A Distributed Genetic Algorithm with Adaptive Diversity Maintenance for Ordered Problems

Ryoma Ohira
School of Information Communication and Technology
Griffith University
Southport, Australia
Email: r.ohira@griffith.edu.au

Md. Saiful Islam
School of Information Communication and Technology
Griffith University
Southport, Australia
saiful.islam@griffith.edu.au

***Abstract*—Maintaining population diversity is critical to the performance of a Genetic Algorithm (GA). Applying appropriate strategies for measuring population diversity is important in order to ensure that the mechanisms for controlling population diversity are provided with accurate feedback. Sequence-wise approaches to measuring population diversity have demonstrated their effectiveness in assisting with maintaining population diversity for ordered problems, however these processes increase the computational costs for solving ordered problems. Research in distributed GAs have demonstrated how applying different distribution models can affect an GA's ability to scale and effectively search the solution space. This paper proposes a distributed GA with adaptive parameter controls for solving ordered problems such as the travelling salesman problem (TSP), capacitated vehicle routing problem (CVRP) and the job-shop scheduling problem (JSSP). Extensive experimental results demonstrate the superiority of the proposed approach.**

***Index Terms*—Distributed Genetic Algorithm**

I. Introduction

Inspired by Darwinian theory on evolution, Genetic Algorithms (GAs) search for solutions to combinatorial optimisation problems by improving on the solutions found in previous generations. This iterative process has demonstrated the effectiveness of GAs for finding good quality solutions to real-world problems [1]. GAs conduct local search by combining good, known solutions into a new solution for the next generation while conducting global search by mutating existing solutions. Maintaining a balance between exploiting these known solutions and exploring new areas of the solution space is critical in preventing the GA from prematurely converging on a local optima [2]. Common methods for maintaining this balance include controlling the ratio between sub-populations for exploitation and exploration [3], and maintaining multiple populations where solutions can be shared when needed [4]. While strategies that implement controls for exploitation and exploration introduce adaptive parameter controls to adjust the GA's focus between the two, distributed GAs with multiple populations generally rely on diversity being maintained through the migration of solutions between populations. Where a single population is limited in its search by its population size and the influence of its individuals, multiple populations would allow for a GA to focus its search in multiple areas of the solution space in parallel. The problems for maintaining population diversity in parallel GAs can be identified as the following research questions (RQs):

RQ1 How can a GA be encouraged to concurrently search in multiple different areas of the solution space?
RQ2 How many individuals need to migrate in order to sufficiently increase the diversity of the current population?
RQ3 How can a population select individuals from other populations that best improve its quality and the diversity?

In this paper, we propose a distributed genetic algorithm that incorporates adaptive parameter controls so that each parallel population is able to independently manage its own exploration and exploitation of the solution space to address RQ1. Furthermore, the adaptive parameter mechanisms for controlling the number of individuals to migrate to a population as well as the selection of individuals that best improve the population's diversity and fitness are implemented to address RQ2 and RQ3. In doing so, we demonstrate how a distributed adaptive GA with parallel populations can improve on the performance of both distributed GAs and adaptive GAs.

II. Related Works

GAs are an inherently parallel process where individuals of a population can be processed independently. While this improves the processing time, it does not inherently improve on the search capacity of a GA. However, by running parallel populations, a GA can potentially improve its search capacity by searching in multiple areas of the solution space [5]. While the number of parallel processes can be limited when run on a single machine, developing methods and strategies for distributing the search optimisation process has been an ongoing research topic. Gong et al. [4] identified the characteristics of distributed evolutionary algorithms. In particular, the authors identified master-slave model of distribution with island and cellular population models as the most commonly used forms of distributing GAs in the literature. Dubreuil et al. [6] describe the master-slave model as a distributed approach where a master node communicates with slave nodes in order to allocate work. This is most commonly implemented as a master node responsible for the selection, crossover and mutation of the population while the slave nodes are tasked with the more computationally expensive tasks such as calculating the fitness of each individual. The island model creates parallel populations that are independent from one another but individuals can migrate between these populations when permitted [7]. The literature [1], [8] identifies GAs as one of the most commonly used approaches to solving scheduling problems. Tosun et al. [9] found that by applying parallel island-based populations, a GA was more effective at solving scheduling problems. Furthermore, by distributing parallel populations, Liu et al. [10] demonstrated the scalability of distributed GAs with large processor counts. Ghosn et al. [11] proposed a distributed and scalable GA for solving the Open-Shop Scheduling Problem (OSSP) using the island model. While these distributed GAs introduce novel strategies for conducting parallel search processes or distributing the computational load, the parameter settings for operators remained static and do not adapt to changing states in the search process.

Adaptive GAs aim to introduce intelligence to the search process by monitoring the state of the population and adjusting the GA's parameters in an online manner. Strategies for both monitoring and maintaining diversity are ongoing research topics [3], [12], [13]. Common methods for measuring population diversity include measuring the Euclidean distances [13] between individuals as well as the Hamming distance [14]–[16]. These methods compare the values of genes at each gene position to measure the difference between genotypes. This diversity measurement is then used as a feedback mechanism for adjusting the GA's parameters. McGinley et al [17] introduced an adaptive mechanism for balancing exploration and exploitation sub-population sizes depending on the level of diversity of the population that performed strongly on both ordered and non-ordered problems. Zhang et al. [16] used Hamming distance measurements to guide the search to new areas of the solution space in the BEGA framework. However, in our previous work [18], we demonstrated how this gene-wise approach does not accurately reflect population diversity in problems where the order or sequence of the genes is more important than their positions such as the TSP. Although adaptive GAs have developed into highly effective algorithms for maintaining diversity within a single population, the centralised nature of adaptive GAs prevents the GA from maintaining concurrent search focuses in different areas of the solution space. By deploying adaptive GAs in parallel populations, each population can balance its own exploration and exploitation. Through controlled migration, a distributed adaptive GA can encourage each population to search in different areas of the solution space thereby increasing the likelihood of the global optimum being discovered.

III. Adaptive Parallel LCSB-AGA

The LCSB-AGA [18] is an adaptive GA that maintains population diversity according to the similarity in gene sequences of each individual. The proposed Adaptive Parallel LCSB-AGA distributes multiple LCSB-AGA populations across compute nodes in a master-slave relationship (Fig. 1) where the master node distributes work to slave nodes and each slave node is responsible for maintaining its LCSB-AGA population until the master node initiates a migration event. This ensures

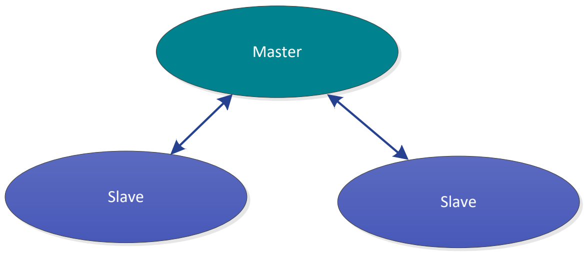


Fig. 1. Synchronous Master-Slave architecture where each slave processor is responsible for the evolution of a population until the master processor initiates a migration.

that each slave process searches the problem space without bias from the others. These slave processes follow the same evolution process as the LCSB-AGA where the population is split into exploration and exploitation sub-populations [18]. The exploitation sub-population is responsible for local search where new individuals are derived from parents selected by a tournament where the tournament size is dependent on the degree of similarity between individuals of the population. The exploration sub-population focuses on global search by having an adaptive mutation rate that is dependant on the degree to which the population has converged. This allows for the population to maintain a healthy level of diversity to allow it to continue searching the global solution space while improving upon its known solutions. Each slave process is responsible for monitoring its population diversity through calculating the average LCS distance and is shown in Eq. 1.

$$D^{\mu}(\mathcal{P}) = \frac{\sum_{i=2}^{|\mathcal{P}|} N - LCS(\mathcal{P}_1, \mathcal{P}_i)}{|\mathcal{P}| - 1} \tag{1}$$

The diversity $D^{\mu}(\mathcal{P})$ is the LCS distance of the fittest individual $(\mathcal{P}_1)$ and the rest of the population. Where the LCS length shows the sequence-wise similarity between two genotypes, the LCS distance describes the difference between the genotype length (N) and the LCS length. A high LCS distance means a low similarity between genotypes and contributes to a higher $D^{\mu}(\mathcal{P})$. This is used to calculate when migration is to occur and how much migration is necessary to maintain a healthy level of population diversity.

Convergence Criteria. The first step for adaptive migration to occur is to calculate the convergence criteria. This is the number of generations where the given population has not experienced an improvement in its fitness. This is demonstrated in Eq. 2 where the convergence criteria is denoted by C.

$$C = N \cdot D^{\mu}(\mathcal{P}) \tag{2}$$

The convergence criteria is the product of the genotype length (N) and D^{μ} thus considers both the size of the problem and the population where a larger problem or a larger population will allow for more generations before migration is initiated. After each generation, if the slave process has not improved its fitness for C generations, the master process initiates a migration between each population. When each population is able to maintain a healthy level of diversity, migration occurs less frequently. As the rate of improvement decreases, migration occurs more frequently.

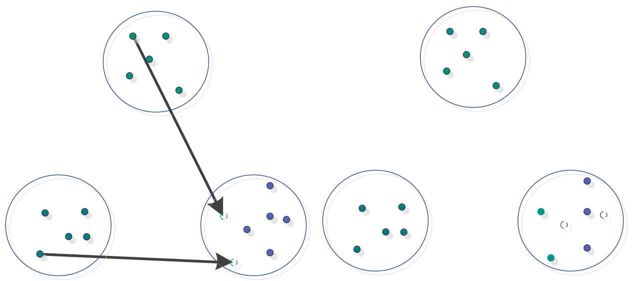

(a) Top-k selection (b) Removal of worst-k

Fig. 2. Top-k individuals that improves a population's diversity and fitness are selected from the other populations (a). When these individuals migrate to the new population, the k individuals that contribute the least to a population's diversity and fitness are removed (b).

Adaptive Migration. Once the convergence criteria has been met, the migration process consists of a number of steps in order to determine how many individuals are to migrate to the given population as well as which individuals are selected. These steps are outlined as the following:

Step 1. Calculate the number individuals to migrate to the selected population. The average LCS distance of the fittest individual of the population $\mathcal{P}_1$ and the remaining individuals of the population ($\mathcal{P}_i$) is normalised against genotype length (N). This ensures that $0 \leq D^{\mu}(\mathcal{P}) \leq 1$ and is used to calculate the number of individuals for migration. This is shown in Eq. 3.

$$k_{migration} = \frac{D^{\mu}(\mathcal{P})}{N} \cdot k_{max} \quad (3)$$

k_{max} is the maximum number of individuals for migration and must be $0 \leq k_{max} \leq N$. A high k_{max} value leads to a high number of individuals that can migrate between populations. This can lead to a faster rate of convergence where all populations begin to converge to similar areas of the solution space as well as increased computation costs for selecting the top-k individuals. Empirical tests indicate that k_{max} value of $|\mathcal{P}|/8$ provides a good balance between maintaining population diversity and computational costs.

Step 2. Given $k_{migration}$, the LCSB-AGA selects the top-k individuals from the other populations that contribute the greatest to the current population's fitness and diversity. The top-k individuals are selected through the weighted sum dominance test. The dominance (Eq. 4) maximises the LCS distance and the fitness of individuals selected for the top-k set compared to the individuals of the population.

$$Q(\mathcal{P}, i) = \sum_{j=1}^{|\mathcal{P}|} w_f \cdot \frac{f(i)}{f(\mathcal{P}_1)} + w_d \cdot \frac{N - LCS(i,j)}{N} \quad (4)$$

Given a population of $\mathcal{P}$, the individual i that maximises $Q(\mathcal{P}, i)$ is selected as part of the top-k where fitness (w_f) and distance (w_d) weights are applied to the individual's fitness ($f(i)$) and LCS distance respectively ($N - LCS(i,j)$). $f(i)$ is normalised against the fitness of the fittest individual of the population ($\mathcal{P}_1$) and the LCS distance is normalised to N.

TABLE I
PARAMETER SETTINGS FOR EACH GA FRAMEWORK FOR POPULATION SIZE ($|\mathcal{P}|$), CROSSOVER (P_c) AND MUTATION (P_m) RATES.

Framework	$\|\mathcal{P}\|$	P_c	P_m
GA	50	0.95	0.05
ACR.	50	Adapt.	Adapt.
ReduceGap	400	0.25	0.75
LCSB-AGA	50	Adapt.	Adapt.
Parallel LCSB-AGA	50	Adapt.	Adapt.
Parallel TK LCSB-AGA	50	Adapt.	Adapt.

Step 3. Replace the bottom-k individuals from the given population with the individuals in the top-k set. This is similar to the previous step but removes the k individuals that minimises $F(\mathcal{P}, i)$.

Migration is permitted globally where individuals from any population can migrate to a given population. This ensures that while each population is independently adjusting its own search parameters, good quality solutions can be disseminated to each population without negatively affecting their diversity. In doing so, migration can occur without the search process of one population heavily influencing another.

IV. EXPERIMENTS

A simple GA with parallel populations was implemented as a baseline for the experiments. The parameters, selection, crossover and mutation operators were selected for their optimal performance [19]. Each population was allowed to evolve for G generations before the best performing individuals were broadcast globally. A parallel implementation of ACROMUSE with a similar migration strategy was also implemented due to the GA's performance and adaptive properties. ACROMUSE was selected for its gene-wise approach to diversity maintenance and its ability to find good quality solutions in ordered problems. The ReduceGap [11] framework was also selected as a benchmark algorithm due to its distributed design and its performance in solving the open-shop scheduling problem (OSSP). The ReduceGap framework was implemented according to the authors' descriptions with the recommended parameters. Parameters for each GA are outlined in Table I.

LCSB-AGA with a single population provides a baseline for comparing the changes in performance when implementing a Parallel LCSB-AGA. A second Adaptive Parallel LCSB-AGA (AP-LCSB-AGA) implements the convergence criteria (Eq. 2) and adaptive migration (Eqs. 3-4) to investigate adaptive migration strategies.

A range of ordered optimisation problems were selected for their problem constraints. The Travelling Salesman Problem (TSP) represents the simplest problem where the only constraint is the unique ordering of genes in its sequence. The Capacitated Vehicle Routing Problem (CVRP) is a generalised TSP where there are N deliveries to be made. While similar to the TSP, in the CVRP each delivery has a weight and can be delivered from one of K trucks with their own weight capacity. The optimum solution is the K routes that minimises the total distance to complete all N deliveries from a given depot.

The Job-shop Scheduling Problem (JSSP) is another ordered combinatorial problem where there are j jobs that require m

TABLE II
TRAVELLING SALESMAN PROBLEM RESULTS

Instance	Algorithm	Solution Quality			Avg.
		Mean	St. Dev.	Best	LCS Dist.
berlin52	P-GA	8,707	457.9	7,975	16.0
n: 52	P-ACROMUSE	7,542	0.0	7,542	14.6
Run time: 00:07:00	ReduceGap	21,025	390.2	20,583	12.1
Opt. : 7,542	LCSB-AGA	7,553	33.8	7,542	29.8
	P-LCSB-AGA	7,542	0.0	7,542	33.9
	AP-LCSB-AGA	7,542	0.0	7,542	37.5
pr107	P-GA	52,607	8,077.6	44,303	27.1
n: 107	P-ACROMUSE	44,303	0.0	44,303	31.2
Run time: 00:08:00	ReduceGap	125,393	883.4	123,756	26.8
Opt. : 44,303	LCSB-AGA	44,303	0.0	44,303	57.2
	P-LCSB-AGA	44,303	0.0	44,303	81.6
	AP-LCSB-AGA	44,303	0.0	44,303	81.6
pr152	P-GA	144,976	15,270.5	121,151	23.3
n: 152	P-ACROMUSE	104,998	4,320.1	98,899	23.9
Run time: 00:10:00	ReduceGap	227,318	2,003.3	223,400	39.1
Opt. : 73,682	LCSB-AGA	81,355	1,082.4	79,556	70.8
	P-LCSB-AGA	73,770	279.1	73,682	119.9
	AP-LCSB-AGA	73,682	0.0	73,682	118.6
rat195	P-GA	7,124	185.1	6,870	33.4
n: 195	P-ACROMUSE	5,200	305.9	4,783	31.6
Run time: 00:15:10	ReduceGap	10,509	90.3	10,317	51.1
Opt. : 2,323	LCSB-AGA	4,871	41.6	4,818	93.2
	P-LCSB-AGA	4,168	359.6	3,770	153.4
	AP-LCSB-AGA	3,991	28.5	3,955	152.9
pr264	P-GA	67,859	5,998.2	60,951	48.4
n: 264	P-ACROMUSE	64,402	6,464.8	56,286	41.3
Run time: 00:22:00	ReduceGap	92,367	1,054.6	90,155	70.4
Opt. : 49,135	LCSB-AGA	66,042	873.2	65,223	107.3
	P-LCSB-AGA	61,441	2,832.5	58,303	214.1
	AP-LCSB-AGA	54,618	546.6	53,828	209.1

TABLE III
CAPACITATED VEHICLE ROUTING PROBLEM RESULTS

Instance	Algorithm	Solution Quality			Avg.
		Mean	St. Dev.	Best	LCS Dist.
A-n32-k5	P-GA	785	3.6	784	8.1
N: 32	P-ACROMUSE	784	0.0	784	16.4
K: 5	ReduceGap	1,265	21.3	1,216	11.2
Run time: 00:00:11	LCSB-AGA	784	0.0	784	12.4
Opt. : 784.0	P-LCSB-AGA	784	0.0	784	20.7
	AP-LCSB-AGA	1,174	0.0	784	20.5
A-n65-k9	P-GA	1,181	14.4	1,174	8.4
N: 65	P-ACROMUSE	1,174	0.0	1,174	36.6
K: 9	ReduceGap	2,933	28.6	2,874	15.2
Run time: 00:00:48	LCSB-AGA	1,174	0.0	1,174	28.4
Opt. : 1,763	P-LCSB-AGA	1,174	0.0	1,174	47.2
	AP-LCSB-AGA	1,174	1.7	1,174	47.1
A-n80-k10	P-GA	2,551	70.8	2,387	6.6
N: 80	P-ACROMUSE	2,439	141.0	2,130	22.7
K: 10	ReduceGap	3,554	47.6	3,453	5.8
Run time: 00:01:12	LCSB-AGA	1,897	81.6	1,812	35.6
Opt. : 1,763	P-LCSB-AGA	1,773	22.9	1,763	59.7
	AP-LCSB-AGA	1,774	31.5	1,763	59.1
P-n101-k4	P-GA	1,298	71.8	1,164	10.4
N: 101	P-ACROMUSE	1,209	92.3	1,037	29.0
K: 4	ReduceGap	2,759	23.0	2,715	7.5
Run time: 00:01:57	LCSB-AGA	919	47.5	833	45.7
Opt. : 681	P-LCSB-AGA	921	32.7	876	76.7
	AP-LCSB-AGA	899	25.6	819	76.1
F-n135-k7	P-GA	2,023	140.5	1,790	11.0
N: 135	P-ACROMUSE	1,752	74.7	1,573	39.7
K: 7	ReduceGap	5,310	83.4	5,124	10.3
Run time: 00:03:29	LCSB-AGA	1,467	45.1	1,358	49.5
Opt. : 1,162	P-LCSB-AGA	1,370	53.7	1,287	103.3
	AP-LCSB-AGA	1,333	52.4	1,253	103.6

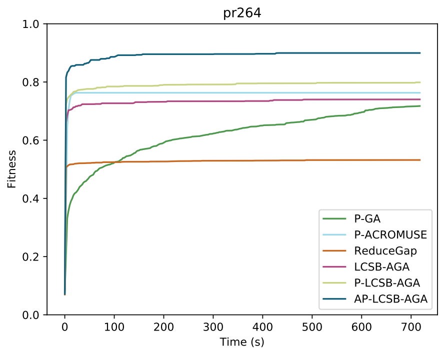
machines to complete. Furthermore, each machine is only able to process one job at a time and each process for a job requires a specific order. These constraints contribute to an increased complexity in solving the JSSP compared to the TSP and CVRP.

A range of instances have been selected for each problem according to their spread in the problem's size. For each problem instance, the GAs are run for the same run time duration with the best solution at the end of the run being selected. The run time itself is dependent on the size of the instance to allow for sufficient processing time. All GAs are tested on each instance for 50 runs in order to calculate mean and distribution of the solution quality. The z-test is used to calculate the statistical significance of the GA frameworks.

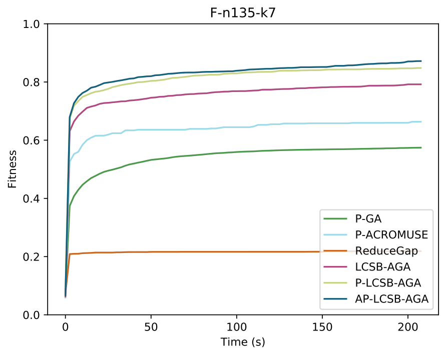
The experiment environment is a Windows Server 2016 HPC cluster consisting of a single master node and 8 compute nodes running on Intel i7 4770K each with 16GB of main memory. Each distributed GA was run with 8 populations, one on each compute node.

V. RESULTS

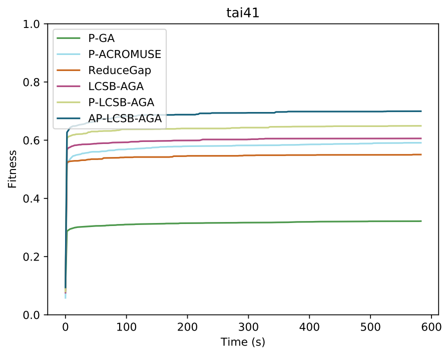
The TSP and CVRP results in Tables II-III demonstrate the performance of each GA with a range of instances where there are similar constraints. All GAs perform well when the N is relatively small with most finding the global optimum. In particular, Parallel GA (P-GA), Parallel ACROMUSE (P-ACROMUSE) and the LCSB-AGA variants are consistently able to discover the global optimum when $N \leq 107$. However, as the problem size increases, the Parallel LCSB-AGA (P-LCSB-AGA) and Adaptive Parallel LCSB-AGA (AP-LCSB-AGA) are able to outperform the other GAs. In particular, the AP-LCSB-AGA is able to maintain a higher level of consistency than the P-LCSB-AGA. The population diversity is shown as the average LCS distance where a higher number indicates a higher level of diversity. This is measured as the diversity of the global population and can be used to indicate whether the different populations were searching in different areas of the solution space. One exception to these observations is ReduceGap which was designed for the OSSP and is hindered by its crossover and mutation operators when applied to the TSP and CVRP. However, due to its large population sizes, ReduceGap is able to maintain a higher level of population diversity in comparison to P-ACROMUSE and P-GA. The effects of introducing further constraints with the JSSP can be seen in Table IV. One of the challenges with maintaining diversity for JSSP instances is the manner in which the problem is encoded. As each gene value appears m times in the sequence, many duplicates can appear. As this increases the likelihood of individuals having the same gene values in a given position, this can result in a lower level of diversity when using a gene-wise strategy for diversity maintenance. With a sequence-wise approach to diversity maintenance, the LCSB-AGA benefits from implementing multiple populations. LCSB-AGA with a single population maintains a sequence-wise diversity of 15-25% of N while the multi-population variants (P-LCSB-AGA and AP-LCSB-AGA) maintain between 45-60%. This suggests that each population is independently searching the solution space and, when applied to the JSSP, AP-LCSB-AGA maintains a higher



(a) TSP Pr264



(b) CVRP F-n135-27



(c) JSSP Tai41

Fig. 3. Average solution quality normalised against the optimal solution (fitness) for TSP (Pr264), CVRP (F-n135-k7) and JSSP (Tai41)

TABLE IV
Job-shop Scheduling Problem Results

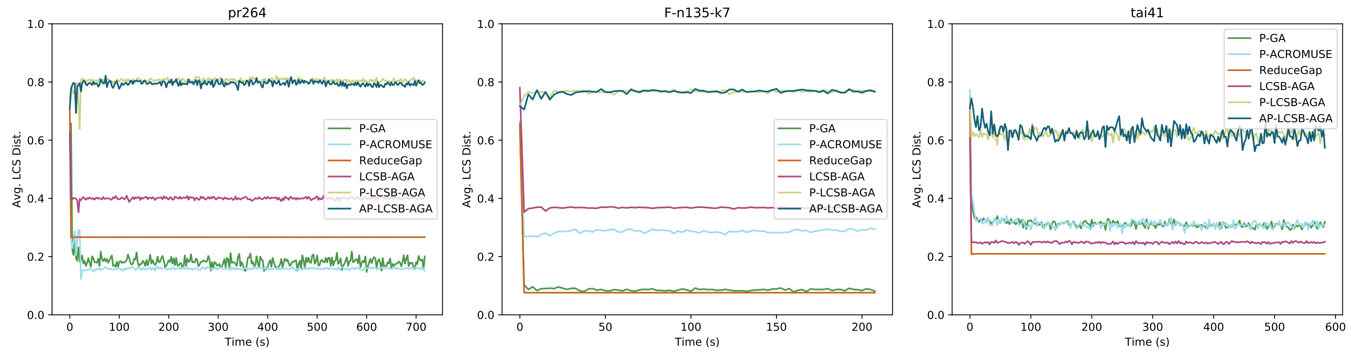
Instance	Algorithm	Solution Quality			Avg. LCS Dist.
		Mean	St. Dev.	Best	
tai01	P-GA	1,231	0.0	1,231	56.3
N: 255	P-ACROMUSE	1,231	0.0	1,231	95.5
m: 15	ReduceGap	1,231	0.0	1,231	40.8
j: 15	LCSB-AGA	1,231	0.0	1,231	37.2
Run time: 00:00:31	P-LCSB-AGA	1,231	0.0	1,231	113.0
Opt. : 1,231	AP-LCSB-AGA	1,231	0.0	1,231	176.0
tai05	P-GA	1,254	31.7	1,224	58.4
N: 225	P-ACROMUSE	1,224	0.0	1,224	116.5
m: 15	ReduceGap	1,224	0.0	1,224	40.7
j: 15	LCSB-AGA	1,224	0.0	1,224	49.1
Run time: 00:02:13	P-LCSB-AGA	1,224	0.0	1,224	153.6
Opt. : 1,224	AP-LCSB-AGA	1,224	0.0	1,224	163.2
tai11	P-GA	1,476	34.7	1,413	89.6
N: 300	P-ACROMUSE	1,457	41.4	1,401	82.6
m: 15	ReduceGap	1,669	38.5	1,587	29.1
j: 20	LCSB-AGA	1,500	22.5	1,447	68.3
Run time: 00:04:28	P-LCSB-AGA	1,404	11.1	1,382	172.0
Opt. : 1,357	AP-LCSB-AGA	1,394	25.3	1,374	184.0
tai21	P-GA	3,215	99.9	3,062	114.1
N: 400	P-ACROMUSE	2,242	76.2	2,108	116.6
m: 20	ReduceGap	2,552	21.4	2,522	61.7
j: 20	LCSB-AGA	2,388	25.3	2,341	94.4
Run time: 00:06:21	P-LCSB-AGA	2,153	27.3	2,104	201.4
Opt. : 1,642	AP-LCSB-AGA	1,888	46.2	1,835	219.0
tai41	P-GA	5,866	156.5	5,589	176.4
N: 600	P-ACROMUSE	3,170	118.6	2,996	187.9
m: 20	ReduceGap	3,448	38.6	3,394	125.7
j: 30	LCSB-AGA	3,125	38.9	3,035	148.9
Run time: 00:11:59	P-LCSB-AGA	2,927	32.5	2,856	369.8
Opt. : 1,906	AP-LCSB-AGA	2,697	78.6	2,552	391.2

level of diversity in the global population. This further suggests that the adaptive migration and convergence criteria is ensuring that migration is not influencing the populations of the AP-LCSB-AGA to search in the same area of the solution space. Further insight into the characteristics of each GA can be gained by investigating how each GA evolves its solution with each problem.

Fig. 3 illustrates the performance of each GA for the largest instance in each problem. For comparison purposes, the fitness for each GA is normalised against the known optimum. P-GA manages a steady improvement in solution quality over time. This suggests that each migration successfully disseminated good solutions so that the populations are more successful in finding better solutions. With the TSP instance Pr264 (Fig. 3a), while the other GAs are much faster at finding good quality solutions, given enough time, P-GA is likely to find a solution comparable to the LCSB-AGA. This supports the argument where adaptive parameter controls can contribute to the a GA's ability to find good solutions in a similar fashion to how parallel populations can. With adaptive parameter controls implemented along side parallel populations, P-ACROMUSE, P-LCSB-AGA and AP-LCSB-AGA are able to outperform both LCSB-AGA and P-GA. This is further demonstrated when more constraints are introduced with the CVRP (Fig. 3b) and The JSSP (Fig. 3c).

Table 4 highlights how each GA maintains diversity. For comparison purposes, the average LCS distance has been normalised against the geneotype length. Each GA initialises with a relatively high level of diversity but the GAs without diversity maintenance mechanisms quickly begin to converge. As P-GA is only able to maintain diversity through migration and mutation, it is unable to maintain a stable population diversity over 20% of its genotype length for the TSP and CVRP instances. While P-ACROMUSE is designed to maintain gene-wise diversity, the exploitation process can result in a high number of individuals with similar sequences. Despite this, P-ACROMUSE manages to achieve good results which suggests that any implementation of diversity management can improve a GA's performance. Due to its large population and no adaptive diversity maintenance strategies, ReduceGap maintains a consistent but low level of diversity. LCSB-AGA variants have a higher degree of sequence-wise diversity that is subject to large deviations. This demonstrates the adaptive diversity maintenance strategies where the GA injects higher diversity when a drop in diversity is identified. The spikes are likely due to the GA introducing too much diversity or too much convergence and adjusting its strategy accordingly. This is particularly noticeable in Tai41 (Fig. 4c) and demonstrates the difficulty in maintaining population diversity as the problem size and complexity increases.

In order to quantify the benefits of introducing sequence-wise adaptive diversity maintenance and parallel populations, the results of each experiment have been summarised with a z-test highlighting the significance of the improvements in performance (Table V). Each GA is compared to the P-



(a) TSP Pr264 (b) CVRP F-n135-27 (c) JSSP Tai41

Fig. 4. Average LCS length normalised against the genotype length (N) for TSP (Pr264), CVRP (F-n135-k7) and JSSP (Tai41)

TABLE V
STATISTICAL SIGNIFICANCE OF EXPERIMENT RESULTS.

Instance	Parallel LCSB-AGA				Adaptive Parallel LCSB-AGA				
	P. GA	P. ACR.	RG	LCSB	P. GA	P. ACR.	RG	LCSB	P. LCSB
berlin52	++	*	++	+	++	*	++	+	*
pr107	++	*	++	*	++	*	++	*	*
pr152	++	++	++	*	++	++	++	*	+
rat195	++	++	++	++	++	++	++	++	++
pr264	++	++	++	++	++	++	++	++	++
A-n32-k5	*	*	++	*	*	*	++	*	*
A-n65-k9	++	*	++	*	++	*	++	*	*
A-n80-k10	++	++	++	++	++	++	++	++	*
P-n101-k4	++	++	++	++	++	++	++	++	++
F-n135-k7	++	++	++	++	++	++	++	++	++
tai01	*	*	*	*	*	*	*	*	*
tai05	++	*	*	*	*	++	*	*	*
tai11	++	++	++	++	++	++	++	++	+
tai21	++	++	++	++	++	++	++	++	++
tai41	++	++	++	++	++	++	++	++	++

LCSB-AGA and AP-LCSB-AGA where significant and very significant improvements are denoted by a "+" and "++" respectively. Instances where there were no statistically significant differences is denoted by a "*". The results from the z-tests indicate that while the GAs perform similarly for smaller instances, combining both adaptive diversity maintenance and parallel populations significantly improve the results on larger and more complex problems. Furthermore, by introducing adaptive migration and convergence criteria, AP-LCSB-AGA significantly outperforms P-LCSB-AGA with the larger problems. Given that all GAs were given the same run-time duration, AP-LCSB-AGA provides a robust solution as a distributed adaptive GA.

VI. CONCLUSION

In this paper we have proposed a distributed LCSB-AGA framework that is able to independently maintain the diversity of each population. By implementing adaptive migration controls, we have demonstrated how strategies from adaptive GAs can be used to enable a parallel GA to determine its own migration size and which individuals to select for migration. In doing so, the GA can encourage each population to search in different areas of the solution space.

Future work identified during this study include finding the optimal process for determining the diversity of a population during migration. While this study implemented a weighted summation of a dominance test, an investigation on identifying the optimal diversity measure may lead to improved efficiency and improvements in solution quality.

REFERENCES

[1] I. A. Chaudhry, A. A. Khan, A research survey: review of flexible job shop scheduling techniques, ITOR 23 (3) (2016) 551–591.
[2] E. K. Burke, S. Gustafson, G. Kendall, Diversity in genetic programming: An analysis of measures and correlation with fitness, IEEE Trans. Evol. Comput. 8 (1) (2004) 47–62.
[3] M. Črepinšek, S.-H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: A survey, ACM CSUR 45 (3) (2013) 35.
[4] Y.-J. Gong, W.-N. Chen, Z.-H. Zhan, J. Zhang, Y. Li, Q. Zhang, J.-J. Li, Distributed evolutionary algorithms and their models: A survey of the state-of-the-art, Applied Soft Computing 34 (2015) 286–300.
[5] A. J. Chipperfield, P. Fleming, Parallel genetic algorithms: A survey.
[6] M. Dubreuil, C. Gagné, M. Parizeau, Analysis of a master-slave architecture for distributed evolutionary computations, IEEE T. Syst. Man. Cy-C 36 (1) (2006) 229–235.
[7] J. I. Hidalgo, J. Lanchares, F. Fernández de Vega, D. Lombraña, Is the island model fault tolerant?, in: GECCO, 2007, pp. 2737–2744.
[8] A. A. R. Hosseinabadi, J. Vahidi, B. Saemi, A. K. Sangaiah, M. Elhoseny, Extended genetic algorithm for solving open-shop scheduling problem, Soft computing 23 (13) (2019) 5099–5116.
[9] U. Tosun, T. Dokeroglu, A. Cosar, A robust island parallel genetic algorithm for the quadratic assignment problem, IJPR 51 (14).
[10] Y. Y. Liu, S. Wang, A scalable parallel genetic algorithm for the generalized assignment problem, Parallel computing 46 (2015) 98–119.
[11] S. B. Ghosn, F. Drouby, H. M. Harmanani, A parallel genetic algorithm for the open-shop scheduling problem using deterministic and random moves, Int. J. Artif. Intell 14 (1) (2016) 130–144.
[12] A. Aleti, I. Moser, A systematic literature review of adaptive parameter control methods for evolutionary algorithms, ACM CSUR 49 (3).
[13] G. Karafotias, M. Hoogendoorn, Á. E. Eiben, Parameter control in evolutionary algorithms: Trends and challenges, CEC 19 (2).
[14] D. Whitley, T. Starkweather, Genitor ii: A distributed genetic algorithm, J. Exp. Theor. Artif. Intell. 2 (3) (1990) 189–214.
[15] J. Maturana, F. Saubion, A compass to guide genetic algorithms, in: PPSN, 2008, pp. 256–265.
[16] H. Zhang, Y. Liu, J. Zhou, Balanced-evolution genetic algorithm for combinatorial optimization problems: the general outline and implementation of balanced-evolution strategy based on linear diversity index, Natural Computing 17 (3) (2018) 611–639.
[17] B. Mc Ginley, J. Maher, C. O'Riordan, F. Morgan, Maintaining healthy population diversity using adaptive crossover, mutation, and selection, IEEE Trans. Evolutionary Computation 15 (5) (2011) 692–714.
[18] R. Ohira, M. S. Islam, J. Jo, B. Stantic, Lcs based diversity maintenance in adaptive genetic algorithms, in: AusDM, 2018, pp. 56–68.
[19] R. Ohira, M. S. Islam, J. Jo, B. Stantic, Amga: An adaptive and modular genetic algorithm for the traveling salesman problem, in: ISDA, 2018, pp. 1096–1109.