Часопис из области економије, менаџмента и информатике
БизИнфо (БЛАЦЕ)

www.bizinfo.edu.rs

Journal of Economics, Management and Informatics
BizInfo (BLACE)

A machine learning-based selection approach for solving the single machine scheduling problem with Early/Tardy jobs

Pristup odabira zasnovan na mašinskom učenju za rešavanje problema rasporeda jedne mašine sa ranim/kasnim zadacima

Ahmed-Adnane Abdessemed [a*], Leila-Hayet Mouss[a], Khaled Benaggoune[a], Toufik Bentrcia[a]

[a] University of Batna 2, Laboratory of Automation and Production Engineering (LAP), Department of Industrial Engineering, Batna, Algeria

Article info

Original scientific paper/ Originalan naučni rad

Received/ Rukopis je primljen:
31 July, 2023
Revised/ Korigovan:
1 January, 2024
Accepted/ Prihvaćen:
2 February, 2024

DOI:
https://doi.org/10.5937/bizinfo2401001A

UDC/ UDK:
004.85

Abstract

Today, the algorithm selection paradigm has become one of the promising approaches in the field of optimization problems. Its main goal is to solve each case of an optimization problem with the most accurate algorithm using machine learning techniques. This paper treats the issue of the algorithm selection for the Single Machine Scheduling Problem with Early/Tardy jobs by adapting three meta-heuristics from the state-of-the-art, namely genetic algorithm, particle swarm optimization, and tabu search. In the proposed framework, we combine the running time and the cost function to get a new performance criterion. A large set composed of 98000 instances of the problem is generated with 12 features characterizing each instance. We carry a statistical comparison of the implemented meta-heuristics, and we evaluate 10 classifiers. It can be deduced that the Dagging algorithm combined with the Random Forest is the most likely to be the best classifier, which achieves 88.44% of the maximum accuracy.
Keywords: *single machine scheduling problem, early/tardy jobs, algorithm selection, machine learning, meta-heuristics*

Sažetak

Danas je paradigma izbora algoritama jedan od obećavajućih pristupa u oblasti optimizacijskih problema. Njegov glavni cilj je da reši svaki slučaj problema optimizacije najtačnijim algoritmom koristeći tehnike mašinskog učenja. Ovaj rad obrađuje pitanje izbora algoritma za problem raspoređivanja jedne mašine sa ranim/kasnim zadacima prilagođavanjem tri meta-heuristike iz najnovije tehnike, odnosno genetskog algoritma, optimizacije roja čestica i tabu pretraživanja. U predloženom okviru kombinujemo vreme rada i funkciju troškova da bismo dobili novi kriterijum učinka. Generiše se veliki skup sastavljen od 98000 instanci problema sa 12 karakteristika koje karakterišu svaku instancu. Izvodimo statističko poređenje implementirane metaheuristike i procenjujemo 10 klasifikatora. Može se zaključiti da je Dagging algoritam u kombinaciji sa Random Forestom najverovatnije najbolji klasifikator, koji postiže 88,44% maksimalne tačnosti.
Ključne reči: problem raspoređivanja jedne mašine, rani/kasni zadaci, izbor algoritma, mašinsko učenje, metaheuristika

1. Introduction

Current technological advancements in industrial systems have offered a substantial opportunity, which helped the boost of the organization's production ability to meet the customer's expectations such as the delivery date. For this reason, most modern production and manufacturing systems follow robust and well-defined philosophies including flexible manufacturing systems, and just in time (Chan et al., 2010). In fact, unpredicted challenges such as the early or late product delivery times do negatively affect the total production charges. So, numerous studies have been conducted in the literature to remedy this problem. The Single Machine Scheduling Problem, including Earliness/Tardiness penalties and distinct due dates (SMSPET), is amongst the most studied scheduling problems with non-regular objective function (Pinedo, 2012). Such problem belongs to the NP-hard class (see (Sourd et al., 2005)) and is typically harder to manipulate with respect to problems having regular objective

*Corresponding author
E-mail address: adnane.abdessemed@univ-batna2.dz

functions (Yau et al., 2008). Despite that several authors tried to solve this problem with different methods, there is no best algorithm for all configurations of the problem.

The SMSPET can be defined as follows: We consider J a set of n jobs to be executed on a single machine, where no idle times or job preemptions are permitted. It is also assumed that all jobs are ready for processing at t=0. Each job i is characterized by: processing time p_i, due date d_i, earliness penalty θ_i if i is terminated before the due date and tardiness penalty β_i if i is achieved after the due date. Hence, the goal is to determine a processing order for the elements of J to optimize the total sum of earliness and tardiness penalties. If we denote by C_i the completion time associated to the job i then the Earliness and the Tardiness formulas can be expressed as $E_i=\max\ \{d_i - C_i, 0\}$, and $T_i=\max\ \{C_i - d_i, 0\}$, respectively. At this level, the aim is to solve the following optimization problem:

$$Minimize \sum_{i=1}^{n}(\theta_i E_i + \beta_i T_i) \qquad (1)$$

Many authors have attempted to solve this problem based on exact methods, starting with the earliest works of (Abdul-Razaq et al., 1988), where a dynamic programming procedure is presented. Furthermore, two branch and bound algorithms have been investigated by (Sourd, 2009) for problems in which common and general due dates are assumed. In Tanaka et al., (2009), a successive sublimation dynamic programming-based exact approach is involved for the standard single machine scheduling problem.

Exact methods are computationally expensive, yet in most real-world cases, they are not quite practical. Exact methods ensure locating the ideal arrangement besides its optimality for each small size instance of the problem. However, the optimal solution cannot always be determined in a reasonable time. Hence, the assurance of finding an optimal solution can be abandoned for getting a generally good solution in polynomial time, and that is the goal of approximate algorithms.

Numerous heuristics have been deployed to solve the SMSPET, such as the beam search method in (Ow et al., 1989). A heuristic search was developed by (Sourd et al., 2005). Besides, simple linear dispatching rules have been considered to solve this problem as expressed by Longest Processing Time (LPT), Earliest Due Date (EDD), and Shortest Processing Time (SPT) rules (Valente, 2007). Heuristic algorithms can find feasible solutions in polynomial time; however, they cannot explore the total solution space.

Meta-heuristics can practically get near-optimal solutions in an acceptable time. Many meta-heuristics for the SMSPET were proposed in the literature. (Wan et al., 2002) examined the single machine scheduling problem with distinct due windows and weighted earliness/tardiness using Tabu Search procedure, including an optimal timing algorithm, with problem sizes of 15 to 80 jobs. A particle swarm optimization algorithm was proposed by (Tasgetiren et al., 2004) to solve the single machine problem with total weighted tardiness. (Tsai, 2007) proposed a genetic algorithm for solving a single machine scheduling problem with Early/Tardy jobs and with ready times. In addition, (Chang et al., 2008) recommended a genetic algorithm with injecting artificial chromosomes for the SMSPET. Ant colony and variable neighborhood search-based approaches were suggested in (M'Hallah et al., 2016) to solve SMSPET.

It is worth noting that the existence of several algorithms to solve the problem may appear as an advantageous aspect from a theoretical viewpoint. However, it practically yields a drawback in choosing the best algorithm among the proposed candidates. Testing all the existing algorithms for each new problem in an industrial environment is prohibitive in terms of computational complexity. Hence, such critical challenge can be stated as: 'How to select the more adequate algorithm for a new instance of the problem without testing?'.

As claimed by the No-Free Lunch theorem (NFL), the assumption that an algorithm outperforms the others in all cases needs to be revised. This theorem affirms that no algorithm has the best overall performance (Ho et al.,2001). However, if we consider that one specific algorithm can be used as a solver for a single instance of the problem, we are facing an Algorithm Selection Problem (ASP). The NFL states that one cannot estimate an algorithm's results relying only on the computational complexity. Instead, researchers have focused on empirical methods to estimate the best algorithm for every instance based on its performance on previously solved problems.

To the best of our knowledge, this work showcases an original contribution related to the development of reliable meta-heuristic selection frameworks for SMSPET, which fits well into the industrial needs. Three meta-heuristics were deployed to solve the SMSPET. Moreover, a set of machine learning algorithms has been tested to find the most accurate classifier for this problem. A statistical comparison of meta-heuristic results has been conducted, aiming to scrutinize each algorithm's behavior for SMSPET. We also try to define a meaningful performance metric that is not only adapted to industrial constraints but also agrees with the just in time philosophy. Therefore, our main objective is to study the impact of the algorithm selection technique on the quality of solutions.

The remainder of this paper is organized as follows. In the second section, we provide the state of the art related to the algorithm selection problem. The third section is dedicated to highlighting the elaborated methodology, including the description of its different components. Section four is devoted to the experimental results and discussions, and we end the paper with some conclusions and future perspectives in section 5.

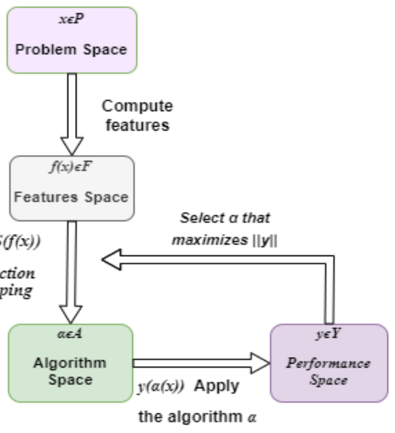
2. Background of the algorithm selection problem

The algorithm selection strategy is widely used in various ways to solve different problems. In the following, we discuss the theoretical aspect and state of the art related to the algorithm selection problem.

ASP has been tackled by several research attempts dating back to the earliest work of (Rice, 1976) in which he formulated the formal abstract model of ASP. The main purpose behind the ASP framework is to facilitate the selection of the most accurate algorithm from a set of algorithms to solve a given instance of a problem by maximizing the performance of solutions without involving all existing algorithms. Rice model can be stated as follows:

Let $x \in P$, be a given problem defined with the characteristics $f(x) \in F$, one must find the selection mapping $S(f(x))$ into algorithm space A, so that the picked algorithm $\alpha \in A$ optimizes a known performance measure $y(\alpha(x)) \in Y$. where P is the set of problem instances, F is the set of characteristics defining the problems; A is the set of all the algorithms considered to solve the problem, Y is the set of performance measures to be optimized, as illustrated in Figure 1.

Figure 1. Algorithm selection model proposed by Rice as reproduced from (Smith-Miles,2009)



The importance of Rice's model has notably increased in a parallel way with the development of the approximate methods of problem resolution, and this has created an issue in deciding which one is the best algorithm for every single problem. Consequently, researchers started to apply this model with different approaches in different fields. In the literature, several successful algorithm selection applications were published, such as SATzilla2012 (Xu et al., 2012) and Autofolio (Lindauer et al., 2015), which utilized the empirical hardness models to estimate the best algorithm for the SAT problems. Many NP-hard optimization problems were solved by different algorithm selection approaches. In Table 1, we summarize the core research works published in the field. For a detailed survey of algorithm selection, the interested reader can refer to (Kerschke et al., 2019).

Table 1. Essential works on the algorithm selection problem published in the literature.

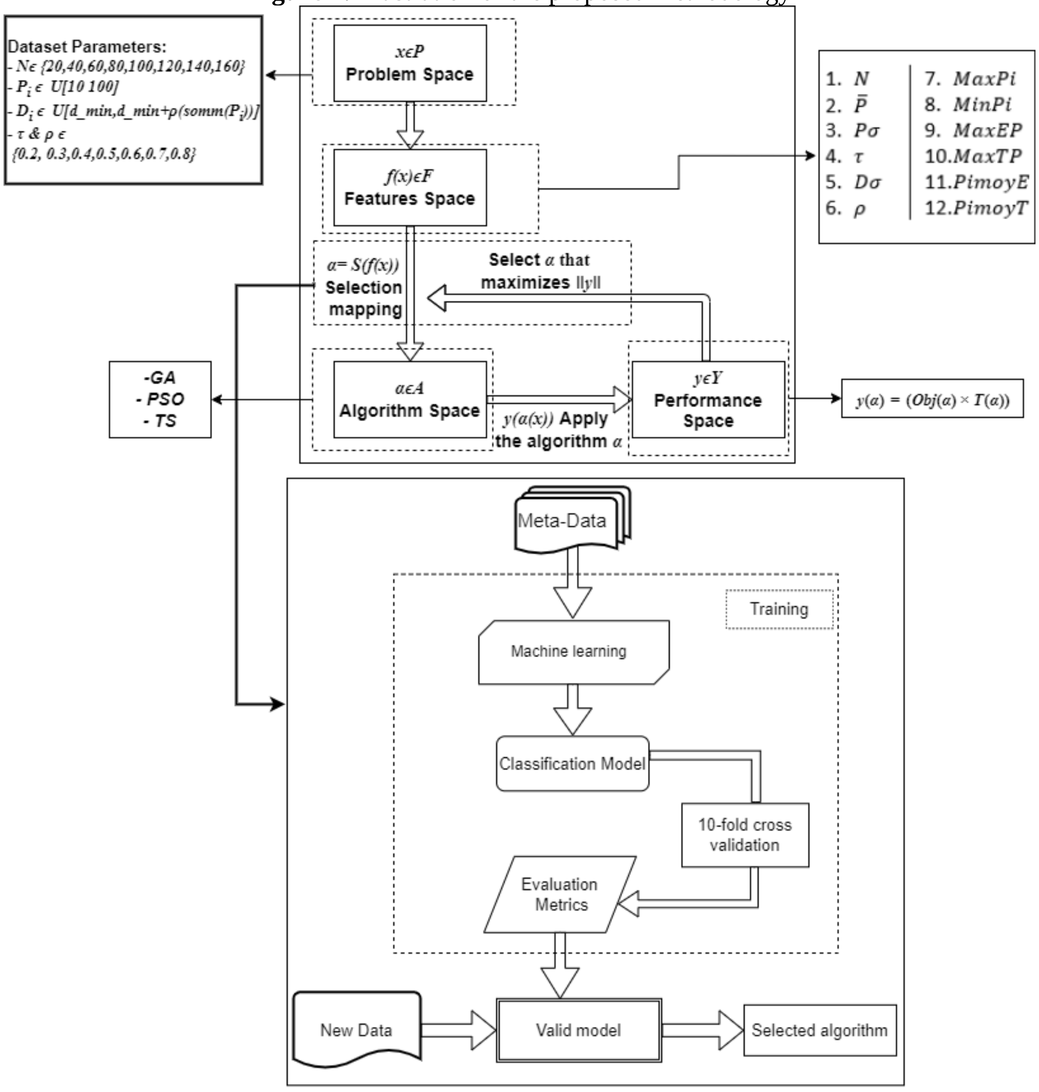
Reference	Studied Problem (P)	Selected algorithms (A)	Learning techniques $S(f(x))$
Nudelman et al., 2004	SAT Problem	Heuristics	Regression
Guo et al., 2007	MPE Problem	Exacts (Clique-Tree Propagation) Approximate (Stochastic sampling, Hybrid, Search)	Decision tree Bayesian network
Smith-Miles et al., 2009	Single Machine Scheduling Problem (Early/Tardy)	Heuristics (EDD, SPT)	Neural Networks, Decision tree, Self-Organization Maps
Kandaetal., 2011	Traveling Salesman Problem (TSP)	Meta-heuristics (TS, GRASP, SA, GA)	Multi-label classification (KNN,DT,SVM,NB)
Smith-Miles et al.,2013	Graph coloring problem	8 Meta-heuristics	Naive Bayes classifier, SVMs
Pihera et al.,2014	Traveling Salesman Problem (TSP)	Meta-heuristics (MAOSetLKH)	Bayesian Network, Decision tree, Random Forests, SVM
Wagneretal., 2018	TravelingThief Problem (TTP)	21Heuristics	Different, well-known algorithm selection approaches (Flexfolio, SATzilla, ISAC, 3S)
Scott et al., 2023	Satisfiability Modulo Theories (SMT)	-23 configurations of the cvc5 -Bitwuzla, cvc5 andZ3	AdaBoosting, Multi layer perceptron, linear ridge regressio
Kerschkeetal., 2019	Survey	-	-

Due to the scarce references in the literature regarding the algorithm selection theory on the SMSPET, only very few papers can be found, for instance, the work of (Smith-Miles et al., 2009). They applied the ASP using the Meta-learning approach. The aforementioned study is limited to only two heuristics, EDD and SPT, and its aim was to understand the connection between the scheduling problem configuration and the heuristic efficiency. In practice, heuristic methods are restricted in the sense of generating only admissible solutions. In contrast, the algorithm selection will be more complex using meta-heuristic algorithms.

3. Methodology

In this section, we explain in detail how we can adapt Rice's model to select the best meta-heuristic for the SMSPET. Figure 2 describes the principal components of the proposed framework.

Figure 2. Illustration of the proposed methodology



To adjust Rice's model within the context of our study, we proceed in the following steps. We start by generating a set of instances of SMSPET (problem instances space (P)), then we calculate the features for each instance (features space (F)). After that, we have to create an algorithm portfolio of meta-heuristics (algorithm space (A)) and define a performance measure (Y) to compare the algorithms. Machine learning algorithms are used to determine the selection mapping $S(F(x))$, which can be seen as a multi-class classification problem.

3.1. Instances generation

One of the most critical steps in this study is obtaining a large dataset representative of real-world problems. The existing benchmarks are not large enough to fit within the context of our study. Therefore, a dataset of 98000 instances is generated according to the following rules defined in (Sourd et al., 2005):

- Number of tasks (N): $N \in \{20, 40, 60, 80, 100, 120, 140, 160\}$;
- Processing Time (p_i): Generated randomly from a uniform distribution of the interval $U\,[10, 100]$;
- Due dates (d_i): For each task, a random due date is generated from the uniform distribution: $U[d_{min},d_{min}+\rho(somm(p_i))]$ where $d_{min}=max(0,(somm(p_i))(\tau - \rho/2))$, τ denotes the delay parameter and ρ the parameter of the row of times;
- We take the parameters τ and ρ from the following intervals: $\tau \in \{0.2, 0.3,0.4,0.5,0.6,0.7,0.8\}$, $\rho \in \{0.2,0.3,0.4,0.5,0.6,0.7,0.8\}$;
- Each time 25 problems are created.

3.2 Features definition

The following features can characterize every instance of the SMSPET. The first six features were used in (Smith-miles et al., 2009), and we propose six additional features in the context of our study as follows:

- Number of jobs N assigned to the instance;
- Mean Processing Time $\bar{P}$: The average processing time of all jobs in a given instance;
- Processing Time Range $p\sigma$: The range of all jobs' processing time in the instance;
- Tardiness Factor τ: In (Baker et al.,1974), the tardiness factor τ of a problem is defined as a coarse measure of the ratio of jobs, which are estimated to be tardy in an arbitrary sequence. This feature is given by $\tau = 1 - \frac{\Sigma d_i}{n \Sigma p_i}$;
- Due Date Range $D\sigma$: Defines the deviation of the due dates from the average due date for all jobs in the instance. It is designated by $D\sigma = \frac{(b-a)}{\Sigma p_i}$, where b indicates the maximum due date in the instance and a stand for the minimum due date in the instance;
- Penalty Ratio ρ: The maximum ratio of the tardy penalty to the early penalty over all jobs in the instance;
- MaxPi: the maximum processing time in the instance;
- MinPi: the minimum processing time in the instance;
- MaxEP: the maximum earliness penalty;
- MaxTP: the maximum tardiness penalty;
- PinmoyE: the mean earliness penalty;
- PinmoyT: the mean tardiness penalty.

The Correlation Attribute Evaluator (built-in Weka) is used to rank the features in terms of their correlation with the best algorithm. The features have been ranked as follows: (N, Drange, Pratio, MaxEP, MaxTP, Tfact, MinPi, Prange, MaxPi, PinmoyT, Pmoy, PinmoyE).

3.3. Meta-heuristics description

To solve SMSPET, we create a portfolio of algorithms from different types, namely population-based, swarm-based, and neighborhood search-based meta-heuristics. The algorithms used in this research are adapted from the state-of-the-art. In the following, we present a concise description of the algorithms used in this study.

Genetic Algorithm*:* The genetic algorithm (GA) is a population-based meta-heuristic originally developed by (Holland et al.,1992), where the population is composed of several chromosomes. Each chromosome is a set of genes. In our problem, every gene is a job to be run on the machine, and a chromosome represents a feasible solution, which is a sequence of jobs. The efficiency of good exploitation and exploration of the solution space has been proven using the genetic operators such as the crossover and the mutation operators. A standard GA is illustrated in Figure 3.A. In this paper the GA proposed in (Tasi, 2007) is adapted to solve the SMSPET problem.

The initial population is generated using some heuristic rules to accelerate the convergence of the algorithm. In this work, we create an initial population as follows:

- Sort tasks in ascending order of the execution time (SPT);
- Sort tasks in ascending order of the due date (EDD);
- Sort tasks in ascending order of the earliness penalty θ_i;
- Sort tasks in ascending order of the tardiness penalty β_i;
- The rest of chromosomes are generated randomly.

Figure 3. Standard representation of the metaheuristic's flowcharts

Figure 3.A. GA.

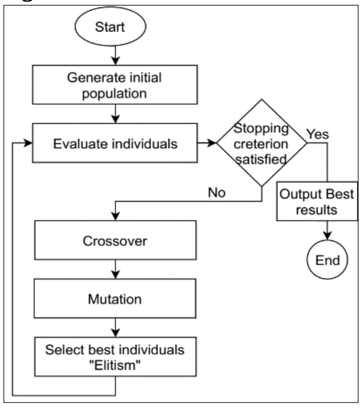

Start
Generate initial population
Evaluate individuals
Stopping creterion satisfied
Yes
No
Output Best results
End
Crossover
Mutation
Select best individuals "Elitism"

Figure 3.B. PSO

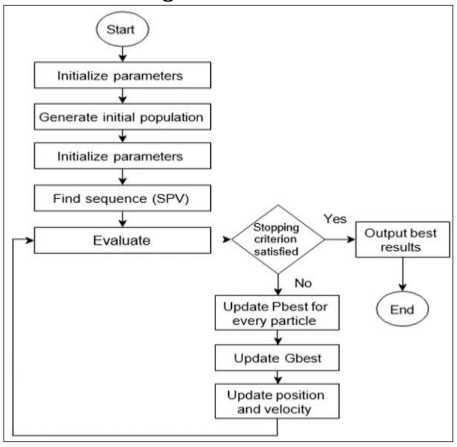
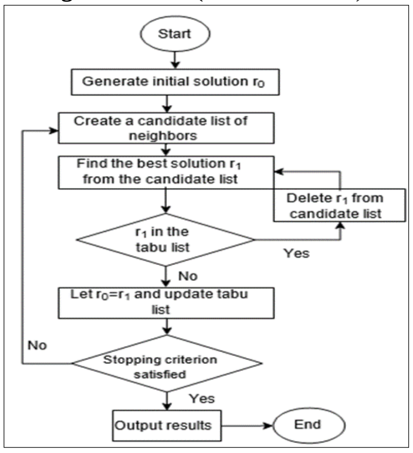

Start
Initialize parameters
Generate initial population
Initialize parameters
Find sequence (SPV)
Evaluate
Stopping criterion satisfied
Yes
Output best results
End
No
Update Pbest for every particle
Update Gbest
Update position and velocity

Figure 3.C. TS (Hao et al., 2017)


Start
Generate initial solution r0
Create a candidate list of neighbors
Find the best solution r1 from the candidate list
Delete r1 from candidate list
r1 in the tabu list
Yes
No
Let r0=r1 and update tabu list
No
Stopping criterion satisfied
Yes
Output results
End

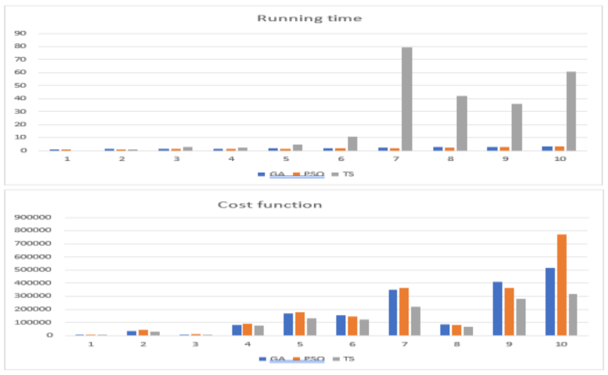
Tabu Search: The tabu search (TS) is a neighborhood search method, based on the hill-climbing algorithm with several steps directions (Laguna et al., 1991). The TS initially proposed in (Glover, 1989) and (Glover, 1990), is designed to avoid the local optima. By allowing moves from the current solution r to its best neighborhood solution r_1 even if r is better than r_1, with the condition that this solution is not in the tabu list. In this work, we adapt the TS proposed in (Wan et al., 2002) to the SMSPET. The TS starts from an initial solution r_0, which can be generated using a simple heuristic such as EDD, and iteratively moves from this solution to its best neighbor generated by exchanging positions of jobs. TS is presented in Figure 3.C.

***Particle Swarm Optimization**:* PSO is an optimization algorithm, inspired by the social behavior of flock of birds or school of fish. A swarm of particles simultaneously explores a problem's search space with the objective of finding the global optimum configuration. The original PSO was proposed by Eberhart and Kennedy (Eberhart et al.,1995) to optimize continuous functions. In this paper, the PSO proposed in (Tasgetiren et al., 2004) is adapted to fit the SMSPET. To enable the continuous PSO solving the SMSPET, we use a heuristic rule called *Smallest Position Value* (SPV). The PSO algorithm is presented in Figure 3.B.

3.4. Performance definition

To define the best algorithm among a set of candidate algorithms, one should ask the following question, 'How to compare many algorithms?'. The intuitive answer is to compare algorithms based on their best cost function values. However, in practice, the running time is an important factor which cannot be neglected. In some cases, the algorithm can find a good solution in a large running time, one can sacrifice the quality of solution for good running time. To clarify this aspect let us consider for example 10 instances randomly selected from our dataset, and try to solve them using GA, PSO, and TS. Figure 4 shows a bar plot of the best cost function and the running time obtained by each of these algorithms.

Figure 4. Illustrative example for the compromise between the cost function and the execution time



In the seventh instance, we notice that the TS has found the best solution, but the running time was extremely large compared to the other algorithms. However, for the eighth instance there is a small difference between the solutions obtained by these three algorithms, but the running time of the TS is too large compared with GA and PSO. So, if we have to privilege one of these algorithms over its counterparts, one should consider jointly the running time and the cost function. Consequently, we propose as an evaluation measure the product of the running time by the cost function value, as the objective is to minimize both of the aforementioned criteria.

Let $Obj(\alpha)$ be the best penalty obtained by the algorithm α and $T(\alpha)$ be the time needed to solve the given instance; our objective is to select the algorithm which has the minimum performance $Y(\alpha) = (Obj(\alpha) \times T(\alpha))$.

3.5 Algorithm prediction

After getting the algorithms results, we define the best algorithm by comparing the algorithm results for all instances. A single integer variable represents the best algorithm. Let $best_m$= 1 if GA is the best algorithm, $best_m$= 2 if PSO is the best algorithm, and $best_m$= 3 if TS is the best algorithm for the instance m. Every sample of the meta-data is a vector of features labeled by the value of the variable $best_m$.

The selection mapping $s(f(x))$ which associates for each set of features from F an algorithm α from A, is a learning problem (see Smith-Miles et al., 2009). Therefore, a classification model is built using machine learning algorithms on the meta-data. This model is used to predict the best algorithm for new incoming data based on the previously solved instances. A variety of machine learning algorithms have been tested; such as decision trees, neural networks, and ensemble learning algorithms. The classification model is validated by the ten-fold cross-validation process. (For more details on the evaluation metrics, the interested reader may refer to (Sammut et al., 2011)).

4. Computational results

In this section, we elucidate the main outcomes of the conducted numerical experiments and we provide some remarks and discussions.

Algorithms

All the algorithms used in this study were implemented and tested in MATLAB using a computer with the following features: Intel(R) Core (TM) i5-2430M CPU @ 2.40GHz and RAM = 6,00GB. Due to the high computational costs, we choose a small size population, and we run the algorithms with the same number of iterations.

Meta-heuristics results and competitiveness

In order to evaluate the modified meta-heuristics adopted in this paper, the algorithms are tested based on the benchmark used in (Tanaka et al., 2009). The investigated meta-heuristics are evaluated and compared on benchmark instances of size N=40, by computing the mean deviation from the optimal solution. The results are presented in Figure 5. Also, a competitiveness evaluation is performed to verify the efficiency of the proposed algorithm selection process. We qualify a set of algorithms as competitive when each algorithm surpasses the remaining ones on a subset of instances while being surpassed by at least another algorithm on the rest of instances, and these subsets are large enough. The competitiveness ratio *Com* is defined in (Messelis et al.,2014) as

$$Com = 2min\,(|A|/|T|,\ |B|/|T|) \qquad (2)$$

where, A stands for the ensemble of instances on which **algorithm A** exhibits better performance than **algorithm B**, and B denotes the ensemble of instances on which **algorithm B** do better than **algorithm A**, and T is the total set of instances. In the case of three algorithms *Com* can be defined as,

$$Com = 3min\ (|A|/|T|,\ |B|/|T|,\ |C|/|T|) \quad (3)$$

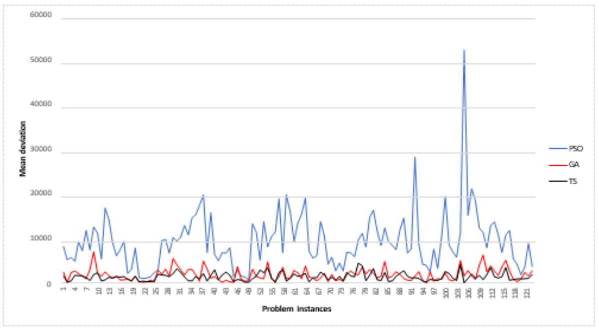
The closer *Com* is to 1, the more the competitiveness of the algorithms is verified to each other. In the proposed framework the competitiveness factor equals to:

The closer *Com* is to 1, the more the competitiveness of the algorithms is verified to each other. In the proposed framework the competitiveness factor equals to:

$$3min\ (28384/98000,\ 45106/98000,\ 24510/98000) = 0.75$$

which shows that the algorithms are sufficiently competitive, and the algorithm selection is useful.

Figure 5. Mean deviations of each algorithm from the optimal solution on instances with n=40



Discussion of the algorithms results

After running the proposed algorithms on the whole dataset, we categorize the obtained results by problem size (number of jobs). For each category, we have computed the Average measure provided in (9) and the standard deviation STD given in (10) of the solutions obtained by each algorithm over all the instances in the category, and we consider the best and the worst solutions for every category. The statistical results are classified in the following tables, where Table 2 shows the cost function and the running time results, and in Table 3, we depict the results of the proposed performance criterion Y.

$$Average = \frac{1}{Nbins} \sum_{r=1}^{Nbins} sol_r \quad (4)$$

$$STD = \sqrt{\frac{1}{Nbins} \sum_{r=1}^{nbins} (sol_r - Average)^2} \quad (5)$$

where *Nbins* is the total number of instances in the category, sol_r is the solution obtained for the instance r.

The analysis of the statistical results leads to the following remarks:

- The PSO algorithm is the fastest algorithm among the algorithms set;
- The TS algorithm provides excellent results on the small instances. Whereas, the running time is too large in large instances;
- GA generally has good results and an average running time;
- These results agree with the NFL theorem, so there is no best algorithm over all the algorithms used to solve the SMSPET;
- Let us consider a medium-size category of problems (N=100). If the best algorithm selection is decided according to the cost function value, the best algorithm is TS with the lowest average cost function. Nevertheless, the average running time of TS is six times larger than the PSO's average running time. This problem occurs in most categories.

The overview given by the statistical analysis is not satisfactory enough to help in algorithm selection. Therefore, our research is consolidated by using machine learning techniques to estimate the best algorithm for every instance.

Table 2. Cost function and running time statistics (The lowest values are written in bold)

		Cost function				Running time			
N	**Algorithm**	**Best**	**Worst**	**Average**	**STD**	**Best**	**Worst**	**Average**	**STD**
20	GA	279	**40266**	6267.14	**4692.24**	1.08	5.26	1.18	0.09
	PSO	341	40906	7189.24	5002.34	0.75	6.39	0.84	0.19
	TS	**276**	40300	**6230.47**	4695.21	**0.08**	**0.44**	**0.11**	**0.02**
40	GA	973	142729	23504.61	17010.74	1.26	3.20	1.34	0.09
	PSO	2110	147874	30961.07	19652.33	0.99	**1.82**	1.14	**0.05**
	TS	**694**	**142263**	**22979.44**	**16960.21**	**0.58**	2.25	**0.77**	**0.05**
60	GA	2477	242523	59000.13	41016.08	1.47	**1.98**	1.51	**0.02**
	PSO	5714	309591	74578.92	45552.82	**1.07**	5.7	**1.32**	0.08
	TS	**2012**	**239309**	**53358.02**	**38696.08**	2.28	3.48	2.58	0.15
80	GA	5022	529708	122254.38	81549.16	1.68	**2.08**	1.73	**0.02**
	PSO	11053	526278	139817.69	83258.55	**1.19**	4.81	**1.47**	0.08
	TS	**3519**	**476360**	**99920.85**	**71351.20**	4.88	8.85	5.85	0.84
100	GA	7847	769645	212492.33	139483.69	1.92	3.26	2.02	**0.08**
	PSO	21055	770606	228850.06	135372.82	**1.58**	**2.66**	**1.83**	**0.08**
	TS	**5799**	**662881**	**164989.74**	**117644.92**	10.51	24.17	12.04	1.34
120	GA	10336	1201430	326758.10	210294.96	2.20	**5.74**	2.57	0.22
	PSO	33595	1208934	341845.12	198984.65	**1.68**	5.89	**2.21**	**0.12**
	TS	**6813**	**1078785**	**248878.82**	**175812.08**	20.15	124.28	26.32	16.48
140	GA	15823	1621060	464788.80	296045.17	2.73	6.53	3.01	0.12
	PSO	43852	1565819	479409.08	276271.35	**2.37**	**5.05**	**2.62**	**0.11**
	TS	**10430**	**1505435**	**353361.03**	**248298.74**	35.01	311.65	49.01	31.42
160	GA	22374	2075765	624775.70	395411.90	3.14	7.63	3.49	0.17
	PSO	65212	2120556	643076.37	369213.60	**2.67**	**4.81**	**3.02**	**0.16**
	TS	**14037**	**1822343**	**478379.85**	**335102.18**	56.84	11791.01	78.67	152.98

Learning phase

Weka machine learning tool (Hall et al., 2009) is used for classification tasks. Many classification methods were tested on the meta-data, such as the trees algorithms (Random Forest, Functional trees …), neural networks (Multilayer perceptron, RBF Network, ...), and the classifiers with best accuracy were boosted using the Dagging algorithm as an ensemble learning technique (See for more details on the classifiers Fernandez et al., 2014). The classification results are presented in Table 4.

In general, ensemble learning techniques give the best results on all metrics (Table 4). The best result was obtained by the Dagging algorithm combined with Random Forest, for which accuracy equals to 88.44%, PRC Area equals to 0.93 and ROC Area equals to 0.96. Also, the best single classifier was the Functional Tree (FT), which has accuracy equals to 88.29%, PRC Area equals to 0.85 and ROC Area equals to 0.91. From the scheduling point of view, and for the Dagging algorithm combined with Random Forest case with a TP rate of 0.884, if we have 100 problems that require algorithm A as the best solution, the likelihood of selecting this algorithm with our approach is 88 out of 100, which is a good result regarding the fact that we don't need to evaluate all the algorithms.

To clarify our methodology's impact on a concrete case, let us consider the example used in a previous section (section 3.4). We tested each algorithm on the set of ten instances, and we applied the methodology of algorithm selection with the best classification model previously obtained (Dagging (Random Forest)). Table 5 shows a comparison of experimental results between every single algorithm and the algorithm selection methodology. On the total of the ten instances, the algorithm selection method holds the best average performance metric. In this example, the proposed framework guarantees finding at least 80% of the overall best solution.

Table 3. Performance criterion (*Y*) statistics (The lowest values are written in bold)

N	Algo	Best	Worst	Average	STD	N	Algo	Best	Worst	Average	STD
	GA	335.82	48838.32	7405.97	5620.49		GA	**16312.80**	1714905.65	429814.10	283452.60
20	PSO	267.41	44123.20	5988.93	4293.24	100	PSO	37308.12	**1412290.31**	**419585.49**	**248461.69**
	TS	**24.85**	**5364.68**	**699.29**	**545.09**		TS	63088.63	8777262.37	1986400.23	1441923.24
	GA	1257.04	185898.39	31422.13	22878.59		GA	**27488.67**	3292784.57	838283.19	545280.53
40	PSO	2391.18	162280.08	35355.88	22499.55	120	PSO	74286.32	**2685772.27**	**754506.09**	**442545.26**
	TS	**495.06**	**101119.51**	**17623.33**	**13079.81**		TS	140673.19	78521316.15	648270 0.26	6500909.50
	GA	**3764.39**	**367419.41**	**89291.36**	62081.78		GA	**46973.10**	4812429.82	1399808.71	893292.16
60	PSO	7575.33	426476.50	98480.26	**60381.28**	140	PSO	112750.87	**3944084.54**	**1256778.98**	**725562.94**
	TS	4854.87	688540.53	137445.30	100027.21		TS	372733.80	252903918.74	17066678.08	17447015.43
	GA	**8664.72**	920567.53	211951.70	141411.38		GA	**77863.32**	7695352.81	2178771.95	1382693.26
80	PSO	15997.67	**758210.31**	**205577.75**	**122842.50**	160	PSO	182644.01	**6669725.67**	**1940352.03**	**1116907.02**
	TS	20210.47	3163177.66	583304.76	428615.94		TS	891241.13	8369035749.57	37747485.37	110947476.55

Table 4. Machine learning results.

Algorithm	TP Rate	FP Rate	Precision	REcall	ROCArea	PRCArea	Accuracy
WiSARD	0.548	0.188	0.678	0.548	0.776	0.610	54.765
RBFNetwork	0.726	0.183	0.728	0.726	0.879	0.785	72.622
Multilayerperceptron	0.875	0.081	0.875	0.875	0.953	0.915	87.531
RandomForest	0.881	0.077	0.880	0.881	0.960	0.929	88.095
Dagging (LMT)	0.881	0.075	0.881	0.881	0.960	0.929	88.113
Trees.LMT	0.882	0.077	0.882	0.882	0.961	0.931	88.240
Dagging (CSForest)	0.883	0.080	0.882	0.883	0.960	0.931	88.292
FT	0.883	0.077	0.882	0.883	0.917	0.851	88.293
Dagging (FT)	0.883	0.076	0.883	0.883	0.954	0.917	88.327
Dagging (RandomForest)	0.884	0.074	0.884	0.884	0.962	0.934	88.440

Table 5. Algorithm selection on a test set using Dagging (Random Forest), where the bold results reflect the best outcomes and the underlined entries indicate the false-positive classified instance

N	GA			PSO			TS			Selected		
	solution	time	Y	solution	time	Y	solution	time	Y	solution	time	Y
1	4759.0	1.15	5472.85	5427.00	0.75	4097.07	**5005.00**	**0.12**	589.00	5005.00	0.12	589.00
2	32630.0	1.36	44376.80	42646.00	1.13	48278.06	**30361.00**	**0.77**	23377.27	30361.00	0.77	23377.27
3	**7684.0**	**1.54**	11833.36	10782.00	1.31	14177.14	7213.00	2.69	19405.47	7684.00	1.54	11832.61
4	**80431.0**	**1.50**	120804.77	91238.00	1.36	123960.87	77380.00	2.37	183741.45	91238.00	1.36	123960.87
5	170096.0	1.73	294657.33	**178025.00**	**1.45**	258080.34	130954.00	4.94	646590.22	170096.00	1.73	294657.33
6	153491.0	1.97	302620.03	**142743.00**	**1.83**	261239.47	123892.00	10.90	1350937.62	142743.00	1.83	261239.47
7	347734.0	2.28	792793.03	**361824.00**	**2.08**	752971.88	220782.00	79.23	17493034.99	361824.00	2.08	752971.88
8	83771.0	2.97	248708.55	**79698.00**	**2.59**	206412.12	64270.00	41.90	269269.,42	79698.00	2.59	206412.12
9	408991.0	2.90	1185175.93	**364455.00**	**2.68**	978174.89	278409.00	35.74	9950410.97	364455.00	2.68	978174.89
10	**516500.0**	**3.47**	1789907.50	770195.00	3.22	2481827.68	316202.00	60.58	19156977.41	516500.00	3.47	1789907.50
Avr	180608.7	2.09	376892.47	204703.30	1.84	377014.84	**125446.80**	23.92	3001287.43	176960.40	**1.82**	**321519.73**

5. Conclusions

This work presented a study of meta-heuristic algorithm selection for the SMSPET based on Rice's ASP model. Three meta-heuristics (GA, PSO, TS) were used to build an algorithm portfolio in the proposed framework, and 10 classifiers were tested for the algorithm selection. A novel performance metric is also proposed to evaluate each meta- heuristic performance by combining the running time and the cost function values. The experiments were carried on a dataset of 98000 instances characterized by 12 features. The features selection procedure shows that eight features are sufficient to get satisfying accuracy. The ensemble learning algorithm Dagging with Random forests possesses the best accuracy, which is 88.44%, and for the single machine learning algorithm, the functional trees have 88.29% accuracy.

This study has shed light on the mandatory requirement of implementing reliable algorithm selection frameworks, where additional insights are offered to managers for solving intractable industrial scheduling problems more efficiently. In addition, the presented machine learning results can enable readers to have a general perception of the behavior of different machine learning algorithms on the ASP for the SMSPET.

The proposed approach has been applied to a single machine scheduling problem; however, the steps are generic and can be extended to other case studies such as flow shop or job shop problems. The analysis of the data quality impact on machine learning outcomes and the adoption of the framework in a distributed environment such as in cloud computing for selecting the best service in an online mode are also interesting aspects to be addressed in the future.

References

Abdul-Razaq, T. S., & Potts, C. N. (1988). Dynamic programming state-space relaxation for single-machine scheduling. *Journal of the Operational Research Society*, *39*(2), 141-152. https://doi.org/10.1057/jors.1988.26

Baker, K. R., & Martin, J. B. (1974). An experimental comparison of solution algorithms for the single-machine tardiness problem. *Naval Research Logistics Quarterly*, *21*(1), 187-199. https://doi.org/10.1002/nav.3800210114

Chan, H. K., Yin, S., & Chan, F. T. (2010). Implementing just-in-time philosophy to reverse logistics systems: a review. *International Journal of Production Research*, *48*(21), 6293-6313. https://doi.org/10.1080/00207540903225213

Chang, P. C., Chen, S. S., & Fan, C. Y. (2008). Mining gene structures to inject artificial chromosomes for genetic algorithm in single machine scheduling problems. *Applied Soft Computing*, *8*(1), 767-777. https://doi.org/10.1016/j.asoc.2007.06.005.

Eberhart, R., & Kennedy, J. (1995, October). A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the sixth international symposium on micro machine and human science* (pp. 39-43). IEEE. https://doi.org/10.1109/MHS.1995.494215

Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems?. *The journal of machine learning research*, *15*(1), 3133-3181.

Glover, F. (1989). Tabu search—part I. *ORSA Journal on computing*, *1*(3), 190-206. https://doi.org/10.1287/ijoc.1.3.190

Glover, F. (1990). Tabu search: A tutorial. *Interfaces*, *20*(4), 74-94. https://doi.org/10.1287/inte.20.4.74

Guo, H., & Hsu, W. H. (2007). A machine learning approach to algorithm selection for NP-hard optimization problems: a case study on the MPE problem. *Annals of Operations Research*, *156*(1), 61. https://doi.org/10.1007/s10479-007-0229-6

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, *11*(1), 10-18. https://doi.org/10.1145/1656274.1656278

Hao, P., Wang, Z., Wu, G., Boriboonsomsin, K., & Barth, M. (2017, October). Intra-platoon vehicle sequence optimization for eco-cooperative adaptive cruise control. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)* (pp. 1-6). IEEE. https://doi.org/10.1109/ITSC.2017.8317879

Ho, Y. C., & Pepyne, D. L. (2001, December). Simple explanation of the no free lunch theorem of optimization. In *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No. 01CH37228)* (Vol. 5, pp. 4409-4414). IEEE. https://doi.org/10.1109/CDC.2001.980896.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Kanda, J., Carvalho, A., Hruschka, E., & Soares, C. (2011). Selection of algorithms to solve traveling salesman problems using meta-learning. *International Journal of Hybrid Intelligent Systems*, *8*(3), 117-128. https://doi.org/10.3233/HIS-2011-0133

Kerschke, P., Hoos, H. H., Neumann, F., & Trautmann, H. (2019). Automated algorithm selection: Survey and perspectives. *Evolutionary computation*, *27*(1), 3-45. https://doi.org/10.1162/evco_a_00242

Laguna, M., Barnes, J. W., & Glover, F. W. (1991). Tabu search methods for a single machine scheduling problem. *Journal of Intelligent Manufacturing*, *2*, 63-73. https://doi.org/10.1007/BF01471219

Lindauer, M., Hoos, H. H., Hutter, F., & Schaub, T. (2015). Autofolio: An automatically configured algorithm selector. *Journal of Artificial Intelligence Research*, *53*, 745-778. https://doi.org/10.1613/jair.4726

M'Hallah, R., & Alhajraf, A. (2016). Ant colony systems for the single-machine total weighted earliness tardiness scheduling problem. *Journal of Scheduling*, *19*, 191-205. https://doi.org/10.1007/s10951-015-0429-x

Messelis, T., & De Causmaecker, P. (2014). An automatic algorithm selection approach for the multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, *233*(3), 511-528. https://doi.org/10.1016/j.ejor.2013.08.021

Nudelman, E., Leyton-Brown, K., Devkar, A., Shoham, Y., & Hoos, H. (2004). Satzilla: An algorithm portfolio for SAT. *Solver description, SAT competition*, *2004*.

Ow, P. S., & Morton, T. E. (1989). The single machine early/tardy problem. *Management science*, *35*(2), 177-191. https://doi.org/10.1287/mnsc.35.2.177

Pihera, J., & Musliu, N. (2014, November). Application of machine learning to algorithm selection for TSP. In *2014 IEEE 26th International Conference on Tools with Artificial Intelligence* (pp. 47-54). IEEE. https://doi.org/10.1109/ICTAI.2014.18

Pinedo, M. L. (2012). *Scheduling* (Vol. 29). New York: Springer. https://doi.org/10.1007/978-3-319-26580-3

Rice, J. R. (1976). The algorithm selection problem. In *Advances in computers* (Vol. 15, pp. 65-118). Elsevier. https://doi.org/10.1016/S0065-2458(08)60520-3

Sammut, C., & Webb, G. I. (Eds.). (2011). *Encyclopedia of machine learning*. Springer Science & Business Media. https://doi.org/10.1007/978-0-387-30164-8

Scott, J., Niemetz, A., Preiner, M., Nejati, S., & Ganesh, V. (2023). Algorithm selection for SMT: MachSMT: Machine Learning Driven Algorithm Selection for SMT Solvers. *International Journal on Software Tools for Technology Transfer*, 1-21. https://doi.org/10.1007/s10009-023-00696-0

Smith-Miles, K. A. (2009). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)*, *41*(1), 1-25. https://doi.org/10.1145/1456650.1456656

Smith-Miles, K., Baatar, D., Wreford, B., & Lewis, R. (2014). Towards objective measures of algorithm performance across instance space. *Computers & Operations Research*, *45*, 12-24. https://doi.org/10.1016/j.cor.2013.11.015

Smith-Miles, K., James, R., Giffin, J., & Tu, Y. (2009). Understanding the relationship between scheduling problem structure and heuristic performance using knowledge discovery. *Learning and Intelligent Optimization*, *LION*, *3*.

Sourd, F. (2009). New exact algorithms for one-machine earliness-tardiness scheduling. *INFORMS Journal on Computing*, *21*(1), 167-175. https://doi.org/10.1287/ijoc.1080.0287

Sourd, F., & Kedad-Sidhoum, S. (2005). An efficient algorithm for the earliness-tardiness scheduling problem. *Optimisation Online,(1205)*.

Tanaka, S., Fujikuma, S., & Araki, M. (2009). An exact algorithm for single-machine scheduling without machine idle time. *Journal of Scheduling*, *12*, 575-593. https://doi.org/10.1007/s10951-008-0093-5

Tasgetiren, M. F., Sevkli, M., Liang, Y. C., & Gençyilmaz, G. (2004, June). Particle swarm optimization algorithm for single machine total weighted tardiness problem. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)* (Vol. 2, pp. 1412-1419). IEEE. https://doi.org/10.1109/CEC.2004.1331062

Tsai, T. I. (2007). A genetic algorithm for solving the single machine earliness/tardiness problem with distinct due dates and ready times. *The International Journal of Advanced Manufacturing Technology*, *31*(9-10), 994-1000. https://doi.org/10.1007/s00170-005-0261-0

Valente, J. M. (2007). Heuristics for the single machine scheduling problem with early and quadratic tardy penalties. *European Journal of Industrial Engineering*, *1*(4), 431-448. https://doi.org/10.1504/EJIE.2007.015391

Wagner, M., Lindauer, M., Mısır, M., Nallaperuma, S., & Hutter, F. (2018). A case study of algorithm selection for the traveling thief problem. *Journal of Heuristics*, *24*, 295-320. https://doi.org/10.1007/s10732-017-9328-y

Wan, G., & Yen, B. P. C. (2002). Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. *European Journal of Operational Research*, *142*(2), 271-281. https://doi.org/10.1016/S0377-2217(01)00302-2

Xu, L., Hutter, F., Shen, J., Hoos, H. H., & Leyton-Brown, K. (2012). SATzilla2012: Improved algorithm selection based on cost-sensitive classification models. *Proceedings of SAT Challenge*, 57-58.

Yau, H., Pan, Y., & Shi, L. (2008). New solution approaches to the general single-machine earliness-tardiness problem. *IEEE Transactions on Automation Science and Engineering*, *5*(2), 349-360. https://doi.org/10.1109/TASE.2007.895219