Approximate Dynamic Programming With Combined Policy Functions for Solving Multi-Stage Nurse Rostering Problem

Peng Shi and Dario Landa-Silva

School of Computer Science, ASAP Research Group
The University of Nottingham, United Kingdom
{peng.shi, dario.landasilva}@nottingham.ac.uk

Abstract. An approximate dynamic programming that incorporates a combined policy, value function approximation and lookahead policy, is proposed. The algorithm is validated by applying it to solve a set of instances of the nurse rostering problem tackled as a multi-stage problem. In each stage of the problem, a weekly roster is constructed taking into consideration historical information about the nurse rosters in the previous week and assuming the future demand for the following weeks as unknown. The proposed method consists of three phases. First, a pre-process phase generates a set of valid shift patterns. Next, a local phase solves the weekly optimization problem using value function approximation policy. Finally, the global phase uses lookahead policy to evaluate the weekly rosters within a lookahead period. Experiments are conducted using instances from the Second International Nurse Rostering Competition and results indicate that the method is able to solve large instances of the problem which was not possible with a previous version of approximate dynamic programming.

Keywords: Dynamic Programming, Approximation Function, Policy Function, Nurse Scheduling Problem

1 Introduction

This paper investigates the ability of approximate dynamic programming using a combined policy function to tackle a multi-stage nurse rostering problem. Approximate dynamic programming (ADP) is designed to tackle the Markov Decision Process that dynamic programming is unable to solve in practice [1]. ADP aims to learn the selection of the optimal policy for mapping the state space into the action space. The purpose of policies in ADP is to determine decisions. The technique presented here is a hybrid approach that combines the lookahead policy and the value function approximation policy. The *lookahead policy* makes decisions now by explicitly optimizing over some time horizon by combining some approximation of future information while the *value function approximation policy* refers to an approximation of the value of being in a future state as a result of a decision made now [2].

The Nurse Rostering Problem (NRP) is an NP-Hard problem that consists in constructing rosters for a number of nurses over a time horizon of typically no more than a few weeks. Constructing a roster involves assigning shifts types of each nurse for each day in order to fulfill daily duty requirements plus satisfying a number of soft and hard constraints [3]. In this paper, the NRP is tackled as a multi-stage optimisation problem is used to test the proposed technique because it is a widely investigated problem and presents an interesting challenge to ADP. Tackling the NRP as a multi-stage problem was proposed by [4].

Solving the NRP with dynamic programming is impractical due to the curse of dimensionality [2, 5]. Our previous work investigated ADP to solve NRP, where a value function approximation based method was proposed to tackle various instances of the NRP [6]. However, the computation time required for constructing solution samples and the memory space required for recording rewards increased exponentially for larger problem instances. Hence, that shortfall has motivated the present work. A number of ADP practical issues related to the complexity of the environment, in particular when dealing with large state or action space, are reported in the literature [5]. The technique proposed in this paper enhances the ability of ADP to solve NRP as a multi-stage problem by combining two policy functions, value function approximation to solve the weekly problem, and lookahead policy to evaluate weekly rosters with artificially constructed future demand within a given lookahead period.

The contribution of this paper is an enhanced approximate dynamic programming approach that takes advantage of tackling the NRP in multiple stages and is able to tackle instances of this problem with longer planning horizons. The rest of paper is structured as follows. Section 2 describes NRP used in this investigation and its modelling as a Markov Decision Process. Section 3 explains the details of the proposed algorithm. Section 4 presents the experimental results. Section 5 concludes the paper and outlines future work.

2 The Multi-Stage Nurse Rostering Problem

In the multi-stage nurse rostering problem the planning horizon is seen as multiple non-overlapping stages, nurse rosters should be selected one stage at a time. A stage is a part of the planning period for which the demands are completely known at its start [7]. In this paper, the Second International Nurse Rostering Competition (INRC-II) instances are used for experimentation. In these instances, each stage is a week under the competition setting. This section outlines the problem and its modelling as a Markov Decision Process proposed in a previous paper [6].

2.1 Problem Description

An instance in the INRC-II consists of three data parts, *global information*, *week requirement* and *history data*. The global constraints, listed below, are those that are the same for each stage of the problem and those that are applicable to the last stage only.

H1 A nurse can be assigned at most one working shift per day.
H3 Two consecutive shifts of a nurse must follow a legal shift type successor, for example a late shift could not be followed by a early shift.
H4 A shift of a given skill must be fulfilled by a nurse having that skill.
S5 Each nurse is required to either work or rest on both days of weekends.
S6 For the whole planning period, each nurse has a minimum and maximum total number of working assignments.
S7 For the whole planning period, each nurse works a maximum number of weekends.

Week requirement is a list of specific hard or soft constraints in each week:

H2 For each day, shift or skill combination, the assigned number of nurses must cover the minimum requirement.
S1 The number of nurses for each shift with each skill must be equal to the optimal requirement.
S2 Maximum and minimum number of consecutive assignment per shift or day.
S3 Maximum and minimum number of consecutive days off.
S4 Respect to the specific shift requirement for each nurse.

History data is a summary of the acutal roster for the previous stage which is required when tackling the problem. If the first week is the current solving stage, history data is randomly selected from built-in artificial files [4]. History data for each stage must be produced by solvers before processing to the next stage and it should include the following information for each individual roster:

- the last assignment of previous week.
- consecutive assignments of the same type as last day.
- total number of worked shifts.
- total number of worked weekends.

In the above list of constraints, H indicates hard constraints that must be satisfied by a solution to be considered feasible and S indicates soft constraints that incur a penalty if violated.

2.2 Problem Modification

Given that in each stage the future demand in this multi-stage NRP is considered as unknown, we apply the framework by Powell [2] which considers the exogenous information. The Markov Decision Process (MDP) notation is summarized as $\{S, A, W, Tr(S, A, W)\}$.

S is a **state** variable, split as pre-decision state and post-decision state. The pre-decision state is the start point and the post-decision state is a termination for each stage. For each stage t in the NRP, the pre-decision state variable corresponds to the combination of weekly schedules from stages 1 to $t-1$, and S is the empty set for the first stage. The post-decision state is the combination of weekly schedules including the one for the current stage t.

A is an **action** variable which determines the policy selected in the current stage. In the NRP, A is a weekly roster where each nurse is assigned a combination of integer variables indicating the shift type for each day. The feasibility of a solution is controlled by the selection of decisions.

W is defined as **exogenous information** which is available only within each stage t. In the NRP, W represents the weekly requirements (local constraints) described above.

The **transition function** $Tr(S, A, W)$ transfers a pre-decision state to the post-decision state with the decision A and the exogenous information W. In the NRP considered here, the transition function performs two roles, one is to update the solution with weekly roster A and week data W and the other one is to update the nurse historical information based on the value of A and W.

3 Proposed Algorithm

The structure of the proposed algorithm is exhibited in figure 1 and consists of three parts. First, the pre-process phase sets up the search space. Then, the local phase is an enhancement of our previous work [6] for solving the weekly optimization problems. Finally, the global phase applies a lookahead policy for future demand evaluation. Each of these parts is explained below.

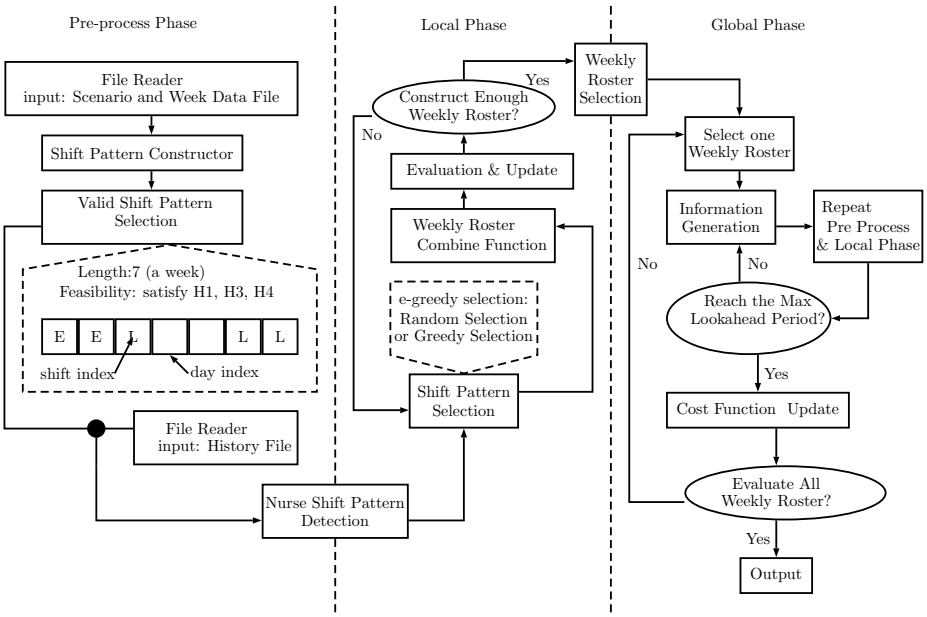

Fig. 1. Overview of the proposed algorithm applying ADP wit combined policy functions

3.1 Pre-process Phase

If a shift pattern (SP) is defined as a weekly roster of a nurse, then a solution should be described as the combination of nurses' shift patterns. A solution is feasible if and only if each constructed SP satisfies all the hard constraints. Exploring infeasible solutions is not required in the principle-of-optimality approaches [2]. Instead, evaluating feasible-shift-pattern based solutions has the potential to make the search more efficient. With this purpose, the pre-process phase is designed to construct a reduced search space for the subsequent local and global phases.

The pseudocode of this pre-process phase is shown in algorithm 1. Hard constraints selected to filter shift patterns belong to *global information* (2.1) which each individual nurse roster is expected to obey. The set that contains all feasible shift patterns is defined as feasible set. Lines 2-6 are the selection steps, where *sp* indicates a single shift pattern and *vsp* represents the feasible set.

Once the feasible set is prepared, some shift patterns are not available to specific nurses with the consideration of nurse history data. For example, if the last assignment of a nurse in history data is a late shift, then any pattern starting with an early shift in the feasible set becomes infeasible for this nurse (2.1 **H3**). Lines 7-13 represent the specific shift pattern selection procedure of each nurse with the consideration of related history data.

Algorithm 1 Pre-process Phase

```
1: vsp ← null;
2: repeat
3:     sp ← ShiftPatternConstructor();
4:     if sp satisfy hard constraints then
5:         add sp to vsp;
6: until no more action from constructor
7: for Each Nurse n do
8:     ivsp ← null;
9:     Collect the last assigned shift type x_last;
10:    for each sp ∈ vsp do
11:        Select x_1 from sp;
12:        if {x_last, x_1} satisfy hard constraint then
13:            add sp to ivsp
```

3.2 Local Phase - Value Function Approximation

Given the output of the pre-process phase, the weekly nurse rostering optimization problem can be seen as selecting a proper shift pattern for each nurse, so as to satisfy constraint **H2** and minimize the soft constraints violation cost. The input to this phase are the *ivsp* for each nurse. Since the future demand is assumed not known in each particular week, the local optimal weekly roster is not

guaranteed to be the one incorporated into the overall solution. Therefore, the output of this local phase is a selection of weekly rosters as depicted in figure 1.

$$Q(S,a) = r(S,a) + \gamma max_{a'} Q(\delta(S,a), a') \tag{1}$$

The Q-learning function, presented in Eq (1), is applied to tackle the local phase problem. The aim is to update the value of S when changes are made by the selected a. In this multi-stage nurse rostering problem, S is a weekly roster and a is a list of selected shift patterns for nurses. Shift patterns are selected based on two methods. *Random Selection* is applied if S is not fully constructed or sample size of S is small. Shift patterns of unassigned nurses in the roster will be randomly selected. This selection is replaced by *Greedy Selection* after constructing a number of S. For a fully constructed S, shift pattern of one or a list of nurses is updated by the one with minimum cost, or equally described as highest reward, from previous steps. $r(S,a)$ is the reward function and calculated from two aspects, the overall constraint violation update and times of the selected a. The pseudocode of this local phase is shown in algorithm 2.

Algorithm 2 Local Phase - Value Function Approximation

```
1: Initial value of max_iter, ϵ
2: i ← 0, M ← Empty S_List ← Empty;
3: while i < max_iter do
4:     Sol ← Empty
5:     for Each Nurse n do
6:         rnd ← RandomNumberGenerator()
7:         if rnd < ϵ then
8:             sp ← RandomSelection(ivsp)
9:         else
10:            sp ← GreedySelection(ivsp)
11:        Insert(Sol, sp)
12:        c = CostFunction(sp)
13:        UpdateValue(V(Sol), c)
14:    Add(S_List, Sol)
15:    e = ExpectedFunction(S_List)
16:    γ = Parameter(V(Sol), e)
17:    UpdateValue(V(Sol), γ × e)
18:    Update(ϵ)
19:    i ← i + 1
20: WeeklyRosterSelection(S_List, M)
```

A sample here is a weekly roster which is constructed by selecting shift patterns from each nurse. The shift pattern selection function in lines 6-10 uses *RandomSelection* or *GreedySelection* which selects the shift pattern with minimum cost. This is known as ϵ-greedy selection function [2]. This shift pattern selection function ensures that the local phase constructs a weekly rosters set

with a degree of variety and not only concentrating on the local optimum. The selected shift pattern *sp* is added to *Sol* in line 11. In line 12, *CostFunction* calculates the shift pattern cost (c) according to the violation of soft constraints **S1**-**S5** and then the value of this weekly roster is updated in line 13.

In line 14, the fully constructed weekly roster *Sol* is stored in the sample list S_{List}. Lines 15-18 correspond to the *Evaluation & Update* in figure 1. The purpose of the expected function is to indicate the average value of the constructed weekly roster while γ is an importance factor and its value is adjusted in the opposite direction to the value of the constructed weekly roster. For instance, if the cost value of a particular weekly roster is larger than the expected value, the value of γ is set to a smaller value, and vice verse.

The end of this local phase in line 20 results in the output set M which is a subset of S_{List}, i.e. a set of weekly rosters some with small constraint violation cost (due to the greedy selection) and others with possibly large cost (due to the random selection). This set M is the input to the global phase described in the following subsection.

3.3 Global Phase - Lookahead Policy

In the local phase, the weekly rosters are evaluated for the weekly constraints only, i.e. from **H1** to **H4** and from **S1** to **S5**. However, since in each week the future demand is unknown, the global constraints **S6** and **S7** are not considered. Then, this global phase evaluates the weekly rosters with artificial future demand through a lookahead period. The lookahead policy seeks to construct a potential solution within a lookahead period based on the weekly roster and artificial future demand in order to evaluate the solution for the global constraints. The input to this global phase is the set of weekly rosters M from the local phase. The output is one weekly roster only as the final solution to the weekly optimization problem. The pseudocode of the global phase is shown in algorithm 3 which is applied to each weekly roster in M. The method *Information Generation* will be explained in section 4, here we assume all the artificial future demand is obtained in advance.

$LK(S)$ is the lookahead value for each weekly roster S and calculated using equation 2. n is the nurse index. *stage* is the week index. T is the lookahead period. sp_n is a single shift pattern of nurse n in the weekly roster S. x_{nt} is a shift pattern at the lookahead stage t of nurse n. x_{nt} belongs to the valid shift pattern set VSP_{nt}.

$$LK(S) = \sum_{n=1}^{N} \sum_{t=stage}^{T+stage} min_n V(sp_n, x_{nt}) \tag{2}$$

This global phase incorporates the pre-process and local phases described above. For each nurse n, the valid shift pattern set VSP_{nt} is constructed in line 7 based on the current shift pattern sp_n and the artificial weekly demand at lookahead period t. We select the shift pattern x_{nt} in VSP_{nt} with the lowest cost and build up an ideal individual assignment with the combination of sp_n

Algorithm 3 Global Phase

1: Initial value of $LK(S)$;
2: **for** Each Nurse n **do**
3: select sp_n from S
4: Initial $ideal\_sol = Insert(sp_n, \phi)$
5: $V(ideal\_sol) = V(sp_n)$
6: **for** $t \leftarrow stage \quad to \quad T + stage$ **do**
7: $VSP_{nt} \leftarrow Pre - processPhase$
8: $x_{nt} \leftarrow GreedySelection(VSP)$
9: $c \leftarrow CostFunction(x_nt)$
10: $UpdateValue(V(ideal\_sol), c)$
11: $Insert(ideal\_sol, x_{nt})$
12: $c \leftarrow CostFunction(ideal\_sol)$
13: $UpdateValue(V(ideal\_sol), c)$
14: $UpdateValue(LK(S), V(ideal\_sol))$

and x_{nt} in lines 8 and 11. The initial value of this ideal solution is the same value of sp_n and is updated with the constraint violation cost of x_{nt} in line 10. Lines 7-11 are repeated until reaching the last lookahead stage $T + stage$. The value of *ideal_sol* is then added the constraint violation of **S6** and **S7**. This is the evaluation of a single shift pattern sp_n and this value is added to $LK(S)$ for each nurse n.

Once all the weekly rosters are evaluated through the algorithm 3, the one with lowest $LK(S)$ will be selected as the final weekly solution and the nurse historical information is updated for the following week.

4 Experimental Design and Results Analysis

The problem instances for evaluating the proposed approach are selected from the Second International Nurse Rostering Competition (INRC-II) [4]. The are three sets of instances, all available at [8]. One is a test set with small number (up to 21) of nurses. Another is the competition set released to the competitors. The last set is a hidden set that was made available at the end of the competition. For the experiments here we use the first two data set only.

The proposed algorithm described in section 3 was implemented in Java (JDK 1.7) and all computations were performed on an Intel (R) Core (TM) i7 CPU with 3.2 GHz and RAM 6 GB.

4.1 Experimental Settings

For a problem that considers 3 working shifts and 1 day off per day of the week, the total number of possible shift patterns is 16384 (4^7). The pre-process phase reduces this number to 1607 making possible to apply the proposed approach to solve large NRP instances. There are three different representations in the value function approximation, lookup table, parametric model and non-parametric

model. As the search space is considerably small after the pre-process phase, we implement a lookup table in the local phase procedure.

The initial value of ϵ is set to 0.9 and is updated based on the Generalized Harmonic Step Size Function [2]. Through preliminary experimentation we tuned the size of the simulation sample $S_{List} = 100$ and the output set $M = 30$ in the local phase. Also through preliminary experiments and results analysis, we decided to select elements from S_{List} for M following the 1-6-3 rule. That is, 10% is selected from S_List with the lowest $V(S)$, the 90% of S_List is split into two subgroups, good and bad, based on the constraint violation cost. Then 60% is randomly selected from the good subgroup and 30% is randomly selected from the bad subgroup.

The cost value for both single shift pattern sp and weekly roster S is calculated using Eq. (3) where c_s is the soft constraint violation cost and V_{sc} is the number of violation for each constraint. The calculation of the constraint violation is fully described in [4].

$$c = \sum_{eachconstraint} c_s \times V_{sc} \tag{3}$$

The artificial future demand is generated by randomly selecting a week data file per week in the lookahead period. Back to the algorithm described in the section 3, only one future path is evaluated for each weekly roster. Less evaluations of lookahead policy is not ideal but more evaluations consume much computation time and memory. By preliminary experiments we found that 1000 evaluations is the minimum to achieve the level of performance in our results while still using considerably short computation time. The value of $LK(S)$ is updated based on Eq. (4). All experimental results presented in the rest of this section correspond to 20 runs for each problem instance.

$$LK(S) = \frac{1}{k}\sum_{i=1}^{k} LK_i(S) \tag{4}$$

4.2 Lookahead Period Comparison

We tested various lookahead periods for each planning horizon. The lookahead period T for scenarios with 4 weeks is set as 1, 2 and 3 and as 3, 5 and 7 for scenarios with 8 weeks. All the scenarios from the test set were used for these experiments comparing the different values of T and results are presented in table 1.

In the table, *Obj* is the average objective value and *Std.* is the standard deviation. The performance of using longer lookahead period is not much better than when using a shorter one for the smallest problem instance (n005w4). But for the larger problem, either with longer planning horizon or larger number of nurses, the average objective value when using that largest T is the best, as much as 20% improvement is achieved in instance $n021w4$. The standard deviation value is smaller as the value of T increases indicating that the algorithm performance is more robust with longer lookahead period.

	T=1		T=2		T=3	
Instance	Obj	Std.	Obj	Std.	Obj	Std.
n005w4_1	456	55.724	452	49.67	**451.5**	23.573
n005w4_2	436.5	35.6735	**430.5**	31.578	**430.5**	14.568
n005w4_3	541	67.456	530.5	54.674	**530**	33.584
n021w4_1	2176	435.754	2056.5	343.563	**1815**	185.683
n021w4_2	3059.5	563.743	2375.5	484.626	**2150**	254.673
n021w4_3	3415	447.784	2767.5	306.639	**2035**	186.460
	T=3		T=5		T=7	
Instance	Obj	Std.	Obj	Std	Obj	Std
n012w8_1	1527.5	435.375	1375.5	368.466	**1237.5**	235.256
n012w8_2	1747	373.692	1623.5	275.573	**1544**	205.574
n012w8_3	1928.5	563.681	1736.5	503.684	**1515.5**	385.678

Table 1. The Average Objective Value and Standard Deviation Obtained with Various Lookahead Periods for Each Instance. Best values are indicated in bold.

4.3 Algorithm Validation and Comparison

Based on the observations from the experiments with the test set, the lookahead period was set to $T = 3$ for 4-week instances and to $T = 7$ for 8-week instances on experiments with the competition data set. Results are presented in table 2.

A value of *99999* in the table indicates that the approach ran out of memory. The performance of the proposed ADP-CP is evaluated through two aspects for each instance. In the left part of the table 2 we compare it with each individual policy. The solution constructed by individual simulation approach is a combination of optimal weekly rosters. The global constraints are considered only when solving the weekly optimization problem in the last stage. On the other hand, the individual lookahead policy focuses on the solution evaluation of global constraints but each weekly solution is solved with random selection approaches. Looking further has the benefit on the overall solution by comparing the value in columns 2 and 4. Local optimum is only concentrated on the assignment patterns, such as the consecutive working patterns and the consecutive days off. We select the instance $n030w4\_1$ as an example. The number of working shifts for each nurse is set as 4 to avoid local constraint violations. The total working days for each nurse is 16 in the final solution. However in some contract, the minimum total working days is 20. A significant large global constraint violation cost is added to the final objective value. A good weekly roster also improves the optimality of lookahead policy with the comparison of columns 3 and 4.

The right part of table 2 seeks to validate our ADP-CP approach by comparing the quality of the solutions obtained to the *Best* and *Worst* reported for the competition. The performance of ADP-CP is close to the best in the instance *n030w4*. It also achieved a good gap from the best in instances *n040w4* and *n050w4*. However, the performance is not so close to the best solutions for larger problem instances. Nevertheless, the quality of the solutions produced with the proposed ADP-CP is in the middle among all the competition results which were produced by several different algorithms. We believe that this work has accomplished good progress in making possible the application of dynamic program-

Instance	Simulation	Lookahead	ADP-CP	Best	Worst
n030w4_1	1925	2725	**1780**	1745	9850
n030w4_2	2650	2710	**1610**	1935	10605
n030w8_1	5350	6645	**4830**	2295	21185
n030w8_2	6310	5820	**4855**	1900	21145
n040w4_1	8120	3945	**3270**	1765	14680
n040w4_2	6895	4260	**3735**	1910	14460
n040w8_1	14720	10125	**9305**	3105	35010
n040w8_2	19255	10165	**8975**	2975	33000
n050w4_1	5900	4070	**3535**	1525	17745
n050w4_2	6210	4070	**3030**	1480	15380
n050w8_1	19525	10045	**8965**	5560	43040
n050w8_2	13905	9725	**8420**	5475	42765
n060w4_1	18480	16977	**12282**	2830	19230
n060w4_2	20945	17794	**15019**	2950	20400
n060w8_1	20215	**9590**	9720	2840	44130
n060w8_2	17545	11000	**10160**	3200	44430
n080w4_1	23195	21870	**18350**	3474	26935
n080w4_2	26305	21435	**16885**	3535	27210
n080w8_1	48505	44880	**35975**	4845	64915
n080w8_2	47355	44065	**38800**	5105	66515
n100w4_1	19625	19295	**16045**	1445	33740
n100w4_2	20530	20270	**17885**	2070	33465
n100w8_1	53155	39550	**35690**	3095	85260
n100w8_2	50340	40755	**35440**	3135	87445
n120w4_1	99999	24075	**22960**	2470	36235
n120w4_2	99999	22680	**22065**	2530	36320
n120w8_1	99999	43215	**39170**	3555	83590
n120w8_2	99999	**40840**	41350	3435	82145

Table 2. Experimental Results of the Proposed ADP with Combined Policy (ADP-CP), Individual Simulation Approach, Individual Lookahead Policy, Best and the Worst Results from the Competition. Best values are indicated in bold.

ming, with the approximation policies, for solving this complicated multi-stage nurse rostering problem. This is an important step towards making dynamic programming practical in its application for solving difficult combinatorial optimization problems when a multi-stage solving approach can be followed.

5 Conclusion

This paper proposed a three-phase approximate dynamic programming (ADP) algorithm to solve the multi-stage nurse rostering problem. This is a problem where a roster is constructed for each week with the future demand assumed not known and the history information for the previous week needs to be considered. The first phase of the proposed approach is a pre-process that generates a set of valid shift patterns. The second phase is a local phase that applies the the value function approximation, to solve the weekly optimization problem and

generate a set of weekly rosters. The third phase is a global phase that implements a lookahead policy to evaluate the effect of the future uncertainty within a lookahead period. The proposed ADP then combines value function approximation and lookahead policy. The instances from the Second Nurse Rostering Competition (INRC-II) are used in the experiments to validate the performance of this proposed algorithm. Experimental results show that the combined policy approach in the proposed algorithm produces better performance than the individual policies. Besides, the results obtained with the proposed algorithm on some of the INRC-II problem instances are close to the best solutions reported for the competition. Future works should be focused on improving the solution quality and reducing the computational time. These improvements could be achieved by applying different methods to evaluate the lookahead samples. Furthermore, improving the quality of weekly rosters could also benefit the lookahead policy as arguably better weekly rosters could help to achieve better results with shorter lookahead periods and also reduce the computation time.

References

1. Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.
2. Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.
3. Edmund K Burke, Patrick De Causmaecker, Greet Vanden Berghe, and Hendrik Van Landeghem. The state of the art of nurse rostering. *Journal of scheduling*, 7(6):441–499, 2004.
4. Sara Ceschia, Nguyen Thi Thanh Dang, Patrick De Causmaecker, Stefaan Haspeslagh, and Andrea Schaerf. Second international nurse rostering competition (inrc-ii)—problem description and rules—. *arXiv preprint arXiv:1501.04177*, 2015.
5. Gerald Tesauro. Practical issues in temporal difference learning. In *Reinforcement Learning*, pages 33–53. Springer, 1992.
6. Peng Shi and Dario Landa-Silva. Dynamic programming with approximation function for nurse scheduling. In *International Workshop on Machine Learning, Optimization and Big Data*, pages 269–280. Springer, 2016.
7. Nguyen Thi Thanh Dang, Sara Ceschia, Andrea Schaerf, Patrick De Causmaecker, and Stefaan Haspeslagh. Solving the multi-stage nurse rostering problem. In *Proceedings of the 11th International Conference of the Practice and Theory of Automated Timetabling*, pages 473–475, 2016.
8. INRC-II the second nurse rostering competition. http://mobiz.vives.be/inrc2/. Accessed: 2016-05-23.