Optimisation Algorithms for Parallel Machine Scheduling Problems with Setup Times

Fabian Kittel
Technische Hochschule Mittelhessen
Friedberg, Germany
Fabian.Kittel@mnd.thm.de

Jannik Enenkel
Technische Hochschule Mittelhessen
Friedberg, Germany
Jannik.Enenkel@mnd.thm.de

Jana Holznigenkemper
Philipps-Universtät Marburg
Marburg, Germany
holznigenkemper@mathematik.uni-marburg.de

Neil Urquhart
Edinburgh Napier University
Edinburgh, United Kingdom
n.urquhart@napier.ac.uk

Michael Guckert
Technische Hochschule Mittelhessen
Friedberg, Germany
Michael.Guckert@mnd.thm.de

CCS CONCEPTS

• **Theory of computation** → **Scheduling algorithms**; • **Applied computing** → *Industry and manufacturing*; • **Computing methodologies** → *Planning and scheduling*; • **Mathematics of computing** → Probabilistic algorithms;

KEYWORDS

Best Response Dynamics, Parallel Machine Scheduling with Setup Times, Heuristics

ACM Reference Format:
Fabian Kittel, Jannik Enenkel, Jana Holznigenkemper, Neil Urquhart, and Michael Guckert. 2021. Optimisation Algorithms for Parallel Machine Scheduling Problems with Setup Times. In *Proceedings of the Genetic and Evolutionary Computation Conference 2021 (GECCO '21).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The parallel machines scheduling problem in which jobs have to be assigned to machines at particular times is one of the oldest problems of operations research with high practical relevance for the manufacturing industry. Minimising the time it takes for a product to pass through manufacturing not only affects the revenue of the company but also improves customer satisfaction [8]. The objective of this well known problem, is to assign jobs to machines at particular times, such that a schedule is created which completes all jobs if possible on time or with minimal deviation from due dates and in full.

In this work, we address a variant of the parallel machine scheduling in which the goal is to find an optimal machine schedule minimising an unweighted combination of earliness, tardiness and setup times that are aggregated in a single objective function (AOF). Tardiness is obviously a crucial aspect of planning as it leads to delayed delivery intrinsic in the plan, earliness is also a cause of customer dissatisfaction, especially in times of just-in-time production. Additionally, setup times indicate considerable effort in setting up machines which is not only time consuming but also likely to cause considerable costs. We intentionally left out weighting factors in that sum considering the criteria as of equal importance. This can be changed according to specific requirements.

In literature, production planning problems are categorised by multiple criteria [1] such as the number of involved machines, equality of machines, non-preemptive jobs or the relevance of setup times. Most research in this area focuses on the minimisation of the makespan, which is the maximum completion time of all jobs [7, 9]. However, tardiness and earliness optimisation are covered by some papers as well [2, 3, 10, 18]. Setup times are often neglected in academic research [1], but the sequence of jobs may have a huge impact on how a production schedule is developed [18]. This is why this work deals with a non-identical parallel machine environment regarding sequence dependant setup times, earliness and tardiness.

As the described problem is known to be NP-hard [16], we face a typical case for applying meta-heuristics such as Evolutionary Algorithms (EAs). This has successfully been done in the past [18], we revisit this technique and discuss an EA implementation minimising for our problem setting. Local-search algorithms also have been applied and provided good results in solving scheduling problems (e.g. [17]). We extend this strand with a new variant inspired by game theory including a non deterministic stochastic element. We call this approach *p-Best Response Dynamic* (p-BRD).

We compare both algorithms in terms of solution quality and runtime by applying them to a series of test sets.

2 P-BRD

Best Response Dynamics is an iterative algorithm for searching Nash equilibria in n player non-cooperative games. Agents successively change their strategy to be a best response to the strategies of at least a subset of the other agents [15]. A combination of strategies for which none of the agents can be better off by a unilateral change of its strategy is called a Nash equilibrium [13].

Potential games are a subset of strategic games for which the effect of a change of strategy can be expressed by a single global

GECCO '21, July 10–14, 2021, Lille, France

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

function [12]. For potential games, best response dynamics always converges to a Nash equilibrium [15].

We view the machine schedule optimisation problem as a game called the *scheduling game*. Payoff is interpreted as costs and agents therefore seek to minimise their utility. Jobs are represented as participating agents in the scheduling game and each agent tries to get an optimal position on a machine. The position directly contributes to the value of AOF. This function represents the utility for each agent and therefore our game is obviously a potential game.

Agents interact in the search for an optimal solution by mutually changing their positions. The procedure terminates in a Nash equilibrium, which embodies the final production plan that is returned by the algorithm. By using a stochastic element that excepts intermediate results even when its counter indicated by the objective function the search process can be considered to consist of an exploration and an exploitation phases.

Input to BRD is a set of jobs to be scheduled together with available machines and a setup matrix. Each job agent J_i has full access to information about the material it produces, its due date and the machines $\mathcal{M}_i$ it can be executed on together with the production time for each machine and their current start and end time. In an initialisation phase job agents are first sorted in ascending order by due date and maximum production times and then get assigned to an initial position on a suitable machine.

Through this initialisation phase, doubly linked lists l_M of job agents are constructed for each machine $M \in \mathcal{M}$. At first, artificial *INIT* agents are placed on the machines that become head of the lists. They do not participate in the scheduling game, their end defines the first availability of the machine defining the start of the planning horizon. Start and end date of jobs placed in the list define time spans in which the machine is available. For each job agent and each machine it can be processed on gaps in the linked list sufficiently large for production and setup time are located. The job agent chooses the gap in which it can be placed as close to its due date as possible. Therefore, each job J_j seeks the predecessor J_k for which $pred(J_k, J_j)$ defined as the value $|c_j - due_j|$ when J_j is inserted as successor of J_k is minimal while respecting setup times and shifting j as close to is due date as possible.

p-BRD now starts with the same list of jobs as the initialisation phase with each job having an initial position on a machine. Each job agent a has a set of options to change its strategy, i.e. its position in the schedule in each round. It can either swap its position with another agent a' if machines match (they are compatible (see Algorithm 1) and production times allow the change without interfering with start and end dates of other agents or - in case that a' cannot run on the machine a is assigned to - change its position by moving behind a' if a' has no successor. A swap is performed if it improves the AOF. However, rejecting options too early entails the threat to be stuck in a local minimum too early. Therefore, non improving swaps increasing AOF are accepted with probability p. This probability decays by multiplying it with a damping factor d in each iteration thus successively moving the process from exploration to exploitation. BRD stops as soon as no more swaps occur and the game has reached a Nash equilibrium. Details of handling setup times determined by the products of the jobs are straightforward and left out for the sake of brevity. The best solution found during exploration is always preserved, so in case an inferior solution is accepted due to p, the best solution found is still accessible and will be returned at termination.

Algorithm 1: p-BRD

```
Data: J list of job agents each with initial positions on
      machine; probability p; damping factor d
Result: seq optimised assignment of jobs
repeat
    for J_j in J do
        for J_k in J \ {J_j} do
            if machines of J_j and J_k are compatible then
                if (swap(J_j, J_k) improves AOF) or (rand<p)
                 then
                    swap(J_j, J_k);
                    break;
                end
            end
            else if J_j can run on machine of J_k and J_k has no
             successor then
                if moving J_j after J_k improves AOF then
                    move J_j after J_k;
                end
            end
        end
    end
    p = p · d ;
until no more swaps;
```

In this implementation agents take the first opportunity that improves AOF without checking further alternatives. Figure 1 illustrates the mechanism of swapping. Let J_3 be the agent currently in action. J_3 could swap with J_2 (on the same machine) or - if machines allow for it - with J_1. Alternative option is placing it behind J_1. Note that $INITAgent1$ and $INITAgent2$ are only available as predecessors if they are alone on their respective machine.

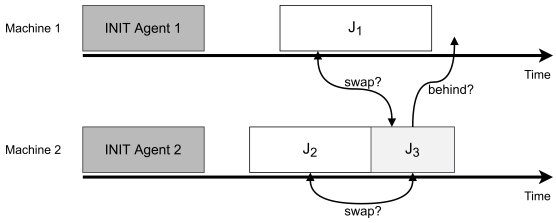

Figure 1: p-BRD search process

3 EVOLUTIONARY ALGORITHM

A vanilla Evolutionary Algorithm using the typical operators to evolve the population – e.g. tournament selection, single point crossover, swap mutation, and tournament replacement – is compared against p-BRD. The fitness of an individual is determined by its phenotype, i.e. AOF of the production schedule that is generated

by the decoder. The decoding of the GA is adapted according to the given problem.

4 DATA SETS

For the first test case we create three data sets to imitate real world production setups. Our method for creating test cases is derived from the approach proposed by Bagheri and Zandieh (2011) [4], Vilcot and Billaut (2008) [19] and Demirkol et al. (1998) [6]. The structure of a test set can be controlled by a number of input parameters that generate a diversity analogous to problems encountered in real world data.

We generate three test sets representing different scenarios. Data set URGENCY contains a considerable amount of jobs but only a few machines. In addition, the jobs are due early provoking a high delay and a high importance of setup times. In contrast Data set SPREAD contains jobs that vary strongly in their due dates and aims for an optimal sequence according to the deviation of the jobs from their due dates. Data set LARGE focuses on the scalability of the algorithms in terms of runtime using a large number of jobs. The datasets can be obtained URL.

For the second test case the published Oliver 30 benchmark for the travelling salesmen problem (TSP) [14] is reformulated as a scheduling problem for which solutions can be assessed by AOF. The TSP is defined for 30 locations with the distance for the shortest round trip (i.e. a tour visiting each location once, starting and ending at the same location) being 419 [11].

5 RESULTS

Evolutionary Algorithms and p-BRD depend on parameter settings that have a strong influence on the performance. Therefore, we applied Tree-structured Parzen Estimators [5] – which is a form of Bayesian optimisation – to fine tune the parameter settings. The obtained parameters can be found URL.

The p-BRD algorithm as such produced poorer results on the OLIVER30 data set (see Figure 2) using AOF. This is explained by the fact that the algorithm aims to minimise AOF instead of the tour length which not only includes setup times in the sequence of jobs but also the setup time between the last and the first job. However, the algorithm can easily be adapted to this effect by adjusting the objective function. The object oriented implementation allows for this with a simple subclass. With the modified objective function p-BRD nearly always reached a tour length near to 424 with a mean value of 424.39 and a standard deviation of 0.39. The Evolutionary Algorithm scatters rather widely between 423 and 565, while the mean is significantly larger than that of p-BRD (see Table 1)

For data set LARGE (see Figure 3), p-BRD performs better than the GA with a smaller standard deviation (see Table 1). Potentially the Evolutionary Algorithm has not yet converged ultimately after the defined number of generations for the large scheduling problem. However, increasing the number of maximal iterations might improve its performance but would further deteriorate runtime.

On the data set URGENCY the differences are clearly less pronounced in terms of the mean but show a distinction in terms of standard deviation in favour of p-BRD.

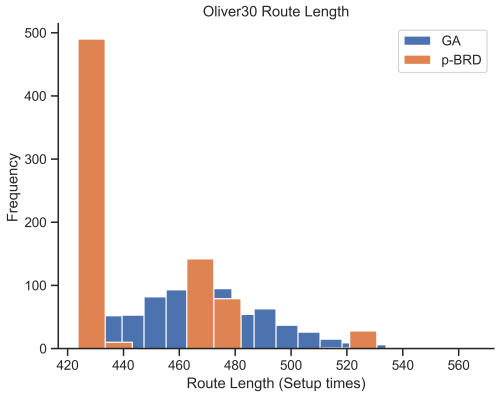

Figure 2: Setup time component of AOF on OLIVER30 data set optimised for AOF.

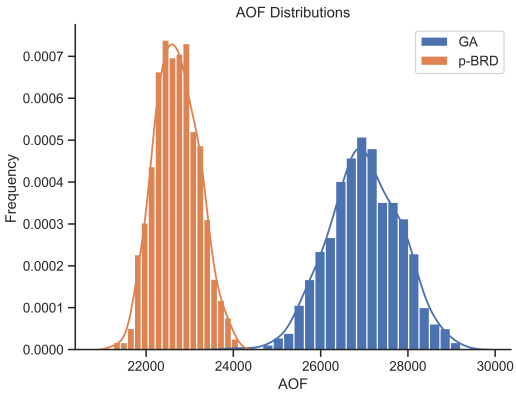

Figure 3: Comparison of AOF on data set LARGE.

The distribution for the data set SPREAD of p-BRD has a low standard deviation, due to the very low probability to accept temporarily worse solutions that was found during parameter optimisation ($p = 0.0012$). The standard deviation of the Evolutionary Algorithm is larger, but the algorithm often finds a better result than p-BRD. The observed behaviour may be explained by the fact that the initialisation strategy for p-BRD that is currently implemented (see Section 2) is well suited to solve the problem for the spread data set and exploration of the search space is less necessary.

In conclusion p-BRD produces better results regarding AOF on two of the data sets and comparable ones on the third, while being significantly faster in each case.

6 CONCLUSION AND FUTURE WORK

As there are no publicly accessible benchmark tests available for presented machine scheduling problem, we created new test sets of data for our comparison. In the experiments, p-BRD showed good results with better runtime behaviour than the Evolutionary Algorithm. Therefore, p-BRD can be considered as an algorithm feasible

Table 1: Statistical measures for the experiments

	GA			p-BRD		
Data set	Mean	STD	Runtime	Mean	STD	Runtime
Urgency	7587.8	224.6	9776ms	7467.01	156.8	4397ms
Spread	3803.2	178.2	7075ms	3868.6	135.4	100ms
Large	26999.2	820.1	35519ms	22700.2	500.1	5955ms
Oliver30	464.91	26.09	6217ms	443.23	27.72	3984ms

for real world application. The vanilla EA should be replaced by a state of the art algorithm to further validate this impression. Additional research on p-BRD may investigate more elaborate swapping strategies and analyse the effect on result and runtime. Applicability on further problems will be in focus as well an extension of the heuristics that closes gaps in the timeline of the production schedule.

ACKNOWLEDGMENTS

This work has partially been funded by Hessen Agentur under the Grant No. 593/18-16.

REFERENCES

[1] Ali Allahverdi. 2015. The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research* 246, 2 (2015), 345–378.

[2] Filipe Alvelos, Manuel Lopes, and Henrique Lopes. 2016. A matheuristic based on column generation for parallel machine scheduling with sequence dependent setup times. In *Computational Management Science*. Springer, 233–238.

[3] Sarahí Báez, Francisco Angel-Bello, Ada Alvarez, and Belén Melián-Batista. 2019. A hybrid metaheuristic algorithm for a parallel machine scheduling problem with dependent setup times. *Computers & Industrial Engineering* 131 (2019), 295–305.

[4] A Bagheri and M Zandieh. 2011. Bi-criteria flexible job-shop scheduling with sequence-dependent setup times—Variable neighborhood search approach. *Journal of Manufacturing Systems* 30, 1 (2011), 8–15.

[5] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for Hyper-Parameter Optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems (NIPS'11)*. Curran Associates Inc., Red Hook, NY, USA, 2546–2554.

[6] Ebru Demirkol, Sanjay Mehta, and Reha Uzsoy. 1998. Benchmarks for shop scheduling problems. *European Journal of Operational Research* 109, 1 (1998), 137–141.

[7] Yousef German, Ibrahim Badi, Ahmed Bakir, and Ali Shetwan. 2016. Scheduling to Minimize Makespan on IdenticalParallel Machines. *International Journal of Scientific & Engineering Research* 7, 3 (2016).

[8] Mikell P Groover. 2020. *Fundamentals of modern manufacturing: materials, processes, and systems* (7th. ed.). John Wiley & Sons.

[9] Vahid Kayvanfar, Amin Aalaei, Mahtab Hosseininia, and Mahdi Rajabi. 2014. Unrelated parallel machine scheduling problem with sequence dependent setup times. In *International conference on industrial engineering and operations management, Bali*. 7–9.

[10] Jae-Gon Kim, Seokwoo Song, and BongJoo Jeong. 2020. Minimising total tardiness for the identical parallel machine scheduling problem with splitting jobs and sequence-dependent setup times. *International Journal of Production Research* 58, 6 (2020), 1628–1643.

[11] Victor M Kureichick, Victor V Miagkikh, and Alexander P Topchy. 1996. Genetic algorithm for solution of the traveling salesman problem with new features against premature convergence. *Proc. of GA+ SE* 96 (1996).

[12] Dov Monderer and Lloyd S. Shapley. 1996. Potential Games. *Games and Economic Behavior* 14, 1 (1996), 124 – 143. https://doi.org/10.1006/game.1996.0044

[13] John F. Nash. 1950. Equilibrium Points in N-Person Games. *Proceedings of the National Academy of Sciences of the United States of America* 36, 1 (1950), 48–9.

[14] IM Oliver, DJd Smith, and John RC Holland. 1987. Study of permutation crossover operators on the traveling salesman problem. In *Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA*. Hillsdale, NJ: L. Erlhaum Associates, 1987., Hillsdale, N.J. : L. Erlhaum Associates, 224—-230.

[15] Tim Roughgarden. 2016. *Twenty lectures on algorithmic game theory*. Cambridge University Press.

[16] Jeffrey D. Ullman. 1975. NP-complete scheduling problems. *Journal of Computer and System sciences* 10, 3 (1975), 384–393.

[17] Nodari Vakhania. 2000. Global and Local Search for Scheduling Job Shop with Parallel Machines. In *Advances in Artificial Intelligence*, Maria Carolina Monard and Jaime Simão Sichman (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 63–75.

[18] Eva Vallada and Rubén Ruiz. 2011. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research* 211, 3 (2011), 612–622.

[19] Geoffrey Vilcot and Jean-Charles Billaut. 2008. A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem. *European Journal of Operational Research* 190, 2 (2008), 398–411.