A Variable Neighbourhood Search for the Workforce Scheduling and Routing Problem

Rodrigo Lankaites Pinheiro, Dario Landa-Silva and Jason Atkin

Abstract The workforce scheduling and routing problem (WSRP) is a combinatorial optimisation problem where a set of workers must perform visits to geographically scattered locations. We present a Variable Neighbourhood Search (VNS) metaheuristic algorithm to tackle this problem, incorporating two novel heuristics tailored to the problem-domain. The first heuristic restricts the search space using a priority list of candidate workers and the second heuristic seeks to reduce the violation of specific soft constraints. We also present two greedy constructive heuristics to give the VNS a good starting point. We show that the use of domain-knowledge in the design of the algorithm can provide substantial improvements in the quality of solutions. The proposed VNS provides the first benchmark results for the set of real-world WSRP scenarios considered.

Keywords Workforce scheduling and routing problems · Home healthcare scheduling · Variable neighbourhood search · constructive heuristics

1 Introduction

In workforce scheduling and routing problems (WSRP) a mobile workforce must perform tasks in scattered geographical locations. Solving this problem requires defining a schedule and a route plan for each worker such that all tasks (where possible) are covered. These problems combine features from both scheduling and routing problems, making them very challenging optimisation problems [1–3]. Examples of

R.L. Pinheiro (✉) · D. Landa-Silva · J. Atkin
School of Computer Science, ASAP Research Group, The University of Nottingham, Nottingham, UK
e-mail: psxrp2@nottingham.ac.uk

D. Landa-Silva
e-mail: pszjds@nottingham.ac.uk

J. Atkin
e-mail: pszja@nottingham.ac.uk

N. Pillay et al. (eds.), *Advances in Nature and Biologically Inspired Computing*,
Advances in Intelligent Systems and Computing 419,
DOI 10.1007/978-3-319-27400-3_22

practical applications of WSRP are home healthcare scheduling [4, 5], technician scheduling [6, 7] and security personnel scheduling [8]. Here we consider the problem of scheduling nurses and care workers to visit and provide care services to patients in their homes. Data from four distinct home healthcare companies is used, provided by our industrial partner, who provide an enterprise resource planning software for home healthcare companies. This software includes a scheduling tool where a decision maker must manually set the schedules and routes for each worker; our work aims to automate this process.

The four real-world scenarios used here have been tackled before. Exact methods for these problems were investigated in [3], and the large size and complexity of the mixed integer programming model required a decomposition method before applying an exact solver. A study of genetic operators for the same problem scenarios was later performed in [9]. These two previous studies focused on understanding the problem scenarios and the behaviour of specific solution techniques rather than providing benchmark results.

Here we present a Variable Neighbourhood Search (VNS) algorithm to tackle the aforementioned scenarios and provide benchmark results. VNS algorithms have been successfully applied to both scheduling and routing problems, e.g. nurse scheduling [10, 11], job shop scheduling [12], vehicle routing problem with time windows [13] and the multi-depot version of the vehicle routing problem with time windows [14], a problem inherently similar to the WSRP.

The proposed VNS employs two domain-specific local search neighbourhoods. The first one sorts the workers by priority, identifying the best workers for each task and restricting the search space to the highest priority workers. The second one attempts to eliminate time and area violations from a solution. As mentioned above, results for the WSRP scenarios considered here are not currently available for many techniques. We therefore compare the variants of our VNS against each other, and also propose two constructive greedy heuristics to generate initial solutions quickly, showing that the proposed VNS outperforms simpler versions with less domain-knowledge integrated.

The contributions of this paper are twofold. The first is the proposed VNS algorithm that produces benchmark best known solutions for these real-world WSRP scenarios. The second is an improved understanding of the WSRP, obtained through assessing both the performance of the proposed algorithms and the impact of the tailored techniques which incorporate the domain knowledge. The remainder of this paper is structured as follows: Sect. 2 presents an overview of the WSRP. Section 3 describes the proposed algorithmic approach. Section 4 details the experiments and presents the results, while Sect. 5 concludes the paper.

2 The Workforce Scheduling and Routing Problem

For reasons of space, the WSRP is only briefly explained in this section. More details and a survey can be found in [15], while a full mathematical formulation of the problem can be found in [3]. The WSRP involves both scheduling and routing. A set

Assignment	a_1	a_2	a_3	a_4	...	a_s
Visit	v_1	v_1	v_2	v_3	...	v_n
Worker	w_5	w_1	w_3	w_9	...	w_5

Fig. 1 Example of solution representation. Note that two distinct workers are assigned to visit v_1

of m workers $\{w_1, w_2, \dots, w_m\}$, must perform tasks at a set of n visits $\{v_1, v_2, \dots, v_n\}$, which are located at various geographical locations. Each worker possesses a set of skills, time and area availabilities, and working area preferences. Visits have both required and preferred skills, and possibly have specific preferred workers. A worker-visit match requires matching the required skills, the worker's contract must allow him/her to perform that visit, and the specific allocation may incur additional costs.

A solution for a WSRP instance is a set of s assignments or pairs (v_j, w_i), specifying that worker w_i $(i \in \{1 \dots m\})$ is assigned to perform visit v_j $(j \in \{1 \dots n\})$. Note that $s \geq n$ because some visits might require more than one worker. An example of a solution is shown in Fig. 1, where workers 5 and 1 are assigned to visit 1, worker 5 is also assigned to visit n, and workers 3 and 9 are assigned to visits 2 and 3 respectively.

There are requirements that should be met if possible when assigning workers to visits. Such visit requirements include preferred skills (patient preferences), preferred workers (service provider preferences) and preferred working areas (staff preferences). Also, the workers availability in terms of time and geographical areas should be observed. A solution requires all visits to be served hence it is not always possible to meet all visit requirements and workers availability. To evaluate the quality of a solution, the tier-based minimisation objective function shown in Eq. (1) is utilised, which is employed by our industrial partner and also commonly used in the literature [2, 16].

$$f(S) = \lambda_1(d + c) + \lambda_2(3s - \rho_s - \rho_w - \rho_a) + \lambda_3(\psi_a + \psi_t) + \lambda_4\omega + \lambda_5\phi \qquad (1)$$

The objective function has five main components each multiplied by a coefficient $(\lambda_1, \dots \lambda_5)$ to enforce tier-based objectives, i.e. the component multiplied by λ_5 is more important than the one multiplied by λ_4, and so on. The first component represents operational cost in terms of total travel distance (d) and staff cost (c). The second component represents visit requirements for preferred skills (ρ_s), workers (ρ_w) and areas (ρ_a). The value $3s$ is used because the values of ρ_s, ρ_w and ρ_a can be between 0 and 1 for each of the s visits. The third component represents the number of violations of workers availability in terms of area (ψ_a) and time (ψ_t). The fourth and fifth components represent the number of unassigned visits (ω) and the number of time conflicts (ϕ) respectively; a time conflict occurs when a worker is assigned to visits overlapping in time.

3 Proposed Algorithms

A VNS algorithm is proposed for solving the instances of the WSRP which were provided by our industrial partner and a number of algorithm variants are considered. VNS is an improvement metaheuristic proposed in [17]. It starts from an initial solution and performs successive local searches using multiple neighbourhoods to improve the solution. In order to escape local optima, VNS randomly disturbs the current solution (possibly making it worse) at the end of each iteration.

The VNS variant used in this paper has two stages, which are repeated: a *shaking phase* and a *local search phase*. In the *shaking phase*, one of seven shaking neighbourhoods is randomly selected and applied. If no change is made to the solution, another shaking neighbourhood is selected. This process is repeated until a change is made to the solution. The changed solution is then passed to the *local search phase*, which uses two neighbourhood search operators to hopefully generate better neighbouring solutions. One iteration of the local search phase consists of applying the two neighbourhood searches in some random order. If an improved solution is obtained from the current iteration, then another local search iteration (execution of both neighbourhood searches) takes place. When no improvement has been achieved in an iteration, the algorithm goes back to the shaking phase. We evaluate different configurations of the local search in our experiments.

3.1 Shaking Neighbourhood Structures

Solutions generated with the seven shaking neighbourhoods described below may be infeasible (e.g. one worker assigned to two simultaneous visits), but will still be kept.

Random Flip. Randomly picks a visit and changes the assigned worker to a random different worker that is skilled to perform that visit.

Area Availability Flip. Randomly picks a visit where the area availability constraint is violated and attempts to fix the violation by picking any other worker that is skilled to perform that visit and is available to work in that area.

Time Availability Flip. Randomly picks a visit where the time availability constraint is violated and attempts to fix the violation by picking any other worker that is skilled to perform that visit and is available to work at the visit time.

Preferred Worker Flip. Randomly picks a visit where the assigned worker is not the most preferred and replaces the worker by the most preferred worker for the visit.

Preferred Skills Flip. As *Preferred Worker Flip*, but uses the preferred skills value.

Preferred Area Flip. As *Preferred Worker Flip*, but uses the preferred areas value.

Priority-Based Flip. Uses a priority list (defined in Sect. 3.3). Selects a random visit for which the currently assigned worker is not the top one in the priority list and assigns the top priority worker instead.

3.2 Neighbourhood Searches

We define the following three neighbourhood searches to be evaluated as the operators used in the local search phase of the proposed VNS algorithm.

Randomised Hill Climbing (RHC). This is a very simple hill climbing local search which iteratively processes all unassigned visits in random order, greedily selecting the best worker in terms of the overall cost $f(S)$ and assigning that worker to the visit.

Priority-Based Search (PBS). This is detailed in Sect. 3.3 and exploits problem domain knowledge to make an estimation of the overall cost $f(S)$. Such estimation considers the objective function components associated to λ_1 (except travel distance), λ_2 and λ_3.

Availability Violations Search (AVS). It was observed that minimising violations of time and area workers availability is very difficult to achieve for the majority of the scenarios and that high costs resulted from these violations. This local search aims to resolve this and is explained in detail in Sect. 3.4.

3.3 Priority-Based Search (PBS)

This search is applied to prioritise workers for each visit. The concept is shown in Fig. 2 and involves the following steps:

1. An $m \times n$ cost matrix C is defined, containing the estimated costs of assigning each worker w_i to each visit v_j. As explained above, an estimation of $f(S)$ is used for each assignment. More specifically, the cost c_{ij} for each assignment in C is the weighted sum of the *staff costs* (greedily selecting the cheapest contract); preferred *skills*, *workers* and *areas*; and workers availability for *area* and *time*. For each assignment where w_i does not have the required skills or a valid contract to perform the visit v_j we set $c_{ij} = \infty$.
2. A priority list P^j is built for each visit v_j, sorting the workers into ascending order of c_{ij}. All visits are recorded as 'unmarked'.
3. Pick a random unmarked visit, select it for use in steps 4 to 6 and mark it.
4. Use P^j to determine if there is a worker with lower costs for the current visit v_j.
5. If one or more such workers exist, pick the first in the list and check if assigning that worker will generate a time conflict.

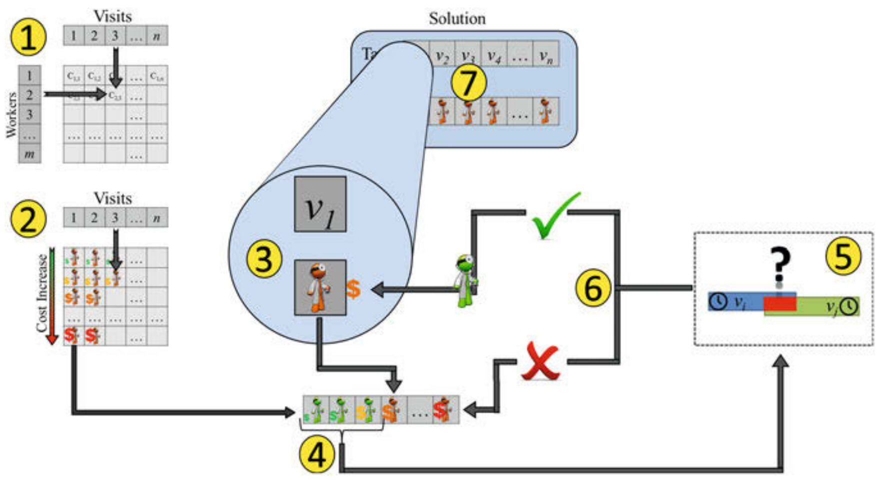

Fig. 2 Diagram for the Priority-Based Search (PBS)

6. If the assignment can be done (no time conflict), then assign the worker and evaluate the solution. If this reduces the cost, the assignment is accepted, otherwise the assignment is reverted. If the assignment generates a time conflict (hence reverted), repeat from step 5, selecting the next valid worker on the list.
7. Repeat from step 3 until all visits have been marked.

3.4 Availability Violations Search (AVS)

The concept of this search is presented in Fig. 3 and involves the following steps:

1. For each visit, identify the candidate worker with the minimum number of time and area availability violations for that visit. If multiple workers meet that criterion, choose the one with the lowest value in the cost matrix C used by the PBS.
2. Pick an unmarked visit in which the number of area and time availability violations is larger than the selected candidate for that visit. Mark the current visit.
3. Replace the current assigned worker with the candidate worker.
4. Identify whether time conflicts were created in step 3.
5. Where time conflicts occur, each conflicting visit is unassigned then perform steps 4 to 6 of the PBS heuristic to find a new worker for the visit. This will eliminate all time conflicts.
6. Evaluate the solution. If the costs improved (compared with those prior to step 3), accept the changes, otherwise revert the changes.
7. Repeat from step 2 while there are still visits for which candidates were identified in step 1.

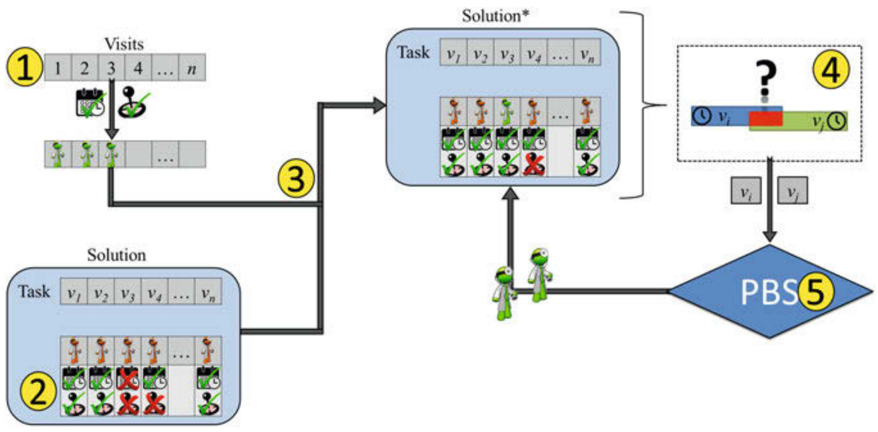
Visits
1
1 2 3 4 … n
…
3
Solution
Task v_1 v_2 v_3 v_4 … v_n
2
Solution*
Task v_1 v_2 v_3 v_4 … v_n
?
4
v_i
v_j
PBS
5

Fig. 3 Diagram for the Availability Violations Seeker Search

3.5 *Constructive Heuristics*

VNS algorithms are well known for producing faster and better results when given improved initial solutions. As suggested in [1], constructive heuristics were considered for finding initial feasible solutions.

Greedy Heuristic. This simple greedy heuristic achieved competitive results on its own (as shown in Sect. 4). The algorithm starts with an empty solution and considers each visit in a random order, choosing the best of the remaining workers for that visit, i.e. the worker that results on the best value of $f(S)$.

Flat Costs Heuristic (FCH). This heuristic performs a greedy search using the estimation of the overall cost $f(S)$ based on the matrix C as explained in the PBS method. The heuristic iterates through the visits in a random order and identifies the best (lowest cost in C) of the compatible remaining workers (skills and times match) for the visit, then assigning that worker to the visit.

4 Experiments and Results

We use four real-world scenarios provided by an industrial partner. Each dataset, A, B, C and D, is composed of 7 instances for a total of 28 problem instances. These scenarios come from different home healthcare companies, hence having different requirements and features. Set A has small instances (number of visits and workers) while set D has the largest instances.

All experiments were performed on Intel quad-core i7 machines with 16GB DDR2 RAM memory and each algorithm was executed eight times, computing the average solution. For scenarios A, B and C the runtime limit was set to 15 minutes, for scenarios D it was one hour. The following experiments were performed to evaluate the VNS algorithm.

4.1 Evaluation of Individual Components

The performance of the PBS and AVS local searches was evaluated. Four configurations of the algorithm were considered, all starting from the same feasible solution which was obtained by the constructive heuristics:

- **Full-VNS** is the full proposed algorithm including both PBS and AVS local searches and all seven shaking neighbourhoods.
- **PBS-VNS** is the Full-VNS without AVS (using only PBS as the local search).
- **AVS-VNS** is the Full-VNS without PBS (using only AVS as the local search).
- **HC-VNS** uses only six shaking neighbourhoods (excluding the Priority-based Flip) and uses only RHC as the local search (not PBS or AVS).

Results are shown in Fig. 4, where the *y* axis presents the gap between the average solution found by each algorithm for each of the problem instances in each dataset, compared to the best solution found by all runs of all algorithms for that instance. For some instances (sets A and B and instances C2, C4, C5 and C7), this is also the optimal solution, obtained by a mathematical solver. All three algorithm variants perform well for the smaller sets (A and B). However, for sets C and D PBS-VNS and AVS-VNS alternate, showing that on some scenarios one local search has an edge over the other. Notably, the AVS provided better solutions than the PBS for scenarios where the number of time and area availability violations is high, whereas PBS provided better solutions for the remaining scenarios. Importantly, the Full-VNS produced better results overall, especially for sets C and D.

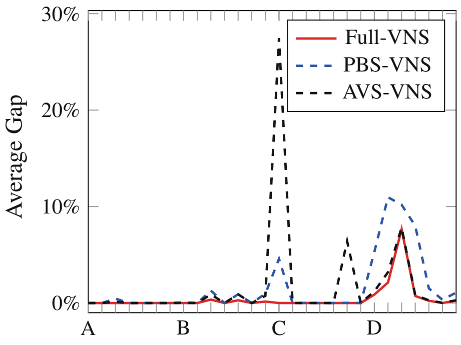

Fig. 4 Relative gap comparison between the Full-VNS, the PBS-VNS and the AVS-VNS

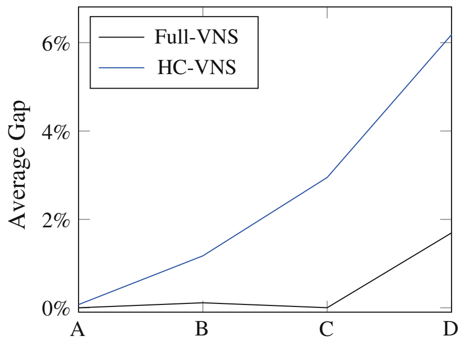

Fig. 5 Relative gap comparison between the Full-VNS and the HC-VNS

4.2 Overall Performance

The overall performance of the Full-VNS algorithm was evaluated and compared to HC-VNS. Results are shown in Fig. 5. We can see that HC-VNS produced good solutions (the gap is always below 7 %). However, its performance worsens as the size of the problem grows. This is due to hill climbing being slow for this problem. A single iteration of the HC-VNS algorithm for the largest D instance took over ten minutes while the Full-VNS could iterate in less than one minute. The proposed VNS produced better solutions for all instances, and did so in a faster computation time than HC-VNS. Additionally, the proposed VNS was able to reach the optimal solution for all instances for which the optimal solution is known. None of the other algorithm variants were able to do this. Full detailed results are shown in Table 1, where the best average results obtained by the VNS variants for each dataset are shown in bold.

4.3 Constructive Heuristics

The performance of the constructive heuristics was also evaluated. Figure 6 shows the average gaps obtained by the constructive algorithms alone (without the VNS). Both techniques show similar performance, with the Greedy outperforming the FCH on sets B and C and the FCH outperforming the Greedy on set D, however, the difference is always small. Figure 7 presents a comparison between the runtimes for the Greedy and the FCH heuristics. Both are fast for sets A, B and C, providing solutions in less than a second. The greedy algorithm is much slower for set D (up to 107 seconds compared to less than one second for FCH).

Table 1 Average results for each algorithm

		Greedy	FCH	HC-VNS	PBS-VNS	AVS-VNS	Full-VNS	Best Known[a]			Greedy	FCH	HC-VNS	PBS-VNS	AVS-VNS	Full-VNS	Best Known[a]
A	**1**	3.812	3.812	**3.487**	**3.487**	**3.487**	**3.487**	*3.487*	**C**	**1**	148.65	149.09	141.59	119.69	157.47	**114.21**	*114.21*
	2	3.355	3.519	**2.491**	**2.491**	**2.491**	**2.491**	*2.491*		**2**	**3.05**	**3.05**	**3.05**	**3.05**	**3.05**	**3.05**	*3.05*
	3	4.908	4.999	3.007	3.009	3.000	**2.995**	*2.995*		**3**	141.79	135.51	104.37	103.54	**103.52**	**103.52**	*103.52*
	4	1.568	1.562	1.422	**1.421**	**1.421**	**1.421**	*1.421*		**4**	11.48	11.68	11.15	**11.15**	**11.15**	**11.15**	*11.15*
	5	3.103	2.959	**2.420**	**2.420**	**2.420**	**2.420**	*2.420*		**5**	12.81	12.89	**12.34**	**12.34**	**12.34**	**12.34**	*12.34*
	6	3.632	3.916	**3.549**	**3.549**	**3.549**	**3.549**	*3.549*		**6**	176.19	181.58	141.13	140.47	150.07	**140.44**	*140.44*
	7	4.345	4.102	**3.714**	**3.714**	**3.714**	**3.714**	*3.714*		**7**	**4.30**	**4.30**	**4.30**	**4.30**	**4.30**	**4.30**	*4.30*
B	**1**	1.861	1.897	1.705	**1.703**	1.704	**1.703**	*1.703*	**D**	**1**	178.17	177.98	180.62	178.30	170.92	**170.33**	*168.84*
	2	1.812	1.825	**1.755**	**1.755**	**1.755**	**1.755**	*1.755*		**2**	179.37	178.63	183.62	180.12	165.66	**163.85**	*160.36*
	3	2.011	2.013	1.779	1.740	1.733	**1.724**	*1.718*		**3**	185.18	185.18	183.98	183.23	178.53	**178.19**	*164.56*
	4	2.131	2.130	2.077	**2.074**	**2.074**	**2.074**	*2.074*		**4**	178.44	178.75	183.77	180.33	167.27	**167.11**	*165.89*
	5	2.027	2.080	1.859	1.840	1.840	**1.828**	*1.823*		**5**	164.69	164.51	163.36	163.22	161.20	**161.11**	*160.74*
	6	1.835	1.864	1.638	**1.620**	**1.620**	**1.620**	*1.620*		**6**	179.50	179.93	178.46	178.02	177.46	**177.43**	*177.42*
	7	1.966	2.010	1.816	1.807	1.803	**1.792**	*1.790*		**7**	180.17	180.48	180.32	179.37	177.94	**177.86**	*177.42*

[a]Best solutions found among all runs of all algorithms

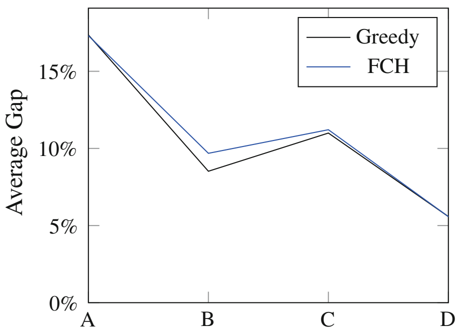

Fig. 6 Performance comparison of the FCH heuristic against the Greedy heuristic

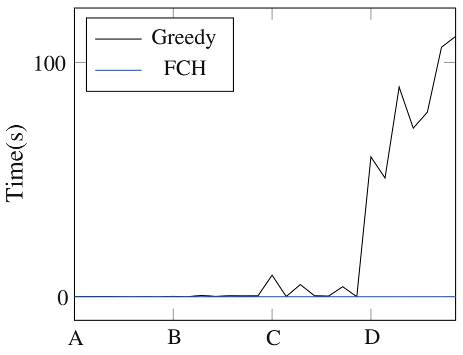

Fig. 7 Time comparison between the FCH heuristic and the Greedy heuristic

4.4 Discussion

The sizes of the instances in set D provoke unacceptably high runtimes for a simple greedy algorithm, potentially hindering the performance of search methodologies that rely on the systematic exploration of neighbourhoods (such as the HC-VNS). The design of the Priority-Based Search (PBS) proved to be much more efficient, allowing a much faster exploration of the neighbourhoods.

The constructive heuristics produce reasonable results compared to the other techniques, with the average gap to the best known solution being roughly 9.5 %. This is due to the structure of the problem, which favours the assignment of the best worker to each visit unless there are time conflicts. The preferred worker requirement is related to the continuity of care, i.e. the same worker providing care to the same patient. So, a worker that is preferred for a visit is usually the worker who most often performs the visits for that patient. Thus, a worker with a high preference value is often also available for that area and time, and is likely to live nearby. In fact, since the FCH provided results only 9 % worse on average than the proposed VNS, this constructive heuristic may be sufficient to provide good enough solutions very quickly, illustrating the efficacy of the developed approach for combining the objectives considered here.

Both PBS-VNS and AVS-VNS outperformed the HC-VNS. Since PBS outperformed RHC in all scenarios (even the small ones where runtime was less of an issue), the restriction of workers by the priority list does not appear to have eliminated the best candidates for assignments. Moreover, the AVS heuristic, which fixes the time and area availabilities using the first worker available in the priority list, could also reach the optimal and provide some of the best results. Comparing the results for the PBS and the AVS shows differences between sets. The latter technique performs 3.5 % better for set D. This shows that the occurrence of area and time violations are common in these datasets and that a focus on fixing these violations is beneficial. On the other hand, for set C the priority-based approach performs 4 % better.

Finally, when comparing all algorithm variants, it can be observed that increased domain knowledge gives improved results. HC-VNS includes some domain-specific shaking methods, giving reasonably good solutions. AVS-VNS and PBS-VNS use more domain-specific knowledge, giving even better results. The full proposed VNS makes the most use of domain-knowledge and provides the best results overall.

5 Conclusion

Heuristic algorithms to tackle difficult instances of workforce scheduling and routing problems (WSRP) were presented. First we introduced a VNS that employs two domain specific local search procedures. The first local search procedure attempts to reduce the search space while focusing on the interesting assignments. The second local search procedure tries to reduce the number of area and time availability violations. We also introduced two greedy heuristics—one straightforward and another that uses an estimation of costs to provide faster results.

We assessed the proposed algorithms on a set of real-world problem instances. We showed that it may be difficult for the VNS to find a feasible solution in some scenarios, hence the use of constructive heuristics is a useful strategy. We also showed that each local search procedure can perform well on its own, however the combined effect of both local searches provides improved results. Additionally, we observed that the greedy algorithms show good performance on this problem, considering the trade-off *quality versus computation time*. Finally, we discussed how the algorithms exhibited distinct performances on different sets of the WSRP, which allowed us to understand more of the nature of the problem and its difficulties. It became clear that adding problem domain knowledge to the solution algorithms improves their performance.

Our future work will further investigate mechanisms to make a fast yet accurate enough estimation of costs when exploring local moves and neighbour solutions. The results in this paper show that much computational effort can be avoided by using this technique and that the quality of the solutions is not affected much. We could apply this concept to other local search procedures for the VNS or even exact methods [3].

References

1. Castillo-Salazar, J.A., Landa-Silva, D., Qu, R.: Computational study for workforce scheduling and routing problems. In: ICORES 2014—Proceedings of the 3rd International Conference on Operations Research and Enterprise Systems, pp. 434–444 (2014)
2. Castillo-Salazar, J.A., Landa-Silva, D., Qu, R.: Workforce scheduling and routing problems: literature survey and computational study. Ann. Oper. Res. (2014)
3. Laesanklang, W., Landa-Silva, D., Salazar, J.A.C.: Mixed integer programming with decomposition to solve a workforce scheduling and routing problem. In: Proceedings of the International Conference on Operations Research and Enterprise Systems, pp. 283–293 (2015)
4. Cheng, E., Rich, J.L.: A home health care routing and scheduling problem (1998)
5. Akjiratikarl, C., Yenradee, P., Drake, P.R.: Pso-based algorithm for home care worker scheduling in the UK. Comput. Ind. Eng. **53**, 559–583 (2007)
6. Xu, J., Chiu, S.: Effective heuristic procedures for a field technician scheduling problem. J. Heuristics **7**(5), 495–509 (2001)
7. Cordeau, J.F., Laporte, G., Pasin, F., Ropke, S.: Scheduling technicians and tasks in a telecommunications company. J. Sched. **13**(4), 393–409 (2010)
8. Misir, M., Smet, P., Verbeeck, K., Vanden Berghe, G.: Security personnel routing and rostering: a hyper-heuristic approach. In: Proceedings of the 3rd International Conference on Applied Operational Research, vol. 3 (2011)
9. Algethami, H., Landa-Silva, D.: A study of genetic operators for the workforce scheduling and routing problem. In: Proceedings of the XI Metaheuristics International Conference (MIC 2015) (2015)
10. Burke, E., De Causmaecker, P., Petrovic, S., Berghe, G.: Variable neighborhood search for nurse rostering problems. In: Metaheuristics: Computer Decision-Making. Applied Optimization. vol. 86, pp. 153–172. Springer (2004)
11. Constantino, A.A., Tozzo, E., Pinheiro, R.L., Landa-Silva, D., Romão, W.: A variable neighbourhood search for nurse scheduling with balanced preference satisfaction. In: 17th International Conference on Enterprise Information Systems (ICEIS 2015), Barcelona, Spain, Scitepress, Scitepress (2015)
12. Roshanaei, V., Naderi, B., Jolai, F., Khalili, M.: A variable neighborhood search for job shop scheduling with set-up times to minimize makespan. Future Gen. Comput. Syst. **25**(6), 654–661 (2009)
13. Bräysy, O.: A reactive variable neighborhood search for the vehicle routing problem with time windows. INFORMS J. Comput. **15**, 347–368 (2003)
14. Polacek, M., Hartl, R.F., Doerner, K., Reimann, M.: A variable neighborhood search for the multi depot vehicle routing problem with time windows. J. Heuristics **10**(6), 613–627 (2004)
15. Castillo-Salazar, J.A., Landa-Silva, D., Qu, R.: A survey on workforce scheduling and routing problems. In: Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012), pp. 283–302. Son, Norway, August 2012
16. Rasmussen, M.S., Justesen, T., Dohn, A., Larsen, J.: The home care crew scheduling problem: preference-based visit clustering and temporal dependencies. Eur. J. Oper. Res. **219**(3), 598–610 (2012)
17. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)