SAM: A META-HEURISTIC ALGORITHM FOR SINGLE MACHINE SCHEDULING PROBLEMS

Natasha Zlobinsky[1,2] and Ling Cheng[1]

[1]*School of Electrical and Information Engineering, University of the Witwatersrand, Johannesburg*
[2]*Meraka, Council for Scientific and Industrial Research, Meiring Naude Road, Pretoria*

Abstract: The main contribution of this work is to investigate the hypothesis that the performance of the Simulated Annealing (SA) algorithm can be improved by combining it with other sampling methods in solving the single machine weighted earliness and tardiness scheduling problem. In this paper we present the formulation of our novel hybrid algorithm, SAM, and the main results. The algorithm SAM, which stands for Simulated Annealing with Metropolis-Hastings, is a two-step process. To initialise, the search space of possible feasible schedules is divided into a number of sections. In the first step Metropolis-Hastings sampling is performed over the sections in order to obtain characteristics of a likelihood function over the sections so that a section with a high likelihood of containing the optimal schedule is chosen for step two. In step two SA is run on the pruned search space to find a solution schedule. This relies on a novel way of visualising the search space in a geometric way as a wheel of indices. The results show that low deviation solutions can be obtained in significantly shorter runs with SAM than seen in the literature or required of the basic SA algorithm. We can achieve a 4.5 times reduction in required algorithm run time to achieve a less than 2% deviation from the optimum value. SAM even enables us to find the optimal solution in as few as 1000 iterations of SA in some cases.

Keywords: Simulated Annealing, scheduling, single-machine, earliness-tardiness, multi-objective, MCMC, Metropolis-Hastings, meta-heuristic

1. INTRODUCTION

Scheduling can be described as the allocation of a set of tasks over any sparse resource in order to optimise certain objectives [1]. Numerous applications in a wide variety of fields has resulted in a rich history of research into optimisation for scheduling. Although it is a fundamental form of the problem, the single machine scheduling problem with two or more objectives is known to be NP-hard or NP-complete [2], and so meta-heuristic algorithms such as Genetic Algorithms (GAs), Tabu search and Simulated Annealing (SA) are the accepted solution technique since no optimal polynomial time algorithm exists (unless $P = NP$) [3]. These meta-heuristic algorithms may provide acceptable answers in a much shorter time than exhaustive search or exact methods, but there is still a time cost to pay.

Consider the problem of finding the optimal schedule according to specific objectives in the case of only 10 tasks. The search space has an overwhelming size of over 3.6 million options. Now consider another very modest problem with 100 different-sized tasks. This has of the order of 10^{157} possible solutions. With a history of research mainly centred around the manufacturing industry, traditional scheduling methods may come short of meeting the needs of more modern applications, where results may be more time-sensitive. In wireless communications networks, for instance, tasks such as data packets ought to be cleverly scheduled in time or frequency domains or both in order to maximise data throughput whilst optimising use of spectrum resources in a context where the tolerance for computational delays is very low. The issue of spectrum scarcity has become increasingly contentious and expensive. The algorithms developed in this work if applied to scheduling packets for wireless communications can help to improve spectral efficiency. In this application algorithms must find solutions quickly enough to ensure the user experience is not negatively affected. With such a large search space, more efficient ways to obtain good quality solutions are required. There is great practical necessity and potential benefit if a general way can be found to reduce the solution search space and shorten run times without significantly compromising the quality of solutions, where good solutions are those that are minimally deviant from the optimal value.

In this work we investigate if the performance of Simulated Annealing in solving the single machine weighted earliness-tardiness scheduling problem can be improved by combining the SA algorithm with other sampling methods in a two-step process, the first step being a pre-sampling step to reduce the search space, and the second being to run SA on the reduced search space. One of the most powerful sampling methods available at this time is Markov Chain Monte Carlo methods based on Bayesian inference, which we have chosen for pre-sampling. For our experiments we have generated thirteen problem

instances with different parameters, which we contend are sufficiently representative for the results here to be of more general significance. Performance is evaluated in terms of running time and quality of the solution: on the basis of required total number of iterations, and percentage deviation from the optimum value respectively.

The contributions of this work are:

- to present the first hybrid MH-SA algorithm for use in optimisation of scheduling problems that improves on the efficiency of existing algorithms.
- to investigate the performance of SA in solving the single machine scheduling problem and compare with the new SAM algorithm.
- to present a unique application method of Metropolis-Hastings Monte Carlo to a problem that has not been done in the literature before, in a way that recognises and uses specific properties of the search space in solving this particular problem. It also relies on a new perspective of the search space.

Despite its ability to specialise to the characteristics of the search space, it is also a general method applicable to a wide range of different problems and objectives.
The remainder of the document is structured as follows: Section 2. presents some of the most pertinent work related to the present work, the problem and solution models and approaches are detailed in Section 3., experimental results are presented in Section 4. before concluding in Section 5..

2. REVIEW OF RELATED WORK

2.1 Simulated Annealing

Using SA, Tan and Narasimhan investigate minimising tardiness on a single machine with the addition of sequence-dependent setup times [4]. A rather unique situation is investigated by Jozefowska et al., where each task can be executed in one of several modes, which are divided according to activity resource requirements and duration [5]. They use SA for minimising the makespan with no pre-emption allowed. The researchers also perform pre-processing on the search space to eliminate various options. For 10-job schedules average relative deviation without penalty functions is 0.93% but increasing significantly to 110% maximum, in 60 000 iterations, with performance generally improving for larger problem instances. This trend may suggest that the results presented in Section 4. may also be improved in larger problem instances. Wu at al.'s experiments on SA with learning effect had results ranging from an average percentage deviation of 1% to maximum 1371% and even 3867% for 12-job instances [6]. Such large discrepancies indicate that more research is required on scheduling using the SA algorithm to get more consistent results.

Mahdavi et al. use SA for the single machine scheduling problem to minimise total weighted earliness and tardiness [7]. In their formulation, however, they assume fixed due dates and controllable processing times where job lengths can be expanded or compressed within certain limits. They seek to find an optimal set of expansions and compressions as well as job sequence. Multiple machine single objective scheduling is approached using SA by Kim et al. [8].

An attempt is made to find a general solution to multi-objective scheduling problems with two or more objectives by Loukil et al. [9]. Their method requires that a set of "potentially efficient" points be initialised at the start and a family of weighting functions be defined to direct the statistical search to potentially more efficient solutions. It also involves a filtering procedure where the solutions in potentially efficient sets undergo pairwise comparison to remove solutions unfairly dominated by any one of the objectives. These functions are used in every iteration of the algorithm to update the set of potentially efficient points, adding significant complexity. The choice of weighting functions is both arbitrary and problem-dependent, and can have a significant impact on the solution. The authors also concede that a large number of experiments is necessary to determine the number of sets of weights that give "good" solutions. It is our opinion that this method is unnecessarily complex and contains too many variables that have a large impact on the solution quality and run time, and that there is space for more innovation in multi-objective scheduling algorithms.

Neighbour generation is known to be a strategic part of the implementation of SA having an influence on the performance [10], yet most commonly in the literature the same methods are employed i.e. selecting random positions and performing various shift, swap, and insertion operations; see, for example [4–6, 8, 9, 11]. These methods both involve unnecessary complexity and add to the algorithm run time, necessitating the storage and generation of a number of arrays and performing numerous operations on them many thousands of times. We take an entirely different approach to neighbour generation which significantly reduces the complexity.

Initial temperature is an important parameter to fix and is often determined by setting a starting acceptance ratio and deriving temperature from that [8]. It has been asserted that initial solution is also an important factor in algorithm performance and much of the literature sees the use of priority dispatch rules or other methods to generate a good starting solution [8, 12, 13]. The effect of starting solution on the final solution is investigated by Tan and Narasimhan who find no evidence that "better than randomly selected" initial solutions lead to lower average objective function values in the final solution and prompting the researchers to conclude that a properly tuned SA algorithm can produce good results regardless of the initial solution.

Some researchers design the algorithm such that a large number of solution updates is performed at every temperature value or within a temperature range [4, 10]. This number is yet another variable that is to be determined experimentally. Tan and Narasimhan conclude that it would yield better results to use a smaller temperature decay function than a larger number of updates at each temperature [14]. We rely on this conclusion and choose only to do one update at each temperature and use a log temperature decay function to ensure a gentle reduction.

2.2 Other algorithms

Genetic Algorithms are popular in the scheduling literature although there are few papers on the same optimisation objectives as ours. Hamidinia et al. look at a similar problem of minimising weighted tardiness and earliness without pre-emption but in a batched delivery system and with single due dates [15]. Other Genetic Algorithm approaches have various differences from the problem of this work [16–19]. Wan and Yen investigate the performance of Tabu search methods in solving the weighted earliness-tardiness problem with problem sizes of 15 to 80 jobs [20]. They claim that the method usually obtains deviations within 5% of the optimal value, with an average 2% deviation for 15-job problems. However, the worst case values or variances are not given. Hino and Ronconi also solve the earliness-tardiness problem with Tabu search, including heuristics for finding a suitable starting schedule [21]. They find deviations ranging from 0% to 0.25% from the heuristic benchmark (not the optimal) for 10-job problems, solving 216 of 280 instances optimally. The complexity of this algorithm is $O(n^2)$. Wodecki finds errors below 5% for 40-job earliness-tardiness problems after $2n^2$ iterations, but using another algorithm to find a suitable starting point [22].

Tan et al. find that branch and bound (B&B) would be the preferred solution for smaller problems compared to SA and Genetic Algorithms since it yields an optimum solution in an acceptable time period [14]. Their particular problem considers sequence-dependent setup times in the minimisation of total tardiness. Other examples of branch and bound approaches are [23–26]. Mazdeh et al. compare the performance of a B&B algorithm with a Dynamic Programming (DP) method and show that using B&B gives significantly better efficiency than DP owing to the time complexity of DP [13]. DP for scheduling is also investigated in [27–36].

Neighbourhood search [37, 38], particle swarm [39] and recovering beam search [40] are all examples of alternative techniques seen in the literature. Problems modelled as multiple competing agents are presented by Perez-Gonzalez and Framinan [41] and Mor and Mosheiov [42] who present a polynomial time solution for a specific two-agent problem. The proposed Goal Programming method of Li, Fonseca and Chen promises to ensure a global optimum solution is found but fails to highlight the trade-off with computational complexity [43]. Linear programming solutions that include certain upper bounds are presented by Ng, Cheng and Kovyalov [44] and an analysis of a linear programming heuristic by Potts shows that for the minimisation of maximum completion time on two parallel machines, a linear time algorithm can be used [45].

2.3 Other hybrid or combined algorithms

Yannibelli and Amandi combine SA and an evolutionary algorithm [46]. In their formulation the stage of the evolutionary algorithm and level of diversity of the population changes the behaviour of the SA algorithm. Gupta and Smith tackle the single machine scheduling problem for minimising total tardiness, with the sequence dependent setup times [47] using a greedy randomised adaptive search procedure. Other approximation algorithms [48] and memetic algorithms (which include scatter search) [49, 50] used are claimed to combine the strengths of a hierarchical population approach such as in GA, and the "intensification power" of local search procedures [50] .*

2.4 Metropolis-Hastings Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) methods have arguably been the most influential algorithms of the 20th century, being used in fields as varied as Physics, Engineering, Econometrics, Statistics and Computer Science. Surprisingly, this kind of Monte Carlo does not appear to have been used in scheduling applications at all. Developments in MCMC are ongoing with improvements such as adaptive MCMC [52, 53], Hamiltonian Monte Carlo [55] and various papers on convergence diagnostics [57–60].

3. FORMULATION OF THE MODEL

3.1 The Scheduling Problem

Figure 1 illustrates elements of the general scheduling problem. The single machine scheduling problem is defined by a set of n jobs or tasks $J = (J_1, J_2, ..., J_n)$ to be processed by a single machine. Each job $j \in J$ can be characterised by its processing time (p_j), due window beginning at e_j and ending at d_j, and possible weightings (w_j) which indicate relative importance. The unique weighted earliness of job j is $w'_j E_j$ and the weighted tardiness is $w''_j T_j$. The job's start time is denoted s_j and completion time is C_j. We assume pre-emption is

* Many researchers use mean CPU run times of their algorithms as a performance measure (examples are [6, 7, 15]) but neglect to present the complexity order of their algorithms. This makes comparisons fairly meaningless as myriad factors affect run times, including the platform, other software running on the platform, the language in which the algorithms are coded, and the developer's coding technique. This is why we have opted not to record running times.

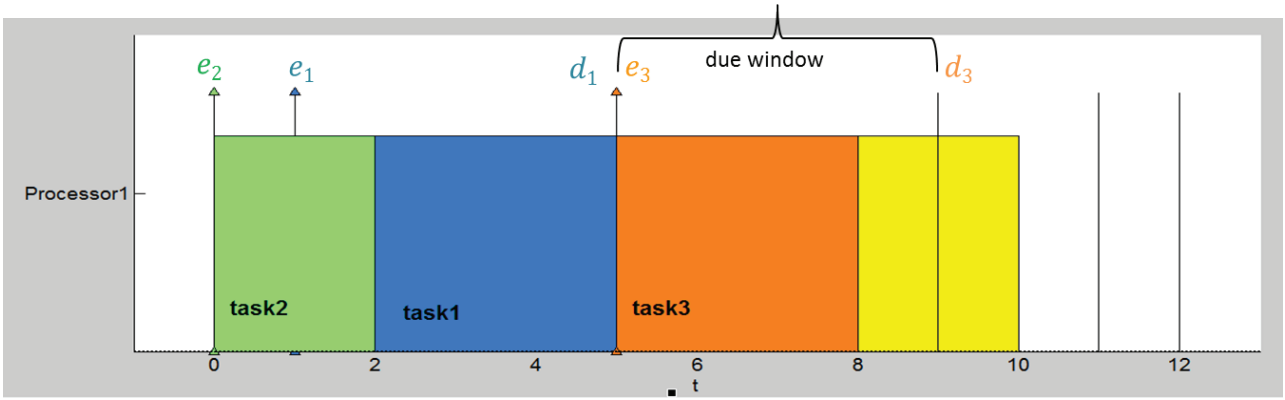


Figure 1: Example representation of a schedule

not allowed, i.e. the job must be completed without interruption, so we can define completion time as

$$C_j = s_j + p_j$$

Earliness (Equation (1)) and tardiness (Equation (2)) are defined for a job with due window as

$$E_j = \max\{e_j - C_j;\, 0\} \tag{1}$$

and

$$T_j = \max\{C_j - d_j;\, 0\} = \max\{L_j;\, 0\} \tag{2}$$

The scheduling objective of our work is Equation (3)

$$\min \gamma = \min \sum_{j \in J} (w'_j E_j + w''_j T_j) \tag{3}$$

and we seek a solution that has a small deviation from the optimum (less than 5%) in an efficient way.

A feasible schedule has no overlapping jobs and no job starting earlier than the schedule start time, i.e condition Equation (4) is met.

$$\{s_i \geq C_j \;\forall\; i > j\} \wedge \{s_i \geq t_0\} \;\forall\; i, j \in J \tag{4}$$

We assume the processing times of jobs are known *a priori* to the scheduling activity and all jobs are available from the start of the scheduling activity, that machine idle time is not allowed and no setup time is considered, pre-emption is not allowed, and there are no precedence relationships between jobs. Therefore the scheduling decision involves only the order in which jobs are to be scheduled.

3.2 *Solution Approach*

The solution approach relies on a novel way of visualising the search space not as $n!$ schedules consisting of permutations of n jobs, but as an array of indices 0 to $n! - 1$ arranged radially similar to the hours of a clock, as shown in Figure 2. The indices are keys to the permutations of the base schedule $\{1, 2, ..., 10\}$ sorted into lexicographical order.

In order to find our proverbial needle in a haystack we introduce a two-step process, outlined as:

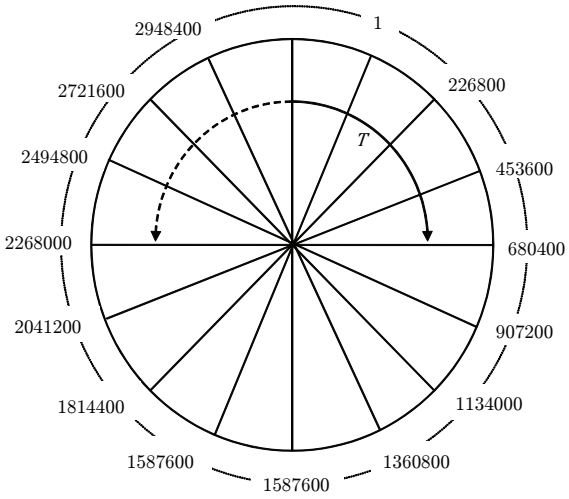


Figure 2: Representation of the search space as schedule indices arranged radially in a similar manner to the hours of a clock

- In the first step the search space of schedule indices is divided into m equal sections. We then perform pre-sampling by Metropolis-Hastings Markov Chain Monte Carlo on the *sections* to obtain a distribution of the likelihood of each section containing an optimal solution.
- In the second step we select the section with the highest apparent likelihood of containing an optimum schedule and perform SA searching for a *schedule* index only within the bounds of the chosen section.

We call this hybrid algorithm SAM (SA with Metropolis-Hastings). Using SAM we are able to reduce the search space by a factor of 20 or even 100 so that SA can find a minimally deviant, or in many cases the optimum solution, significantly faster than when performed on the complete search space. The circular representation of schedule indices enables us to use any sampling procedure that is not bounded by simply continuing around the clock by however many revolutions

are required for values sampled over $n! - 1$.

Pre-sampling: Formally, in the first step each index $i \in \mathbf{I}$ maps to the set of schedules $S(i) \subset \mathbf{S}$ within a section i of size $\frac{|\mathbf{S}|}{m}$ as per Equation (5).

$$i \rightarrow S(i) \; = \; S_x, S_{x+1}, ..., S_{x+[\frac{n!}{m}]-1} \tag{5}$$

Metropolis-Hastings MCMC requires the definition of three functions, a target distribution $p(x)$, a proposal distribution $q(\cdot)$ from which to sample and a sampling distribution. The Markov Chain then evolves by choosing a candidate point from the proposal distribution using the sampling distribution and calculating whether to reject or accept the candidate sample (state) according to the acceptance probability (Equation (6)).

$$\alpha = \frac{p(i^{(t)})q(i^{(t)}|i^{(t-1)})}{p(i^{(t-1)})q(i^{(t-1)}|i^{(t)})} \tag{6}$$

In this implementation the Metropolis-Hastings sampling involves samples drawn from section indices instead of schedules, starting with the uniform prior distribution on the set of section indices $I = \{1, 2, \ldots, m\}$

$$I \sim P(I) : p(i) = \frac{1}{m} \quad \forall \quad i \in I \tag{7}$$

The target distribution is the posterior distribution of I, which is inferred from data gathered about the schedules within the section after every sample drawn from $P(I)$. Every time a section i is sampled, uniform inner sampling is performed within the section and every inner sample adds to the evidence dataset $D_i \in \mathbf{D}$. Once a section has been sampled we use a function to calculate the likelihood $L(\, S(i) \,|\, i \,)$ using D_i. The likelihood is a representation of our belief that the optimal schedule lies within the section i. We have defined the likelihood as Equation (8)).

$$L(S(i)|D_i) = \frac{\left(\min\limits_{s \in D_i} \gamma(S_s)\right)^{-1}}{\sum_{j \in \mathbf{D}} \left(\min\limits_{s \in j} \gamma(S_s)\right)^{-1}} \tag{8}$$

The evidence dataset D_i is drawn from a uniform distribution since no further information is available about the distribution of cost values in each section to justify any other distribution. This assumption of a uniform distribution is maintained throughout the procedure as it is not the goal in this pre-sampling process to define the nature of the function within each section precisely. The posterior (Equation (9)) is updated at every iteration and eventually the actual distribution emerges from the Metropolis-Hastings (MH) procedure after a sufficient number of iterations.

$$P(\mathbf{I}|\mathbf{D}) = \frac{\mathbf{L}(\mathbf{D}|\mathbf{I}) \times \mathbf{P}(\mathbf{I})}{\mathbf{P}(\mathbf{D})} \tag{9}$$

Once the algorithm has suitably converged, the section with the highest likelihood is determined, which forms the bounds of the search space for the second step, i.e. SA.

Reduced search SA: SA is also a form of Monte Carlo and so after choosing a starting point schedule with energy cost E_0 the algorithm progresses by generating a candidate schedule index, calculating the energy cost of the candidate section (E_i) and accepting if the energy cost is lower or, if not, accepts or rejects the sample according to the condition on the Boltzmann distribution (Equation (10)).

$$u < \exp\left(-\Delta E / k T_i\right) \tag{10}$$

where $\Delta E = E_i - E_0$, k is the Boltzmann constant and T_i denotes temperature at iteration i. The term u is a random real number generated from a uniform distribution in the interval $(0, 1)$. This process continues until termination conditions are met.

We generate new candidate schedules in a unique way, based on Figure 2. The temperature T is related to how different the new candidate schedule is from the current. As mentioned, schedule indices are keys to the schedule permutations sorted in lexicographical order. The set of permutations is generated once and stored. Generating a new candidate then requires only a lookup. This eliminates the need to perform shift, insertion or swapping operations every iteration as has been done in all the literature we have seen (see Section 2.).

Two indices that are far apart refer to correspondingly very different schedule permutations whereas indices closer together refer to schedules that are more alike owing to the lexicographical ordering. $[-T; T]$ as indicated by the arrow in Figure 2 gives the bounds of the uniform distribution from which the new candidate schedule is selected so we see that a larger temperature is more likely to generate more different candidate solutions than lower temperatures as the bounds are closer for lower temperatures. The temperature follows a logarithmic relationship relative to the starting temperature T_0, and changes value every iteration i according to Equation (11).

$$T_i = \frac{T_0}{\ln(i)} \tag{11}$$

4. SIMULATION RESULTS

The most common problem instance generation technique used in the literature is that proposed by Potts and Van Wassenhove [45], or similarly by Hall and Posner [61], which is considered the classical approach [6, 26] and is also the method we use. The method is adapted to generate due windows instead of due dates. We use a range of tardiness factor and relative due date values to generate the problems so we contend that the 13 problem instances tested are sufficiently representative of real problems in as far as proving the algorithm and methodology we present are valid and for these performance evaluations to be at least indicative of expected results for any given problem instance.

The experiments compare the performance of our two-step SAM algorithm with basic SA run on the complete search space. The quality of the solution obtained by the tested algorithm is measured by the percentage deviation from the optimal, defined in Equation (12).

$$\%\, dev = \frac{x - x*}{x*} \tag{12}$$

where $x*$ is the absolute optimal solution for the problem set and x is currently obtained solution.

Basic SA with two different starting temperatures ($0.5n!$ and $0.25n!$) was run on all 13 problem sets, 50 times per unique combination of parameters. Average, minimum and maximum values were recorded as well as variance. Experiments were performed on SAM with 20, 50 and 100 section divisions and the results compared against one another and against basic SA. The starting temperature of reduced search SA in SAM was half the size of a section. The starting point for all SA runs was the median point of the search space.

Table 1 shows that shorter runs of our SAM algorithm yield significantly better results than basic SA of the same length. In particular because the variances are much smaller for SAM than basic SA (maximum variance of 29% for SAM vs. maximum variance of 47% for basic SA in 1000 iteration run length), the worst case values of SAM are significantly better. This can also be seen graphically in Figure 3b and is true up to 5000 iteration run lengths. If this algorithm is to be deployed in network managers for scheduling packets in wireless networks the time cost impact is of utmost importance. In this situation our SAM algorithm would be the preferred choice over basic SA since it can yield low deviations and near-optimal values as close as 0.03% deviation and in a large number of cases the optimal value, in a mere 1000 iterations.

However, there is a point in run lengths where the two methods cross over and it is more effective to run basic SA longer than to use SAM with longer runs. We see from Table 1 and Figure 3a that the differences in % deviation results are likely to be significantly better for SAM than basic SA on the lower iteration count spectrum and significantly better for basic SA than SAM at the high iteration counts but in the middle iteration count ranges the performance is not generally statistically significantly different. This point where the difference becomes statistically insignificant (t-test returns false) is generally between 100 000 and 150 000 iterations. If the accuracy of the MH results is good then SAM can provide significant improvements but when a bad suboptimal section is chosen, such as when the algorithm does not escape from a local minimum, the deviation can be quite large – as large as 24% in one of our experiments.

Large deviations after long runs are more prevalent in SAM with 20 and 50 sections where the section and SA search space size is larger than in those with 100 section divisions, as can be seen in both Figure 3a and Figure 3b. However, even in this instance it must be noted that the deviation on average does not exceed 3.5% in the cases where the deviation is larger than that of basic SA. This performance is still acceptable (we define up to 5% deviation as "low"). On the other end of the spectrum even for large sections (20 sections) very small deviations (less than 0.2% deviation) or even the optimal value can be obtained. The optimal value was found in 103 of 130 instances (79% of instances). For different datasets and depending on the sections chosen by MH there is no clear winner between 20, 50 or 100 sections as they each perform differently on the datasets tested but SAM is the clear winner over basic SA when run time is an important factor as is the case in the communications application for which we intend the algorithm.

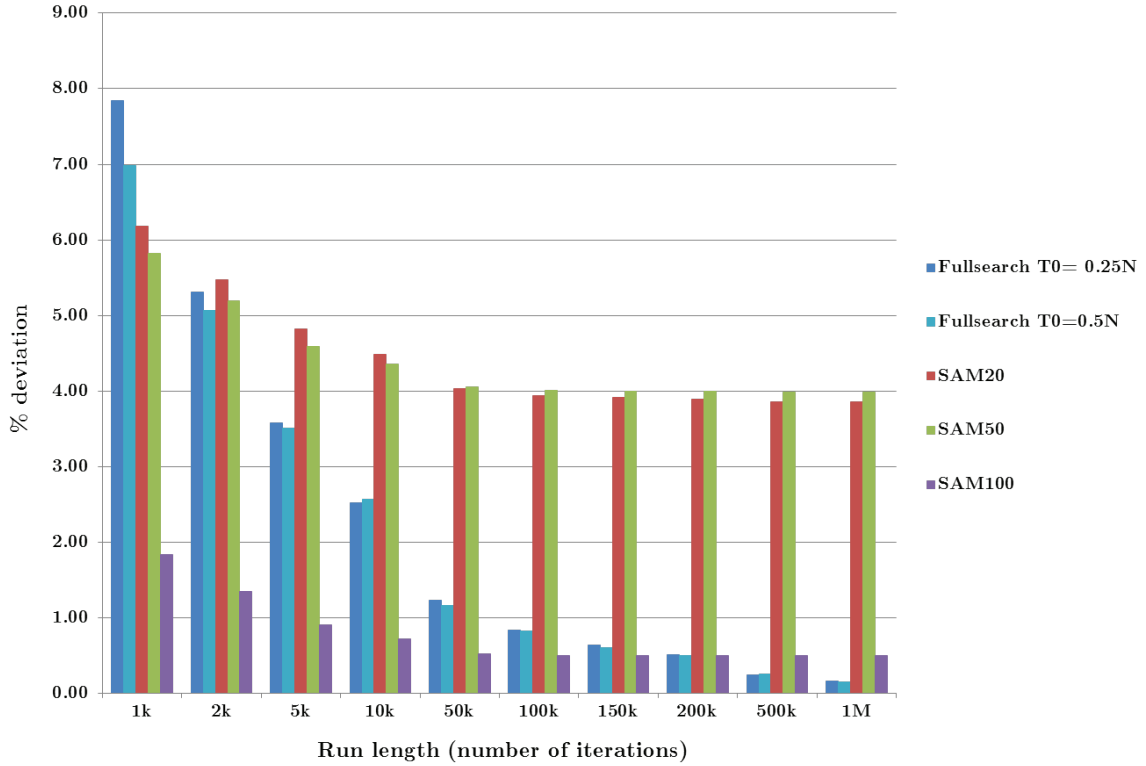
In summary it is observed that if the coarse MH algorithm is run for 1000 iterations, reducing the search space by a factor of 20 and then running SA on the reduced search space for 10 000 iterations, a similar deviation can be achieved as running SA on the full search space for 50 000 iterations (about 5%). The improvement achieved is reducing the run time by a factor of 4.5. While this may seem modest, it must be noted that job sets to be scheduled will typically be far larger than ten as used in our experiments. For 50 jobs, let us assume that the full search will require 1 million iterations. The cost calculation is an unavoidable bottleneck with a complexity $O(n)$, which would need to be calculated 1 million times. The combined algorithm we have presented would reduce this to 222 223, a significantly more realistically acceptable value for practical applications.

5. CONCLUSION

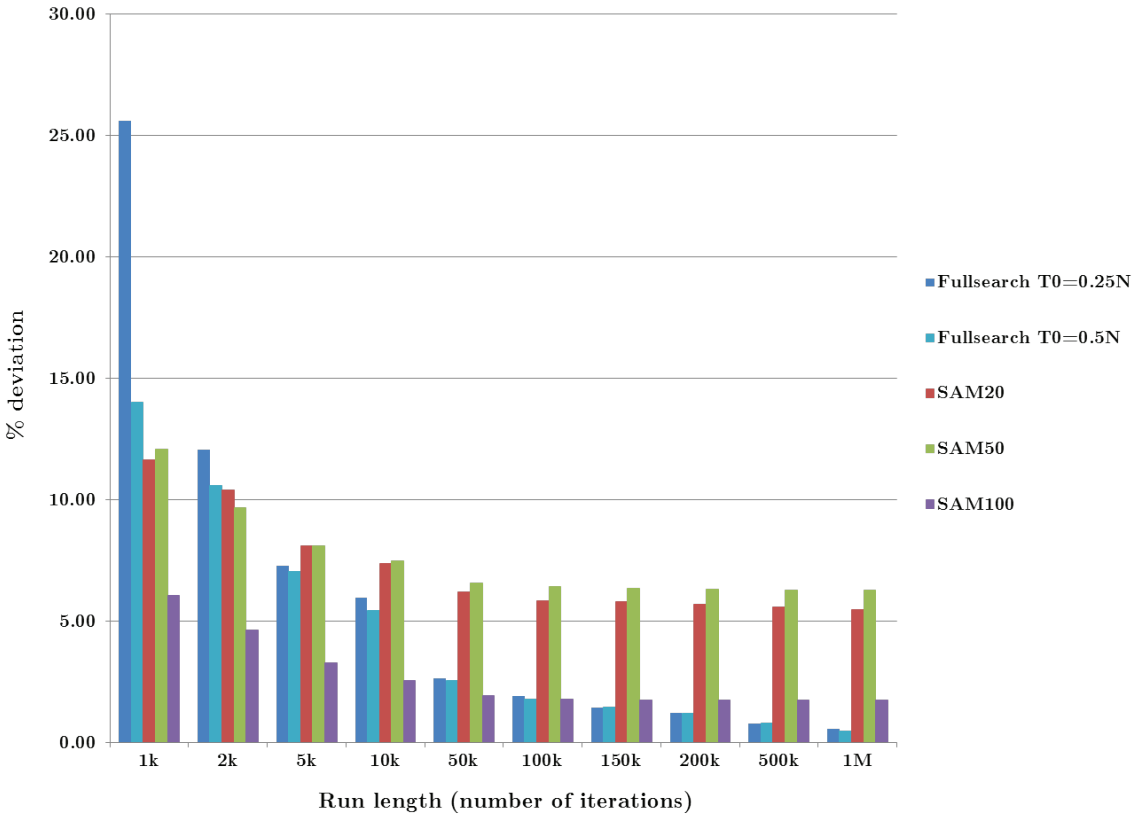
In this paper the results of the investigation into the hypothesis that the performance of Simulated Annealing can be improved by combining it with other sampling methods are presented. The review of current literature has shown that the work has importance and fills part of a gap in the literature of finding more efficient solutions to the single machine weighted earliness-tardiness problem for current applications.

We present the problem and formulate our solution approach called SAM, including a novel implementation of Metropolis-Hastings Monte Carlo to get a coarse but sufficiently accurate distribution over the search space that enables a 20-100 times reduction in the size of the search space, and a unique way of visualising the search space that facilitates a much simplified neighbour generation method.

While worst case computational complexity is not lower for SAM than SA if the same neighbour generation method



(a) Average values



(b) Maximum values

Figure 3: Histogram of the results of SAM, taken over all problem sets, shown for SAM with 20 MH sections (SAM20), 50 sections and 100 sections (SAM50 and SAM100). The results of basic SA for two different starting temperatures are also shown for ease of comparison.

Table 1: Average percentage deviation of SAM results compared with basic SA

	Basic SA T1		**Basic SA T2**		**SAM20**		**SAM50**		**SAM100**	
N	***Avg %***	***Var***	***Avg %***	***Var***	***Avg %***	***Var***	***Avg %***	***Var***	***Avg %***	***Var***
1k	7.84	168.78	6.98	47.72	6.19	19.74	5.82	29.03	1.83	9.89
2k	5.31	39.80	5.06	33.42	5.48	16.94	5.20	17.43	1.34	5.60
5k	3.57	13.53	3.51	12.01	4.82	8.33	4.59	12.07	0.91	2.87
10k	2.52	9.77	2.57	7.87	4.48	6.24	4.36	11.13	0.71	1.92
50k	1.23	2.00	1.16	2.00	4.03	4.21	4.06	10.12	0.52	1.25
100k	0.83	1.14	0.82	0.96	3.94	3.78	4.01	9.84	0.50	1.12
150k	0.64	0.64	0.60	0.70	3.91	3.70	4.00	9.88	0.50	1.10
200k	0.51	0.48	0.49	0.43	3.89	3.65	3.99	9.81	0.50	1.08
500k	0.25	0.21	0.25	0.26	3.86	3.46	3.99	9.80	0.50	1.08
1M	0.16	0.16	0.14	0.12	3.85	3.39	3.99	9.80	0.50	1.08

is used, the actual required run time is lower. We are able to find low-deviation solutions in much shorter runs than seen in the literature (as highlighted in Section 2.) or required of the basic SA algorithm, as shown by our results. We estimate a conservative 4.5 time reduction in required algorithm run time can be achieved. SAM enables us to find the optimal solution in as few as 1000 iterations. Our neighbour generation method also displays significantly reduced complexity. We conclude that benefit can indeed be derived in obtaining faster solutions using SAM than basic SA and possibly other algorithms. An extension of this work using larger problem instances with 100 or more jobs is recommended.

REFERENCES

[1] R.L. L. Graham, E.L. L. Lawler, J.K. K. Lenstra, and A.H.G. H G Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5(C):287–326, 1979. ISSN 01675060. doi: 10.1016/S0167-5060(08)70356-X.

[2] Michael L. Pinedo. *Scheduling Theory, Algorithms, and Systems*. Springer Science+Business MediA, New York, USA, third edition, 2008. ISBN 9780387789347.

[3] M.R. Garey and D.S. Johnson. *Computers and intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and company, New York, 1979. ISBN 0716710447.

[4] K.C. Tan and R. Narasimhan. Minimizing tardiness on a single processor with sequence-dependent setup times: a simulated annealing approach. *Omega, International Journal of Management Science*, 25(6): 619–634, 1997. ISSN 03050483. doi: 10.1016/S0305-0483(97)00024-8.

[5] Jan Weglarz Joanna Jozefowska, Marek Mika, Rafal Rozycki, Grzegorz Waligora. Simulated Annealing for Multi-Mode Resource-Constrained Project Scheduling. *Annals of Operations Research*, 102:137–155, 2001.

[6] Chin Chia Wu, Hung Ming Chen, Shuenn Ren Cheng, Chou Jung Hsu, and Wen Hung Wu. Simulated annealing approach for the single-machine total late work scheduling problem with a position-based learning. *2011 IEEE 18th International Conference on Industrial Engineering and Engineering Management, IE and EM 2011*, pages 839–843, 2011. doi: 10.1109/IEEM.2011.6035289.

[7] Iraj Mahdavi, Vahid Kayvanfar, and G. M. Komaki. Minimizing total tardiness and earliness problem with controllable processing times using an effective heuristic. *40th International Conference on Computers and Industrial Engineering: Soft Computing Techniques for Advanced Manufacturing and Service Systems, CIE40 2010*, 2010. doi: 10.1109/ICCIE.2010.5668166.

[8] Dong-Won Kim, Kyong-Hee Kim, Wooseung Jang, and F Frank Chen. Unrelated parallel machine scheduling with setup times using simulated annealing. *Robotics and Computer-Integrated Manufacturing*, 18:223–231, 2002. ISSN 07365845. doi: 10.1016/S0736-5845(02)00013-3.

[9] T. Loukil, J. Teghem, and D. Tuyttens. Solving multi-objective production scheduling problems using metaheuristics. *European Journal of Operational Research*, 161:42–61, 2005. ISSN 03772217. doi: 10.1016/j.ejor.2003.08.029.

[10] João Tomé Saraiva, Marcelo Leandro Pereira, Virgílio Torrado Mendes, and José Carlos Sousa. A Simulated Annealing based approach to solve the

generator maintenance scheduling problem. *Electric Power Systems Research*, 81:1283–1291, 2011. doi: 10.1016/j.epsr.2011.01.013. URL www.elsevier.com/locate/epsr.

[11] Sergio Fichera, Fulvio Cappadonna, Antonio Costa, and Alberto Fichera. A Simulated Annealing Algorithm for Single Machine Scheduling Problem with Release Dates, Learning and Deteriorating Effects. *Proceedings of the World ...*, I:6–8, 2013. ISSN 20780958. URL http://www.iaeng.org/publication/WCE2013/WCE2013{_}pp584-586.pdf.

[12] W. H M Raaymakers and J. a. Hoogeveen. Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing. *European Journal of Operational Research*, 126:131–151, 2000. ISSN 03772217. doi: 10.1016/S0377-2217(99)00285-4.

[13] Mohammad Mahdavi Mazdeh, Mansoor Sarhadi, and Khalil S. Hindi. A branch-and-bound algorithm for single-machine scheduling with batch delivery minimizing flow times and delivery costs. *European Journal of Operational Research*, 183:74–86, 2007. ISSN 03772217. doi: 10.1016/j.ejor.2006.09.087.

[14] K Tan. A comparison of four methods for minimizing total tardiness on a single processor with sequence dependent setup times. *Omega*, 28:313–326, 2000. ISSN 03050483. doi: 10.1016/S0305-0483(99)00050-X.

[15] Amir Hamidinia, Sahand Khakabimamaghani, Mohammad Mahdavi Mazdeh, and Mostafa Jafari. A genetic algorithm for minimizing total tardiness/earliness of weighted jobs in a batched delivery system. *Computers and Industrial Engineering*, 62(1):29–38, 2012. ISSN 03608352. doi: 10.1016/j.cie.2011.08.014. URL http://dx.doi.org/10.1016/j.cie.2011.08.014.

[16] Amir Ebrahimy Zade and Mohammad Bagher Fakhrzad. A Dynamic Genetic Algorithm for Solving a Single Machine Scheduling Problem with Periodic Maintenance. *ISRN Industrial Engineering*, 2013, 2013.

[17] Mahesh C. Gupta, Yash P. Gupta, and Anup Kumar. Minimizing flow time variance in a single machine system using genetic algorithms. *European Journal of Operational Research*, 70:289–303, 1993. ISSN 03772217. doi: 10.1016/0377-2217(93)90240-N.

[18] Murat Köksalan and Ahmet Burak Keha. Using genetic algorithms for single-machine bicriteria scheduling problems. *European Journal of Operational Research*, 145(3):543–556, 2003. ISSN 03772217. doi: 10.1016/S0377-2217(02)00220-5. URL http://www.sciencedirect.com/science/article/pii/S0377221702002205.

[19] Fariborz Jolai, M. Rabbani, S. Amalnick, a. Dabaghi, M. Dehghan, and M. Yazadn Parast. Genetic algorithm for bi-criteria single machine scheduling problem of minimizing maximum earliness and number of tardy jobs. *Applied Mathematics and Computation*, 194:552–560, 2007. ISSN 00963003. doi: 10.1016/j.amc.2007.04.063.

[20] Guohua Wan and B. P C Yen. Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties, 2002. ISSN 03772217.

[21] Celso M. Hino, Débora P. Ronconi, and André B. Mendes. Minimizing earliness and tardiness penalties in a single-machine problem with a common due date. *European Journal of Operational Research*, 160:190–201, 2005. ISSN 03772217. doi: 10.1016/j.ejor.2004.03.006.

[22] Mieczysław Wodecki. A block approach to earliness-tardiness scheduling problems. *International Journal of Advanced Manufacturing Technology*, 40(7-8):797–807, 2009. ISSN 02683768. doi: 10.1007/s00170-008-1395-7.

[23] Adam Janiak, Władysław a Janiak, Tomasz Krysiak, and Tomasz Kwiatkowski. A survey on scheduling problems with due windows. *European Journal of Operational Research*, 242:347–357, 2015. doi: 10.1016/j.ejor.2014.09.043.

[24] Candace A. Yano and Y. D. Kim. Algorithms for single machine scheduling problems minimizing tardiness and earliness. Technical report, 1986.

[25] Bora P Ronconi and Rcio S Kawamura. The single machine earliness and tardiness scheduling problem : lower bounds and a branch-and-bound algorithm *. *Computational & applied mathematics*, 29:107–124, 2010. ISSN 01018205. doi: 10.1590/S1807-03022010000200002.

[26] Guohua Wan and Benjamin P.-C. C Yen. Single machine scheduling to minimize total weighted earliness subject to minimal number of tardy jobs. *European Journal of Operational Research*, 195(1):89–97, may 2009. ISSN 03772217. doi: 10.1016/j.ejor.2008.01.029. URL http://www.sciencedirect.com/science/article/pii/S0377221708001628http://dx.doi.org/10.1016/j.ejor.2008.01.029.

[27] Candace Arai Yano and Yeong-dae Kim. Algorithms for a class of single-machine weighted tardiness and earliness problems. *European Journal of Operational Research*, 52:167–178, 1991.

[28] Suresh Chand, Hans Schneeberger, and West Lafayette. Theory and Methodology Single machine scheduling to minimize weighted earliness subject to no tardy jobs. *European Journal of Operational Research*, 34:221–230, 1988.

[29] Wooseung Jang. Dynamic scheduling of stochastic jobs on a single machine. *European Journal of Operational Research*, 138:518–530, 2002. ISSN 03772217. doi: 10.1016/S0377-2217(01)00174-6.

[30] Toshihide Ibaraki and Yuichi Nakamura. A dynamic programming method for single machine scheduling. *European Journal of Operational Research*, 76: 72–82, 1994. ISSN 03772217. doi: 10.1016/0377-2217(94)90007-8.

[31] Valery S. Gordon and Vitaly a. Strusevich. Single machine scheduling and due date assignment with positionally dependent processing times. *European Journal of Operational Research*, 198(1):57–62, 2009. ISSN 03772217. doi: 10.1016/j.ejor.2008.07.044. URL http://dx.doi.org/10.1016/j.ejor.2008.07.044.

[32] Min Ji and T.C.E. Cheng. Batch scheduling of simple linear deteriorating jobs on a single machine to minimize makespan. *European Journal of Operational Research*, 202(1):90–98, apr 2010. ISSN 03772217. doi: 10.1016/j.ejor.2009.05.021. URL http://www.sciencedirect.com/science/article/pii/S0377221709003518.

[33] Francis Sourd. Optimal timing of a sequence of tasks with general completion costs. *European Journal of Operational Research*, 165:82–96, 2005. ISSN 03772217. doi: 10.1016/j.ejor.2004.01.025.

[34] W. K. Yeung, Ceyda Oguz, and T. C Edwin Cheng. Single-machine scheduling with a common due window. *Computers and Operations Research*, 28: 157–175, 2000. ISSN 03050548. doi: 10.1016/S0305-0548(99)00097-0.

[35] Shunji Tanaka. An Exact Algorithm for the Single-Machine Earliness Tardiness Scheduling Problem. pages 21–41, 2012. doi: 10.1007/978-1-4614-1123-9.

[36] Dvir Shabtay. The single machine serial batch scheduling problem with rejection to minimize total completion time and total rejection cost. *European Journal of Operational Research*, 233(1):64–74, 2014. ISSN 03772217. doi: 10.1016/j.ejor.2013.08.013. URL http://dx.doi.org/10.1016/j.ejor.2013.08.013.

[37] Marcelo Ferreira Rego, Marcone Jamilson, Freitas Souza, Marcone Jamilson Freitas Souza, and José Elias Claudio Arroyo. Multi-objective algorithms for the single machine scheduling problem with sequence-dependent family setups. *Proceedings - International Conference of the Chilean Computer Science Society, SCCC*, pages 142–151, 2013. ISSN 15224902. doi: 10.1109/SCCC.2012.24.

[38] Zhao Ruiguo Zhao Ruiguo and Li Jiejia Li Jiejia. Neighborhood search algorithm for one-machine scheduling problem with time lags. *2009 Chinese Control and Decision Conference*, pages 1937–1940, 2009. doi: 10.1109/CCDC.2009.5191609.

[39] Davide Anghinolfi and Massimo Paolucci. A new discrete particle swarm optimization approach for the single-machine total weighted tardiness scheduling problem with sequence-dependent setup times. *European Journal of Operational Research*, 193:73–85, 2009. ISSN 03772217. doi: 10.1016/j.ejor.2007.10.044.

[40] B. Esteve, C. Aubijoux, A. Chartier, V T Õ, and V. T'kindt. A recovering beam search algorithm for the single machine Just-in-Time scheduling problem. *European Journal of Operational Research*, 172(3): 798–813, aug 2006. ISSN 03772217. doi: 10.1016/j.ejor.2004.11.014. URL http://www.sciencedirect.com/science/article/pii/S0377221704008574.

[41] Paz Perez-Gonzalez and Jose M. Framinan. A common framework and taxonomy for multicriteria scheduling problems with interfering and competing jobs: Multi-agent scheduling problems. *European Journal of Operational Research*, 235(1):1–16, 2014. ISSN 03772217. doi: 10.1016/j.ejor.2013.09.017. URL http://dx.doi.org/10.1016/j.ejor.2013.09.017.

[42] Baruch Mor and Gur Mosheiov. Single machine batch scheduling with two competing agents to minimize total flowtime. *European Journal of Operational Research*, 215(3):524–531, 2011. ISSN 03772217. doi: 10.1016/j.ejor.2011.06.037. URL http://dx.doi.org/10.1016/j.ejor.2011.06.037.

[43] Lei Li, Daniel J. Fonseca, and Der S. Chen. Earliness-tardiness production planning for just-in-time manufacturing: A unifying approach by goal programming. *European Journal of Operational Research*, 175:508–515, 2006. ISSN 03772217. doi: 10.1016/j.ejor.2005.06.009.

[44] C T Daniel Ng, T C Edwin Cheng, and Mikhail Y Kovalyov. Single machine batch scheduling with jointly compressible setup and processing times. 153: 211–219, 2004. doi: 10.1016/S0377-2217(02)00732-4.

[45] Chris N Potts and Luk N Van Wassenhove. A Branch and Bound Algorithm for the Total Weighted Tardiness Problem. *Operations Research*, 33(2): 363–377, 1985.

[46] Virginia Yannibelli and Analía Amandi. Expert Systems with Applications Hybridizing a multi-objective simulated annealing algorithm with a multi-objective evolutionary algorithm to solve a multi-objective project scheduling problem. *Expert Systems*

With Applications, 40(7):2421–2434, 2013. ISSN 0957-4174. doi: 10.1016/j.eswa.2012.10.058. URL http://dx.doi.org/10.1016/j.eswa.2012.10.058.

[47] Skylab R. Gupta and Jeffrey S. Smith. Algorithms for single machine total tardiness scheduling with sequence dependent setups. *European Journal of Operational Research*, 175(2):722–739, dec 2006. ISSN 03772217. doi: 10.1016/j.ejor.2005.05.018. URL http://www.sciencedirect.com/science/article/pii/S0377221705005060.

[48] Stavros G Kolliopoulos and George Steiner. Approximation algorithms for scheduling problems with a modified total weighted tardiness objective. *Operations Research Letters*, 35(5):685–692, sep 2007. ISSN 0167-6377. doi: http://dx.doi.org/10.1016/j.orl.2006.12.002. URL http://www.sciencedirect.com/science/article/pii/S0167637706001337.

[49] J. Talebi, H. Badri, F. Ghaderi, and E. Khosravian. An efficient scatter search algorithm for minimizing earliness and tardiness penalties in a single-machine scheduling problem with a common due date. *2009 IEEE Congress on Evolutionary Computation, CEC 2009*, pages 1012–1018, 2009. doi: 10.1109/CEC.2009.4983056.

[50] Paulo M França, Alexandre Mendes, Pablo Moscato, Paulo M Franc, Alexandre Mendes, and Pablo Moscato. A memetic algorithm for the total tardiness single machine scheduling problem. *European Journal of Operational Research*, 132(1):224–242, jul 2001. ISSN 03772217. doi: 10.1016/S0377-2217(00)00140-5. URL http://www.sciencedirect.com/science/article/pii/S0377221700001405.

[51] Gareth O. Roberts and Jeffrey S. Rosenthal. Examples of Adaptive MCMC. Technical Report 0610, University of Toronto Department of Statistics, 2008.

[52] David Luengo and Luca Martino. Fully Adaptive Gaussian Mixture Metropolis-Hastings Algorithm. *arXiv preprint arXiv:1212.0122*, (Mcmc):1–10, 2012. URL http://arxiv.org/abs/1212.0122.

[53] Nimalan Mahendran, Z Wang, F Hamze, and N. De Freitas. Adaptive MCMC with Bayesian Optimization. *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 9:152, 2010.

[54] Heikki Haario, Eero Saksman, and Johanna Tamminen. Adaptive proposal distribution for random walk Metropolis algorithm. *Computational Statistics*, 14 (3):375, 1999. ISSN 09434062. doi: 10.1007/s001800050022.

[55] Andrei Kramer, Ben Calderhead, and Nicole Radde. Hamiltonian Monte Carlo methods for efficient parameter estimation in steady state dynamical systems. *BMC bioinformatics*, 15(1):253, 2014. ISSN 1471-2105. doi: 10.1186/1471-2105-15-253. URL http://www.biomedcentral.com/1471-2105/15/253.

[56] Peter J Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, 1995. ISSN 0006-3444. doi: 10.1093/biomet/82.4.711.

[57] G. O. Roberts, a. Gelman, and W. R. Gilks. Weak convergence and optimal scaling of random walk Metropolis algorithms, 1997. ISSN 10505164.

[58] Stephen P Brooks and Gareth O. Roberts. Assessing Convergence of Markov Chain Monte Carlo Algorithms. *Statistics and Computing*, 8:319–335, 1997. ISSN 0717-6163. doi: 10.1007/s13398-014-0173-7.2.

[59] M Jerrum and A. Sinclair. The Markov chain Monte Carlo method: an approach to approximate counting and integration. *Approximation algorithms for NP-hard problems*, pages 482–520, 1997. doi: 10.1109/GLOCOM.2004.1377963.

[60] M K Cowles and B P Carlin. Markov chain Monte Carlo convergence diagnostics: a comparative review. *J. Amer. Stat. Assoc.*, 91(434):883–904, 1996.

[61] Nicholas G Hall and Marc E. Posner. Generating Experimental Data for Computational Testing with Machine Scheduling Applications. *Operations Research*, 49(6):854–865, 2001.