Local Search for Constraint Satisfaction Problem: Cyclic Workforce Scheduling

Abstract

The Workforce Scheduling Problem (WSP) is a complex Constraint Satisfaction Problem which is critical in "round-the-clock" industries such as healthcare. This project focuses on cyclic WSP, which involves assigning employees to shifts while satisfying temporal requirements, shift sequence rules, and other operational constraints. Due to its NP-hard nature, exact solutions become computationally infeasible as problem size increases and demands the need for heuristic-based local search. We present an approach combining greedy initialization, weighted constraint prioritization, and local search optimization techniques to address this challenge. We compared three heuristics: Min-Conflict with Simulated Annealing (SA), Min-Conflict with Tabu Search, and Constraint-Guided with Simulated Annealing (CG-SA) to minimize conflicts and improve scheduling efficiency. We analyzed

20 problem instances, including both benchmark datasets and real-world scenarios. Tabu Search exhibited scalability and runtime performance, However, for smaller instances, Simulated Annealing and Constraint-Guided heuristics produced competitive outcomes. The results show that combining heuristics and optimization methods can help make scheduling solutions that are flexible and effective.

1 Introduction

The workforce scheduling problem (WSP) is a pertinent problem in combinatorial optimization and is applicable in every industry. It requires organizations to determine how many workers should be assigned to each planned period of work and is relevant due to the benefits associated with implementing this system. For instance, scheduling employees to run a bakery while ensuring that customer demand during peak hours is well-managed would help boost customer satisfaction and sales. WSP is particularly critical in "round-the-clock" industries such as healthcare; the Department of Health and Social Care (2021) announced "£7.5 million to support digital shift scheduling across 38 NHS trusts" to improve staff availability for patients. The digital demand-driven scheduling system allocates staff shifts based on patient inflow and healthcare worker availability to address workforce shortages and streamline the delivery of services. In both situations, the optimal solution entails creating a schedule which satisfies the work constraints of an optimal number of workers. Thus, factors such as worker productivity, customer satisfaction, business performance and efficiency are all outcomes of a successful scheduling solution, justifying the relevance of this constraint satisfaction problem (CSP). We proceed with a general, simplified framework based on a cyclic schedule, to deploy our methods on the benchmark instances found in the literature. A cyclic schedule assumes constant demand pattern for the specified time period (e.g. 7 days) and asserts that all employees are available at all times. The constraints

considered include temporal requirements, minimum (maximum) length of work day blocks and off days, minimum (maximum) length of each shift type and sequences of shifts allowed to be assigned to employees; these will be explained in detail in Section 2.

An initial solution for WSP is based on the traditional set-covering formulation proposed by Dantzig (1963), which entails that we minimize the total cost of assigning employees to their shifts (e.g. wages paid, employee's undesirability of shifts) such that the number of employees assigned to a given shift is at least the number of minimum required employees for that shift. However, the key downside of this approach is that as the problem size increases, obtaining such an integer formulation is infeasible. The NP-hard nature of the problem would result in a combinatorial explosion where the number of possible assignments grow exponentially with the number of workers, shifts, and constraints. Notably, Bartholdi III (1981) illustrated that the cyclic WSP is NP-complete, which implies that a fast, optimal solution-finding algorithm is unlikely to be found, especially when conflicting constraints are encountered. Hence, we direct our outlook towards non-exact methods to solve this constrained satisfaction problem. Literature on heuristic-based formulations were initiated with the LP relaxation of the set covering formulation, where alternative shifts are selected heuristically, as conducted by Henderson and Berry (1976). Musliu et al. (2002) proposed pruning the search space by incorporating a feedback loop via the decision maker during the generation of partial solutions. However, for large instances it becomes infeasible for the decision maker to evaluate all the enumerated schedules. To overcome this, Simeunovic et al. (2017) presented a decision support tool leveraging neural networks for WSP; the model complexity, black-box nature of the model, data quality and parameter sensitivity raise concerns over whether decision makers can fully rely on the schedules recommended by the model. Alternatively, Balakrishnan and Wong (1990) formulated and solved the scheduling problem as a network flow problem, but this approach fails to capture realworld constraints such as fairness, employee preferences or time-off requests. Therefore, in search of a simple, flexible and general model which can account for multiple constraints, we encounter a local search with min-conflicts based heuristics for the WSP researched by Musliu (2005), which further includes a Tabu mechanism to escape local optima.

Even with large-scale instances, this approach allows for computational efficiency while incorporating various constraints, thereby justifying our focus on an iterative local search. We explore min-conflict and constraint-guided heuristics within the local search, in combination with optimization algorithms such as Simulated Annealing and Tabu search. We introduce two additional steps to further enhance the local search. Firstly, we use both random and greedy methods to initialize the assignments of workers to shifts based on the temporal requirements. Secondly, we use a weighting scheme for the constraints, to give more importance to constraints such as temporal requirements and less importance to work block length for instance. This will help the the local search to prioritize reducing the violations of the important constraints. We show the problem formulation in Section 2, describe the proposed solution in further detail in Section 3, evaluate our results across various benchmark instances in Section 4 and provide a conclusion in Section 5.

2 Problem Formulation

We formulate the cyclic WSP, assuming constant demand and full availability of workers, using four constraints. We introduce the following notation: let an employee be denoted by i, such that we have n employees ($i = \{1,...,n\}$). Let j be a given day and m be the number of days in the schedule (length of the schedule), such that $j = \{1,...,m\}$. Let $k = \{D,A,N,Off\}$ be the shift type representing day, afternoon, night and off-shifts respectively. The first constraint includes temporal requirements that detail the number of employees required for a particular shift k in a given time period j. Secondly, we enforce the minimum and maximum number of consecutive shifts for each type of shift k, including the block of off-shifts. Thirdly, we establish the minimum and maximum number of consecutive work days that must be present in the schedule. Lastly, we ensure that employees are not assigned forbidden shift sequences, for example, a day shift cannot be followed after a night shift. Note that the list of forbidden sequences is different in each benchmark instance.

2.1 Temporal Requirements

Let the matrix R represent temporal requirements, where element r_{kj} indicates the total number of employees that are required to do shift k on day j. Also, let x_{ikj} be a binary variable such that it is equal to 1 when employee i works shift k on day j. We can then show that the following constraint in equation (1) should hold. For a given shift k and day j, the sum of x_{ikj} for all employees must be equal to r_{kj}.

$$\sum_{i=1}^{n} x_{ikj} = r_{kj}, \forall k \in \{D,A,N,Off\}, \forall j \in \{1,...,m\} \tag{1}$$

2.2 Minimum and maximum number of consecutive shifts

Let y_{ikj} be a binary variable that indicates whether an employee i has worked two consecutive shifts of type k on the current day j and previous day $j-1$. The relation between $y_{ikj}, x_{ikj}, x_{i,k,j-1}$ is represented by the constraint in equation (2). We examine three cases: firstly, if a given employee i works two consecutive shifts, then $x_{ikj} = 1$ and $x_{i,k,j-1} = 1$. Hence equation (2) becomes $y_{ikj} \geq 1 + 1 - 1 = 1$; since y_{ikj} is binary it must equal 1, which correctly suggests the presence of two consecutive shifts. Secondly, we have the case of no consecutive shifts; then $x_{ikj} = 1$ and $x_{i,k,j-1} = 0$ or vice-versa, which leads to $y_{ikj} \geq 1 + 0 - 1 = 0$. Again, since y_{ikj} is binary it takes the value 0, indicating the absence of two consecutive shifts. Thirdly, if neither day has a shift, $x_{ikj} = 0$ and $x_{i,k,j-1} = 0$, then $y_{ikj} \geq 0 + 0 - 1 = -1$ which will be satisfied given the domain of y_{ikj}, and thus $y_{ikj} = 0$.

$$y_{ikj} \geq x_{ikj} + x_{i,k,j-1} - 1, \forall i \in \{1,...,n\}, \forall j \in \{2,...,m\}, k \in \{D,A,N,Off\} \tag{2}$$

$$\sum_{j+max_k-1}$$

$$min_k \leq \sum_{d=j}^{X} y_{ijk} + 1 \leq max_k, \forall i \in \{1,...,n\}, \forall k \in \{D,A,N,Off\} \quad (3)$$

When $y_{ikj} = 1$, days $j - 1$ and j have consecutive shifts of type k. A block of consecutive shifts is formed when multiple y_{ikj} variables take on the value 1 one after the other, where all other shifts immediately before and after this block are different than shift type k. For example, consider $y_{ik2} = 0$, $y_{ik3} = y_{ik4} = 1$, and $y_{ik5} = 0$. This implies that the pairs of shifts x_{ik2},x_{ik3} and x_{ik3},x_{ik4} are consecutive, and thus the block contains consecutive shifts of type k on days 2, 3 and 4. To calculate the total number of shifts in a block of consecutive shifts, we sum y_{ikj} and add 1, because y_{ikj} only tracks pairs of consecutive shifts, hence there is another shift at the start of the block that needs to be accounted for. We sum y_{ikj} starting from day j until the maximum number of consecutive shifts allowed for shift type k. Overall, equations (2) and (3) in conjunction allows us to create and dynamically evaluate all possible work blocks of consecutive shifts (type k), bounded by the minimum and maximum number of permitted consecutive shifts for a given shift type k. Note that this constraint works for all types of shift, including off-days.

2.3 Minimum and maximum number of consecutive work days

We extend the logic of Section 2.2 to define the constraint for the minimum and maximum number of consecutive work days. Equation (4) describes the relation between $y_{ikj},x_{ikj},x_{i,k,j-1}$ as shown before in equation (2), with a key adjustment: now $k \in \{D,A,N\}$, to account only for the shifts constituting a working block and not off-days. Equation (5) then denotes the same constraint as equation (3), but we now add an additional summation over all shift types k which can contribute to a working block.

$$y_{ikj} \geq x_{ikj} + x_{i,k,j-1} - 1, \forall i \in \{1,...,n\}, \forall j \in \{2,...,m\}, k \in \{D,A,N\} \quad (4)$$

$$min_W \leq \sum_{k}^{X} \sum_{d=j}^{j+max_W-1} y_{ijk} + 1 \leq max_W, \forall i \in \{1,...,n\} \quad (5)$$

2.4 Forbidden sequence of shifts

The goal is to prohibit assigning the sequence of shifts $\{s_1,s_2\}$ consecutively on day j and $j - 1$ to employee i. The constraint shown in equation (6) implies that the case where $x_{i,s1,j} = x_{i,s2,j-1} = 1$ is forbidden, because the sum can be at most 1; forcing $x_{i,s1,j} = 1$ and $x_{i,s1,j-1} = 0$ or vice versa. The constraint also allows for both $x_{i,s1,j} = x_{i,s1,j-1} = 0$ since the upper bound of 1 is still satisfied.

$$x_{i,s1,j} + x_{i,s2,j-1} \leq 1, i \in \{1,...,n\}, j \in \{2,...,m\}, s_1,s_2 \in \{D,A,N,Off\} \quad (6)$$

3 Proposed Solution

Combining strong initialization tactics, constraint management, and optimization methodologies is necessary for efficient employee scheduling. Multiple techniques are used in the solution, such as greedy initialization, weighted constraints, heuristics and optimization algorithms. Together, these techniques help to find a feasible solution that minimizes the conflicts and tries to satisfy all the constraints.

3.1 Greedy Initialization

Greedy initialization creates an initial schedule by assigning shifts based on demand. It prioritizes meeting temporal requirements by filling the most needed shifts first. Employees are assigned shifts in descending order of demand for each day. This ensures that the generated initial schedule has minimal critical violations (e.g., unmet temporal requirements). It produces a feasible initial solution and reduces the computation time for the optimization algorithms. Therefore, by implementing a greedy initial solution, a baseline is created that minimizes severe constraint violations, enabling faster convergence in optimization algorithms. However, this alone can lead to suboptimal solutions that do not account for the other constraints in the problem.

3.2 Weighted Constraints

Weighted constraints are used to evaluate conflicts in a schedule based on their severity and importance. Each type of constraint is assigned a weight that determines its penalty for violations. This helps differentiate between critical and minor violations, ensuring the local search focuses on the most impactful constraints. However, it is important to determine the appropriate weights based on the domain to prevent over-penalization of less critical constraints which would affect the overall feasibility of the solution. Three different methods to weigh the constraints are used to determine the ideal method specific to this problem.

Experimentation showed that the temporal requirements and day-off constraints were difficult to satisfy. As a result, in **hierarchical weighting**, to ensure they were giving higher importance, they were assigned penalties 4 and 3 respectively. Forbidden sequences constraint, being moderately important, was assigned a penalty of 2 and the other constraints of lower importance are assigned a penalty of 1. In an **alternate hierarchical weighting** method, the temporal requirements and day-off constraints were assigned a higher penalty of 2 while the other constraints were assigned a uniform penalty of 1. Lastly, in **scaled weighting**, the penalty for each violation is proportional to the degree of the violation. For example, the difference between the actual and required number of employees for a shift determines the penalty. Temporal requirements and forbidden sequences (hard constraint) are given the highest weight since meeting staffing levels for each shift and ensuring allowed shift schedules is very important. Days-off constraints is moderately weighted to ensure fairness and balance. Shift-specific block and work block constraints are given lower weights to provide flexibility for minor deviations. These are then scaled proportional to the degree of the violation to determine the final penalty of violating a particular constraint.

3.3 Heuristics

The **min-conflict heuristic** evaluates potential changes to a schedule and selects the one that minimizes conflicts. For example, during each step of the simulated annealing algorithm, the heuristic evaluates all possible shifts for a given employee on a specific day. The shift with the least conflicts is chosen, ensuring the solution improves or remains stable. The optimization algorithm uses the heuristic to guide the search towards the regions in the solution space that minimize conflict. As a result it helps the algorithm converge faster by avoiding moves that increase conflicts. However, min-conflict heuristic alone focuses only on immediate conflict reduction and often gets stuck in the local optima.

The **constraint-guided heuristic** evaluates potential changes based on their impact on specific constraints rather than overall conflicts. It prioritizes critical constraints during neighbor selection for optimization. It also ensures moves align with the broader objective of creating a balanced schedule. Therefore, it improves the optimization algorithm's search by focusing on the important constraints. However, it can lead to over emphasis on certain constraints and add computational complexity.

3.4 Optimization Algorithms

Simulated Annealing (SA) is a probabilistic optimization algorithm that iteratively refines solutions by exploring the search space. It balances exploration and exploitation using a temperature parameter that decreases over time. SA starts with the initial solution and improves it by making incremental changes. The algorithm accepts worse solutions with a temperature-based probability, helping it escape a local optima. As a result it effectively explores a large and complex search space. However, it is important to tune the cooling rate and temperature parameter to obtain an optimal solution.

Tabu Search is a local search algorithm that avoids cycling by maintaining a list of recently visited solutions (tabu list).The algorithm begins with an initial schedule and iteratively explores neighboring solutions generated using the Generate Neighbors Heuristic to reduce conflicts. Therefore, tabu ensures the solution space has been explored without repetitions. Additionally, it does not get stuck in a local optima as it explores all neighbours. However, the performance depends on the size of the tabu list and the quality of neighbor generation.

4 Numerical Experiments

We address a complex CSP by designing employee schedules that minimize conflicts while adhering to constraints such as temporal requirements, forbidden sequences, and shift block limits. The process begins with parsing the input data and exploring different initialization strategies. In the first phase, we tested Simulated Annealing (SA) with Min-Conflict across multiple weighting methods (Methods 1, 2, and 3) and initialization strategies (random and greedy). The results showed that Method 2 with Greedy initialization consistently achieved the lowest conflict score of 12 in the shortest time (1.1923 seconds). This combination also ensured a better balance between temporal requirements and other constraints, significantly outperforming the alternatives and also resulted in a well-balanced distribution where shifts

D, A, N were allocated more evenly among employees. For example, random initialization with Method 1 resulted in 38 conflicts, while Method 3 yielded a high conflict score of 10,050. Similarly, greedy initialization with Method 3 still resulted in 5,800 conflicts, demonstrating the inefficiency of this weighting scheme. Based on these insights, we selected Greedy initialization and Method 2 would be the best setup for the next analysis.

In the second phase, we compared three heuristics: Simulated Annealing (SA) with Min-Conflict, Tabu Search with Min-Conflict, and Constraint-Guided Simulated Annealing (CG-SA). These techniques, by avoiding local minima and repeatedly enhancing the schedules, eventually produced high-quality results. The final output consists of the optimized schedule, conflict scores, and performance metrics (runtime). The algorithm runs rationally, acting like a human scheduler who might prioritize meeting staffing requirements before addressing fairness or continuity. For example, just like human schedulers, our local search attempts to resolve temporal requirements, which are non-negotiable, while other constraints such as like shift block limits, may be temporarily relaxed. This flexibility is shown by the weighted conflict evaluation, which gives more weight to important constraints.

We used a dataset of 20 problem instances. The first three examples are benchmark examples and the remaining 17 examples appeared in real life situations in different organizations. Each method was assessed based on its runtime and the number of constraint violations. The complexity of the problem varied across instances with the number of employees ranging from 7 to 163.

Inst.	**Emp.**	**C-G + SA**		**MC + SA**		**MC + Tabu**	
		T(s)	**Conf.**	**T(s)**	**Conf.**	**T(s)**	**Conf.**
1	9	1.5	8	1.6	23	1.1	19
2	9	1.6	11	1.6	6	1.2	34
3	17	1.8	37	1.9	53	1.4	34
4	13	2.1	44	2.1	41	1.6	22
5	11	2.1	65	2.0	65	1.5	20
6	7	1.9	40	1.9	40	1.3	16
7	29	2.3	44	2.3	41	1.8	68
8	16	1.8	33	1.8	31	1.4	41
9	47	3.2	108	3.1	128	2.3	126
10	27	2.4	62	2.4	87	1.8	63
11	30	2.3	47	2.4	48	1.7	63
12	20	1.3	26	1.3	46	1.2	29
13	24	2.1	38	2.1	40	1.6	61
14	13	2.1	51	2.1	40	1.5	23
15	64	5.0	246	5.0	258	3.7	147
16	29	2.3	44	2.4	46	1.8	68

17	33	1.6	60	1.6	58	1.5	52
18	53	3.3	63	3.4	117	2.5	124
19	120	6.2	258	6.1	271	4.7	303
20	163	10.2	523	10.2	515	7.8	309

Table 1: Compact Performance Comparison Table

The Constraint-Guided + SA method demonstrated a systematic approach to addressing constraints, starting with basic feasibility and eventually using more complex requirements. This approach worked well for smaller problem, achieving low runtimes (1.5–3.2 seconds) and reasonable conflict counts. However, as the problem complexity increased, its performance declined. For instance, in Instance 15 with 64 employees, the conflict count rose to 246, and for Instance 20 with 163 employees, the conflicts peaked at 523. This indicates that while Constraint-Guided + SA effectively creates feasible schedules, it struggles to optimize conflicts in large-scale problems. Simulated Annealing with Min-Conflict offered a balanced performance in terms of runtime and conflict minimization. For smaller instances, such as Instance 2 with 9 employees, it achieved just 6 conflicts within 1.6 seconds, reflecting its efficiency. Even for medium-sized problems, such as Instance 7 with 29 employees, it maintained relatively low conflicts (41) and manageable runtimes. However, like Constraint-Guided + SA, it exhibited scalability issues for larger instances like Instance 20, where the conflict count reached 515. Despite this, it generally performed better than Constraint-Guided + SA. Tabu Search with Min-Conflict, characterized by its use of memory structures, shows the ability to avoid cycles and consistently produced the fastest runtimes across instances, making it particularly well-suited for larger and more complex problems. For smaller problems, such as Instance 1 with 9 employees, it completed the schedule in just 1.1 seconds with 19 conflicts, outperforming the other two methods. In larger instances, while it did not eliminate conflicts completely, it still shows superior performance. Its efficient memory utilization prevents redundant calculations and cycles, improving computational efficiency. For example 15 with 64 employees, Tabu Search resulted in 147 conflicts, significantly lower than the conflicts reported by the other methods.

5 Conclusions

This paper explored different local search methods with different initialization strategies (random, greedy), weighting schemes for constraints (hierarchical, basic, scaled), heuristics (min-conflict and constraint-guided) and optimization algorithms (simulated annealing, tabu search). The aim was to obtain the most feasible solution, or equivalently a solution with the least penalty count. The results highlight distinct strengths and weaknesses across the different heuristic techniques. Constraint-Guided + SA systematically addresses constraints effectively, although it struggles with large, complex cases. The Simulated Annealing with Min-Conflict method provides a good balance between runtime and accuracy for medium-sized problems, with conflicts spread variably across employee schedules. Although no

methodology completely eliminated conflicts, Tabu Search was relatively more efficient across small and large instances, providing quick execution and fewer conflicts. Furthermore, when analysing shift distribution from the output schedules, we observed that some techniques favoured an even spread of shifts D, A, N across employees, while others focused on certain shift in fewer employees. Tabu search on the other hand produced schedules with fewer gaps and more uniform workload distribution, proving to be the key strength of our research. Future research could explore hybrid methods combining these techniques' strengths or delve into more advanced heuristics. Enhancements in balancing shift distribution and scaling for high-complexity cases are key areas for development.

Limitations with respect to the problem formulation include that our work assumes a cyclic schedule, which is less relevant in the case of a dynamic work environment such as hospitals, where the demand for nurses is highly variable (based on patient inflow) and nurses are not available at all times. Secondly, to measure the quality of our solutions, we tracked number of constraints violated in each solution. However, other subjective measures can also be used to evaluate the performance of a schedule, such as its flexibility, quality and worker happiness. Hence, future suggestions would be to solve an acyclic WSP for each day separately instead of a week, to account for variable demand and availability of employees. Additionally, we can introduce can introduce a broader range of constraints dealing with fairness, employee preferences, skill matching or time-off requests, and potentially construct a multi-dimensional problem to address multiple objectives while minimising the number of violations.

References

Nagraj Balakrishnan and Richard T. Wong. A network model for the rotating workforce scheduling problem. *Networks*, 20(1):25–42, Jan 1990. doi: 10.1002/net.3230200103.

John J Bartholdi III. A guaranteed-accuracy round-off algorithm for cyclic scheduling and set covering. *Operations Research*, 29(3):501–510, 1981.

George Dantzig. Linear programming and extensions. *Operations Research*, 1963. doi: 10.7249/r366.

Department of Health and Social Care. £7.5 million to digitally schedule shifts and save nhs staff time. GOV.UK, 2021. URL https://www.gov.uk/government/news/ 75-million-to-digitally-schedule-shifts-and-save-nhs-staff-time.

Willie B. Henderson and William L. Berry. Heuristic methods for telephone operator shift scheduling: An experimental analysis. *Management Science*, 22(12):1372–1380, Aug 1976. doi: 10.1287/mnsc.22.12.1372.

Nysret Musliu. Min conflicts based heuristics for rotating workforce scheduling problem. *The Sixth Metaheuristics International Conference (MIC2005)*, Aug 2005.

Nysret Musliu, Johannes G¨artner, and Wolfgang Slany. Efficient generation of rotating workforce schedules. *Discrete Applied Mathematics*, 118(1–2):85–98, Apr 2002. doi: 10.1016/s0166-218x(01)00258-x.

N. Simeunovic, I. Kamenko, V. Bugarski, M. Jovanovic, and B. Lalic. Improving workforce scheduling using artificial neural networks model. *Advances in Production Engineering Management*, 12(4):337–352, Dec 2017. doi: 10.14743/apem2017.4.262.