A Comparison of HGSADC and ACS on the Site Dependent Vehicle Routing Problem

Paul Jason Mello
Department of Computer Science and Engineering
University of Nevada, Reno
Reno, USA
pmello@unr.edu

***Abstract*—The Traveling Salesman Problem (TSP) is a fundamental combinatorial optimization problem with important consequences in the field of computer science. The Vehicle Routing Problem (VRP) is a generalization of the TSP focused on identifying optimal routes over a network by utilizing metaheuristic optimization strategies to narrow the search space. In this research, we present a comparison between two methods, an Ant Colony System (ACS) and a Hybrid Genetic Search with Adaptive Diversity Control (HGSADC), on the Site Dependent Vehicle Routing Problem (SDVRP). The SDVRP is a common constrained version of VRPs that focuses on customers only being serviced by certain vehicle types, which are constrained by capacity and duration. Utilizing a benchmark SDVRP dataset provided by the OR-Library, an evaluation of ACS and HGSADC is made. Through empirical experiments we find a size-dependent crossover: HGSADC matches the best known solutions on the smallest instances, while ACS produces better solutions as the number of customers scales. Overall, HGSADC and ACS provide worse results when compared to the competitive method Adaptive Large Neighborhood Search (ALNS), but produce better results than other competitive models like Skewed Variable Neighborhood Search (Skewed-VNS). Through this work we benchmark recent advancements in metaheuristic search algorithms and compare them using SDVRP finding advantages and disadvantages of using either metaheuristic strategy.**

***Index Terms*—HGSADC, ACS, Diversity, Metaheuristic, SDVRP**

I. Introduction

VRPs are part of a challenging class of optimization problems which seek to minimize traversal costs over all routes in a network. VRPs are a generalization of the TSP, an NP-hard problem, and were first introduced in the paper The Truck Dispatch Problem by Dantzig and Ramser in 1959 [1], where the authors sought to efficiently route petrol deliveries. This paper illustrated the importance of efficient management dispatching and has since become a core idea tied to the profitability of many businesses and domains, including logistics, transportation, and scheduling. As a result of this reliance, significant research has been conducted in developing efficient methods for traversing these graphs resulting in a rich set of approach methodologies. These methods include evolutionary algorithms [2], [3], ant colony optimization [4], [5], metaheuristic search methods [6]–[8], and simulated annealing [9], [10] among others.

As methods have been developed, VRP constraints have been added to benchmark each method's effectiveness. Modern VRP variants include combinations of the following attributes such as multiple depot [11], time windows [12], periodic deliveries [13], split delivery [14], and site dependency [15]. Each of these constraints exponentially increases the difficulty of VRPs and strains the methods developed to efficiently traverse the search space. Reducing the bounds of the search space of feasible solutions has become a core component of novel methods. To attack these VRP variants methods have been developed such as local search [16] and neighborhood heuristics [2]. These methods, among others, have expanded the field by introducing algorithm reduction mechanisms. Future work may find more efficient probabilistic methods which diverge from these circuit like improvements.

In a cyclical fashion, as VRP constraints have exponentially increased the difficulty, novel methods have emerged which aim to become more efficient. In this research work, a comparison of two of these methods, Zare-Reisabadi et al.'s ACS [18] and Vidal et al.'s HGSADC [19], is conducted on SDVRP. While prior approaches have compared ACOs to hybrid genetic algorithms (HGAs) demonstrating equivalence under similar conditions [21], a modern study with updated state-of-the-art (SOTA) comparisons has not yet been conducted to identify the progress and effectiveness of both strategies. We first implement ACS and HGSADC, then run parameter sweeps across each problem. Utilizing Cordeau and Laporte's dataset [22] from OR-Library, a benchmark of the implementations is provided. Through this work, we hope to demonstrate the strengths and weaknesses of each approach and shed light on the effectiveness of modern methods on SDVRP.

The rest of this paper is organized in the following manner: Section 2 provides an overview of background and related work in evolutionary and swarm based methods. In section 3, an explanation of the various components of ACS in Zare-Reisabadi et al.'s [18] paper is provided. Section 4 then focuses on describing Vidal et al.'s HGSADC method [19] which is considered a benchmark strategy pattern. Section 5 will cover our methodology and illustrate how the experiments were carried out. Section 6 will present and analyze the results gathered from our methodology along with limitations, strengths, and weaknesses of each approach. Finally, in section 7 we conclude our work with a summary, limitations, and plans for future work.

II. Related Work

To compare HGSADC against ACS and other methods, we prepare a small survey of related works which represent the foundations and best approaches leading up to HGSADC and ACS. Metaheuristics like HGSADC and ACS have become widely popular for their capabilities at tackling NP-hard problems. Generally, the literature underlines evolutionary and swarm intelligence methods have become the most dominant on VRP variants for their ability to easily integrate efficient memetic algorithms. While genetic algorithms have had many decades of significant research, ant colony optimization has also only recently begun to be researched due to its introduction in the 1990s. In this section, we summarize key developments in GAs and ACOs on VRPs which have led to the modern updated HGSADC and ACS methods.

A. Genetic Algorithms on VRPs

GAs have grown increasingly sophisticated under many years of deep research. Early experiments often required usage of repair mechanisms to fix the many issues incurred with standard generic crossover and mutation operators. These operators have become increasingly complex and robust, and are often specialized and integrated into other system operations [23]. The most significant improvement to GAs came shortly after with the introduction of the giant tour representation with optimal route splitting. This approach utilized dynamic programming to split the giant tour into feasible subroutes. Originally designed for capacitated vehicle routing, it has since proven its general effectiveness across a wide range of VRP variants [2].

Hybridization with local search was the next major step for GAs as it introduced geometric and topologically aware information to explore and optimize GA solutions without relying solely on probabilistic evolution. This class of metaheuristic algorithms which utilize local search have been called memetic algorithms. Works like that of Nagata et al. [24] demonstrated the effectiveness of memetic algorithms, utilizing 2-opt, and penalty based control to achieve SOTA benchmarks in the early 2000's. Various other works have proposed changes and general operator tuning to GAs to achieve stronger results. This culminated in recent years with the focus shifting from GAs hand tuned for specific problems to more flexible frameworks capable of addressing broader sets of VRP variants. Together, these additions to GAs and approaches to VRPs demonstrate a clear progression from simple schemes to complex memetic algorithms.

B. Ant Colony Optimization on VRPs

ACOs have grown with increasing popularity since their inception in the mid 1990s. ACOs excel at constructing optimal paths along graph networks by maintaining pheromone matrices that guide probabilistic decisions. Early work applied ant systems to VRPs and later improved them with topological awareness to better utilize the decisions and movements of ants. For example, Bell et al. [25] refined ACO by introducing dynamic pheromone updates and local search to improve the geometric structuring of paths. These approaches, while limited, have been quickly iterated to mature ACOs to modern memetic algorithms. These rapid improvements have made ACOs a competitive metaheuristic across various problems.

Subsequent research on ACOs has been mostly focused on improving the instability of the system and tuning the methods to more diverse VRP variants. In an important work by Reimann et al. [26], the authors introduce the classic divide and conquer strategy to ACO to enable it to work efficiently on large graph networks. Other methods have sought to make components of the system more dynamic including, update rules, ants having weights, and diversification mechanisms. Each of these mechanisms creates additional adaptation through heuristic improvements to develop more efficient search methods. More recently, these methods have become increasingly hybrid and adaptive, which has provided increased flexibility to approach NP-hard problem variants. Interestingly, in the short time of ACOs existence, significant progress has been made as a result of applying the improvement strategies developed in GA research to ACOs.

C. Prior Work

Prior to Vidal's HGSADC [19] and Zare-Reisabadi's ACS [18], a 2005 paper proposed by Silva et al. [21] compared GAs and ACOs using a logistic scheduling problem as the benchmark. The paper, titled "A logistic Process and Scheduling Problem: Genetic Algorithms or Ant Colony Optimization", presented a comparison similar to our work on ancestral GA and ACO variants. The authors first surveyed prior GA and ACO results on TSP and VRP variants and distilled the works down to common similarities, effectively highlighting the expected tradeoffs between both methods. In their empirical study, they implemented a binary encoded GA and an incremental ACO that constructed only feasible solutions. They ran their experiments over a 30 day simulation period, meaning the problem fundamentally changed from day to day to create a general GA and ACO algorithm. While the GA was generally found to be faster, ACO routinely provided better and more interpretable solutions. Our work differs in that we utilize recent updates to GAs and ACOs through the use of HGSADC and ACS. We also allow each of our algorithms to specialize on each given problem unlike the generalization of Silva's method. Silva's method becomes the backbone of our research as we build a modern benchmark comparison and compare against a wider variety of diverse metaheuristic algorithms.

III. Ant Colony System

The ACS proposed by Zare-Reisabadi et al. [18] is an iterative, population based, metaheuristic method inspired by ants searching for food. Here "ants coordinate their activities through stigmergy: an indirect communication accomplished by modifying the environment in which they move" [18]. As ants travel along the graph, they release pheromones which end up marking a trail. As more ants follow a given path, the pheromone becomes more attractive for all subsequent

ants which in turn creates a positive feedback loop. Paired with local search optimizations, their method becomes a strong two phase algorithm for traversing a network of any size. In phase one, each ant traverses the route to create a feasible solution based on standard ACO, while in phase two each ant's solution is improved through four local searches. This balance between pheromone updates and topologically aware path updates constitutes a strong graph traversal method.

We begin by initializing the graph with a uniform pheromone trail to enable random sampling of the network by each ant. This allows all routes to be given an equal chance to be traversed. The pheromone trail is saved in a three dimensional τ_{ij}^{v} matrix which maps a pheromone weight τ to each edge in the graph $(i \rightarrow j)$ when traversed by a vehicle v. These pheromones are then paired with independent heuristic information given by η_{ij} where η defines the distance matrix of the edge in the graph $(i \rightarrow j)$. This distance heuristic is defined by the inverse distance $\frac{1}{d_{ij}}$ and plays an important role by informing ants that closer customers are more attractive. This provides local optimization pressures and encodes a geometric understanding about the graph topology for the ants.

In ACS, the selection criteria for the next customer j is calculated by an attractiveness score composed of $\tau^{\alpha} \times \eta^{\beta}$. Here, α and β define pheromone weights where a high α means ants tend to follow similar paths while a low α tells ants to be more independent, and heuristic weights where a high β means ants tend to pick closer customers and a lower β means ants tend to follow pheromone trails regardless of distance. To determine which customer will be visited from i to j, we draw a uniform random value $q \in [0, 1]$ and compare it against a probabilistic tradeoff parameter q_0 which controls exploitation and exploration for the ants' search process. This turns the initial equation to identify j_i^v into the following:

$$j_i^v = \text{argmax}_{\mathcal{F}_i^v}(\tau_{ij}^v)^{\alpha}(\eta_{ij})^{\beta} \text{ if } q \leq q_0 \tag{1}$$

To narrow the search space, we focus on a feasible set of customers and sum over all feasible candidates given by $\mathcal{F}_i^v$ from node i with vehicle v.

$$P_{ij}^v = \frac{\left(\tau_{ij}^v\right)^{\alpha}\left(\eta_{ij}\right)^{\beta}}{\sum_{k \in \mathcal{F}_i^v}\left(\tau_{ik}^v\right)^{\alpha}\left(\eta_{ik}\right)^{\beta}} \tag{2}$$

This equation 2 defines a roulette wheel rule with a probability distribution to identify the best candidate j in the set of feasible candidates.

The updating mechanism of the pheromone trails consists of an evaporation process which guides ants to explore other paths as a means to avoid being trapped in a local optimum. This updating process is done both globally and locally. Local updates are supported by reducing the pheromone trail of the edges traversed in the current iteration. The update at time t is simply provided by the following equation where ρ is a parameter controlling the evaporation rate of the trails and τ_0 is the initial pheromone strength:

$$\tau_{ij}^v(t) = (1-\rho)\tau_{ij}^v(t-1) + \rho(\tau_0) \tag{3}$$

Where $\tau_{ij}^v(t)$ and $\tau_{ij}^v(t-1)$ represent the pheromone strength of an edge before and after updating. Through these processes, at each iteration a best solution will be found and a subsequent global update will be performed by the following equation:

$$\tau_{ij}^v(t) = (1-\varphi)\tau_{ij}^v(t-1) + \varphi(N/T_b) \tag{4}$$

φ is the global reinforcement rate for all pheromone trails, N defines the number of customers in the SDVRP problem, and T_b is the cost of the best solution found so far. These processes define phase one of ACS and more generally the processes of most ACOs.

In phase two ACS leverages local search methods to optimize the best routes found in phase one. The authors utilize four sequential local search operators: 2-opt, swap, crossover, and mutation. Through these sequential optimizations, local search enables ACS to escape many local optima that result from the stochastic route exploration strategies applied in phase one.

IV. Hybrid Genetic Search with Adaptive Diversity Control

The HGSADC method proposed by Vidal et al. [19] is a genetic search algorithm which combines evolutionary methods for search with local optimization, an adaptive penalty system, and problem decomposition. Here, populations are divided into feasible and infeasible subpopulations to manage the diversity of solutions and are evolved separately. This population diversity becomes a crucial optimization objective as the populations are combined into offspring which go through local search in a process the authors call "Education". As the infeasible population grows and adapts, HGSADC utilizes it to diversify genetics and combine with feasible subpopulation in a process the authors call diversification. Then, geometric decompositions are done to navigate the search space more efficiently and escape local optima. Through these processes the authors develop an efficient evolutionary method that reduces the exponentially large search spaces through increased genetic diversity as a means to tackle VRP variants. We cover these processes in more depth in the following paragraphs.

Maintenance and usage of a diverse dual population consisting of feasible and infeasible solutions becomes a critical objective of this work. The authors select ordered crossover with local search education to explore the search space and refine offspring. These crossover and local search operations allow for diverse feasible and infeasible subpopulations to mix effectively. This mixing necessitates a mechanism to control the growing imbalance of feasible to infeasible solutions. To rectify this, HGSADC uses an adaptive diversity penalty mechanism which dynamically adjusts penalties to the cost function to seek a roughly 20% feasible and 80% infeasible solution split. This diversity component is very useful as infeasible solutions can often have significant global optimizations that

feasible solutions may be unable to identify after significant convergence.

Diverse genetics play an integral part of this work ensuring ample exploration of the search space. Besides dual populations, HGSADC utilizes a process of diversification which occurs periodically during the training phase. During diversification, the algorithm will preserve $\frac{\mu}{3}$ or roughly 33% of the population with the best genetic material determined by the fitness function. HGSADC then generates 2μ completely random solutions resulting in a population of $\frac{8\mu}{3}$.

Following these processes, decomposition applies a geometric partitioning sequence at consistent intervals to deal with certain problems that become apparent at scale. As the number of customers increases, so does the necessary search space for the education process. HGSADC thus implements a mechanism to separate customers into clockwise polar angles described by their angle from the depot. This allows each space to be sorted and partitioned into a granular series of subproblems which exploit spatial coherence in the coordinate system. Each subproblem is independently optimized through local search which can significantly boost the efficiency of each iteration by reducing the candidate list of potential customers. From this decomposition, each subproblem optimization is then reconstituted into three solutions which are reintegrated into the population. Rather than having to check all possible customers, we rely on spatial relationships to determine the set of feasible graph optimizations, decompose the route then recombine it. While diversification can be trapped in these subpopulations, this decomposition ensures that there is local structural diversity within each sub routine and reduces the computational complexity. The complete HGSADC algorithm can be seen in Algorithm 1.

Algorithm 1 HGSADC

```
Initialize population
while number of iterations without improvement < It_NI and time < T_max do
    Select parent solutions P_1 and P_2
    Create offspring C from P_1 and P_2 (crossover)
    Educate C (local search procedure)
    if C infeasible then
        Insert C into infeasible subpopulation
        Repair with probability P_rep
    end if
    if C feasible then
        Insert C into feasible subpopulation
    end if
    if maximum subpopulation size reached then
        Select survivors
    end if
    if best solution not improved for It_div iterations then
        Diversify population
    end if
    Adjust penalty parameters for infeasibility
    if number of iterations = k × It_dec where k ∈ ℕ* then
        Decompose the master problem
        Use HGSADC on each subproblem
        Reconstitute three solutions and insert them in the population
    end if
end while
Return best feasible solution =0
```

V. Experiments

For this work, benchmarking implementations utilizes Cordeau and Laporte's [22] SDVRP dataset provided by the OR-Library. It consists of 12 problems of varying customer counts and vehicle types. Due to resource constraints, trials are only run for problems 1–10 as problems 11 and 12 are significantly larger. Benchmarks are compared to prior work from Horn [27] which runs 15 trials and provides their average and best solutions found along with times. Our experiments utilize the best solutions to support our benchmarks and to make accurate comparisons against Horn [27], but we also include our average results for completeness. Each experiment is run using an Intel i9-13900K with multiprocessing support and Python as our language for its simplicity. Horn's work utilizes C to implement ALNS and Skewed-VNS which significantly reduces loop time leading to differences in run time. To explore the parameter space, Bayesian hyperparameter tuning is done using Optuna's TPE sampler on implementations of HGSADC and ACS on SDVRP. For each problem in the problem set, an exploration of 100 parameter configurations were carried out, each of which was evaluated using 5 independent trials across random seeds as can be seen in table I and table II. Each of these configurations ran until there were 100 iterations without improvement. We then run HGSADC and ACS utilizing the best parameter configurations found. These runs are done over 30 trials and present the best found solutions of each implementation. The results of our experiments can be found in the results section below VI.

TABLE I
BEST HGSADC PARAMETERS FOR EACH PROBLEM INSTANCE

Problem	μ	λ	t_{size}	nb_{elite}	nb_{close}	g_{size}	P_{rep}	it_{div}
pr01	48	11	6	7	9	78	0.8482	0.7723
pr02	38	12	4	7	5	48	0.7444	0.7998
pr03	38	11	6	2	7	97	0.8498	0.3153
pr04	34	12	6	9	5	69	0.8539	0.5442
pr05	36	12	6	7	7	71	0.6503	0.7545
pr06	30	10	7	3	4	46	0.5796	0.7400
pr07	39	12	5	9	6	58	0.6971	0.1366
pr08	34	10	7	8	7	36	0.6777	0.7399
pr09	30	12	7	4	1	100	0.7574	0.4220
pr10	36	12	7	4	1	49	0.8319	0.6649

Notably, for the sake of efficiency, many parameters were selected to be within specified ranges to avoid excessive computation demand and extreme resource utilization. These sweep experiments illustrated that multiple parameters tested could have been significantly larger. We found multiple param-

TABLE II
BEST ACS PARAMETERS FOR EACH PROBLEM INSTANCE

Problem	*ants*	α	β	q_0	τ_0	ρ	φ
pr01	92	1.2658	1.3908	0.7205	0.2791	0.1250	0.0748
pr02	70	0.6223	4.2464	0.7951	0.2424	0.2645	0.1909
pr03	72	2.2126	2.4062	0.7389	0.4147	0.2055	0.0844
pr04	98	0.8518	2.6449	0.9843	0.3524	0.0284	0.2835
pr05	97	1.9239	4.3351	0.9860	0.4258	0.1266	0.0658
pr06	84	1.7731	2.4551	0.8193	0.3820	0.2277	0.1037
pr07	60	1.1309	3.4846	0.8910	0.1312	0.0533	0.2564
pr08	32	0.7369	3.9780	0.7772	0.0731	0.0838	0.0136
pr09	72	1.7244	4.9745	0.9841	0.1955	0.1733	0.2795
pr10	87	1.1856	4.8161	0.7235	0.0427	0.0429	0.1803

eters set to their maximum values including the tournament size t_{size} which had a maximum sweep value set to 7 and offspring size λ which had a maximum sweep value set to 12. This is intuitive as HGSADC heavily relies on offspring diversity so pushing more offspring provides better genetic coverage of the search space. Despite this, the total population size μ was found to be between $30-40$ on average, while the number of ants $ants$ was found to be slightly more diverse between the ranges of $70-100$ with a few more exceptions. From our experimental results, it was determined that this interesting discrepancy lies in how the sampling is done for ACS and HGSADC, while ACS works through iteration level sampling, HGSADC works through genetic population level sampling. More on this later.

VI. RESULTS

In this section, a comparative evaluation between HGSADC and ACS is conducted against benchmark results gathered from Horn [27] for Skewed-VNS and ALNS. Each implementation utilizes a diverse array of operators and methods to extract efficiency of the given methods. HGSADC focuses on increasing the diversity of its genetic population. ACS works through iteration level exploration with heuristic distance knowledge regarding the topology of the graph. ALNS utilizes adaptive operator choices based on past successes of operators and exploiting global neighborhood knowledge. While Skewed-VNS applies local searches similar to ACS and HGSADC, while also measuring distance between solutions and biasing solution selection. For more information on ALNS and Skewed-VNS we point to Horn's work [27]. Each of these approaches are tailored to VRP variants and have been shown to achieve high quality results.

A. Solution Quality and Convergence Speed

Through our experimentation, we discovered that HGSADC, on average, performs better than other methods when the number of customers is kept low as seen in table III. This points to the exploitation of genetic diversity being better at traversing more limited search spaces. This makes sense as the introduction of random offspring and culling carried out by HGSADC to seek better solutions wastes significantly more computation as it parses a large random genetic set. As we scale the number of customers to the $100s$ and presumably beyond, ALNS provides better solutions, likely based on its adaptability of operator selection. ACS fails to compete with ALNS, but produces higher quality solutions based on costs than HGSADC as the problem scales. This is likely a result of its topologically aware distance objective and sampling procedures. Additionally, ACS's two phase processes of pheromone updates and optimizations completes full tours of the graph through exploration and exploitation leading to better solutions at the cost of increased computations.

Similar patterns to the ones found in the average tables hold true for the best solutions found by HGSADC and ACS. Table IV illustrates this, however we find that the gap between the BKS and the solutions found by HGSADC and ACS are significantly smaller. HGSADC has significantly more diversity as the customers counts increase, but interestingly this variance does not translate to providing better solutions. In fact, in problem 6 we see the variance in the best scenario far exceed the worst solutions found by ACS in Figure 1. This demonstrates that ACS provides more consistent high quality results than HGSADC.

ACS applies heuristic information into its next customer selection, and evenly weights all the paths during initialization providing a greedy uniform spread of routes. The smaller variance of ACS in Figure 1 is important as it describes each solution having more efficient search of the search space converging to specific solution or strategies. HGSADC conversely, relies on random initializations of genetics to identify improvements and uses significantly more methods like education, decomposition, diversity, adaptive control, and dual populations. This may lead to many poor quality solutions as the problem scales, as a result of the complex probabilistic nature of the HGSADC algorithm. This complexity may be the cause of the increased variation in the best solutions found across all problems and trials. Despite this, ALNS continues to perform the best in class across the range of tested algorithms and problems with near optimal solutions found for every problem.

Similar to the previous described results, table III and table IV both demonstrate a significant amount of time must be utilized for HGSADC and ACS when compared to their competitors. Some of this discrepancy can be directly attributed to the programming language. For example our HGSADC and ACS utilized Python implementations with many nested loops, while ALNS and Skewed-VNS are coded in the C language. The remaining difference between the theoretically expected runtimes and those found through experimentation may be a result of inefficient implementations. Despite this, various measures were taken to speed up the processes including the precomputation of the graph, multiprocessing for multiple trials at the same time, and simple caching mechanisms to reduce expensive evaluations like feasibility checks.

A typical convergence graph representative of the vast majority of problems is included for completeness. Problem 9

TABLE III

AVERAGE PERFORMANCE OVER ALL TRIALS FOR SDVRP BENCHMARK INSTANCES OF CORDEAU AND LAPORTE [22]. REPORTED VALUES INCLUDE AVERAGE SOLUTION COST f_{avg}, AVERAGE RUNTIME T_{avg}, AND AVERAGE PERCENTAGE GAP TO THE BKS. TRIAL COUNTS ARE 15 FOR SKEWED-VNS AND ALNS, AND 30 FOR HGSADC AND ACS. BEST AVERAGE SOLUTION PER INSTANCE IS SHOWN IN BOLD.

Problem	Customers	Vehicle Types	BKS	Skewed-VNS			ALNS			HGSADC			ACS		
				f_{avg}	T_{avg} [s]	gap [%]	f_{avg}	T_{avg} [s]	gap [%]	f_{avg}	T_{avg} [s]	gap [%]	f_{avg}	T_{avg} [s]	gap [%]
pr01	48	4	**1380.77**	1381.24	61	0.03	1393.85	19	0.95	**1380.77**	11	0.00	1429.90	139	3.56
pr02	96	4	**2311.54**	2339.85	210	0.40	**2330.60**	63	0.82	2393.50	1713	3.55	2540.55	610	9.91
pr03	144	4	**2602.13**	2663.12	388	2.13	**2607.66**	140	0.21	2803.01	2239	7.72	2845.94	2152	9.37
pr04	192	4	**3474.01**	3597.11	510	3.08	**3489.51**	191	0.45	3899.99	3173	12.26	3759.26	4626	8.21
pr05	240	4	**4416.38**	4549.61	791	2.67	**4431.16**	251	0.34	5202.42	4263	17.80	4770.07	8983	8.01
pr06	288	4	**4444.52**	4623.06	772	3.54	**4465.18**	314	0.46	6548.22	2338	47.33	5190.03	12651	16.77
pr07	72	6	**1889.82**	1968.76	124	2.73	1916.50	39	1.41	**1889.82**	116	0.00	1923.47	138	1.78
pr08	144	6	**2977.50**	3049.51	413	1.38	**3007.99**	135	1.02	3167.47	1834	6.38	3177.10	966	6.70
pr09	216	6	**3536.20**	3686.82	680	3.35	**3567.15**	226	0.87	4532.48	2998	28.17	4129.44	5473	16.78
pr10	288	6	**4648.76**	4839.76	729	3.55	**4673.67**	322	0.54	6621.24	3061	42.43	5392.09	13347	15.99

TABLE IV

BEST RESULTS OVER ALL TRIALS FOR THE SDVRP BENCHMARK INSTANCES OF CORDEAU AND LAPORTE [22]. REPORTED VALUES INCLUDE THE BEST OBJECTIVE VALUE f_{best}, THE TIME TO REACH IT T_{best}, AND THE PERCENTAGE GAP RELATIVE TO THE BKS. TRIAL COUNTS ARE 15 FOR SKEWED-VNS AND ALNS, AND 30 FOR HGSADC AND ACS. BKS AND ALL BEST OBTAINED VALUES ARE SHOWN IN BOLD.

Problem	Customers	Vehicle Types	BKS	Skewed-VNS			ALNS			HGSADC			ACS		
				f_{best}	T_{best} [s]	gap [%]	f_{best}	T_{best} [s]	gap [%]	f_{best}	T_{best} [s]	gap [%]	f_{best}	T_{best} [s]	gap [%]
pr01	48	4	**1380.77**	**1380.77**	10	0.00	**1380.77**	–	0.00	**1380.77**	5	0.00	**1380.77**	10	0.00
pr02	96	4	**2311.54**	2316.51	112	0.22	**2311.54**	–	0.00	2334.55	3078	1.00	2379.35	231	2.93
pr03	144	4	**2602.13**	2606.35	398	0.16	**2602.13**	–	0.00	2661.10	1935	2.27	2721.87	2111	4.60
pr04	192	4	**3474.01**	3519.08	583	1.30	**3474.01**	–	0.00	3646.30	4086	4.96	3572.56	2239	2.84
pr05	240	4	**4416.38**	4508.91	840	2.10	**4416.38**	–	0.00	4833.45	5380	9.44	4612.70	5728	4.45
pr06	288	4	**4444.52**	4532.65	930	1.98	**4444.52**	–	0.00	6007.81	1729	35.17	4900.94	11582	10.27
pr07	72	6	**1889.82**	**1889.82**	53	0.00	**1889.82**	–	0.00	**1889.82**	56	0.00	**1889.82**	68	0.00
pr08	144	6	**2977.50**	3004.52	436	0.91	**2977.50**	–	0.00	**2977.50**	2119	0.00	**2977.50**	971	0.00
pr09	216	6	**3536.20**	3629.15	630	2.63	**3536.20**	–	0.00	4062.47	2895	14.88	3930.73	4462	11.16
pr10	288	6	**4648.76**	4734.87	825	1.85	**4648.76**	–	0.00	5972.28	2550	28.47	5206.46	9638	12.00

illustrates a very common pattern as the number of customers and vehicle types scale. HGSADC, initially starts at a much worse cost as it prioritizes genetic diversity, while ACS starts from a much better cost thanks to its heuristic distance information. While HGSADC never beats ACS as the number of customers increases, it does close a significantly larger gap in similar time periods when compared to ACS. Meaning that it improves its cost by about 5000 in the same period that ACS improves its cost by about 500, an order of magnitude less. This illustrates the genetic diversity of search in HGSADC compared to the exploitation practices of informed search by ACS. Despite this, ACS remains the better approach finding better solutions significantly faster than HGSADC. Figure 2 demonstrates this typical convergence pattern.

While ACS tends to outperform HGSADC it also has a distinct advantage beyond the intrinsic heuristic objective. ACS collects additional information during training beyond its final best solution. It collects the vector information of the graph which can be researched and used more readily, especially if the graph may dynamically change. While HGSADC develops a strict genetic code for a given problem that must be retrained each time the graph changes, ACS is able to dynamically update its pathing decisions from the vector information as the graph changes. This makes ACS much more performant and dynamic than evolutionary approaches which also suffer from the same issue. At best GAs will maintain a suboptimal solution with only minor graph changes, while ACS contains a pheromone matrix which can quickly find a new optimal solution solely through the pheromone trails left by the ants.

The ACS has a fundamental framework where the ants can read, update, and exchange information about the network in a dynamic way. This added benefit of dynamic solutions is a byproduct of the significant computational costs associated with running ACS. This difference in time can be

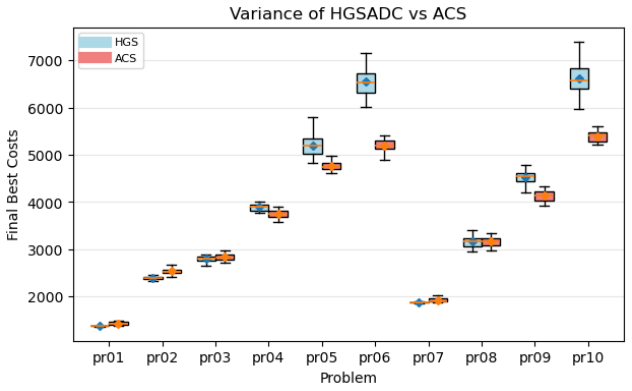

Fig. 1. HGSADC vs. ACS best cost variance for each problem across 30 runs.

seen evidently in table III and table IV which illustrate the huge computational increase of ACS over the other presented algorithms. While HGSADC runs for significantly less time, ACS has the advantage of dynamic graph updates. Despite these tradeoffs between HGSADC and ACS, the clear winner is still ALNS, which produces the best solutions on average, while also achieving the fastest runtimes. However, ALNS still shares the same limitations present in genetic algorithms, if the underlying graph dynamics change, it typically must be retrained. Without this retraining, it may produce suboptimal solutions. With continued improvements to ACS, the simple choice of always choosing ALNS may become more complex as the dynamic advantages of ACS become more beneficial.

One final, but important, result of this work is the explicit distinction between the HGSADC and ACS process. While ACS is an iteration level sampling technique, HGSADC is a population based evolutionary sampling technique. Herein lie the core differences that affect the success of each approach. HGSADC uses the samples differently from ACS. For HGSADC the key component is the genetic diversity of offspring. Generating more offspring per iteration increases the chance of producing novel, high quality genetic recombinations. However, once the population is large enough to maintain diversity and sustain the selection pressures, increasing the population size offers diminishing returns because it only adds more evaluations rather than new genetic material. ACS controls the amount of constructive samples and the stability of the pheromone updates. More ants generally means there are more solutions per iteration which can improve the probability of finding better routes and reduces the uncertainty of the pheromone solution. Since the effectiveness of ACS is determined by its pheromone trails, insufficient sampling can lead to poor solutions. Consistent with this, the parameter sweep favored a higher number of ants, which broadly resulted in better solutions for our experiments. This is in contrast to HGSADC which would have likely benefited significantly more from smaller population sizes with much larger offspring counts.

VII. Conclusion

In this research, we implemented and explored two metaheuristic algorithms which use evolutionary and swarm intelligence methods to attack SDVRP. While HGSADC utilizes diverse genetic exploration and aggressive population control, it failed to efficiently solve the SDVRP problem. ACS on the other hand produced better results as customer counts were scaled. These discrepancies highlight the differences in the nature of population based sampling and iteration based sampling. Through extensive trial runs, we show ACS outperforms HGSADC generally, but both are beaten by ALNS. ALNS produces the best results on SDVRP closing the gap to the best known solutions to approximately 1% while ACS often produces gaps of $5\% - 10\%$, and HGSADC produces gaps up to 50%. These results indicate that methods relying solely on complex genetic diversification processes may struggle to navigate the complex search space of SDVRP.

A. Limitations

One limitation of our approach is that the implementation pipeline is heavily constrained by the Python runtime overhead. Competing methods such as ALNS and Skewed-VNS benefit from C speedups, while ACS and HGSADC execute substantially fewer iterations under identical wall clock limits. The iterations without improvement approach further compounds this imbalance and influences the relative quality of solutions reached by each method. Further resource constraints related to time and hardware forced a reduction in the parameter sweep scope for each variable and required our iterations without improvements to be shorter than those presented in the HGSADC paper. Moreover, the general variance that can be introduced randomly by HGSADC operators like survivor selection makes confidence in the parameter sweep difficult to measure. With a larger parameter sweep survey it is possible that HGSADC may find better solutions, specifically with significant increases to the offspring population count. Conversely, ACS's high reliance on parameter settings makes it a significantly stronger method for traversal, making the parameter sweep especially efficient in terms of ROI on computational costs.

B. Future Work

While we have demonstrated the effectiveness of HGSADC and ACS on SDVRP, significant future work still exists. Replacement of the iterations with no improvement method with a maximum iteration count and wall clock time limit may demonstrate other interesting facets and would provide interesting perspectives on the evaluation capabilities and efficiency of each method. This study was omitted from this paper due to time constraints and code implementation complexity. Moreover, given the complexity of HGSADC's implementation future tuning of HGSADC's operators may yield significant improvements. An ablation study over education, dual populations, diversification, and decomposition may find out which of these operations yield the most benefit to cost improvements. This would explicitly demonstrate which types

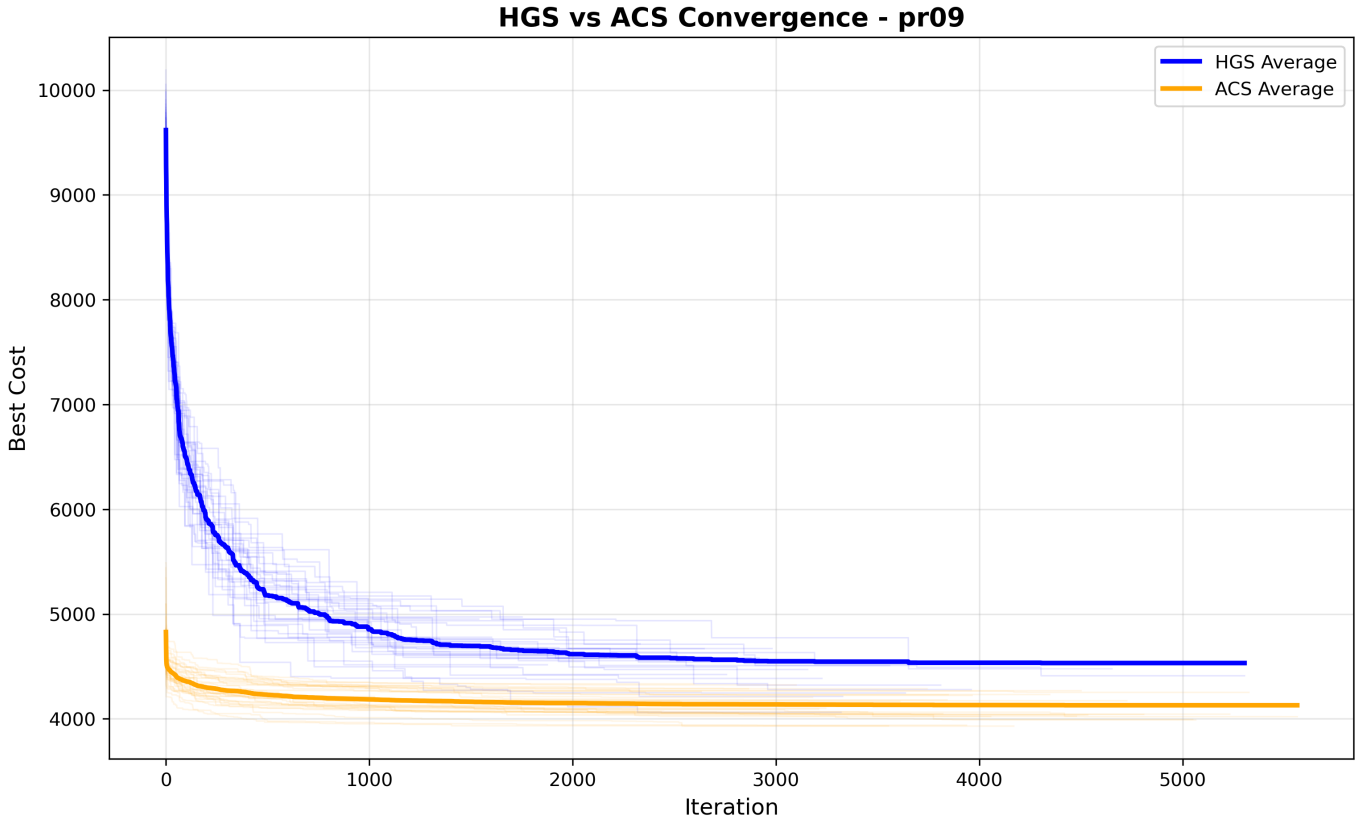

Fig. 2. A typical convergence pattern for HGSADC vs ACS as the number of customers combinatorially explodes the problem space. These averages were collected over a total of 30 runs on problem 9 from Cordeau and Laporte [22] SDVRP dataset. When the number of customers is lower than 70, HGSADC tends to have lower costs than ACS. However, exceeding that threshold, ACS performs better.

of operations can traverse the search space more effectively to tailor future research works. ACS however, would likely benefit from a larger parameter sweep, incorporation of more modern hybridization strategies, and selective pheromone update schemes to reduce the computational overhead. Overall, these directions offer a means to improve the efficiency of search and would help to explain what about a given approach is so powerful.

REFERENCES

[1] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. 80–91, 1959.

[2] C. Prins, "A simple and effective evolutionary algorithm for the vehicle routing problem," *Computers & Operations Research*, vol. 31, no. 12, pp. 1985–2002, 2004.

[3] K. C. Tan, T. H. Lee, Y. H. Chew, and L. H. Lee, "A multiobjective evolutionary algorithm for solving vehicle routing problem with time windows," in *Proc. IEEE Int. Conf. Systems, Man and Cybernetics*, vol. 1, 2003, pp. 361–366.

[4] M. Reimann, K. Dörner, and R. Hartl, "Insertion based ants for vehicle routing problems with backhauls and time windows," in *Ant Algorithms*, M. Dorigo, G. Di Caro, and M. Sampels, Eds., ser. Lecture Notes in Computer Science, vol. 2463. Berlin, Germany: Springer, 2002, pp. 135–148.

[5] L. M. Gambardella, É. Taillard, and G. Agazzi, "MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. London, U.K.: McGraw-Hill, 1999, pp. 63–76.

[6] J.-F. Cordeau, M. Gendreau, G. Laporte, J.-Y. Potvin, and F. Semet, "A guide to vehicle routing heuristics," *Journal of the Operational Research Society*, vol. 53, no. 5, pp. 512–522, 2002.

[7] B. L. Golden, E. A. Wasil, J. P. Kelly, and I.-M. Chao, "The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results," in *Fleet Management and Logistics*, T. G. Crainic and G. Laporte, Eds. Boston, MA, USA: Springer, 1998, pp. 33–56.

[8] G. Laporte, M. Gendreau, J.-Y. Potvin, and F. Semet, "Classical and modern heuristics for the vehicle routing problem," *International Transactions in Operational Research*, vol. 7, no. 4–5, pp. 285–300, 2000.

[9] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," in *Neurocomputing: Foundations of Research*, J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA, USA: MIT Press, 1988, pp. 551–567.

[10] I. H. Osman, "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem," *Annals of Operations Research*, vol. 41, no. 4, pp. 421–451, 1993.

[11] J. Renaud, G. Laporte, and F. F. Boctor, "A tabu search heuristic for the multi-depot vehicle routing problem," *Computers & Operations Research*, vol. 23, no. 3, pp. 229–235, 1996.

[12] M. Desrochers, J. Desrosiers, and M. Solomon, "The vehicle routing problem with time windows part I: Tabu search," *INFORMS Journal on Computing*, vol. 8, no. 2, pp. 158–164, May 1996.

[13] M. Gaudioso and G. Paletta, "A heuristic for the periodic vehicle routing problem," *Transportation Science*, vol. 26, no. 2, pp. 86–92, 1992.

[14] M. Dror and P. Trudeau, "Savings by split delivery routing," *Transportation Science*, vol. 23, no. 2, pp. 141–145, 1989.

[15] I.-M. Chao, B. Golden, and E. Wasil, "A computational study of a new heuristic for the site-dependent vehicle routing problem," *INFOR: Information Systems and Operational Research*, vol. 37, no. 3, pp. 319–336, 1999.

[16] O. Bräysy, "Fast local searches for the vehicle routing problem with time windows," *INFOR: Information Systems and Operational Research*, vol. 40, no. 4, pp. 319–330, 2002.
[17] M. Dorigo, "Optimization, learning and natural algorithms," Ph.D. dissertation, Politecnico di Milano, Milan, Italy, 1992.
[18] E. Zare-Reisabadi and S. H. Mirmohammadi, "Site dependent vehicle routing problem with soft time window: Modeling and solution approach," *Computers & Industrial Engineering*, vol. 90, pp. 177–185, 2015.
[19] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, "A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows," *Computers & Operations Research*, vol. 40, no. 1, pp. 475–489, 2013.
[20] M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo, "Model-based search for combinatorial optimization: A critical survey," *Annals of Operations Research*, vol. 131, no. 1, pp. 373–395, Oct. 2004.
[21] C. A. Silva, J. M. Sousa, T. Runkler, and J. M. G. Sá da Costa, "A logistic process scheduling problem: Genetic algorithms or ant colony optimization?" *IFAC Proc. Volumes*, vol. 38, no. 1, pp. 206–211, 2005.
[22] J.-F. Cordeau, M. Gendreau, and G. Laporte, "Benchmark instances for the site-dependent vehicle routing problem (SDVRP)," Vehicle Routing Problem Repository (VRP-REP), Dataset `Cordeau_al_1997_SDVRP`, ID 2017-0015, 1997. [Online]. Available: http://www.vrp-rep.org/datasets/item/2017-0015.html
[23] J.-Y. Potvin, "State-of-the-art review — evolutionary algorithms for vehicle routing," *INFORMS Journal on Computing*, vol. 21, no. 4, pp. 518–548, 2009.
[24] B. Ombuki-Berman, B. Ross, and F. Hanshar, "Multi-objective genetic algorithms for vehicle routing problem with time windows," *Applied Intelligence*, vol. 24, no. 1, pp. 17–30, 2006.
[25] J. E. Bell and P. R. McMullen, "Ant colony optimization techniques for the vehicle routing problem," *Advanced Engineering Informatics*, vol. 18, no. 1, pp. 41–48, 2004.
[26] M. Reimann, K. Doerner, and R. F. Hartl, "D-Ants: Savings based ants divide and conquer the vehicle routing problem," *Computers & Operations Research*, vol. 31, no. 4, pp. 563–591, 2004.
[27] M. Horn, "A heuristic framework for dynamic vehicle routing with site-dependent constraints," Diploma thesis, Technische Universität Wien, Vienna, Austria, 2017.