A Genetic Algorithm for a Workforce Scheduling and Routing Problem

Haneen Algethami, Rodrigo Lankaites Pinheiro, Dario Landa-Silva
School of Computer Science, ASAP Research Group
The University of Nottingham, United Kingdom
Email: {psxha7, psxrp2, dario.landasilva}@nottingham.ac.uk

***Abstract*—The Workforce Scheduling and Routing Problem refers to the assignment of personnel to visits across various geographical locations. Solving this problem demands tackling scheduling and routing constraints while aiming to minimise the total operational cost. This paper presents a Genetic Algorithm (GA) tailored to tackle a set of real-world instances of this problem. The proposed GA uses a customised chromosome representation to maintain the feasibility of solutions. The performance of several genetic operators is investigated in relation to the tailored chromosome representation. This paper also presents a study of parameter settings for the proposed GA in relation to the various problem instances considered. Results show that the proposed GA, which incorporates tailored components, performs very well and is an effective baseline evolutionary algorithm for this difficult problem.**

Keywords*—Genetic Algorithms, Indirect Solution Representation, Genetic Operators, Workforce Scheduling and Routing

I. Introduction

The Workforce Scheduling and Routing Problem (WSRP) is described as the assignment of personnel to visits, requested by customers, across different geographical locations. This problem combines personnel scheduling and routing problems, both of which are known to be NP-Hard [1]. The *scheduling aspect* of the problem assigns personnel to visits in order to fulfil the work demands and other requirements. The *routing aspect* of the problem consists of generating routes for the workers to service customers across various locations within given time-windows. The objective is to minimise operational costs while attending the additional requirements expressed by customers, workers and the business. A type of WSRP arises in home health care where nurses and care workers should be assigned to visit patients in their homes in order to carry out some tasks, e.g. administering medication, monitoring serious illness and unstable health status and injections. A set of real-world home health care problem instances presented in [2] are tackled in this paper with a tailored genetic algorithm.

Genetic Algorithms (GAs) have been shown to be effective approaches to find solutions for scheduling problems where exact methods are less effective, e.g. [3], [4]. GAs have also been applied to problems combining scheduling and routing [5], [6]. Nevertheless, to the best of our knowledge, genetic components have not been explicitly investigated for WSRP. A comprehensive study of genetic operators within a steady state GA for WSRP was presented in [7]. That study showed that achieving feasibility in solutions to this problem is a serious challenge for most known genetic operators. Even after incorporating a repair operator, infeasibility was a serious issue for the steady state GA.

Given the infeasibility issue mentioned above, this paper presents an efficient GA customised for tackling the combined setting of scheduling and routing in WSRP. To this end, the specific objectives of this research are:

- To develop an indirect chromosome encoding scheme that ensures the feasibility of solutions.
- To develop cost-based genetic operators tailored for the indirect chromosome encoding scheme.
- To carry out computational tests to evaluate the proposed GA and compare the performance of the various genetic operators considered. A total of 42 problem instances from six different real-world home health care scenarios were used in the experiment.

The remainder of this paper is organised as follows. Section II reviews recent developments in designing problem-specific chromosome representations. Then, Section III describes the WSRP, its formulation and the instances used in this paper. Section IV outlines the proposed GA. Section V presents the experimental study and discusses the results. The paper is then concluded in Section VI.

II. Related Work

Research has been conducted on WSRP scenarios with *connected activities* or tasks. Activities are said to be connected when there is some precedence relation between activities or when some activities need two or more workers. The existence of connected activities gives raise to *time-dependent activities constraints*. A study by Castillo-salazar et al. [8] applied a mixed integer programming (MIP) solver to tackle WSRP with time-dependent activities constraints. That study revealed that tackling instances with 50 activities or more required considerable computational time. Hence, a greedy heuristic tailored for such scenarios with connected activities was proposed in [9] and an MIP with decomposition method was proposed in [10]. That decomposition method splits each problem into sub-problems and solves each sub-problem with an MIP solver. Then, the partial solutions are combined and the solution to the overall problem is further improved with a heuristic repair process. The MIP with decomposition outperformed the greedy heuristic in 56% of all instances although using more computational time due to the heuristic repair process.

Research has also been conducted on WSRP scenarios with *non-connected activities* or tasks, i.e. problems with no *time-dependent activities constraints*. An MIP with decomposition method [11] required considerable computation time (up to several hours) to solve larger problem instances with hundreds of tasks, indicating the need for faster solution methods. A Variable Neighbourhood Search (VNS) algorithm using problem-specific neighbourhood heuristics was presented in [2]. The VNS obtained high-quality solutions and in less computation time for the same set of problem instances used in [11]. The present paper considers WSRP scenarios with non-connected activities. An initial investigation was presented in [7] comparing various genetic operators within a simple GA to tackle the subject problem. The components of that simple GA included a direct representation scheme, a time-window conflict reduction heuristic and a repair mechanism to tackle infeasibility. Up to 39.68% of solutions in the last iteration were found to be infeasible even with the time-window conflict reduction heuristic. Using the repair operator was too expensive in terms of computational time. Hence the motivation for designing a GA capable of maintaining feasibility in these WSRP scenarios with non-connected activities.

Only few works have investigated Evolutionary Algorithms (EAs) to solve problems combining scheduling and routing. A fuzzy GA for home health care worker scheduling was implemented in [6]. However, in that work attention was given to developing an algorithm to produce better solutions without explicitly investigating the genetic elements involved. Also, repair heuristics were used to deal with infeasible solutions. Researchers have investigated various methods to maintain solution quality throughout the execution of GAs by sustaining a healthy population. Two quality measures of a healthy population can be *solution feasibility* and *population diversity*. Constraint handling techniques such as initialisation methods, tailored operators and repair heuristics can be used to control solution feasibility while maintaining a diverse population.

Two main types of chromosome encoding techniques exist in the literature: direct an indirect representations. A *direct representation* can be a vector that directly encodes the solution, e.g. a simple permutation of visits, jobs, operations, etc. [12]. Direct representations are easy to implement and can produce a large number of encoded solutions in the search space. However, constraints are usually not handled by the representation, thus it can produce infeasible solutions and hence complicate the search [13]. An *indirect representation* encodes an abstract form of the solution that requires an encoding mechanism prior to the chromosome decoding in order to obtain a solution. Indirect representations often use a straightforward procedure to transform a chromosome into a candidate solution [13]. The level of abstraction is chosen in order to balance between genotype simplicity and fast encoding/decoding. One main advantage of indirect representations is that problem-specific information can be incorporated into the design of the encoding scheme, thus constraints can be handled by the representation. As a result, smaller and simpler search spaces are produced. For example, special encoding schemes have been developed to exploit problem-specific knowledge when tackling a highly constrained real-world problem such as nurse rostering [14].

The study in [3] investigated different solution representation schemes with different *degrees of directness* in scheduling problems. The algorithms with less direct solution representations required less computation time than algorithms with direct representations. It was also shown that indirect representations can achieve better results in scenarios with large search spaces. Similarly, [4] described an encoding scheme designed for machine selection and operation sequencing that respects all constraints while implementing different crossovers. The encoding scheme ensured the generation of a high-quality initial population with better performance when compared to previously published algorithms. Recent work on WSRP also suggests that incorporating problem-specific knowledge into the representation helps to maintain the feasibility of solutions [2], [7]. Indirect representation techniques have also been used in [15] where a WSRP was tackled with particle swarm optimisation. However, in that work, solutions still required improvements with a repair operator.

The intended contribution of this work is to present an efficient customised chromosome representation combined with problem-specific genetic operators for tackling real-world WSRP instances. To the best of our knowledge, such approach has not yet been proposed in the literature. The proposed solution encoding is capable of producing competitive feasible solutions for a highly constrained WSRP. The paper also presents a study of genetic operators using the proposed solution encoding. Eight well-known genetic operators plus five proposed tailored operators are implemented. The proposed GA harnesses the power of these genetic components to solve the WSRP instances considered.

III. Problem Description

The goal in the WSRP tackled here is to generate a daily plan of visits by workers to customers at different locations. There is a set of workers $W = \{w_1, w_2, \ldots, w_{|W|}\}$ and a set of tasks $T = \{t_1, t_2, \ldots, t_{|T|}\}$ requested by customers. The assignment of a worker to travel to a customer location to perform a task is called a visit. Note that some tasks might require more than one worker, hence a task might generate more than one visit. In particular, 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 are important in solutions to HHC scenarios, such as as distance travelled, assigning all visits, customers and workers requirements and preferences [16]. Thus, it is desirable that a good quality plan for a HHC planning problem has all those features while minimising operational cost. Please refer to [11] for details of the MIP model for the WSRP tackled here.

Seven problem datasets from real-world HHC scenarios are tackled in this paper. These datasets have been provided by an industrial partner and correspond to HHC scenarios in the UK. Each dataset is composed of seven problem instances for a total of 42 instances. Table I shows the main features of each problem instance and the mean for all instances in each dataset. These features are: the number of visits, the number of workers and the number of geographical areas (visit locations are grouped into areas). Note that dataset A has the smallest instances while instances in dataset B are larger and the instance size increases with D, E and F. Problem instances in dataset C are different as they have a much larger number of workers.

Table I. MAIN FEATURES OF THE 42 HOME HEALTH CARE PROBLEM INSTANCES.

	A								B							
	01	**02**	**03**	**04**	**05**	**06**	**07**	**mean**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**mean**
Number of Visits	31	31	38	28	13	28	13	26	36	12	69	30	61	57	61	46
Number of Workers	23	22	22	19	19	21	21	21	25	25	34	34	32	32	32	30
Number of Areas	6	4	5	4	4	8	4	5	6	5	7	5	8	8	7	7

	C								D							
	01	**02**	**03**	**04**	**05**	**06**	**07**	**mean**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**mean**
Number of Visits	177	7	150	32	29	158	6	80	483	454	585	520	538	610	611	543
Number of Workers	1037	618	1077	979	821	816	349	813	164	166	174	174	173	174	173	171
Number of Areas	8	4	7	8	6	11	6	7	13	12	15	15	15	15	15	14

	E								F							
	01	**02**	**03**	**04**	**05**	**06**	**07**	**mean**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**mean**
Number of Visits	418	425	462	351	461	301	498	416	1211	1243	1479	1448	1599	1582	1726	1470
Number of Workers	243	244	267	266	278	278	302	268	805	769	898	789	889	783	1011	901
Number of Areas	13	14	15	13	15	13	16	14	45	46	54	47	59	44	64	51

Table II. OBJECTIVES AND CONSTRAINTS IN THE WSRP.

Objectives	**Hard Constraints**	**Soft Constraints**
Travelling Cost	Skills Requirements	Unassigned Visits
Payment Cost	Time-Conflicts	Area Availability
	Maximum Hours	Time Availability
		Preferences (skills,worker,area)

A solution to the WSRP considered here requires all visits to be assigned while satisfying some requirements as follows. Assigning visits must be done according to workers skills, time-conflicts must be avoided, workers' maximum working hours must not be exceeded. Also, workers time and area availability as well as preferences expressed by workers' and customer' should be observed. Such preferences include workers preferring to work in certain geographical areas, customers requiring workers with special skills and customers preferring certain workers. A time-conflict occurs when a worker is assigned to visits overlapping in time. Table II lists the WSRP objectives and constraints considered here. From the hard constraints in the table, skills requirements and maximum hours are enforced by the assignment of only suitable workers to visits. The time-conflicts hard constraint and all the soft constraints are dealt within the objective function.

For a solution S, the objective function includes the **operational cost** and the **penalty cost** due to the non-satisfaction of requirements. The operational cost is given by $O(S) = c + d$ with payment cost represented by c and travelling cost represented by d, i.e. wages plus the journeys cost for all workers. The penalty cost includes the non-satisfaction of preferred skills (ρ_s), preferred workers (ρ_w) and preferred areas (ρ_a). This is given by $P(S) = 3V - \rho_s - \rho_w - \rho_a$. The value $3V$ is used because the values of (ρ_s), (ρ_w) and (ρ_a) can be between 0 and 1. The number of assignments or visits is V. The penalty cost also includes the violation of workers availability in terms of area (ψ_a) and time (ψ_t). This is given by $A(S) = \psi_a + \psi_t$. Finally, the number of unassigned visits (ω) and the number of time-conflicts (ϕ) are also part of the penalty cost.

$$f(S) = \lambda_1 \times O(S) + \lambda_2 \times P(S) + \lambda_3 \times A(S) + \lambda_4 \omega + \lambda_5 \phi \quad (1)$$

Then, the objective function for a solution S is given by equation (1). Each of the five terms is multiplied by a coefficient ($\lambda_1, \dots, \lambda_5$) to enforce levels of priority with λ_5 being the highest and λ_1 the lowest. Hence, the occurrence of time-conflicts (ϕ) carries a higher penalty than the occurrence of unassigned visits (ω) and so on. Note that there are two terms in the objective function associated to the area in which workers are assigned to visit. If a worker is assigned outside his/her available area, this is accounted through ψ_a in $A(S)$. If a worker is not assigned to one of his/her preferred areas, this is accounted through ρ_a in $P(S)$.

IV. PROPOSED GENETIC ALGORITHM (GA)

An effective baseline GA to tackle the HHC problem instances in the previous section is described here. The proposed GA uses an indirect chromosome encoding and incorporates various genetic operators including some new problem-specific ones that utilise heuristics to generate improved offspring. The proposed GA works as follows. First, an initial population of N individuals (one-day plans) is created based on the indirect chromosome encoding to ensure solutions feasibility. At the start of each generation, $9N/20$ pairs of parent individuals are selected using binary tournaments. With some probability, one of the eight studied crossovers is applied to each pair of parents to generate two offspring. With some probability, each offspring goes through one of the five studied mutation operators. At the end of each generation, the population is updated using an elitism strategy. The $N/10$ best individuals from the current population are kept and along with the $9N/20$ offspring individuals generated, the new population of N solutions is formed. The indirect chromosome encoding and genetic operators are described next.

A. Indirect Chromosome Encoding

Each individual in the population is represented by a vector of integers of length V (number of visits). Each position in the vector corresponds to a visit and the integer represents the worker assigned to that visit. The visits in the chromosome are sorted in a non-decreasing order of their start time. A *worker suitability list (wsl)* is created for each visit. For each visit, *wsl* contains suitable workers only and ranked by a penalty scoring mechanism. The penalty score of each worker is calculated by estimating the impact of assigning that worker to that visit, considering incurred operational cost and penalty cost due to preferences and availability restrictions. The lower the penalty score the better. Then, the first worker in the *wsl* for a gene is the most suitable one for that visit, followed by the next best suitable worker and so on. Figure 1 illustrates an example of the indirect chromosome encoding for a day plan with $V = 7$ visits. Each worker w_j has a penalty score according to their suitability to work in visit v_k, where $k = 1, 2 \ldots, V$. Each visit has a *wsl* with the four most suitable workers and the best worker (lowest penalty score) at the top. On the right of the figure, the chromosome shows the encoded solution and gives an index of a worker in the *wsl* for each visit. Below the chromosome, the decoded solution shows the actual worker assigned to each visit.

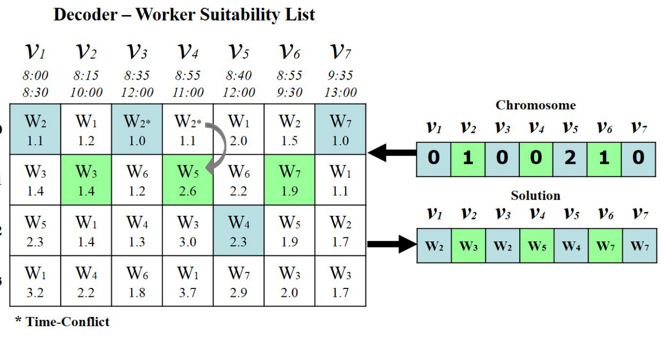
Decoder – Worker Suitability List
v1 8:00 8:30
v2 8:15 10:00
v3 8:35 12:00
v4 8:55 11:00
v5 8:40 12:00
v6 8:55 9:30
v7 9:35 13:00
0: W2 1.1 | W1 1.2 | W2* 1.0 | W2* 1.1 | W1 2.0 | W2 1.5 | W7 1.0
1: W3 1.4 | W3 1.4 | W6 1.2 | W5 2.6 | W6 2.2 | W7 1.9 | W1 1.1
2: W5 2.3 | W1 1.4 | W4 1.3 | W3 3.0 | W4 2.3 | W5 1.9 | W2 1.7
3: W1 3.2 | W4 2.2 | W6 1.8 | W1 3.7 | W7 2.9 | W3 2.0 | W3 1.7
* Time-Conflict
Chromosome
v1 v2 v3 v4 v5 v6 v7
0 1 0 0 2 1 0
Solution
v1 v2 v3 v4 v5 v6 v7
W2 W3 W2 W5 W4 W7 W7

Fig. 1. Example of the Indirect Chromosome Encoding Scheme.

Another key usability of the *wsl* is time-conflict reduction. When a worker is being considered for a specific visit, the solution is evaluated to find out if time-conflicts arise. The initial population for the GA is created randomly by generating a vector of V integers between 0 and $L_k - 1$ where L_k is the length of *wsl* for visit v_k. If a time-conflict occurs because of the worker randomly assigned to visit v_k, the next worker in the *wsl* for that visit is considered until a suitable worker is found with no time-conflicts arising. This time-conflict reduction mechanism can be illustrated with Figure 1 as follows. Time-conflicts that arise when processing the visits are indicated with $*$. For example, w_2 is assigned to both v_1 and v_3 while w_3 is assigned to v_2. No time-conflict arises because v_1 and v_3 do not overlap. However, the chromosome assigns w_2 to v_4 and a time-conflict occurs as v_3 overlaps with v_4. Then, the next most suitable worker that does not provoke a time-conflict, in this case w_5, is assigned to v_4. Hence, the *wsl* decoder in this indirect chromosome encoding scheme helps to assign suitable workers to visits while avoiding time-conflicts. Note that the penalty scores are not used during the generation of initial solutions, they are used for the tailored genetic operators described below.

B. *Genetic Operators*

Several crossover and mutation operators are implemented here. Some of these operators are taken from the literature and others are specially designed for the problem and solution representation considered here. Four crossover operators are taken from the literature: Single Point (1PX), Two Point (2PX), Uniform (UX) and Half Uniform (HUX) [17]. Four mutation operators are taken from the literature: Swap, Inversion, Flip and Scramble [18]. Four problem-specific crossover operators are designed here: Cost-Based Greedy (CGX), Partially Matched Greedy (PMGreedyX), Flat-Costs (FCX) and Partially Matched Flat-Costs (PMFCX). Also, one problem-specific mutation operator is designed here: Zero-Flip Mutation. The indirect chromosome encoding scheme described above allows to implement all genetic operators while maintaining the feasibility of solutions. The problem-specific crossover and mutation operators developed here are described next.

Algorithm 1 Cost-Based Greedy Crossover (CGX)

```
Require: Pair of parent individuals $p_1$ and $p_2$
Ensure: Pair of offspring individuals $o_1$ and $o_2$
 1: $o_1, o_2 \leftarrow$ new empty individuals
 2: $o'_1, o''_1 \leftarrow$ temporary individuals
 3: for $i \leftarrow 1$ to $V$ do
 4:     $o'_1 \leftarrow o_1 \cup p^i_1$
 5:     $o''_1 \leftarrow o_1 \cup p^i_2$
 6:     if $f(o'_1) < f(o''_1)$ then
 7:         $o_1 \leftarrow o_1 \cup p^i_1$
 8:         $o_2 \leftarrow o_2 \cup p^i_2$
 9:     else
10:         $o_2 \leftarrow o_2 \cup p^i_1$
11:         $o_1 \leftarrow o_1 \cup p^i_2$
12:     end if
13: end for
```

The **Cost-Based Greedy Crossover (CGX)** is essentially a variation of a local search constructive heuristic that uses the objective function to transfer the best genes from each parent to one of the offspring individuals. Algorithm 1 shows how this CGX crossover operator works. The operator goes through each of the V positions in the chromosome. For each i^{th} position, it copies offspring o_1 plus the corresponding gene from each parent respectively to o'_1 and o''_1 (steps 4 and 5). Then, the parent's gene that results in a better individual is actually added to offspring o_1 while the other parent's gene is added to offspring o_2 (steps 6 to 12). Then, CGX produces one offspring (o_1) with the best parent's gene selected at each step and one offspring (o_2) with the other parent's gene.

The **Partially Matched Greedy Crossover (PMGreedyX)** is an extension of the well-known partially matched crossover (PMX). Algorithm 2 shows how PMGreedyX works. Basically, it selects a crossover segment (steps 4 and 5) to exchange genes as usual (steps 17 to 20) and applies the same mechanism as CGX outside the crossover segment (steps 6 to 16).

The **Flat-Costs Crossover (FCX)** operator is an extension of the constructive heuristic proposed by [2] that successfully obtained good results for WSRP. Instead of computing the objective function $f(S)$ for the partial solution as in the CGX operator, the FCX crossover uses the penalty scores in the *wsl*. Algorithm 3 shows how this FCX crossover operator works. The operator goes through each of the V positions in the chromosome. For each i^{th} position, the crossover selects the better scored worker from the two parents for offspring o_1 and the other worker for offspring o_2. The FCX operator also produces one offspring (o_1) with the best parent's gene selected at each step and one offspring (o_2) with the other parent's gene, but a 'flat-cost' is used instead of the full $f(S)$ calculation.

Like PMGreedyX, the **Partially Matched Flat-Costs Crossover (PMFCX)** is an extension of the well-known PMX crossover, but in this case the mechanism applied outside the crossover segment is the one used in FCX. Due to limited

Algorithm 2 Partially Matched Greedy Crossover (PM-GreedyX)

```
Require: Pair of parent individuals $p_1$ and $p_2$
Ensure: Pair of offspring individuals $o_1$ and $o_2$
 1: $o_1, o_2 \leftarrow$ new empty individuals
 2: $o'_1, o''_1 \leftarrow$ temporary individuals
 3: $rand$ := Number chosen at random from [0,1)
 4: $Point_1 := 2 + |rand * (V - 4)|$
 5: $Point_2 := Point_1 + 2 + |rand * (V - Point_1) - 2)|$
 6: for $i \leftarrow 1$ to $Point_1 - 1$ and $i \leftarrow Point_2 + 1$ to $V$ do
 7:     $o'_1 \leftarrow o_1 \cup p_1^i$
 8:     $o''_1 \leftarrow o_1 \cup p_2^i$
 9:     if $f(o'_1) < f(o''_1)$ then
10:         $o_1 \leftarrow o_1 \cup p_1^i$
11:         $o_2 \leftarrow o_2 \cup p_2^i$
12:     else
13:         $o_2 \leftarrow o_2 \cup p_1^i$
14:         $o_1 \leftarrow o_1 \cup p_2^i$
15:     end if
16: end for
17: for $i \leftarrow Point_1$ to $Point_2$ do
18:     $o_2 \leftarrow o_2 \cup p_1^i$
19:     $o_1 \leftarrow o_1 \cup p_2^i$
20: end for
```

Algorithm 3 Flat-Costs Crossover (FCX)

```
Require: Pair of parent individuals $p_1$ and $p_2$. The set of wsl
    for all visits, $M_{i,j}$ is the penalty score for the $j^{th}$ worker
    in the wsl for visit $v_i$.
Ensure: Pair of offspring individuals $o_1$ and $o_2$.
 1: $o_1, o_2 \leftarrow$ new empty individual
 2: for $i \leftarrow 1$ to $V$ do
 3:     if $(M_{i,p_1^i} < M_{i,p_2^i})$ then
 4:         $o_1 \leftarrow o_1 \cup p_1^i$
 5:         $o_2 \leftarrow o_2 \cup p_2^i$
 6:     else
 7:         $o_2 \leftarrow o_2 \cup p_1^i$
 8:         $o_1 \leftarrow o_1 \cup p_2^i$
 9:     end if
10: end for
```

space, the pseudocode for PMFCX is omitted from this paper. Besides, as it will be discussed in the results section, this operator did not perform very well.

The **Zero-flip Mutation** operator simply replaces the worker currently assigned to a visit by the most suitable worker for that visit, regardless of time-conflicts. That is, the integer in position i of the chromosome is changed to 0 indicating that the first worker in *wsl* is now assigned to visit v_i.

V. Experimental Study and Results

Experiments were conducted to assess the performance of the proposed GA on the 42 real-world WSRP scenarios from Table I. The machine used was an I7 4-core with hyper-threading and 16GB, the code was implemented in Java.

The first set of experiments was to identify the best genetic operators and parameter settings for each problem instance except those in dataset F (the largest ones). The different

Table III. GA Parameter Settings

Variable	Value
Population Size N	100, 250 and 500
Mutation Rate P_m	1%, 10%, 30%, 40% and 50%
Crossover Rate P_c	10%, 50% and 100%
Stopping Criterion	5 minutes
Selection Strategy	Elitist

parameter settings in Table III were tested. This means that for each of the 35 problems instances in groups A to E, there were 1800 configurations (8 crossovers $\times$ 5 mutations $\times$ 45 parameter settings). Each configuration used the same initial population and was executed 4 times, each one ran for 5 minutes. This means a massive amount of computation time but since it was possible to execute parallel runs, the total computation time was drastically reduced but still accounted for many days. Since each mutation operator was combined with one crossover operator at a time, in order to compare results for each dataset, average values of runs were computed. These values were normalised because of the different scales in the results for the different groups of problem instances. The observations made in these experiments informed the setting of parameters for the F problem instances.

The performance of each mutation operator is shown in Figure 2. The x axis presents the instance sets while the y axis presents the average gap to the best-known solutions. For each mutation operator, a point in the graph corresponds to the average gap considering all instances in the corresponding set. The dash line joining the 5 points is only used to visualise the overall performance of the mutation operator across all sets. Fig. 2 shows that for all instances the Flip mutation operator produced clearly better results on sets A, B and C. It also performed better on sets D and E but by a smaller margin. The Flip mutation operator is the only one that introduces new genes (using the *wsl*) to the chromosome encoding. The other mutation operators only reallocate existing genes. This may be the contributing factor that introduces diversity to enhance the overall performance of the Flip mutation. The much better results obtained by the Flip mutation on sets A, B and C might be because the algorithm was able to perform many more generations than for sets E and F. This increased the algorithm's ability to bring new genes into the population and boost its performance.

In order to investigate whether certain mutation rate is better overall, aggregated results are now presented in Figure 3 for each mutation rate on each set of instances. Setting an appropriate mutation rate helps a GA not to get stuck in local optima. The order of the mutation rates in Figure 3 for sets A and B is inverted with respect to the order for sets E and F, with set C being the inflection point. For the smaller instances, a high mutation rate proves to be considerably more effective than a low mutation rate. However, as the instances grow in size, the opposite happens, with a low mutation rate being more effective. The reason for this is that a high mutation rate reduces the convergence rate of the algorithm but increases diversity in the population [19]. Hence, on the long term, having a high mutation rate aids the overall performance of the GA. Although the algorithm is converging slower, it is exploring a wider region of the search space and avoiding

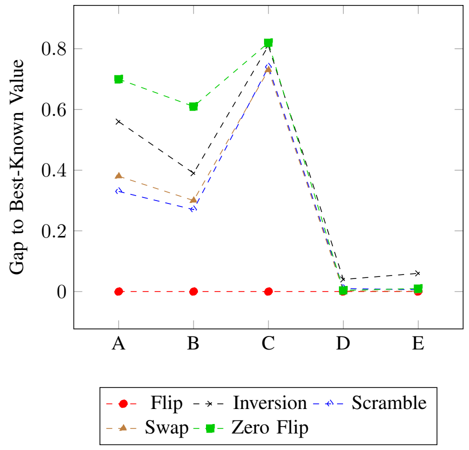

Fig. 2. Performance Comparison Between the Mutation Operators.

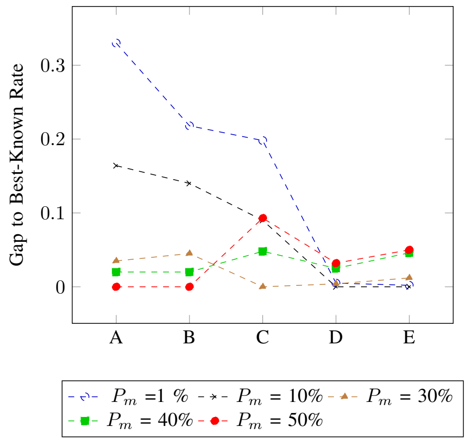

Fig. 3. Performance Comparison Between the Mutation Rates P_m.

local optima with a higher probability. A low mutation rate, however, helps the algorithm to converge much faster, but the likelihood of getting stuck into local optima is higher. Given the time limit set for each run in these experiments, many more generations take place for sets A and B than for sets D and E. Hence, the highest mutation rates provides better results on set A and B and the lowest mutation rate provides better results on sets D and E. Then, on smaller problem instances, higher mutation rates give better results because the increase in diversity allows the algorithm to avoid local optima. On larger problem instances, lower mutation rates give better results because with few generations, speed of convergence is more important than diversity. Perhaps with much longer computation time, higher mutation rates could produce better results on sets D and E but such experiments were not conducted here.

The performance of each crossover operator is shown in Figure 4. The figure shows that two of the four proposed crossover operators, CGX and PMGreedyX, outperformed well-known crossovers such as 1-point and UX. These results support the idea of having problem-specific operators to help the GA to converge to better solutions. These cost-based crossover operators CGX and PMGreedyX exploit domain-knowledge to provide improved solutions. Figure 4 shows that the performance of the crossover operators also depends on

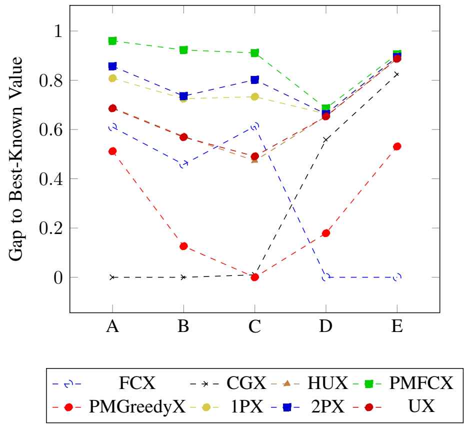

Fig. 4. Performance Comparison Between the Crossover Operators.

the problem size. For sets A and B, CGX obtained the best results. Since problem instances in sets A and B have less number of visits, the greedy evaluation in CGX is fast and helps in the convergence to better solutions. Also on sets A and B, PMGreedyX performed better than FCX and PMFCX but worse than CGX. This might be a consequence of the additional time that this PMX extended crossover spends on constructing an offspring compared to CGX. For the larger problem instances in sets D and E, FCX performed the best, while it performed worse than other operators in sets A, B and C. This seems to indicate that instead of the greedy evaluation, using the pre-calculated penalty scores in *wsl* is more effective in larger problems but it reduces the search space too much in smaller problem instances. However, one of the proposed operators with pre-calculated penalty scores, PMFCX, performed the worse across all sets. This might be an indication that combining PMX with FCX to construct an offspring is not too successful. The flat-cost evaluation affected the production of offspring with an increase in the number of generations. Thus, estimated costs were not accurate enough in the case of PMFCX. Note that for the set C, PMGreedy obtained slightly better results that CGX. This might be due to the much larger number of workers in the problem instances of set C. This could be an advantage for PMXGreedy when exchanging workers between the two crossover points based on the greedy evaluation.

Figure 5 shows the comparison between crossover rates across sets. Results indicate that a low crossover rate of 10% was not successful. For the higher crossover rates of 50% and 100%, the following observations are made. For larger problems (longer chromosomes) in sets D and E, the highest crossover rate is better. But for smaller problems in sets A, B and C, a rate of 50% gives better results. The number of generations may also play a role in these results. A lower crossover rate with slower convergence might be fine for smaller instances given the limited run-time. However, for larger instances, the number of generations is lower hence higher crossover rate could help to speed up the convergence.

Fig. 6 shows the comparison between population sizes for all sets. For the smaller problems in sets A, B and C, larger population sizes give better results. For the larger problems in sets D and E, the smaller the population size the better results obtained. The size of the problem instance is of course

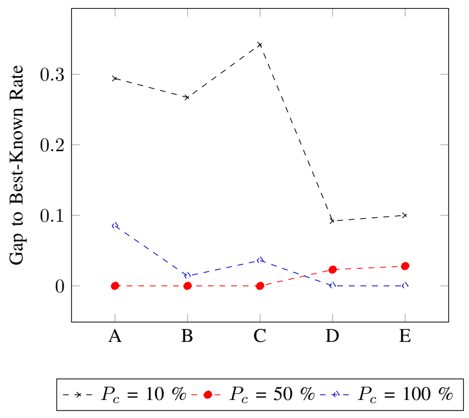

Fig. 5. Performance Comparison Between the Crossover Rates P_c.

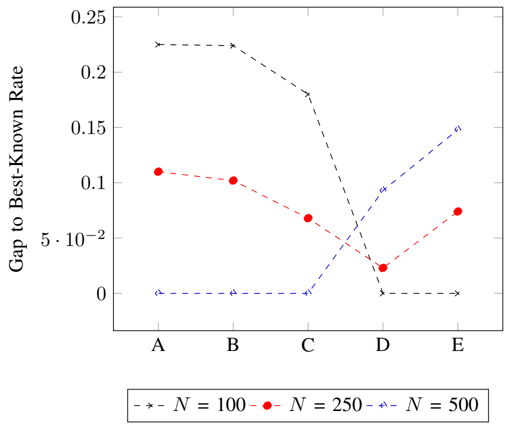

Fig. 6. Performance Comparison Between the Population Sizes N.

reflected in the length of the chromosome. This means that in smaller problem instances, a larger population helps to have more diversity, while in larger problem instances the longer chromosomes means less need for a larger population.

Based on the above findings, the combinations of genetic operators and their corresponding rates proposed here are as follows for each set of problem instances:

A,B: Flip, P_m = 50 %, GCX, P_c= 50%, N= 500.

C: Flip, P_m = 30 %, GCX, P_c= 50%, N= 500.

D,E: Flip, P_m = 10 %, FCX, P_c= 100%, N= 100.

Using the above settings, further experiments were conducted for all problem instances but with longer computation times. The experiments for the largest problem instances in set F were executed using the same settings as for sets D and E. The computation times used were as follows: 15 minutes per run for sets A, B and C; 60 minutes per run for sets D and E; 8 hours per run for set F. Note that due to the much longer run-time for instances in sets of D, E and F, the population size was increased to $N = 250$ to allow more exploration.

Table IV presents the average best results obtained with the proposed baseline GA using the settings recommended above. Results from the GA are compared to the MIP decomposition method in [11] and the VNS algorithm in [2]. For each problem instance, the table shows the solution quality $f(S)$ obtained by each method and the computation time in seconds *Cpt* in which the best solution by the method was found. When available, the optimal solution quality and corresponding computation time are also shown. These optimal results were obtained with an MIP solver solving the whole problem instance as reported in [20]. For each problem instance, the best known solution quality value (this is of course the optimal solution when available) and best computation time are highlighted in bold.

The results in Table IV indicate that for the smaller problems in sets A, B and C, the proposed GA is quite competitive matching the best known results for many of those instances. The VNS seems to provide better overall results but the GA outperforms the MIP decomposition method. For some of these small problem instances the GA spends considerable more time than the other methods, however, still below 15 minutes which is quite acceptable. The MIP decomposition method solves sub-problems to optimality but the integrated solutions are not optimal. Nevertheless, the MIP decomposition obtains good quality solutions in short computation times.

For the largest problems instances in sets D and F, the proposed GA performed better than the MIP decomposition method (results with the VNS were not available). For instances in set D, the GA did not match the results from the VNS but it got very close and the GA outperformed the MIP decomposition in those instances. It is argued that the proposed baseline GA obtained the better results on the largest instances in sets E and F because of the tailored indirect chromosome encoding scheme and genetic operators.

VI. Conclusions

This study was undertaken to design problem-specific elements on a Genetic Algorithm (GA) to tackle a Workforce Scheduling and Routing Problem (WSRP) and evaluate their offline tuning. An indirect chromosome encoding scheme was designed to produce and maintain feasible solutions throughout the evolutionary process. In addition to four crossover and four mutation operators taken from the literature, four crossover operators and one mutation operator were designed for the subject problem. Hence, eight crossover operators and five mutation operators were implemented in this study.

Comprehensive computational experiments were conducted using various settings of crossover rate, mutation rate and population size, for each combination of one crossover operator and one mutation operator. A total of 35 real-world problem instances, grouped into five sets, were used for these experiments in order to identify the recommended GA settings for each set of problems. The results from this experiments showed that the indirect chromosome encoding is effective in maintaining the feasibility of solutions and that some of the proposed crossover operators helped the GA to perform well depending on the size of the problem instances. Among the mutation operators implemented, the flip mutation made a significant difference improving the performance of the GA, particularly on smaller instances.

Following the experiments on the GA settings, further experiments with longer computation times were executed using the identified GA settings. Also, another set of 7 problem instances was used, these are the largest of all problems. Results obtained with the proposed baseline GA were compared

Table IV. Comparison of Overall Results for All Solution Methods on All Problem Instances.

		Optimal		GA		VNS		MIP DECOMP.				Optimal		GA		VNS		MIP DECOMP.	
		f(S)	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*			*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*
	01	**3.5**	7.0	**3.5**	4.5	**3.5**	**3.0**	5.7	3.7		**01**	-	-	171.7	3414.5	**170.3**	3036.2	496.4	**1060.0**
	02	**2.5**	8.0	**2.5**	20.6	**2.5**	**0.3**	4.5	3.6		**02**	-	-	168.2	3456.5	**163.9**	2840.8	372.9	**1192.1**
	03	**3.0**	14.0	3.2	169.2	**3.0**	29.3	10.7	**3.7**		**03**	-	-	181.7	3473.7	**178.2**	3172.8	3213.3	**1209.0**
A	**04**	**1.4**	5.0	**1.4**	238.5	**1.4**	6.9	3.1	**2.9**	**D**	**04**	-	-	170.2	3438.0	**167.1**	3313.8	418.9	**3005.1**
	05	**2.4**	1.0	**2.4**	1.3	**2.4**	**0.1**	3.5	1.8		**05**	-	-	162.0	3499.8	**161.1**	2818.5	243.9	**1306.7**
	06	**3.5**	5.0	**3.5**	90.0	3.6	29.5	3.7	**2.4**		**06**	-	-	178.4	3532.3	**177.4**	2996.0	1411.3	**1222.0**
	07	**3.7**	1.0	**3.7**	0.9	**3.7**	**0.1**	4.8	1.6		**07**	-	-	178.7	3465.3	**177.9**	2930.4	753.3	**1361.9**
	01	**1.7**	21.0	**1.7**	449.0	**1.7**	156.5	1.8	**8.1**		**01**	-	-	**1.2**	**3435.6**	-	-	33.0	8408.0
	02	**1.8**	2.0	**1.8**	0.8	**1.8**	**0.0**	1.9	4.3		**02**	-	-	**1.2**	**3512.8**	-	-	26.0	12448.4
	03	**1.7**	6003.0	1.8	444.7	**1.7**	473.2	2.1	**32.9**		**03**	-	-	**1.2**	**3536.2**	-	-	29.0	20746.6
B	**04**	**2.1**	25.0	**2.1**	118.2	**2.1**	**9.0**	2.2	15.2	**E**	**04**	-	-	**1.3**	**3511.8**	-	-	28.5	15190.5
	05	**1.8**	585.0	1.9	328.2	**1.8**	135.3	4.7	**25.3**		**05**	-	-	**2.2**	**3460.7**	-	-	270.1	32619.2
	06	**1.6**	184.0	1.7	354.0	**1.6**	194.3	2.5	**24.1**		**06**	-	-	**1.3**	**3350.1**	-	-	24.6	24212.1
	07	**1.8**	300.0	**1.8**	315.9	**1.8**	305.0	4.1	**23.6**		**07**	-	-	**1.7**	**3502.1**	-	-	427.8	51057.3
	01	-	-	118.0	939.5	**114.2**	301.3	905.0	**211.6**		**01**	-	-	**2109.0**	**28628.6**	-	-	64305.1	3446.4
	02	**3.2**	6.0	**3.2**	**0.0**	**3.2**	**0.0**	3.6	0.6		**02**	-	-	**2116.4**	**28418.7**	-	-	73291.2	1111.0
	03	-	-	105.0	861.2	**103.5**	550.3	1186.3	**26.3**		**03**	-	-	**609.4**	**28591.2**	-	-	115235.2	4555.1
C	**04**	**11.1**	90.0	**11.1**	5.2	11.2	**0.9**	81.3	3.1	**F**	**04**	-	-	**1331.5**	**28256.4**	-	-	102994.2	4219.2
	05	**12.3**	55.0	12.5	96.7	**12.3**	**0.3**	68.9	1.1		**05**	-	-	**200.6**	**28667.2**	-	-	101438.2	6156.5
	06	-	-	142.0	847.0	**140.4**	323.5	3102.3	**47.0**		**06**	-	-	**627.2**	**28181.4**	-	-	76007.1	9695.6
	07	**4.3**	1.0	**4.3**	**0.0**	**4.3**	**0.0**	5.3	0.2		**07**	-	-	**3499.1**	**28262.4**	-	-	176540.6	3832.8

to published results. The comparison showed that the GA is capable of finding some best-known results for the larger problem instances and matching the best-known results for some of the smaller ones. Hence, the proposed baseline GA with problem-specific components is an effective evolutionary approach to solve the highly-constrained WSRP scenarios considered. Future work will aim to improve the performance of this baseline GA.

References

[1] R. Lassaigne and M. De Rougemont, *Logic and complexity*. Springer Science & Business Media, 2012.

[2] R. L. Pinheiro, D. Landa-Silva, and J. Atkin, "A variable neighbourhood search for the workforce scheduling and routing problem," *Proceedings of the 7th World Congress on Nature and Biologically Inspired Computing (NaBIC 2015)*, December 2015.

[3] T. Urlings, R. Ruiz, and F. S. Serifoglu, "Genetic algorithms with different representation schemes for complex hybrid flexible flow line problems," *International Journal of Metaheuristics*, vol. 1, no. 1, pp. 30–54, 2010.

[4] G. Zhang, L. Gao, and Y. Shi, "An effective genetic algorithm for the flexible job-shop scheduling problem," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3563–3573, 2011.

[5] P. Cowling, N. Colledge, K. Dahal, and S. Remde, "The trade off between diversity and quality for multi-objective workforce scheduling," in *Evolutionary Computation in Combinatorial Optimization*, ser. Lecture Notes in Computer Science, J. Gottlieb and G. Raidl, Eds. Springer Berlin Heidelberg, 2006, vol. 3906, pp. 13–24.

[6] M. Mutingi and C. Mbohwa, "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, 2014.

[7] H. Algethami and D. Landa-Silva, "A study of genetic operators for the workforce scheduling and routing problem," *MIC 2015 conference proceedings*, 2015.

[8] J. A. Castillo-Salazar, D. Landa-Silva, and R. Qu, "Workforce scheduling and routing problems: literature survey and computational study," *Annals of Operations Research*, pp. 1–29, 2014.

[9] J. A. Castillo-Salazar and D. Landa-Silva, "A greedy heuristic for workforce scheduling and routing with time-dependent activities constraints," *Proceedings of the 4th International Conference on Operations Research and Enterprise Systems (ICORES 2015)*, pp. 367–375, January 2015.

[10] W. Laesanklang, D. Landa-Silva, and J. A. Castillo-Salazar, "Mixed integer programming with decomposition for workforce scheduling and routing with time-dependent activities constraints," in *Proceedings of the 5th International Conference on Operations Research and Enterprise Systems (ICORES 2016)*, February 2016, pp. 330–339, published in: Proceedings of the 5th International Conference in Operations Research and Enterprise Systems. ISBN 9789897581717, pp. 330-339.

[11] W. Laesanklang, R. Pinheiro, H. Algethami, and D. Landa-Silva, "Extended decomposition for mixed integer programming to solve a workforce scheduling and routing problem," in *Operations Research and Enterprise Systems*, ser. Communications in Computer and Information Science, D. de Werra, G. H. Parlier, and B. Vitoriano, Eds. Springer International Publishing, 2015, vol. 577, pp. 191–211.

[12] R. Cheng, M. Gen, and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithmsi. representation," *Computers & industrial engineering*, vol. 30, no. 4, pp. 983–997, 1996.

[13] F. Rothlauf, "Representations for genetic and evolutionary algorithms," *Studies in Fuzziness and Soft Computing*, vol. 104, pp. 9–32, 2003.

[14] U. Aickelin and K. A. Dowsland, "An indirect genetic algorithm for a nurse-scheduling problem," *Computers & Operations Research*, vol. 31, no. 5, pp. 761 – 778, 2004.

[15] C. Akjiratikarl, P. Yenradee, and P. R. Drake, "Pso-based algorithm for home care worker scheduling in the {UK}," *Computers & Industrial Engineering*, vol. 53, no. 4, pp. 559–583, 2007.

[16] D. Mankowska, F. Meisel, and C. Bierwirth, "The home health care routing and scheduling problem with interdependent services," *Health Care Management Science*, vol. 17, no. 1, pp. 15–30, 2014.

[17] S. Hartmann, "A competitive genetic algorithm for resource-constrained project scheduling," *Naval Research Logistics (NRL)*, vol. 45, no. 7, pp. 733–750, 1998.

[18] S. N. Sivanandam and S. N. Deepa, "Terminology and operators of ga," in *Introduction to genetic algorithms*, 1st ed. Springer, 2007.

[19] E. K. Burke, J. P. Newall, and R. F. Weare, "Initialization strategies and diversity in evolutionary timetabling," *Evolutionary computation*, vol. 6, no. 1, pp. 81–103, 1998.

[20] W. Laesanklang and D. Landa-Silva, "Mixed integer programming with decomposition to solve a workforce scheduling and routing problem," *Proceedings of the 4th International Conference on Operations Research and Enterprise Systems (ICORES 2015)*, pp. 283–293, January 2015.