МИНИСТАРСТВО ОДБРАНЕ
И ВОЈСКА СРБИЈЕ
50.
СИМПОЗИЈУМ
О ОПЕРАЦИОНИМ
ИСТРАЖИВАЊИМА
SYM-OP-IS 2023
Тара, 18–21. 9. 2023.

STATISTICAL CONSIDERATIONS ABOUT MODELING PERFORMANCE OF EXACT SOLVERS ON PROBLEM INSTANCES OF $P||C_{MAX}$

MARIA BRACKIN[1] AND TATJANA JAKŠIĆ KRÜGER[2]

[1] Mohammed VI Polytechnic University, Rabat, brackinmaria@gmail.com
[2] Mathematical Institute, Serbian Academy of Sciences and Arts, Belgrade, tatjana@mi.sanu.ac.rs

Abstract: *When assessing a new algorithmic solution for an optimization problem, a set of problem instances is required on which the proposed algorithms may be compared against existing state-of-the art solvers. To achieve proper evaluation, we must identify key predictors of hardness and performance, i.e. an algorithm's ability to produce an optimal or best-known solution for a given problem instance. Considering the scheduling problem $P||C_{max}$, we find that the existing literature focuses on problem size and the ratio of tasks to processors. We suggest that this approach omits other problem features that may plausibly influence hardness and performance. Furthermore, existing methods do not systematically assess the influence of problem features in algorithm tests by considering the full range of values and all combinations of these values. Inspired by methods in the social sciences, we outline an experimental approach that considers a wider range of hardness and performance predictors over all conditions.*

Keywords: *combinatorial optimization, scheduling independent jobs to identical machines, benchmark instances, problem hardness, problem features, experimental setting*

1. INTRODUCTION

Modelling the performance of optimization algorithms is an integral step in tackling optimization problems [7,11]. Algorithm performance is assessed through the use of *empirical hardness models* [9], which built upon collected runtimes of the chosen algorithm on seen problem instances to predict runtime on unseen (novel) instances [7]. In order to properly asses the outcome of such performance tests, the model must consider different properties (*features*) of the problems attempted in hardness models.

Typical problem features, such as problem size, are conventionally assumed to cause an exponential rise in runtime [1]. However, the literature on problem hardness and algorithm performance suggests that the influence of such features is not straightforward [7]. Moreover, the assessment of problem properties is often limited to a narrow set of features, whereas some authors find promising approaches with non-conventional features; for example, [1] utilize the coefficient of variation and regularity index to investigate hardness of TSP instances.

We propose borrowing an approach from the social sciences: randomized experiments. We describe how creating a experiment, in which problem features are systematically and exhaustively tested, can improve understanding of problem hardness and algorithm performance. This approach includes two core principles: the inclusion of all plausibly relevant problem features, and the assessment of all combinations of these features through experimental conditions. Because we are not obligated to deal with only one type of feature, we define an expanded set of *structural features* [1] considered to be important; we also suggest a significantly increased number of runs over the space of possible experimental conditions.

After identifying the problem features of interest, we employ linear regression models to assess the contribution of these features to problem hardness and algorithm performance. We assess three dependent variables: (1) whether or not a problem instance is solved optimally in a given cutoff

time (binary outcome), (2) the proportion of problem instances solved optimally, and (3) the runtime of an algorithm (difficulty).

We outline our approach with respect to the scheduling of independent jobs to identical machines $P||C_{max}$ [4]. Specifically, we address the question of determining problem properties that contribute to the hardness and performance of an exact solver in [10] as well as other exact and heuristic algorithms for $P||C_{max}$ presented in [2,6,10]. All algorithms are compared against existing or well-known solvers previously published, thus, it is valuable to observe which problem instances have been difficult and if the conclusions have changed over time.

The paper is organized in the following way. After a brief description of $P||C_{max}$ instances, we review popular benchmark sets. In Section 3 we define problem features and outline our proposed experimental approach. In Section 4 we provide our conclusions and suggestions for future work.

2. RELATED WORK

$P||C_{max}$ originated as a scheduling problem on a computer, but due to its general form and simple formulation, it is of interest in both academic and industrial settings as a subproblem of other optimization problems. $P||C_{max}$ has been intensively studied in literature, however, producing the most up-to-date solver is an ongoing task.

2.1. Description of $P||C_{max}$ problem instances

The problem is usually defined via size parameters: number of tasks n and number of machines m. For example, a problem instance might be described in an input file like this:

```
20
10
42 68 35 1 70 25 79 59 63 65 6 46 82 28 62 92 96 43 28 37
```
(5)

where $n = 20, m = 10$ and the third line contains processing times for n independent tasks ($L = \{l_1, \dots, l_n\}$). The processing times (tasks' length) follow a predefined distribution which we denote as *class*. We introduce the parameter *index* to indicate difference between problem instances of the same size. Consequently, the problem instance uniqueness is defined by the following tuple: $(n, m, index, class)$.

The utilization of different problem instances is mainly motivated by the research question: how does problem size influence hardness of an instance? New research avenue investigates ratio n/m, as it was proposed in [6]. In what follows, we review papers that introduce ratio n/m as a major predictor of the instance difficulty, finding that the answer is not straightforward.

2.2. Literature Review

The study of problem instance hardness in the academic literature focuses primarily on problem size parameters. Similar to the satisfiability problem and phase transition in hardness, as the ratio n/k changes, researchers have aimed to find key values for a phase change in scheduling problems. Ideally, a relationship between n and m will be determined such that easy and hard problem instances are consistently identified and distinguished.

There are various benchmark sets in the literature for testing and comparing optimization algorithms. Five well known problem sets are listed chronologically in Table 1. Dell'Amico and Martello instances (DAM) are the most known benchmark set in the literature, often utilized for comparing exact solution algorithms [2]. Authors suggest five distinctive classes where processing times conform to the following distributions: (1) uniform in range [1,100]; (2) uniform in range [20,100]; (3) uniform in range [50,100]; (4) normal such that $\mathcal{N}(100,50)$; (5) normal such that $\mathcal{N}(100,20)$. The authors test their B&B algorithm when $n \in [10,10000]$ and $m \in \{2, 3, 5, 10, 15\}$ and solved almost all except small-sized instances: $(25,10)$, $(50,10)$, $(50,15)$ from class 3 and class 5. Additionally, perfect packing instances were proposed - for which the optimal solution

value is equal to $\sum_{i=1}^{n} l_i/m$ - as hard P||C_{max} instances. The size parameters were the same as previously outlined, while processing times were controlled by parameter Q. Results in Table 3 of their paper indicate interactions between parameters n, m and Q; however, these interactions have not been analyzed. The cases where not all instances are solved are for "small"-sized instances i.e. $n \leq 500$ while the hardest instances have been reported for larger Q. The B&B algorithm became very important as a method of differentiating easy from "difficult" instances as well as a method of determining solution's quality [3].

Table 1: Known P||C_{max} benchmarks sets from the literature.

2019.	Publication	Abbreviation
1	Dell Amico and Martello [2]	DAM
2	E1, E2, E3 and E4 instances [5]	E-inst
3	França-Frangioni instances[3]	FF
4	Haouri and Jemmali [6]	HJ
5	Mrad and Souayah [10]	MS

França - Frangioni instances have been considered difficult due to large values for n. The FF set consists of 780 files. Processing times may follow two types of distributions: uniform (U) and non-uniform binomal distribution (NU) such that task lengths vary between [10,100], [1,1000], [1, 10000] and $m \in \{5, 10, 25\}$. (https://site.unibo.it/operations-research/en/research/library-of-codes-and-instances-1.) The authors recognize that for the considered algorithms, UNIFORM instances are "easy" i.e. solved to optimality in a short amount of time. In addition, they report that as the range of processing times grows so does instances' hardness. [3] are the first in the literature to recognize the ratio n/m as an important indicator of instances hardness. On the complete set, the authors report that for $2 < n/m \leq 10$ instances are the most difficult. Authors utilize two more benchmark sets: BINPACK and TRIPLET instances (see http://people.brunel.ac.uk/~mastjjb/jeb/info.html). The former consists of easy and hard, and the later set constitutes only of hard instances.

In [6] the authors test their algorithm on different benchmark sets from [2]. Perfect packing instances are solved to optimality, which constitutes them "easy" for their B&B algorithm. Next, on the set of 1900 instances, n takes values between 10 and 10000, while $m \in \{3, 5, 10, 15\}$. For each combination (n, m), 10 problem instances are randomly generated and the cutoff time set to 1200 seconds. Results show that the most instances are solved to optimality, whereas 5 instances for which $(n, m) = (50,15)$ have not been solved for the given CPU time. Next, [6] test their algorithm on the FF instances. Out of 780, exactly 21 were not solved. The authors conclude that all considered instances are too easy, thus they have proposed a new set of problem instances which we denote as HJ instances. The number of machine is fixed and set to $2n/5$ and $l_i \in U[n/5, n/2]$. For each $n \in \{20,30, 40, 50, 60, 70, 80, 90, 100, 150, 200\}$ a set of 20 instances is generated. The instances have shown to be more challenging and are used in the following papers as difficult.

In [8] author presented several solution methods for the benchmark instances. In case of the FF instances, the WL algorithm strongly depends on the n/m values. For $n/m = \{2,4,5\}$, the easiest instances are when $n/m = 2$, as performance deteriorates for larger n/m. Similarly, in case of DAM instances, those which satisfy $n/m \leq 2.5$ are easier solved. However, both [2] and [6] report that small-sized instances (n=50, m=15) were not solved within 1200 sec.

[11] introduce exact algorithm based on an arc-flow model. Until recently, the algorithm was the state-of-the-art exact method for P||C_{max}. This algorithm was tested on HJ and DAM benchmark sets, as well as on two new additional classes for DAM instances. Class 6 contains tasks where $l_i \in U[n, 4n]$, and class 7 presents processing times l_i that follow a normal distribution $\mathcal{N}(4n, n)$. The results show that HJ instances are easy for the exact solver. Unlike DAM instances in literature, [11] generated new instances such that they reflect the ratio $n/m = \{2,2.25,2.5,2.75,3\}$. The first 5 classes of problem were easy to solve, however, class 6 and class 7 seemed to be difficult with

regard to runtime and the number of solved instances per problem size. Additionally, the most difficult instances reported for DAM instances were not tackled by [11].

What we have observed in the review literature is the lack of systematic review of the executions that should be conducted in order to determine influence of problems size parameters as well as other structural features. In the following section, we present structural features of the $P||C_{max}$ problem that are potential additional predictors of difficulty of the considered problem set.

3. PROBLEM FEATURES AND METHODOLOGY

In this section we describe the problem features, experimental approach and statistical models we suggest to estimate the most influential features for key outcomes. Identification of relevant problem features is essential for estimation of the performance of optimization algorithms. In our proposed method, we seek to meet both goals described in the introduction: the inclusion of all plausibly relevant problem features, and the assessment of all combinations of these features through experimental conditions.

In Table 2 we list all the problem features first presented in [10] and included in our proposed models. Classically, a problem is defined by at least three features: n, m and $class$ (1-3 in Table 2), where $class$ is a specific distribution of task length, as presented in Section 2. We also include n/m in line with the academic literature as outlined above. Additionally, we compute features based on generation of problem instances: average length (av. length), median value ($median$), standard deviation ($std.dev$), maximum length (max), minimum length (min), range ($range$), and number of distinct elements (k) (items 4-10). These features are likely to influence difficulty and algorithm performance. We employ a set of problem features used that are quickly measurable [11].

The purpose of testing algorithms on a series of problems with such features is to assess their performance on important outcomes. We have identified three different outcomes as indicators of problem hardness as well as of a good algorithm's performance: $runtime$, $CPLEXstatus$ and $prop_{opt}$ (items 13-15). In this study, each of these outcomes is assessed separately as a depependent variable in a statistical model.

Through artificially generation of problem instances, it is possible to generate problem instances with the same dimension and different hardness, here defined as $runtime$. For instance, two problems with the same values of n, m and $class$ will take a different amount of time to solve (if solved at all). The difference in runtime will influence whether a problem is solved to optimality.

We determine whether a problem is solved optimally through assignment of $CPLEXstatus$ (1 or 0) based on a cutoff time (1200 seconds) above which a solution, while possible, is insufficiently fast. For each subtype and class $(n, m, class)$, our dataset contains 10 individual problem instances which may include some optimally and some non-optimally solved (or unsolved) instances. Therefore we also evaluate $prop_{opt}$, the proportion of problem instances solved optimally.

Table 2: List of features used in difficulty estimation module for $P||C_{max}$

	Feature notation	Definition (predictors)
1.	$class$	Probability distribution for elements of set *L*
2.	n	Cardinality of set *L* i.e. number of tasks
3.	m	Number of machines
4.	$av.length$	Average processing time
5.	$median$	Median processing time
6.	$std.dev$	Standard deviation of set *L*
7.	max	Maximum processing time
8.	min	Minimum processing time
9.	$range$	$max - min$
10.	k	Number of different task lengths in set *L*
11.	n/m	Average number of tasks per machine

12.	$index$	Index of an instance within group of instances with the same problem size
13.	$runtime$	Execution time
14.	$CPLEXstatus$	Status report if objective value is found
15.	$prop_{opt}$	Proportion of problem instances solved optimally per problem

3.1. Experimental conditions

Having defined and included all plausibly relevant problem features, we now define the experimental conditions. The goal of an experimental approach is to systematically manipulate problem features to cover the space of all possible problems. In the social sciences, this involves exposing participants to a certain set of experimental stimuli; here, we generate problem instances based on a set of inputs.

Our experimental conditions will be combinations of n, m and $class$. The existing dataset in [11] contains several combinations of n and m, but leaves large gaps in possible values for these features. In order to determine the individual contributions of n, m and n/m to an outcome, we must generate a near-continuous range of values for each feature and permute all combinations of these values.

For n and m, we propose a range of values between 0 and 200 by units of 5. Therefore, n/m will also range between 0 and 200. For example, these two conditions vary by n only, which results in different values of n/m:

- $class = 1 \;\; n = 20 \;\; m = 10 \;\; n/m = 2$
- $class = 1 \;\; n = 25 \;\; m = 10 \;\; n/m = 2.5$

With a total of 40 values for each of n and m, the combination of possible values is 1,600. Over 7 classes of problem, we would have 11,200 conditions in our experiment.

3.2. Linear Regression Models

The problem features we describe in Table 2 can be reasonably expected to contribute to optimal solving of problems and to hardness of problems. Parameters n, n/m, $av.lengt$, max, and k, are expected to increase difficulty and decrease the likelihood of optimal solutions. By contrast, the number of machines m can be expected to decrease difficulty and increase optimal solutions.

We propose three regression models to assess the influence of problem features on the outcomes of interest:

$$CPLEXstatus = \alpha + \beta_1 class + \beta_2 n \beta_3 m \beta_4 n/m + \beta_5 av.length + \beta_6\, median + \beta_7\, std.dev + \beta_8 max + \beta_9 min + \beta_{10} range + \beta_{11} k + \epsilon$$

$$runtime = \alpha + \beta_1 class + \beta_2 n \beta_3 m \beta_4 n/m + \beta_5 av.length + \beta_6\, median + \beta_7\, std.dev + \beta_8 max + \beta_9 min + \beta_{10} range + \beta_{11} k + \epsilon$$

$$prop_{opt} = \alpha + \beta_1 class + \beta_2 n \beta_3 m \beta_4 n/m + \beta_5 av.length + \beta_6\, median + \beta_7\, std.dev + \beta_8 max + \beta_9 min + \beta_{10} range + \beta_{11} k + \epsilon$$

The models include interaction terms between n, m and n/m as parameters influence one another.

3.3. Power Analysis

The final step in our experimental design is to determine the number of observations necessary to run the statistical analysis outlined above. In order to be able to detect the effect of problem features on the outcomes of interest, we require a sufficient number of data points.

The effect size in power tests of linear regression is the difference between the fit of two models, typically a full model with all parameters and a reduced (e.g. random variance) model. The f^2 statistic is defined as follows, where R^2 is the statistical fit for the two models being compared:

$$\frac{R^2_{full} - R^2_{reduce}}{1 - R^2_{full}}.$$

As inputs to the power calculation we include the number of parameters, statistical significance, statistical power, and minimum detectable effect as defined above. Including interaction terms, the

number of model parameters in our regression models is 15. In line with convention, we assume a significance level of $\alpha = 0.05$ and $power = 0.80$. We use an f^2 from Cohen (1988) where a small effect has a value of $f^2 = 0.02$. With these assumptions, to detect a small effect of problem features on CPLEXstatus, $prop_{opt}$ and $runtime$, we would require 953 observations for each of our 11,200 conditions, for a total of 10,673,600 observations.

5. FUTURE WORK

In future work, the first task will be to generate large datasets of algorithm performance based on the experimental approach outlined above. With a large (> 10 million) number of datapoints, this will be a computationally intensive task. Second, these datasets will be analyzed statistically using a variety of methods: linear regression models as well as Machine Learning techniques. Interesting similarities and differences may emerge when comparing these techniques, elucidating the benefits and limitations of the experimental method as a complementary approach.

Alongside these empirical tasks, we will work on the conceptual development of the concepts of hardness and performance. For instance, we can challenge or justify cut-off points for the determination of optimal solving within a given time frame. We can also consider introducing the results of SLACK and LPTUB heuristics as predictors for hardness, further expanding the list of predictors of algorithm performance.

ACKNOWLEDGEMENT

This work was supported by the Ministry of Science, Technological Development and Innovations of Republic of Serbia agreements nos. 451-03-47/2023-01/200029 and 451-03-47/2023-01/ 200122.

REFERENCES

[1] Cerasela Crisan, G., Nechita, E., and Simian. D. On randomness and structure in euclidean TSP instances: A study with heuristic methods. IEEE Access, 9:5312–5331, 2021.

[2] Dell'Amico, M., & Martello, S. (1995). Optimal scheduling of tasks on identical parallel processors. ORSA Journal on Computing, 7(2), 191-200.

[3] Frangioni, A., Necciari, E. and Scutella, M. G. A multi-exchange neighbor-hood for minimum makespan parallel machine scheduling problems. J. Comb. Optim., 8(2):195–220, 2004.

[4] Garey, M. R. and Johnson, D. S. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, 1979.

[5] Gupta, J. and Ruiz-Torres, A. J. A listfit heuristic for minimizing makespan on identical parallel machines. Production Planning & Control, 12(1):28–36, 2001.

[6] Haouari, M., Gharbi, A. and Jemmali, M. Tight bounds for the identical parallel machine scheduling problem. Int Trans Oper Res, 13(6):529–548, 2006.

[7] Hutter, F., Xu, L,. Hoos, H. and Leyton-Brown, K. Algorithm runtime prediction: Methods & evaluation. Artificial Intelligence, 206:79–111, 2014.

[8] Lawrinenko, A. Identical parallel machine scheduling problems: structural patterns, bounding techniques and solution procedures. PhD thesis, 2017.

[9] Leyton-Brown, K., Nudelman, E. and Shoham, Y. Empirical hardness models: Methodology and a case study on combinatorial auctions. Journal of the ACM, 56(4):1–52, 2009.

[10] Maleš, U., et al. Controlling the difficulty of combinatorial optimization problems for fair proof-of-useful-work-based blockchain consensus protocol. Symmetry, 15(1):140, 2023.

[11] Mrad, M. and Souayah, N. An arc-flow model for the makespan minimization problem on identical parallel machines. IEEE Access, 6:5300–5307, 2018.

[12] Smith-Miles, K. and Lopes, L. Measuring instance difficulty for combinatorial optimization problems. Computers & Operations Research, 39(5):875–889, 2012