Proceedings of the Federated Conference on Computer Science and Information Systems pp. 407–414
DOI: 10.15439/2017F237
ISSN 2300-5963 ACSIS, Vol. 11

An Integer Programming based Ant Colony Optimisation Method for Nurse Rostering

Joe D. Bunton, Andreas T. Ernst
School of Mathematical Sciences,
Monash University,
Melbourne, Australia
Email: {joe.bunton, andreas.ernst}@monash.edu

Mohan Krishnamoorthy
Department of Mechanical and Aerospace Engineering,
Monash University,
Melbourne, Australia
Email: mohan.krishnamoorthy@monash.edu

***Abstract*—Nurse rostering problems are typically too large and hard to be solved exactly. In order to achieve quality solutions to these difficult problems, meta-heuristics are often employed. One such meta-heuristic is Ant Colony Optimisation (ACO), inspired by the pheromone trails left by ants. ACO works by guiding a heuristic solution construction by using these pheromones to direct weighted random choices. When the problem to be solved is highly constrained, finding feasible solutions is difficult, which can result in poor performance for ACO. To address this, we propose an ACO algorithm using an integer programming based solution construction method to ensure feasibility and select from a collection of schedules. The approach also uses a novel solution merging step that combines the information from multiple ants to generate a better final roster. We discuss several challenges inherent in this approach, and how they may be overcome. Computational results on highly constrained nurse rostering problem instances from the literature demonstrate the effectiveness of our proposed new hybrid metaheuristic.**

I. Introduction

Rostering problems involve the assignment of employees to shifts in order to satisfy cover demands, subject to hard and soft constraints according to rules and preferences respectively. For a review of rostering problems and methods, see [1], or more recently [2].

The Ant Colony Optimisation (ACO) metaheuristic [3] has been applied successfully to a broad range of combinatorial optimisation problems since its inception, including rostering problems. ACO is inspired by the behaviour of real ants that leave trails of pheromones in order to communicate. In ACO, pheromones are used to guide solution construction by encouraging the inclusion of solution components with a higher pheromone presence. Pheromones are left by ants in quantities proportional to solution quality to push future ants towards higher quality solutions. In the past, ACO has been applied to a dynamic version of the Nurse Rostering Problem (NRP) [4] and to a very loosely constrained NRP variant [5]. This is the only ACO and exact algorithm hybrid we have encountered for the NRP.

Various hybrid techniques involving ACO have been implemented in literature, including hybrids with Constraint Programming (CP) [6], [7], Lagrangian Relaxation [8], [9], and Linear Programming [10]. Hybrids of exact methods with other meta-heuristics have also been applied to the NRP, including Tabu Search with CP [11] and Integer Linear Programming [12], Integer Programming with Variable Neighbourhood Search [13], and Iterated Local Search with CP [14]. More generally, hybridisation of exact algorithms and meta-heuristics is discussed and classified in [15].

What we are proposing in this paper is a new type of hybrid meta-heuristic in which Integer Programming is not only used to handle maintaining feasibility of constraints, but also to make an objective guided selection between a subset of possible schedules during solution construction of work-lines. The ACO framework then provides a way to manage solution diversification and intensification.

Highly constrained rostering problems feature a set of hard constraints that cause many roster combinations to be infeasible, reducing the feasible space of the problem. This can hinder the performance of meta-heuristics such as ACO where the random solution construction cannot achieve feasibility. In some cases this can be addressed by using problem specific knowledge to ensure the construction of only feasible solutions, however this is not always possible. Alternatively, this set of hard constraints (or a subset of) may be relaxed and penalised heavily to discourage violations. Such penalty approaches can be problematic as they distort the fitness landscape, often creating many more local optima so that it is more difficult for the optimisation to find the global optimum.

To address the problem of constructing feasible solutions, we use integer programming to generate feasible work-lines for nurses which can be combined to form a complete roster. This allows us to ensure feasibility in the construction of solutions for our highly constrained NRP, as well as select the best option from a subset of schedules, however the use of Integer Programming can introduce other challenges. Integer programs can be slow to solve for difficult problems. Quick solution construction is desired for use within metaheuristics in order to meaningfully explore the solution space. Optimally solving Integer Programming sub-problems can also lead to less solution diversity and quicker convergence to a local optimum. Our proposed hybrid algorithm addresses this challenge by using ideas from ACO to manage diversification and intensification of the search.

We propose an Integer Programming based ACO method for highly constrained rostering problems, making use of the benefits of this hybrid approach while addressing possible

concerns. While our approach makes use of features that are general to rostering problems, we begin developing this new hybrid method by applying it to instances from a highly constrained NRP dataset to demonstrate its effectiveness at generating good solutions quickly, in a way that scales well for larger problems.

II. Nurse Rostering Problem

The NRP consists of scheduling of nurses in hospitals to satisfy shift requirements. There are several types of shifts each day e.g. Day, Night, Early, Late, each with cover demands. The assignment of nurses to shifts is subject to various work contract constraints, both hard and soft, that determine legal and preferred components of nurse schedules. Individual nurse preferences for shifts on/off are also desired to be satisfied. For reviews of models and methods for NRPs, see [16], [17].

Due to the difficulty inherent in NRPs, in order to achieve good solutions quickly, metaheuristics are often applied to problems. Approaches attempted include Simulated Annealing [18], Variable Neighbourhood Search [19], Genetic Algorithm [20], and Tabu search [12], among others.

As discussed in [21], constraints for the NRP can be put in 3 categories. *Sequence* constraints that are applied within shift sequences (work-stretches), e.g. allowed shift transitions and maximum / minimum consecutive work days. *Schedule* constraints that apply to a work-line for a single nurse (a combination of work-stretches), e.g. maximum number of assignments, maximum weekends worked, personal shift requests on / off. *Roster* constraints that apply across nurse work-lines for the entire roster, e.g. cover requirements.

Various descriptions of the NRP have been presented in literature, featuring different constraints and different combinations of these being considered hard / soft constraints. We explore the NRP as defined in [22], and use their set of benchmark datasets hosted online along with best known bounds[1].

Three methods are applied to the NRP dataset in [22], the ejection chain and branch-and-price from [23], and solving the formulation provided with the integer programming software Gurobi 5.6.3. The instances of the NRP dataset have also been solved as a partially weighted maxSAT problem [24] The objective is to minimise the weighted sum of undercover, overcover, and not satisfied nurse shift preferences. This is subject to 10 requirements (with their respective category: *sequence*, *schedule*, or *roster*):

1) A nurse cannot be assigned more that one shift on a single day - *sequence*.
2) Certain shifts cannot follow each other on consecutive days, i.e. a Day shift cannot immediately follow a Night shift - *sequence*.
3) Nurses cannot be assigned more than a certain number each type of shift - *schedule*.
4) Nurses cannot work less than a minimum or more than a maximum number of hours in the schedule - *schedule*.
5) Nurses cannot work more than a maximum number of days in a row without a day off - *sequence*.
6) Nurses cannot work less than a minimum number of days in a row before having a day off - *sequence*.
7) Nurses cannot take less than a minimum number of days off in a row - *sequence*.
8) Nurses cannot work more than a maximum number of weekends in a schedule - *schedule*.
9) Nurses cannot work on days on which they have booked leave - *sequence*.
10) There is an ideal cover requirement to be achieved each day, with over cover / under cover penalised -*roster*.

[1] http://www.cs.nott.ac.uk/~psztc/NRP/index.html

TABLE I
24 benchmark instances

Instance	Weeks	Nurses	Shift Types
1	2	8	1
2	2	14	2
3	2	20	3
4	4	10	2
5	4	16	2
6	4	18	3
7	4	20	3
8	4	30	4
9	4	36	4
10	4	40	5
11	4	50	6
12	4	60	10
13	4	120	18
14	6	32	4
15	6	45	6
16	8	20	3
17	8	32	4
18	12	22	3
19	12	40	5
20	26	50	6
21	26	100	8
22	52	50	10
23	52	100	16
24	52	150	32

A summary of the benchmark instances is given in Table I, varying in the 3 problem dimensions: number of weeks, nurses, and shift types. These instances are highly constrained due to requirement 4, which is both an upper and lower bound on hours worked. This requirement is often quite strict, not allowing much variation in the number of shifts each nurse is required to work. As a result, purely random constructions heuristics perform poorly for these problems.

III. Model

Here we will present the Integer Programming formulation for our solution construction method. The solution construction involves solving an integer program (IP) for single nurse's work-line. A typical integer programming formulation for a NRP will utilise variables representing the assignment of a particular nurse to a specific shift on a given day, as in [22], [23].

As solution construction time is of concern, we make use of the concept of work-stretches for the variables of our IP in order to reduce complexity. We define a work-stretch as a

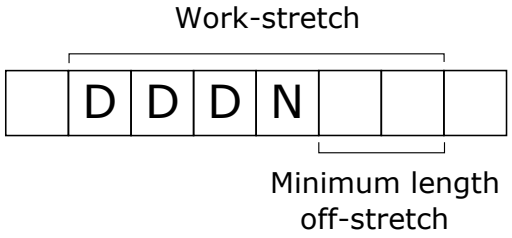

Fig. 1. Definition of a work-stretch as a continuous sequence of work shifts including the following minimum length stretch of off-days (requirement 7).

continual sequence of shift assignments for a specific nurse with no off-days in between, combined with a stretch of off-days of the minimum required length as shown in Figure 1. These work-stretches are able to incorporate all the *sequence* constraints of the problem, as well as the costs due to nurse preferences not satisfied. Only work-stretches that satisfy all of the *sequence* constraints (and so are feasible for our problem) are generated and added to the problem as variables.

There are several examples in literature of the use of work-stretch like structures for the NRP, mainly for use in heuristics, including [21] who also present a brief summary of work-stretches in NRP. For rostering problems more generally, [25] present a general column generation approach based upon work-stretches to reduce complexity. They use a nested resource constrained shortest path that builds work-stretches then uses these to construct columns of work-lines for nurses. Their method is further refined and utilised in a branch-and-price framework [26].

In using work-stretches to model our NRP, we reduce the number of constraints we need to formulate explicitly in our model. Each nurse's schedule is modelled as a network flow with side constraints that cover only the *schedule* and *roster* constraints. By introducing single off-day variables, we can model all combinations of shifts on and off by allowing transitions from work-stretches to other work-stretches or to these single off-day variables. The mathematical formulation of this integer programming model for a given nurse $i \in I$ is given here in terms of the notation described in Table II.

First, we construct the network flow for this nurse. Equations (1) and (2) set up the source and sink of the flow respectively. Equation (3) defines the flow conservation for all other nodes, that the sum of work-stretches or off days finishing on a given day equals the sum of those starting the next day.

Day 0 flow:

$$\sum_{j \in W_{i0}} x_{ij} + o_{i0} = 1. \tag{1}$$

Day h (last day) flow:

$$\sum_{j \in W^E_{ih+1}} x_{ij} + o_{ih} = 1. \tag{2}$$

Middle day flows:

$$\sum_{j \in W^E_{id}} x_{ij} + o_{id-1} = \sum_{j \in W_{id}} x_{ij} + o_{id}, \qquad \forall d \in D \backslash \{0, h\}. \tag{3}$$

TABLE II
SETS, VARIABLES AND PARAMETERS FOR WORK-STRETCH NRP FORMULATION

Component	Type	Description
I	Set	Set of Nurses i
D	Set	Set of Days in the schedule period d
T	Set	Set of Shift types t, e.g. $t \in \{D, A, N\}$
W_{id}	Set	Set of all work-stretches j that start on day $d \in D$ for nurse $i \in I$
W^E_{id}	Set	Set of all work-stretches j that end the day before day $d \in D$ for nurse $i \in I$
W^N_{it}	Set	Set of all work-stretches j that contribute a shift of type $t \in T$ for nurse $i \in I$
W^C_{td}	Set	Set of all work-stretches j that contribute a shift of type $t \in T$ on day $d \in D$
l_t	Parameter	length of shift type $t \in T$ in hours
cst_j	Parameter	penalty cost of assigning work-stretch j
$csto_{id}$	Parameter	penalty cost of assigning nurse $i \in I$ an off-day on day $d \in D$
we_j	Parameter	1 if work-stretch j involves working a weekend, 0 otherwise
c_{itj}	Parameter	number of shifts of type $t \in T$ that work-stretch j contributes for nurse $i \in I$
a_i^{max}	Parameter	maximum number of weekends that nurse $i \in I$ can work
m_{it}^{max}	Parameter	maximum number of shifts of type $t \in T$ that can be assigned to nurse $i \in I$
b_i^{min}	Parameter	minimum number of hours that nurse $i \in I$ must be assigned
b_i^{max}	Parameter	maximum number of hours that nurse $i \in I$ can be assigned
h	Parameter	last day of the horizon
u_{dt}	Parameter	preferred total number of nurses assigned shift type $t \in T$ on day $d \in D$
v_{td}^{min}	Parameter	weight if below the preferred cover for shift type $t \in T$ on day $d \in D$
v_{td}^{max}	Parameter	weight if exceeding the preferred cover for shift type $t \in T$ on day $d \in D$
x_{ij}	Variable	1 if nurse $i \in I$ is assigned work-stretch $j \in \bigcup_{d \in D} W_{id}$, 0 otherwise
o_{id}	Variable	1 if nurse $i \in I$ is assigned an off-day on day $d \in D$
y_{td}	Variable	total below the preferred cover for shift type $t \in T$ on day $d \in D$
z_{td}	Variable	total above the preferred cover for shift type $t \in T$ on day $d \in D$
cv_{td}	Variable	Total cover for shifts of type $t \in T$ on day $d \in D$

Equations (4), (5), and (6) cover the *schedule* requirements 3, 4, and 8 respectively for the nurse. Equation (4) specifies that for each shift type $t \in T$ the sum of shifts of that type worked by the nurse is less than the maximum allowed. Equation (5) specifies that the sum of hours worked by the nurse is within the allowed bounds. Equation (6) specifies that the sum of weekends worked by the nurse is less that the allowed number.

$$\sum_{d \in D} \sum_{j \in W^N_{it}} c_{itj} x_{ij} \leq m_{it}^{max} \qquad \forall t \in T, \tag{4}$$

$$b_i^{min} \leq \sum_{t \in T} \sum_{j \in W^N_{it}} l_t x_{ij} \leq b_i^{max}, \tag{5}$$

$$\sum_{d \in D} \sum_{j \in W_{id}} we_j x_{ij} \leq a_i^{max}. \tag{6}$$

For the *roster* requirement of meeting cover demand, we sum assigned work-stretches for all nurses in the current solution:

$$cv_{td} = \sum_{i' \in I} \sum_{j \in W^C_{td}} x_{i'j}. \tag{7}$$

Assigning under and over-cover variables the correct values:

$$y_{td} \geq u_{td} - cv_{td} \qquad \forall t \in T, d \in D, \tag{8}$$

$$z_{td} \geq cv_{td} - u_{td} \qquad \forall t \in T, d \in D. \tag{9}$$

The objective function is then the sum of work-stretch and off-day costs with under and over cover:

$$\min \sum_{i' \in I} \sum_{d \in D} \sum_{j \in W_{i'd}} cst_j x^w_{i'j} + \sum_{i' \in I} \sum_{d \in D} csto_{i'd} o_{i'd} + \sum_{t \in T} \sum_{d \in D} \left(v^{min}_{td} y_{td} + v^{max}_{td} z_{td} \right). \tag{10}$$

IV. ACO-IP Algorithm

ACO is a meta-heuristic based upon quality solutions leaving pheromones to encourage future solutions. Generally, the solution construction heuristic that is guided by these pheromones is just a weighted random selection. In the case of highly constrained rostering problems, this weighted random selection may choose shift / off-day combinations early on that means upper or lower bounds for work hours cannot be satisfied. To address this, we use an integer programming based ant construction in our ACO-IP hybrid algorithm.

Our ant construction is still guided by random choices in the ACO fashion, using a heuristic component, η, calculated using problem specific knowledge, and a pheromone component, τ. Typically, there is one η and τ component per decision made in the ant construction. As our decisions are the assignment of whole work-stretches, the number of which is exponential in number of shift types, we instead use one of each component η and τ for each shift for each nurse for each day. Rather than directly informing the choice of work-stretch, the weightings are used to choose the reduced set of shifts that will make the components of the work-stretches. All feasible work-stretches are then generated from this subset of shifts, with integer programming used to select the best schedule from these subset of all work-stretches. From the set of available shifts, each is given a probability of being chosen, then shifts are selected for each day without replacement until the desired number of shifts are chosen. Only work-stretches comprising the chosen set of shifts on their given days will be added to the integer programming problem. Thus we have for each ant:

$$p_{itd}(S) = \frac{\tau_{itd}(S)^\alpha \cdot \eta^\beta_{itd}}{\sum_{u \in T} \tau_{iud}(S)^\alpha \cdot \eta^\beta_{iud}} \quad \forall i \in I, t \in T, d \in D, \tag{11}$$

where S is the current solution, $p_{itd}(S)$ is the probability of choosing shift t for nurse i and day d given solution S, $\eta_{itd}(S)$ is the calculated heuristic value of shift t for nurse i on day d given solution S, τ_{itd} is the pheromone value, and α and β are parameters to adjust the influence of η and τ.

After each iteration, all of the pheromone components are evaporated according to some evaporation rate, ρ, then updated with an additional pheromone amounts for each solution in that iteration. The amount of pheromone left is proportional to the quality of the solution, $\frac{1}{objval(S)}$ for a minimisation problem with solution S, scaled by a constant Q:

$$\tau_{itd} = (1-\rho) \cdot \tau_{itd} + \sum_{S \in S_n} \frac{Q}{objval(S)} \qquad \forall i \in I, t \in T, d \in D, \tag{12}$$

where S_n is the set of solutions for iteration n. To avoid extreme pheromone values, it is typical to control the values of pheromones using fixed maximum and minimum pheromone levels. We do not run our algorithm for long enough for this to become necessary.

The approach as described above makes use of the η and τ components to choose the work-stretches that are included in the integer programming problem for each nurse, but not to influence the decisions made directly. The decision of what schedule to choose is limited by the options given by the pheromone guided random choice of shifts, but is made to minimise the objective given in Equation 10 for the available options of work-stretches. This limits the influence of the pheromones in directing the search. It is possible to address this by similarly randomly weighting the objective coefficients for undercover of the corresponding shift and day combinations.

Algorithm 1 Ant Construction: make_ants()

```
for ant in num_ants do
    new_sol = best solution copy
    for all nurse in nurses do
        remove work-line for current nurse from new_sol
        for all day in horizon do
            calculate heuristic weight from new_sol cover
            randomly choose num_shift shifts weighted by
            heuristic weight and pheromone
        end for
        make_workstretches(chosen_shifts)
        solve_nurse()
        add work-line to new_sol
    end for
end for
```

Also of interest are the heuristic weights η, which if calculated for the first few nurses of a solution, will give little information as to which shifts should be scheduled as most of the solution is empty. This can be addressed with a pseudo-elitist strategy where the best know solution is assumed to be present for nurses whose work-lines have not yet been constructed. This both gives the η components a more insightful value and encourages solutions closer to the best known solution. The algorithm for the ant construction in this way is shown in Algorithm 1. The construction of multiple

ants is able to parallelised easily, and we make use of this in our application of the method.

Algorithm 2 ACO-IP: ACO meta-heuristic and merge solve

```
while not iteration limit & not time limit do
  make_ants()
  if new best solution then
    store best solution
  end if
  update_pheromones()
end while
for last n solutions do
  add unique solution work-stretches to merge_stretches
end for
add merge_stretches to merge_IP
add best solution to merge_IP (as incumbent)
solve merge_IP
```

Solving single work-lines to optimality in serial can result in less solution diversity than other random construction methods, especially when we do not alter the objective coefficients for undercover of shifts. To consistently achieve good solutions and properly explore the search space, it is desirable to have some diversity in the solutions being constructed.

In order to improve solution quality, we can make use what diversity there is in the solution set, even for solutions of varied quality, by solving an IP as a final merging step for solutions explored. By combining work-stretches from previous solutions into an IP and taking the current best solution as an incumbent solution, we can explore the neighbourhood around our best solution. As we do not consider all possible work-stretches, this keeps the IP a more manageable size.

The overarching ACO algorithm is described in Algorithm 2, with the integer programming merge step as described as a final step to improve solutions.

V. Experimental Results

To determine the effectiveness of our new approach, we evaluate its performance on the 24 NRP benchmark instances discussed above. Parameter tuning was performed on a subset of the benchmark instances to improve solution quality. Our algorithm was then applied to the full NRP benchmark dataset with the tuned parameters, with ant construction run in parallel. We also analyse the variance of performance of the algorithm on a subset of the NRP dataset. The results are compared with existing results for this NRP dataset from the literature. The ACO-IP hybrid algorithm was implemented in Python to construct 4 ants in parallel for each iteration, using the commercial solver Gurobi 7.0 to evaluate the integer programs. All runs were conducted on 4 threads of an Intel Xeon CPU E5-2680 v3 @ 2.50GHz.

A. Parameter Tuning

There are several parameters in our algorithm that can affect performance. Here we present a brief study over several choices of parameter values to tune our algorithm. The selection of some parameters were made explicitly with time or computational hardware considerations in mind.

The termination criteria was set to 50 iterations of the ACO loop then 5 minutes for the integer programming merge step (or until optimality is proven). The iteration limit chosen is aimed at reducing the run time for instances.

The number of shift types to choose as options for building work-stretches in the ant construction was chosen to be 3 for all instances. As the number of work-stretches can be exponential in the number of shift types, this number of shifts was chosen to give a balance of choices available, which influences convergence performance, and also keeping solve times short. For instances with 3 shift types or less, this means that we are reducing greatly the variability in our approach, especially when the number of employees is small and time horizon is short. As such we omitted the 7 smallest instances from our experiments.

The ant construction in our algorithm is able to be done in parallel, this allows multiple ants to be constructed at each iteration without increase in overall solution time (given enough CPUs). The ant population size was chosen to be 4 ants per iteration for all instances. This was mainly due to computational hardware constraints, enabling each ant construction one CPU core in parallel.

The pheromone evaporation rate ρ was chosen to be 0.05. As we initiate all pheromone values at 1, the evaporation rate was chosen such that after the 50 iteration limit the pheromone values would be reduced by about an order of magnitude.

The parameters selected for tuning were the constant multiplier for pheromone placement, Q, and the heuristic and pheromone influence parameters, α and β. These parameters were tuned for the whole of our ACO-IP algorithm, not including the integer programming merge step, as the variation in solutions generated in the ACO-loop of our algorithm also influences the performance of the merge step.

The choice of the constant Q affects the amount of pheromones placed by ants each iteration. This effects the convergence of pheromone values. As the solution quality ($\frac{1}{objval}$) is instance dependent, this constant Q was chosen in terms of the objective value of the initial solution, scaled by some constant Q_s. This gives:

$$Q = \frac{Q_s \cdot inisol}{objval}, \tag{13}$$

where $inisol$ is the objective value of the initial solution for the solve. This makes the choice of value specific for the instance, without the need for any a priori knowledge.

The α and β parameters effect the relative influence of the heuristic information and pheromone values on the random solution construction. To determine the best combination of Q_s, α, and β, combinations were evaluated for a subset of instances (instances 12, 15, and 19) for 10 runs each.

The choices of parameters for testing were $Q_x \in [0.1, 0.5, 1]$, $\alpha \in [0, 0.5, 1]$, and $\beta \in [1]$. The results of these runs are shown in Table III. The use of pheromones to guide the search can be seen to have a beneficial effect as

TABLE III
AVERAGE BEST SOLUTION AFTER ACO LOOP OF OUR ALGORITHM FOR DIFFERENT Q_s, α, AND β COMBINATIONS AFTER 10 RUNS.

	$Q_s = 0.1$, $\beta = 1$			$Q_s = 0.5$, $\beta = 1$			$Q_s = 1$, $\beta = 1$		
Instance	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$
12	5875	4518.7	5031.1	5682.1	5034.8	5043.6	5064	4911.9	5059.7
15	5026.3	5220.2	5511.2	5060	5392.9	5435.1	5234.5	5552.3	5548.6
19	4154.5	3756.5	3773.8	3722.9	3788.2	3845.8	4017.5	3799.5	3919.2

TABLE IV
COMPARISON OF RESULTS WITH EXISTING APPROACHES. BEST RESULTS ARE IN BOLD, AND OOM INDICATES THE SOLVE RAN OUT OF AVAILABLE MEMORY. INSTANCE DIMENSIONS IN TERMS OF NUMBER OF WEEKS IN HORIZON (W), NUMBER OF NURSES TO SCHEDULE (N), AND NUMBER OF SHIFT TYPES (S) ARE SHOWN ALONG WITH INSTANCE NUMBER. BEST RESULTS HIGHLIGHTED IN BOLD.

	Gurobi 7.0	WPM3	B&P	Ejection Chain	ACO-IP	
Instance (WxNxS)	Sol.	Sol.	Sol.	Sol.	Avg. Sol.	Avg. Time (s)
8 (4x30x4)	**1306**	11018	1308	2260	1450.2	1059.3
9 (4x36x4)	**439**	10949	439	463	570.6	1562.49
10 (4x40x5)	**4631**	16435	4631	4797	4891.1	1433.99
11 (4x50x6)	**3443**	12183	3443	3661	3460	1897.53
12 (4x60x10)	**4040**	18770	4046	5211	4350.8	2698.77
13 (4x120x18)	**2663**	6110163	OoM	3037	6423.9	4007.09
14 (6x32x4)	**1278**	16303	OoM	1847	1456.9	1586.9
15 (6x45x6)	**4843**	30833	OoM	5935	5074	2417.75
16 (8x20x3)	**3225**	10292	3323	4048	3547.1	828.37
17 (8x32x4)	**5749**	22002	OoM	7835	5853.4	1388.78
18 (12x22x3)	**5078**	18498	OoM	6404	5347	1120.19
19 (12x40x5)	**3591**	1698538	OoM	5531	3760	2295.05
20 (26x50x6)	132445	5519316	OoM	9750	**5177.3**	5782.54
21 (26x100x8)	265504	14715064	OoM	36688	**2247.4**	9186.65
22 (52x50x10)	-	-	OoM	516686	**34262.3**	14295.70
23 (52x100x16)	-	-	OoM	54384	**34068.2**	19648.3
24 (52x150x32)	-	-	OoM	156858	**98552**	25188.57

performance is worse when α is set to 0 (when pheromone values are ignored). The algorithm tends to perform better with more of an influence on the heuristic values. This may be due to the limited number of iterations not allowing convergence of the pheromone values. For further runs a Q_s value of 0.1, α value of 0.5, and β value of 1 were chosen.

B. Variance of Performance

As with other random searches, the variance in performance of the ACO algorithm can be quite large. Here we aim to analyse the variance of performance for our ACO-IP hybrid algorithm. To do this we tested our algorithm more extensively on a subset of the NRP dataset (instances 12, 15, and 19).

Our algorithm was run 30 times on each instance, with the mean performance and standard deviation presented for both after the ACO loop is complete, and after the final integer programming merge step. The results are summarised in Table V. It is clear that our integer programming merge step leads to a significant improvement in solutions achieved. The standard deviation of solution achieved does not decrease significantly after the merge step, and in fact increases, indicating there is still variability in the performance of the integer programming merge step.

TABLE V
AVERAGE (AVG.) AND STANDARD DEVIATION (SD) OF BEST SOLUTION AFTER THE ACO LOOP AND FINAL SOLUTION AFTER THE INTEGER PROGRAMMING MERGE STEP AFTER 30 RUNS OF OUR ACO-IP ALGORITHM.

	ACO Search		Final	
Instance	Avg.	SD	Avg.	SD
12	5577.03	229.23	4347.3	334.76
15	5735.86	220.93	5055.6	326.38
19	4411.22	214.32	3782.22	371.04

C. Comparison of Results

Finally our algorithm was run on all instances of the NRP dataset with tuned parameters for comparison with existing results in literature for the dataset used. Existing results on this dataset for comparison are shown in Table IV, including the work of [24] who model the NRP using Partial Weighted maxSAT and solve it using the WPM3 algorithm of [27] for 4 hours runtime, both an ejection chain heuristic method (reported after 10 and 60 minutes, solutions after 60 minutes shown) and a branch-and-price method implemented by [22] as in [23], and finally results for a complete integer programming implementation of the problem instances from [22], run on Gurobi 7.0 with a 1 hour runtime limit.

The best results for these methods are compared with the average result of 10 runs of our ACO-IP approach in Table IV. Note that in our parameter tuning we trained on Instances 12, 15, and 19. Other methods were not similarly trained on a subset of the instances used for comparison.

Best results across all methods are highlighted in bold. While Gurobi obtains the best solution for the largest number of test instances, it is clear that it does not scale well with increasing problem size. Indeed, as the problem size increases our method clearly starts to outperform the others presented

here. For the medium to large instances, our method is comparable to or outperforms the best other heuristic method (ejection chain). While the ability to solve larger problems is encouraging for the scalability of our method, further research is required to improve the method. Gurobi is able to generate solutions of better quality for the medium sized instances, and further work is required to ensure we are generating good solutions for the larger instances.

VI. Conclusions

We have presented a new ACO-IP hybrid metaheuristic for highly constrained rostering problems. It uses an integer programming based solution construction to avoid problems of finding feasible solutions inherent in other random construction methods typical of ACO when problems are highly constrained, as well as to enhance the quality of the schedule chosen over a subset of all options. Performance of the algorithm is improved by a novel integer programming merge step which uses past solutions to explore the neighbourhood around the best solution achieved. While unable to compete against Gurobi for solution quality in the small to medium sized instances, our method scales well and outperforms Gurobi and all other methods for large instances, and is generally able to achieve good solutions to medium instance comparable to or better than the ejection chain heuristic.

These results show that our ACO-IP hybrid algorithm can be effective for highly constrained problems, this encourages the further improvement of the method and application to rostering problems more generally.

References

[1] A. T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier, "Staff scheduling and rostering: A review of applications, methods and models," *European Journal of Operational Research*, vol. 153, no. 1, pp. 3–27, Feb. 2004. http://dx.doi.org/10.1016/S0377-2217(03)00095-X

[2] J. Van den Bergh, J. Belien, P. De Bruecker, E. Demeulemeester, and L. De Boeck, "Personnel scheduling: A literature review," *European Journal of Operational Research*, vol. 226, no. 3, pp. 367–385, May 2013. http://dx.doi.org/10.1016/j.ejor.2012.11.029

[3] M. Dorigo and T. Stutzle, "The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances," in *Handbook of Metaheuristics*, ser. International Series in Operations Research & Management Science, F. Glover and G. A. Kochenberger, Eds. Springer US, 2003, no. 57, pp. 250–285. ISBN 978-1-4020-7263-5 978-0-306-48056-0. http://dx.doi.org/10.1007/0-306-48056-5_9

[4] W. J. Gutjahr and M. S. Rauner, "An ACO algorithm for a dynamic regional nurse-scheduling problem in Austria," *Computers & Operations Research*, vol. 34, no. 3, pp. 642–666, Mar. 2007. http://dx.doi.org/10.1016/j.cor.2005.03.018

[5] J. j. Wu, Y. Lin, Z. h. Zhan, W. n. Chen, Y. b. Lin, and J. y. Chen, "An Ant Colony Optimization Approach for Nurse Rostering Problem," in *2013 IEEE International Conference on Systems, Man, and Cybernetics*, Oct. 2013, pp. 1672–1676. http://dx.doi.org/10.1109/SMC.2013.288

[6] B. Meyer, "Hybrids of Constructive Metaheuristics and Constraint Programming: A Case Study with ACO," in *Hybrid Metaheuristics*, ser. Studies in Computational Intelligence, D. C. Blum, D. M. J. B. Aguilera, D. A. Roli, and D. M. Sampels, Eds. Springer Berlin Heidelberg, 2008, no. 114, pp. 151–183. ISBN 978-3-540-78294-0 978-3-540-78295-7. http://dx.doi.org/10.1007/978-3-540-78295-7_6

[7] M. Khichane, P. Albert, and C. Solnon, "Integration of ACO in a Constraint Programming Language," in *SpringerLink*. Springer, Berlin, Heidelberg, Sep. 2008, pp. 84–95. http://dx.doi.org/10.1007/978-3-540-87527-7_8

[8] D. Thiruvady, A. Ernst, and M. Wallace, "A Lagrangian-ACO matheuristic for car sequencing," *EURO Journal on Computational Optimization; Heidelberg*, vol. 2, no. 4, pp. 279–296, Nov. 2014. http://dx.doi.org/10.1007/s13675-014-0023-6

[9] D. Thiruvady, G. Singh, and A. T. Ernst, "Hybrids of Integer Programming and ACO for Resource Constrained Job Scheduling," in *Hybrid Metaheuristics*. Springer, Cham, Jun. 2014, pp. 130–144, dOI: 10.1007/978-3-319-07644-7_10. http://dx.doi.org/10.1007/978-3-319-07644-7_10

[10] S. Al-Shihabi, "A hybrid of maxâĂŞmin ant system and linear programming for the k-covering problem," *Computers & Operations Research*, vol. 76, pp. 1–11, Dec. 2016. http://dx.doi.org/10.1016/j.cor.2016.06.006

[11] H. Li, A. Lim, and B. Rodrigues, "A Hybrid AI Approach for Nurse Rostering Problem," in *Proceedings of the 2003 ACM Symposium on Applied Computing*, ser. SAC '03. New York, NY, USA: ACM, 2003. ISBN 978-1-58113-624-1 pp. 730–735. http://dx.doi.org/10.1145/952532.952675

[12] C. Valouxis and E. Housos, "Hybrid optimization techniques for the workshift and rest assignment of nursing personnel," *Artificial Intelligence in Medicine*, vol. 20, no. 2, pp. 155–175, Oct. 2000. http://dx.doi.org/10.1016/S0933-3657(00)00062-2

[13] E. K. Burke, J. Li, and R. Qu, "A hybrid model of integer programming and variable neighbourhood search for highly-constrained nurse rostering problems," *European Journal of Operational Research*, vol. 203, no. 2, pp. 484–493, Jun. 2010. http://dx.doi.org/10.1016/j.ejor.2009.07.036

[14] M. Stolevik, T. E. Nordlander, A. Riise, and H. Froyseth, "A Hybrid Approach for Solving Real-World Nurse Rostering Problems," in *Principles and Practice of Constraint Programming âĂŞ CP 2011*. Springer, Berlin, Heidelberg, Sep. 2011, pp. 85–99. http://dx.doi.org/10.1007/978-3-642-23786-7_9

[15] J. Puchinger and G. R. Raidl, "Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification," in *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*. Springer, Berlin, Heidelberg, Jun. 2005, pp. 41–53. http://dx.doi.org/10.1007/11499305_5

[16] B. Cheang, H. Li, A. Lim, and B. Rodrigues, "Nurse rostering problems - a bibliographic survey," *European Journal of Operational Research*, vol. 151, no. 3, pp. 447–460, Dec. 2003. http://dx.doi.org/10.1016/S0377-2217(03)00021-3

[17] E. K. Burke, P. D. Causmaecker, G. V. Berghe, and H. V. Landeghem, "The State of the Art of Nurse Rostering," *Journal of Scheduling*, vol. 7, no. 6, pp. 441–499, Nov. 2004. http://dx.doi.org/10.1023/B:JOSH.0000046076.75950.0b

[18] M. J. Brusco and L. W. Jacobs, "Cost analysis of alternative formulations for personnel scheduling in continuously operating organizations," *European Journal of Operational Research*, vol. 86, no. 2, pp. 249–261, Oct. 1995. http://dx.doi.org/10.1016/0377-2217(94)00063-I

[19] E. K. Burke, T. Curtois, G. Post, R. Qu, and B. Veltman, "A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem," *European Journal of Operational Research*, vol. 188, no. 2, pp. 330–341, Jul. 2008. http://dx.doi.org/10.1016/j.ejor.2007.04.030

[20] 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, Apr. 2004. http://dx.doi.org/10.1016/S0305-0548(03)00034-0

[21] P. Brucker, E. K. Burke, T. Curtois, R. Qu, and G. V. Berghe, "A shift sequence based approach for nurse scheduling and a new benchmark dataset," *Journal of Heuristics*, vol. 16, no. 4, pp. 559–573, Nov. 2008. http://dx.doi.org/10.1007/s10732-008-9099-6

[22] T. Curtois and R. Qu, "New computational results for nurse rostering benchmark instances," 2014. http://www.cs.nott.ac.uk/~psztc/new_computational_results_for_nurse_rostering_benchmark_instances.pdf

[23] E. K. Burke and T. Curtois, "New approaches to nurse rostering benchmark instances," *European Journal of Operational Research*, vol. 237, no. 1, pp. 71–81, Aug. 2014. http://dx.doi.org/10.1016/j.ejor.2014.01.039

[24] E. Demirovic, N. Musliu, and F. Winter, "Modeling and Solving Staff Scheduling with Partial Weighted maxSAT," in *PATAT 2016: Proceedings of the 11th International Conference of the Practice and Theory of Automated Timetabling*, Udine, Italy, Aug. 2016.

[25] A. J. Mason and M. C. Smith, "A Nested Column Generator for solving Rostering Problems with Integer Programming," in *L. Caccetta; K. L. Teo; P. F. Siew; Y. H. Leung; L. S. Jennings, and V. Rehbock (eds.)*, Curtin University of Technology, Perth, Australia, Apr. 1998, pp. p827–834.

[26] A. Dohn and A. Mason, "Branch-and-price for staff rostering: An efficient implementation using generic programming and nested column generation," *European Journal of Operational Research*, vol. 230, no. 1, pp. 157–169, Oct. 2013. http://dx.doi.org/10.1016/j.ejor.2013.03.018

[27] C. Ansotegui, F. Didier, and J. Gabas, "Exploiting the Structure of Unsatisfiable Cores in MaxSAT," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15. Buenos Aires, Argentina: AAAI Press, 2015. ISBN 978-1-57735-738-4 pp. 283–289.