Expert Systems With Applications 195 (2022) 116464

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A novel genetic algorithm based system for the scheduling of medical treatments

Matthew Squires [a], Xiaohui Tao [a,*], Soman Elangovan [b], Raj Gururajan [c], Xujuan Zhou [c], Udyavara Rajendra Acharya [d]

[a] School of Sciences, University of Southern Queensland, Australia
[b] Belmont Private Hospital, Brisbane, Australia
[c] School of Business, University of Southern Queensland, Australia
[d] Singapore University of Social Science, Singapore

ARTICLE INFO

Keywords:
Genetic Algorithm
List Scheduling Wildcard Tournament Genetic Algorithm (LSWT-GA)
Medical scheduling
repetitive Transcranial Magnetic Stimulation (rTMS)

ABSTRACT

The manual scheduling of medical treatment in a health centre is a complex, time consuming, and error prone task. Furthermore, there is no guarantee a manually generated schedule maximises the operational efficiency of the centre. Scheduling problems have seen extensive research across several domains. The current work presents a novel genetic algorithm for the scheduling of repetitive Transcranial Magnetic Stimulation (rTMS) appointments. The proposed List Scheduling Wildcard Tournament Genetic Algorithm (LSWT-GA) combines an innovative survivor selection policy with heuristic population initialisation. The algorithm aims to optimise the operational efficiency of a medical centre through efficient rTMS appointment scheduling. Additionally, the algorithm has the capacity to consider patient priority. Empirical experiments were conducted to evaluate the performance of the proposed algorithm, using a synthetic data set specifically developed to simulate the medical treatment scheduling problem. The experimental results showed the LSWT-GA algorithm outperforms other algorithms, obtaining the optimal makespan more frequently than a List Scheduling Genetic Algorithm (LS-GA) using traditional survivor selection policies and a standard genetic algorithm using random population initialisation (Random-GA). In addition to the novel genetic algorithm, LSWT-GA, the paper also makes a theoretical contribution by evaluating the run time of the LSWT-GA for makespan minimisation. The proposed algorithm and related findings can be applied directly to the administration systems in medical and healthcare centres and helps improve the deployment of medical resources for better treatment effect.

1. Introduction

The manual scheduling of medical treatment in a health centre is a complex (Podgorelec & Kokol, 1997), time consuming (Chien, Tseng, & Chen, 2008; Hamrock, Paige, Parks, Scheulen, & Levin, 2013; Sauré & Puterman, 2014) and error prone task. Furthermore, there is no guarantee the resultant schedule maximises the operational efficiency of the health centre. As such, automated scheduling programs are desirable to reduce work for administrators and improve the efficiency of medical facilities.

Scheduling problems have seen extensive research across several domains and many of these combinatorial optimisation problems are known to be NP-Hard. Despite this, many metaheuristic approaches have been shown to produce good results on complex scheduling problems. Some researchers have taken these scheduling techniques and applied them to the task of medical schedule generation. For example, Petrovic, Morshed, and Petrovic (2011) implemented a genetic algorithm for the scheduling of radiotherapy patients. Similarly, Golgoun and Sepidnam (2018) showed a genetic algorithm to develop optimal appointment schedules for different priority patients.

Medical scheduling can be divided into two types of problems, allocation scheduling or advance scheduling (Dai, Geng, & Xie, 2021; Jiang, Abouee-Mehrizi, & Diao, 2020; Sauré, Begen, & Patrick, 2020). Advance scheduling seeks to develop schedules before the appointment day leaving some variables, such as number of patients unknown. However, allocation scheduling waits until the exact pool of patients is known before making appointments. Allocation problems closely mirror an inpatient hospital setting where patient numbers and treatments are known. This leaves schedulers to distribute hospital resources to

* Corresponding author.
E-mail addresses: matthew.squires@usq.edu.au (M. Squires), xiaohui.tao@usq.edu.au (X. Tao), soman.elangovan@healthecare.com.au (S. Elangovan), Raj.Gururajan@usq.edu.au (R. Gururajan), xujuan.zhou@usq.edu.au (X. Zhou), Rajendra_Udyavara_ACHARYA@np.edu.sg (U.R. Acharya).

https://doi.org/10.1016/j.eswa.2021.116464
Received 16 May 2020; Received in revised form 12 December 2021; Accepted 23 December 2021
Available online 14 January 2022

each patient. The aim of this paper is the development of a system for the scheduling of repetitive Transcranial Magnetic Stimulation (rTMS) appointments.

Depression is a highly prevalent (Australian Beureau of Statistics, 2007; World Health Organisation, 2018) and destructive (Lam, 2018) mental illness. Patients suffering from Depression experience debilitating sadness over a prolonged period of time (Alhanai, Ghassemi, & Glass, 2018). Front line treatments of depression typically consist of medication, psychotherapy or a combination of the two. However, in some cases these treatments are ineffective (Berlim, Fleck, & Turecki, 2008). For these cases rTMS is an alternative evidence based treatment (Conelea et al., 2017; Hovington, McGirr, Lepage, & Berlim, 2013).

rTMS involves electromagnetic stimulation of the brain through coils applied to the patient's scalp (Razza et al., 2018). The applied magnetic field induces an electrical current within the brain which over time alters the underlying structures (George & Taylor, 2014). rTMS sessions vary in their intensity and duration. Patients are prescribed treatment times in advance with a variety of treatment times being available to doctors (Fitzgerald et al., 2020).

Recently the Australian Federal government announced rTMS treatment would be funded under the Medicare Benefit Scheme (Fitzgerald, George, & Pridmore, 2021). This announcement is expected to increase the demand for rTMS treatment. In response to this demand, hospitals seek to maximise the efficiency of allocating rTMS machine time. Automated scheduling systems are one such option for distributing rTMS resources. Scheduling systems have the potential to improve hospital efficiency (Abdalkareem, Amir, Al-Betar, Ekhan, & Hammouri, 2021; Griffiths, Williams, & Wood, 2012) through optimal resource allocation.

In the current paper we evaluate the efficacy of a novel genetic algorithm for the scheduling of rTMS treatments. We focus exclusively on the scheduling of inpatient rTMS treatment, for patients who have been admitted to Hospital. For this problem we assume known treatment times and a static patient pool, that is no additional patients are added to the patient list once job assignment is commenced. As such, the task of scheduling rTMS appointments can be defined as a parallel machine problem of unrelated jobs where preemption is not allowed. That is, once a treatment starts it must be completed. In the first experiment we evaluate the performance of our LSWT-GA to distribute jobs across rTMS machines to minimise completion time. Makespan minimisation of parallel machine problems where $m > 2$ and preemption is not allowed are known to be NP-Hard (Baker, 2019).

Computer scientists have a broad way of classifying problems as either P or NP. Problems in class P have known algorithms which can generate a solution in polynomial time (Eiben, 2015). When a problem can be solved in polynomial time n^k for $k > 0$, the problem is said to be solved (Arora, 2009). An example of a polynomial algorithm might be selection sort, which has a time complexity of n^2 such that sorting a list can be said to be a problem of class P. A second class of problems are the NP problems. These are problems for which a solution can easily be checked in polynomial time, however, no known algorithm exists to solve the problem (Arora, 2009). In general NP problems are more difficult than problems from class P. NP-hard problems do not belong to the class NP (Papadimitriou & Steiglitz, 1998). These problems are more difficult than any NP problem. Problems of class NP-hard have solutions that "cannot necessarily be verified in polynomial time" (Eiben, 2015, p. 11).

For the makespan problem we compare the performance of our novel genetic algorithm to the results obtained by Graham's list scheduling (Graham, 1969). The second arm of the experiment introduces the concept of patient priority. The aim of this second experiment is the minimisation of weighted flowtime such that patients deemed higher priority are seen more urgently. The weighted flowtime problem on parallel machines is also known to be NP-Hard (Baker, 2019). For this problem we compare performance of our LSWT-GA with the H_1 algorithm. The H_1 algorithm is a list scheduling heuristic used for the minimisation of total weighed flowtime (Baker, 2019).

In the current paper we make the following contribution:

- We present a novel genetic algorithm, LSWT-GA, which incorporates an innovative survivor selection policy along with heuristic population initialisation. Scheduling of rTMS treatments is an emerging problem due to increasing demand of mental health services. The genetic algorithm is known to produce good solutions for scheduling of unrelated jobs on parallel machines motivating its use for the current problem. This work forms a pioneer work in the emerging problem of rTMS treatment scheduling.
- We claim a theoretical contribution for our evaluation of the LSWT-GA run time for makespan minimisation. In our review of the literature we note either incomplete or non-existent discussions of algorithm run time. Within this analysis we explore the impact of using heuristics when generating the initial population. Furthermore, we believe for the development of a product to be functionally useful to hospitals it must be able to find optimal solutions in a run time that is practical.
- Given a lack of publicly available data sets for researchers seeking to investigate schedule optimisation for rTMS treatments we developed an original synthetic data. The sensitive nature of data involving patients with psychiatric conditions means it can be difficult to obtain data. As such, we were motivated to create an original data set for use by other researchers. Thus making a methodological contribution to the research community.

The current paper is structured as follows, Section 2 provides a thorough review of the healthcare scheduling literature. In Section 3, we formally define the current problem with scheduling notation. In Section 4 we present the conceptual framework of the current work including a detailed breakdown of the operators used in the LSWT-GA. In Section 5 we outline the computational experiments conducted in the current paper including the hypothesis and synthetic data set. In Section 6 we present our results including a performance comparison between our LSWT-GA, LS-GA and Random-GA followed by some final remarks regarding the current work and future directions of rTMS scheduling.

2. Related work

Manual scheduling of medical appointments takes a significant amount of time. To reduce time spent creating schedules, automated techniques are of interest to hospital administrators. Griffiths et al. (2012) showed automated scheduling of inpatient physiotherapy appointments saved roughly six hours per week in schedule construction time. Researchers have applied a variety of scheduling techniques to the problem of healthcare scheduling. These techniques include, mathematical programming (Braaksma, Kortbeek, Post, & Nollet, 2014; Gartner & Kolisch, 2014), heuristics (Petrovic, Leung, Song, & Sundar, 2006), evolutionary algorithms (Aickelin & Dowsland, 2004; Chien et al., 2008; Golgoun & Sepidnam, 2018; Petrovic et al., 2011; Podgorelec & Kokol, 1997) and data mining (Jiang et al., 2020).

Mathematical programming models are widely used to solve optimisation problems. The models use mathematical relationships to represent characteristics and practical relationships in systems (Williams, 2013). Recently, Gartner and Padman (2017) showed mixed integer programming, a widely used type of mathematical programming was able to increase hospital revenue through more efficient allocation of resources for elective patients. Similarly, Braaksma et al. (2014) showed integer programming was able to more efficiently schedule outpatient rehabilitation sessions across multiple disciplines. While some espouse the strengths of mathematical programming for scheduling problems, others (Chien et al., 2008; Xu & Yang, 2013) assert for large parallel machine problems mathematical programming is not reasonable. For example, Chien et al. (2008) when comparing mixed integer programming with a genetic algorithm showed for small problems ($n = 10$) the two algorithms performed similarly. As such, the genetic algorithm and mixed integer programs arrived at optimal solutions in

similar time. However, for larger problems ($n = 20$) computation time for the mixed integer program quickly grew to 496.5 s compared to 20 s for a genetic algorithm. Xu and Yang (2013) assert for parallel machine scheduling problems with greater than 10 jobs computation time for mathematical programming techniques quickly becomes unreasonable.

For parallel machine problems the genetic algorithm is seen as a state of the art technique (Zang et al., 2019). Consequently, there exist several examples of the genetic algorithm being applied to scheduling in healthcare. For example, Chien et al. (2008) showed the genetic algorithm improved operational efficiency for the scheduling of physical therapy patients in Taiwan. More recently, Golgoun and Sepidnam (2018) employed a genetic algorithm for the scheduling of medical appointments. They observed many classical appointment scheduling systems do not consider patient priority, as such, their genetic algorithm assigned patients weights based on their priority. Under this approach larger weights were assigned to more urgent patients to minimise their wait time. Using a genetic algorithm patient weights can be incorporated into the fitness function. By setting an appropriate fitness function more urgent patients are moved towards the front of the queue. Adopting a similar methodology, Petrovic et al. (2011) showed a genetic algorithm ensured more cancer patients met their radiotherapy due dates. They assigned patients different priorities depending on their cancer status ensuring high priority patients were seen more quickly.

Outside of healthcare scheduling, genetic algorithms have been used on several famous combinatorial problems such as the travelling salesman problem. The knowledge gained in these domains can easily be transferred to scheduling of rTMS treatments. The genetic algorithm involves several highly customisable steps including: selection, crossover, mutation and survivor selection. Within these steps several options are available to researchers. For example, Petrovic et al. (2011) used elitist selection with linear order crossover. Chien et al. (2008) used roulette wheel selection, preserving order based crossover, precedence preserving shift mutation and a 10% elitist survivor selection policy. In their work, Aickelin and Dowsland (2004) provide a detailed analysis of several crossover operators in their work on the nurse scheduling problem. The nurse scheduling problem is a highly constrained task which involves creating nurses work schedules while considering nurse shift preference, fairness, contract requirements, legal requirements and the experience level of staff on each shift. Aickelin and Dowsland (2004) showed partially mapped crossover (PMX) to perform the best for the scheduling of nurse shifts. Table 1 provides an overview of related work, the methods used and the topics explored.

More recently researchers have combined data mining with heuristic techniques such as list scheduling and strict priority scheduling. For example, Jiang et al. (2020) used historical records to forecast demand for future MRI procedures. They contend techniques built and tested on empirical data are likely more practicable in the physical clinical setting. Given this, where possible it is preferred algorithms are tested on empirical data sets. However, given the dearth of literature on the scheduling of rTMS treatments we note a lack of readily available data for researchers to explore this problem.

Many other works (Aickelin & Dowsland, 2004; Chien et al., 2008; Golgoun & Sepidnam, 2018; Petrovic et al., 2011) all provide sound examples of the effectiveness of the genetic algorithm for the scheduling of healthcare appointments. However, we observe each of these studies show limited or no timing analysis. As such it is difficult to know how the genetic algorithm responds to larger queues and hence, their practicality for commercial implementations. Chien et al. (2008) provide a limited discussion of run time but none for queues >20. Furthermore, Aickelin and Dowsland (2004) note for their scheduling algorithm solutions were found in less than 10 s, however, provide no further detail such as error bounds, or commentary on how this run time might scale as generations or queue lengths grow. Given the need for an automated scheduling system for commercial use to run in a reasonable computation time we were motivated to include a timing evaluation and sensitivity analysis to provide a theoretic contribution to the research community. Consequently, we provide original experiments on the speed of the genetic algorithm for appointment scheduling. Furthermore, we note the preference for empirical data sets for developing systems and a shortage of data sets for researchers to explore scheduling of rTMS treatments. As such, in the sections below we provide synthetic data that we believe to be the first rTMS patient database published for use by the research community. Given the literature discussed above we observe the genetic algorithm to be highly customisable thus providing an excellent starting point for the scheduling of rTMS treatments. Literature suggests this is a pioneer work proposing an automated scheduling system for rTMS treatments.

Table 1
Literature summary.

Author	Method	Topic
Jiang et al. (2020)	Data mining and heuristic	MRI schedule forecasting
Rivera, Cisneros, Sánchez-Solís, Rangel-Valdez, and Rodas-Osollo (2020)	Genetic algorithm	Surgery scheduling
Golgoun and Sepidnam (2018)	Genetic algorithm	Patient scheduling including priority
Zhao, Chien, and Gen (2015)	Genetic algorithm	Rehabilitation scheduling
Gartner and Kolisch (2014)	Mixed integer programming	Scheduling of elective patients to maximise hospital revenue
Braaksma et al. (2014)	Integer linear program	Outpatient rehabilitation scheduling
Griffiths et al. (2012)	Local search	Physiotherapy appointment scheduling
Petrovic et al. (2011)	Genetic algorithm	Scheduling cancer treatments
Chien et al. (2008)	Genetic algorithm	Rehabilitation patient scheduling
Aickelin and Dowsland (2004)	Genetic algorithm	Nurse scheduling
Petrovic et al. (2006)	Algorithm/Heuristic	Radiotherapy scheduling
Podgorelec and Kokol (1997)	Genetic algorithm	Generic solution applied to physical therapy

3. Problem definition

In this section we formally define the notation for the scheduling of rTMS sessions for the treatment of depression. For the purpose of this research let m be a set of rTMS machines such that, $m = \{m_1, m_2, m_3, \dots, m_i\}$ where $i = |m|$. Let P be a set of patients queued for a treatment where, $P = \{P_1, P_2, P_3, \dots, P_j\}$ and $j = |P|$. Let p_j denote the treatment time of patient P_j such that $\{p_j \in \mathbb{Z} | 20, 30, 45\}$

Eq. (1) formally defines the scheduling problem we seek to optimise. The problem can be classed as makespan minimisation of a parallel machine problem with deterministic job processing times with preemption not allowed.

$$P \parallel C_{max} \tag{1}$$

where we seek:

$$\min \ C_{max}$$

The objective of the first arm of this experiment is the minimisation of total job processing time denoted by C_{max}.

For the second arm we aim to minimise weighted flowtime F_w as described by Eq. (2). For this problem we apply the same constraints as part 1 however we add in patient priority weights, w. Using these weights we seek to minimise the flowtime of the patient pool.

$$P \parallel F_w \tag{2}$$

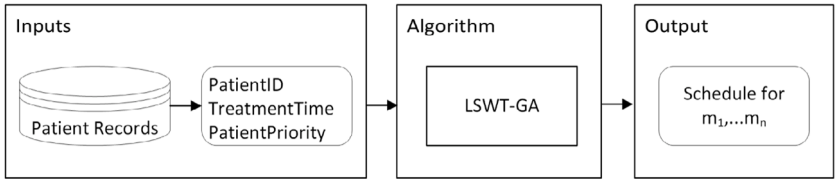

Fig. 1. Scheduling process.

Table 2
Notation summary.

Symbol	Description
m	rTMS machine
m_i	rTMS machine number i
P	Set of patients
P_j	Patient j
p	Processing time of patient. The length of rTMS session
p_j	Treatment time for patient j
F_w	Weighted flowtime
w_j	Priority weighting assigned to patient j

where:

$$F_w = \sum_{j=1}^{x} w_j F_j$$

Let F_j denote the flowtime of patient P_j. Where flowtime is defined by Eq. (3):

$$F_j = C_j - r_j \tag{3}$$

And we hope to obtain:

$$\min \ F_w$$

Such that C_j is the completion time of treatment for patient P_j and r_j is the release time of the job. Let w_j denote the weight assigned to a patient P_j where $\{w_j \in \mathbb{Z} | 1, 2, 5, 10\}$.

For convenience, Table 2 provides a summary of the notation used and a description for each symbol.

4. Methodology

4.1. Framework

In this section we introduce the proposed LSWT-GA. The algorithm seeks to automate the scheduling of rTMS treatments. Fig. 1 shows patient records being input into the scheduling algorithm. The Patient records include a patient's unique id number, treatment time and priority level. These values are input into the scheduling algorithm which distributes jobs across available machines to produce a schedule. In a practical sense, hospital administrators input a set of patients and the number of available rTMS machines, following input, the algorithm will then output a schedule for the planned rTMS sessions.

Fig. 2 provides a more detailed overview of the components within the scheduling algorithm. From the figure we see a population of size μ is initialised. Chromosomes for reproduction are picked using a fitness proportionate roulette wheel selection method where two parent chromosomes are selected. These chromosomes produce offspring using the partially mapped crossover (PMX) and swap mutation before a modified survivor selection policy is implemented. This process continues for the specified number of generations g. Further technical details are explained in the following section.

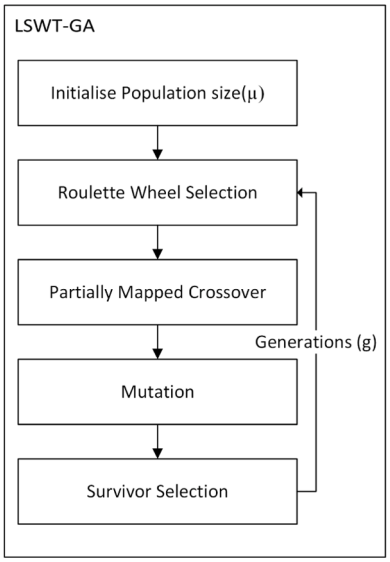

Fig. 2. Genetic algorithm modules.

4.2. LSWT-GA

In this section we present our novel genetic algorithm for use in the current work. Inspired by evolution, the genetic algorithm builds upon natural selection by selecting the fittest chromosomes, as assessed by the fitness function, for the reproduction steps of the genetic algorithm (Goldberg, 1989). Within each phase of the algorithm there are many types of operators to be selected from. Algorithm 1 provides an overview of the genetic algorithm used for the current work. Line 4 describes the heuristic population initialisation strategy used in the current work (see Algorithm 2). Lines 22 and 23 highlight the novel survivor selection policy of the LSWT-GA. Further details about this approach can be seen in Section 4.2.6. Similarly, further discussion on the operators used throughout the algorithm are included in the sections below.

4.2.1. Chromosome representation

For parallel machine problems, chromosome representation must be modified. The current algorithm uses a representation inspired by Ak and Koc (2012). They propose using dummy activities to populate the chromosome to reduce the need for any repair function. For example, if we let jobs $= J$ such that $J = \{J_1, J_2, J_3\}$ and $m = 3$, the chromosome would take the form of Fig. 3. In our representation dummy jobs are represented by negative integers. Using this representation chromosome length will always be of length $|J| \times m$.

4.2.2. Population initialisation

For both problems random population initialisation was attempted, however, random initialisation was unable to find optimal solutions as frequently as the proposed method. Fig. 7 provides a performance comparison of random genetic algorithms compared with algorithms which initialised the population using list scheduling. In the LSWT-GA

Algorithm 1: Genetic Algorithm

```
Input: Population size, μ
       Number of Generations, g
       A set of machines, m
Output: C_max, F_w, Schedule
Data: Set of Patients P
def Fitness(Chromosome):
    computeFitness(Chromosome)
    return Fitness
def InitialisePopulation(μ, P,m):
  for i in range(μ) :
      Chromosome = ListScheduling(P)
      Fitness(Chromosome)
  return Population
def Selection(Population):
  return Parent1, Parent2
def Reproduction(Parent1, Parent2):
    PMX(Parent1, Parent2)
    SwapMutation(Parent1, Parent2)
    return Offspring1, Offspring2
def GeneticAlgorithm(μ, g, P, m):
  InitialisePopulation(μ, P)
  for i in range(g) :
      Selection(Population)
      Reproduction(Parent1, Parent2)
      Fit = max(Fitness(Offspring1, Offspring2))
      MinPopulation = min(Fitness(Population))
      if MaxOffspring > MinPopulation then
          Swap(MaxOffspring, MinPopulation)
      end
  return Schedule, C_max, F_w
```

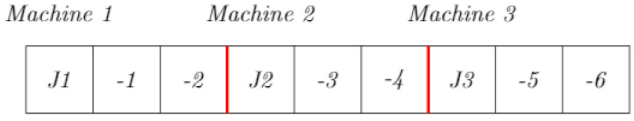

Fig. 3. Chromosome representation for parallel machine problem.

the population was initialised using list scheduling. As such, each chromosome was constructed by applying the heuristic scheduling method to a job queue then implementing the chromosome representation described above. Similarly for $P \parallel F_w$ random initialisation achieved poor results. Results were greatly improved by sorting jobs in priority order.

4.2.3. Selection

Selection is the first step in the reproduction process of the genetic algorithm. Our LSWT-GA uses a fitness proportionate roulette wheel selection policy. Roulette wheel selection is a widely used genetic algorithm selection operator (Lipowski & Lipowska, 2012).

In roulette wheel selection each chromosome has its fitness evaluated by the specified fitness function. For the makespan problem fitness was defined by dividing M^* by C, where C is the completion time of a given chromosome and M^* is the optimal schedule as found by list scheduling. The fitness function for the makespan minimisation problem is seen in Eq. (4).

$$Fitness = \frac{M^*}{C} \tag{4}$$

Examining the behaviour of this function we see that as C increases the fitness of a schedule decreases. As such, larger makespans are deemed less fit and less likely to be carried forward to the next generation.

For the second arm of the experiment a similar strategy is used. In the second phase we seek the minimisation of weighted flowtime F_w. To reflect this different aim the fitness function is adjusted to Eq. (5).

$$Fitness = \frac{B(m)}{F_w} \tag{5}$$

where B_m is the theoretical lower bound obtained from Eq. (10). Under this procedure larger values of F_w result in smaller fitness values meaning the chromosome is less likely to be included in the next generation.

These fitness values then inform the probability of chromosome being selected as a mate for reproduction. Selection probabilities within a population are given by Eq. (6).

$$P_j = \frac{Fitness_j}{\sum_{j=1}^{n} Fitness} \tag{6}$$

Using Eq. (6) each chromosome is assigned a selection probability. From there, selection of mates are drawn from what can be thought of as a biased roulette wheel (Goldberg, 1989). Schedules with greater fitness have higher selection probabilities on the roulette wheel and are more likely to be selected for reproduction.

4.2.4. Crossover

For enumerated chromosomes the crossover method must yield a valid chromosome. The PMX (Goldberg & Lingle, 1985) crossover method was constructed for use in the TSP. For non-binary chromosomes PMX is a widely used crossover operator (Ting, Su, & Lee, 2010). PMX randomly finds two cut points along a chromosome. The section between these two cut points is mapped to the offspring. Fig. 4 provides an example of reproduction using PMX.

The red section denotes the crossover section, first we populate the offspring chromosome with the crossover section denoted by $\{-2, J3, -3\}$. The offspring is then populated with chromosomes from mate 1 unless the chromosomes appear in the crossover section from mate 2. If the chromosomes appear in the crossover section a mapping is used so there is no repetition within the chromosome. For example, the mapping component for Fig. 4 is $-2 \rightarrow -3 \rightarrow -4$ and $J3 \rightarrow J2$. These mappings are essential to ensure the chromosome produces a feasible solution.

4.2.5. Mutation

Following crossover each offspring undergoes mutation. In the current paper swapping mutation was used. Swapping mutation involves the selection of two random jobs and swaps them. Built into the mutation operator was the condition that at least one job being swapped was a job rather than dummy activity. This condition was built in to improve performance so mutation was not a trivial swap of dummy jobs.

4.2.6. Survivor selection

After mutation we must decide which chromosomes to carry forward to the next generation. Several fitness based survival criteria were tested. Fitness-based survivor selection policies keep the chromosomes that best satisfy the fitness function, while discarding the ones with lower fitness values.

In the current implementation the LSWT-GA incorporates a novel survivor selection policy. In a standard genetic algorithm, for example, in the work done by Ahmed (2010), the population μ plus offspring λ is sort from most fit to least and only chromosomes in the top μ are retained in the next generation. Under this survivor policy in the worst case none off the offspring will remain in the next population or in the best case both offspring are included in the next generation. This approach was used as the basis of the LS-GA and Random-GA 6.3. However, some authors argue methods which replace the least fit chromosomes are known to have high mean fitness but can converge quickly on sub optimal local minima (Eiben, 2015). By keeping some

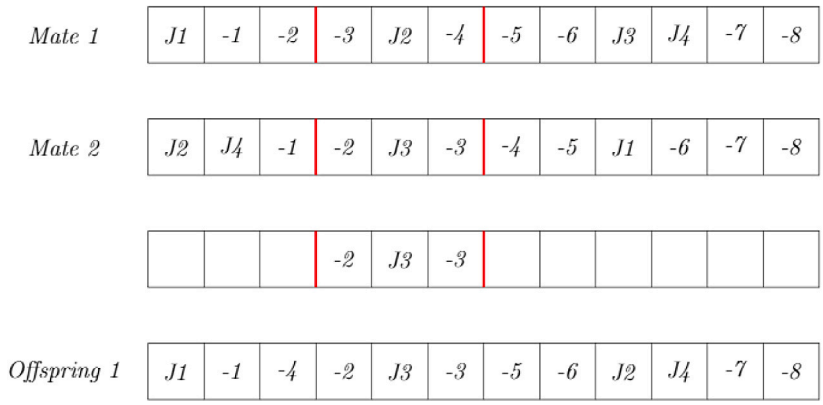

Fig. 4. Partially mapped crossover.

diversity in the population it increases the chances of the algorithm leaving local minima and finding globally optimal solutions. As such, the LSWT-GA uses a modified selection policy where only the number 1 ranked offspring is included in the next generation if it is fitter than the least fit chromosome in the current population. This process shown in lines 22 and 23 of Algorithm 1 allows the population to maintain enough diversity to escape local minima and converge on the global optimum.

5. Experimental design

This section outlines the computational experiments to test the hypothesis outlined in the sections below. Each algorithm was tested on a synthetic data set. The initialisation of this data was informed by the depression and rTMS literature. Further details on the construction of this data are included in Section 5.1.

For the first arm of the experiment several simulations were run on various sample sizes. The algorithm received a patient's ID and rTMS treatment time. For the purpose of this research treatment time was treated as deterministic and assumed to encompass all set up and finishing time for a patient. From the initial data set several samples were drawn. For each sample a simulation was run using list scheduling (Graham, 1969) to find the likelihood or obtaining the optimal schedule by chance. The simulation involved randomly shuffling the job queue then assigning each patient to the first available rTMS machine. These results were recorded and provide the theoretical probability of a schedule maker achieving the optimal makespan under a generic scheduling strategy. Following this a simulation was created where the list scheduling heuristic was run ten times and the schedule with the shortest makespan selected. This simulation allows for a fair comparison between any genetic algorithm and list scheduling procedure by ensuring the performance of the genetic algorithm cannot be attributed solely to the optimal schedule being initialised in the initial population μ. The list scheduling simulation was run 1000 times and compared to 1000 runs of the LSWT-GA. Included within this simulation the resultant schedule from each run is validated. This process confirms the feasibility of the schedule ensuring all patients are assigned to an rTMS machine.

In Section 6.3 comparisons between LSWT-GA, LS-GA and Random-GA were compared for a sample size of 10 and 1000 runs of each algorithm. For each algorithm run time was collected to compare the speed of each algorithm. These times were collected using Python's inbuilt timing function 'timeit.'

The second arm of the experiment, the flowtime problem used a slightly different procedure. The H_1 algorithm (Baker, 2019) produces only 2 schedules depending on the selected R algorithm. As such, there was no need to repeat the process as the same result is obtained each time. For the genetic algorithm however, the simulation was run 1000 times to visualise both the performance and probability of achieving minimum flowtime. Section 5.4 provides a detailed breakdown of the technical steps of the H_1 algorithm while Section 5.5 provides an overview of the comparison between algorithms.

Table 3
Synthetic data description.

Name	Description
Patient ID	A unique identifier
Name	A random name generated by the faker package
Treatment time	TMS treatment time in minutes
MADRS	MADRS score
DASS-D	DASS-Depression score
DASS-A	DASS-Anxiety score
DASS-S	Dass-Stress score
Patient priority	Treatment priority with 1 being highest priority
Weights	Weights for the F_w problem

5.1. Data set

Due to the sensitive nature of data for patients with psychiatric conditions a synthetic data set was generated. The data was constructed using Python's random and faker package. The constructed data set was then used to construct a MySQL database which forms the permanent data set used for the current research. The data set contains simulated results for two psychiatric diagnostic questionnaires the Montgomery–Asberg Depression Rating Scale (MADRS) (Montgomery & Asberg, 1979) and the Depression Anxiety Stress Scales (DASS) (Lovibond & Lovibond, 1995). The MADRS is a clinician administered semi structured interview that generates a score from 0 to 60 with higher scores indicating more severe depression. The DASS however, is a self report questionnaire that patients complete. Scores are then interpreted by clinicians. The DASS yields results for three dimensions of mental health, depression, anxiety and stress. In the current work DASS-21 a short form of the 42 item DASS was used to generate scores. In the DASS-21 each dimension has a maximum score of 42. For depression a score greater than 21 is deemed severe depression (Beaufort, De Weert-Van Oene, Buwalda, de Leeuw, & Goudriaan, 2017).

Scores for each questionnaire were randomly simulated in the range of allowable results for the given test. Scores were generated using Python's random package. Further detailed description of the synthetic data set fields can be seen in Table 3.

The utilisation of quantitative methods for assigning patient priority based on combinations of psychometric questionnaires is an emerging problem in mental health services. As a crude approximation of priority we used a linear combination of each patient's normalised MADRS and DASS-D score summed creating a severity of depression index. The results of this linear combination were sorted from largest to smallest and divided into quartiles with the first quartile being assigned a

Table 4
Priority weighting.

Priority level	Weighting
1	Penalised 10 unit for every minute waiting
2	Penalised 5 units for every minute waiting
3	Penalised 2 units for every minute waiting
4	Penalised 1 units for every minute waiting

patient priority level of 1 and the quartile with the lowest score being assigned a priority of 4. The process described above can be thought of as a computational approach to triage. Using this approach means people who record high scores on both MADRS and DASS-D will have higher depression index and hence higher priority.

Applying a method based on the work done by Golgoun and Sepidnam (2018), weights are assigned using the mapping in Table 4. These weights w will be used to formulate the F_w problem.

Patients were randomly assigned to treatment times of 20,30 or 45 min. These treatment times were informed by rTMS literature (see Fitzgerald et al., 2020) and hospital protocols. Detailed descriptive statistics for this data set are included in Section 6.

5.2. Hypothesis

This paper proposes two experiments. The first experiment seeks to test whether the novel LSWT-GA obtains the optimal makespan more frequently than Graham's list scheduling (Graham, 1969).

The second experimental arm seeks to investigate whether the LSWT-GA can obtain the flowtime of the H_1 algorithm (Baker, 2019) for the weighted flowtime problem $P \parallel F_w$.

As an adjunct to these hypothesis we also compare the performance of our LSWT-GA against a LS-GA and Random-GA to evaluate the performance of the novel operators within our genetic algorithm

5.3. List scheduling algorithm

The C_{max} problem on parallel machines is known to be NP-Hard (Baker, 2019). Graham (1969) put forward a simple heuristic for the scheduling of non related jobs across m identical machines. The algorithm takes a list of jobs J and distributes each job to the first available machine. Below Algorithm 2 provides an algorithmic overview of the list scheduling procedure.

Algorithm 2: List Scheduling

Input: Processing times for a set of jobs, J
A set of machines m
Output: C_{max}

```
def ListScheduling(J, m):
    random.shuffle(J)
    for i in range(J) :
        FirstAvailable = min(m)
        FirstAvailable.append(J)
    C_max = max(m)
    return C_max
```

Both Galambos and Woeginger (1993) and Graham (1969) assert this algorithm comes with a performance guarantee that the obtained schedule will be no worse than:

$$\frac{M}{M^*} \leq 2 - \frac{1}{m}$$

where M denotes the obtained schedule, M^* the optimal schedule and m the number of available machines. From this equation for $m = 4$ we know a schedule obtained by list scheduling is no worse than $\frac{7}{4}$ times larger than the optimal schedule. It must be noted, the performance of list scheduling is heavily influenced by the initial queue. For this reason before each run the queue was randomly shuffled (line 2), otherwise the algorithm would produce the same result for each run.

5.4. H_1 Algorithm

The H_1 algorithm for makespan minimisation of unrelated parallel machine problems incorporates many of the same ideas as the list scheduling procedure discussed above. The flowtime minimisation, or F_w problem is also known to be NP-hard (Baker, 2019). Per suggested by Baker (2019), the H_1 algorithm involves the following steps:

1. Sort jobs according to some rule (SWPT, LWPT)
2. Distribute jobs to first available machine as per list scheduling
3. Apply SWPT jobs on each machine

The expression below gives the ratio of processing time to weights of each treatment

$$\frac{p_j}{w_j}$$

When sorting by shortest weighted processing time (SWPT) jobs are sorted by this ratio from smallest to largest. Under longest weighted processing time (LWPT) jobs are sorted by this ratio in descending order.

5.5. Performance measuring scheme

To compare the performance of the genetic algorithm several hypothesis tests were used. For $P \parallel C_{max}$ where possible the Chi-Square test, shown in Eq. (7), was used to compare results. For cells with less than 5 results it was not possible to use Chi-Square. In these situations, Cramer's V, shown in Eq. (8), was used.

$$\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected} \tag{7}$$

$$\phi_c = \sqrt{\frac{\chi^2}{n(k-1)}} \tag{8}$$

where, Cramer's V, denoted by ϕ_c uses the conventional notation. That is, n is the size of the sample, and k the number of variables being evaluated.

Additionally a single tail proportion test was used to validate the difference between algorithms and confirm $P_{Observed} > P_{Expected}$.

$$z = \frac{(P_{Observed} - P_{Expected})^2}{SD(P_{Observed})} \tag{9}$$

Where

$$SD(P_{Observed}) = \sqrt{\frac{P_{Observed} \cdot Q_{Observed}}{n}}$$

For $P \parallel F_{max}$ performance was assessed by comparing which algorithm returned results closer to the lower bound $B(m)$.

6. Evaluation results

6.1. Descriptive statistics

In the following section we outline the descriptive statistics for the synthetic data set.

Starting with the variables for Experiment 1, $P \parallel C_{max}$, Table 5 describes the distribution of treatment times drawn from the synthetic data set.

Using sample 1 ($n = 10$), Fig. 5 shows the probability of obtaining a given makespan following the list scheduling procedure. From this graph we can see the chance of obtaining the optimal schedule by list scheduling without repetition is $\frac{29}{1000}$. We use this distribution as

Table 5
Makespan problem sample descriptives.

Sample size	Treatment time	Frequency
10	20	3
	30	4
	45	3
20	20	6
	30	7
	45	7
30	20	11
	30	12
	45	7
40	20	17
	30	12
	45	11
50	20	20
	30	17
	45	13

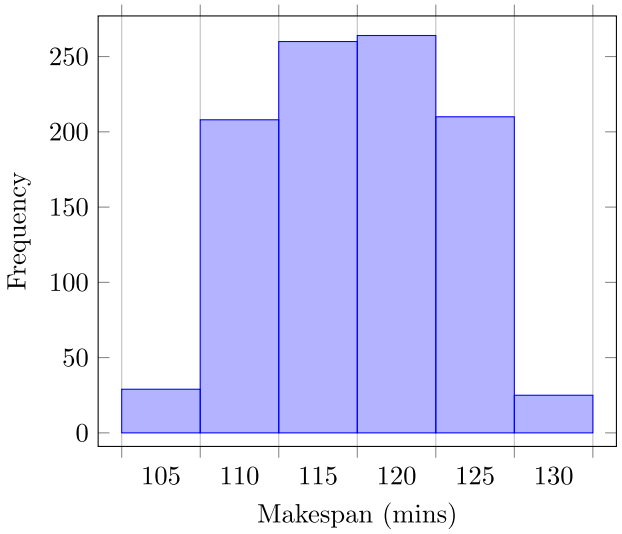

Fig. 5. Distribution of makespans obtained by list scheduling when n = 10.

a theoretical representation of possible probabilities obtained by a schedule designer using a simple scheduling routine.

Using a binomial probability distribution we can see for sample 1, $n = 10$ the likelihood of the optimal schedule being initialised in the initial population of the genetic algorithm for $\mu = 10$ is:

$$Pr(M^* \geq 1) = 0.255$$

6.2. Makespan minimisation

Table 6 shows the results of 1000 simulations of 10 runs of the list scheduling algorithm. For each 10 runs of the list scheduling algorithm the smallest makespan was recorded. This was repeated 1000 times. This is necessary to test the performance of the proposed LSWT-GA. This procedure ensures the performance of the genetic algorithm cannot be attributed to the optimal schedule being in the initialised population by chance. Given the survivor selection policy selected, if the optimal schedule was found by chance, it would be carried through to the final output. As such, this performance cannot be attributed to the performance of the LSWT-GA. Hence, we are interested in the number of suboptimal schedules that are turned into optimal schedules using the genetic algorithm.

Comparing Tables 6 and 7 we can compare the frequency at which the optimal schedule was obtained. Only sample 1 has enough values

Table 6
Makespans from 1000 list scheduling.

Sample size	Makespan	Frequency	Percentage frequency (%)
10	105	210	21.0
	110	677	67.7
	115	113	11.3
20	215	243	24.3
	220	698	69.8
	225	58	5.8
	230	1	0.1
30	300	559	55.9
	305	434	43.4
	310	7	0.7
40	400	572	57.2
	405	419	41.9
	410	9	0.9
50	500	571	57.1
	505	421	42.1
	510	8	0.8

Table 7
Makespans obtained from 1000 LSWT-GA runs (g = 40).

Sample size	Makespan	Frequency	Percentage frequency (%)
10	105	441	44.1
	110	551	55.1
	115	8	0.8
20	215	378	37.8
	220	618	61.8
	225	4	0.4
30	300	829	82.9
	305	171	17.1
40	400	809	80.9
	405	191	19.1
50	500	807	80.7
	505	193	19.3

Table 8
Results of Cramer's V.

Sample size	ϕ_c
20	0.203
30	0.295
40	0.260
50	0.260

in each cell to run a Chi-Square test. For sample 1 we get $\chi^2 = 375.12$ and $p < 0.001$. Hence for $n = 10$ we can reject the null hypothesis and accept that each algorithm produces different results.

For the remaining samples we use Cramer's V. The results of which can be seen in Table 8

The results of Cramer's V show moderate to low association between algorithm and makespan. This result is surprising given we observe large differences between the frequency at which the optimal schedule is obtained shown in Tables 6 and 7. To further evaluate algorithmic performance we can use the proportion test to verify the observed difference between the frequency each algorithm obtained the optimal makespan. Table 9 shows for each sample we can confirm the LSWT-GA achieves the optimal schedule significantly more frequently than the list scheduling algorithm.

As such from this section we can conclude for the scheduling of patients to three rTMS machines our LSWT-GA obtains the optimal makespan significantly more frequently than the list scheduling algorithm.

6.3. Sensitivity and timing analysis

In this section we compare the sensitivity and the run time of the list scheduling procedure, LSWT-GA, LS-GA and Random-GA. The

Table 9
Optimal schedule frequency proportion test.

Sample size	Percentage frequency of optimal schedule		$p < x$
	List scheduling	LSWT-GA	
10	21.0	44.1	0.001
20	24.3	37.8	0.001
30	55.9	82.9	0.001
40	57.2	80.9	0.001
50	57.1	80.7	0.001

Table 10
List scheduling computation time (milliseconds).

Sample size	Milliseconds	$\pm SD$
10	0.0248	.00308
20	0.0585	.0109
30	0.0777	.00594
40	0.118	0.011
50	0.165	0.011

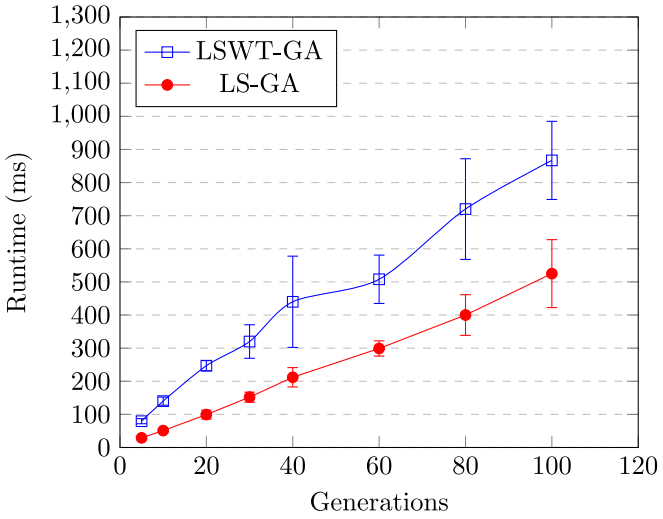

Fig. 6. Run time for genetic algorithm against g.

sensitivity analysis evaluates the percentage frequency the optimal solution is found by each algorithm for $n = 10$.

Using experimental data to consider the run time of list scheduling, Table 10 shows the run time in milliseconds for each sample size. Table 10 shows the list scheduling procedure takes minimal computation time and can easily be incorporated into a genetic algorithm without negatively impacting run time.

Looking at Fig. 6 we see a linear increase in time for both LSWT-GA and LS-GA as g increases. The LSWT-GA takes slightly longer to compute compared to LS-GA, however, finds the optimal solution more frequently as shown in Fig. 7.

Furthermore, in Fig. 7 when comparing the frequency the optimal schedule was obtained by each algorithm we see for all values of g the LSWT-GA outperforms both the LS-GA and Random-GA. For each algorithm as the number of generations increases so does the frequency at which the optimal schedule is obtained. However, as shown in the timing analysis the computation grows as the number of generations grows larger. As such, a trade off must be made, larger values of g increase the algorithm's scheduling performance but reduce the efficiency. As such for this paper we settled on setting $g = 40$ to ensure the run time of large simulations did not become unmanageable while still achieving good performance.

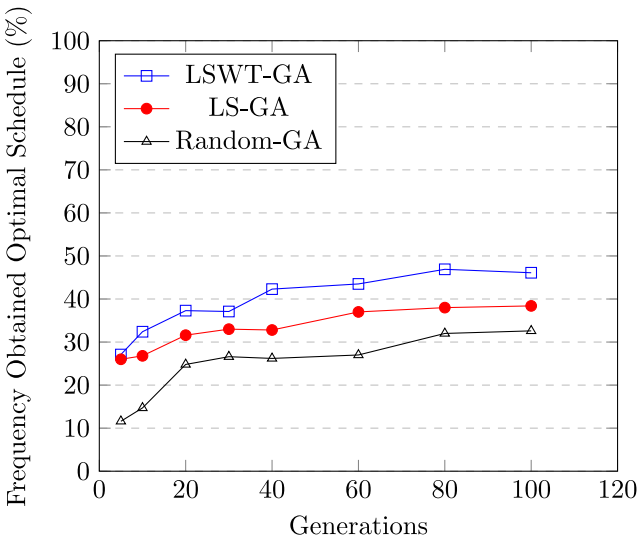

Fig. 7. Optimal schedule percentage for 1000 trials of sample size = 10 against g.

6.4. Flowtime

The distribution of Patient priority for $P \parallel F_w$ problem can be seen in Table 11. Mathematically we can calculate a lower bound for the optimum F_w. Inspired by Baker (2019), we see Eq. (10):

$$B(m) = \frac{1}{2m}[(m-1)B(n) + 2B(1)] \tag{10}$$

where;

$$B(1) = F_w \, using \, SWPT \, on \, a \, single \, machine$$

and

$$B(n) = F_w \, for \, jobs \, distributed \, over \, n \, machines$$

where n is the number of jobs. In combination these processes provide a lower bound approximation for $P \parallel F_w$.

This equation can be used to compute a theoretical lower bound for the flowtime of the data described in Table 12, where For $m = 3$ and $n = 6$ we get:

$$B(6) = 20 \cdot 10 + 30 \cdot 10 + 30 \cdot 5 + 30 \cdot 5 + 20 \cdot 2 + 45 \cdot 2$$

Hence $B(6) = 930$. Furthermore, for $B(1)$ using SWPT we get $B(1) = 2260$. Using Eq. (10) we obtain:

$$B(3) = \frac{1}{2 \cdot 3}[(3-1) \cdot 930 + 2 \cdot 2260]$$

$$B(3) = 1063$$

Table 13 presents the F_w results using the H_1 algorithm outlined in the sections above.

For the data from Table 12 the LSWT-GA achieved a $F_w = 1150$ and $C_{max} = 75$.

Applying the H_1 algorithm to the full data set described in Table 11 we achieve the results shown in Table 14.

Both Tables 13 and 14 highlight in terms of the H_1 algorithm, for the current work $R = SWPT$ outperforms $R = LWPT$ for the flowtime problem. However, in both cases LWPT obtains a better makespan.

In the best case, Fig. 8 shows the LSWT-GA is unable to obtain a weighted flowtime for either R used in the H_1 algorithm.

To boost the performance of the LSWT-GA following the completion of the algorithm a SWPT shift was applied on each machine. The results of which can be seen in Fig. 9 with the best case $F_w = 34600$. This

Table 11
Distribution of patient priority across treatment times. This lower bound can then be used by the LSWT-GA fitness function.

Treatment time	Priority	Frequency
20	1	5
	2	4
	3	6
	4	5
30	1	4
	2	4
	3	3
	4	6
45	1	4
	2	4
	3	3
	4	2

Table 12
Random sample for n = 6.

Patient ID	Treatment time	Weight	$\frac{p_j}{w_j}$
13	20	10	2.0
33	30	10	3.0
36	30	5	6.0
24	30	5	6.0
4	20	2	10.0
10	45	2	22.5

Table 13
F_w and C_{max} using H_1 algorithm for n = 6.

R	F_w	C_{max}
LWPT	1180	65
SWPT	1150	75

Table 14
F_w and C_{max} using H_1 algorithm for n = 50.

R	F_w	C_{max}
LWPT	34 560	500
SWPT	34 550	515

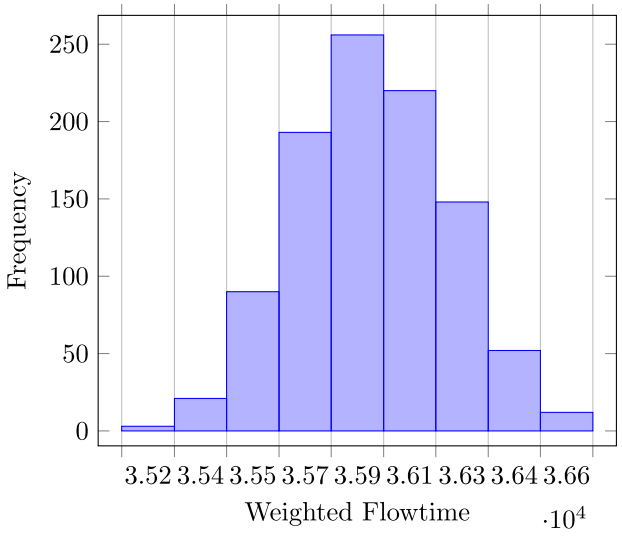

Fig. 8. Distribution of flowtime.

value is still higher than those achieved by the H_1 heuristic shown in Table 14. Even after incorporating a SWPT shift on each machine the LSWT-GA was unable to achieve the flowtime of the H_1 algorithm.

Given these results we can draw two conclusions, for a scheduling system that seeks to minimise the total time to complete all jobs we recommend the LSWT-GA. This method was shown to achieve optimal schedules far more frequently than other methods. For developing schedules based on patient priority we suggest using Baker's H_1 algorithm (Baker, 2019) with $R = SWPT$ for the minimisation of weighted flowtime.

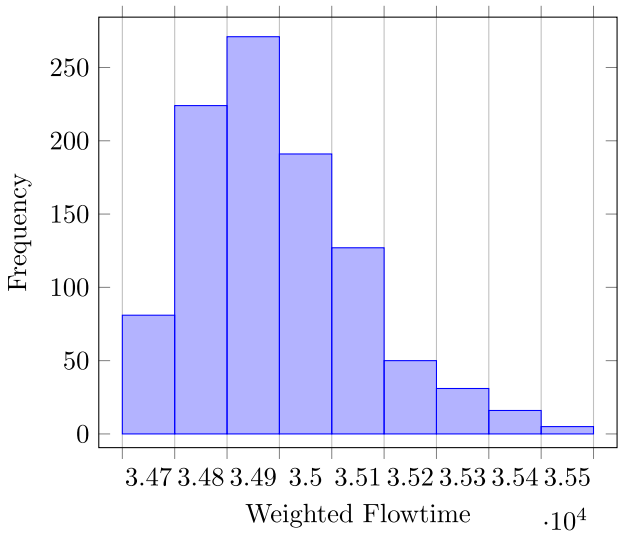

Fig. 9. Distribution of flowtime following SWPT shift.

However, it must be noted there is a trade off, as shown in Tables 13 and 14 optimising patient priority may mean extending total job processing time. As such, a proposed automated scheduling system should allow hospital administrators to choose between priority scheduling and the minimisation of the makespan.

One possible middle ground could be the use of the genetic algorithm for makespan minimisation followed by a SWPT shift on each machine to ensure more urgent patients are seen earlier in the schedule. This guarantees optimal time efficiency while also recognising patient priority.

7. Conclusion and future work

In this paper we present a novel genetic algorithm, LSWT-GA, for automated schedule optimisation of rTMS treatments. We explore the use of this algorithm for the $P \parallel C_{max}$ makespan minimisation problem and the $P \parallel F_w$ weighted flowtime minimisation problem.

Our results show an automated scheduling system built using our LSWT-GA to produce the optimal schedule for $P \parallel C_{max}$ more frequently than other approaches and in a reasonable computation time. Furthermore, we show LSWT-GA outperforms a simple list scheduling procedure. However, for priority scheduling the LSWT-GA was unable to improve on the H_1 algorithm.

These results suggest in urgent situations where patient priority is the determining factor the H_1 algorithm should be used. Otherwise, in situations where total job processing time is the only consideration the genetic algorithm is the preferred option. An alternative option for minimising makespan while considering patient priority is using the genetic algorithm to obtain the optimal schedule then sorting jobs on each machine by SWPT.

This work is an originate research exploring automated scheduling of rTMS treatments. The research provides a useful starting point for the automated scheduling of rTMS appointments and other medical treatments with deterministic treatment times. However, the authors acknowledge the current work is limited by the lack of constraints placed on the scheduling system. This work does not consider alternative patient centred variables such as patient preference or provider request. Future work should incorporate dynamic arrivals, machine

down time, patient preference and provider request to more closely mirror a real world setting. Furthermore, future work may seek to consider patient priority in terms of due dates as opposed to weighted flowtime.

CRediT authorship contribution statement

Matthew Squires: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing. **Xiaohui Tao:** Conceptualization, Methodology, Formal analysis, Investigation, Supervision, Writing. **Soman Elangovan:** Conceptualization, Methodology, Investigation. **Raj Gururajan:** Conceptualization, Supervision, Writing. **Xujuan Zhou:** Conceptualization, Supervision, Writing. **Udyavara Rajendra Acharya:** Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is partially funded by Belmont Private Hospital, Brisbane. We gratefully acknowledge the support from Belmont Private Hospital team members, especially, Ms Mary Williams (CEO), Rachel Stark (Area Manager), Dr Mark Spelman (Psychiatrist), Dr Sean Gills (Psychiatrist), and Dr Tom Moore (Psychiatrist). Without their kind support, this work would not be possible.

References

Abdalkareem, Z. A., Amir, A., Al-Betar, M. A., Ekhan, P., & Hammouri, A. I. (2021). Healthcare scheduling in optimization context: a review. *Health and Technology*, *11*(3), 445–469. http://dx.doi.org/10.1007/s12553-021-00547-5.

Ahmed, Z. H. (2010). Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator. *International Journal of Biometrics & Bioinformatics (IJBB)*, *3*(6), 96.

Aickelin, U., & Dowsland, K. A. (2004). An indirect genetic algorithm for a nurse-scheduling problem. *Computers & Operations Research*, *31*(5), 761–778. http://dx.doi.org/10.1016/s0305-0548(03)00034-0.

Ak, B., & Koc, E. (2012). A guide for genetic algorithm based on parallel machine scheduling and flexible job-shop scheduling. *Procedia - Social and Behavioral Sciences*, *62*, 817–823. http://dx.doi.org/10.1016/j.sbspro.2012.09.138.

Alhanai, T., Ghassemi, M., & Glass, J. (2018). Detecting depression with audio/text sequence modeling of interviews. In *Interspeech 2018, ISCA*. http://dx.doi.org/10.21437/interspeech.2018-2522.

Arora, S. (2009). *Computational complexity a modern approach*. Cambridge: Cambridge University Press.

Australian Beureau of Statistics (2007). National survey of mental health and wellbeing: Summary or results.

Baker, K. R. (2019). *Principles of sequencing and scheduling* (second edition. Edition). Hoboken, NJ, USA: John Wiley & Sons, Inc..

Beaufort, I. N., De Weert-Van Oene, G. H., Buwalda, V. A., de Leeuw, J. R. J., & Goudriaan, A. E. (2017). The depression, anxiety and stress scale (DASS-21) as a screener for depression in substance use disorder inpatients: A pilot study. *European Addiction Research*, *23*(5), 260–268. http://dx.doi.org/10.1159/000485182.

Berlim, M. T., Fleck, M. P., & Turecki, G. (2008). Current trends in the assessment and somatic treatment of resistant/refractory major depression: An overview. *Annals of Medicine*, *40*(2), 149–159. http://dx.doi.org/10.1080/07853890701769728.

Braaksma, A., Kortbeek, N., Post, G., & Nollet, F. (2014). Integral multidisciplinary rehabilitation treatment planning. *Operations Research for Health Care*, *3*(3), 145–159. http://dx.doi.org/10.1016/j.orhc.2014.02.001.

Chien, C.-F., Tseng, F.-P., & Chen, C.-H. (2008). An evolutionary approach to rehabilitation patient scheduling: A case study. *European Journal of Operational Research*, *189*(3), 1234–1253. http://dx.doi.org/10.1016/j.ejor.2007.01.062.

Conelea, C. A., Philip, N. S., Yip, A. G., Barnes, J. L., Niedzwiecki, M. J., Greenberg, B. D., et al. (2017). Transcranial magnetic stimulation for treatment-resistant depression: Naturalistic treatment outcomes for younger versus older patients. *Journal of Affective Disorders*, *217*, 42–47. http://dx.doi.org/10.1016/j.jad.2017.03.063.

Dai, J., Geng, N., & Xie, X. (2021). Dynamic advance scheduling of outpatient appointments in a moving booking window. *European Journal of Operational Research*, *292*(2), 622–632. http://dx.doi.org/10.1016/j.ejor.2020.11.030.

Eiben, A. (2015). *Natural Computing Series, Introduction to evolutionary computing* (2nd ed.). Berlin, Heidelberg: Springer Berlin Heidelberg.

Fitzgerald, P. B., George, M. S., & Pridmore, S. (2021). The evidence is in: Repetitive transcranial magnetic stimulation is an effective, safe and well-tolerated treatment for patients with major depressive disorder. *Australian & New Zealand Journal of Psychiatry*, http://dx.doi.org/10.1177/00048674211043047.

Fitzgerald, P. B., Hoy, K. E., Reynolds, J., Singh, A., Gunewardene, R., Slack, C., et al. (2020). A pragmatic randomized controlled trial exploring the relationship between pulse number and response to repetitive transcranial magnetic stimulation treatment in depression. *Brain Stimulation*, *13*(1), 145–152. http://dx.doi.org/10.1016/j.brs.2019.09.001.

Galambos, G., & Woeginger, G. J. (1993). An on-line scheduling heuristic with better worst-case ratio than graham's list scheduling. *SIAM Journal on Computing*, *22*(2), 349–355. http://dx.doi.org/10.1137/0222026.

Gartner, D., & Kolisch, R. (2014). Scheduling the hospital-wide flow of elective patients. *European Journal of Operational Research*, *233*(3), 89–699. http://dx.doi.org/10.1016/j.ejor.2013.08.026.

Gartner, D., & Padman, R. (2017). Mathematical programming and heuristics for patient scheduling in hospitals. In *Handbook of research on healthcare administration and management, IGI global* (pp. 627–645). http://dx.doi.org/10.4018/978-1-5225-0920-2.ch038.

George, M. S., & Taylor, J. J. (2014). Theoretical basis for transcranial magnetic stimulation. In *A clinical guide to transcranial magnetic stimulation*. Oxford University Press.

Goldberg, D. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading, Mass: Addison-Wesley Publishing Company.

Goldberg, D. E., & Lingle, R. (1985). Alleles, loci, and the traveling salesman problem. In *Proceedings of an international conference on genetic algorithms and their applications, Vol. 154* (pp. 154–159). Hillsdale, NJ: Lawrence Erlbaum.

Golgoun, A. S., & Sepidnam, G. (2018). The optimized algorithm for prioritizing and scheduling of patient appointment at a health center according to the highest rating in waiting queue. *International Journal of Scientific and Technology Research*, *7*, 240–245.

Graham, R. L. (1969). Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, *17*(2), 416–429. http://dx.doi.org/10.1137/0117039.

Griffiths, J., Williams, J., & Wood, R. (2012). Scheduling physiotherapy treatment in an inpatient setting. *Operations Research for Health Care*, *1*(4), 65–72. http://dx.doi.org/10.1016/j.orhc.2012.08.001.

Hamrock, E., Paige, K., Parks, J., Scheulen, J., & Levin, S. (2013). Discrete event simulation for healthcare organizations: A tool for decision making. *Journal of Healthcare Management / American College of Healthcare Executives*, *58*, 110–124. http://dx.doi.org/10.1097/00115514-201303000-00007, discussion 124.

Hovington, C. L., McGirr, A., Lepage, M., & Berlim, M. T. (2013). Repetitive transcranial magnetic stimulation (rTMS) for treating major depression and schizophrenia: a systematic review of recent meta-analyses. *Annals of Medicine*, *45*(4), 308–321. http://dx.doi.org/10.3109/07853890.2013.783993.

Jiang, Y., Abouee-Mehrizi, H., & Diao, Y. (2020). Data-driven analytics to support scheduling of multi-priority multi-class patients with wait time targets. *European Journal of Operational Research*, *281*(3), 597–611. http://dx.doi.org/10.1016/j.ejor.2018.05.017.

Lam, R. (2018). *Depression* (third edn.). Oxford, England: Oxford Psychiatry Library, Oxford University Press.

Lipowski, A., & Lipowska, D. (2012). Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications*, *391*(6), 2193–2196. http://dx.doi.org/10.1016/j.physa.2011.12.004.

Lovibond, S., & Lovibond, P. F. (1995). *Manual for the depression anxiety stress scales* (2nd. edn.). Sydney: Psychology Foundation.

Montgomery, S., & Asberg, M. (1979). A new depression scale designed to be sensitive to change. *British Journal of Psychiatry*, *134*, 382–389.

Papadimitriou, C. H., & Steiglitz, K. (1998). *Combinatorial optimization: Algorithms and complexity*. Courier Corporation.

Petrovic, S., Leung, W., Song, X., & Sundar, S. (2006). Algorithms for radiotherapy treatment booking. In *25th Workshop of the UK planning and scheduling special interest group* (pp. 105–112).

Petrovic, D., Morshed, M., & Petrovic, S. (2011). Multi-objective genetic algorithms for scheduling of radiotherapy treatments for categorised cancer patients. *Expert Systems with Applications*, *38*(6), 6994–7002. http://dx.doi.org/10.1016/j.eswa.2010.12.015.

Podgorelec, V., & Kokol, P. (1997). Genetic algorithm based system for patient scheduling in highly constrained situations. *Journal of Medical Systems*, *21*(6), 417–427.

Razza, L. B., Moffa, A. H., Moreno, M. L., Carvalho, A. F., Padberg, F., Fregni, F., et al. (2018). A systematic review and meta-analysis on placebo response to repetitive transcranial magnetic stimulation for depression trials. *Progress in Neuro-Psychopharmacology and Biological Psychiatry*, *81*, 105–113. http://dx.doi.org/10.1016/j.pnpbp.2017.10.016.

Rivera, G., Cisneros, L., Sánchez-Solís, P., Rangel-Valdez, N., & Rodas-Osollo, J. (2020). Genetic algorithm for scheduling optimization considering heterogeneous containers: A real-world case study. *Axioms*, *9*(1), 27. http://dx.doi.org/10.3390/axioms9010027.

Sauré, A., Begen, M. A., & Patrick, J. (2020). Dynamic multi-priority. *Multi-Class Patient Scheduling with Stochastic Service Times, European Journal of Operational Research*, *280*(1), 254–265. http://dx.doi.org/10.1016/j.ejor.2019.06.040.

Sauré, A., & Puterman, M. L. (2014). The appointment scheduling game. *INFORMS Transactions on Education*, *14*(2), 73–85. http://dx.doi.org/10.1287/ited.2013.0119.

Ting, C.-K., Su, C.-H., & Lee, C.-N. (2010). Multi-parent extension of partially mapped crossover for combinatorial optimization problems. *Expert Systems with Applications*, *37*(3), 1879–1886. http://dx.doi.org/10.1016/j.eswa.2009.07.082.

Williams, H. P. (2013). *Model building in mathematical programming* (5th edn.). Hoboken, N.J: Wiley.

World Health Organisation (2018). Depression. https://www.who.int/news-room/fact-sheets/detail/depression, accessed on 01 2021.

Xu, D., & Yang, D.-L. (2013). Makespan minimization for two parallel machines scheduling with a periodic availability constraint: Mathematical programming model, average-case analysis, and anomalies. *Applied Mathematical Modelling*, *37*(14–15), 7561–7567. http://dx.doi.org/10.1016/j.apm.2013.03.001.

Zang, Z., Wang, W., Song, Y., Lu, L., Li, W., Wang, Y., et al. (2019). Hybrid deep neural network scheduler for job-shop problem based on convolution two-dimensional transformation. *Computational Intelligence and Neuroscience*, *2019*, 1–19. http://dx.doi.org/10.1155/2019/7172842.

Zhao, L., Chien, C.-F., & Gen, M. (2015). A bi-objective genetic algorithm for intelligent rehabilitation scheduling considering therapy precedence constraints. *Journal of Intelligent Manufacturing*, *29*(5), 973–988. http://dx.doi.org/10.1007/s10845-015-1149-y.