bookbuildeR

painlessly create computer science books

Thomas Weise

http://github.com/thomasWeise/bookbuildeR

Creating Teaching Material

- OK, so you want to write an electronic book or script for your students or a general audience.
- That is always a good idea.
- But it costs much work.
- Actually: work and *overhead*.
- *work* = you write the contents of the book
- *overhead* = installation of software, compiling, uploading, synchronizing, versioning, testing example code, …

Creating Teaching Material pt. 2

- But how do you do write the book?
- Why not use `LaTeX` and create `pdf`?
- But maybe you also want to have a `html` version for the web?
- Or an `epub`/`azw3` version for handheld devices?
- You may need several conversion tools.
- You will have some sort of build process.
- Which will require some software and packages.

Creating Teaching Material pt. 3

- Now you want to add example program code.
- How do you do that?
- Just “write it in the book”?
- Will it compile? Is it correct?
- How to “give it to the students”?
- Put it in another folder structure that you somehow reference and that you can compile and test?
- Sounds uncomfortable, too.

Creating Teaching Material pt. 4

- Now you want to publish the book.
- So you need to upload it to a web space.
- All in all, you will have a process of several minutes.

- Can you collaborate with other authors? How?
- Can your readers/students submit feedback, questions, problems, or even directly suggest changes to the book?

Idea

- Write book in `markdown` text format with `svg` figures (and `LaTeX` math and macros).
- Put book source code in a `GitHub` (write book iteratively, get versioning and collaboration for free).
- When you make a change to the book, commit it to the `GitHub` repository.
- The book is automatically compiled to `pdf`, `html`, `epub`, and `azw3` and the result is uploaded to a website (`GitHub pages` branch of repository).
- Possibility: Comments/issues submitted by users

Idea part 2

- You can link to a source *code repository* where you keep all example program codes and data.
- You can include snippets from the files in the *code repository* as formatted program code into your book via a simple reference mechanism.
- Whenever you compile the book, the latest version of the *code repository* is checked out and used.
- If you commit to the *codes repository*, it will trigger a build/testing which triggers a rebuild of the book.
- You can create fully-fledged programs as examples.

Realization

- We use only OSS tools such as `pandoc`, `TeX live`, and `calibre` to compile `markdown` sources with `svg` figures to `pdf`, `html`, `epub`, and `azw3`.
- The tools are executed by a single build process implemented as `R` package: `bookbuildeR`
- The tool chain will find the linked source code repository, check it out, extract the referenced code snippets, include them into the book sources, and compile the book sources

Realization pt. 2

- All required software is packaged into a `docker` container
- The build process using the container is triggered by a commit to the book repository and runs in a continuous integration environment (`Travis CI`, free for OSS)
- `Travis CI` allows uploading the result of a build to the `GitHub pages` branch of a repository, which we use to automatically publish the compiled book
- `Travis CI` allows to trigger the build of another repository after a successful build of one repository, which we use to automatically rebuild the book after the code repo changes (and passes its unit tests)

Example

An Introduction to Optimization Algorithms

Thomas Weise

2019-10-14

1 Introduction

Today, algorithms influence a bigger and bigger part in our daily life and the economy. They support us by suggesting good decisions in a variety of fields, ranging from engineering, timetabling and scheduling, product design, over travel and logistic planning to even product or movie recommendations. They will be the most important element of the transition of our industry to smarter manufacturing and intelligent production, where they can automate a variety of tasks, as illustrated in Figure 1.1.

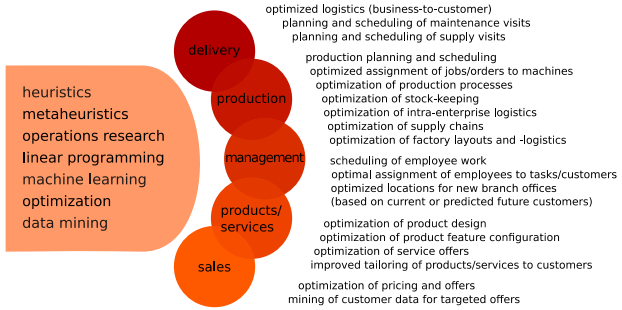

Figure 1.1: Examples for applications of optimization, computational intelligence, machine learning techniques in five fields of smart manufacturing: the production itself, the delivery of the products, the management of the production, the products and services, and the sales level.

Optimization and Operations Research provide us with algorithms that propose good solutions to such a wide range of questions. Usually, it is applied in scenarios where we can choose from many possible options. The goal is that the algorithms propose solutions which minimize (at least) one resource requirement, be it costs, energy, space, etc. If they can do this well, they also offer another important advantage: Solutions that minimize resource consumption are of en not only cheaper from

Table 2.2: The lower bounds $lb f$ for the makespan of the optimal solutions for our example problems. For the instances abz7, la24, and yn4, research literature (last column) provides better (i.e., higher) lower bounds $lb(f)^*$.

name	n	m	$lb(f)$	$lb(f)^*$	source for $lb(f)^*$
demo	4	5	180	180	Equation (2.2)
abz7	20	15	638	656	[117,157,161,162]
la24	15	10	872	935	[10,157]
swv15	50	10	2885	2885	Equation (2.2)
yn4	20	20	818	929	[157,161,162]

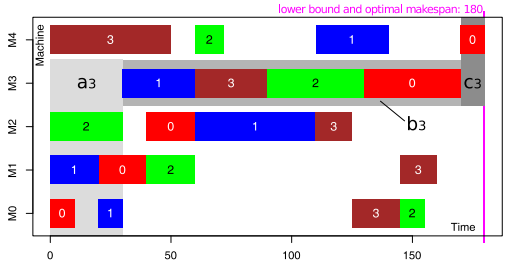

Figure 2.7: The globally optimal solution of the demo instance Figure 2.1, whose makespan happens to be the same as the lower bound.

Figure 2.7 illustrates the globally optimal solution for our small demo JSSP instance defined in Figure 2.1 (we will get to how to find such a solution later). Here we were lucky: The objective value of this solution happens to be the same as the lower bound for the makespan. Upon closer inspection, the limiting machine is the one at index 3.

We will find this by again looking at Figure 2.1. Regardless with which job we would start here, it would need to initially wait at least $a_3 = 30$ time units. The reason is that no first sub-job of any job starts at machine 3. Job 0 would get to machine 3 the earliest after 50 time units, job 1 after 30, job 2 after 62,

Listing 3.9 An excerpt of the 1swap operator for the JSSP, an implementation of the unary search operation interface Listing 2.9. 1swap swaps two jobs in our encoding of Gantt diagrams. (src)

```
1  public class JSSPUnaryOperator1Swap
2      implements IUnarySearchOperator<int[]> {
3    public void apply(int[] x, int[] dest,
4        Random random) {
5  // copy the source point in search space to the dest
6      System.arraycopy(x, 0, dest, 0, x.length);
7
8  // choose the index of the first sub-job to swap
9      int i = random.nextInt(dest.length);
10     int job_i = dest[i]; // remember job id
11
12     for (;;) { // try to find a location j with a different job
13       int j = random.nextInt(dest.length);
14       int job_j = dest[j];
15       if (job_i != job_j) { // we found two locations with two
16         dest[i] = job_j; // different values
17         dest[j] = job_i; // then we swap the values
18         return; // and are done
19       }
20     }
21   }
22 }
```

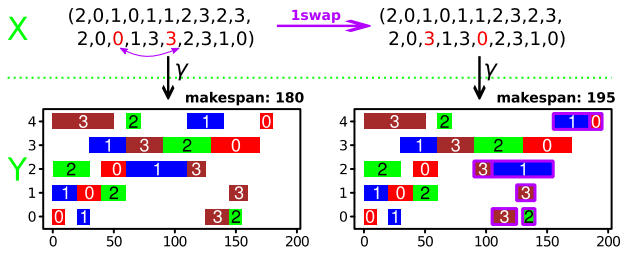

Figure 3.5: An example for the application of 1swap to an existing point in the search space (top-left) for the demo JSSP instance. It yields a slightly modified copy (top-right) with two jobs swapped. If we map these to the solution space (bottom) using the representation mapping γ, the changes marked with violet frames occur (bottom-right).

widely used in statistics applies Equation (4.2) as default [17,95]. In an ideally-sized data sample, the number of elements minus 1, i.e., $n-1$, would be a multiple of q. In this case, the k^{th} cut point would directly be located at index $h = (n-1)\frac{k}{q}$. Both in Equation (4.2) and in the formula for the median Equation (4.1), this is included the first of the two alternative options. Otherwise, both Equation (4.1) and Equation (4.2) interpolate linearly between the elements at the two closest indices, namely $\lfloor h \rfloor$ and $\lfloor h \rfloor + 1$.

$$\text{quantile}_q^k(A) = \begin{cases} h = (n-1)\frac{k}{q} & \\ a_h & \text{if } h \text{ is integer} \\ a_{\lfloor h \rfloor} + (h - \lfloor h \rfloor) * a_{\lfloor h \rfloor + 1} - a_{\lfloor h \rfloor} & \text{otherwise} \end{cases} \tag{4.2}$$

Quantiles are more robust against skewed distributions and outliers.

If we do not assume that the data sample is distributed symmetrically, it makes sense to describe the spreads both left and right from the median. A good impression can be obtained by using quantile_4^1 and quantile_4^3, which are usually called the first and third quartile (while $\text{med} = \text{quantile}_4^2$).

4.4.3.2 Outliers

Let us look again at our previous example with the two data samples

- $A = (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 14)$
- $B = (1, 3, 4, 4, 4, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 11, 12, 10008)$

We find that:

- $\text{var}(A) = \frac{1}{19-1} \sum_{i=0}^{n-1} (a_i - 7)^2 = \frac{198}{18} = 11$ and
- $\text{var}(B) = \frac{1}{19-1} \sum_{i=0}^{n-1} (b_i - 533)^2 = \frac{94763306}{18} \approx 5264628.1$, meaning
- $\text{sd}(A) = \sqrt{\text{var}(A)} \approx 3.317$ and
- $\text{sd}(B) = \sqrt{\text{var}(B)} \approx 2294.5$, while on the other hand
- $\text{quantile}_4^1(A) = \text{quantile}_4^1(B) = 4.5$ and
- $\text{quantile}_4^3(A) = \text{quantile}_4^3(B) = 9$.

4.4.3.3 Summary

There again two take-away messages from this section:

1. An average measure without a measure of dispersion does not give us much information, as we do not know whether we can rely on getting results similar to the average or not.

Bibliography

[1] Scott Aaronson. 2008. The limits of quantum computers. Scientific American 298, 3 (2008), 62–69. DOI:https://doi.org/10.1038/scientificamerican0308-62

[2] Tamer F. Abdelmaguid. 2010. Representations in genetic algorithm for the job shop scheduling problem: A computational study. Journal of Software Engineering and Applications (JSEA) 3, 12 (2010), 1155–1162. DOI:https://doi.org/10.4236/jsea.2010.312135

[3] Joseph Adams, Egon Balas, and Daniel Zawack. 1988. The shifting bottleneck procedure for job shop scheduling. Management Science 34, 3 (1988), 391–401. DOI:https://doi.org/10.1287/mnsc.34.3.391

[4] Kashif Akram, Khurram Kamal, and Alam Zeb. 2016. Fast simulated annealing hybridized with quenching for solving job shop scheduling problem. Applied Soft Computing Journal (ASOC) 49, (2016), 510–523. DOI:https://doi.org/10.1016/j.asoc.2016.08.037

[5] Ali Allahverdi, C. T. Ng, T. C. Edwin Cheng, and Mikhail Y. Kovalyov. 2008. A survey of scheduling problems with setup times or costs. European Journal of Operational Research (EJOR) 187, 3 (2008), 985–1032. DOI:https://doi.org/10.1016/j.ejor.2006.06.060

[6] Lee Altenberg. 1997. NK fitness landscapes. In Handbook of evolutionary computation, Thomas Bäck, David B. Fogel and Zbigniew Michalewicz (eds.). Oxford University Press, New York, NY, USA. Retrieved from http://dynamics.org/Altenberg/FILES/LeeNKFL.pdf

[7] Gene M. Amdahl. 1967. Validity of the single processor approach to achieving large-scale computing capabilities. In American federation of information processing societies: Proceedings of the spring joint computer conference (AFIPS), April 18–20, 167, Atlantic City, NJ, USA. 483–485. DOI:https://doi.org/10.1145/1465482.1465560

[8] Mehrdad Amirghasemi and Reza Zamani. 2015. An effective asexual genetic algorithm for solving the job shop scheduling problem. Computers & Industrial Engineering 83, (2015), 123–138. DOI:https://doi.org/10.1016/j.cie.2015.02.011

[9] David Lee Applegate, Robert E. Bixby, Vašek Chvátal, and William John Cook. 2007. The traveling salesman problem: A computational study (2nd ed.). Princeton University Press, Princeton, NJ, USA.

[10] David Lee Applegate and William John Cook. 1991. A computational study of the job-shop scheduling problem. ORSA Journal on Computing 3, 2 (1991), 149–156. DOI:https://doi.org/10.1287/ijoc.3.2.149

Summary

- Creating educational material is good
- Let us not waste time with overhead and instead concentrate on work
- Iterative work is better than creating “complete and immutable” editions of books.
- The `bookbuildeR` tool chain is a first attempt to give some support.
- It is not professional, just a personal project.
- But it shows what can be possible.
- `http://github.com/thomasWeise/bookbuildeR`

Thanks. 谢谢

Prof. Dr. Thomas WEISE
Institute of Applied Optimization
Hefei University, Hefei, Anhui, China
tweise@hfuu.edu.cn, tweise@ustc.edu.cn
http://iao.hfuu.edu.cn

汤卫思
应用优化研究所
合肥学院