Towards the 50th anniversary of Machine Scheduling

Sergey Sevastianov
Institute of Mathematics, Novosibirsk, Russia.
seva@math.nsc.ru

Abstract
The most urgent organizational problems of the present-day Theory of Scheduling (TS) are discussed, namely: creating a database in the Internet aimed to accumulate all information about the results obtained in TS; elaboration of a new problem classification for TS that could be taken for the foundation of the proposed database and would be targeted to eliminate the gap between Project and Machine Scheduling.

In 1954 the well-known (now classical) paper by S. Johnson issued which, most likely, should be treated as the first theoretical paper in Machine Scheduling. Since passed nearly 50 years. The flow of papers followed Johnson's paper and related to analysis of problems, where one should schedule some jobs on some machines, have made up an area of knowledge known today as Machine Scheduling (MS, for short). Of course, on the threshold of the jubilee it would be nice to recall interesting results obtained in MS, to speak about achievements, but I'd prefer to talk about the problems accumulated in MS (and TS, in a whole) for the past years (I mean not special scheduling problems, like the Johnson problem, but rather organisational ones), about the reasons hampering a further development of TS, and discuss possible ways of resolving those problems.

The problems of TS, we are going to speak about, are unavoidable corollaries of the nature of TS itself. What is TS today? A typical feature of TS is that it models **guided discrete processes running in time**. Such processes could be found practically in any sphere of human activity. (That is why the variety of models and problems being considered in TS is so huge.) It is known that the mankind's activity grows exponentially in time. The number of various technological processes generated by man increases swiftly (as well as their complexity). And all those processes need be organised. – Otherwise the mankind is threatened with going down in the chaos of technogeneous disasters. Theory of Scheduling is just the theory, which is aimed to organise such processes feasibly (and preferably, optimally) in time. That is why **the role of TS will be permanently increasing in future**. And that is why the troubles arising in TS and hampering its further development are so urgent. Which troubles are meant? I'd like to focus on the following 4 problems.

1. Practical inadequacy. Problem settings being currently considered and analysed within the framework of the traditional TS are too narrow to reflect and accommodate the large variety of real-life scheduling problems.

2. Informational inadequacy. Too many different models and problems in TS requiring different approaches for their solution. (As was noted above, this is just a slight reflection of a much wider variety of real-life technological processes around us.) And too many results obtained for those hundreds of problems for the last 50 years. Too much information which is poorly accessible. Today it is quite complicated to get an information about results obtained for a particular problem (different kinds of results obtained for this and close problems).

3. Theoretical insufficiency. There is an evident lack of strong and generally valid theoretical results and universal methods in TS. Classical mathematics is used insufficiently. Too much attention and human resources is paid to developing technical solutions of partial problems. Hundreds of machine programs have been written for the last years to implement numerous variants of the local search approach without any theoretical guarantees, and this mostly gives nothing to theory. A known wisdom has been forgotten that "Nothing is as good for practice as a good theory".

Note that in the case that Problem no.1 is solved somehow, the variety of problems that are to be considered in TS will drastically increase, and this will redouble the second problem (the informational inadequacy). At the same time, the need in strong and universal theoretical methods will also increase, because it is impossible to solve 1 million problems separately, one-by-one (it would require a huge army of schedulers). Thus, the third problem will be also redoubled. So, all three problems should be considered as an indivisible complex, all together.

Finally, **problem no.4** relates to integrity of TS, eliminating of a parallelism in research. Today there are two large areas in scheduling named Project Scheduling (PS) and Machine Scheduling (MS) that are developed mostly in parallel by different groups of people which poorly contact with each other. As a result, the areas differ by the methods in use and by research directions. Up to date, I would say that MS is more advanced in theory: many polynomial time approximation schemes have been developed, there are inapproximability results, on-line algorithms, etc. However, there is a strong feeling that most results obtained in MS could be extended towards PS models. Therefore, an approach of PS and MS would be mutually beneficial. The first steps in this direction have already been done: Brucker a.o. [2] elaborated a 3fold classification scheme for PS-problems similar to the scheme developed for MS about 20 years ago. Yet, an ultimate solution would be a unified classification of TS-problems, combining these two areas of TS completely. To that end, I think, MS should perform the first step towards PS, — a step that would enable one to consider (within the framework of MS) problems admitting more general precedence constraints on the set of all operations and more general resource constraints. As such a first step, our new classification scheme could be treated. Its full description requires about 20 pages, so I am able to speak about its general principles only. Yet before let us return to the first 3 problems.

To begin with, let us make certain that the first shortcoming of TS (practical inadequacy) really takes place. The last generally adopted classification of MS-problems was presented by Bo Chen, Potts and Woeginger [3] in 1998 and, to our mind, was out of date at the very first day when it was issued. Indeed, that classification was based on the following principles:

— each operation belongs to one job only;
— each operation requires a single machine;
— no two operations of the same job can be processed simultaneously;
— no two operations can be processed on the same machine simultaneously.

At the same time, for each of the above requirements a counter-example can be easily shown — a real-life scheduling model in which the requirement does not hold. Thus, **the first way** to obtain new scheduling problems is a refusal of the above canonical requirements.

The second way is analysis and synthesis. The first part (analysis) assumes a decomposition of any problem into elementary independent components ("atoms"), while the second one is a creation of new problem settings by constructing new combinations of atoms. Which "atoms" are meant? Let us consider the classical Job Shop problem commonly denoted by a single letter "J". This "single letter" can be decomposed into 4 independent "atoms":

— digraph $\overline{G}_j = (O_j, U_j)$ specifying an order of processing the operations of job J_j; in the case of the Job Shop problem it is a simple path;

— constraints on the length of arcs of each graph $\overline{G}_j$; in our case each arc has zero length; in the general case it may be any real number (either positive, or negative, or zero);

— graph $G_j = (O_j, E_j)$ specifying incompatibility restrictions given for pairs of operations of each job J_j; in our case this is a complete graph;

— restrictions on the choice of executors for each operation; in our case this is "D", which means "*Dedicated machines*" (i.e., a single executor is assigned beforehand to each operation).

The first three atoms obviously relate to job requirements, while the last one — to machine environment. (Note that in the standard classification scheme the letter "J" stands in the first field, destined to specify a machine environment.)

We can easily obtain another problem by changing any of the 4 atoms of letter "J". For instance, we can replace the value "D" of the last atom by one of the values "Sh", "Mp", "P", "M", etc., and will obtain a similar problem but in a different machine environment. Here "Sh" (which means *Shop model*) extends "D" to the case when the whole set of machines is partitioned into subsets (shops), and each operation may be executed on any machine of a certain shop (thus, we have *dedicated shops* and alternative machines in each shop); "Mp" denotes a more general environment of *Multi-purpose machines*, "P" is just *Parallel machines* (a special case of "Sh", when there is a single shop), and "M" (*Multi-processor tasks*) extends "D" to the case, when any operation requires several machines at a time, when it is being processed. Of course, there are many more possible values for this "atom", but I have no time to mention them all.

Thus, a great bulk of real-life scheduling models have never been considered and analysed within TS just because the existing classification scheme cannot identify them. Apparently, the inadequacy of the common classification is being felt by many schedulers today. E.g., in the third edition of Brucker's book [1] it is suggested to introduce parameters $\alpha_1' \in \{\mathbf{P,Q,R,PMPM,QMPM},\circ\}$, $\alpha_1'' \in \{\mathbf{G,X,J,F,O},\circ\}$, and $\alpha_2 \in \{1, m, \circ\}$ in the first field of a 3fold classification. This enables one to consider, e.g., combinations of *Parallel machines* with different *job requirements* specified by parameter α_1''. In fact, the letters **J, F, O** have here not quite traditional meanings, because they are deprived of their machine environment feature. This, however, generates some questions, e.g.: how to define the combination of **PMPM** with **F**? where the *shop environment* is hidden here? why the *job requirements* are placed in the 1st field? etc.

Next, the **3rd way** to obtain new scheduling problems is a decreasing of the level of requirements down to the *operational level*. As for example:

— specifying the precedence and incompatibility constraints on the set of **all** operations;

— constraints like *no wait* for particular pairs of operations (instead of **all** operations of every job);

— permitting interruption in processing of particular operations (unlike the present tradition, when either **every operation** can be interrupted **any number** of times, or **no interruption** is allowed);

— a feasible time interval pre-assigned to each particular operation (instead of the whole job), etc.

Now we return to the **informational inadequacy** of TS. What should be done, so that the information about all results (of any kind) obtained for a particular problem would be easily accessible? The idea to write a thick book is a bad idea. Firstly, the author has to run across thousands of papers written in English, Russian, Chinese, Japanese,… and scatted through hundreds of journal. Today this work is beyond of strength of any scheduling group. Secondly, such a book will not be really accessible for most schedulers. And thirdly, it will not be convenient to extract a required information from this book. Let me explain. Each problem, according to our principles of problem classification, is nearly a 20-dimentional vector consisting of values of 20 independent elementary parameters. Suppose, I'd like to get an information about the results obtained for a particular problem, for all its sub-problems and super-problems, as well as for all near-by problems (where the distance between vectors is measured, say, in the Hamming metric). The question is, which order of problems (read: 20-dimentional vectors) should be chosen in the book so as to provide a quick extraction of the required information? The answer is **no one**.

In fact, the only thing that would enable one to do this is a well-organised database in the Internet. To our mind, the desired database should enable one to:

1. create an **informational request** by specifying:
    - **a.** the main problem you are interested in — just by choosing one of the suggested values for each independent parameter (from the set of those 20 basis parameters);
    - **b.** the *radius* of the interest (whether you wish to learn something about "close" problems; how much "close");
    - **c.** whether you wish to learn about sub-problems and super-problems;
    - **d.** what kind of results you are interested in;

    e. whether you want to learn the near-by open problems;

2. **add an information** about a new result **automatically**, by specifying: a problem (like in **a**), a type of the result (like **d**), a short abstract, an obligatory reference to a source paper;
3. **add comments** to a particular result present in the database (say, that the result is wrong, or, may be, that there is a stronger result, etc.); a reference to a source is also desirable;
4. submit suggestions to the database administrator concerning the **list of the basis parameters** and the **lists of variants of their possible values**.

I insist on the word "automatically" in p.2, because such a huge database can be created and permanently maintained only if hundreds of schedulers are able to participate in this work. And p.3 is aimed to ensure a control of the correctness of the results presented to the database. In fact, there will be not two, but rather hundreds of referees per each result.

What do we have today in the Internet related to scheduling? We have two sites belonging to P. Brucker & S. Knust [4] and to P. Crescenzi & V. Kann [5]. They are evidently insufficient. First of all, they contain results of only four types (polynomial solvability and NP-completeness of exact solutions, approximability and nonapproximability — with ratio performance guarantees only), while many interesting and theoretically important results of other kinds remain out of the scope of these sites. Actually, even in those 4 directions, the collections of results on those sites are far from being complete. And in fact, the sites are not databases and do not provide a convenient access to the information. Of course, the information contained on the sites should be used while creating the desired database.

What is needed? First, probably, an international grant is needed to elaborate the database. May be, the ORS or MPS could finance such a project. Second, a more adequate problem classification should be designed. One of possible versions of such a classification has been elaborated in Novosibirsk. It is based on a General Problem (GP), so that every problem in the classification is just a sub-problem of GP. The latter combines independent features in their most general form, which ensures that all other feasible combinations of values of the elementary features will also be compatible. The complete record of a problem is divided into **9 fields**: machine environment, distribution of the operations over jobs, batch constraints, interruptions, precedence and incompatibility constraints on the set of operations (machine routes of jobs are also here), specificity of processing times, families of available time intervals for the operations, special requirements (like, say, permutational schedules), objective functions. Each field can normally be described by several elementary parameters.

The suggested classification is definitely far from being perfect and need be widely discussed to promote its further improvement. However, even in its current form it considerably contributes to solving problems 1, 2 and 4, and creates a necessary basis for solving problem 3.

Acknowledgements

Supported by the Russian Foundation for Fundamental Research (Grant 99-01-00601).

References

[1] P. Brucker, Scheduling Algorithms, third edition, Springer, 2001.

[2] P. Brucker, A. Drexl, R.H. Möhring, K. Neumann, E. Pesch, Resource-constrained project scheduling: notation, classification, models and methods, *Eur. J. Oper. Res.* **112** (1999), 3-14.

[3] B. Chen, C. N. Potts, G. J. Woeginger, A review of machine scheduling: complexity, algorithms and approximability, *Handbook of Combinatorial Optimization*, Boston: Kluwer Acad. Publ. Vol. **3**. (1998), 21-169.

[4] http://www.mathematik.uni-osnabrueck.de/research/OR/class/

[5] http://www.nada.kth.se/~viggo/problemlist/compendium.html