Scheduling Optimisation of a Manufacturing Design Area: A Case Study

C.A. Garcia-Santiago(✉), Anna Rotondo, and Fergus Quilligan

Irish Manufacturing Research, Dublin, Ireland
carlos.garcia@imr.ie

Abstract. This paper presents the work implemented to improve the production scheduling of a real life manufacturing plant in Ireland. Since the scheduling algorithms are integrated in a wider cognitive system, where human and machine intelligence collaborate, an important constraint will be the maximum allowed computational time. This paper will also demonstrate the impact of using heuristic rules for the generation of initial solutions and provide a comparison between Hill Climbing, Harmony Search and Simulated Annealing.

Keywords: Optimisation · Scheduling · Metaheuristics · Harmony search

1 Introduction

This paper presents the work developed and implemented in a tooling design department of a high-tech manufacturing company in Ireland that designs and manufactures tooling solutions for the pharmaceutical sector. In this department, 28 engineers design the parts that will be later manufactured on site. Each tooling to manufacture is unique, and the design process is the first step of the production, followed by the manufacturing, assembly and testing of the final products. In this paper we will consider a job as the set of 3 operations with precedence constraints that will ultimately output the list of parts to manufacture. The first operation in the job checks that all necessary information for the correct completion of the design is available. If something is missing or badly understood the job will not continue until the necessary information is corrected, and in this case the job will start again. This pre-check operation is always assigned a fixed 15 min slot, and is always performed by one of the 4 most experienced designers in the department. There is an additional constraint in the number of pre-check operations that any given designer can perform daily. The second operation is the actual design process, which takes the greatest time to perform. Here one designer uses a 3D CAD software to produce the blueprints for the tooling. Finally, the third operation is the checking of the previous design step, trying to find possible defects or omissions. The main constraint for this check operation is that the designer performing it must be different from the one who made the design.

J. Del Ser (ed.), *Harmony Search Algorithm*, Advances in Intelligent Systems and Computing 514, DOI 10.1007/978-981-10-3728-3_13

The assignment of operations to resources is done manually by the department manager. Some designers are only permitted to perform a subset of the operations. This measure is usually taken to ensure that the more junior designers only perform the design operation, or for some very senior designers to perform only the checking operation. For the research in this paper we will also consider that each designer can only perform one operation at any given time, and that once started, the designer cannot perform any other operation until the current one is completed.

Usually the design department is overloaded and can cause a bottleneck for the rest of the production line. To ensure the correct balance of all the line, each of these design jobs have an internal due date that must be met to avoid later delays in the manufacturing line. The main constraint of the process is satisfying the customer demand on time, by (i) minimizing of the number of jobs missing the due date and (ii) for those jobs that are late minimise the number of days past the due date. A secondary goal is balancing the workload of the designers. Workload is quite unbalanced in the original schedule, primarily due to differences between designers regarding skill and experience. The skill of an engineer designing a particular piece of machinery is a very important factor in the quality of the final product. If the design is not correct, the manufacturing phase can be impossible to do, causing the job to go back to design phase, or the final customer will return the part due to imperfections or malfunctioning.

The research presented in this paper consists of finding new ways to help the planning and scheduling department of the company by rescheduling the production upon introduction of new constraints by the design department manager. The optimisation algorithms work as part of a cognitive system where human and machine intelligence cooperate to find optimal schedules in the event of unexpected disruption events like resource unavailability or new orders in the system. In this scenario, an important requirement is the speed of results, necessary to allow a "communication" between user and machine intelligence. In this research we follow the results of recent studies suggesting that the human attention timespan is decreasing in last years, from 12 to 8 s in just a 10 years time span, see [19].

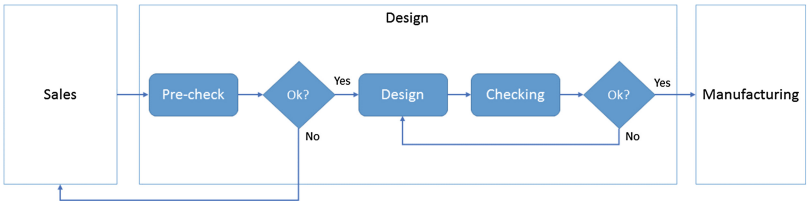

Fig. 1. Design process

1.1 Ensuring Quality

In many scheduling problems a very important factor that production mangers need to take into consideration is the quality of the performed work. The final product quality in the present case is heavily dependent on the skill of the designers. This issue has been studied in the past, for example in the framework of the Resource-Constrained Project Schedule Problem. In the literature we find two main methods to deal with quality issues; the first one modifies the activity duration depending on the skill of the assigned resource [1], while in other cases an additional re-work activity is scheduled if the skill of the resource assigned to the original activity is not high enough for the activity complexity [2]. In this use case quality was incorporated by the company by developing a job risk vs resource grade heuristic and by always having a check activity after every design activity, to ensure that the quality of the work performed by the first designer was good enough. On top of it, the manager sometimes introduces new constraints by selecting the designers that are available for the check operation among those with most experience.

The skill of a designer is an indicator of the experience that he has had in the past designing a particular tooling. This experience will not only have a positive impact on the time required to perform the operation, but will also decrease the number of non conformities or errors found by customers in the final product, as an unskilled designer is more prone to make errors in the design (faulty product), and as such is a very important key performance indicator (KPI) for the plant managers. Obviously it would be preferable if we could always assign each operation to the designer with highest skill in it, but the constraints on the availability of resources means this would often have negative consequences in the balance of workload.

The system implemented in the use case company to ensure the best possible quality in the jobs is based around two different values. The first one is a 'risk value' assigned to each job, depending on several factors including similarity with past jobs, type of tool, the machine where the tool will be installed, etc. Against this risk we map another value known as 'grade', which is manually assigned to each designer by the design department manager as an indication of his seniority and capacity to take on difficult jobs. There are currently four grades in use.

The system works as follows:

1. First, for each operation the available designers are selected based on the risk of the job and on their grade. In other words, we find the subset of designers with grade equal or greater than the minimum required for the job's risk value.
2. Second, from this first subset we filter out those designers without any skill in the operation.
3. In the case that there's no element in this final subset (i.e., no one has the necessary grade and experience for the operation), we will default to the subset of designers with the highest grade.

2 The Algorithms

In this section we will review the two metaheuristics used in this research. A metaheuristic is an algorithm that uses other algorithms iteratively, to search for the global optimum of a function across all the search space. Every metaheuristic starts exploring the search space in one particular point (in the case of single solution metaheuristics) or in several starting ones (population based). Though the standard approach is using a random initial solution(s), another approach is to use some heuristic to create the initial solution. Another important consideration in all metaheuristics is how to handle the constraints. In this section we will see how the research tackles these two issues, and also an introduction to the two metaheuristics used.

2.1 Representation

The encoding of the harmonies is always an important decision when designing the HS algorithm. In this paper we apply the Random Keys encoding, which was first described by James C. Bean in [9]. The main reason for using this representation is to use a representation that maintains the feasibility of any solution under any permutation, and also avoids the eventually increased computational load that specialized repair methods or solution discarding approaches would require. Under this representation the resource where an operation is to be processed and the sequence of the operation in that specific resource are jointly described by a single real number. The integer part is used to identify the resource, while the fractional part provides the sequencing of operations in every resource. Thus, taking a simple example with two resources and four operations, the following harmony

$$[1.05][2.86][1.87][2.19]$$

corresponds to the schedule

Resource 1: [Operation 1] - [Operation 3],
Resource 2: [Operation 4] - [Operation 2],

2.2 Initial Solution

Metaheuristic optimisation methods aim at effectively and efficiently exploring the solution domain of a combinatorial optimisation problem so as to refine the search directions and focus on domain areas characterised by better performing solutions, that is areas where the optimum solution is expected to lie. When dealing with the problem of identifying the initial search point across the entire solution space, two main approaches are generally considered. The ancestor, that is the initial solution, can either consist of a randomly generated solution or, alternatively, problem specific logics can be used to generate one. In the latter case, the ancestor is expected to perform better than the randomly generated ancestor as it corresponds to a heuristic solution to the problem. As a

result, using the logical ancestor, the metaheuristic approach starts from a better solution, however, the likelihood of terminating the search at local optima increases.

The performance of the two approaches applied to the problem investigated here is compared in this section. The random initial solution is generated by assigning random feasible resources to each operation while taking all applicable constraints into account. On the other hand, the heuristic based solution is constructed using the Earliest Finish (EF) concept [20]. In the EF heuristics, jobs are first ordered by increasing due date. Then, the heuristic considers one job at a time and tentatively assigns the corresponding operations to eligible resources; the operations are considered one at a time following precedence constraints. The resource able to complete the operation at the earliest time is chosen and its availability is updated accordingly. In order to correctly calculate completion times of an operation, all events limiting the resource availability are considered; these events include previously assigned operations, time dedicated to other tasks, such as training or management meetings, and holidays. A non pre-emptive approach is used to calculate completion times in the sense that an operation cannot be interrupted by another operation; however, operations can be split across non consecutive available time slots to accommodate for non-production related tasks and holidays. This assignment logic is exemplified in Fig. 1 where the operation marked in dark blue (i.e. operation X) is tentatively assigned to resource A, B and C, respectively, whose schedule is partially built. Resource A can complete operation X at time 25; the slot between time 9 and 13 is too short to complete the operation and is bounded by another two operations; hence, operation X is moved to the next available time slot. Resource B can complete operation X at time 20; in this case, the operation is interrupted at time 13 to allow resource B to complete a non-productive task and is resumed soon after. As regards resource C, the operation cannot be interrupted at time 14 since the non-productive task starting at time 14 is followed by a previously assigned operation; hence, the pre-emption assumption would be violated. As a result, resource B is chosen for assignment (Fig. 2).

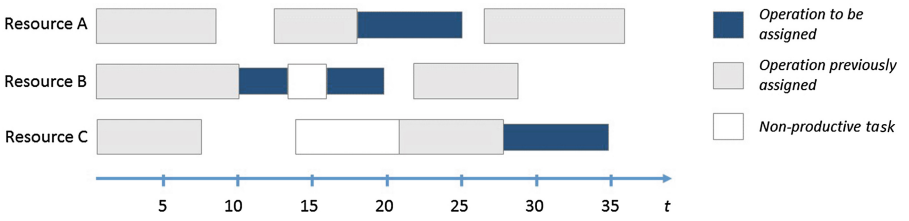

Fig. 2. Tentative assignment of an operation to eligible resources

The assignment algorithm described here can be implemented by using arrays of ordered available time slots for each resource and scanning through them until sufficient cumulative duration is found. When an operation is assigned to a resource, the time slot corresponding to the operation processing is deducted from the array [1].

2.3 Constraint Handling

In cognitive systems, where humans and automated agents share the responsibility of control, system observability and usability are two major concerns. One of the most important characteristics that a system like this must have to overcome those issues is responsiveness. To be able to cooperate, the automated system, must present a solution to a new input from the user in a short time. From an algorithmic point of view, this responsiveness represents a main constraint: the maximum allowed running time before a solution is presented to the user.

As mentioned in the introduction, the time to perform computations is limited by the user attention timespan. In this study we refrain from using constraint handling alternatives that might generate a great number of infeasible solutions, since this would require the evaluation of many invalid solutions. As a result, we won't use penalty functions or repair algorithms, but will rely on in specific operators that will generate only feasible solutions.

Using these operators ensures that every solution evaluated in the simulation represents a feasible schedule, see [3].

In the case of Simulated Annealing the only operator we need is the mutation. The standard mutation operator of genetic algorithms and other similar metaheuristics act by modifying a single value of the solution. In Boolean representations it changes one boolean value, while in real-valued encodings it modifies the value by a small random value.

1. For each chromosome, randomly select one gene to change.
2. For the selected gene, find the subset of designers that are valid.
3. Select a random designer from the subset.

In Harmony Search, to provide a more meaningful comparison, we will use the same mutation operator as Random operator.

2.4 Simulated Annealing

Simulated Annealing was developed by various researchers, in particular Kirkpatrick et al. [4] and V. Černỳ [6], for combinatorial optimization problems, though it has been later extended for its use in continuous optimization.

Mathematically SA is derived from the Metropolis algorithm, developed by Nicholas Metropolis et al. [7]. The main characteristic of Simulated Annealing is that sometimes, in the process of improving the candidate solution, it will accept a successor S with a worse fitness than its parent R, with probability

$$P\left(t, R, S\right) = e^{\frac{fitness(R) - fitness(s)}{t}}, \tag{1}$$

where t represents the temperature in the physical process, and acts as a tunable parameter. When t is high, the probability of accepting a worse solution than

its parent is close to 1, while the probability is almost 0 when t approaches 0. When t is set at a high value at the start of the algorithm, and then allowed to decrease with every iteration, the algorithm will exhibit a explorative behaviour at the start and an exploitative one at the end. An important caveat with SA is that, as we decrease the value of t, the algorithm is more exploitative and more similar to a hill-climbing procedure. This is important in our case, as the limited amount of time to perform computations does not allow for a highly explorative search.

2.5 Harmony Search

Where SA only uses a single solution that evolves over time, Harmony Search [8] uses a set (called harmony memory) of solutions.

Harmony Search has been shown to outperform other meta-heuristic solvers in a wide range of application scenarios such as ground water engineering [10], localization [11], structural optimization [12], radar [13], telecommunications [14], music composition [15], power saving in manufacturing machines [16] and artificial vision [17], among many others. A thorough survey on applications of the HS algorithm can be found in [18].

The algorithm works iteratively by evolving an initial population in the following steps: (1) an initial population of size HMS (Harmony Memory Size) is created by adding random solutions; (2) following a procedure involving the application of three different operators, a new solution is created and evaluated; (3) if this new solution possess a better fitness than the worst solution in the HM, the new solution substitutes the worst one in the population (in this way the population size never changes). The pseudo-code is presented next:

Algorithm 1. Harmony Search pseudocode

```
Initialize the harmony memory with HMS randomly generated solutions
while termination criteria not met do
    create a new solution
    if rnd < HMCR then
        use a value of one of the solutions in the harmony memory (selected uniformly at random)
        if rnd < PAR then
            change this value slightly
        end if
    else
        Generate a random note
    end if
    if new solution is better than the worst solution in the harmony memory then
        replace the worst solution by the new one
    end if
end while
Return the best solution in the harmony memory.
```

The main parameters directing the creation of new solutions are usually only three: the harmony memory size, the Harmony Memory Considering Rate (HMCR) and the Pitch Adjustment Rate (PAR).

The Harmony Memory Considering Rate controls how much information from the harmony memory is used for the generation of a new solution. As such, it controls the rate of convergence of the algorithm. It does that by setting the probability $\varphi \in [0, 1]$ that the new value for a certain note is taken uniformly at random from the values of this same note in all the other solutions in the HM. In case that (with a probability $1 - \varphi$) the memory consideration step is not performed, the new value will be randomly chosen from their alphabet (the set of feasible values).

In the case that HMCR is performed, the probability $\vartheta \in [0, 1]$ will control the application of the Pitch Adjusting Rate. This operator controls the frequency of adjustment of the note selected, by replacing the note by a neighbouring value. In the present paper, this operator acts by swapping the operation in the same resource with one of its neighbours.

2.6 Fitness Function

Three objectives are considered in the fitness function, and we use a weighted sum approach in order to prioritise them. The objectives are (i) the number of late jobs (ii), the summation of all the late days and (iii), the amount of idle time in all resources during the next 7 days, with respective weights of 5, 5 and 1 selected by the users.

Each objective is normalised by using a minimum and a maximum value for each one. In the case of late jobs, the minimum is the number of currently late jobs, while the maximum is taken as the total number of jobs to schedule. The minimum value of late days is taken as the current that each late job is already passed its due date, while the maximum was selected after running multiple simulations and taking the worst value. As for idle time, the minimum value was 0, while the maximum value is the total idle time after removing the length of the current in progress ops (given them time to be finished) and all the time that the designer will be doing other tasks during the next 7 days like holidays, training etc.

3 Computational Results

In this section we will present the results obtained in the computational experiments. These have been carried out in an Intel Core i7-5600 CPU @2.6 GHz, with 12 GB RAM, using only 1 core. All the experiments have been implemented in .NET framework. First we will look at the initialisation approaches, followed by the preliminary tests to find out the best possible parameters for each algorithm in the current scenario. Then we will compare the performance of SA and HS versus a hill climbing procedure.

The assumptions for all the tests are the following:

- Each test will run for 10 s, for the reasons explained in the introduction.
- Each test will be performed 30 times.
- Each run is independent. In particular, a new random seed is used for each run for the random number generator.
- First, for SA we will find the best temperature, by running the tests for different temperatures. Then we will run SA with both a random initial solution and an EF one.
- For HS we will experimentally find good values for HMS, HMCR and PAR.

The data we will use for the experiments is taken directly from the company's ERP[1] system, and represents a normal workload for the design department. The design department team is composed of 28 resources with varying levels of expertise. The work ready to schedule is composed by a total of 409 operations, belonging to 206 jobs. Of these jobs, 76 were already late as the due date was previous to the time the data snapshot was taken, for a total of 468 current days late. There were 23 constraints set by the manager on different operations, of which 6 set a starting time for operations, 1 modifies the run time of an operation and the rest are holidays and training times for the designers.

First we will look at the impact of using a good heuristic to find an initial solution for the algorithms. Table 1 shows the results obtained when using the two initialisation approaches applied to a problem instance; the random solution values are averaged across 100 repetitions. As expected, the EF approach provides a much better solution than a randomly generated ancestor with no significant increase in computational times. In particular, the effect on idle time is very noticeable.

Table 1. Comparison of initialization methods

	Late jobs	Late days	Total idle time (h)
Random	137	1124	264.7
EF	123	700	76.9

Next we present the results of the analysis on the impact of the most important parameters of the presented metaheuristics. In the case of SA, initial temperature in the solution fitness when constraint the computation time to 10 s. In Fig. 3 it is clear that the initial temperature must stay very small in order to perform well. Since the initial temperature is so small, we should expect that SA will perform quite similar to a plain hill climbing algorithm.

As for HS, we begin by performing a set of initial experiments to find a range of good parameters for the harmony search when applied to the current data set.

[1] Enterprise Resource Planning, the information system that manages production, sales, customers etc.

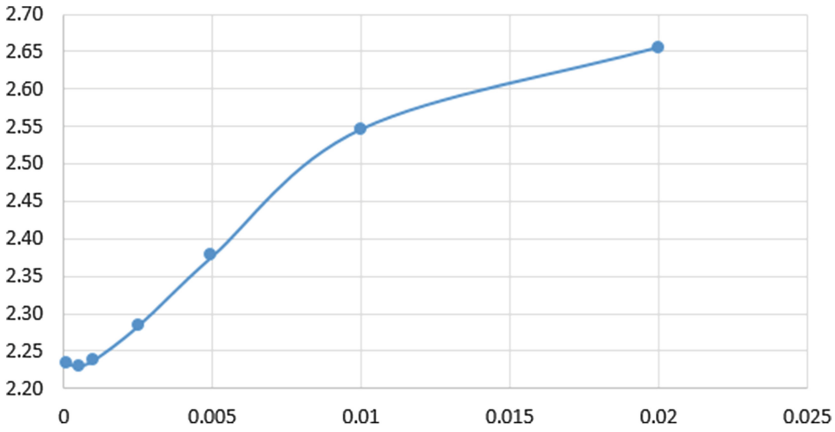

Fig. 3. Effect of SA initial temperature on fitness

In Table 3 the population is set to 10, with 7 random solutions and 3 EF solutions found by the heuristic described in Sect. 2.2. This table shows the relation between HMCR and PAR parameters.

The effect of the harmony memory size is studied in Fig. 4. This figure shows that smaller populations are preferable. The other consequence of varying the HMS is the number of iterations that the algorithm can go through in the fixed span of time, since the larger the population the more solutions need to be evaluated. This relation is shown in Table 2 (Fig. 5).

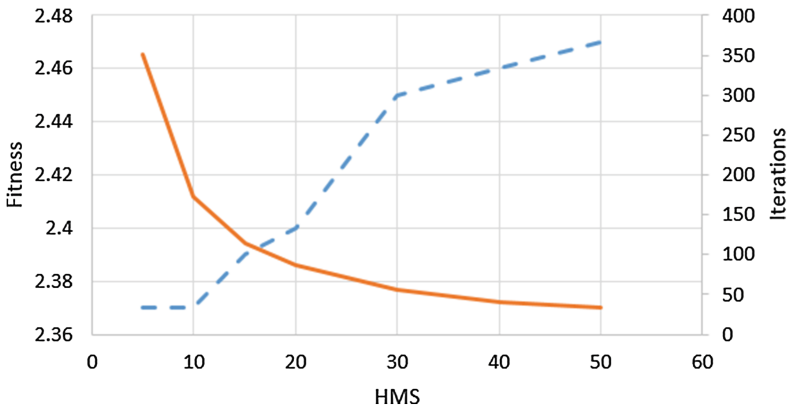

Fig. 4. Harmony memory size

Table 2. HMS vs. iterations

HMS	5	10	15	20	30	40	50
Fitness	2.37	2.37	2.39	2.4	2.45	2.46	2.47
Iterations	351	172	114	86	55	41	33

Table 3. HMCR vs. PAR

PAR	0.003	0.03	0.1	0.2	0.3
0.999	2.39	2.38	2.46	2.49	2.54
0.99	2.46	2.45	2.51	2.56	2.58
0.9	2.75	2.76	2.76	2.76	2.76

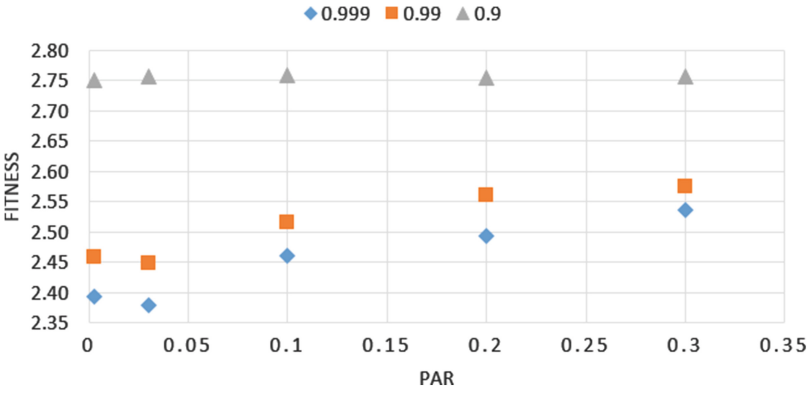

Fig. 5. HMCR vs. PAR

After these initial experiments, the parameters are thus set to:

- Simulated Annealing: $temperature = 0.0005$
- Harmony Search
  - $HMS = 10$
  - $HMCR = 0.999$
  - $PAR = 0.03$

With these parameters set, we run each algorithm 30 times, each time running for 10 s. The results in Table 4 are those obtained when running each algorithm with a Earliest Finish initial solution (3 EF solutions were used together with 7 random ones in case of the HS memory size), and Table 5 shows the results when the initial solution is a random one (100% random initial solutions in the HS memory).

As expected, the performance of simulated annealing when initial temperature is set to a very small value is very similar to that of the hill climbing procedure. Such initial temperature does not allow for a wide exploration of the solution space, and instead the algorithm mostly exploits the initial solution.

Table 4. EF initial solution

	Avg. fitness	Avg. late jobs	Avg. late days	Avg. idle time (h)
HC	2.22	117	692	107.1
SA	2.24	117	693	107.8
HS	2.38	118	695	142.1

Table 5. Random initial solution

	Avg. fitness	Avg. late jobs	Avg. late days	Avg. idle time (h)
HC	2.98	118	792	131.7
SA	3.05	118	799	142.3
HS	4.44	133	927	213.5

A follow-on experiment was performed, where 30 runs of simulated annealing were run with the stopping criterion of reaching a solution with the same fitness as that of the average hill climbing (2.22), in order to have an order of magnitude of the necessary time for SA. After 30 runs of SA, the average running time for reaching such fitness was 14.3 s (43% greater than the necessary time for the hill climbing).

4 Conclusions

The research presented in this paper aims at improving the quality of the production scheduling in real-life manufacturing scenarios. In those cases, where decision support systems need to react to unseen changes and in particular provide quick feedback to human users, the computational time becomes a limited resource. The experiments presented here show how an appropriate use of a good initialisation heuristic, like the earliest Finish approach presented in this paper, followed by an exploitation of the initial solution using simple hill climbing outperforms the use of more complex metaheuristics like Simulated Annealing or Harmony Search.

A following extension of this research will be the study of different local search strategies to work in conjunction with the initialisation heuristic to improve the quality of the solutions, always in scenarios where the time allowed for computations is limited by real-life user experience factors.

Acknowledgements. This work has been supported by Enterprise Ireland through the Innovation Partnership program.

References

1. Alcaraz, J., Maroto, C., Ruiz, R.: Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. J. Oper. Res. Soc. **54**(6), 614–626 (2003)

2. Tiwari, V., Patterson, J.H., Mabert, V.A.: Scheduling projects with heterogeneous resources to meet time and quality objectives. Eur. J. Oper. Res. **193**(3), 780–790 (2009)
3. Michalewicz, Z., Janikow, C.Z.: Handling constraints in genetic algorithms. In: ICGA, pp. 151–157 (1991)
4. Kirkpatrick, S.: Optimization by simulated annealing: quantitative studies. J. Stat. Phys. **34**(5–6), 975–986 (1984)
5. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1999)
6. Cerny, V.: Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. J. Optim. Theor. Appl. **45**(1), 41–51 (1985)
7. Metropolis, N., et al.: Equation of state calculations by fast computing machines. J. Chem. Phys. **21**(6), 1087–1092 (1953)
8. Geem, Z.W., Kim, J.-H., Loganathan, G.V.: A new heuristic optimization algorithm: harmony search. Simulation **76**(2), 60–68 (2001)
9. Bean, J.C., Norman, B.A.: Random keys for job shop scheduling (1993)
10. Ayvaz, M.T.: Simultaneous determination of aquifer parameters and zone structures with fuzzy c-means clustering and meta-heuristic harmony search algorithm. Adv. Water Resour. **30**(11), 2326–2338 (2007)
11. Manjarres, D., Del Ser, J., Gil-Lopez, S., Vecchio, M., Landa-Torres, I., Lopez-Valcarce, R.: A novel heuristic approach for distance-and connectivity-based multihop node localization in wireless sensor networks. Soft Comput. **17**(1), 17–28 (2013)
12. Lee, K.S., Geem, Z.W., Lee, S.H., Bae, K.W.: The harmony search heuristic algorithm for discrete structural optimization. Eng. Optim. **37**(7), 663–684 (2005)
13. Gil-Lopez, S., Del Ser, J., Salcedo-Sanz, S., Perez-Bellido, A.M., Cabero, J.M., Portilla-Figueras, J.A.: A hybrid harmony search algorithm for the spread spectrum radar polyphase codes design problem. Exp. Syst. Appl. **39**(12), 11089–11093 (2012)
14. Del Ser, J., Matinmikko, M., Gil-Lopez, S., Mustonen, M.: Centralized and distributed spectrum channel assignment in cognitive wireless networks: a harmony search approach. Appl. Soft Comput. **12**(2), 921–930 (2012)
15. Geem, Z.W., Choi, J.Y.: Music composition using harmony search algorithm. In: Workshops on Applications of Evolutionary Computation, pp. 593–600 (2007)
16. Garcia-Santiago, C.A., Del Ser, J., Upton, C., Quilligan, F., Gil-Lopez, S., Salcedo-Sanz, S.: A random-key encoded harmony search approach for energy-efficient production scheduling with shared resources. Eng. Optim. **47**(11), 1481–1496 (2015)
17. Fourie, J., Mills, S., Green, R.: Harmony filter: a robust visual tracking system using the improved harmony search algorithm. Image Vision Comput. **28**(12), 1702–1716 (2010)
18. Manjarres, D., Landa-Torres, I., Gil-Lopez, S., Del Ser, J., Bilbao, M.N., Salcedo-Sanz, S., Geem, Z.W.: A survey on applications of the harmony search algorithm. Eng. Appl. Artif. Intell. **26**(8), 1818–1831 (2013)
19. Microsoft Canada: Attention Spans. https://advertising.microsoft.com/en/WWDocs/User/display/cl/researchreport/31966/en/microsoft-attention-spans-research-report.pdf
20. Rotondo, A., Young, P., Geraghty, J.: EF-based strategies for productivity improvements at wet-etch stations. Prod. Plann. Control **25**(11), 958–968 (2014)