Operations Research Perspectives 2 (2015) 97–105

Contents lists available at ScienceDirect

Operations Research Perspectives

journal homepage: www.elsevier.com/locate/orp

A critical analysis of the harmony search algorithm—How not to solve sudoku

Dennis Weyland *

Università della Svizzera italiana, Lugano, Switzerland
Department of Economics and Management, University of Brescia, Italy

ARTICLE INFO

Article history:
Available online 30 April 2015

Keywords:
Heuristics
Metaheuristics
Harmony search
Evolution strategies

ABSTRACT

This article presents a critical analysis of the harmony search metaheuristic framework. We formally prove that the harmony search algorithm is a special case of evolution strategies. First, this implies that the harmony search algorithm itself does not offer any novelty, apart from using a different terminology. Second, the performance of the best harmony search algorithm is always bounded by the performance that can be obtained by evolution strategies. Additionally, more than a decade of research about harmony search has not revealed any other sort of novelty or has led to any new insights or significant contributions in the field of heuristics. In short, there is no reason for harmony search to exist as a separate metaheuristic framework.

Based on these findings, we carefully examine the results found in the paper *Harmony search algorithm for solving sudoku*. A theoretical investigation and a reimplementation of the harmony search algorithm both reveal that these results are fundamentally flawed.

1. Introduction

In recent years a huge number of novel metaheuristics were proposed. These metaheuristics are usually based on metaphors describing natural processes or social phenomena. The metaphors used to derive the working mechanisms of such novel metaheuristics are getting increasingly absurd and the connection between the metaphors on the one hand and optimization on the other hand is getting increasingly vague. It is not really clear what the flow of water [1], the leaps of frogs [2] or a salmon run [3] have to do with optimization. Additionally, it seems that the underlying working mechanisms of these methods are very similar, and in some cases even identical, to those of well-established heuristics. For example, the differences between the particle swarm optimization metaheuristic [4] and "novel" metaheuristics like the firefly algorithm [5], the fruit fly optimization algorithm [6], the fish swarm optimization algorithm [7] or the cat swarm optimization algorithm [8] seem negligible. Nevertheless, the literature is full of results which certify exceptional performance to these "novel" methods. Obviously, there is something going wrong. This whole development had been ignored for quite a while, but recently open criticism has emerged. In [9,10] the harmony search algorithm, one of those "novel" metaheuristics, was identified as a special case of evolution strategies [11], while general criticism to this questionable development was raised in [12].

In this work we once again focus on the harmony search algorithm. It is not really fair to single out this specific method, but it serves very well as a representative for the whole set of recently proposed metaphor-based metaheuristics and it is a perfect example of what is exactly going wrong. The harmony search algorithm has been proposed in 2001 by Geem [13]. In its basic version, this algorithm is a heuristic method for solving discrete combinatorial optimization problems. The working mechanisms of this algorithm were designed in analogy to jazz music. Here the author has identified jazz music as an optimization process, where the different musicians that are playing together are optimizing harmonies over time. The number of publications about harmony search is growing in an enormous speed. In 2010 a google scholar search for "harmony search" (including the quotation marks) showed around 500 results [9]. Two years later, in 2012, the same search showed already 3000 results [12] and while writing this article, at the end of 2014, the number of results exceeded 9000. The majority of publications deals with applications of harmony search in different areas, such as power flow control [14], flow shop problems [15], optimization of truss structures [16], design of water distribution networks [17], orienteering problems [18], scheduling of dam systems [19], solving sudoku [20], design of steel sway frames [21], design of geodesic domes [22], and many more. Other publications

* Correspondence to: Università della Svizzera italiana, Lugano, Switzerland.
E-mail address: dennisweyland@gmail.com.

http://dx.doi.org/10.1016/j.orp.2015.04.001

focus on enhancements over the algorithm's basic version [23,24,14,25,26], on hybridizations with other metaheuristics [27,28,15,16,29] and on theoretical results regarding the harmony search algorithm [30]. Even a few textbooks on harmony search have been published in recent years [31–34]. At a first glance, harmony search seems to be a novel and extremely successful metaheuristic, but looking at it more carefully, several doubts have to be raised.

Such doubts about harmony search were first formalized in an article in 2010 [9]. In that article harmony search was identified as a special case of evolution strategies [11], a heuristic that had existed already for many decades before the harmony search algorithm was proposed. It is interesting to note, that the same results were independently shown for a special case of the harmony search algorithm two years later in [10]. Some conceptual problems with empirical investigations of the harmony search algorithm for solving the design of water distribution networks were revealed in [35] and it seems that these issues generalize to many of the other applications of the harmony search algorithm. As a response to [9], the founder of the harmony search algorithm published a research commentary [36]. This research commentary was called "less than fully convincing" in [12]. One of the main issues raised in the research commentary is that the results in [9] had not been formally proven. Therefore, we present a formal proof of those results in full detail and with full rigor in Section 2. Based on these findings, we then carefully examine the results reported in the publication *Harmony search algorithm for solving sudoku* [20] in Section 3. This includes a reimplementation of the algorithm followed by empirical studies, as well as a theoretical analysis of the algorithm. These investigations reveal that the results reported in [20] are fundamentally flawed. Finally, Section 4 concludes the paper with a thorough discussion of the results and their implications.

2. Harmony search *really* is a special case of evolution strategies

In the 2010 article [9] the harmony search algorithm was identified as a special case of evolution strategies, without doubt one of the most classic search heuristics. Already in the same year the founder of the harmony search algorithm published a research commentary [36] as a reaction to these results. Due to the issues raised in the research commentary and due to the growing number of publications in harmony search, it is obvious that many researchers are not aware of the results presented in [9], ignore them or just do not accept them. One reason for this issue could be the informal presentation of the results in [9], which was also criticized in the research commentary. Therefore, we will derive these results in a strictly formal way, following the precise mathematical structure of theorem and proof.

In the remainder of this section we will derive the results for certain discrete combinatorial optimization problems, but it will be clear that the same arguments can be used for obtaining an identical result in a more general context or for the continuous case. We have given an n-dimensional search space $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_n$ and an objective function $f : \mathcal{X} \to \mathbb{R}$. Here the sets $\mathcal{X}_i$, $1 \leq i \leq n$, are intervals of integers with smallest elements $l_i \in \mathbb{N}$ and largest elements $u_i \in \mathbb{N}$. The goal is to find some $x \in \mathcal{X}$ such that $f(x) \leq f(x')$ holds for every $x' \in \mathcal{X}$. Such an element is called a global optimum or just an optimal solution. We will now discuss how the harmony search algorithm and evolution strategies tackle this kind of optimization problems.

2.1. The harmony search algorithm

The basic version of the harmony search algorithm for discrete optimization problems has three parameters, the harmony memory size HMS, the harmony memory consideration rate HMCR and the pitch adjustment rate PAR. Additionally, a maximum number of iterations is given. In the first step, HMS many solutions $x^{(1)}, x^{(2)}, \ldots, x^{(\mathrm{HMS})}$ are created uniformly at random. Then iteratively new solutions are generated until the maximum number of iterations has been reached. The solutions are generated in the following way. Here the same process is applied independently to each of the n decision variables. With a probability of HMCR a step called memory consideration is performed. The value for a decision variable is taken uniformly at random from the corresponding values of the solutions stored in the harmony memory. A memory consideration step is followed by a pitch adjustment step with a probability of PAR. With a probability of 1/2 the pitch adjustment increments the decision variable by 1, otherwise it decrements the decision variable by 1. In case the interval bounds are crossed, the decision variable keeps its previous value. In case no memory consideration step is performed, that means with a probability of 1 − HMCR, a random selection step is performed instead. This step just assigns to the decision variable a value uniformly at random. After generating the new solution, it is evaluated by the given objective function. If it is better than the worst solution in the harmony memory, the worst solution is replaced by the new one. After the maximum number of iterations has been performed, the best solution from the harmony memory is returned. The pseudo code for this algorithm is shown in Algorithm 1.

Algorithm 1 The harmony search algorithm

```
initialize the harmony memory with HMS randomly generated solutions
repeat
    create a new solution in the following way
    for all decision variables do
        with probability HMCR use a value of one of the solutions
        in the harmony memory (selected uniformly at random)
        and additionally change this value slightly with probability PAR
        otherwise (with probability 1 - HMCR) use a random value
        for this decision variable
    end for
    if the new solution is better than the worst solution in the
    harmony memory then
        replace the worst solution by the new one
    end if
until the maximum number of iterations has been reached
return the best solution in the harmony memory
```

2.2. Evolution strategies

At this point we will not discuss evolution strategies in general, but focus on the specific variant which will be used in the analysis, the so called $(\mu + 1)$ evolution strategy [11]. During the initialization μ individuals are created uniformly at random. Then, iteratively new solutions are generated until the maximum number of iterations has been reached. The new solutions are created by using so called recombination and mutation operators. If the newly generated solution is better than the worst solution in the population, the worst solution is replaced by the new one. When the maximum number of iterations has been reached, the best solution from the population is returned. The pseudo code for this algorithm is shown in Algorithm 2.

This is a very generic method and we still have to specify the recombination and mutation operators that are used. For our analysis we selected the global discrete recombination operator

Algorithm 2 $(\mu + 1)$ evolution strategy

1: initialize the population with μ randomly generated solutions
2: **repeat**
3: create a new solution using recombination and mutation operators
4: **if** the new solution is better than the worst solution in the population **then**
5: replace the worst solution by the new one
6: **end if**
7: **until** the maximum number of iterations has been reached
8: **return** the best solution in the harmony memory

[37–39]. Here each decision variable of the new solution uses the value of the corresponding decision variable of a randomly selected solution in the current population. This operator is always applied. On top of that, two mutation operators are applied. The first one is applied to each decision variable independently with a probability of p_1. With a probability of 1/2 this operator increases the value of the decision variable by 1, otherwise it decreases the value of the decision variable by 1. If the interval borders are crossed, the variable is set to its previous value. The second mutation operator is applied to each decision variable independently with a probability of p_2. This operator just sets the decision variable to a value uniformly at random within the given interval. Both operators are very common mutation operators [11,40–42,37,43]. The resulting algorithm is shown in Algorithm 3.

Algorithm 3 $(\mu + 1)$ evolution strategy with details about the recombination and mutation operators

1: initialize the population with μ randomly generated solutions
2: **repeat**
3: create a new solution in the following way
4: **for all** decision variables **do**
5: select a solution from the population uniformly at random and set the decision variable to the corresponding value of the selected solution
6: with probability p_1 change the value slightly
7: with probability p_2 change the value to a random one
8: **end for**
9: **if** the new solution is better than the worst solution in the population **then**
10: replace the worst solution by the new one
11: **end if**
12: **until** the maximum number of iterations has been reached
13: **return** the best solution in the harmony memory

2.3. A formal proof

Now it remains to formally proof that the harmony search algorithm is a special case of evolution strategies. For this purpose we will follow the slightly informal argumentation of the 2010 article [9], but this time with the full rigor of a formal proof.

Theorem 1. *The harmony search algorithm is a special case of evolution strategies. Moreover, the recombination and mutation operators used in this proof are common choices for such operators.*

Proof. What we will show here is that the harmony search algorithm as depicted in Algorithm 1 is a special case of the $(\mu+1)$ evolution strategy as depicted in Algorithm 3. One could say that this relationship is obvious by looking at the pseudo codes of these two algorithms, one could also say that the slightly informal argumentation of [9] is sufficient, but we better be careful not to get deceived by jumping to conclusions.[1]

What remains to do, is to show that for each parameter setting (HMS, HMCR, PAR) for the harmony search algorithm, we can give a parameter setting (μ, p_1, p_2) for the $(\mu + 1)$ evolution strategy, such that the resulting two algorithms are equivalent. Here equivalent means that the probability distribution over the solutions returned by the two algorithms are identical. This is a very result oriented definition of equivalence among search heuristics. There a certainly other possible definitions of equivalence, but for our purposes this result oriented definition seems the most appropriate.

By setting $\mu = \text{HMS}$ we immediately see that the initialization phases of the two algorithms (apart from differences in terminology) are identical. Both algorithms create a pool of possible solutions and the probability distributions over these pools are identical for both algorithms. They then maintain this pool throughout their whole running time and never alter its size. They both execute a loop for the same number of iterations and finally return the best solution of their solution pool. If there is a difference between the two methods, it has to be within the main loop (steps 2 until 11 in Algorithm 1 and steps 2 until 12 in Algorithm 3). We can now use the solution pool as a kind of invariance property for our analysis. We will show that, given a probability distribution over possible solution pools of size $\mu = \text{HMS}$, one iteration of the main loop of both algorithms leads to the same probability distribution over solution pools. Using induction and the fact that the probability distributions are the same after the initialization step, we are done.

So let $x^{(1)}, x^{(2)}, \ldots, x^{(\mu)}$ be random variables for the solutions in the pool at the beginning of an iteration. Since the selection step is the same for both algorithms, it is in fact sufficient to show that the newly generated solutions are drawn from identical probability distributions for both algorithms. Since the decision variables of these solutions are handled separately, it is already sufficient to show that for a given decision variable the probability distribution over the values in the newly generated solution are identical for both algorithms.

So let us focus on the decision variable with index i. We use $\mathcal{P}_{HS}(x'_i = j)$ and $\mathcal{P}_{ES}(x'_i = j)$ to denote the probability that the decision variable with index i of the newly generated solution has the value j for the harmony search algorithm and the $(\mu + 1)$ evolution strategy, respectively. We have to be a bit careful in the case where j coincides with one of the interval borders, but we will skip the corresponding details at this point.

$$\begin{aligned}
\mathcal{P}_{HS}\left(x'_i = j\right) &= \sum_{k=1}^{\mu} 1/\mu \cdot \text{HMCR} \cdot \text{PAR} \cdot 1/2 \cdot \mathcal{P}\left(x^{(k)} = j-1\right) \\
&\quad + \sum_{k=1}^{\mu} 1/\mu \cdot \text{HMCR} \cdot \text{PAR} \cdot 1/2 \cdot \mathcal{P}\left(x^{(k)} = j+1\right) \\
&\quad + \sum_{k=1}^{\mu} 1/\mu \cdot \text{HMCR} \cdot (1-\text{PAR}) \cdot \mathcal{P}\left(x^{(k)} = j\right) \\
&\quad + (1-\text{HMCR}) \cdot \frac{1}{u_i - l_i + 1} \\
\mathcal{P}_{ES}\left(x'_i = j\right) &= \sum_{k=1}^{\mu} 1/\mu \cdot p_1 \cdot (1-p_2) \cdot 1/2 \cdot \mathcal{P}\left(x^{(k)} = j-1\right) \\
&\quad + \sum_{k=1}^{\mu} 1/\mu \cdot p_1 \cdot (1-p_2) \cdot 1/2 \cdot \mathcal{P}\left(x^{(k)} = j+1\right)
\end{aligned}$$

[1] At this point we feel obliged to apologize for the fact that we could not find an appropriate way for highlighting ironic passages. In any case, these passages should not be too difficult to identify.

	5		3		6			7
				8	5		2	4
	9	8	4	2		6		3
9		1			3	2		6
	3						1	
5		7	2	6		9		8
4		5		9		3	8	
	1		5	7				2
8			1		4		7	

(a) The sudoku puzzle from [20].

2	5	4	3	1	6	8	9	7
7	6	3	9	8	5	1	2	4
1	9	8	4	2	7	6	5	3
9	8	1	7	5	3	2	4	6
6	3	2	8	4	9	7	1	5
5	4	7	2	6	1	9	3	8
4	7	5	6	9	2	3	8	1
3	1	9	5	7	8	4	6	2
8	2	6	1	3	4	5	7	9

(b) The unique solution to the sudoku puzzle.

Fig. 1. An example of a sudoku puzzle and its unique solution.

$$+ \sum_{k=1}^{\mu} 1/\mu \cdot (1-p_1) \cdot (1-p_2) \cdot \mathcal{P}\left(x^{(k)} = j\right)$$
$$+ p_2 \cdot \frac{1}{u_i - l_i + 1}.$$

For $p_1 = \text{PAR}$ and $p_2 = 1 - \text{HMCR}$ we have

$$\mathcal{P}_{HS}\left(x_i' = j\right) = \mathcal{P}_{ES}\left(x_i' = j\right).$$

Since i and j have been arbitrary, this finally concludes our proof. □

To avoid misunderstandings as in the research commentary [36], we would like to briefly comment this result. Theorem 1 states that the harmony search algorithm is a special case of evolution strategies. Just to make it clear, this does *not* mean that these two methods are equivalent. Instead, the theorem states that for each harmony search algorithm, there exists an evolution strategy which produces the same results. This means that evolution strategies are a superset of the harmony search algorithm. In fact, it is easy to see that this is a proper inclusion, which implies that there are evolution strategies for which no equivalent harmony search algorithm exists. Therefore, the only novelty of the harmony search algorithm is its metaphor, which can hardly be seen as an argument for more than 9000 publications. Apart from that, Theorem 1 has also implications about the performance of harmony search. Since harmony search is a special case of evolution strategies, the best harmony search algorithm can never outperform the best evolution strategy. Therefore, it is only possible to conclude that the many publications about successful applications of harmony search contain serious flaws from a conceptual point of view.

3. How not to solve sudoku

This section contains a detailed investigation of one selected harmony search publication. The idea behind this investigation was basically to reproduce the results reported in some of the numerous harmony search publications. At the end, the work selected for this section was the 2007 publication *Harmony search algorithm for solving sudoku* [20] by Geem, the founder of the harmony search algorithm. This particular publication was selected due to the simplicity of the problem under investigation and due to low implementation efforts for repeating the experiments.

In the above mentioned publication, the harmony search algorithm is applied to solve the famous sudoku puzzle [44]. We have given a square of size 9×9, whose cells are either empty or contain numbers in the range between 1 and 9. The goal is to fill the empty cells using the numbers between 1 and 9, such that each of these numbers are used exactly once in each of the 9 rows, in each of the 9 columns and in each of the 9 squares of size 3×3 (in which the original square can be uniquely partitioned). The initial numbers are usually set in a way such that there exists a unique solution to the problem. An example of such an instance of the sudoku puzzle together with its unique solution is given in Fig. 1. In fact, this is the instance which was used in the experiments of [20] and which will be used throughout this section as well.

Before we continue with our investigation, we would like to mention that such puzzles can usually be solved within a fraction of a second using problem specific algorithms [44]. It is therefore highly questionable, why a heuristic should be applied to this problem at all. Although this seems to be a major conceptual issue, we will not go further into detail here. In fact, the following subsections will reveal shocking facts which overshadow this conceptual issue by far.

3.1. The objective function

It is clear that a potential solution for the sudoku puzzle can be represented by a 9×9 matrix $x \in \{1, 2, \ldots, 9\}^{9\times 9}$. For $1 \leq i \leq 9$ and $1 \leq j \leq 9$ let $x_{i,j}$ denote the value at row i and column j of x. For a solution to be valid, we additionally require that the values coincide with the initial values at their corresponding locations. In [20] the following objective function for the sudoku puzzle has been proposed.

$$f(x) = \sum_{i=1}^{9} \left| \sum_{j=1}^{9} x_{i,j} - 45 \right| + \sum_{j=1}^{9} \left| \sum_{i=1}^{9} x_{i,j} - 45 \right| + \sum_{k=1}^{9} \left| \sum_{(i,j)\in B_k} x_{i,j} - 45 \right|.$$

Here the sets B_k, $1 \leq k \leq 9$ contain the coordinates of the cells for the subsquares of size 3×3. In words, we sum the absolute differences of 45 and the sums of values in each row, in each column and in each subsquare. Since the sum of the values $1, 2, \ldots, 9$ is 45, it is clear that the unique solution to the sudoku puzzle has an objective value of 0 and is therefore an optimal solution with respect to the given objective function. Unfortunately, it is not obvious whether there are other optimal solutions with respect to the given objective function or not. In [20] the author states the following regarding this issue:

> "It should be noted that, although the sum of each row, each column, or each block equals 45, it does not guarantee that the numbers 1 through 9 are used exactly once. However, any violation of the uniqueness affects other row, column, or block which contains the wrong value jointly".

2	5	4	3	1	6	8	9	7
7	6	3	9	8	5	1	2	4
1	9	8	4	2	7	6	5	3
9	8	1	7	5	3	2	4	6
6	3	2	8	4	9	7	1	5
5	4	7	2	6	1	9	3	8
4	7	5	6	9	2	3	8	1
3	1	9	5	7	8	4	6	2
8	2	6	1	3	4	5	7	9

(a) The unique solution to the sudoku puzzle.

3	5	3	3	1	6	8	9	7
6	6	4	9	8	5	1	2	4
1	9	8	4	2	7	6	5	3
9	8	1	7	5	3	2	4	6
6	3	2	8	4	9	7	1	5
5	4	7	2	6	1	9	3	8
4	7	5	6	9	2	3	8	1
3	1	9	5	7	8	4	6	2
8	2	6	1	3	4	5	7	9

(b) A solution with optimal objective value.

4	5	2	3	1	6	8	9	7
5	6	5	9	8	5	1	2	4
1	9	8	4	2	7	6	5	3
9	8	1	7	5	3	2	4	6
6	3	2	8	4	9	7	1	5
5	4	7	2	6	1	9	3	8
4	7	5	6	9	2	3	8	1
3	1	9	5	7	8	4	6	2
8	2	6	1	3	4	5	7	9

(c) A solution with optimal objective value.

5	5	1	3	1	6	8	9	7
4	6	6	9	8	5	1	2	4
1	9	8	4	2	7	6	5	3
9	8	1	7	5	3	2	4	6
6	3	2	8	4	9	7	1	5
5	4	7	2	6	1	9	3	8
4	7	5	6	9	2	3	8	1
3	1	9	5	7	8	4	6	2
8	2	6	1	3	4	5	7	9

(d) A solution with optimal objective value.

Fig. 2. The unique solution to the given sudoku puzzle and three alternative solutions, which also obtain the optimal objective value of 0. Cells in the alternative solutions which differ from the unique solution to the sudoku puzzle are highlighted.

This statement itself is true, but does it really explain that there is only one optimal solution with respect to the given objective function? It turns out that this is not true in general. In fact, lets look at the entries $x_{1,1}, x_{1,3}, x_{2,1}$ and $x_{2,3}$. The corresponding values of the unique solution to the sudoku puzzle are $x_{1,1} = 2, x_{1,3} = 4$, $x_{2,1} = 7$ and $x_{2,3} = 3$. By increasing the values of $x_{1,1}$ and $x_{2,3}$ by 1 and decreasing the values of $x_{1,3}$ and $x_{2,1}$ by 1 we obtain another optimal solution with an objective value of 0, which is not the unique solution to the sudoku puzzle. This step can even be repeated two more times. The unique solution to the sudoku puzzle is shown in Fig. 2 together with the additional three optimal solutions that have been just generated. Apart from that, there are many more possibilities to derive other optimal solutions from a given optimal solution. Therefore, we cannot assume that the unique solution to the given sudoku puzzle is the *only* optimal solution with respect to the given objective function. Quite the contrary, it seems that in general there exists a huge number of optimal solutions. But this also implies that an optimal solution returned by some heuristic is not necessarily the unique solution of the given sudoku puzzle.

It seems very likely that the author if [20] was not aware of this issue. He often uses terms as "the optimal solution" or "the global optimum", which both imply uniqueness of the solution with respect to the given objective function. It is therefore also not clear if the results reported in [20] are with respect to finding the unique solution of the sudoku puzzle or with respect to finding any of the optimal solutions. At least in the illustrative example, the author shows the evolution of the best solution found by the harmony search algorithm, which finishes by reaching the unique solution of the sudoku puzzle. In any case, these findings regarding the objective function strongly motivate an independent repetition of the experiments. But before we turn to this issue, we would like to investigate the results reported in [20] from a more theoretical point of view.

3.2. Theoretical investigations

In this section we would like to investigate the behavior of the proposed harmony search algorithm for the sudoku puzzle from a more theoretical point of view. More in detail, we will derive bounds for the probability that the harmony search algorithm finds the unique solution to the sudoku puzzle within a given number of iterations. For this purpose we first bound the probability that the unique solution to the sudoku puzzle is obtained during the initialization process. After that, we bound the probability that the unique solution to the sudoku puzzle is obtained in one iteration of the proposed harmony search algorithm. Here we make use of the facts that solutions are generated in a stochastic process and that the amount of randomness for any newly generated solution is quite substantial.

During the initialization process, a certain number of solutions is generated uniformly at random and stored in the harmony memory. The exact number of generated solutions is specified by the harmony memory size. The sudoku puzzle under investigation contains 41 empty cells, and therefore the probability to generate the unique solution equals $(1/9)^{41}$. Let HMS denote the harmony memory size. The probability that at least one of the initially generated solutions is in fact the unique solution to the sudoku puzzle is then $1-[1-(1/9)^{41}]^{\mathrm{HMS}}$. For all the different HMS values of 1, 2, 10 and 50, this term can be bounded from above by 10^{-37}.

That means it is quite unlikely to obtain the unique solution to the sudoku puzzle already during the initialization.

Let us now focus on the probability to generate the unique solution in one iteration of the harmony search algorithm. To obtain some useful bounds here, it is crucial to observe that any newly generated solution is substantially random. This is due to the two mechanisms called random selection and pitch adjustment, combined with the fact that the values are generated separately for each of the decision variables. In the case of random selection, the value of a decision variable is set uniformly at random to a value between 1 and 9. This mechanism is used with a probability of 1 − HMCR. Otherwise, that means with a probability of HMCR, a value is taken from the harmony memory. This value is then further modified by pitch adjustment, which is used with a probability of PAR. The pitch adjustments change the value with a probability of 1/2 to one of the two neighbor values (or in case where only 1 neighbor value exists, it remains unchanged with probability 1/2 and is changed to the unique neighbor value otherwise). The probability to generate the correct value for one decision variable can therefore be bounded from above by

$$(1 - \text{HMCR}) \cdot 1/9 + \text{HMCR} \cdot (1 - \text{PAR}) + \text{HMCR} \cdot \text{PAR} \cdot 1/2.$$

The first term corresponds to the probability of obtaining the correct value by random selection. The second term corresponds to memory consideration without pitch adjustment, where we generously assume to select the correct value. The last term corresponds to memory consideration, followed by pitch adjustment, where we generate the correct value with a probability of at most 1/2. Please note, that this probability is independent of the solutions stored in the harmony memory.

To get the probability that all the decision variables of the newly generated solution coincide with the unique solution to the sudoku puzzle, we have to further exponentiate the whole term with the number of decision variables, which is 41 in this case. The success probability for a single iteration can therefore be written as

$$p_1 \leq ((1 - \text{HMCR}) \cdot 1/9 + \text{HMCR} \cdot (1 - \text{PAR}) + \text{HMCR} \cdot \text{PAR} \cdot 1/2)^{41}.$$

Using this value, we can now derive an upper bound for the success probability for a given number of i iterations.

$$\begin{aligned} p_i &= 1 - [1 - p_1]^i \\ &\leq 1 - \left[1 - ((1 - \text{HMCR}) \cdot 1/9 + \text{HMCR} \cdot (1 - \text{PAR}) + \text{HMCR} \cdot \text{PAR} \cdot 1/2)^{41}\right]^i. \end{aligned}$$

Combining this bound with the probability that we obtain the unique solution during the initialization and setting the maximum number of iterations to 10^4, we get the following upper bound for the success probability of the harmony search algorithm proposed in [20].

$$10^{-37} + 1 - \left[1 - ((1 - \text{HMCR}) \cdot 1/9 + \text{HMCR} \cdot (1 - \text{PAR}) + \text{HMCR} \cdot \text{PAR} \cdot 1/2)^{41}\right]^{10000}.$$

Table 1 shows the upper bounds for the success probabilities of the experiments conducted in [20]. Please note that these bounds are the same for all the different values of the harmony memory size used in the experiments. Although this bound is far from being tight, the success probabilities are extremely low for most of the parameter settings. The success probability is smaller for low values of HMCR and high values of PAR. In fact, for a HMCR value of 0.9 combined with a PAR value of 0.01 and 0.1 we could not get any useful bound with this approach. Ironically, in [20] the worst results are reported for these parameter settings.

Looking more in detail at the results of [20], we can see that all the 12 runs with a HMCR parameter of 0.5 were successful and that the number of iterations until the solution was found is rather low for all of these runs. Using our bounds for the success probability, we can bound the probability that all of these 12 runs are successful within the maximum number of iterations by

$$\left(2.84210^{-7}\right)^4 \cdot \left(5.17610^{-8}\right)^4 \cdot \left(9.89110^{-12}\right)^4 \leq 4.8310^{-100},$$

an astronomically small probability. To get a feeling for how small this probability really is, we would like to refer to the number of atoms in the visible universe, which is commonly estimated to be around 10^{80}. This further motivates a repetition of the experiments of [20].

Table 1
Upper bounds for the probability that the harmony search algorithm finds the unique solution to the sudoku puzzle within 10^4 iterations.

HMCR	PAR	Success probability (upper bound)
0.5	0.01	2.84210^{-7}
	0.1	5.17610^{-8}
	0.5	9.89110^{-12}
0.7	0.01	2.43710^{-2}
	0.1	4.03410^{-3}
	0.5	4.19510^{-7}
0.9	0.01	1.000
	0.1	1.000
	0.5	1.95810^{-3}

3.3. Empirical investigations

As we said before, the issues about the objective function and the results of our theoretical investigations strongly motivate a repetition of the experiments performed in [20]. For this purpose, we reimplemented the corresponding algorithm. The experiments are performed using the same parameters as in [20]. More in detail, the parameters are all combinations of the four values 1, 2, 10 and 50 for the harmony memory size, the three values 0.5, 0.7 and 0.9 for the harmony memory consideration rate and the three values 0.01, 0.1 and 0.5 for the pitch adjustment rate. This gives a total of 36 different parameter settings. For each parameter settings we perform 20 independent runs, which gives a total of 720 runs. The maximum number of iterations is set to 10^6 (instead of the 10^4 used in [20]). We then measure the number of optimal solutions found within 10^4, 10^5 and 10^6 iterations and additionally check if these solutions correspond to the unique solution of the sudoku puzzle. The complete source code of the algorithm and the experiments, as well as the results of the experiments are available at the author's website [45].

The results of our experiments are summarized in Table 2. They are shocking and they are in stark contrast to the results reported in [20]. None of the 720 runs could find the unique solution of the sudoku puzzle within the maximum number of 10^6 iterations. Please note that these are 100 times more iterations than used in [20]. Additionally, there is not a single run in which any of the optimal solutions was found within 10^4 iterations, the maximum number of iterations used in [20]. Moreover (and not shown in Table 2), in only 29 runs an optimal solution (not the unique solution to the sudoku puzzle) was found within 10^5 iterations and in only 96 runs an optimal solution (not the unique solution to the sudoku puzzle) could be found within the maximum number of 10^6 iterations. Only with a harmony memory consideration rate of 0.9 combined with a pitch adjustment rate of 0.01 or 0.1 optimal solutions could be found. According to our theoretical investigations, these are the least destructive parameter combinations.

Before we continue with a discussion of our results, we would like to point out another inconsistency. While the computational

Table 2
The results of the empirical investigations. Reported are for each parameter setting the number of runs (out of 20) which lead to the unique solution of the sudoku puzzle within 10^6 iterations and any optimal solution within 10^4 iterations. Additionally, the number of iterations for obtaining the optimal solution according to [20] is given. Please note that the maximum number of iterations used in the original experiments was 10^4.

HMS	HMCR	PAR	Runs finding the unique solution in 10^6 iterations	Runs finding an optimal solution in 10^4 iterations	Iterations to obtain the optimal solution in [20]
1	0.5	0.01	0	0	66
		0.1	0	0	337
		0.5	0	0	422
	0.7	0.01	0	0	287
		0.1	0	0	3413
		0.5	0	0	56
	0.9	0.01	0	0	260
		0.1	0	0	not found
		0.5	0	0	1003
2	0.5	0.01	0	0	31
		0.1	0	0	94
		0.5	0	0	175
	0.7	0.01	0	0	102
		0.1	0	0	77
		0.5	0	0	99
	0.9	0.01	0	0	not found
		0.1	0	0	not found
		0.5	0	0	1325
10	0.5	0.01	0	0	49
		0.1	0	0	280
		0.5	0	0	188
	0.7	0.01	0	0	56
		0.1	0	0	146
		0.5	0	0	259
	0.9	0.01	0	0	180
		0.1	0	0	217
		0.5	0	0	350
50	0.5	0.01	0	0	147
		0.1	0	0	372
		0.5	0	0	649
	0.7	0.01	0	0	165
		0.1	0	0	285
		0.5	0	0	453
	0.9	0.01	0	0	87
		0.1	0	0	329
		0.5	0	0	352

time for a run with 10^4 iterations of our algorithm on a customary laptop was just 0.02 s, the computational times reported in [20] are all several seconds. It is not really clear why 66 iterations of the harmony search algorithm applied to the sudoku puzzle should require a computational time of 5 s or why 10^4 iterations should require a computational time of almost 2 min.

3.4. Conclusions

In this section we performed a detailed investigation of the publication *Harmony search algorithm for solving sudoku* [20] by Geem, the founder of the harmony search algorithm. Our investigations show that this publication contains fundamental inconsistencies: (1) It is highly questionable, why a heuristic should be applied for solving the sudoku puzzle. This is a huge conceptual issue. (2) The proposed objective function is not appropriate for the sudoku problem. As we have seen, there are in general many optimal solutions (with respect to the objective function), whereas there is only a single solution for the original sudoku puzzle. It seems that the author of [20] was not aware of this fact. Additionally, this issue makes it difficult to interpret the results of [20]. What is actually reported in that publication, the number of iterations until an optimal solution has been found or the number of iterations until the unique solution to the sudoku puzzle has been obtained? (3) Our theoretical investigation shows that the success probability of the proposed algorithm is extremely small for many of the parameter settings. It is therefore very surprising that so many successful runs are reported in [20]. In fact, the likelihood for the reported results are astronomically small. (4) The previous issues necessitated a repetition of the experiments conducted in [20]. Our results differ completely from those reported in [20]. While the original results were mainly positive, we could not obtain the unique solution to the sudoku puzzle in any of our runs. Even obtaining one of the many other optimal solutions with respect to the objective function turned out to be a difficult task, which could only be tackled by running the algorithm for many more iterations under very specific parameter settings. (5) The computational times reported in [20] do not seem plausible for such a simple algorithm. In fact, our reimplementation was around 4 orders of magnitude faster.

4. Discussion and conclusions

In this article we gave a formal proof for the fact that the harmony search algorithm is a special case of evolution strategies. The two main implications are that the harmony search

algorithm offers no novelty, apart from its metaphor, and that the performance of the harmony search algorithm is always limited by the performance of evolution strategies. This confirms and strengthens the conclusions drawn in the 2010 article [9], namely that "research in harmony search is fundamentally misguided" and that "future research effort could better be devoted to more promising areas".

Additionally, we investigated the publication *Harmony search algorithm for solving sudoku* [20] by Geem, the founder of the harmony search algorithm, more in detail. The results of our detailed analysis of the objective function, of our theoretical investigation of the algorithm's success probability and of our empirical investigations raise doubts about the conclusions drawn in that publication. Considering the huge disparity between our results and the results reported in [20], it is of great importance to resolve this discrepancy. Our analysis of the objective function and our theoretical analysis of the algorithm's success probability are self-contained and regarding the empirical investigations all algorithms and their source codes are publicly available. To be able to resolve this issue, it would therefore be very helpful if the algorithms and the corresponding source codes of [20] would become publicly available.

We would like to outline three directions for further research, which all have the potential to contribute in improving the current situation in research and especially in the fields of optimization and heuristics. The first direction is about establishing standards for assessing and comparing the performance of heuristics. Many publications contain serious conceptual flaws which makes it extremely difficult or even impossible to draw conclusions based on the experimental results. A transparent system which allows to perform meaningful research by following a few guidelines is of uttermost importance and urgency. Additionally, the community has to actively enforce that conclusions drawn from empirical investigations are correct and sound. The second direction deals with a classification of search heuristics, considering their history, their relations among each other, their similarities and differences and their advantages and disadvantages. In this context an emphasis on the contributions of "novel" search heuristics would be of great interest. The last direction targets an improvement of the research system. It is obvious that the current system gives flawed incentives to researchers and other involved parties. Of course, this is a broad subject and we cannot expect any rapid changes and improvements, especially considering the current situation with all the interweaved structures combined with many financial interests. But it is clear that nothing will change and nothing will improve as long as we continue with our attitude of passiveness.

Acknowledgments

We would like to thank the reviewers for their helpful comments and suggestions. This research has been supported by the *Swiss National Science Foundation* as part of the *Early Postdoc.Mobility* grant 152293.

References

[1] Tran TH, Ng KM. A water-flow algorithm for flexible flow shop scheduling with intermediate buffers. J Sched 2011;14(5):483–500.
[2] Eusuff M, Lansey K, Pasha F. Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. Eng Optim 2006;38(2):129–54.
[3] Mozaffari A, Fathi A, Behzadipour S. The great salmon run: a novel bio-inspired algorithm for artificial system design and optimisation. Int J Bio-inspired Comput 2012;4(5):286–301.
[4] Eberhart R, Kennedy J. A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science. IEEE; 1995. p. 39–43.
[5] Yang XS. Firefly algorithm, levy flights and global optimization. In: Research and development in intelligent systems XXVI. Springer; 2010. p. 209–18.
[6] Pan WT. A new fruit fly optimization algorithm: taking the financial distress model as an example. Knowl -Based Syst 2012;26:69–74.
[7] Li XL, Qian JX. Studies on artificial fish swarm optimization algorithm based on decomposition and coordination techniques. J Circuits Syst 2003;1(1–6).
[8] Chu SC, Tsai PW, Pan JS. Cat swarm optimization. In: PRICAI 2006: Trends in artificial intelligence. Springer; 2006. p. 854–8.
[9] Weyland Dennis. A rigorous analysis of the harmony search algorithm: How the research community can be misled by a "novel" methodology. Int J Appl Metaheuristic Comput 2010;1(2):50–60.
[10] Padberg M. Harmony search algorithms for binary optimization problems. In: Operations research proceedings 2011. Springer; 2012. p. 343–8.
[11] Rechenberg I. Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution. 1973.
[12] Sörensen K. Metaheuristics—the metaphor exposed. Int Trans Oper Res 2013.
[13] Geem ZW, Kim JH, Loganathan GV. A new heuristic optimization algorithm: harmony search. Simulation 2001;76(2):60–8.
[14] Sivasubramani S, Swarup KS. Multi-objective harmony search algorithm for optimal power flow problem. Int J Electr Power Energy Syst 2011;33(3):745–52.
[15] Wang L, Pan QK, Fatih Tasgetiren M. Minimizing the total flow time in a flow shop with blocking by using hybrid harmony search algorithms. Expert Syst Appl 2010;37(12):7929–36.
[16] Kaveh A, Talatahari S. Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Comput Struct 2009;87(5):267–83.
[17] Geem ZW. Optimal design of water distribution networks using harmony search (Ph.D. thesis), Korea University; 2000.
[18] Geem ZW, Tseng CL, Park Y. Harmony search for generalized orienteering problem: best touring in china. In: Advances in natural computation. Springer; 2005. p. 741–50.
[19] Geem ZW. Optimal scheduling of multiple dam system using harmony search algorithm. In: Computational and ambient intelligence. Springer; 2007. p. 316–23.
[20] Geem ZW. Harmony search algorithm for solving sudoku. In: Knowledge-Based Intelligent Information and Engineering Systems. Springer; 2007. p. 371–8.
[21] Saka MP. Optimum design of steel sway frames to bs5950 using harmony search algorithm. J Construct Steel Res 2009;65(1):36–43.
[22] Saka MP. Optimum geometry design of geodesic domes using harmony search algorithm. Adv Struct Eng 2007;10(6):595–606.
[23] Wang CM, Huang YF. Self-adaptive harmony search algorithm for optimization. Expert Syst Appl 2010;37(4):2826–37.
[24] Mahdavi M, Fesanghary M, Damangir E. An improved harmony search algorithm for solving optimization problems. Appl Math Comput 2007;188(2):1567–79.
[25] Omran MGH, Mahdavi M. Global-best harmony search. Appl Math Comput 2008;198(2):643–56.
[26] Geem ZW. Improved harmony search from ensemble of music players. Lect Notes Comput Sci 2006;4251(86).
[27] Fesanghary M, Mahdavi M, Minary-Jolandan M, Alizadeh Y. Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems. Comput Methods Appl Mech Engrg 2008;197(33):3080–91.
[28] Yildiz AR. Hybrid taguchi-harmony search algorithm for solving engineering optimization problems. Int J Ind Eng Theory Appl Pract 2008;15(3):286–93.
[29] Li HQ, Li L. A novel hybrid particle swarm optimization algorithm combined with harmony search for high dimensional optimization problems. In: Intelligent pervasive computing, 2007. IPC. The 2007 international conference on. IEEE; 2007. p. 94–7.
[30] Mukhopadhyay A, Roy A, Das S, Das S, Abraham A. Population-variance and explorative power of harmony search: an analysis. In: Second national conference on mathematical techniques: Emerging paradigms for electronics and IT Industries (MATEIT 2008), New Delhi, India. 2008.
[31] Panchal A. Harmony search optimization for HDR prostate brachytherapy. ProQuest; 2008.
[32] Geem ZW. Music-inspired harmony search algorithm: theory and applications, vol. 191. Springer Verlag; 2009.
[33] Geem ZW. Harmony search algorithms for structural design optimization, vol. 239. Springer-Verlag New York Incorporated; 2009.
[34] Geem ZW. Recent advances in harmony search algorithm, vol. 270. Springer Verlag; 2010.
[35] De Corte A, Sörensen K. Optimisation of gravity-fed water distribution network design: A critical review. European J Oper Res 2013;228(1):1–10.
[36] Geem ZW. Research commentary: Survival of the fittest algorithm or the novelest algorithm?. Int J Appl Metaheuristic Comput 2010;1(4):75–9.
[37] Bäck T, Schwefel HP. An overview of evolutionary algorithms for parameter optimization. Evol Comput 1993;1(1):1–23.
[38] Bäck T, Rudolph G, Schwefel HP. Evolutionary programming and evolution strategies: Similarities and differences. In: Proceedings of the second annual conference on evolutionary programming. Citeseer; 1993. p. 11–22.
[39] Bäck T., Hoffmeister F., Schwefel H.. A survey of evolution strategies. In: Proceedings of the fourth international conference on genetic algorithms, 1991.

[40] Schwefel HP. Numerical optimization of computer models. New York, NY, USA: John Wiley & Sons, Inc.; 1981.
[41] Fogel DB, Atmar JW. Comparing genetic operators with Gaussian mutations in simulated evolutionary processes using linear systems. Biol Cybern 1990; 63(2):111–4.
[42] Michalewicz Z, Janikow CZ, Krawczyk JB. A modified genetic algorithm for optimal control problems. Comput Math Appl 1992;23(12):83–94.
[43] Michalewicz Z. Genetic algorithms + data structures = evolution programs. Springer; 1996.
[44] Sudoku solutions: Sudoku solver. http://www.sudoku-solutions.com. Accessed: 26.07.2014.
[45] Additional material. http://www.dennisweyland.net – The documents are accessible from the publications page, they are listed in the corresponding entry for this article.