Stuart Miller

A SURVEY OF APPROACHES TO SHIFT-SCHEDULING ALGORITHMS

Abstract – Shift scheduling is a long-studied problem in computer science. Not only is it a highly relevant problem, but it is one that has produced a myriad of viable solutions, each tailored to the needs and purposes of an individual application. A survey of the various methods used to approach this problem reveals the variable and ingenuity of such algorithms.

1 INTRODUCTION

This paper serves to survey and evaluate existing solutions to the shift scheduling problem. In addition to considering the major algorithmic approaches, the option to not implement an algorithm at all, that is, to let human intuition determine the best schedule will also be considered. Furthermore, this paper will survey the availability of open-source libraries and implementations of these algorithms.

2 BACKGROUND

The shift scheduling problem is recorded to have been studied as early as 1961, published by William Vickrey in the American *Journal of Finance* [1]. Therein, Vickrey devised a system to assign employees to duties by a mathematical auction system. The problem is classically referred to as the "nurse scheduling problem", first named by Lagatie, et. al due to the complex needs of emergency room nursing staff [2].

Although shift-scheduling may appear simple on the surface (and indeed, most that are tasked with solving the problem in the workplace, take a simplistic brute-force approach to solving it), it is realistically much more complicated. It is a problem that, in general form, is non-deterministic polynomial time (NP)-complete. A proof of this was established Cooper and Kingston in 1995 [3]. As such, many have taken a heuristic approach to solving these problems.

3 METHODOLOGY

3.1 Human Approaches

Human approaches to solving have long proven to be tedious and inefficient. A human serving as a shift manager is regularly tasked with inventing a feasible schedule and is often ill-equipped to do so. Because this is such a complex problem, involving many variables, humans often approach it by simplification. The first step a human performs is to aggregate or eliminate variables or constants [4]. This, however, frequently leads to oversimplification and often produces a sub-optimal schedule.

Augustine, et.al set out to determine how a real-world hospitals actually deal with the nurse scheduling problem, and discovered that the hospital they partnered with simply avoided the problem, choosing to let the nurses schedule themselves and trade shifts as they saw fit [5]. It seems the complexity of the problem can be daunting when attempting a more optimal solution. Burke et.al, discovered a different real-world solution to the problem. They found that as scheduling option increase, expected nursing expenses

decreases [6]. They credited this to the fact that larger data sets with more options tend to be more forgiving and result a larger number of desirable schedules, making a decent one easier to find.

Li and Aickelin suggest that a human builds up a schedule step-by-step, following an established set of rules. As the human repeatedly creates the schedule, they become more flexible with the rules and are better able to identify successful solution components as they go [7].

Wilson and MacCarthy proposed that most humans approach scheduling with a "mental model" of the situation, which they use to assess the current state of affairs before attempting a solution [4]. Alternatively, they may use a "mental look-up table" which similarly provides a preconceived representation of the problems needing to be solved. While solutions such as these certainly prove to be more efficient, they limit the scheduler in that they are based on what has already been established. The schedule is more likely to be merely an iteration of a past solution, while a new or more radical solution may be possible as the scheduling needs evolve.

A further concern when allowing humans to determine shift schedules is the idea of discriminatory bias. A human scheduler may be prone to unfairly select more favorable shifts for certain employees, precluding the opportunity to create an optimal schedule. It is undoubtable that such bias exists in human-derived solutions to the scheduling problem. Reskin argues exactly this in her overview of "Sex and Race Inequality In Work Organizations" [8]. Furthermore, Stoop and Wiers argue that the conflicting goals of those with different responsibilities may results in poor performance in schedule development and lead to a clear bias [9].

3.2 Heuristic Approaches

Many solutions to the scheduling problem involve artifically intelligent, or genetic alogrithms. Rather than brute-force solving the problem, it is possible to generate a soltuion, analyze it, and branch to a slightly modified solution which is percieved to be more optimal.

Simulated Annealing is one such algorithm which solves this problem quite readily. At its core, simulated annealing is a hill-climbing function that adds in a random probability of accepting the less optimal solution. This randomness is derived by calculating an acceptance probability and comparing it to a random number. For the shift scheduling problem, this could be defined by the equation below (Figure 1), where *c* represents the cost of the schedules being compared and *D* represents the depth of the solution (i.e. the number of schedules evaulated on this branch). This encourage hill climbing towards an optimal solution, while still providing randomness so that the algorithm can avoid settling on a local optimum. [10]

$$a = e^{\frac{c_{old} - c_{new}}{D}}$$

Figure 1: Simulated Annealing Acceptance Probability

In order to apply these concepts to shift scheduling, one has to define an effective cost function, although, it is rather simple to define it as the total hours worked multiplied by the respective hourly rates for those hours. Ko, et al. implemented two simulated annealing algorithms in the C language and applied them to a test data set. Their output graphs the cost function over the iterative depth of the algorithm for repeated runs (Figure 2). Notice how the algorithm starts with a suboptimal, costly schedule, then iterates towards a less costly schedule [11].

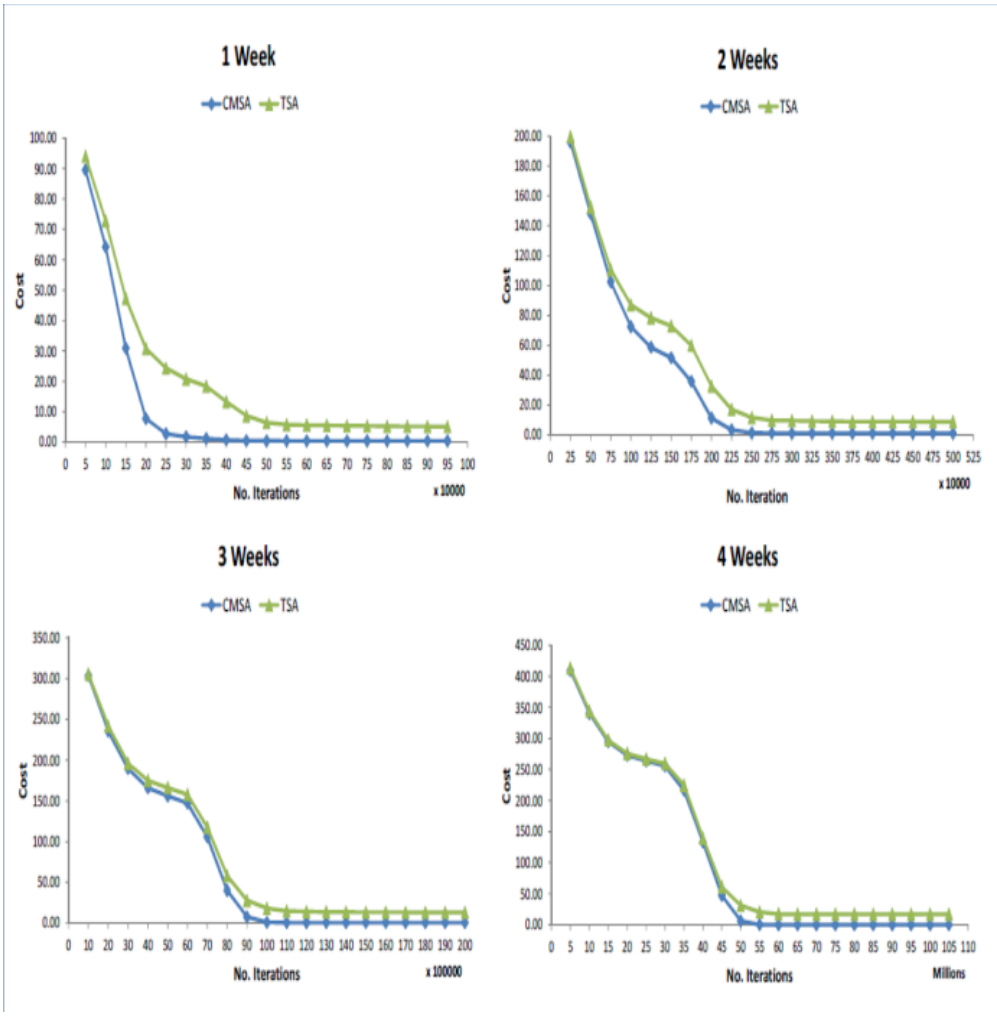

Figure 2: Cost function of solution generated vai Simulated Annealing [11]

There are very few options for simulated annealing libraries; most of the papers surveyed simply wrote their own algorithms. The algorithm is simple enough that it can be written simply in only a few lines of pseudo-code. Bwonlee demonstrates this in his book, *"Clever Algorithms: Nature-Inspired Programming Recipes"* [12]. Perhaps this is why there are so few open-source simulated annealing implementations available. Regardless, the GNU scientific library does include a series of functions for simulated annealing, the top level function being called *gsl_siman_solve()* [13]. Additionally, MATLAB provides the *simulannealbnd()* function as part of its Global Optimization Toolbox [14].

Additionally, genetic algorithms can easily be implemented to solve the shift scheduling problem. A genetic algorithm starts by picking several solutions at random, and then choosing the best solutions based on a cost function. The best individuals are then chosen to generate new solutions by slightly modifying their parameters. Again, the best individuals are chosen again, and so the generation of new solutions continues until a solution within acceptable bounds is generated. As such, genetic algorithms are not guaranteed to be optimal, and provide an easy tradeoff between optimality and computation cost depending on how many generations are evolved [15].

Ohki uses a genetic algorithm to solve the nurse scheduling problem in his paper "A parameter free nurse scheduling" [16]. Possible solutions were defined as arrays of characters (shown in Figure 3), each representing predetermined shifts (d for Daytime, h for Holiday, M for midnight, N for night, etc.) He ran the problem through a cooperative genetic algorithm, the methodology of which he elaborates upon in a related work [17]. Okhi defined a total of 14 penalty functions, ranging from penalties for three or more days worked in a row, too many nurses scheduled during suboptimal time (defined as evening/overnight shifts), for repeated night shifts for a single nurse, etc. The output graph (Figure 4) shows each penalty function as new solutions are evolved. Notice how each penalty function decrease in value over time.

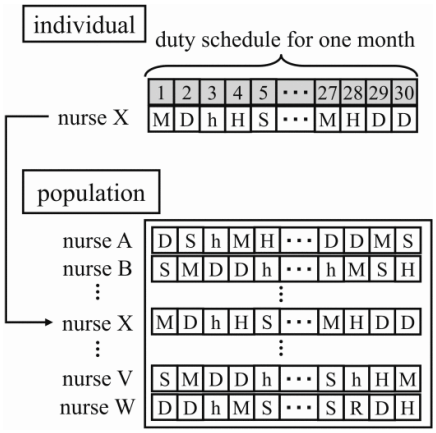

Figure 3: Ohki's feasible solutions (forming a population in the genetic algorithm) [16]

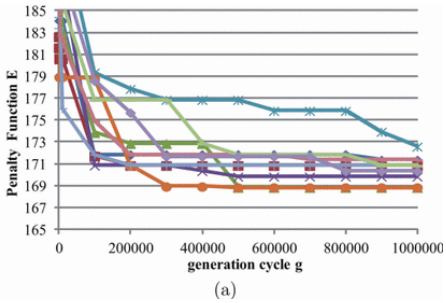

Figure 4: Cost of generated solutions in Ohki's genetic algorithm [16]

Finally, another heuristic search method frequency used for shift scheduling is tabu search. First defined by Fred Glover in 1986, Tabu search works by implementing several local hill-climbing searches, but by keeping potential solutions stored in memory and marking them as "tabu". Revising a stored solution is invalid and encourages the algorithm to consider new solutions [18].

Burke and Soubeiga implemented a tabu search algorithm to solve the nurse scheduling problem in their paper "Scheduling Nurses Using A Tabu-Search Hyperheuristic" [19] . Similar to the previously mentioned methodologies, Burke and Souberga first had to define penalty functions (Figure 5). The model they used defined the functions shows in Figure x. Equation (1) shows the desire to minimize the penalty cost P_{ij} of each schedule choice x_{ij} for each nurse i and shift pattern j. The penalty cost is subject to functions (2), (3), and (4), where (2) restricts each solution to be feasible as defined by $F(i)$, (3) ensures the skill b_{ij} of nurses

is greater than the demand S_{kr} for that shift, and (4) x_{ij} defines our solution to be one if nurse *i* works shift *j* and zero otherwise.

$$Min \quad PC = \sum_{i=1}^{n} \sum_{j=1}^{s} p_{ij} x_{ij} \tag{1}$$

$$\sum_{j \in \mathbf{F(i)}} x_{ij} = 1, \quad \forall i \tag{2}$$

$$\sum_{j=1}^{s} \sum_{i=1}^{n} b_{ir} a_{jk} x_{ij} \geq S_{kr}, \forall k, r \tag{3}$$

$$x_{ij} \in \{0, 1\}, \forall i, j \tag{4}$$

Figure 5: Tabu search penalty functions [16]

Like simulated annealing, tabu search is a well-researched algorithm that is relatively simple in its basic design. It, too, can be implemented in just a few lines of code, as shown by Brownlee in Clever Algorithms [20]. There are a few tabu search libraries available for use. The Coir-Or project provides MetsLib, an Open-Source C++ library for Tabu Search and related metahueristics [21]. Additionally, OpenTS is a Java library containing very popular and well-documented Tabu Search functionality [22].

3.3 Integer Linear Programming (ILP) Approaches

Many solutions to the shift scheduling problem rely on reducing the problem to a set of rules that can be modeled by integer linear programming (ILP). In short, ILP is an optimization problem in which the input parameters are constrained to integers. An ILP can be expressed as the following (Figure 6) where ***c*** and ***b*** are vectors one-dimensions integer matrices, and ***A*** is a two-dimensional integer matrix [23].

$$\begin{array}{ll} \textbf{maximize} & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ \textbf{subject to} & A\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}, \\ \textbf{and} & \mathbf{x} \in \mathbb{Z}^n, \end{array}$$

Figure 6: ILP constraints generalized

Furthermore, integer linear programs are NP-hard, meaning there is no known solution to the problem which is solvable in polynomial time. This is proved by Vempala, by reducing a known SAT problem (which is NP-complete) to a given ILP problem, thus classifying it as NP-hard [24]

The basics of formulating an ILP problem involve defining several constraints (in this case, hours per worker, times available, varying workload, etc.) and then attempting to maximize or minimize the total number of employees. This is defined by Dai and Huo [25] (Figure 7), where X_{IJ} represents the number of employees assigned to the shift that begins in planning period I of day J. From there, constraints, such as total number of shifts, guarantees that shifts are filled, etc, could be defined.

$$Min\ Z = \sum_{i=1}^{I} \sum_{j=1}^{J} X_{ij}$$

Figure 7: ILP optimization constraint, shift scheduling problem [25]

To help define these constraints, Rai and Chandak [26] have defined several stages to scheduling. They consist of: determining the temporal requirements for staffing levels, constructing a staffing level for each shift, then finally calculating the total number of employees needed. Nie and Liu [27] have gone even further and defined a multileveled approach for staff member of different skill levels.

Of particular note is the paper by Ceneno et al [28], which details the implementation of a simulation-ILP based tool for scheduling ER staff. While their busy emergency room is a very extreme example of schedule formulation, it is worth noting that they follow exactly the same problem formulation steps in that they seek merely to minimize cost, albeit with a more variable demand; their model is able to accumulate a labor cost for shifts during different times of the day, rather than assuming constant staffing needs during a simple 8-5 working scenario.

Much of the surveyed literature refers to open-source implementations of integer linear programing algorithms. Perhaps the foremost of these is the lpsolve ILP solver [29]. Although lpsolve provides a standalone program, it is notable for defining the MPS file format [29] which many ILP implementations take as input.

A brief survey of additional available libraries includes PuLP [30], a python library, Sat4j [31], a Java library for solving SAT and ILP problems, GPLK, the GNU Linear Programming Kit [32], which is aimed at solving large-scale integer-programming problems in C, and Coin-or Branch and Cut (CBC) [33], a C++ library, which is directly callable, or built to be integrated into many other applications. Additionally, there are several standalone programs that act as ILP solvers, including IBM's CPLEX Optimizer [34], and MiniSat+ [35], a SAT solver which can be applied to ILP problems.

Another interesting solution to this problem is the modeling language AMPL (A Mathematical Programming Language) [36]. It provides an entire language for modeling such problems and is supported by several solvers, most notable the aforementioned CBC [33]. This is a popular format for representing problems but is likely overly complex for the implementation needs here.

MATLAB also provides built-in functionality for linear program solvers through the *intlinprog* function [37]. This is particularly useful as it includes full instructions for implementing a demonstration problem and is easily translatable to other subjects such as shift scheduling. Shure shows an approach to using this functionality in MATLAB to determine and plot a viable schedule to a test set of workers read in from an Excel document [38]. Shure approaches this by reading in each employee's, min/max hours per day, their hourly wage, and their availability. She then parses this into tables. Figure 8 shows one of her constraint matrices, wherein the top portion represents the minimum hour requirement (24 topmost rows, one for each hour of the day) and the bottom portion represents the one-shift-a-day requirement (20 bottom rows, one for each employee). Figure 9 shows a plot of her results matrix, parsed and display in a table and plotted in a total cost graph.

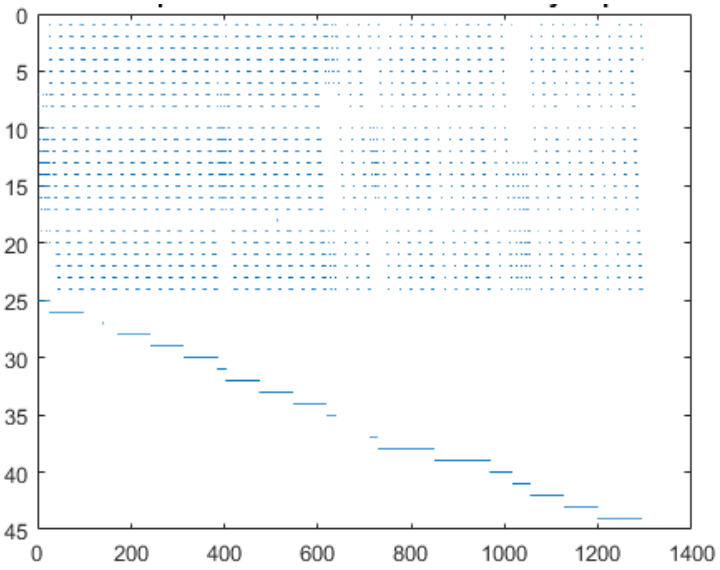

*Figure 8: Shure's ILP '**A**' constraint matrix* [38]

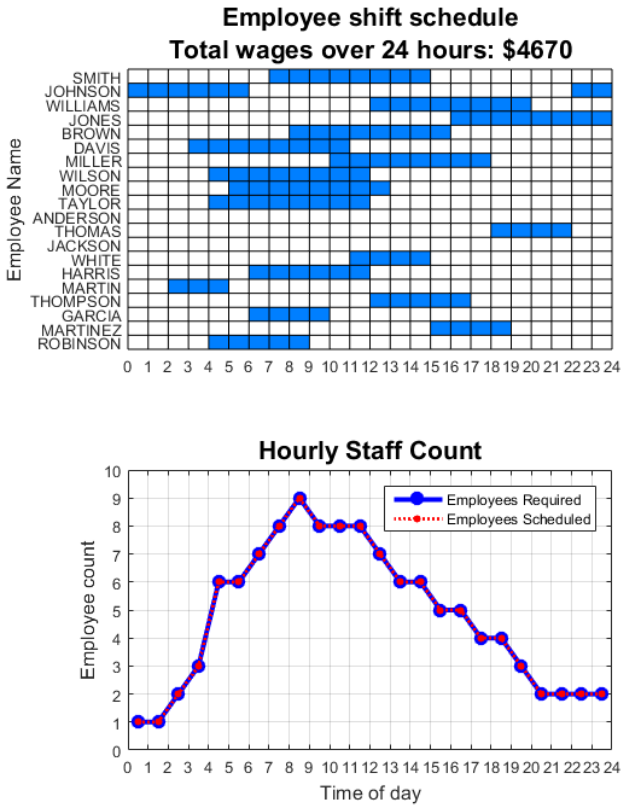

Figure 9: Shure's Plotted results [38]

4 RESULTS AND DISCUSSION

It follows that there are a wide variety of methodologies used to solve the shift scheduling problem. Most fall into the distinct categories of integer linear programming, or those that utilize heuristics. Each has costs and benefits. Additionally, it is worthy comparing the human option, that is, using human intuition as a means to derive a schedule. Although not optimal, it is often simpler and less costly.

4.1 Comparison Chart

Approach	Variety of Solutions	Complexity	Optimality	Availability of Open-Source implementations	Implementation Cost
Human	Many personal methodologies	Simple	Biased	N/A	None
Integer Linear Programming	Integer linear programming	NP-complete	Optimal	Available in most languages	Expensive
Heuristics	Tabu search, simulated annealing, genetic algorithm, etc.	Can tradeoff with computation time	Suboptimal	Mostly only MATLAB, a few C++ & Java	Expensive

Figure 10: Comparison of shift scheduling approaches

4.2 Results Evaluation

In general, implementing a scheduling algorithm boils down to two decisions: Is it worth implanting an algorithm at all? And how optimal must the solution be? While the human solution may be easily discounted, it does have several benefits in that it has no implementation and complexity-wise, it is simple to maintain. However, it is not at all optimal in any regards. The choice between an ILP-based approach and a heuristic approach all depends on how much optimality matters. The single largest difference between the two is the optimality for complexity.

4.3 Future Work

This survey has provided a great deal of options for solving the shift scheduling problem efficiently. However, there is very little work considering using parallel processing to speed up this process. Parallel processing/cloud/shared computing is a growing field and one that seems would have potential to further optimize this problem. Baumelt, et. al published a single work in the European Journal of Operational Research detailing their approach for solving the nurse scheduling problem with a parallel algorithm and cite impressive speedups of up to 15x the same algorithm performed sequentially [39]. Unfortunately, they have not made their paper publicly available for free.

5 CONCLUSIONS

In conclusion, the choice of a scheduling algorithm is highly variable and dependent on the situation at hand. Organizations who have a need to schedule a small amount of staff and have highly variable constraints are best left to calculating a schedule by human hand. For these situations, the cost of implementing and regularly calculating and maintain an algorithm is simply not worth the return of an optimal schedule. Indeed, human scheduler lead to a much greater flexibility due to the fact that they can

adapt easily without losing a great deal of optimality. Nakamura and Salvendy even go so far as to argue that even with the increasing use of technology, the flexibility of human scheduling will always be valued in manufacturing environments [40].

On the other hand, large organizations, such as the big-city hospitals frequency studied in the nurse scheduling problems above, are much better suited to an algorithmic approach. Extremely large organizations could benefit greatly from the optimality of an ILP-based approach, and will likely find the implementation/maintenance cost and complexity of problem formulation worth the return of the cost savings of a slightly more optimal schedule.

All in all, there is no easy answer. The field of shift scheduling algorithms is widely varied for good reason; the real-world situations they model have needs that are just as widely varied.

6 REFERENCES

[1] W. Vickrey, "Counterspeculation, Auction, and Competitive Sealed Tenders," *The Journal of Finance,* 1961.

[2] R. Lagatie, S. Haspeslagh and P. De Causmaecker, "Negotiation Protocols for Distributed Nurse," *Proceedings of the 21st Benelux Conference on Artificial Intelligence,* 2009.

[3] T. B. Cooper and J. H. Kingston, "The Complexity of Timetable Construction Problems," *The University of Synder Technical Report Number 495,* 1995.

[4] B. MacCarthy and J. R. Wilson, Human Performance in Planning and Scheduling, CRC Press, 2003.

[5] L. Augustine, M. Faer, A. Kavountzis and R. Patel, "A Brief Study of the Nurse Scheduling," *Carnegie Mellon School of Computer Science,* pp. 1-11, 15 December 2009.

[6] E. K. Burke, P. De Causmaecker, G. Vanden Berghe and H. Van Landeghem, "The State of The Art of Nurse Rostering," *Journal of Scheduling,* vol. 7, pp. 441-499, 2004.

[7] U. Aickelin and J. Li, "Explicit Learning: an Effort towards Human," *The 1st Multidisciplinary International Conference on Scheduling: Theory and Applications MISTA,* pp. 240-241, 2003.

[8] B. F. Reskin, "Getting It Right: Sex and Race Inequality In Work Organizations," *Annual Review of Sociology,* vol. 26, pp. 707-709, 2000.

[9] P. Stoop and V. Wiers, "The complexity of scheduling in practice," *International Journal of Operations & Production Management,* vol. 16, no. 10, pp. 37-53, 1996.

[10] K. E. Geltman, "The Simulated Annealing Algorithm," 20 February 2014. [Online]. Available: http://katrinaeg.com/simulated-annealing.html. [Accessed 4 May 2017].

[11] Y.-W. Ko, D. Kim, M. Jeong, W. Jeon,, S. Uhmn and J. Kim, "An Improvement Technique for Simulated Annealing and Its Application to Nurse Scheduling Problem," *International Journal of Software Engineering and Its Applications,* vol. 7, no. 4, 2013.

[12] J. Brownlee, "Simulated Annealing," in *Clever Algorithms: Nature-Inspired Programming Recipes,* lulu.com, 2012.

[13] GNU.org, "26.2 Simulated Annealing functions," 2017. [Online]. Available: https://www.gnu.org/software/gsl/manual/html_node/Simulated-Annealing-functions.html#Simulated-Annealing-functions. [Accessed 4 May 2017].

[14] MathWorks, "Minimization Using Simulated Annealing Algorithm," 2017. [Online]. Available: https://www.mathworks.com/help/gads/examples/minimization-using-simulated-annealing-algorithm.html. [Accessed 4 May 2017].

[15] MathWorks, "MathWorks," What Is the Genetic Algorithm?, 2017. [Online]. Available: https://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html. [Accessed 4 May 2017].

[16] M. Ohki, "A parameter free nurse scheduling," *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1-7, 2015.

[17] M. Ohki, A. Morimoto and M. Kosuke, "Nurse Scheduling by Using Cooperative GA with Efficient Mutation and Mountain-Climbing Operators," *Intelligent Systems, 2006 3rd International IEEE Conference*, pp. 164-169, 2006.

[18] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & Operations Research*, vol. 13, no. 5, pp. 533-549, 1986.

[19] E. Burke and E. Soubergia, "Scheduling Nurses Using a Tabu-Search Hyperhueristic".

[20] J. Brownlee, "Tabu Search," in *Clever Algorithms: Nature-Inspired Programming Recipes*, lulu.com, 2012.

[21] M. Maischberger, "METSlib tabu search framework home page," 14 May 2015. [Online]. Available: https://projects.coin-or.org/metslib&usg=AFQjCNGzi39Kk8PgSaTbOiMdgxw8qSqs1A&sig2=gJaoIU0bOYDuXtAEOwYI2A. [Accessed 4 May 2017].

[22] R. Harder, "OpenTS - Java Tabu Search," 2016. [Online]. Available: http://opents.iharder.net/&usg=AFQjCNFH5WlpnxjEOd4yC-_XdrsIGUxRzA&sig2=YoS1PE4YylPSqdiK0p_2Cw. [Accessed 4 May 2017].

[23] C. H. Papadimitriou and K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Dover Books, 1998.

[24] S. Vempala, "Combinatorial Optimization NP-completeness," [Online]. Available: https://ocw.mit.edu/courses/mathematics/18-433-combinatorial-optimization-fall-2003/lecture-notes/l20.pdf. [Accessed 4 May 2017].

[25] T. Dai and J. Huo, "A research on shift scheduling problem in multi-skill call center," *2008 IEEE International Conference on Service Operations and Logistics, and Informatics*, pp. 655-658, 2008.

[26] V. K. Rai and P. Chandak, "Shift planning and scheduling for IT service operations management," *2015 Annual IEEE Systems Conference (SysCon) Proceedings*, pp. 645-652, 2015.

[27] H. Nie and B. Liu, "Combining MILP with Memetic Algorithm for Scheduling and Staffing Construction Project with a Multi-skilled Workforce," *2013 International Conference on Computational and Information Science*, pp. 1150-1153, 2013.

[28] M. A. Centeno, R. L. Giachetti and A. M. Ismail, "A simulation-ILP based tool for scheduling ER staff," *Proceedings of the 2003 Winter Simulation Conference*, vol. 2, pp. 1930-1938, 2003.

[29] "lp_solve reference guide menu," 24 September 2016. [Online]. Available: http://lpsolve.sourceforge.net/5.5/. [Accessed February 2017].

[30] J.-S. Roy, "PuLP 1.1 A Linear Programming modeler," python, 2017. [Online]. Available: https://pypi.python.org/pypi/PuLP/1.1. [Accessed February 2016].

[31] "Sat4j, the boolean satisfaction and optimization library in Java," March 2017. [Online]. Available: http://sat4j.org/. [Accessed 4 May 2017].

[32] A. Makhorin, "GLPK (GNU Linear Programming Kit)," GNU, 23 June 2012. [Online]. Available: http://www.gnu.org/software/glpk/. [Accessed February 2017].

[33] "Cbc (Coin-or branch and cut)," Computation Infrastructure for Operations Research, 26 February 2016. [Online]. Available: https://projects.coin-or.org/Cbc. [Accessed February 2017].

[34] "CPLEX Optimizer," 2017. [Online]. Available: https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/index.html. [Accessed 4 May 2017].

[35] N. Een and N. Sorensson, "The MiniSat Page," 2007. [Online]. Available: http://minisat.se/MiniSat+.html. [Accessed 4 May 2017].

[36] R. Fourer, D. M. Gay and B. W. Kernighan, AMPL: A Modeling Language for Mathematical Programming, Thomson/Brooks/Cole, 2003.

[37] "Mixed-Integer Linear Programming Basics," MathWorks, 2016. [Online]. Available: https://www.mathworks.com/help/optim/ug/mixed-integer-linear-programming-basics.html. [Accessed February 2016].

[38] L. Shure, "Generating an Optimal Employee Work Schedule Using Integer Linear Programming," MathWorks, 6 January 2016. [Online]. Available: https://blogs.mathworks.com/loren/2016/01/06/generating-an-optimal-employee-work-schedule-using-integer-linear-programming/. [Accessed 4 May 2017].

[39] Z. Daumelt, J. Dvorak, P. Sucha and Z. Hanzalek, "A novel approach for nurse rerostering based on a parallel algorithm," *European Journal of Operational Research*, vol. 251, no. 2, pp. 624-639, 2016.

[40] N. Nakamure and G. Salvendy, "Chapter 12: Human Planner and Scheduler," in *Design of Work and Development of Personnel in Advanced Manufacturing*, Wiley-Interscience, 1994, pp. 331-333.