/prod/files/oeprod/DocumentsandMedia/primer.pdf

[47] K. Okedu, "Special Issue 'Emerging Topics in Renewable Energy Research in Smart Grids,'" MDPI. Accessed: Feb. 21, 2023. [Online]. Available: https://www.mdpi.com/journal/energies/special_issues/Renewabl_Energy_Research_in_Smart_Grids

[48] A. M. Fathabad, J. Cheng, K. Pan, and F. Qiu, "Data-Driven Planning for Renewable Distributed Generation Integration," *IEEE Trans. Power Syst.*, vol. 35, no. 6, pp. 4357–4368, Nov. 2020, doi: 10.1109/TPWRS.2020.3001235.

[49] W. S. Tounsi Fokui, M. J. Saulo, and L. Ngoo, "Optimal Placement of Electric Vehicle Charging Stations in a Distribution Network With Randomly Distributed Rooftop Photovoltaic Systems," *IEEE Access*, vol. 9, pp. 132397–132411, 2021, doi: 10.1109/ACCESS.2021.3112847.

[50] M. Z. Zeb *et al.*, "Optimal Placement of Electric Vehicle Charging Stations in the Active Distribution Network," *IEEE Access*, vol. 8, pp. 68124–68134, 2020, doi: 10.1109/ACCESS.2020.2984127.

[51] B. Xu *et al.*, "Reactive power optimization of a distribution network with high-penetration of wind and solar renewable energy and electric vehicles," *Prot. Control Mod. Power Syst.*, vol. 7, no. 1, pp. 1–13, 2022, doi: 10.1186/s41601-022-00271-w.

[52] "Smart Grid: The Smart Grid | SmartGrid.gov," SmartGrid.gov. Accessed: Feb. 21, 2023. [Online]. Available: https://www.smartgrid.gov/the_smart_grid/smart_grid.html

[53] "Electric Vehicles: The Smart Grid | SmartGrid.gov," SmartGrid.gov. Accessed: Feb. 21, 2023. [Online]. Available: https://www.smartgrid.gov/the_smart_grid/electric_vehicles.html

[54] K. Prabakar, "Microgrids," National Renewable Energy Laboratory. Accessed: Feb. 21, 2023. [Online]. Available: https://www.nrel.gov/grid/microgrids.html

[55] M. B. Cain, R. P. O'Neill, and A. Castillo, "History of Optimal Power Flow and Formulations," Federal Energy Regulatory Commission, Aug. 2013. [Online]. Available: https://www.ferc.gov/sites/default/files/2020-05/acopf-1-history-formulation-testing.pdf

[56] J. Carpentier, "Contribution à l'étude du dispatching économique," *Bull Société Fr Électriciens*, vol. 3, no. 8, Aug. 1962.

[57] M. J. Laly, E. P. Cheriyan, R. Sunitha, and H. H. Alhelou, "Heuristic power flow optimization of hybrid power systems integrated with bulk renewable energy sources and battery storage.," *Int. J. Low Carbon Technol.*, vol. 17, pp. 1124–1133, Jan. 2022, doi: 10.1093/ijlct/ctac066.

[58] F. B. Faruque, S. Chowdhury, Md. S. Nazim, M. Sowmitra, and A. K. M. A. M. Azad, "Optimized Distributed Generation Planning for Radial Distribution System Using Particle Swarm Optimization Algorithm," in *2021 13th IEEE PES Asia Pacific Power & Energy Engineering Conference (APPEEC)*, Nov. 2021, pp. 1–6. doi: 10.1109/APPEEC50844.2021.9687727.

[59] M. S. Alanazi, "A MILP model for optimal renewable wind DG allocation in smart distribution systems considering voltage stability and line loss," *Alex. Eng. J.*, vol. 61, no. 8, pp. 5887–5901, Aug. 2022, doi: 10.1016/j.aej.2021.11.017.

[60] "Alternative Fuels Data Center: Developing Infrastructure to Charge Electric Vehicles." Accessed: May 30, 2023. [Online]. Available: https://afdc.energy.gov/fuels/electricity_infrastructure.html

[61] R. C. Green, L. Wang, and M. Alam, "Applications and Trends of High Performance Computing for Electric Power Systems: Focusing on Smart Grid," *IEEE Trans. Smart Grid*, vol. 4, no. 2, pp. 922–931, Jun. 2013, doi: 10.1109/TSG.2012.2225646.

[62] T. Erdelić and T. Carić, "A Survey on the Electric Vehicle Routing Problem: Variants and Solution Approaches.," *J. Adv. Transp.*, pp. 1–48, May 2019, doi: 10.1155/2019/5075671.

[63] W. Qi, Z. Xu, Z.-J. M. Shen, Z. Hu, and Y. Song, "Hierarchical Coordinated Control of Plug-in Electric Vehicles Charging in Multifamily Dwellings," *IEEE Trans. Smart Grid*, vol. 5, no. 3, pp. 1465–1474, May 2014, doi: 10.1109/TSG.2014.2308217.

[64] M. Nour, J. P. Chaves-Ávila, G. Magdy, and Á. Sánchez-Miralles, "Review of Positive and Negative Impacts of Electric Vehicles Charging on Electric Power Systems," *Energies*, vol. 13, no. 18, Art. no. 18, Jan. 2020, doi: 10.3390/en13184675.

[65] M. Ansari, A. T. Al-Awami, E. Sortomme, and M. A. Abido, "Coordinated bidding of ancillary services for vehicle-to-grid using fuzzy optimization," *IEEE Trans. Smart Grid*, vol. 6, no. 1, pp. 261–270, Jan. 2015, doi: 10.1109/TSG.2014.2341625.

[66] "Electric vehicle charging," Natural Resources Canada. Accessed: Jan. 11, 2024. [Online]. Available: https://natural-resources.canada.ca/energy-efficiency/transportation-alternative-fuels/electric-vehicle-charging/25049
[67] Y. Zheng, S. Niu, Y. Shang, Z. Shao, and L. Jian, "Integrating plug-in electric vehicles into power grids: A comprehensive review on power interaction mode, scheduling methodology and mathematical foundation," *Renew. Sustain. Energy Rev.*, vol. 112, pp. 424–439, Sep. 2019, doi: 10.1016/j.rser.2019.05.059.
[68] Y. Zhang, P. You, and L. Cai, "Optimal Charging Scheduling by Pricing for EV Charging Station With Dual Charging Modes," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 9, pp. 3386–3396, Sep. 2019, doi: 10.1109/TITS.2018.2876287.
[69] "Electric Charging and Alternative Fuelling Stations Locator," Natural Resources Canada. Accessed: Jul. 18, 2023. [Online]. Available: https://natural-resources.canada.ca/energy-efficiency/transportation-alternative-fuels/electric-charging-alternative-fuelling-stationslocator-map/20487
[70] R. Das, Y. Wang, K. Busawon, G. Putrus, and M. Neaimeh, "Real-time multi-objective optimisation for electric vehicle charging management," *J. Clean. Prod.*, vol. 292, p. 126066, Apr. 2021, doi: 10.1016/j.jclepro.2021.126066.
[71] C.-K. Wen, J.-C. Chen, J.-H. Teng, and P. Ting, "Decentralized Plug-in Electric Vehicle Charging Selection Algorithm in Power Systems," *IEEE Trans. Smart Grid*, vol. 3, no. 4, pp. 1779–1789, Dec. 2012, doi: 10.1109/TSG.2012.2217761.
[72] L. Zhang, V. Kekatos, and G. B. Giannakis, "Scalable Electric Vehicle Charging Protocols," *IEEE Trans. Power Syst.*, vol. 32, no. 2, pp. 1451–1462, Mar. 2017, doi: 10.1109/TPWRS.2016.2582903.
[73] P. Richardson, D. Flynn, and A. Keane, "Local Versus Centralized Charging Strategies for Electric Vehicles in Low Voltage Distribution Systems," *IEEE Trans. Smart Grid*, vol. 3, no. 2, pp. 1020–1028, Jun. 2012, doi: 10.1109/TSG.2012.2185523.
[74] S. Stüdli, E. Crisostomi, R. H. Middleton, and R. Shorten, "A flexible distributed framework for realising electric and plug-in hybrid vehicle charging policies," *Int. J. Control*, vol. 85, pp. 1130–1145, Aug. 2012, doi: 10.1080/00207179.2012.679970.
[75] Z. Peng and L. Hao, "Decentralized Coordination of Electric Vehicle Charging Stations for Active Power Compensation," in *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, Sep. 2017, pp. 1–5. doi: 10.1109/VTCFall.2017.8288320.
[76] S. Vandael, B. Claessens, M. Hommelberg, T. Holvoet, and G. Deconinck, "A Scalable Three-Step Approach for Demand Side Management of Plug-in Hybrid Vehicles," *IEEE Trans. Smart Grid*, vol. 4, no. 2, pp. 720–728, Jun. 2013, doi: 10.1109/TSG.2012.2213847.
[77] C. P. Mediwaththe and D. B. Smith, "Game-Theoretic Electric Vehicle Charging Management Resilient to Non-Ideal User Behavior," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 11, pp. 3486–3495, Nov. 2018, doi: 10.1109/TITS.2017.2784418.
[78] W. Lee, L. Xiang, R. Schober, and V. W. S. Wong, "Electric Vehicle Charging Stations With Renewable Power Generators: A Game Theoretical Analysis," *IEEE Trans. Smart Grid*, vol. 6, no. 2, pp. 608–617, Mar. 2015, doi: 10.1109/TSG.2014.2374592.

[79] M. Mierau, R. Kohrs, and C. Wittwer, "A distributed approach to the integration of electric vehicles into future smart grids," in *2012 3rd IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*, Oct. 2012, pp. 1–7. doi: 10.1109/ISGTEurope.2012.6465772.

[80] I. Zaidi, A. Oulamara, L. Idoumghar, and M. Basset, "Electric Vehicle Charging Scheduling Problem: Heuristics and Metaheuristic Approaches," *SN Comput. Sci.*, vol. 4, no. 3, p. 283, Mar. 2023, doi: 10.1007/s42979-023-01708-1.

[81] V. Roberge, M. Tarbouchi, and G. Labonté, "Fast Genetic Algorithm Path Planner for Fixed-Wing Military UAV Using GPU," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 54, no. 5, pp. 2105–2117, Oct. 2018, doi: 10.1109/TAES.2018.2807558.

[82] A. Kapoor, P. Gangwar, A. Sharma, and A. Mohapatra, "Multi-Objective Framework for Optimal Scheduling of Electric Vehicles," in *2020 21st National Power Systems Conference (NPSC)*, Dec. 2020, pp. 1–6. doi: 10.1109/NPSC49263.2020.9331921.

[83] "Electric Vehicle Charging Stations," City of Kingston. Accessed: Jul. 05, 2023. [Online]. Available: https://www.cityofkingston.ca/residents/environment-sustainability/climate-change-energy/electric-vehicle-charging-stations

[84] J. C. Mukherjee and A. Gupta, "Distributed Charge Scheduling of Plug-In Electric Vehicles Using Inter-Aggregator Collaboration," *IEEE Trans. Smart Grid*, vol. 8, no. 1, pp. 331–341, Jan. 2017, doi: 10.1109/TSG.2016.2515849.

[85] O. M. Abdelwahab and M. F. Shaaban, "A New Day-Ahead Scheduling Approach for Smart EV Parking Lots," in *2020 6th IEEE International Energy Conference (ENERGYCon)*, Sep. 2020, pp. 906–910. doi: 10.1109/ENERGYCon48941.2020.9236616.

[86] A. K. Kalakanti and S. Rao, "A Hybrid Cooperative Method With Lévy Flights for Electric Vehicle Charge Scheduling," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 14306–14321, Sep. 2022, doi: 10.1109/TITS.2021.3127352.

[87] F. Kong, X. Liu, and I. Lee, "Joint Rate Control and Demand Balancing for Electric Vehicle Charging," in *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, Apr. 2018, pp. 213–224. doi: 10.1109/IoTDI.2018.00029.

[88] Y. He, B. Venkatesh, and L. Guan, "Optimal Scheduling for Charging and Discharging of Electric Vehicles," *IEEE Trans. Smart Grid*, vol. 3, no. 3, pp. 1095–1105, Sep. 2012, doi: 10.1109/TSG.2011.2173507.

[89] B. Alinia, M. H. Hajiesmaili, and N. Crespi, "Online EV Charging Scheduling With On-Arrival Commitment," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 12, pp. 4524–4537, Dec. 2019, doi: 10.1109/TITS.2018.2887194.

[90] Z. Xu, Z. Hu, Y. Song, W. Zhao, and Y. Zhang, "Coordination of PEVs charging across multiple aggregators," *Appl. Energy*, vol. 136, pp. 582–589, Dec. 2014, doi: 10.1016/j.apenergy.2014.08.116.

[91] D. M. Anand, R. T. de Salis, Y. Cheng, J. Moyne, and D. M. Tilbury, "A Hierarchical Incentive Arbitration Scheme for Coordinated PEV Charging Stations," *IEEE Trans. Smart Grid*, vol. 6, no. 4, pp. 1775–1784, Jul. 2015, doi: 10.1109/TSG.2015.2408213.

[92] M. R. M. Dahalan, N. M. Sapari, and M. F. Darus, "Electric Vehicle Charging Coordination on Distribution Network by Using Particle Swarm Optimization and

Genetic Algorithm," *Int. J. Innov. Technol. Explor. Eng. IJITEE*, vol. 8, no. 12, pp. 5673–5676, Oct. 2019, doi: 10.35940/ijitee.L3994.1081219.

[93] A. M. Sanchez, G. E. Coria, A. A. Romero, and S. R. Rivera, "An Improved Methodology for the Hierarchical Coordination of PEV Charging," *IEEE Access*, vol. 7, pp. 141754–141765, 2019, doi: 10.1109/ACCESS.2019.2943295.

[94] L. Luo, P. He, W. Gu, W. Sheng, K. Liu, and M. Bai, "Temporal-spatial scheduling of electric vehicles in AC/DC distribution networks," *Energy*, vol. 255, p. 124512, Sep. 2022, doi: 10.1016/j.energy.2022.124512.

[95] N. Bañol Arias, J. F. Franco, M. Lavorato, and R. Romero, "Metaheuristic optimization algorithms for the optimal coordination of plug-in electric vehicle charging in distribution systems with distributed generation," *Electr. Power Syst. Res.*, vol. 142, pp. 351–361, Jan. 2017, doi: 10.1016/j.epsr.2016.09.018.

[96] J. Peppanen and S. Grijalva, "Neighborhood electric vehicle charging scheduling using particle swarm optimization," in *2014 IEEE PES General Meeting | Conference & Exposition*, Jul. 2014, pp. 1–5. doi: 10.1109/PESGM.2014.6939912.

[97] R. T. Rockafellar, "What is Optimization?" Depts. of Mathematics and Applied Mathematics, University of Washington, 1997. Accessed: Feb. 23, 2023. [Online]. Available: https://sites.math.washington.edu/~burke/crs/515/notes/nt_1.pdf

[98] S. P. Boyd and L. Vandenberghe, *Convex optimization*, Seventh. Cambridge, UK ; New York: Cambridge University Press, 2009. [Online]. Available: https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf#page=143

[99] M. S. Hillier and F. S. Hillier, "Conventional Optimization Techniques," in *Evolutionary Optimization*, vol. 48, in International Series in Operations Research & Management Science, vol. 48. , Boston, MA: Springer, 2003. doi: 10.1007/0-306-48041-7_1.

[100] S. Ida Evangeline and P. Rathika, "A real-time multi-objective optimization framework for wind farm integrated power systems," *J. Power Sources*, vol. 498, p. 229914, Jun. 2021, doi: 10.1016/j.jpowsour.2021.229914.

[101] V. Roberge, M. Tarbouchi, and F. A. Okou, "Distribution System Optimization on Graphics Processing Unit," *IEEE Trans. Smart Grid*, vol. 8, no. 4, pp. 1689–1699, Jul. 2017, doi: 10.1109/TSG.2015.2502066.

[102] V. Kumar and S. M. Yadav, "A state-of-the-Art review of heuristic and metaheuristic optimization techniques for the management of water resources," *Water Supply*, vol. 22, no. 4, pp. 3702–3728, Jan. 2022, doi: 10.2166/ws.2022.010.

[103] F. Rothlauf, *Design of Modern Heuristics: Principles and Application*, 1st ed. in Natural Computing Series. Berlin, Heidelberg: Springer, 2011.

[104] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, Nov. 1995, pp. 1942–1948 vol.4. doi: 10.1109/ICNN.1995.488968.

[105] C. K. Das, O. Bass, G. Kothapalli, T. S. Mahmoud, and D. Habibi, "Overview of energy storage systems in distribution networks: Placement, sizing, operation, and power quality," *Renew. Sustain. Energy Rev.*, vol. 91, pp. 1205–1230, Aug. 2018, doi: 10.1016/j.rser.2018.03.068.

[106] S. K. Joshi and J. C. Bansal, "Parameter tuning for meta-heuristics," *Knowl.-Based Syst.*, vol. 189, p. 105094, Feb. 2020, doi: 10.1016/j.knosys.2019.105094.

[107] G. F. Savari, V. Krishnasamy, V. Sugavanam, and K. Vakesan, “Optimal Charging Scheduling of Electric Vehicles in Micro Grids Using Priority Algorithms and Particle Swarm Optimization,” *Mob. Netw. Appl.*, vol. 24, no. 6, pp. 1835–1847, Dec. 2019, doi: 10.1007/s11036-019-01380-x.

[108] B. Amirhosseini and S. M. H. Hosseini, “Scheduling charging of hybrid-electric vehicles according to supply and demand based on particle swarm optimization, imperialist competitive and teaching-learning algorithms,” *Sustain. Cities Soc.*, vol. 43, pp. 339–349, Nov. 2018, doi: 10.1016/j.scs.2018.09.002.

[109] B. Ahmadi, N. B. Arias, G. Hoogsteen, and J. L. Hurink, “Multi-objective Advanced Grey Wolf optimization Framework for Smart Charging Scheduling of EVs in Distribution Grids,” in *2022 57th International Universities Power Engineering Conference (UPEC)*, Aug. 2022, pp. 1–6. doi: 10.1109/UPEC55022.2022.9917961.

[110] Y. Zhuo, T. Zhang, F. Du, and R. Liu, “A parallel particle swarm optimization algorithm based on GPU/CUDA,” *Appl. Soft Comput.*, vol. 144, p. 110499, Sep. 2023, doi: 10.1016/j.asoc.2023.110499.

[111] C. Deng, N. Liang, J. Tan, and G. Wang, “Multi-Objective Scheduling of Electric Vehicles in Smart Distribution Network,” *Sustainability*, vol. 8, no. 12, Art. no. 12, Dec. 2016, doi: 10.3390/su8121234.

[112] V. Roberge and M. Tarbouchi, “Comparison of Parallel Particle Swarm Optimizers for Graphical Processing Units and Multicore Processors,” *Int. J. Comput. Intell. Appl.*, vol. 12, no. 1, Mar. 2013, doi: 10.1142/S1469026813500065.

[113] M. Jain, V. Saihjpal, N. Singh, and S. B. Singh, “An Overview of Variants and Advancements of PSO Algorithm,” *Appl. Sci.*, vol. 12, no. 17, Art. no. 17, Jan. 2022, doi: 10.3390/app12178392.

[114] J. J. Jamian, M. N. Abdullah, H. Mokhlis, M. W. Mustafa, and A. H. A. Bakar, “Global Particle Swarm Optimization for High Dimension Numerical Functions Analysis,” *J. Appl. Math.*, vol. 2014, p. e329193, Feb. 2014, doi: 10.1155/2014/329193.

[115] “What is HPC? Introduction to high-performance computing,” IBM. Accessed: Mar. 01, 2023. [Online]. Available: https://www.ibm.com/topics/hpc

[116] H. Andrade and I. Crnkovic, “A Review on Software Architectures for Heterogeneous Platforms,” in *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*, Dec. 2018, pp. 209–218. doi: 10.1109/APSEC.2018.00035.

[117] “What is high performance computing (HPC)? | Glossary,” Hewlett Packard Enterprise. Accessed: Jul. 27, 2023. [Online]. Available: https://www.hpe.com/ca/en/what-is/high-performance-computing.html

[118] V. Roberge, “Contribution à l’optimisation des réseaux électriques intelligents par le développement d’un cadriciel pour métaheuristiques parallèles sur processeurs graphiques.,” Ph.D. dissertation, Royal Military College of Canada, Kingston, Ontario, Canada, 2016. Accessed: Dec. 01, 2023. [Online]. Available: https://hdl.handle.net/11264/841

[119] M. Bailey, “Parallel Programming: Speedups and Amdahl’s law.” Oregon State University, 2021. [Online]. Available: https://web.engr.oregonstate.edu/~mjb/cs575/Handouts/speedups.and.amdahls.law.1pp.pdf

[120] J. L. Gustafson, "Reevaluating Amdahl's law," *Commun. ACM*, vol. 31, no. 5, pp. 532–533, May 1988, doi: 10.1145/42411.42415.

[121] W. W. Hwu, D. B. Kirk, and I. E. Hajj, *Programming Massively Parallel Processors*, 4th ed. Cambridge, MA: Morgan Kaufmann, 2023.

[122] J. Russell, "HPC Market will Reach $33B in 2023 and Pass $50B by 2026 – Hyperion Research," HPCwire. Accessed: Jul. 06, 2023. [Online]. Available: https://www.hpcwire.com/2023/06/07/hpc-market-will-reach-33b-in-2023-and-pass-50b-by-2026-hyperion-research/

[123] G. Barlas, "Shared-memory programming: OpenMP," in *Multicore and GPU Programming: an Integrated Approach*, Second Edition., Cambridge, MA: Morgan Kaufmann, 2023.

[124] L. Chai, Q. Gao, and D. K. Panda, "Understanding the Impact of Multi-Core Architecture in Cluster Computing: A Case Study with Intel Dual-Core System," in *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid '07)*, May 2007, pp. 471–478. doi: 10.1109/CCGRID.2007.119.

[125] G. D'Angelo and M. Marzolla, "New trends in parallel and distributed simulation: From many-cores to Cloud Computing," *Simul. Model. Pract. Theory*, vol. 49, pp. 320–335, Dec. 2014, doi: 10.1016/j.simpat.2014.06.007.

[126] Pure Storage, "Parallel vs. Distributed Computing: An Overview," Pure Storage Blog. Accessed: Jan. 23, 2024. [Online]. Available: https://blog.purestorage.com/purely-informational/parallel-vs-distributed-computing-an-overview/

[127] "CUDA Toolkit Documentation 12.3 Update 2," NVIDIA Developer. Accessed: Jan. 22, 2024. [Online]. Available: https://docs.nvidia.com/cuda/index.html

[128] Mark Harris, "An Even Easier Introduction to CUDA," NVIDIA Technical Blog. Accessed: Jan. 22, 2024. [Online]. Available: https://developer.nvidia.com/blog/even-easier-introduction-cuda/

[129] G. Venter and J. Sobieszczanski-Sobieski, "A Parallel Particle Swarm Optimization Algorithm Accelerated by Asynchronous Evaluations".

[130] J.F. Schutte, B.J. Fregly, R.T. Haftka, and A. D. George, "A Parallel Particle Swarm Optimizer," in *Proceedings of the 5th World Congress of Structural and Multidisciplinary Optimization (WCSMO-5)*, Lido di Jesolo-Venice, Italy, 2003. Accessed: Jan. 20, 2024. [Online]. Available: https://apps.dtic.mil/sti/pdfs/ADA466417.pdf

[131] D. Freitas, L. G. Lopes, and F. Morgado-Dias, "Particle Swarm Optimisation: A Historical Review Up to the Current Developments," *Entropy*, vol. 22, no. 3, Art. no. 3, Mar. 2020, doi: 10.3390/e22030362.

[132] V. Roberge and M. Tarbouchi, "Comparison of Parallel Metaheuristics for flux optimization for Induction Motor," *WSEAS Trans. Power Syst.*, vol. 9, pp. 352–359, 2014.

[133] S. Huang and V. Dinavahi, "GPU-based parallel real-time volt/var optimisation for distribution network considering distributed generators," *IET Gener. Transm. Distrib.*, vol. 12, no. 20, pp. 4472–4481, 2018, doi: 10.1049/iet-gtd.2017.1887.

[134] L. F. Grisales-Noreña, D. Gonzalez Montoya, and C. A. Ramos-Paja, “Optimal Sizing and Location of Distributed Generators Based on PBIL and PSO Techniques,” *Energies*, vol. 11, no. 4, Art. no. 4, Apr. 2018, doi: 10.3390/en11041018.

[135] L. F. Grisales-Noreña, O. D. Montoya, and C. A. Ramos-Paja, “An energy management system for optimal operation of BSS in DC distributed generation environments based on a parallel PSO algorithm,” *J. Energy Storage*, vol. 29, p. 101488, Jun. 2020, doi: 10.1016/j.est.2020.101488.

[136] V. Roberge, M. Tarbouchi, and G. Labonté, “Parallel Implementation and Comparison of Two UAV Path Planning Algorithms,” in *Proceedings of the International Conference on Evolutionary Computation Theory and Applications (ECTA-2011)*, Paris, France: SciTePress - Science and and Technology Publications, 2011, pp. 162–167. doi: 10.5220/0003663501620167.

[137] V. Roberge, M. Tarbouchi, and G. Labonte, “Comparison of Parallel Genetic Algorithm and Particle Swarm Optimization for Real-Time UAV Path Planning,” *IEEE Trans. Ind. Inform.*, vol. 9, no. 1, pp. 132–141, Feb. 2013, doi: 10.1109/TII.2012.2198665.

[138] V. Roberge and M. Tarbouchi, “Parallel Particle Swarm Optimization on Graphical Processing Unit for Pose Estimation,” *World Sci. Eng. Acad. Soc. WSEAS Trans. Comput.*, vol. 11, no. 6, pp. 170–179, Jun. 2012.

[139] V. Roberge and M. Tarbouchi, “Efficient parallel Particle Swarm Optimizers on GPU for real-time harmonic minimization in multilevel inverters,” in *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, Montreal, QC, Canada: IEEE, Oct. 2012, pp. 2275–2282. doi: 10.1109/IECON.2012.6388882.

[140] S. E. Papadakis and A. G. Bakrtzis, “A GPU accelerated PSO with application to Economic Dispatch problem,” in *2011 16th International Conference on Intelligent System Applications to Power Systems*, Sep. 2011, pp. 1–6. doi: 10.1109/ISAP.2011.6082162.

[141] C.-L. Liao, S.-J. Lee, Y.-S. Chiou, C.-R. Lee, and C.-H. Lee, “Power consumption minimization by distributive particle swarm optimization for luminance control and its parallel implementations,” *Expert Syst. Appl.*, vol. 96, pp. 479–491, Apr. 2018, doi: 10.1016/j.eswa.2017.11.002.

[142] K. Clement-Nyns, E. Haesen, and J. Driesen, “The Impact of Charging Plug-In Hybrid Electric Vehicles on a Residential Distribution Grid,” *IEEE Trans. Power Syst.*, vol. 25, no. 1, pp. 371–380, Feb. 2010, doi: 10.1109/TPWRS.2009.2036481.

[143] A. Amin, A. Mahmood, A. R. Khan, K. Arshad, K. Assaleh, and A. Zoha, “A Two-Stage Multi-Agent EV Charging Coordination Scheme for Maximizing Grid Performance and Customer Satisfaction,” *Sensors*, vol. 23, no. 6, Art. no. 6, Jan. 2023, doi: 10.3390/s23062925.

[144] “Parallel Speedup — Parallel Computing Concepts.” Accessed: Dec. 01, 2023. [Online]. Available: https://selkie.macalester.edu/csinparallel/modules/IntermediateIntroduction/build/html/ParallelSpeedup/ParallelSpeedup.html

[145] D. P. Rodgers, “Improvements in multiprocessor system design,” *ACM SIGARCH Comput. Archit. News*, vol. 13, no. 3, pp. 225–231, Jun. 1985, doi: 10.1145/327070.327215.

[146] G. Sharma and J. Martin, “MATLAB®: A Language for Parallel Computing,” *Int. J. Parallel Program.*, vol. 37, no. 1, pp. 3–36, Feb. 2009, doi: 10.1007/s10766-008-0082-5.

[147] V. Roberge, “Taurus HPC Cluster.” Accessed: Jul. 05, 2023. [Online]. Available: https://roberge.segfaults.net/wordpress/taurus-hpc-cluster/

[148] “Parallel Computing Toolbox,” MathWorks Help Center. Accessed: Nov. 29, 2023. [Online]. Available: https://www.mathworks.com/help/parallel-computing/index.html?s_tid=CRUX_lftnav

[149] “MATLAB Parallel Computing Toolbox version 7.8 (R2023a).” The MathWorks Inc., Natick, Massachusetts, United States, 2023. Accessed: Nov. 22, 2023. [Online]. Available: https://www.mathworks.com

[150] “MATLAB Parallel Server,” MathWorks Help Center. Accessed: Jan. 11, 2024. [Online]. Available: https://www.mathworks.com/help/matlab-parallel-server/index.html?s_tid=CRUX_lftnav

[151] “Execute for-loop iterations in parallel on workers - MATLAB parfor,” MathWorks Help Center. Accessed: Nov. 29, 2023. [Online]. Available: https://www.mathworks.com/help/parallel-computing/parfor.html

[152] “Parallel Computing Toolbox User’s Guide (2023b).” The MathWorks, Inc, 2023. [Online]. Available: https://www.mathworks.com/help/pdf_doc/parallel-computing/parallel-computing.pdf

[153] R. D. Zimmerman and C. E. Murillo-Sánchez, “MATPOWER (version 7.1).” 2020. Accessed: Nov. 15, 2023. [Online]. Available: https://matpower.org

[154] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, “MATPOWER: Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education,” *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, Feb. 2011, doi: 10.1109/TPWRS.2010.2051168.

[155] R. D. Zimmerman and C. E. Murillo-Sánchez, “Matpower User’s Manual, Version 7.1.” 2020. doi: 10.5281/zenodo.4074122.

[156] B. Bruinders, “Which EVs are V2G compatible?,” Climatebiz. Accessed: Jun. 28, 2023. [Online]. Available: https://climatebiz.com/v2g-compatible-evs/

[157] “Which Electric Cars Have Bidirectional Charging (V2L, V2G, V2H)?,” Zecar. Accessed: Jun. 28, 2023. [Online]. Available: https://zecar.com/resources/which-electric-cars-have-bidirectional-charging

[158] Jiawei Han, Micheline Kamber, and Jian Pei, “Chapter 3 - Data Preprocessing,” in *Data Mining: Concepts and Techniques*, Third Edition., in Morgan Kaufmann Series in Data Management Systems. , 2012, pp. 83–124. Accessed: Jan. 16, 2024. [Online]. Available: https://doi.org/10.1016/B978-0-12-381479-1.00003-4

[159] “Nested parfor and for-Loops and Other parfor Requirements - MATLAB & Simulink,” MathWorks Help Center. Accessed: Oct. 27, 2023. [Online]. Available: https://www.mathworks.com/help/parallel-computing/nested-parfor-loops-and-for-loops.html

[160] J. Suomela, “Chapter 3: Multithreading with OpenMP,” Aalto University CS-E4580 Programming Parallel Computers. Accessed: Jan. 09, 2024. [Online]. Available: https://ppc.cs.aalto.fi/ch3/nested/

[161] “OpenMP Application Programming Interface 5.2 Specification.” OpenMP, Nov. 2021. Accessed: Jan. 09, 2024. [Online]. Available: https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5-2.pdf

[162] “Nested Parallelism,” OpenMP API User’s Guide: Sun(TM) Studio 10. Accessed: Jan. 09, 2024. [Online]. Available: https://docs.oracle.com/cd/E19059-01/stud.10/819-0501/2_nested.html

[163] M. Clerc, *Particle Swarm Optimization*. London, UK: ISTE Ltd, 2006.

[164] W. M. Grady, M. J. Samotyj, and A. H. Noyola, “The application of network objective functions for actively minimizing the impact of voltage harmonics in power systems,” *IEEE Trans. Power Deliv.*, vol. 7, no. 3, pp. 1379–1386, Jul. 1992, doi: 10.1109/61.141855.

[165] M. E. Baran and F. F. Wu, “Network reconfiguration in distribution systems for loss reduction and load balancing,” *IEEE Trans. Power Deliv.*, vol. 4, no. 2, pp. 1401–1407, Apr. 1989, doi: 10.1109/61.25627.

[166] M. E. Baran and F. F. Wu, “Optimal capacitor placement on radial distribution systems,” *IEEE Trans. Power Deliv.*, vol. 4, no. 1, pp. 725–734, Jan. 1989, doi: 10.1109/61.19265.

[167] H. M. Khodr, F. G. Olsina, P. M. D. O.-D. Jesus, and J. M. Yusta, “Maximum savings approach for location and sizing of capacitors in distribution systems,” *Electr. Power Syst. Res.*, vol. 78, no. 7, pp. 1192–1203, Jul. 2008, doi: 10.1016/j.epsr.2007.10.002.

[168] “MATLAB version 9.14.0.2254940 (R2023a).” The MathWorks Inc., Natick, Massachusetts, United States, 2023. Accessed: Nov. 22, 2023. [Online]. Available: https://www.mathworks.com

[169] “MATLAB Parallel Server version 7.8 (R2023a).” The MathWorks Inc., Natick, Massachusetts, United States, 2023. Accessed: Jan. 29, 2024. [Online]. Available: https://www.mathworks.com

[170] A. Sahu, S. K. Panigrahi, and S. Pattnaik, “Fast Convergence Particle Swarm Optimization for Functions Optimization,” *Procedia Technol.*, vol. 4, pp. 319–324, Jan. 2012, doi: 10.1016/j.protcy.2012.05.048.

Appendix A
EV Parking Lot Profiles

The following tables contain the EV parking lot charging profiles for EV parking lots 2-9 (Profile 2 to Profile 9).

Table 6.1 Parking Lot Profile 2

EV Number	Arrival Timeslot	Departure Timeslot	Charging Demand (kWh)
1	3	5	18
2	5	10	26
3	5	9	25
4	1	6	26
5	9	10	15
6	2	6	22
7	6	7	14
8	4	9	26
9	1	2	10
10	1	4	20
11	2	7	26
12	3	7	17
13	5	10	26
14	3	8	26
15	4	8	14
16	4	7	12
17	5	8	16
18	3	7	19
19	1	10	28
20	5	10	26

Table 6.2 Parking Lot Profile 3

EV Number	Arrival Timeslot	Departure Timeslot	Charging Demand (kWh)
1	2	7	26
2	5	9	22
3	6	10	19
4	1	10	28
5	2	6	25
6	1	5	17
7	2	4	18
8	2	6	16
9	4	8	25
10	4	7	16
11	1	5	25
12	7	9	16
13	3	7	25
14	7	10	20
15	5	9	25
16	4	9	26
17	3	7	16
18	1	6	26
19	1	3	18
20	3	5	15

Table 6.3 Parking Lot Profile 4

EV Number	Arrival Timeslot	Departure Timeslot	Charging Demand (kWh)
1	2	7	26
2	6	8	22
3	3	7	19
4	8	10	22
5	1	10	28
6	1	6	26
7	6	8	18
8	5	7	22
9	3	6	20
10	1	3	22
11	2	5	20
12	1	10	28
13	5	8	15
14	5	10	26
15	4	8	25
16	2	7	26
17	1	6	26
18	3	8	26
19	2	4	22
20	5	10	26

Table 6.4 Parking Lot Profile 5

EV Number	Arrival Timeslot	Departure Timeslot	Charging Demand (kWh)
1	6	7	14
2	3	4	10
3	3	5	16
4	2	3	10
5	8	9	10
6	1	6	12
7	1	2	10
8	2	4	18
9	1	2	15
10	5	8	12
11	2	5	12
12	8	9	10
13	7	10	12
14	8	9	14
15	6	8	15
16	4	7	12
17	3	4	10
18	6	10	14
19	3	4	10
20	7	9	15

Table 6.5 Parking Lot Profile 6

EV Number	Arrival Timeslot	Departure Timeslot	Charging Demand (kWh)
1	2	5	20
2	5	7	15
3	6	10	25
4	1	3	18
5	9	10	15
6	2	6	22
7	6	7	14
8	5	7	16
9	8	9	15
10	5	8	20
11	5	10	26
12	3	7	17
13	6	9	15
14	1	4	20
15	4	8	14
16	1	2	15
17	5	8	16
18	3	7	19
19	1	10	28
20	4	5	15

Table 6.6 Parking Lot Profile 7

EV Number	Arrival Timeslot	Departure Timeslot	Charging Demand (kWh)
1	7	8	14
2	6	10	22
3	5	9	19
4	1	10	28
5	7	8	10
6	2	6	17
7	1	3	18
8	1	5	16
9	2	3	15
10	4	7	16
11	5	8	12
12	8	10	16
13	5	8	15
14	6	9	20
15	1	5	25
16	3	8	26
17	4	8	16
18	3	7	14
19	2	4	18
20	3	5	15

Table 6.7 Parking Lot Profile 8

EV Number	Arrival Timeslot	Departure Timeslot	Charging Demand (kWh)
1	2	3	14
2	2	3	10
3	4	6	16
4	1	2	10
5	9	10	10
6	2	7	12
7	2	3	10
8	3	5	18
9	1	2	15
10	7	10	12
11	3	6	12
12	7	8	10
13	6	9	12
14	3	4	14
15	3	5	15
16	2	5	12
17	5	6	10
18	1	5	14
19	5	6	10
20	4	6	15

Table 6.8 Parking Lot Profile 9

EV Number	Arrival Timeslot	Departure Timeslot	Charging Demand (kWh)
1	2	7	26
2	6	8	22
3	3	7	19
4	8	10	22
5	1	10	28
6	1	6	26
7	6	8	22
8	5	7	22
9	3	6	20
10	1	3	22
11	2	5	20
12	1	10	28
13	5	7	22
14	5	10	26
15	4	8	25
16	2	7	26
17	1	6	26
18	1	10	28
19	2	4	22
20	5	10	26

Appendix B
Test 1 and 3 Results

This appendix includes an except of the MATLAB simulation results for Test 1 and 3 (Sections 4.3 and 4.5), showing the results for the optimum test results at rounds 93 and 84 of the parallelized and sequential algorithms, as well as the final schedules and result comparisons.

```
Function start time:   7-Dec-2023 17:02:47 -0500

Starting parallel pool (parpool) using the 'Processes_Copy' profile ...
Connected to parallel pool with 9 workers.
RMC Workstation

Testing Start. Parpool startup time = 30.258 s (9 workers)
Function start time:   7-Dec-2023 17:03:18 -0500
```

[…]

```
**********************R O U N D 93**********************
PSO: Parking Lot Power
   Number of parking lots: 1
   Number of particles: 1
   Number of PSO iterations: 1
Power PSO: iteration 1, PART 1 fitness evaluation time: = 14.92 s (0:15)

MATPOWER Version 7.1, 08-Oct-2020 -- AC Power Flow (Newton)

Newton's method power flow (power balance, polar) converged in 4 iterations.

Converged in 0.00 seconds
================================================================================
|     System Summary                                                           |
================================================================================

How many?                How much?              P (MW)            Q (MVAr)
---------------------    -------------------  -------------  -----------------
Buses              18     Total Gen Capacity     100.0        -100.0 to 100.0
Generators          1     On-line Capacity       100.0        -100.0 to 100.0
Committed Gens      1     Generation (actual)     11.9              -2.1
Loads              15     Load                    11.7               7.6
  Fixed            15       Fixed                 11.7               7.6
  Dispatchable      0       Dispatchable          -0.0 of -0.0      -0.0
Shunts             10     Shunt (inj)             -0.0              11.0
Branches           17     Losses (I^2 * Z)         0.26              1.32
Transformers        0     Branch Charging (inj)     -                0.0
Inter-ties          0     Total Inter-tie Flow     0.0               0.0
Areas               1

                          Minimum                      Maximum
                 -------------------------  --------------------------------
Voltage Magnitude   1.027 p.u. @ bus 8          1.054 p.u. @ bus 1
Voltage Angle      -7.43 deg   @ bus 26         0.00 deg   @ bus 51
P Losses (I^2*R)             -                  0.05 MW    @ line 1-20
Q Losses (I^2*X)             -                  0.87 MVAr  @ line 50-1

================================================================================
|     Bus Data                                                                 |
================================================================================
 Bus      Voltage          Generation             Load
  #   Mag(pu) Ang(deg)   P (MW)   Q (MVAr)   P (MW)   Q (MVAr)
----- ------- --------  --------  --------  --------  --------
    1  1.054   -4.398       -         -         -         -
    2  1.051   -4.874       -         -       0.26      0.12
    3  1.045   -5.453       -         -       0.40      0.25
    4  1.042   -5.734       -         -       1.50      0.93
    5  1.036   -6.355       -         -       3.00      2.26
    6  1.035   -6.433       -         -       0.80      0.50
    7  1.032   -6.561       -         -       0.20      0.12
    8  1.027   -6.590       -         -       1.00      0.62
    9  1.050   -4.904       -         -       0.50      0.31
   20  1.050   -5.499       -         -       1.00      0.62
   21  1.049   -6.190       -         -       0.30      0.19
   22  1.048   -6.225       -         -       0.20      0.12
   23  1.045   -7.089       -         -       0.80      0.50
   24  1.048   -7.391       -         -       0.50      0.31
   25  1.042   -7.425       -         -       1.00      0.62
   26  1.041   -7.433       -         -       0.20      0.12
   50  1.050   -0.218       -         -         -         -
   51  1.050    0.000*   11.92     -2.07        -         -
                        --------  --------  --------  --------
               Total:     11.92     -2.07     11.66      7.59

================================================================================
|     Branch Data                                                              |
================================================================================
Brnch   From   To    From Bus Injection   To Bus Injection     Loss (I^2 * Z)
  #     Bus    Bus    P (MW)   Q (MVAr)   P (MW)   Q (MVAr)   P (MW)   Q (MVAr)
-----  -----  -----  --------  --------  --------  --------  --------  --------
   1      1      2      7.75      0.32     -7.73     -0.25     0.023      0.07
   2      2      3      6.97      0.98     -6.94     -0.91     0.027      0.08
   3      3      4      6.54      1.31     -6.53     -1.28     0.013      0.04
   4      4      5      5.03      1.00     -5.01     -0.94     0.022      0.06
   5      5      6      2.01      0.61     -2.01     -0.61     0.001      0.00
   6      6      7      1.21      0.11     -1.21     -0.10     0.002      0.00
   7      7      8      1.01      0.62     -1.00     -0.62     0.005      0.00
   8      2      9      0.50      0.31     -0.50     -0.31     0.001      0.00
   9      1     20      4.12     -1.98     -4.07      2.05     0.055      0.07
  10     20     21      3.07     -2.01     -3.04      2.04     0.027      0.04
  11     21     22      0.20      0.12     -0.20     -0.12     0.000      0.00
  12     21     23      2.54     -1.03     -2.51      1.07     0.027      0.04
  13     23     24      0.51     -1.33     -0.50      1.34     0.005      0.01
  14     23     25      1.21     -0.23     -1.20      0.24     0.005      0.01
  15     25     26      0.20      0.12     -0.20     -0.12     0.000      0.00
  16     50      1     11.91     -0.79    -11.87      1.67     0.040      0.87
  17     50     51    -11.91      2.12     11.92     -2.07     0.007      0.05
                                                             --------  --------
                                                    Total:     0.261      1.32
*****Particle 1: power flow feasibility = 1, penalty = 0
*****Particle 1: fitness = 1.0183
Power PSO: iteration 1, PART 2 fitness evaluation time: = 0.05 s (0:0)
Iteration: 1, best fitness = 1.018258391.
Power PSO Round 93
Cost = $ 53.77
Feasibility = 1
Time per round = 14.97 s (0:15)
Time per round including parpool startup time = 14.97 s (0:15)
```

[…]

```
Minimum charging cost out of 100 rounds: $53.77 from round 93
Average charging cost out of 100 feasible rounds: $57.16
Round 93 Solutions:
Transformer Limits per each Parking Lot:
                  PL 1 (kWh)
                  __________

    Particle 1        60


Parking Lot # 1, transformer limit 60.0000 kWh
**********
           T1          T2             T3          T4          T5          T6         T7          T8          T9        T10      Unmet Demand (kWh)    Met Demand (kWh)    Desired
Demand (kWh)
         ______     __________     __________     ______     _________     ______     ______     __________     ______     ______     __________________     ________________
____________________

EV 1     9.5592       8.4408              0          0            0          0          0             0          0          0                   0                  18                  18
EV 2          0            0              0        7.5          7.5          0          0             0          0          0                   0                  15                  15
EV 3     9.5239        3.493     1.7954e-08     7.9677       4.0155          0          0             0          0          0                   0                  25                  25
EV 4          0       9.1548      0.0059525     8.8392            0          0          0             0          0          0                   0                  18                  18
EV 5          0       9.5989         5.4011          0            0          0          0             0          0          0          -1.7764e-15                  15                  15
EV 6          0            0              0          0            0      7.753     4.6644             0          0     9.5826                   0                  22                  22
EV 7          0            0              0          0            0          0     9.5119        4.4881          0          0                   0                  14                  14
EV 8          0            0              0          0            0          0          0             0     6.4584     9.5416                   0                  16                  16
EV 9          0            0              0          0            0          0     9.5828       0.41722          0          0          -1.7764e-15                  10                  10
EV 10         0            0              0          0            0     9.2887     3.6175             0     7.0938          0           3.5527e-15                  20                  20
EV 11         0            0              0     1.3405       8.2198     8.2198     8.2198    4.2514e-08          0          0                   0                  26                  26
EV 12         0   4.7788e-05              0     6.6806       4.6476     5.6717          0             0          0          0                   0                  17                  17
EV 13         0            0              0          0       7.3372     7.6628          0             0          0          0          -1.7764e-15                  15                  15
EV 14         0            0              0     9.0084    0.0090424     3.4839     3.4984    0.00034914          0          0           1.7764e-15                  16                  16
EV 15         0            0      0.0013075     1.5723       4.9773     5.1414     2.3076             0          0          0                   0                  14                  14
EV 16         0            0              0          0        5.418     2.8946     3.6874             0          0          0                   0                  12                  12
EV 17         0            0              0     1.6007       5.4646     3.0561     5.8786             0          0          0                   0                  16                  16
EV 18         0            0              0          0       9.2136     4.8019      2.475             0     2.5095          0                   0                  19                  19
EV 19    9.5822      0.01481      0.0049156     3.7445      0.11984     2.0031     2.9279     0.0030483          0     9.5996          -3.5527e-15                  28                  28
EV 20    9.5884      0.38607     1.5919e-06      2.952       3.0735          0          0             0          0          0                   0                  16                  16

                   T1        T2        T3        T4        T5        T6        T7        T8        T9        T10
                 ______    ______    ______    ______    ______    ______    ______    ______    ______    ______

Demand (kWh)    38.254    31.088    5.4132    51.206    59.996    59.977    56.371    4.9087    16.062    28.724

               T1        T2        T3        T4        T5        T6        T7        T8        T9        T10      Total Cost ($)
             ______    ______    ______    ______    ______    ______    ______    ______    ______    ______    ______________

Cost ($)    3.8254    6.2177    2.1653    10.241    5.9996    5.9977    11.274    1.9635    3.2123    2.8724        53.769

Total cost = $53.76934 OR $53.76934.
Total unmet demand = 0.000000000 kWh.

-------------------------------------------------------------------------
Parallel pool using the 'Processes_Copy' profile is shutting down.
Function start time:   7-Dec-2023 17:29:33 -0500
```

[…]

```
**********************R O U N D 84**********************
PSO: Parking Lot Power
   Number of parking lots: 1
   Number of particles: 1
   Number of PSO iterations: 1
Power PSO: iteration 1, PART 1 fitness evaluation time: = 18.42 s (0:18)

MATPOWER Version 7.1, 08-Oct-2020 -- AC Power Flow (Newton)

Newton's method power flow (power balance, polar) converged in 4 iterations.

Converged in 0.00 seconds
================================================================================
|     System Summary                                                           |
================================================================================

How many?                How much?              P (MW)            Q (MVAr)
---------------------    -------------------  -------------  -----------------
Buses             18     Total Gen Capacity     100.0        -100.0 to 100.0
Generators         1     On-line Capacity       100.0        -100.0 to 100.0
Committed Gens     1     Generation (actual)     11.9              -2.1
Loads             15     Load                    11.7               7.6
  Fixed           15       Fixed                 11.7               7.6
  Dispatchable     0       Dispatchable          -0.0 of -0.0      -0.0
Shunts            10     Shunt (inj)             -0.0              11.0
Branches          17     Losses (I^2 * Z)         0.26              1.32
Transformers       0     Branch Charging (inj)     -                0.0
Inter-ties         0     Total Inter-tie Flow     0.0               0.0
Areas              1

                          Minimum                      Maximum
                 -------------------------  --------------------------------
Voltage Magnitude   1.027 p.u. @ bus 8          1.054 p.u. @ bus 1
Voltage Angle      -7.43 deg   @ bus 26         0.00 deg   @ bus 51
P Losses (I^2*R)             -                  0.05 MW    @ line 1-20
Q Losses (I^2*X)             -                  0.87 MVAr  @ line 50-1

================================================================================
|     Bus Data                                                                 |
================================================================================
 Bus      Voltage          Generation             Load
  #   Mag(pu) Ang(deg)   P (MW)   Q (MVAr)   P (MW)   Q (MVAr)
----- ------- --------  --------  --------  --------  --------
    1  1.054   -4.398       -         -         -         -
    2  1.051   -4.874       -         -       0.26      0.12
    3  1.045   -5.453       -         -       0.40      0.25
    4  1.042   -5.734       -         -       1.50      0.93
    5  1.036   -6.355       -         -       3.00      2.26
    6  1.035   -6.433       -         -       0.80      0.50
    7  1.032   -6.561       -         -       0.20      0.12
    8  1.027   -6.590       -         -       1.00      0.62
    9  1.050   -4.904       -         -       0.50      0.31
   20  1.050   -5.499       -         -       1.00      0.62
   21  1.049   -6.190       -         -       0.30      0.19
   22  1.048   -6.225       -         -       0.20      0.12
   23  1.045   -7.089       -         -       0.80      0.50
   24  1.048   -7.391       -         -       0.50      0.31
   25  1.042   -7.425       -         -       1.00      0.62
   26  1.041   -7.433       -         -       0.20      0.12
   50  1.050   -0.218       -         -         -         -
   51  1.050    0.000*    11.92     -2.07       -         -
                        --------  --------  --------  --------
               Total:     11.92     -2.07     11.66      7.59

================================================================================
|     Branch Data                                                              |
================================================================================
Brnch   From   To    From Bus Injection   To Bus Injection     Loss (I^2 * Z)
  #     Bus    Bus    P (MW)   Q (MVAr)   P (MW)   Q (MVAr)   P (MW)   Q (MVAr)
-----  -----  -----  --------  --------  --------  --------  --------  --------
   1      1      2      7.75      0.32     -7.73     -0.25     0.023      0.07
   2      2      3      6.97      0.98     -6.94     -0.91     0.027      0.08
   3      3      4      6.54      1.31     -6.53     -1.28     0.013      0.04
   4      4      5      5.03      1.00     -5.01     -0.94     0.022      0.06
   5      5      6      2.01      0.61     -2.01     -0.61     0.001      0.00
   6      6      7      1.21      0.11     -1.21     -0.10     0.002      0.00
   7      7      8      1.01      0.62     -1.00     -0.62     0.005      0.00
   8      2      9      0.50      0.31     -0.50     -0.31     0.001      0.00
   9      1     20      4.12     -1.98     -4.07      2.05     0.055      0.07
  10     20     21      3.07     -2.01     -3.04      2.04     0.027      0.04
  11     21     22      0.20      0.12     -0.20     -0.12     0.000      0.00
  12     21     23      2.54     -1.03     -2.51      1.07     0.027      0.04
  13     23     24      0.51     -1.33     -0.50      1.34     0.005      0.01
  14     23     25      1.21     -0.23     -1.20      0.24     0.005      0.01
  15     25     26      0.20      0.12     -0.20     -0.12     0.000      0.00
  16     50      1     11.91     -0.79    -11.87      1.67     0.040      0.87
  17     50     51    -11.91      2.12     11.92     -2.07     0.007      0.05
                                                            --------  --------
                                                   Total:     0.261      1.32
*****Particle 1: power flow feasibility = 1, penalty = 0
*****Particle 1: fitness = 1.0183
Power PSO: iteration 1, PART 2 fitness evaluation time: = 0.05 s (0:0)
Iteration: 1, best fitness = 1.018255157.
Power PSO Round 84
Cost = $ 53.78
Feasibility = 1
Time per round = 18.47 s (0:18)
Time per round including parpool startup time = 18.47 s (0:18)
```

[…]

```
Minimum charging cost out of 100 rounds: $53.78 from round 84
Average charging cost out of 100 feasible rounds: $57.04
Round 84 Solutions:
Transformer Limits per each Parking Lot:
                  PL 1 (kWh)
                  __________

    Particle 1        60


Parking Lot # 1, transformer limit 60.0000 kWh
**********
           T1         T2          T3          T4           T5           T6         T7           T8            T9<         T10      Unmet Demand (kWh)    Met Demand (kWh)    Desired
Demand (kWh)
         ______     ______     ________     _______     _________     ______     ______     __________     __________     ______     __________________     ________________
____________________

EV 1     9.5971     8.4029           0           0            0            0          0            0             0           0                    0                  18                  18
EV 2          0          0    0.011288      5.8523       9.1364            0          0            0             0           0                    0                  15                  15
EV 3     9.5983     5.0134           0      1.2595       9.1288            0          0            0             0           0                    0                  25                  25
EV 4          0     8.9447           0      9.0553            0            0          0            0             0           0                    0                  18                  18
EV 5          0     9.5993      5.4007           0            0            0          0            0             0           0           1.7764e-15                  15                  15
EV 6          0          0           0           0            0       7.9814     0.7329            0        3.6901      9.5956                    0                  22                  22
EV 7          0          0           0           0            0            0     9.5979       4.4021             0           0          -1.7764e-15                  14                  14
EV 8          0          0           0           0            0            0          0            0        6.4002      9.5998                    0                  16                  16
EV 9          0          0           0           0            0            0     9.5965      0.40349             0           0                    0                  10                  10
EV 10         0          0           0           0            0       8.0408     3.7751            0        8.1841           0                    0                  20                  20
EV 11         0          0           0       2.432       8.5867       8.5102     6.4712            0             0           0                    0                  26                  26
EV 12         0     2.8029           0      5.0997       3.9978       5.0996          0            0             0           0                    0                  17                  17
EV 13         0          0           0           0       8.7571       6.2429          0            0             0           0                    0                  15                  15
EV 14         0          0           0      4.5608    0.0002748       6.3436     5.0953            0             0           0                    0                  16                  16
EV 15         0          0           0           0       5.7555            0     8.2445            0             0           0          -1.7764e-15                  14                  14
EV 16         0          0           0           0       5.7915       4.5593     1.6492   7.7417e-09             0           0                    0                  12                  12
EV 17         0          0           0      2.2925       6.8561       6.8514          0            0             0           0                    0                  16                  16
EV 18         0          0           0           0       1.9864       6.3692     1.8535   0.00052199        8.7903           0                    0                  19                  19
EV 19    9.3238          0           0       9.338            0            0          0   2.2447e-09    0.00025072       9.338                    0                  28                  28
EV 20    9.5984      6.202           0     0.19958            0            0          0            0             0           0           3.5527e-15                  16                  16

                   T1        T2        T3       T4        T5        T6        T7        T8        T9       T10
                 ______    ______    _____    _____    ______    ______    ______    ______    ______    ______

Demand (kWh)    38.118    40.965     5.412    40.09    59.997    59.998    47.016    4.8062    27.065    28.533

              T1        T2        T3        T4        T5        T6        T7        T8        T9       T10      Total Cost ($)
            ______    ______    ______    ______    ______    ______    ______    ______    _____    ______    ______________

Cost ($)    3.8118    8.1931    2.1648    8.0179    5.9997    5.9998    9.4032    1.9225    5.413    2.8533        53.779

Total cost = $53.77904 OR $53.77904.
Total unmet demand = 0.000000000 kWh.

----------------------------------------------------------Comparison complete----------------------------------------------------------
```

[…]

```
Cost Comparison - across 100 (parallelized) and 100 (sequential) feasible rounds of tests
 Parallelized || Sequential
(1) $56.32 || $55.94
(2) $58.23 || $57.02
(3) $55.60 || $56.69
(4) $58.90 || $58.83
(5) $56.37 || $55.18
(6) $58.26 || $57.57
(7) $57.61 || $57.75
(8) $58.99 || $55.59
(9) $56.75 || $53.92
(10) $59.62 || $60.55
(11) $58.15 || $59.27
(12) $56.66 || $57.45
(13) $59.55 || $58.40
(14) $54.17 || $56.75
(15) $57.62 || $59.01
(16) $55.20 || $56.68
(17) $56.52 || $55.15
(18) $57.97 || $54.39
(19) $58.69 || $56.08
(20) $57.75 || $59.35
(21) $58.49 || $55.54
(22) $58.79 || $57.59
(23) $58.14 || $55.07
(24) $54.92 || $56.17
(25) $55.81 || $55.62
(26) $56.21 || $59.49
(27) $55.22 || $55.47
(28) $55.76 || $58.77
(29) $58.60 || $54.62
(30) $55.53 || $55.50
(31) $56.95 || $55.18
(32) $56.76 || $54.62
(33) $59.07 || $59.98
(34) $57.14 || $59.34
(35) $59.27 || $55.67
(36) $56.06 || $53.96
(37) $57.00 || $55.01
(38) $56.48 || $56.67
(39) $58.39 || $56.46
(40) $57.84 || $60.03
(41) $54.00 || $59.10
(42) $56.18 || $55.87
(43) $54.73 || $58.49
(44) $57.63 || $56.97
(45) $55.21 || $57.05
(46) $54.01 || $61.66
(47) $59.56 || $61.05
(48) $59.82 || $60.98
(49) $59.09 || $55.83
(50) $59.88 || $57.51
(51) $57.03 || $54.75
(52) $56.14 || $62.12
(53) $54.13 || $55.60
(54) $55.95 || $58.65
(55) $57.94 || $54.09
(56) $57.60 || $60.63
(57) $59.78 || $54.93
(58) $58.19 || $55.34
(59) $54.57 || $55.25
(60) $55.96 || $55.24
(61) $55.34 || $55.69
(62) $53.78 || $58.59
(63) $59.23 || $54.07
(64) $55.81 || $56.30
(65) $56.91 || $57.78
(66) $59.09 || $58.49
(67) $54.51 || $57.69
(68) $58.57 || $53.80
(69) $55.81 || $61.17
(70) $55.65 || $56.60
(71) $56.41 || $57.18
(72) $56.09 || $55.71
(73) $57.42 || $62.14
(74) $54.35 || $54.69
(75) $60.11 || $59.35
(76) $60.11 || $55.83
```

```
(77) $61.36 || $57.39
(78) $57.51 || $61.91
(79) $56.08 || $55.48
(80) $57.76 || $55.13
(81) $58.42 || $57.30
(82) $56.39 || $56.91
(83) $54.59 || $56.49
(84) $58.47 || $53.78
(85) $56.79 || $58.71
(86) $55.69 || $58.21
(87) $58.14 || $57.83
(88) $55.61 || $60.22
(89) $58.90 || $57.83
(90) $59.18 || $56.04
(91) $59.39 || $58.25
(92) $54.43 || $61.63
(93) $53.77 || $57.31
(94) $59.18 || $57.62
(95) $54.64 || $55.50
(96) $55.19 || $55.40
(97) $57.98 || $54.74
(98) $58.20 || $53.92
(99) $61.00 || $54.01
(100) $59.73 || $53.95
The average feasible cost (parallelized) = $57.16
The average feasible cost (sequential) = $57.04
--> The sequential version provides the lowest average feasible cost, for 1 parking lot optimization, by $0.12
The lowest feasible cost (parallelized) = $53.77
The lowest feasible cost (sequential) = $53.78
--> The parallelized version provides the lowest feasible cost, for 1 parking lot optimization, by $0.01
Parallelized Costs (feasible):
Min: $53.77 ||| Max: $61.36 ||| Mean: $57.16 ||| Median: $57.09 ||| Standard deviation: 1.83
Sequential Costs (feasible):
Min: $53.78 ||| Max: $62.14 ||| Mean: $57.04 ||| Median: $56.69 ||| Standard deviation: 2.19
Percentage of feasible results out of 100 test rounds: 100.00 % (parallelized), 100.00 % (sequential)

Fitness Comparison
```

[…]

```
Parallelized fitnesses (feasible):
Min: 1.0160 ||| Max: 1.0183 ||| Mean: 1.0172 ||| Median: 1.0172 ||| Standard deviation: 0.00
Sequential fitnesses (feasible):
Min: 1.0158 ||| Max: 1.0183 ||| Mean: 1.0173 ||| Median: 1.0173 ||| Standard deviation: 0.00

Time Comparison
Parallelized average time per round: 15.32 s (0:15).
```

[…]

```
Sequential average time per round: 18.75 s (0:19).
```

[…]

```
Parallelized time (s):
Min: 14.91 ||| Max: 19.25 ||| Mean: 15.32 ||| Median: 15.30 ||| Standard deviation: 0.44
Sequential time (s):
Min: 17.47 ||| Max: 36.69 ||| Mean: 18.75 ||| Median: 18.10 ||| Standard deviation: 2.22
```

[…]

```
Speedup (x times) (sequential time / parallelized time)
Min: 1.17 ||| Max: 1.91 ||| Mean: 1.22 ||| Median: 1.18 ||| Standard deviation: 0.14
```

[…]

```
Function end time:   7-Dec-2023 18:00:54 -0500

Total time required to run 100 testing rounds:   00:58:06
```

Appendix C
Test 2 Results

This appendix includes the full table results for Test 2 (Section 4.4).

Table 6.9 Centralized and Decentralized Multiple Parking Lot Results: Cost, Fitness, Time, and Feasibility

Sub -test		Centralized				Decentralized			
		Cost ($)	**Fitness**	**Time (s)**	**Feasibility**	**Cost ($)**	**Fitness**	**Time (s)**	**Feasibility**
2.1	Optimum	182.17	1.0055	253.78	10	189.53	1.0180	17.43	10
	Worst	187.39	1.0053	307.63	(100%)	197.49	1.0136	21.54	(100%)
	Median	183.49	1.0054	264.17		193.00	1.0148	17.48	
	Mean	184.31	1.0054	267.40		193.81	1.0154	17.90	
	Std.	1.80	0.00	14.81		2.78	0.00	1.28	
2.2	Optimum	527.75	1.0019	531.90	10	556.56	1.0268	19.17	10
	Worst	535.29	1.0019	574.48	(100%)	578.57	1.0106	36.07	(100%)
	Median	531.15	1.0019	537.29		565.98	1.0167	19.38	
	Mean	531.65	1.0019	541.09		566.08	1.0168	21.02	
	Std.	2.43	0.00	12.60		6.47	0.00	5.29	
2.3	Optimum	527.45	1.0019	525.51	10	550.41	1.0264	17.99	10
	Worst	541.25	1.0018	576.56	(100%)	573.96	1.0107	46.13	(100%)
	Median	532.15	1.0019	536.89		564.51	1.0169	18.20	
	Mean	532.82	1.0019	538.76		563.94	1.0168	21.92	
	Std.	3.93	0.00	14.56		6.89	0.00	8.73	
2.4	Optimum	1082.49	1.0009	943.63	10	1117.15	1.0271	19.09	8
	Worst	1107.87	1.0009	1018.75	(100%)	*1156.95*	*0.2033*	37.58	(80%)
	Median	1091.19	1.0009	967.31		1128.38	1.1068	25.59	
	Mean	1094.43	1.0009	960.92		1130.15	1.0079	25.78	
	Std.	9.98	0.00	22.05		11.93	0.09	4.72	
2.5	Optimum	1079.51	1.0009	973.00	10	1116.44	1.0267	18.37	9
	Worst	1117.53	1.0009	1021.54	(100%)	1142.61	*0.2033*	37.58	(90%)
	Median	1087.05	1.0009	988.55		1131.05	1.0168	18.46	
	Mean	1091.42	1.0009	991.39		1130.32	1.0124	20.41	
	Std.	12.79	0.00	13.33		9.31	0.06	6.03	
2.6	Optimum	1624.80	1.0006	1455.90	9	1687.61	1.0272	18.98	10
	Worst	*1747.82*	*0.7895*	1628.01	(90%)	1724.23	1.0105	43.34	(100%)
	Median	1657.37	1.0006	1483.13		1692.60	1.0166	21.60	
	Mean	1667.83	*0.9795*	1497.78		1695.77	1.0169	23.70	
	Std.	38.37	0.07	49.21		11.20	0.00	7.4	
2.7	Optimum	184.42	1.0054	129.25	10	187.37	1.0180	8.57	10
	Worst	187.60	1.0053	170.53	(100%)	201.81	1.0136	12.82	(100%)
	Median	185.74	1.0054	131.68		196.60	1.0146	8.63	

	Mean	185.97	1.0053	135.16		195.38	1.0152	9.05	
	Std.	0.92	0.00	12.51		4.24	0.00	1.33	
2.8	Optimum	536.33	1.0019	264.13	10	564.00	1.0262	9.44	10
	Worst	547.26	1.0018	313.25	(100%)	582.25	1.0104	13.58	(100%)
	Median	539.38	1.0019	272.60		573.27	1.0166	9.91	
	Mean	540.63	1.0018	274.81		573.05	1.0166	9.52	
	Std.	3.73	0.00	13.91		5.45	0.00	1.29	
2.9	Optimum	1092.42	1.0009	484.69	10	1129.82	1.0269	12.45	10
	Worst	1127.34	1.0009	553.15	(100%)	1154.47	1.0105	24.14	(100%)
	Median	1102.87	1.0009	491.80		1145.41	1.0166	12.50	
	Mean	1105.76	1.0009	499.43		1143.33	1.0167	13.67	
	Std.	11.14	0.00	20.76		9.28	0.00	3.68	
2.10	Optimum	1637.72	1.0006	737.22	7	1692.00	1.0267	9.32	10
	Worst	*1764.86*	*0.0859*	815.55	(70%)	1720.29	1.0103	19.18	(100%)
	Median	1698.34	1.0006	752.73		1714.33	1.0166	9.41	
	Mean	1702.53	*0.8479*	757.91		1709.21	1.0167	11.11	
	Std.	46.41	0.31	21.32		9.18	0.00	3.23	

Results in *italic* denote an infeasible result

Table 6.10 Cost Comparison for all Sub-Tests ($)

Sub-test		Centralized	Decentralized	Difference
2.1	Optimum	182.17	189.53	7.36
	Worst	187.39	197.49	10.10
	Median	183.49	193.00	9.51
	Mean	184.31	193.81	9.50
	Std.	1.80	2.78	
2.2	Optimum	527.75	556.56	28.81
	Worst	535.29	578.57	43.28
	Median	531.15	565.98	34.83
	Mean	531.65	566.08	34.43
	Std.	2.43	6.47	
2.3	Optimum	527.45	550.41	22.96
	Worst	541.25	573.96	32.71
	Median	532.15	564.51	32.36
	Mean	532.82	563.94	31.12
	Std.	3.93	6.89	
2.4	Optimum	1082.49	1117.15	34.66
	Worst	1107.87	1143.08	35.21
	Median	1091.19	1125.87	34.68
	Mean	1094.43	1126.94	32.51
	Std.	9.98	8.28	
2.5	Optimum	1079.51	1116.44	36.93
	Worst	1117.53	1142.61	25.08
	Median	1087.05	1130.51	43.46
	Mean	1091.42	1126.61	35.19
	Std.	12.79	9.58	
2.6	Optimum	1624.80	1687.61	62.81

	Worst	1720.65	1724.23	3.58
	Median	1656.28	1692.60	36.32
	Mean	1658.94	1695.77	36.83
	Std.	27.71	11.20	
2.7	Optimum	184.42	187.37	2.95
	Worst	187.60	201.81	14.21
	Median	185.74	196.60	10.86
	Mean	185.97	195.38	9.41
	Std.	0.92	4.24	
2.8	Optimum	536.33	564.00	27.67
	Worst	547.26	582.25	34.99
	Median	539.38	573.27	33.89
	Mean	540.63	573.05	32.42
	Std.	3.73	5.45	
2.9	Optimum	1092.42	1129.82	37.40
	Worst	1127.34	1154.47	27.13
	Median	1102.87	1145.41	42.54
	Mean	1105.76	1143.33	37.57
	Std.	11.14	9.28	
2.10	Optimum	1637.72	1692.00	54.28
	Worst	1743.58	1720.29	-23.29
	Median	1691.80	1714.33	22.53
	Mean	1681.45	1709.21	27.76
	Std.	37.97	9.18	

Table 6.11 Cost and Runtime Trendline Equations Generated by Microsoft Excel

Algorithm	Cost Equation	Runtime Equation
Centralized (500 iterations)	y = 61.72x - 15.152	y = 50.982x + 86.874
Centralized (250 iterations)	y = 62.448x - 11.429	y = 25.92x + 47.473
Decentralized (500 iterations)	y = 62.573x + 2.8347	y = 0.2184x + 18.73
Decentralized (250 iterations)	y = 63.109x + 5.936	y = 0.1181x + 9.2515

Appendix D
Transformer Limit Decoding

This appendix describes the equations used to find the decoded version of the transformer limit solution vector $\boldsymbol{limTf}$ referred to in Section 3.3.

In the PSO algorithm, the each "undecoded" candidate solution array term has a value from $[0{,}1]$, and thus must be "decoded" to its proper kW value for the final transformer limit solution vector $\boldsymbol{limTf}$, similar to what was done in equation (3.5) for the single parking lot optimization algorithm.

The equations used decode the transformer limits from the range $[0{,}1]$ to the range $[0, P_{total}]$ in kW is as follows:

$$limTf_p = \left(\frac{limTf_p^u}{\sum_{p=1}^{n_{pl}} limTf_p^u}\right)P_{total}, \qquad p \in Pl \tag{6.1}$$

where $limTf_p$ is the decoded transformer capacity limit for parking lot p, $limTf_p^u$ is the undecoded transformer capacity limit for parking lot p in the range $[0{,}1]$, and P_{total} is the total power capacity available to the aggregator from equation (3.19).

The decoded version of the transformer limit solution vector $\boldsymbol{limTf}$ is what is meant when the solution vector or set of transformer limits for all parking lots $\boldsymbol{limTf}$ is referred to throughout Section 3.3.