Selecting Genetic Operators to Maximise Preference Satisfaction in a Workforce Scheduling and Routing Problem

Haneen Algethami[1], Dario Landa-Silva[1] and Anna Martínez-Gavara[2]
[1]*School of Computer Science, ASAP Research Group, The University of Nottingham, Nottingham, U.K.*
[2]*Estadística y Investigación Operativa, Universidad de Valencia, Valencia, Spain*

Keywords: Genetic Operators, Constraints Satisfaction, Scheduling and Routing Problem, Home Health Care.

Abstract: The Workforce Scheduling and Routing Problem (WSRP) is a combinatorial optimisation problem that involves scheduling and routing of workforce. Tackling this type of problem often requires handling a considerable number of requirements, including customers and workers preferences while minimising both operational costs and travelling distance. This study seeks to determine effective combinations of genetic operators combined with heuristics that help to find good solutions for this constrained combinatorial optimisation problem. In particular, it aims to identify the best set of operators that help to maximise customers and workers preferences satisfaction. This paper advances the understanding of how to effectively employ different operators within two variants of genetic algorithms to tackle WSRPs. To tackle infeasibility, an initialisation heuristic is used to generate a conflict-free initial plan and a repair heuristic is used to ensure the satisfaction of constraints. Experiments are conducted using three sets of real-world Home Health Care (HHC) planning problem instances.

1 INTRODUCTION

The workforce scheduling and routing problem (WSRP) involves scheduling and routing of workforce to visit customers at different locations in order to complete a set of tasks or activities. The problem arises in real-world scenarios, such as home health care, security guard routing and rostering, maintenance personnel scheduling among other worker allocation problems (Castillo-Salazar et al., 2016).

The WRSP is a combination of two combinatorial optimisation problems, personnel scheduling and routing, which are known to be NP-hard problems (Lenstra and Kan, 1981). The scheduling aspect allocates workforce to customers in order to fulfil work demands as well as satisfying their preferences. The routing aspect requires generating routes for workers to visit customers across various locations and within given time windows. Researchers have reported that real-world instances of the WSRP are large and difficult to solve (Mısır et al., 2015; Castillo-Salazar et al., 2016). Hence, there is a need to develop efficient algorithms to solve this type of problem.

Preliminary work evaluated a set of genetic operators within a steady-state genetic algorithm applied to a few instances of a real-world home health care (HHC) problem (Algethami and Landa-Silva, 2015). That work produced evidence that some operators obtain better results than others when used within the steady-state genetic algorithm for WSRP scenarios. The present paper conducts a more comprehensive study in order to achieve a deeper understanding of the behaviour and performance of the various genetic operators when applied to the WSRP.

The aim in this paper is to identify the best combination of genetic operators for each WSRP instance, in order to maximise the satisfaction of customers and workers preference constraints. Twelve genetic operators are considered in different combinations. The two genetic algorithms (GA) applied in this study are a steady state GA and a generational GA.

2 RELATED WORK

The routing component of the WSRP is related to variants of the classical vehicle routing problem (VRP) and in particular to the vehicle routing problem with time windows (VRPTW) (Toth and Vigo, 2014). Many GA applications have been used to tackle VRP including hybrid approaches incorporating heuristic methods and problem-specific operators to avoid pre-

Algethami H., Landa-Silva D. and MartÃnez-Gavara A.
Selecting Genetic Operators to Maximise Preference Satisfaction in a Workforce Scheduling and Routing Problem.
DOI: 10.5220/0006203304160423
In *Proceedings of the 6th International Conference on Operations Research and Enterprise Systems (ICORES 2017)*, pages 416-423
ISBN: 978-989-758-218-9

mature convergence of the GA (Prins, 2004; Chang and Chen, 2007). In addition, the study by (Prins, 2004) suggested the best genetic components for an efficient GA to tackle VRP problems. According to that study, order crossover (OX) is the most suitable operator for VRP-like problems.

Genetic algorithms (GAs) have been effective in providing good solutions relatively quickly, particularly when addressing real-world scheduling problems (Kotecha et al., 2004; Aickelin and Dowsland, 2004). It has been argued that this success is a result of the GA's capability to solve different segments of a problem simultaneously (Rothlauf, 2003).

A number of studies have applied GAs to real-world problems where scheduling and routing are combined. Examples include (Cowling et al., 2006; Mutingi and Mbohwa, 2014). In those works, the focus has been on algorithm design in order to obtain good solutions. However, well-known operators and repair heuristics were used to deal with infeasibility issues. Far too little attention has been given to introducing new genetic operators to reduce the overall cost. To date, the impact of selecting compatible operators for tackling WSRP instances has not yet been investigated.

The focus of this paper is not to produce the most competitive genetic algorithm, but to advance the understanding of how different combinations of genetic operators perform when tackling preference constraints in instances of the WSRP. The problem instances used in this study were also tackled in (Laesanklang and Landa-Silva, 2016; Pinheiro et al., 2016). This work seeks to identify effective combinations of genetic operators for tackling preference constraints in WSRP instances to then inform the design of competitive GAs to tackle this difficult problem.

3 PROBLEM DESCRIPTION

A WSRP solution is a daily plan of visits, i.e. a set of workers $W = \{w_1, w_2, \ldots, w_{|W|}\}$ assigned to perform a set of tasks $T = \{t_1, t_2, \ldots, t_{|T|}\}$ for customers at different locations. The assignment of a worker to travel to a customer location in order to perform a task is called a visit. Thus, $x^w_{i,j}$ is a binary decision variable that indicates if a path connects two nodes (visit i and visit j) or not. The assignment $x^w_{i,j} = 1$ means that worker w travels from visit i to visit j, thus w makes both visits. For visit j, if $x^w_{i,j} = 1$ then $y_j \leq r_j - 1$ where r_j is the *number of workers* required for visit j and y_j is an integer decision variable indicating the number of unsatisfied assignments, hence $\sum_{w \in W} \sum_{i \in T} \sum_{j \in T} x^w_{i,j} + y_j = r_j$.

This paper tackles a home health care (HHC) planning problem, in which workers are nurses, doctors, health carers, etc., and customers are patients receiving health care at their home. Several features have been identified as important in solutions to HHC scenarios, such as distance travelled and customers' and workers' requirements and preferences (Mankowska et al., 2014). A good quality plan for an HHC planning problem should have a low operational cost as well as assigning workforce. Thus, a solution requires all tasks to be assigned while satisfying some requirements. That is, assigning tasks according to workers' skills and avoiding time conflicts in respect to workers' time and area availability. A time conflict occurs when a worker is assigned to visits overlapping in time. Additional preferences include workers preferring to work in certain geographical areas, customers requiring workers with special skills or preferring certain workers to perform a task.

Table 1 lists WSRP objectives and constraints considered here. See (Laesanklang and Landa-Silva, 2016) for details of the MIP model of this WSRP. Note that in (Laesanklang and Landa-Silva, 2016), unassigned visits constraint is considered as a soft constraint. However, here this is a hard constraint, hence all visits must be assigned. Additionally, in this paper, time-conflict constraint is introduced, while the study by (Laesanklang and Landa-Silva, 2016) avoided conflicts. The decomposition method divided a problem into sub-problems, then available workers were updated so that no conflicting assignments exists.

A solution S is a set of assignments to workers in order to make visits. The objective function includes the *operational cost* and the *penalty cost*. The operational cost is the accumulated cost $d_{i,j} + p^w_j$, where $d_{i,j}$ is the distance travelled between visit i to visit j and p^w_j is the cost of assigning worker w to visit j, i.e. wages plus journey costs for all workers, as calculated by the service provider in our HHC scenarios.

The penalty cost is the accumulated penalty for the violations on constraints. An assignment can be written as a tuple $x^w_{i,j}, y_j, a^w_j, \psi^w_j, \theta^w_j, \tau^w_j$. Where, a^w_j is the arrival time of a worker w to the location of a visit j. The assignment is also composed of binary decision variables indicating an assignment of worker w to visit j with violations on area availability (ψ^w_j), time availability (θ^w_j) and conflicting assignments (τ^w_j).

The non-satisfaction of preferences is also included in the penalty cost. There are three types of preferences including preferred worker-customer pairing, worker's preferred region and customer's preferred skills. There is a degree of satisfaction for these preferences when assigning a worker w to a task j and

Table 1: Objectives and constraints in the WSRP.

Objectives	Hard constraints	Soft constraints
Minimise the operational cost	Assign all visits	Respect workers area availability
Minimise the penalty cost	Respect visit time (No time-conflicts)	Respect workers time availability
	Respect max working time per week	Assign preferred workers to visits
	Respect min working time per week	Assign preferred workers with a specific skill
	Assign qualified workforce	Assign workers to preferred areas

is given by ρ_j^w which has a value that ranges between $[0,3]$. For each assignment, the satisfaction value for each preference ranges between $[0,1]$, from not satisfied to satisfied. The satisfaction level is reverted to a penalty by subtracting it from the full satisfaction score, which is $3r_j$ for a visit j.

$$f(S) = \lambda_1 \sum_{w\in W}\sum_{i\in T}\sum_{j\in T}(d_{i,j}+p_j^w)x_{i,j}^w + \lambda_2 \sum_{w\in W}\sum_{i\in T}\sum_{j\in T}(3r_j-\rho_j^w)x_{i,j}^w + \lambda_3 \sum_{w\in W}\sum_{j\in T}(\psi_j^w+\theta_j^w) + \lambda_4 \sum_{j\in T} y_j + \lambda_5 \sum_{w\in W}\sum_{j\in T}\tau_j^w \quad (1)$$

The best solution should have: the least *operational cost* and the least *penalty cost*. A weighted sum is proposed to combine the objectives into a single scalar value (Pinheiro et al., 2016; Algethami et al., 2016). The objective function is written as in equation (1), where weights $\lambda_1,\ldots,\lambda_5$ are defined to establish priority between objectives (more about the weights used here later in the paper).

4 ALGORITHMS AND OPERATORS

This paper investigates the behaviour of various genetic operators when tackling instances of the WSRP by analysing their performance within relatively straightforward implementations of two variations of GAs. A simple solution representation allows direct implementation of genetic operators on the genotype (Rothlauf, 2003). Thus, a *direct representation scheme* is used for chromosome encoding. A vector of integers of length equal to the number of visits, $|T|$, represents a one-day plan. Hence, all visits are assigned. Indexes of the chromosome correspond to the set of tasks T, for example the i^{th} gene in the chromosome means the corresponding visit with index $i \in T$. In order to increase the possibility of obtaining a feasible initial plan, indexes in the chromosome are associated to visits in non-decreasing order of visit start time. In this way, index 1 is for the visit with the earliest start time and index $|T|$ is for the visit with the latest start time. For each visit in the vector, a worker w is selected at random from W. A worker w may undertake more than one visit, and some workers may not be utilised as a part of a particular one-day plan.

4.1 Genetic Algorithms (GA)

Two GAs are implemented in this study: a steady-state genetic algorithm (SSGA) and a generational genetic algorithm (GenGA) (Vavak and Fogarty, 1996). Such relatively simple algorithms were selected in order to analyse the emergent behaviour and performance of the operators on a straightforward GA implementation.

Initially, a *time conflict reduction* (TCR) operator is applied to each individual in the initial population in order to reassign visits and reduce the number of time conflicts. After that, the evolutionary process for each GA is executed as follows. For the *SSGA*, there is only one population P of size M during the whole evolutionary process, where parents are selected and the offspring is inserted; thus, no generations required. Two parents i, j are selected by tournament selection from the parent lists L_1 and L_2, each of size $M/2$. To create a parent list, six different individuals are selected at random from P and split into two groups of three; the best individual of one group is added to L_1 and the best individual of the other group is added to L_2. This process is repeated until the two lists of parents are complete. Then, for i, j from $1,\ldots,M/2$, parent i in L_1 and parent j in L_2 are combined through crossover, producing two offspring. The next step is to apply mutation operator. A mutation operator is applied with some probability to the generated offspring. That is, if the mutation is applied to an individual k, the mutated individual k' replaces k, regardless of the objective function value. The recombination plus mutation process is implemented on the two parents; the best two individuals out of the two parents and the two children are added into P so that the population size remains constant.

For the *GenGA* a new population is created at the

start of each generation. The recombination plus mutation process is repeated $M/2$ times until the new population P' is complete. At this point, individuals in P' are sorted in non-decreasing order of their fitness. The best 10% of solutions found are never removed from the population. However, the worst 10% of individuals in P' are replaced by randomly generated individuals to introduce diversity onto the population. After this, the new population replaces the old one, i.e. $P = P'$. Then, the WSR operator (described below) is applied onto infeasible individuals within the population based on hard constraints violations shown in Table 1. Finally, population P' is passed to the next generation.

4.2 Repair Operators

A *time conflict reduction* (TCR) operator works as follows. Each pair of visits, i and j, are compared to identify any time conflicts, i.e. the same worker w being assigned to the two visits at the same time. If there is a time conflict, worker w is replaced in visit j by another worker w', selected from the list of preferred workers for visit j, if that list exists, or selected at random otherwise. Because TCR is applied only once on an individual, by removing one pair of visits at a time, it cannot ensure that all time conflicts are removed, but it does reduce their number.

The *worker suitability repair* (WSR) operator seeks to improve the suitability of workers for each visit, and works as follows. For each visit in an individual, the assigned worker is checked against the skills requirements, maximum hours constraints and time conflicts (i.e. the hard constraints listed in Table 1). If the worker does not satisfy these requirements, the operator aims to find another worker who is feasible for that visit. If no such worker can be found, the operator leaves the original worker in place. Thus, the WSR operator cannot ensure that all visits have a suitable worker, but it does improve the overall assignment with respect to the constraints.

4.3 Genetic Operators

The aim of this study is to select the best configuration of crossover and mutation operators that can tackle the WSRP within the SSGA and GenGA. Twelve operators are implemented, ten well-known operators plus two cost-based operators tailored for the problem tackled here.

Ten well-known operators were chosen after a literature survey of operators applied in WSRP-related problems (Algethami and Landa-Silva, 2015). The operators selected are divided into two groups. One group has five *scheduling operators*: single-point crossover (1PX), uniform crossover (Mitchell, 1998), two-point crossover (2PX) (Hartmann, 1998), half-uniform crossover (HX) (Eshelman, 1991) and random swap mutation (RSM) (Cicirello and Cernera, 2013). The other group has five *routing operators*: order crossover (OX) (Zheng and Wang, 2003), cycle crossover (CX) (Oliver et al., 1987), partially matched crossover (PMX) (Zhu, 2000), inversion mutation (IM) (Eshelman, 1991) and scramble mutation (SM) (Cicirello and Cernera, 2013).

Algorithm 1: Cost-based uniform crossover (CBUX).

Require: parent individuals p_1 and p_2
Ensure: offspring individuals o_1 and o_2
1: $o_1, o_2 \leftarrow$ new empty individual
2: **for** $i \leftarrow 1$ **to** $|T|$ **do**
3: Let w_1^i be worker assigned to visit $i \in p_1$
4: Let w_2^i be worker assigned to visit $i \in p_2$
5: **if** w_1^i and w_2^i are both available for visit i **then**
6: $o_1 \leftarrow o_1 \cup w_1^i$
7: $o_2 \leftarrow o_2 \cup w_2^i$
8: **else**
9: **if** one of w_1^i or w_2^i, called w^i, is available for visit i **then**
10: $o_1 \leftarrow o_1 \cup w^i$
11: $o_2 \leftarrow o_2 \cup w^i$
12: **else**
13: $o_1 \leftarrow o_1 \cup w_2^i$
14: $o_2 \leftarrow o_2 \cup w_1^i$
15: **end if**
16: **end if**
17: **end for**

Two cost-based operators operators have been purposely designed to improve the satisfaction of soft constraints in the WSRP, even at the expense of having a larger total cost in the solution.

One of these operators is a *cost-based uniform crossover (CBUX)* shown in Algorithm 1. Cost-based crossovers have been applied in the literature to produce improved results by restricting mating to the feasible region only (Kotecha et al., 2004). This CBUX operator works as follows. Each position i for the two parents, corresponding to the worker assigned to visit i, is processed one at a time (line 2). The availability, in terms of time and area, of the worker assigned to visit i is examined for each parent. If both parents have an available worker in that position, their gene is copied to the corresponding offspring (lines 5–7). If only one of the parents has an available worker in that position, that gene is copied to both offspring (lines 9–11). If no parent has an available worker in that position, offspring 1 gets the gene from parent 2 and offspring 2 gets the gene from parent 1 (lines 13–14).

Algorithm 2: Cost-based mutation (CBM).

Require: individual k
1: Choose a random mutation point $i \in k$
2: Let w^i be the worker assigned to visit i
3: Let Ω^i be the list of preferred workers for i
4: **if** $w \notin \Omega^i$ **then**
5: Choose a random worker $w' \in \Omega^i$
6: Replace w with w' in k for visit i
7: **end if**

A *cost-based mutation (CBM)* is shown in Algorithm 2. This operator seeks to ensure that workers assigned to visits are among those considered as *preferred workers* for that visit. As part of the input data in the problem instances considered here, a list of preferred workers is given for each visit as defined by the patients. Then, for position i in the individual, CBM tries to assign one of the preferred workers for that visit i, only if the one already assigned is not a preferred worker (lines 4–6).

5 EXPERIMENTS AND RESULTS

Table 2: Parameter settings used in the experiments.

Parameter	Settings
Population size M	100
Crossover operators	1PX, 2PX, UX, HX, PMX, OX, CX, CBUX
Crossover rate P_c	10%, 50%, 100%
Mutation operators	RSM, IM, SM, CBM
Mutation Rate P_m	1%, 10%, 30%
Running time	5 minutes

Experiments were conducted to evaluate the performance of different operators on real-world WSRP scenarios. The GAs described in Section 4 were implemented with different algorithm configurations as stated in Table 2. Values for mutation and crossover rates are taken from previous parameter tuning experiments.

The best suitable combinations of operators, each combination is one crossover operator with one mutation operator, might be later embedded in a more efficient approach. To this end, the experimental study focused on comparing the performance of the various genetic operator combinations aimed at satisfying customers' and workers' preference constraints. Nevertheless, one of the major issues in the random direct representation is allowing infeasible individuals throughout the search process, so that the end result can have individuals with high number of constraint violations. Thus, reducing hard constraint violations, such as skills required, maximum hours requirements and time conflicts, is required to maintain feasibility in WSRP solutions. As explained in Section 4, the WSR mechanism is applied to individuals that present hard constraint violations. WSR implementation occurs in two stages of the GA: after the TCR, and then again after the mutation operator in the optimisation process.

For each GA, 32 mutation-crossover combinations with 9 different rates were applied. Thus 288×2 algorithm configurations and each one was executed 8 times, all seeded with the same initial population. The best cost solution was obtained from each set with the same amount of computation time. The implementation was in Java running on a PC with I7 four-core processor with hyper-threading enabled and 16GB of RAM.

5.1 Problem Instances

Problem instances from three UK real-world HHC scenarios are used as instances of WSRP. The instances data and weights used here (blue setting) are available at https://drive.google.com/open?id=0B2OtHr1VocuSNGVOT2VSYmp6a2M. In this study, three scenarios were used with 7 problem instances each, for a total of 21 instances. Table 3 shows the main features of each problem instance.

Scenario A instances are considered the smallest, while instances in scenario B are larger. Problem instances in scenario C are very different to the instances in the other 6 scenarios in that the number of workers is much larger than the number of visits.

Table 3: Features of the WSRP instances.

Instance	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**Mean**
Number Visits	31	31	38	28	13	28	13	26
Number Workers	23	22	22	19	19	21	21	21
Number Areas	6	4	5	4	4	8	4	5
Instance	**B1**	**B2**	**B3**	**B4**	**B5**	**B6**	**B7**	**Mean**
Number Visits	36	12	69	30	61	57	61	47
Number Workers	25	25	34	34	32	32	32	31
Number Areas	6	5	7	5	8	8	7	7
Instance	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**Mean**
Number Visits	177	7	150	32	29	158	6	80
Number Workers	1037	618	1077	979	821	816	349	814
Number Areas	8	4	7	8	6	11	6	7

5.2 Performance of Operators

The first set of experiments was designed to select the best combination of the operators that maximise customer/worker requirements and preferences satisfaction. To do so, all operators listed in Table 2 were examined by statistical analysis to determine their performance. There are four mutation operators (RSM, IM, SM, CBM) and eight crossover operators divided into three different groups: routing crossovers

Table 4: Performance Comparison of Crossover Operators Grouped by Category (Routing, Scheduling, Cost-Based) Under a Mutation Operator.

	Mutation	RSM			SM			IM			CBM		
SSGA	**Crossover**	***OX***	*UX*	*CBUX*	***OX***	*UX*	*CBUX*	***OX***	*UX*	*CBUX*	***OX***	*1PX*	*CBUX*
	Dev.	**0.09%**	1.66%	4.99%	**0.07%**	1.23%	4.90%	**0.04%**	1.47%	4.24%	**0.66%**	1.37%	3.23%
	#Best	**0.71**	0.19	0.00	**0.76**	0.14	0.00	**0.86**	0.05	0.00	**0.48**	0.38	0.05
	Score	**0.90**	0.57	0.57	**0.93**	0.60	0.12	**0.98**	0.50	0.17	**0.76**	0.60	0.29
GenGA	**Crossover**	*PMX*	***UX***	*CBUX*	***PMX***	***IPX***	*CBUX*	*PMX*	***IPX***	*CBUX*	***PMX***	*UX*	*CBUX*
	Dev.	0.87%	**0.70%**	8.98%	2.28%	**1.17%**	9.82%	2.90%	**1.92%**	9.65%	**0.65%**	3.12%	5.19%
	#Best	0.38	**0.38**	0.14	**0.43**	0.38	0.14	0.29	**0.52**	0.19	**0.57**	0.19	0.14
	Score	0.67	**0.74**	0.24	**0.74**	0.69	0.24	0.67	**0.74**	0.31	**0.81**	0.50	0.33

(OX, CX, PMX), scheduling crossovers (1PX, 2PX, UX, HX) and cost-based crossovers (CBUX). Each crossover was combined with one mutation at a time for a total of 32 combinations, however only the best performing crossover operators from each group is presented in Table 4. The GA was executed for 5 minutes using the highest rate values, i.e. $P_c = 100\%$ and $P_m = 30\%$ to ensure that the operators were utilised.

The following three metrics were used to measure the performance of the combinations of operators: **Dev.** average percentage deviation from the best preference value (the three preferences satisfaction value of all the configurations applied). **Best** fraction of instances in a set for which a configuration matches the best preference value. This performance metric is absolute and can be compared across existing results in different tables. **Score** fraction of the instances for which the current method produces better solutions than the other configurations, i.e. 'win'. This score is calculated as $((q \times (p-1)) - r)/(q \times (p-1))$, where p is the number of configurations compared, q is the number of problem instances, and r is the number of instances in which the $p-1$ competing configurations find a better result. Hence, the best score value is 1, when $r = 0$, and the worst score value is 0, when $r = q \times (p-1)$. This is a relative measure of performance. Hence, these values are only meaningful within one table and not across different tables.

The results presented in Table 4 are the performance metrics values that correspond to each of the four mutation operators, including the CBM operator proposed in this study. These values are calculated based on the average preferences satisfaction values for each run. The crossover categories are: the best routing operator, the best scheduling operator and the CBUX operator proposed in this study. Thus, each mutation operator has three comparable crossover operators values, and the best crossover out of the three is highlighted in **bold**. The aim is to identify the best crossovers for each mutation with respect to the preferences value by grouping crossovers based on their category, thus mutation operators are not comparable in this table.

Table 5: Performance Comparison Between the Best Combinations of Operators (Mutation - Crossover).

	Procedur	Dev.	#Best	Score
SSGA	***RSM-OX***	**0.01%**	**0.52**	**0.87**
	SM-OX	0.03%	0.29	0.82
	IM-OX	0.15%	0.00	0.50
	CBM-OX	0.23%	0.10	0.45
GenGA	***RSM-UX***	**0.92%**	**0.57**	**0.84**
	SM-PMX	7.57%	0.05	0.43
	IM-1PX	2.73%	0.24	0.68
	CBM-PMX	18.36%	0.05	0.25

For SSGA, OX provides the best scores, with the highest number of best values and the lowest deviation when combined with all mutation operators. For GenGA, UX provides the best scores for RSM with the highest number of the best values and the lowest deviation. Even though UX is selected as the first competing crossover, PMX obtained the same number of the best values for RSM; the winning crossovers are considered in the next overall comparison. Additionally, 1PX provides the best score value with the highest number of best solutions and lowest on deviations among the compared crossovers for IM, while PMX provides the best score value for both SM and CBM, with the lowest deviation obtained for CBM only.

Table 5 shows a comparison between the chosen combinations of operators (mutation - crossover) from Table 4. The aim is to identify the best combination for each GA by using the same performance measurement matrices explained above. The best combination is highlighted in **bold**.

The results indicate that RSM–UX and RSM–UX obtain the highest score and the smallest deviation value among all methods, with the maximum fractions of the best solutions of 0.87 and 0.84 for SSGA and GenGA respectively. Interestingly, cost-based methods failed to achieve good results in comparison to the generic operators. This result might be due to the search space restrictions that led to infeasible areas. However, when combined with more generic

Table 6: Results of the best $f(S)$ produced by different combinations of operators, crossover probabilities P_c and mutation probabilities P_m.

		SSGA					GenGA				
	Instance	*Procedure*	P_c	P_m	$f(S)$	$Cpt(s)$	*Procedure*	P_c	P_m	$f(S)$	$Cpt(s)$
A	1	*SM-PMX*	0.5	0.3	5.1	176.6	*RSM-PMX*	1	0.3	**3.5**	180.8
	2	*SM-OX*	1	0.3	4.4	184.4	*SM-UX*	1	0.3	**2.8**	190.7
	3	*SM-UX*	1	0.3	6.1	240.8	*SM-1PX*	1	0.3	**3.3**	212.3
	4	*SM-UX*	0.5	0.3	2.3	159.9	*SM-UX*	1	0.3	**1.4**	114.5
	5	*SM-1PX*	1	0.1	3.2	50.7	*SM-HX*	1	0.3	**2.4**	52.4
	6	*SM-2PX*	1	0.1	4.4	198.6	*RSM-HX*	1	0.3	**3.6**	109.0
	7	*SM-2PX*	1	0.1	4.2	32.7	*RSM-HX*	1	0.3	**3.7**	83.2
B	1	*SM-UX*	1	0.3	2.1	239.7	*RSM-PMX*	1	0.3	**1.7**	255.0
	2	*SM-OX*	1	0.1	2.4	50.7	*RSM-HX*	1	0.3	**1.8**	14.7
	3	*RSM-PMX*	0.5	0.3	2.8	292.9	*RSM-PMX*	0.5	0.3	**1.9**	274.1
	4	*SM-2PX*	1	0.3	2.9	75.3	*SM-2PX*	1	0.3	**2.1**	138.0
	5	*RSM-UX*	0.5	0.3	3.8	243.5	*RSM-2PX*	1	0.3	**2**	243.2
	6	*RSM-UX*	1	0.3	2.5	226.1	*RSM-2PX*	0.5	0.3	**1.7**	229.2
	7	*RSM-UX*	0.5	0.3	3.2	231.8	*RSM-1PX*	0.5	0.3	**1.9**	288.2
C	1	*CBM-OX*	1	0.1	**5454.5**	285.0	*CBM-OX*	1	0.3	159418.6	304.8
	2	*SM-HX*	1	0.01	4.8	15.0	*SM-HX*	1	0.3	**3.2**	152.7
	3	*RSM-2PX*	1	0.3	**3270.7**	299.4	*SM-OX*	1	0.3	82582	293.9
	4	*RSM-OX*	1	0.01	22.2	210.0	*IM-HX*	1	0.3	**17.6**	269.8
	5	*SM-PMX*	1	0.1	20.1	172.6	*IM-HX*	1	0.3	**16**	267.3
	6	*CBM-PMX*	1	0.01	**20776.5**	266.6	*CBM-OX*	1	0.01	94335.9	297.8
	7	*SM-UX*	1	0.3	4.9	0.7	*SM-UX*	1	0.3	**4.3**	15.9

operators, in the case of CBM, they generate more diverse individuals that led up to high deviation among all mutations.

5.3 Computational Results for Different Instance Sizes

Table 6 presents the best objective values obtained for all instances. The columns under SSGA and GenGA provide the best values obtained for each GA under the stopping criterion for each combination. All values are averaged and only the best values are presented. The remaining column Cpt shows the computation time where the best value is found in seconds. Two issues were considered to compile this table: the GAs performance on each instance and the best performing combinations/settings under each GA with the minimum computation time.

It appears that GenGA provides better results than SSGA on 85.71% of all instances. The best-performing operators under the methods applied are PMX, UX and HX across all instances when combined with RSM and SM. However, CBM managed to obtain some of the best results, especially for scenario C instances. This indicates that problem domain knowledge needs to be incorporated in operators for more complicated instances. The average computation times for the best solution found for all instances are as follows: SSGA, and GenGA are 173.954 s and 189.88 s respectively. In terms of convergence speed, both SSGAs used here converged earlier to a local minima, with poor results in problem sets A and B. For problem set C, more computation-time provided better results when using GenGA.

Despite the fact that the cost values still need to be improved, this study has helped to understand the performance of various combinations of genetic operators executed with different probability rates and implemented on simple steady-state and generational GAs.

6 CONCLUSION

This paper has investigated the suitability of a set of genetic operators when applied within a steady-state genetic algorithm (SSGA) and a generational genetic algorithm (GenGA) to tackle the workforce scheduling and routing problem (WSRP). Twelve operators were considered in this study including two operators incorporating problem domain knowledge, and ten well-known operators (three mutation operators and

seven crossover operators) from the literature. From the experimental results, existing operators such as RSM and UX perform the best. Future research will look at investigating the performance of the repair operators, parameter setting of the operators and the design of an improved evolutionary approach informed by the better understanding achieved in this paper.

REFERENCES

Aickelin, U. and Dowsland, K. A. (2004). An indirect genetic algorithm for a nurse-scheduling problem. *Computers & Operations Research*, 31(5):761 – 778.

Algethami, H. and Landa-Silva, D. (2015). A study of genetic operators for the workforce scheduling and routing problem. In *11th Metaheuristics International Conference (MIC 2015)*, pages 1–11.

Algethami, H., Pinheiro, R. L., and Landa-Silva, D. (2016). A genetic algorithm for a workforce scheduling and routing problem. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 927–934.

Castillo-Salazar, J. A., Landa-Silva, D., and Qu, R. (2016). Workforce scheduling and routing problems: literature survey and computational study. *Annals of Operations Research*, 239(1):39–67.

Chang, Y. and Chen, L. (2007). Solve the vehicle routing problem with time windows via a genetic algorithm. *Discrete and continuous dynamical systems supplement*, pages 240–249.

Cicirello, V. A. and Cernera, R. (2013). Profiling the distance characteristics of mutation operators for permutation-based genetic algorithms. In Boonthum-Denecke, C. and Youngblood, G. M., editors, *FLAIRS Conference*, Florida. AAAI Press.

Cowling, P., Colledge, N., Dahal, K., and Remde, S. (2006). The trade-off between diversity and quality for multi-objective workforce scheduling. In *Proceedings of the 6th European Conference on Evolutionary Computation in Combinatorial Optimization*, EvoCOP'06, pages 13–24. Springer-Verlag.

Eshelman, L. J. (1991). The CHC adaptive search algorithm : How to have safe search when engaging in nontraditional genetic recombination. *Foundations of Genetic Algorithms*, pages 265–283.

Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics (NRL)*, 45(7):733–750.

Kotecha, K., Sanghani, G., and Gambhava, N. (2004). Genetic algorithm for airline crew scheduling problem using cost-based uniform crossover. In Manandhar, S., Austin, J., Desai, U. B., Oyanagi, Y., and Talukder, A. K., editors, *AACC*, volume 3285 of *Lecture Notes in Computer Science*, pages 84–91, Kathmandu, Nepal. Springer.

Laesanklang, W. and Landa-Silva, D. (2016). Decomposition techniques with mixed integer programming and heuristics for home healthcare planning. *Annals of Operations Research*, pages 1–35.

Lenstra, J. K. and Kan, A. H. G. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227.

Mankowska, D., Meisel, F., and Bierwirth, C. (2014). The home health care routing and scheduling problem with interdependent services. *Health Care Management Science*, 17(1):15–30.

Mısır, M., Smet, P., and Vanden Berghe, G. (2015). An analysis of generalised heuristics for vehicle routing and personnel rostering problems. *Journal of the Operational Research Society*, 66(5):858–870.

Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. The MIT Press, Cambridge, MA, USA.

Mutingi, M. and Mbohwa, C. (2014). Health-care staff scheduling in a fuzzy environment: A fuzzy genetic algorithm approach. In *Conference Proceedings (DFC Quality and Operations Management)*. International Conference on Industrial Engineering and Operations Management.

Oliver, I. M., Smith, D. J., and Holland, J. R. C. (1987). A study of permutation crossover operators on the travelling salesman problem. In *Proceedings of the Second International Conference on Genetic Algorithms and their application*, pages 224–230, Hillsdale, NJ, USA. L. Erlbaum Associates Inc.

Pinheiro, R. L., Landa-Silva, D., and Atkin, J. (2016). A variable neighbourhood search for the workforce scheduling and routing problem. In *Advances in Nature and Biologically Inspired Computing*, pages 247–259. Springer, Pietermaritzburg, South Africa.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985 – 2002.

Rothlauf, F. (2003). Representations for genetic and evolutionary algorithms. *Studies in Fuzziness and Soft Computing*, 104:9–32.

Toth, P. and Vigo, D. (2014). *The vehicle routing problem*, volume 18. Siam.

Vavak, F. and Fogarty, T. C. (1996). Comparison of steady state and generational genetic algorithms for use in nonstationary environments. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 192–195. IEEE.

Zheng, D.-Z. and Wang, L. (2003). An effective hybrid heuristic for flow shop scheduling. *The International Journal of Advanced Manufacturing Technology*, 21(1):38–44.

Zhu, K. Q. (2000). A new genetic algorithm for VRPTW. In *Proceedings of the International Conference on Artificial Intelligence*. Citeseer.