Received 18 July 2022, accepted 26 August 2022, date of publication 29 August 2022, date of current version 9 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3202944

RESEARCH ARTICLE

Stochastic Models of Jaya and Semi-Steady-State Jaya Algorithms

UDAY KUMAR CHAKRABORTY
Department of Computer Science, University of Missouri—St. Louis, MO 63121, USA
e-mail: chakrabortyu@umsl.edu

This work was supported in part by a University of Missouri-St. Louis Computer Science Grants Incentive Fund.

ABSTRACT The Jaya algorithm and its variants have enjoyed great success in diverse application areas, but no theoretical analysis of the algorithm, to our knowledge, is available in the literature. In this paper we build stochastic models for analyzing Jaya and semi-steady-state Jaya algorithms. For these algorithms, the computational cost depends on how, at each iteration, the new individual fares against the existing individual. Costs must be incurred for any replacement of individuals and the subsequent update of the population-worst individual's (and/or the population-best individual's) index. We use the following two quantities as the main metrics for analysis: the expected number of updates in a generation of the worst individual's index, and the corresponding expectation for updating the best individual's index. Clearly, the higher these expectations, the costlier the algorithm. The analysis shows that for semi-steady-state Jaya (a) the maximum expected number of worst-index updates per generation is 1.7 regardless of the population size; (b) regardless of the population size, the expectation of the number of best-index updates per generation decreases monotonically with generations; (c) upper bounds as well as asymptotics of the expected best-update counts can be obtained for specific distributions; the upper bound is 0.5 for normal and logistic distributions, $\ln 2$ for the uniform distribution, and $e^{-\gamma} \ln 2$ for the exponential distribution, where γ is the Euler-Mascheroni constant; the asymptotic is $e^{-\gamma} \ln 2$ for logistic and exponential distributions and $\ln 2$ for the uniform distribution (the asymptotic cannot be obtained analytically for the normal distribution). The models lead to the derivation of computational complexities of Jaya and semi-steady-state Jaya. The theoretical analysis is supported with empirical results on a benchmark suite.

INDEX TERMS Evolutionary algorithm, Jaya algorithm, machine learning, metaheuristics, optimization, stochastic model.

I. INTRODUCTION

The Jaya algorithm (technically, heuristic or meta-heuristic) [1], [2] is one of the newest members of the evolutionary computation family. This algorithm and its variants have been highly successful in gradient-free global optimization, both constrained and unconstrained, of non-convex problems in continuous domains and have seen wide applicability in diverse areas, including engineering [3], [4], [5], [6], manufacturing [7], energy [8], fuel cells [9], health-care [10] and finance [11]. Discrete (e.g., [12], [13]) and multi-objective (e.g., [14]) versions of Jaya have also been developed. A recent survey can be found in [15]. Within the genetic and evolutionary computation family, Jaya is unique in its use of a minimum number of parameters, a fact that doubtless contributes to this algorithm's popularity among practitioners. The semi-steady-state Jaya (SJaya for short) [16] has been shown to outperform the standard Jaya on benchmark problems, with the improvement in performance attributed primarily to the new update strategies that SJaya employs for the best and worst members of the population.

Despite their impressive growth, no theoretical analysis of Jaya or its variants has, to our knowledge, been reported in the literature. Such analysis is fundamental to our understanding of why the method works the way it does and is a necessary prerequisite to designing better, newer methods for tackling hard optimization problems. This paper provides a theoretical underpinning of this powerful algorithm, modeling the

The associate editor coordinating the review of this manuscript and approving it for publication was Gustavo Olague.

algorithm as a stochastic process and deriving bounds and asymptotics for important performance metrics. Specifically, we show that for semi-steady-state Jaya (a) the maximum expected number of worst-index updates per generation is 1.7 regardless of the population size; (b) regardless of the population size, the expectation of the number of best-index updates per generation decreases monotonically with generations; (c) upper bounds as well as asymptotics of the expected best-update counts can be obtained for specific distributions; the upper bound is 0.5 for normal and logistic distributions, $\ln 2$ for the uniform distribution, and $e^{-\gamma}\ln 2$ for the exponential distribution, where γ is the Euler-Mascheroni constant; the asymptotic is $e^{-\gamma}\ln 2$ for logistic and exponential distributions and $\ln 2$ for the uniform distribution. The model allows us to investigate the costs of the update strategies, revealing several interesting facts about the working of Jaya and SJaya, leading to the derivation of the computational complexities of the algorithms. We also present empirical results on a five-function test suite and on a real-world problem.

The Jaya pseudocode [9] and SJaya pseudocode [16] are presented here as Algorithms 1 and 2, respectively. The population is initialized randomly or heuristically. The population then evolves over a number of generations, with the stopping condition being determined from one of a number of strategies, e.g., a pre-determined number of generations, a pre-determined number of fitness (objective function) evaluations, a pre-determined quantity of CPU time, finding of a solution of an acceptable quality (this presupposes the availability of some information on what constitutes a good solution), detection of stagnation or lack of significant progress in the search process. As in standard versions of evolutionary algorithms, the population size in Jaya/SJaya stays the same across generations. Each generation examines, in some sequence, each of the population members (individuals or chromosomes) for possible replacement by a new individual, where the new individual is created, nondeterministically, from the current individual and two other individuals: the population-best and the population-worst individuals.

The key step in either algorithm involves the creation of the new individual (line 6 in Algorithm 1 and also in Algorithm 2). A new individual $x^{\text{new}} \in \mathbb{R}^d$ is created from the current individual $x^{\text{current}} \in \mathbb{R}^d$ by using the best individual $x^{\text{best}} \in \mathbb{R}^d$, the worst individual $x^{\text{worst}} \in \mathbb{R}^d$, and two random numbers—each chosen uniformly randomly in (0, 1]—for each of the problem parameters:

$$x_i^{\text{new}} = x_i^{\text{current}} + r_{g,i,1}(x_i^{\text{best}} - |x_i^{\text{current}}|) - r_{g,i,2}(x_i^{\text{worst}} - |x_i^{\text{current}}|) \quad (1)$$

where x_i, $i = 1, \cdots, d$, represent the parameters or variables to be optimized, $r_{g,i,1}$ and $r_{g,i,2}$ are each a random number in (0.0, 1.0], and g indicates the generation number. If any component of x_i^{new} falls outside its problem-specified lower or upper bound, it is clamped at the appropriate bound. A new individual replaces the existing one if the former is at least as good as the latter. A replacement may cause the population-best member or the population-worst member to change. Jaya and SJaya differ in how the post-replacement update, if any, of the best individual (or the worst) is handled and used for the rest of the algorithm. Further details of these two algorithms can be found in [1], [16].

Algorithm 1: Jaya

```
1  initialize the population;
2  while a pre-determined stopping condition is not satisfied do
3      find the best and the worst individuals in the population, and save copies of them;
4      set the parameters, independently of one another, to random values between 0.0 and 1.0;
5      for each individual in the population do
6          create a new individual using the current individual, the (saved copy of the) best individual, the (saved copy of the) worst individual, and the random parameters;
7          if the new individual is at least as good as the current individual then
8              replace the current individual with the new individual;
9          end
10     end
11 end
```

II. FRAMEWORK FOR THE ANALYSIS

We assume, without loss of generality, an indexed representation (e.g., an array) (Fig. 1) of the members of the population. The best and the worst members (individuals) are determined with respect to the fitness / utility / cost or some objective function. A single run of Jaya or SJaya comprises a number (G, say) of generations, and each generation consists of n steps or iterations, where n is the population size.

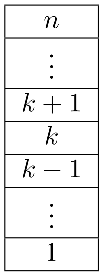


FIGURE 1. Indexed representation of population members (population size = n).

A single iteration involves determining whether or not the member at index i ($i = 1, 2, \cdots, n$) is to be replaced with a new individual. Clearly, it does not matter whether we traverse the population (array) in a top-to-bottom or

Algorithm 2: Semi-Steady-State Jaya

```
1  initialize the population;
2  find the best and the worst individuals in the population,
     and initialize bestIndex to the index of the best
     individual and worstIndex to the index of the worst
     individual;
3  while a pre-determined stopping condition is not
     satisfied do
4      set the parameters, independently of one another,
         to random values between 0.0 and 1.0;
5      for each individual in the population starting from
         the first index do
6          create a new individual using the current
             individual, the individual at bestIndex,
             the individual at worstIndex, and the random
             parameters;
7          if the new individual is at least as good as the
             current individual then
8              replace the current individual with the new
                 individual;
9              if the current individual is better than the
                 individual at bestIndex then
10                 update bestIndex to set it to the current
                     index;
11             end
12             if the current individual's index is the same
                 as worstIndex then
13                 find the worst individual in the
                     population and set worstIndex to the
                     index of the worst individual;
14             end
15         end
16     end
17 end
```

bottom-to-top or any other fashion, as long as no index is left out or considered more than once. Suppose, for ease of discussion, we traverse the population in Fig. 1 sequentially from the top (index n) to the bottom (index 1).

A. TWO METRICS

Algorithms 1 and 2 show that a major component of the complexity hinges on how, at each iteration, the new individual fares against the existing individual (line 7 in either algorithm). Clearly, costs must be incurred for any replacement of individuals and the subsequent update of the worst individual's (and/or the best individual's) index. Jaya and SJaya being randomized algorithms, occurrences of replacements and updates must be studied as stochastic processes, which leads us to use the following two quantities as the main metrics for analysis: the expected number of updates (in a generation) of the worst individual's index, and the corresponding expectation for the best individual's index. Clearly, the higher these expectations, the costlier the algorithm.

III. UPDATING SJaya's WORST-OF-POPULATION INDEX

Because the population changes with time, the index of the population's worst individual is time-dependent; that is, the worst individual's index may change after every replacement of the current (most recent) worst individual. Thus it is possible for the (current) worst individual to be encountered more than once during the top-to-bottom scan in a given generation of the population. The present analysis assumes that when the worst individual is encountered, it is replaced with a new (better or identical-fitness) individual with probability p.

Let us use the notation findWorst() to indicate the function called to find the index of the worst individual in the population, and let worstIndex represent the index of the worst individual at any point in the course of a run. Because a simple linear scan of the population is enough to find the worst fitness, the complexity of findWorst() is $\Theta(n)$.

Let X be the (discrete) random variable representing the total number of calls, in an entire generation, to the function findWorst(). We are interested in finding the expected value of X, because the higher this expectation, the higher the cost of SJaya.

Suppose that at the beginning of a new generation, the worst individual in the entire population is at index k, i.e., worstIndex is k, with $1 \leq k \leq n$. During the course of the generation, when this individual at index k is encountered, it will either stay unaltered or be replaced with a new individual. If it stays unchanged, worstIndex stays unaltered. If, however, it undergoes replacement, we must find (by using a call to findWorst()) which individual in the population is the new worst (it is possible that the newly arrived individual at index k, while better than the individual just replaced, turns out to be the worst in the population at that point in time). The new worst individual, as identified by the above-mentioned call to findWorst(), must be

- either in the already-traversed portion of the population array (at an index between k and n, inclusive, in Fig. 1);
- or in the yet-to-be-traversed part of the population (at an index h, with $1 \leq h \leq k-1$).

In the first case above, no further call to findWorst() is needed for the rest of the generation, while in the second, the story will repeat itself with the new worst individual, necessitating a total of up to h (i.e., at least zero but at most h) further calls to findWorst() for the rest of the current generation.

Let W be the discrete random variable representing the index of the worst individual in the population at a particular iteration of a particular generation during the execution of a run. Let us use t to represent the iteration number (not to be confused with the generation number for which we use the notation g). Thus $1 \leq t \leq n$ and $1 \leq g \leq G$. For the n steps (iterations) in any generation of SJaya, the corresponding variables are $W^{(0)}$ (the index of the worst individual at the start of a new generation), $W^{(1)}$ (the worst individual's index after the first iteration of the generation is over), and so on. Thus $W^{(n)}$ of a given generation is the same as $W^{(0)}$ of the immediately following generation. At the very beginning, under the assumption that the initial population is randomly generated, all slots of the array in Fig. 1 are equally likely to hold the worst one (P stands for probability):

$$P(W^{(0)} = k \mid n) = \frac{1}{n} \quad \text{for } k = 1, 2, \cdots, n. \tag{2}$$

Now, for the present analysis, we do not need $P(W^{(t)} = k \mid n)$ as much as we need the conditional probability $P(W^{(t+1)} = j \mid W^{(t)} = k; n)$, which, in the absence of any further information, is assumed to be uniform:

$$P(W^{(t+1)} = j \mid W^{(t)} = k; n) = \frac{1}{n} \quad \text{for any } (j, k) \text{ pair and any } t. \tag{3}$$

In the course of a generation, when the worst individual is encountered, a new individual is produced and is compared against the worst individual (what happens to the worst individual is no different from what happens to every other individual in the population at the given generation). Now, if the new individual has a fitness that is better than or equal to that of the worst individual, the former replaces the latter, thereby necessitating the finding of which individual in the post-replacement state is the (new) worst in the population. This entails one call to findWorst(). Therefore, in the event of the replacement of the worst individual, at least one call must be made to findWorst(). Thus for an entire generation, we have

$$P(X > 0 \mid W^{(0)} = k; n) = p, \tag{4}$$
$$P(X = 0 \mid W^{(0)} = k; n) = 1 - p. \tag{5}$$

Given $W^{(0)} = k$ for a certain generation (recall that one generation equals n iterations), the variable X can assume one of the following values for that (entire) generation: $0, 1, 2, \cdots, k$. Thus equation 4 can be written more specifically as:

$$\sum_{m=1}^{k} P(X = m \mid W^{(0)} = k; n) = p. \tag{6}$$

Assuming $W^{(0)} = k$, consider the top-to-bottom journey in Fig. 1. The index k may be thought of as indicating the point of demarcation, splitting the population into a top part of size $n - k + 1$ and a bottom part of size $k - 1$. Given $W^{(0)} = k$, we can describe the result of a call to findWorst() as either an "up" move (when the index returned by findWorst() is $\geq k$) or a "down" move (when the returned index is $< k$). Thus, given $W^{(0)} = k$, the event $X = 1$ takes place when, starting at index k, we either move "up" once, never to move anywhere else, or move "down" once and do not move further:

$$\begin{aligned} P(X = 1 \mid W^{(0)} = k \geq 1; n) &= p \times \sum_{i=k}^{n} \frac{1}{n} + p \times \sum_{i=1}^{k-1} \frac{1}{n} \times (1-p) \\ &= \frac{p}{n}(n + p - pk). \end{aligned} \tag{7}$$

The event $X = 2$ occurs in one of the following two scenarios: (a) starting from slot k, the first move is a "down" move to slot $i \in [1, k-1]$, and the second one is a move "up" to any slot $\in [i, n]$; (b) the first two moves are "down" each, followed by no further movement:

$$\begin{aligned} &P(X = 2 \mid W^{(0)} = k \geq 2; n) \\ &= p \sum_{i=1}^{k-1} \left(\frac{1}{n} \times p \sum_{j=i}^{n} \frac{1}{n} \right) \\ &\quad + p \sum_{i=2}^{k-1} \left(\frac{1}{n} \times p \sum_{j=1}^{i-1} \frac{1}{n} \times (1-p) \right) \\ &= \frac{p^2}{n^2}(k-1)\left(n + p - \frac{pk}{2}\right). \end{aligned} \tag{8}$$

Similarly, X is 3 when we have either (a) a "down" move from k to any location $i \in [2, k-1]$, followed by a second "down" move from i to any location $j \in [1, i-1]$, followed, finally, by an "up" move from j to any location $\in [j, n]$; or (b) three successive "down" moves followed by no further movement:

$$\begin{aligned} &P(X = 3 \mid W^{(0)} = k \geq 3; n) \\ &= p \sum_{i=2}^{k-1} \left(\frac{1}{n} \times p \sum_{j=1}^{i-1} \left(\frac{1}{n} \times p \sum_{h=j}^{n} \frac{1}{n} \right) \right) \\ &\quad + p \sum_{i=3}^{k-1} \left(\frac{1}{n} \times p \sum_{j=2}^{i-1} \left(\frac{1}{n} \times p \sum_{h=1}^{j-1} \frac{1}{n} \times (1-p) \right) \right) \\ &= \frac{p^3}{2n^3}(k-1)(k-2)\left(n + p - \frac{pk}{3}\right). \end{aligned} \tag{9}$$

A. THE GENERAL CASE

For the general case $X = m$, where $m \in [1, n]$, we have the following theorem (the product notation Π evaluates to 1 when the upper bound is less than the lower bound):

Theorem 1: For $m \geq 1$,

$$\begin{aligned} &P(X = m \mid W^{(0)} = k \geq m; n) \\ &= \frac{1}{(m-1)!} \left(\frac{p}{n}\right)^m \left(n + p - \frac{pk}{m}\right) \prod_{j=1}^{m-1} (k - j). \end{aligned} \tag{10}$$

Proof: We present a proof by induction. The base cases for $m = 1, 2$ and 3 are already established via equations 7, 8, 9. The proof will be complete when, assuming the theorem is true for $m = q \geq 1$, we show that it is true for $m = q + 1$.

Starting from location k, any $(q+1)$-move sequence comprises a first "down" move to any location $i \in [q, k-1]$, followed by a sequence of q further moves, with the first move of the q-move sequence starting at location i. Thus

$$\begin{aligned} &P(X = q + 1 \mid W^{(0)} = k \geq q + 1; n) \\ &= p \sum_{i=q}^{k-1} \left(\frac{1}{n} \times P(X = q \mid W^{(0)} = i \geq q; n) \right). \end{aligned} \tag{11}$$

Substituting for $P(X = q|W^{(0)} = i \geq q; n)$ from the theorem (equation 10) into the above equation, we have

$$\begin{aligned}
&P(X = q+1 \mid W^{(0)} = k \geq q+1; n)\\
&= \frac{1}{(q-1)!}\left(\frac{p}{n}\right)^{q+1}\sum_{i=q}^{k-1}\left(\left(n+p-\frac{pi}{q}\right)\prod_{j=1}^{q-1}(i-j)\right)\\
&= \frac{1}{(q-1)!}\left(\frac{p}{n}\right)^{q+1}\left[(n+p)\sum_{i=q}^{k-1}\prod_{j=1}^{q-1}(i-j)\right.\\
&\quad \left. -\frac{p}{q}\sum_{i=q}^{k-1}\left(i\prod_{j=1}^{q-1}(i-j)\right)\right]. \qquad (12)
\end{aligned}$$

Now, it can be shown (after some algebra) that

$$\sum_{i=q}^{k-1}\prod_{j=1}^{q-1}(i-j) = \frac{(k-q)\,(k-1)!}{q\,(k-q)!} \qquad (13)$$

and

$$\sum_{i=q}^{k-1}\left(i\prod_{j=1}^{q-1}(i-j)\right) = \frac{k(k-1)(k-2)\cdots(k-q)}{q+1}. \qquad (14)$$

Use of the two identities 13 and 14 in equation 12 followed by some simplification yields

$$\begin{aligned}
&P(X = q+1 \mid W^{(0)} = k \geq q+1; n)\\
&\quad = \frac{1}{q!}\left(\frac{p}{n}\right)^{q+1}\left(n+p-\frac{pk}{q+1}\right)\prod_{j=1}^{q}(k-j). \qquad (15)
\end{aligned}$$

Q.E.D.

The expectation of X, given a particular $W^{(0)}$ and a particular n, is now obtained as

$$\begin{aligned}
&E(X \mid W^{(0)} = k; n)\\
&\quad = 0 \times (1-p) + \sum_{m=1}^{k}(m \times P(X = m \mid W^{(0)} = k \geq m; n))\\
&\quad = \sum_{m=1}^{k}(m \times P(X = m \mid W^{(0)} = k \geq m; n)). \qquad (16)
\end{aligned}$$

Finally, the expectation of X, given an n, is

$$\begin{aligned}
E(X \mid n) &= \sum_{k=1}^{n} E(X \mid W^{(0)} = k; n)P(W^{(0)} = k \mid n)\\
&= \frac{1}{n}\sum_{k=1}^{n} E(X \mid W^{(0)} = k; n). \qquad (17)
\end{aligned}$$

The probability $P(X = m \mid W^{(0)} = k \geq m; n)$ and hence the expectation $E(X \mid n)$ are monotone increasing in p, with $E(X \mid n)$ reaching its highest possible value when $p = 1$. A closed-form expression for the maximum value of $E(X \mid n)$ can be obtained using some algebra:

$$\begin{aligned}
&\text{Max } E(X \mid n)\\
&= \frac{1}{n}\sum_{k=1}^{n}\left(\sum_{m=1}^{k}\frac{m}{(m-1)!}\,\frac{1}{n^m}\left(n+1-\frac{k}{m}\right)\frac{(k-1)!}{(k-m)!}\right)\\
&= \left(1+\frac{1}{n}\right)^n - 1 \qquad (18)
\end{aligned}$$

TABLE 1. Maximum value (theoretical) of $E(X|n)$.

n	$E(X\|n)$
10	1.593742
50	1.691588
100	1.704813
500	1.715568
1500	1.717376
2500	1.717738
3500	1.717893
4500	1.717979
10000	1.718145
20000	1.718213
30000	1.718236
40000	1.718247

Table 1 presents the analytically obtained maximum value of $E(X|n)$ for different values of n. Table 1 shows that the maximum value is almost constant, regardless of the population size. Given the existing body of research on population sizing in evolutionary computation, we can say that the spread of population sizes in Table 1 is wide enough to include almost all cases of practical interest. Ignoring less-than-50 values of n as too small, we arrive at the rather remarkable conclusion that the expected number of worst-index updates per generation is 1.7 for almost any population size. That the upper bound of the maximum of the expectation is 1.7 can be established analytically by using the fact that

$$\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n = e,$$

where $e \approx 2.71828$ is the Euler number.

B. EMPIRICAL RESULTS

1) TEST SUITE

The test suite (see Table 2), taken from [16], comprises five popular benchmark functions (with well-known global optimum) from the literature and one real-world problem for which the global minimum is believed to be mathematically intractable.

The five functions include convex and non-convex, and differentiable and non-differentiable cases, presenting different levels of problem difficulty involving different numbers of local optima.

The real-world problem is the task of designing a proton exchange membrane fuel cell (PEMFC) stack [22], [23], and it involves constrained optimization, with the goal of minimizing the dollar cost of building the stack while meeting specific practical requirements. The objective (cost) function is defined [9], [16], [24], [25] on three variables $N_{\mathrm{p}}, N_{\mathrm{s}}$, A_{cell} and a number of pre-determined constants (we use the

TABLE 2. Empirical test suite.

Name	Definition	Dim.	Global Minimum	Bounds
Ackley [17]	$f(x_1,\cdots,x_d) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e$	30	$f(x^*) = 0$ $x^* = (0,\cdots,0)$	$-10 \le x_i \le 10$
Rosenbrock [18]	$f(x_1,\cdots,x_d) = \sum_{i=1}^{d-1}[100(x_{i+1}-x_i^2)^2 + (1-x_i)^2]$	30	$f(x^*) = 0$ $x^* = (1,\cdots,1)$	$-10 \le x_i \le 10$
Chung-Reynolds [19]	$f(x_1,\cdots,x_d) = \left(\sum_{i=1}^{d} x_i^2\right)^2$	30	$f(x^*) = 0$ $x^* = (0,\cdots,0)$	$-10 \le x_i \le 10$
Step [20]	$f(x_1,\cdots,x_d) = \sum_{i=1}^{d}\lfloor \lvert x_i \rvert \rfloor$	30	$f(x^*) = 0$ $x_i^* \in (-1,1)$	$-100 \le x_i \le 100$
Goldstein-Price [21]	$f(x_1,x_2) = \left[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)\right] \times \left[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)\right]$	2	$f(x^*) = 3$ $x^* = (0,-1)$	$-2 \le x_1, x_2 \le 2$
PEMFC [16]	See equations 19, 20, 21	3	—	See Table 3

TABLE 3. Bounds of the design variables [24].

Variable	Lower Bound	Upper Bound
N_s	1	50
N_p	1	50
A_{cell} (cm^2)	10	400

notational conventions of [16]):

$$\text{Objective function} = K_n \times N_p \times N_s + K_{diff} \times \left|V_{load,r} - V_{load,mpp}\right| + K_a \times A_{cell} + \mathcal{P}, \tag{19}$$

where N_s is the number of cells connected in series in each group; N_p is the number of groups connected in parallel; A_{cell} is the cell area; $V_{load,r}$ is the rated terminal voltage of the stack; $V_{load,mpp}$ represents the output voltage at the maximum power point of the stack; $P_{load,r}$ is the rated output power of the stack; $P_{load,max}$ is the maximum output power of the stack; K_n, K_{diff}, K_a are pre-determined constants [9], [16] used to adjust the relative importance of the different components of the cost function; and $\mathcal{P}$ represents a penalty term [16] given by

$$\mathcal{P} = \begin{cases} 0 & \text{if } P_{load,max} \geq P_{load,r}; \\ c(P_{load,r} - P_{load,max}) & \text{otherwise.} \end{cases} \tag{20}$$

As in [16], $P_{load,max}$ and $V_{load,mpp}$ are obtained numerically from the following equation by iterating over the load current $i_{load,d}$, using a step size of $i_{load} = 1$ mA [16]:

$$V_{st} = N_s\left[E_{Nernst} - A\ln\left(\frac{i_{load,d}/N_p + i_{n,d}}{i_{0,d}}\right) + B\ln\left(1 - \frac{i_{load,d}/N_p + i_{n,d}}{i_{limit,d}}\right) - \left(\frac{i_{load,d}}{N_p} + i_{n,d}\right)r_a\right], \tag{21}$$

where V_{st} is the stack voltage, E_{Nernst} is the Nernst e.m.f., A and B are constants known from electrochemistry, r_a is the area-specific resistance, and the i's represent different types of currents, or rather current densities (the subscript d indicates density) in the cell [22], [26]. The lower and upper bounds of N_s, N_p and A_{cell}, taken from [24], are shown in Table 3, and the numerical values of the constants, taken from [16], are given in Table 4.

2) RESULTS

We obtain empirical estimates of the probability p and the expectation $E(X|n)$ by aggregating (averaging) results from multiple, independent runs of SJaya. Table 5 presents the empirical average and the theoretical expectation for each of the problems in the test-suite in Table 2. Each row in Table 5 corresponds to 500 runs, with each run executed for 20 generations with a specified population size. The theoretical expectation is obtained by plugging in the average empirical p into

TABLE 4. PEMFC parameters and constants [16].

Parameter	Value
$V_{load,r}$	12 V
$P_{load,r}$	200 W
K_n	0.5
K_{diff}	10
K_a	0.001
c	200
r_a	98.0×10^{-6} $K\Omega$ cm^2
$i_{limit,d}$	129 mA/cm^2
$i_{0,d}$	0.21 mA/cm^2
$i_{n,d}$	1.26 mA/cm^2
A	0.05 V
B	0.08 V
E_{Nernst}	1.04 V

TABLE 5. Empirical and theoretical $E(X|n)$ (rounded at the 4th decimal place).

Function	n	p	Empirical E	Theoretical E
Ackley	10	0.9230	1.701	1.4178
	50	0.9977	2.0547	1.6855
	100	0.9985	2.0786	1.7008
	1000	0.9996	2.1632	1.7158
Rosenbrock	10	0.8740	1.5262	1.3115
	50	0.9911	1.9779	1.6682
	100	0.9956	2.0029	1.6931
	1000	0.9988	2.0514	1.7137
Chung-Reynolds	10	0.9335	1.7408	1.4411
	50	0.9987	2.0366	1.6882
	100	0.9994	2.0508	1.7032
	1000	1.0000	2.0984	1.7169
Step	10	0.9590	1.8392	1.4986
	50	0.9994	2.0908	1.6900
	100	0.9998	2.1297	1.7043
	1000	1.0000	2.2024	1.7169
Goldstein-Price	10	0.5059	0.6554	0.6381
	50	0.6286	0.9442	0.8677
	100	0.6806	1.1151	0.9705
	1000	0.7805	1.6763	1.1819
PEMFC	10	0.8950	1.6828	1.3565
	50	0.8712	1.9334	1.3719
	100	0.8611	2.0014	1.3571
	1000	0.8709	2.2516	1.3881

equation 17. The empirical p value is obtained as the average of 500 probabilities, each probability being calculated as a relative frequency from a single run, the data for a single run having been aggregated from the 20 generations comprising the run. In other words, two levels of aggregating (averaging) were implemented: aggregating over runs and aggregating over generations within a single run. While the runs are independent of one another, the generations that make up a single run are not absolutely independent, having been created on top of one another, as if in a chain or cascade. The empirical expectation of X is obtained as the average (per generation per run) number of times the worst individual in the population needs to be found out.

While the empirical E agrees with its theoretical counterpart in that both increase with increasing population size, Table 5 shows that the empirical E is slightly higher than the corresponding theoretical value. This is explained by the fact that, in practice,

$$P(W^{(t+1)} = j \mid W^{(t)} = k; n) > \frac{1}{n} \quad \text{for } j < k \tag{22}$$

where the indexing scheme is as in Figure 1. This non-uniform distribution is difficult to obtain analytically. It can, of course, be qualitatively argued that in a top-to-bottom processing of the population elements, the top part (comprising indices $n, n-1, \cdots, k; 1 \le k \le n$; see Figure 1) gets updated before the bottom part does, and since an update never results in a worse fitness, the probability of the worst individual being found in the bottom part is higher than in the top part at any point during the course of a generation. An empirical corroboration of this can be seen in the matrix of $P(W^{(\text{next})} = j \mid W^{(\text{current})} = k)$ values in Table 6, which is obtained by averaging (in a relative-frequency sense) 5000 independent runs of SJaya on the Chung-Reynolds function (Table 2) of 10 variables, where each run used 10 generations of a population of size 10. The matrix in Table 6, which, clearly, is a stochastic matrix (each row-sum is unity, ignoring floating-point errors), shows that for each row, the entries to the left of the diagonal element are smaller than those to the right of the diagonal element.

As a reference, the very first or initial (before any replacement has taken place) distribution of the worstIndex, obtained from these 5000 runs, is presented in Table 7; the ten values follow an approximately uniform distribution, as is only to be expected.

IV. UPDATING SJaya's BEST-OF-POPULATION INDEX

We will now obtain the expected best-update count both empirically and theoretically.

A knowledge of the expected number of updates, in a generation, of the best-of-population index is required for an analysis of SJaya. Of course, to derive this expectation, we need the underlying (discrete) probability distribution. To compute the probability that a new individual, created in line 6 of Algorithm 1 or Algorithm 2, will replace the existing individual, we need, among other pieces of information, a knowledge of the (typically continuous) distribution of the fitness landscape. Now, the fitness distribution is impossible to know (in advance), except in trivial cases. A generic analysis, however, is possible if we are willing to make an assumption about the nature of this distribution.

An update of the best index is needed whenever the newly minted individual has a fitness better than that of the current population-best. During the course of a run, the expected value of the population-best fitness at the beginning of a fresh generation can be modeled as the expectation of the best (either maximum or minimum, depending on the application) of n i.i.d. samples drawn from a given fitness distribution, with n representing the population size. (The population size is assumed not to change from generation to generation, of course.) We assume maximization without loss of generality.

Let us use f for probability density function (pdf) and F for cumulative distribution function (cdf). The maximum of n i.i.d. samples $x_1, \cdots, x_n$ of a continuous random variable X is another (continuous) random variable; call it $X_{\max}$. Then the expected value of $X_{\max}$ is given by

$$E(X_{\max} \mid n, F_X) = \int_{x=-\infty}^{\infty} x \cdot f_{X_{\max}}(x \mid n, F_X)\, dx \tag{23}$$

where

$$f_{X_{\max}}(x \mid n, F_X) = n \cdot (F_X(x))^{n-1} f_X(x) \tag{24}$$

is the pdf of $X_{\max}$, and

$$F_{X_{\max}}(x \mid n, F_X) = P(X_{\max} < x) = (F_X(x))^n \tag{25}$$

is the cdf of $X_{\max}$, such that

$$P(x < X_{\max} < x + dx \mid n, F_X) = f_{X_{\max}}(x \mid n, F_X)\, dx, \tag{26}$$

where

$$F_X(x) = P(X < x) = \int_{x=-\infty}^{x} f_X(x)\, dx \tag{27}$$

is the cdf of X, with $f_X(x)$ representing its pdf.

To derive the expected number of updates, over a complete generation, of the population-best member, we begin by defining a discrete (binary) random variable $Y_{i,g;n,F}$ representing whether or not an update is made at iteration $i \in \{1, \cdots, n\}$ of generation $g \in \{1, \cdots, G\}$:

$$Y_{i,g;n,F} = \begin{cases} 1 & \text{if } X > E(X_{\max} \mid gn + i - 1, F_X) \\ 0 & \text{otherwise,} \end{cases} \tag{28}$$

with the initial generation ($g = 0$) assumed to have filled the population for the very first time. In other words, $Y_{i,g;n,F}$ is the indicator variable $1_{X > E(X_{\max} \mid gn+i-1, F_X)}$. The expectation of $Y_{i,g;n,F}$ is given by

$$E(Y_{i,g;n,F}) = P(X > E(X_{\max} \mid gn + i - 1, F_X)) \tag{29}$$

If $Y_{g;n,F}$ denotes a random variable representing the total number of updates in a given generation g, we have

$$Y_{g;n,F} = \sum_{i=1}^{n} Y_{i,g;n,F}, \tag{30}$$

and the expectation of $Y_{g;n,F}$ is then obtained as

$$\begin{aligned} E(Y_{g;n,F}) &= E\left(\sum_{i=1}^{n} Y_{i,g;n,F}\right) \\ &= \sum_{i=1}^{n} E(Y_{i,g;n,F}) \\ &= \sum_{i=1}^{n} P(X > E(X_{\max} \mid gn + i - 1, F_X)) \end{aligned} \tag{31}$$

where linearity of expectation has been used (the linearity is applicable regardless of whether or not the $Y_{i,g;n,F}$'s are independent).

TABLE 6. Transition matrix (empirical) of the index of the worst individual.

Current ↓ / Next →	10	9	8	7	6	5	4	3	2	1
10	0.042	0.112	0.123	0.108	0.111	0.106	0.101	0.096	0.102	0.098
9	0.089	0.047	0.118	0.113	0.117	0.105	0.105	0.104	0.103	0.099
8	0.098	0.084	0.042	0.125	0.116	0.113	0.108	0.11	0.104	0.1
7	0.099	0.095	0.096	0.045	0.113	0.121	0.118	0.104	0.105	0.105
6	0.102	0.096	0.092	0.092	0.048	0.123	0.111	0.119	0.11	0.109
5	0.1	0.099	0.097	0.09	0.092	0.041	0.128	0.119	0.121	0.114
4	0.11	0.103	0.105	0.098	0.096	0.093	0.04	0.125	0.112	0.119
3	0.101	0.112	0.104	0.097	0.102	0.101	0.093	0.048	0.126	0.115
2	0.113	0.108	0.105	0.102	0.108	0.095	0.096	0.098	0.046	0.129
1	0.114	0.116	0.108	0.106	0.106	0.105	0.101	0.099	0.099	0.046

TABLE 7. Initial empirical distribution of worstIndex (n = 10).

Index	Probability
10	0.1024
9	0.0954
8	0.1012
7	0.0970
6	0.0976
5	0.0984
4	0.0972
3	0.1030
2	0.1040
1	0.1038

From the definition of $X_{\max}$ it follows that for $n_2 > n_1$,

$$E(X_{\max} \mid n_2, F_X) > E(X_{\max} \mid n_1, F_X), \tag{32}$$

which implies

$$P(X > E(X_{\max} \mid n_2, F_X)) < P(X > E(X_{\max} \mid n_1, F_X)). \tag{33}$$

Thus we have

$$E(X_{\max} \mid g_2 n + i - 1, F_X) > E(X_{\max} \mid g_1 n + i - 1, F_X) \quad \text{for } g_2 > g_1 \geq 1 \tag{34}$$

or equivalently,

$$E(Y_{i,g_2;n,F}) < E(Y_{i,g_1;n,F}) \text{ for } g_2 > g_1 \geq 1. \tag{35}$$

Again

$$E(X_{\max} \mid gn + i_2 - 1, F_X) > E(X_{\max} \mid gn + i_1 - 1, F_X) \quad \text{for } i_2 > i_1, \tag{36}$$

or equivalently,

$$E(Y_{i_2,g;n,F}) < E(Y_{i_1,g;n,F}) \text{ for } i_2 > i_1. \tag{37}$$

Therefore

$$E(X_{\max} \mid (g+1)n, F_X) > E(X_{\max} \mid gn + n - 1, F_X) \tag{38}$$

which shows that the $E(Y_{i,g;n,F})$ value corresponding to the last iteration of any generation is strictly greater than that corresponding to the first iteration of the immediately following generation. Inequalities 34-38 lead to

$$E(Y_{g_2;n,F}) < E(Y_{g_1;n,F}) \quad \text{for } g_2 > g_1 \geq 1. \tag{39}$$

Note that the above inequality holds for any F_X (or equivalently, for any f_X) and for any n. Thus we have proved the following theorem:

Theorem 2: For any problem, in any run, the expected generation-wise best-update count decreases monotonically with generations, regardless of the population size.

Let

$$\bar{Y} = \frac{1}{G} \sum_{g=1}^{G} E(Y_{g;n,F}) \tag{40}$$

stand for the average (over all the generations in a run) of the expected generation-wise best-update counts. Then Theorem 2 implies that $\bar{Y}$ can be made arbitrarily small by making the total number of generations G arbitrarily large, a fact that allows us to argue that the number of best-updates per generation (or per run) should not be a concern, so far as computational costs are considered. We now consider four particular distributions for which we establish upper bounds on $E(Y_{1;n,F})$; these four cases are potential candidates for approximations to the true (unknown) distributions.

A. SPECIAL CASES

1) THE UNIFORM RANDOM DISTRIBUTION

For the Unif(a, b) distribution, the density is given by

$$f_X(x) = \frac{1}{b-a}; \quad b > a, \quad x \in [a, b] \tag{41}$$

and the corresponding cdf is

$$F_X(x) = \frac{x-a}{b-a}, \tag{42}$$

which, when used in equation 23, gives

$$E(X_{\max} \mid n, \text{Unif}_X) = \frac{a + bn}{n+1}, \tag{43}$$

from which we get

$$P(X > E(X_{\max} \mid n, \text{Unif}_X)) = \frac{1}{n+1}. \tag{44}$$

This expression allows us to obtain $E(Y_{1;n,\text{Unif}})$ from equation 31 as

$$\begin{aligned} E(Y_{1;n,\text{Unif}}) &= \sum_{j=0}^{n-1} \frac{1}{n+j+1} \\ &= H_{2n} - H_n \end{aligned} \tag{45}$$

where H_n is the n-th harmonic number [27]:

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}. \tag{46}$$

It is not difficult to prove from equation 45 that $E(Y_{1;n,\text{Unif}})$ is monotone increasing in n.

The case corresponding to an arbitrarily large n can be studied by using the fact [27] that

$$\lim_{n\to\infty} (H_n - \ln n) = \gamma, \tag{47}$$

where $\gamma \approx 0.5772$ is the Euler-Mascheroni constant. Luckily, an upper bound on $E(Y_{1;n,\text{Unif}})$ can be obtained using the following property (easily obtained from equation 47):

$$\lim_{n\to\infty} (H_{2n} - H_n) = \ln 2. \tag{48}$$

Thus, for any n, no matter how large, and any a and b

$$E(Y_{1;n,\text{Unif}}) \leq \ln 2 = 0.6931. \tag{49}$$

The minimum value of the expectation is 1/2 and corresponds to $n = 1$ (recall that the mean of the Unif$(a.b)$ distribution is $(a + b)/2$ and that the area under the pdf box to the right of $(a + b)/2$ is 1/2).

Theoretical expectations of Y_1 values corresponding to different population sizes are presented in Table 8 where the corresponding values for three other distributions are also shown.

2) THE EXPONENTIAL DISTRIBUTION

For the exponential distribution, the pdf and cdf are given by

$$f_X(x) = \lambda e^{-\lambda x}; \quad \lambda > 0,\ x \in [0, +\infty) \tag{50}$$

and

$$F_X(x) = 1 - e^{-\lambda x} \tag{51}$$

which lead to

$$\begin{aligned} E(X_{\max} \mid n, \text{Exp}_X) &= \int_{x=0}^{\infty} x \cdot n\lambda e^{-\lambda x}(1 - e^{-\lambda x})^{n-1}\, dx \\ &= \frac{1}{\lambda}\int_{v=0}^{\infty} v \cdot ne^{-v}(1 - e^{-v})^{n-1}\, dv \end{aligned} \tag{52}$$

An explicit evaluation of the above integral is rather involved; an easier approach is to use, for example, the moment generating function of $X_{\max}$ (as in [28]) and obtain the expectation of $X_{\max}$ from the derivative of the moment generating function evaluated at zero. This leads to

$$\begin{aligned} E(X_{\max} \mid n, \text{Exp}_X) &= \frac{1}{\lambda}\sum_{k=1}^{n}\frac{1}{k} \\ &= \frac{1}{\lambda}H_n \end{aligned} \tag{53}$$

We then have

$$P(X > E(X_{\max} \mid n, \text{Exp}_X)) = e^{-\lambda \times \frac{1}{\lambda}H_n} = e^{-H_n} \tag{54}$$

from which we obtain (by equation 31)

$$E(Y_{1;n,\text{Exp}}) = \sum_{j=0}^{n-1} e^{-H_{n+j}} \tag{55}$$

Now, $E(Y_{1;n,\text{Exp}})$ is monotone non-decreasing in n. Thus the smallest value of this expectation occurs at $n = 1$, and that value, from equation 55, is $1/e$ or 0.3679, a value that is corroborated by the fact that the mean of the exponential distribution is $1/\lambda$ and that the area under the pdf to the right of the point $1/\lambda$ is $e^{-\lambda\times(1/\lambda)}$ or $1/e$.

From equation 55, we have

$$\begin{aligned} \lim_{n\to\infty} E(Y_{1;n,\text{Exp}}) &= \lim_{n\to\infty}\sum_{j=0}^{n-1} e^{-\ln(n+j)-\gamma} \\ &= \frac{1}{e^{\gamma}}\lim_{n\to\infty}\sum_{j=0}^{n-1}\frac{1}{n+j} \\ &= \frac{1}{e^{\gamma}}\lim_{n\to\infty}\left(H_{2n} - H_n + \frac{1}{2n}\right) \\ &= \frac{1}{e^{\gamma}}\ln 2 \quad \text{by eq. 48} \end{aligned} \tag{56}$$

Thus, for any n and any λ,

$$E(Y_{1;n,\text{Exp}}) \leq \frac{\ln 2}{e^{\gamma}} = 0.3892. \tag{57}$$

Table 8 shows how the theoretical $E(Y_{1;n,\text{Exp}})$ varies with n, reaching the limit as n approaches infinity.

3) THE NORMAL DISTRIBUTION

The normal distribution $N(\mu, \sigma^2)$, with mean μ and variance σ^2, and density

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}; \quad \sigma > 0, \quad x \in (-\infty, +\infty) \tag{58}$$

admits of no closed-form expression for $E(X_{\max} \mid n, F_X)$, and the integration in equation 23 must be evaluated numerically. We find $E(Y_{1;n,\text{Norm}})$ numerically, from equation 31 (using numerical routines from Python's *scipy* [29], [30] and also from Mathematica [31]). The numerically obtained Y_1 values corresponding to different population sizes are presented in Table 8. Note that Y_1 for the Gaussian is monotone non-increasing with n. Thus the maximum possible value of $E(Y_{1;n,\text{Norm}})$ is obtained at the smallest possible value of n, namely 1, and the corresponding $E(X_{\max})$ is clearly the mean, μ, of the distribution, which, because of symmetry (the mean equals the median), causes $P(X > E(X_{\max} \mid 1, \text{Norm}_X))$ to be 0.5, leading to $E(Y_{1;1,\text{Norm}}) = 0.5$. Thus, for any n, no matter how large, and any μ and σ, $E(Y_{1;n,\text{Norm}})$ is upper-bounded by

$$E(Y_{1;n,\text{Norm}}) \leq 0.5. \tag{59}$$

The limit of the expectation, as $n \to \infty$, cannot be obtained analytically.

4) THE LOGISTIC DISTRIBUTION

The logistic distribution offers some similarity (e.g., unimodality, symmetry) to the normal. That, coupled with the fact that it is amenable to analytical treatment, affords an alternative to the normal distribution for modeling purposes. For simplicity, let us consider location and scale parameters of 0 and 1, respectively. This does not cause any loss of generality, because any logistic variable X with location a and scale $s > 0$ can be transformed to another logistic variable Z:

$$Z = \frac{X - a}{s}. \tag{60}$$

The cdf and pdf of the logistic distribution are given by

$$F_X(x) = \frac{1}{1 + e^{-x}}; \quad x \in (-\infty, +\infty) \tag{61}$$

and

$$f_X(x) = F_X(x)(1 - F_X(x)) = \frac{e^{-x}}{(1 + e^{-x})^2}. \tag{62}$$

Thus

$$E(X_{\max} \mid n, \text{Logistic}_X) = \int_{x=-\infty}^{\infty} \frac{n \cdot x \cdot e^{-x}}{(1 + e^{-x})^{n+1}}\, dx \tag{63}$$

from which an explicit expression for the expected $X_{\max}$ can be obtained by the moment-generating-function technique (see, e.g., [28]):

$$E(X_{\max} \mid n, \text{Logistic}_X) = H_{n-1}, \tag{64}$$

with H_0 taken to be zero (recall that $n \geq 1$). Then

$$P(X > E(X_{\max} \mid n, \text{Logistic}_X)) = \frac{e^{-H_{n-1}}}{1+e^{-H_{n-1}}} \tag{65}$$

and, by equation 31,

$$E(Y_{1;n,\text{Logistic}}) = \sum_{j=0}^{n-1} \frac{e^{-H_{n+j-1}}}{1 + e^{-H_{n+j-1}}}. \tag{66}$$

$E(Y_{1;n,\text{Logistic}})$ is monotone non-increasing in n. Thus the largest value of the expectation occurs at $n = 1$, and that expectation is obtained from equation 66 as 0.5, a value that is corroborated by the symmetric nature of the distribution. Now, using the large-n approximation to H_n, namely $\lim_{n\to\infty} H_n = \ln n + \gamma$ (recall equation 47), we have

$$\begin{aligned}\lim_{n\to\infty} E(Y_{1;n,\text{Logistic}}) &= \lim_{n\to\infty} \sum_{j=0}^{n-1} \frac{e^{-\ln(n+j-1)-\gamma}}{1 + e^{-\ln(n+j-1)-\gamma}} \\ &= \frac{1}{e^{\gamma}} \lim_{n\to\infty} \sum_{j=0}^{n-1} \frac{1}{n + j - 1 + e^{-\gamma}} \\ &= \frac{1}{e^{\gamma}} \ln 2 \\ &= 0.3892. \end{aligned} \tag{67}$$

TABLE 8. Theoretical growth (or decay) of $E(Y_{1;n,F})$ with n.

n	$E(Y_{1;n,F})$			
	$F = \text{Exp}(\lambda)$	$F = \text{Logistic}(0,1)$	$F = N(\mu, \sigma^2)$	$F = \text{Unif}(a, b)$
1	0.3679	0.5	0.5	0.5
10	0.3889	0.4016	0.4451	0.6688
50	0.3892	0.3916	0.4261	0.6882
100	0.3892	0.3904	0.4212	0.6907
500	0.3892	0.3894	0.4136	0.6926
5000	0.3892	0.3892	0.4074	0.6931
10000	0.3892	0.3892	0.4061	0.6931
∞	0.3892	0.3892	—	0.6931

TABLE 9. $E_{\text{empir}}(Y_{g;n})$ for $g = 1, 10, 20$ (number of runs = 500).

Function	n	$E_{\text{empir}}(Y_{g;n})$			$E_{\text{empir}}(Y_n)$
		Gen 1	Gen 10	Gen 20	
Ackley	10	0.916	0.484	0.452	0.5276
	50	0.488	0.278	0.184	0.2882
	100	0.352	0.216	0.182	0.2261
	1000	0.130	0.156	0.122	0.1495
Rosenbrock	10	0.650	0.396	0.404	0.445
	50	0.254	0.230	0.186	0.2275
	100	0.206	0.168	0.112	0.1763
	1000	0.052	0.104	0.090	0.0868
Chung-Reynolds	10	0.854	0.46	0.49	0.5353
	50	0.418	0.232	0.192	0.2852
	100	0.300	0.174	0.124	0.2148
	1000	0.084	0.146	0.148	0.1386
Step	10	0.904	0.506	0.516	0.5858
	50	0.468	0.276	0.236	0.3116
	100	0.400	0.222	0.154	0.2614
	1000	0.148	0.204	0.106	0.1836
Goldstein-Price	10	0.600	0.170	0.076	0.208
	50	0.610	0.138	0.058	0.196
	100	0.620	0.142	0.009	0.1942
	1000	0.588	0.150	0.074	0.1802
PEMFC	10	1.132	0.724	0.680	0.7421
	50	1.608	0.436	0.428	0.5897
	100	1.572	0.390	0.240	0.5548
	1000	1.566	0.152	0.164	0.4539

B. EMPIRICAL RESULTS

Empirical values of the average counts of best-updates are obtained by aggregating independent SJaya runs for each of the test problems. For a population size of n, the empirical expectation (average) at a given generation g is produced from an ensemble of r runs as:

$$E_{\text{empir}}(Y_{g;n}) = \frac{1}{r} \sum_{k=1}^{r} N_k(g) \tag{68}$$

where $N_k(g)$ is the number (an integral count ≥ 0) of updates of the best-index at generation g in run k. The corresponding average (over all generations) is obtained as

$$E_{\text{empir}}(Y_n) = \frac{1}{G} \sum_{g=1}^{G} E_{\text{empir}}(Y_{g;n}). \tag{69}$$

In the above two equations, $g \geq 1$ ($g = 0$ represents the initial population). Table 9 shows, for different population sizes, $E_{\text{empir}}(Y_n)$ values as well as how $E_{\text{empir}}(Y_{g;n})$ changes with generations ($r = 500$ and $G = 20$ in this table). The runs used in this table are the same as the ones used in Table 5.

V. COMPUTATIONAL COSTS

While the Jaya algorithm finds the best-of-population member and the worst-of-population member exactly once per generation, SJaya does this on a continuous, as-needed basis. The logic for finding the best (or worst) of a given number of elements can be implemented as a simple sequential scan of the elements, consisting of two basic operations for each element: a comparison followed, conditionally, by an assignment. We now find the costs of these two types of operations.

A. COMPARISON AND ASSIGNMENT OPERATIONS FOR BEST-INDEX UPDATE IN SJaya

The total number of comparison operations needed for updating the best index in an entire generation of SJaya (call this number c) is equal to the number of times line 9 in Algorithm 2 is executed (i.e., the condition in line 9 is tested) per generation (this number is the same as the number of times the condition in line 7 evaluates to TRUE in a generation). We need to find the expected value of c.

We can model the new individual being at least as good as the current individual (line 7) by the event $X_2 \geq X_1$, where X_1 and X_2 are two independent random samples drawn from the same distribution.

Defining a random variable

$$Z = 1_{X_2 \geq X_1}, \tag{70}$$

we have

$$Z = \begin{cases} 1 & \text{with probability } P(X_2 \geq X_1) \\ 0 & \text{with probability } 1 - P(X_2 \geq X_1). \end{cases} \tag{71}$$

Thus

$$E(Z) = P(X_2 \geq X_1), \tag{72}$$

and the expectation of the number of times the condition in line 7 evaluates to TRUE in a generation is given by

$$\begin{aligned} E(c) &= E\left(\sum_{i=1}^{n} Z_i\right) \\ &= \sum_{i=1}^{n} E(Z_i) \quad \text{(by linearity)} \\ &= n\,E(Z) \end{aligned} \tag{73}$$

where the last step follows from the fact that the events at the n slots of the population are governed by the same underlying distribution. Thus

$$E(c) = n\,P(X_1 \leq X_2), \tag{74}$$

which, by the law of total probability, gives

$$\begin{aligned} E(c) &= n \int_{x=-\infty}^{\infty} P(X_1 \leq X_2 | X_2 = x) f_{X_2}(x)\,dx \\ &= n \int_{x=-\infty}^{\infty} P(X_1 \leq x) f_{X_2}(x)\,dx \\ &= n \int_{x=-\infty}^{\infty} F_{X_1}(x) f_{X_2}(x)\,dx \\ &= n \int_{x=-\infty}^{\infty} F_X(x) f_X(x)\,dx \\ &= \frac{n}{2} \end{aligned} \tag{75}$$

Next, the total number of assignment operations (call it a) needed for updating the best index in an entire generation of SJaya is equal to the number of times line 10 is executed per generation in Algorithm 2. The expectation of this count, $E(a)$, was already derived in Section IV; $E(a)$ can be taken to be either the generation-specific $E(Y_{g;n,F})$ or the average $\bar{Y}$. This expectation is obviously a function of the population size n, and Section IV obtained the maximum value of this expectation (corresponding to either $n = 1$ or $n \to \infty$, depending on the nature of the distribution) for specific distributions, as follows:

$$\text{Max. of } E(a) = \begin{cases} \dfrac{\ln 2}{e^{\gamma}} & \text{for exponential distribution} \\ \ln 2 & \text{for uniform distribution} \\ 0.5 & \text{for normal distribution} \\ 0.5 & \text{for logistic distribution.} \end{cases} \tag{76}$$

It is difficult to obtain a closed-form analytical expression of this expectation for arbitrary distributions; however, by Theorem 2, this expectation, when averaged over a number of generations, goes down as the number of generations increases, regardless of the underlying distribution.

B. COMPARISON AND ASSIGNMENT OPERATIONS FOR FINDING THE BEST/WORST IN THE NAIVE APPROACH

The naïve approach to sequentially scanning an array for finding the best (or worst) element entails exactly n (or $n - 1$, depending on the implementation) comparisons:

$$c_{\text{naive}} = n. \tag{77}$$

The number of assignments, a_{naive}, however, is not deterministic. Assuming the array index runs from 1 to n, the number of assignments can go from a minimum of 1 to a maximum of k (or from 1 to $n - k + 1$, depending on the implementation), inclusive, when the best (or worst) element is located at index k. The average-case analysis can be performed by noting that when the numbers are uniformly distributed in the array, the j-th element is greater (smaller) than the preceding $j - 1$ elements with probability $1/j$, independently for all $j = 1, \cdots, n$. Thus

$$E(a_{\text{naive}}) = \sum_{j=1}^{n} \frac{1}{j} = H_n \tag{78}$$

C. COMPLEXITY OF SJaya

Given the analyses of the preceding sections, it is now straightforward to obtain the complexity of SJaya. The cost of the initialization step (line 1 in Algorithm 2) is $n \times \phi(d)$, where $\phi(d)$ is the cost of evaluating the fitness (objective function) of a given problem of d dimensions. The cost of finding the best/worst member in the entire population (line

2) is $E(c_{\text{naive}}) \times C_c + E(a_{\text{naive}}) \times C_a$ or $n \times C_c + H_n \times C_a$, where C_c and C_a are the cost of a single comparison and a single assignment, respectively. Setting the random parameters for the solution vector (line 4) has a cost of $C_p \times d$, where C_p is a constant. Creating a single new individual (line 6) incurs a cost of $C_{\text{op}} \times d + \phi(d)$, where C_{op} represents the cost per dimension of applying the algebraic operations involved in the creation of a new individual. Each check for the superiority of the new individual (line 7) costs C_c, and there are n such checks in a generation. The replacement at line 8 takes place $E(c)$ times in a whole generation (recall that the condition at line 7 is true these many times on average in a generation). Again, the condition in line 9 is tested $E(c)$ times in a whole generation. And, as already shown in Sec. V-A, the update in line 10 occurs $E(a)$ times per generation. The condition in line 12 is tested $E(c)$ times in a generation, and finding the worst individual in line 13 is needed a maximum of 1.7 times per generation. The total cost of a single run is thus

$$n\phi(d) + 2(nC_c + H_nC_a) + G\Big[C_pd + n(C_{\text{op}}d + \phi(d)) + n\,C_c + (n/2)\,C_a + (n/2)\,C_c + E(a)\,C_a + (n/2)\,C_c + 1.7(nC_c + H_nC_a)\Big].$$

D. COMPLEXITY OF Jaya

The complexity of Jaya can now be derived easily. Most of the calculations carry over from those of SJaya. Noting that the replacement of the existing individual with the new one (in line 8 of Algorithm 1) takes place $E(c)$ times in an entire generation, the complexity is given by

$$n\phi(d) + G\Big[2(nC_c + H_nC_a) + C_pd + n(C_{\text{op}}d + \phi(d)) + n\,C_c + (n/2)\,C_a\Big].$$

E. COST DIFFERENCE BETWEEN SJaya AND Jaya

Using c as an upper bound of a, and with $E(c) = n/2$, we obtain the following estimate of an upper bound of the additional cost incurred by SJaya over Jaya per generation:

$$\begin{aligned}\text{additional cost} &\le \frac{n}{2}(C_a + 2C_c) - 0.3(nC_c + H_nC_a)\\ &\approx (0.5n - 0.3\ln n - 0.17316)C_a + 0.7\,nC_c\\ &\quad \text{for large } n. \end{aligned} \tag{79}$$

This additional cost is not significant compared to the total cost of evaluating the fitnesses of the n population members in a generation. If, in light of the analysis in Sec. IV, a more realistic value of $E(a) \ll n$ is assumed, the additional cost becomes even lower.

VI. CONCLUSION

Stochastic models were developed for Jaya and its recent improvement SJaya, and a detailed theoretical analysis presented. An interesting fact revealed by the analysis is that the maximum expected number of worst-index updates per generation for SJaya is upper-bounded by $e - 1$, or 1.7, for any population size. Furthermore, regardless of the population size, the expectation of the number of best-index updates per generation decreases monotonically with generations. We derived exact upper bounds of the expected number of best-index updates when the underlying distribution is exponential, logistic, normal or uniform. Asymptotics of expected best-update counts were obtained for exponential, logistic and uniform distributions. Limitations of the analytical approach and the need to resort on occasion to numerical techniques were pointed out. The model allowed us to obtain computational complexities of the algorithms, which showed that the performance improvement afforded by SJaya over Jaya incurs only a modest additional cost. Empirical results on benchmark test problems were obtained and found to corroborate the general trends of the theoretical findings. To our knowledge, this is the first theoretical analysis of this popular family of heuristics. The analytical approach developed here has the potential to be extended to the analysis of other types of evolutionary algorithms. The insights provided by the models should help design new, improved population-based search / optimization heuristics.

ACKNOWLEDGMENT

Thanks to Adrian Clingher for his help with the derivation of the second identity (eq. 14) used in the proof of Theorem 1. Mike Toohey provided help over a weekend, scheduling servers to accommodate my CPU-intensive jobs. Four anonymous reviewers provided useful comments.

NOMENCLATURE

Subscripted and superscripted versions of the same symbol are not included in this list; the symbol itself is listed once. Also excluded are the notations used in the fuel cell problem (Sec. III-B1).

Symbol	Meaning
a	Parameter of the uniform density; also, number of assignment operations.
b	Parameter of the uniform density.
c	Number of comparison operations.
C	Computational cost associated with a single application of an operation (different subscripts are used for different types of cost).
d	Problem dimensions.
e	The Euler constant.
E	Expectation.
$f()$	Fitness (objective) function.
$f_X()$	Probability density function of random variable X.
$F_X()$	Cumulative distribution function of random variable X.
G	Number of generations.
H_n	n-th harmonic number.
g, h, i, j, k, m, q	Indices.
n	Population size.
p	Worst-replacement probability.
P	Probability.

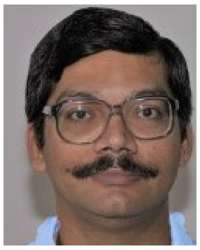
r	Random coefficients used in Jaya/SJaya; also, number of runs.
$\mathbb{R}$	The real domain.
t	Number of iterations.
W	Index of the worst individual.
x	Problem parameter vector (chromosome); also, an instance (observation) of a random variable.
x^*	Optimal problem parameter vector.
X	Random variable (discrete) for the number of times the worst individual needs to be found in a generation; also, a generic random variable (continuous).
Y	Random variable (discrete) for the number of updates of best-index.
$\bar{Y}$	Average expected generation-wise best-update count.
Z	Random variable (used in different contexts).
γ	Euler-Mascheroni constant.
Θ	Asymptotic computational complexity notation.
λ	Parameter of the exponential density.
μ	Mean of the normal density.
σ	Standard deviation of the normal density.
$\phi()$	Cost of evaluating the objective function.

CONFLICT OF INTEREST STATEMENT

The author declares no conflict of interest.

REFERENCES

[1] R. V. Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems," *Int. J. Ind. Eng. Comput.*, vol. 7, no. 1, pp. 19–34, 2016.
[2] R. V. Rao, *Jaya: An Advanced Optimization Algorithm and its Engineering Applications*. Berlin, Germany: Springer, 2019.
[3] R. V. Rao, A. Saroj, P. Oclon, J. Taler, and J. Lakshmi, "A posteriori multi-objective self-adaptive multipopulation Jaya algorithm for optimization of thermal devices and cycles," *IEEE Access*, vol. 7, pp. 4113–4134, 2018.
[4] R. V. Rao, A. Saroj, J. Taler, and P. Oclon, "Multi-objective design optimization of shell-and-tube heat exchanger using multi-objective SAMP-Jaya algorithm," in *Advanced Engineering Optimization Through Intelligent Techniques* (Advances in Intelligent Systems and Computing). Singapore: Springer, 2020, pp. 831–838.
[5] S. Tripathy, M. K. Debnath, and S. K. Kar, "Jaya algorithm tuned FO-PID controller with first order filter for optimum frequency control," in *Proc. 1st Odisha Int. Conf. Electr. Power Eng., Commun. Comput. Technol. (ODICON)*, Jan. 2021, pp. 1–6.
[6] R. Yadav and R. Panwar, "Effective medium approximation fused optimization strategy derived new kind of honeycomb microwave absorbing structure," *IEEE Trans. Magn.*, vol. 58, no. 3, pp. 1–11, Mar. 2022.
[7] N. Agarwal, M. K. Pradhan, and N. Shrivastava, "A new multi-response Jaya algorithm for optimisation of EDM process parameters," *Mater. Today, Proc.*, vol. 5, no. 11, pp. 23759–23768, 2018.
[8] S. Gupta, N. Kumar, and L. Srivastava, "An efficient Jaya algorithm with Powell's pattern search for optimal power flow incorporating distributed generation," *Energy Sources, B, Econ., Planning, Policy*, vol. 16, no. 8, pp. 759–786, Aug. 2021.
[9] U. K. Chakraborty, "Proton exchange membrane fuel cell stack design optimization using an improved Jaya algorithm," *Energies*, vol. 12, no. 16, Aug. 2019, Art. no. 3176.
[10] S. C. Satapathy and V. Rajinikanth, "Jaya algorithm guided procedure to segment tumor from brain MRI," *J. Optim.*, vol. 2018, pp. 1–12, Nov. 2018.
[11] P. Mohapatra, R. Mishra, and T. K. Patra, "A Jaya algorithm trained neural network for stock market prediction," *Int. J. Innov. Technol. Exploring Eng.*, vol. 7, pp. 9–13, Aug. 2018.
[12] M. Gunduz and M. Aslan, "DJAYA: A discrete Jaya algorithm for solving traveling salesman problem," *Appl. Soft Comput.*, vol. 105, Jul. 2021, Art. no. 107275.
[13] S. Belagoune, N. Bali, K. Atif, and H. Labdelaoui, "A discrete chaotic Jaya algorithm for optimal preventive maintenance scheduling of power systems generators," *Appl. Soft Comput.*, vol. 119, Apr. 2022, Art. no. 108608.
[14] Y. R. Naidu, A. Ojha, and V. S. Devi, "Multi-objective Jaya algorithm for solving constrained multi-objective optimization problems," in *Proc. Int. Conf. Harmony Search Algorithm*. Cham, Switzerland: Springer, 2019, pp. 89–98.
[15] R. A. Zitar, M. A. Al-Betar, M. A. Awadallah, I. A. Doush, and K. Assaleh, "An intensive and comprehensive overview of Jaya algorithm, its versions and applications," *Arch. Comput. Methods Eng.*, vol. 28, pp. 1–30, Mar. 2021.
[16] U. K. Chakraborty, "Semi-steady-state Jaya algorithm for optimization," *Appl. Sci.*, vol. 10, no. 15, Aug. 2020, Art. no. 5388.
[17] D. Ackley, *A Connectionist Machine for Genetic Hillclimbing*. Berlin, Germany: Springer, 2012.
[18] H. H. Rosenbrock, "An automatic method for finding the greatest or least value of a function," *Comput. J.*, vol. 3, no. 3, pp. 175–184, 1960.
[19] C.-J. Chung and R. G. Reynolds, "CAEP: An evolution-based tool for real-valued function optimization using cultural algorithms," *Int. J. Artif. Intell. Tools*, vol. 7, no. 3, pp. 239–291, Sep. 1998.
[20] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimisation problems," *Int. J. Math. Modeling Numer. Optim.*, vol. 4, no. 2, pp. 150–194, 2013.
[21] A. A. Goldstein and J. F. Price, "On descent from local minima," *Math. Comput.*, vol. 25, no. 115, pp. 569–574, Jul. 1971.
[22] J. Larminie, A. Dicks, and M. S. McDonald, *Fuel Cell Systems Explained*, vol. 2. Chichester, U.K.: Wiley, 2003.
[23] R. O'Hayre, S.-W. Cha, W. Colella, and F. B. Prinz, *Fuel Cell Fundamentals*, 3rd ed. Hoboken, NJ, USA: Wiley, 2016.
[24] I. Mohamed and N. Jenkins, "Proton exchange membrane (PEM) fuel cell stack configuration using genetic algorithms," *J. Power Sources*, vol. 131, nos. 1–2, pp. 142–146, May 2004.
[25] G. J. Besseris, "Using qualimetric engineering and extremal analysis to optimize a proton exchange membrane fuel cell stack," *Appl. Energy*, vol. 128, pp. 15–26, Sep. 2014.
[26] U. K. Chakraborty, "A new model for constant fuel utilization and constant fuel flow in fuel cells," *Appl. Sci.*, vol. 9, no. 6, Mar. 2019, Art. no. 1066.
[27] D. E. Knuth, *Art of Computer Programming, The: Volume 1: Fundamental Algorithms*, 3rd ed. Reading, MA, USA: Addison-Wesley, 1968.
[28] E. J. Gumbel, *Statistics of Extremes*. Mineola, NY, USA: Dover, 2004.
[29] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, and S. J. Van Der Walt, "Scipy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, pp. 261–272, Mar. 2020.
[30] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, and R. Kern, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020.
[31] *Mathematica, Version 12.3*, WR. Inc, Champaign, IL, USA, 2021.



UDAY KUMAR CHAKRABORTY is currently a Professor of computer science at the University of Missouri—St. Louis, USA. Before joining UMSL in 2001, he was a Senior Research Associate at CAD Center, Calcutta, a Systems Engineer at CMC Ltd., Calcutta and London, a Research Scientist at GMD Forschungszentrum Informationstechnik (German National Research Center for Computer Science; now Fraunhofer Institute), Sankt Augustin, and an Associate Professor of computer science and engineering at Jadavpur University, Calcutta. His research interests include evolutionary computation, machine learning, soft computing, and fuel cell modeling and simulation.