Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-10 (2022) 702–707

Hermit Crab Optimizer (HCO): A Novel Meta-heuristic Algorithm

Keivan Tafakkori. Reza Tavakkoli-Moghaddam

School of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran (Tel: +98 21 88021067; e-mails: {k.tafakkori; tavakoli}@ut.ac.ir)

Abstract: This paper proposes a novel meta-heuristic algorithm (MA), called hermit crab optimizer (HCO), which simulates the swarm intelligence of hermit crabs in nature in finding shells protecting and letting them grow in their lifetime. HCO guides search agents separately and in parallel using new solitary and social search operators. It acts similar to a reinforcement learning process, in which the successful agents and failed ones are treated differently, inspired by the group behavior of hermit crabs and environmental characteristics. Computational experiments with well-known test problems confirm HCO's validity, accuracy, robustness, ability to escape local optima, and balance exploration-exploitation.

Keywords: Optimization, Meta-heuristics, Reinforcement learning, Nature-inspired, Swarm intelligence.

1 INTRODUCTION

In many real-world scenarios, the size of interest (i.e., scale) for optimization, machine learning, and simulation tasks makes using exact classical methods infeasible. There may be no *demonstrable advantage* for the sophistication of logic and using classical exact algorithms, especially when computers can repeatedly do trial-and-error experiments. While decent exact algorithms efficiently handle real-world applications for numerical optimization problems, they rarely handle combinatorial ones. These problems contain many binary/integer variables, constraints, and objectives and are computationally complex. Although the advent of quantum computing in 1980, which uses qubits (working by probabilities and multiple states) instead of bits (working by 0s and 1s and binary states), has helped to solve BQP (bounded-error quantum polynomial time) problems in addition to P class problems, yet problems of NP and NP-Complete classes cannot be solved in polynomial time. Moreover, the expenses of establishing and operating quantum computing infrastructures are currently too high. At the same time, some data could still be confidential, limiting processing by quantum computing and cloud infrastructure, although security is highly ensured using blockchain technology.

Accordingly, competitions for digital transformation in the industry 4.0 era are motivating the researchers to propose better computational algorithms, mainly called *meta-heuristics*, which do not necessarily have a solid theoretical background but are runnable in a reasonable time on personal computers, can derive sufficiently good solutions faster, and are general problem-solvers (GPSs). However, global optimality and sometimes feasibility of the solutions may get compromised, which is a motivation for academic research (i.e., to improve how a solution is derived, in terms of feasibility, solution time, accuracy, and robustness). This improvement is usually confirmed by comparisons on a set of the same test problems, solved by famous algorithms proposed by previous literature. However, according to the *No Free Lunch* (NFL) theorem, there is still no meta-heuristic algorithm (MA) superior to the others in all types of problem classes and types (Wolpert and Macready, 1997).

All MAs or artificially intelligent algorithms (AIAs) (Tafakkori, 2022) do a sequential examination of multiple agents moving in the search space (similar to robots in the physical world), which involves a comparison of each agent with the best agent so far or its previous state together with a strategy (or action, as a function of earlier results) for determining the next state in which each agent will be. Those agents who are successful and better than their previous state (regarding a reward function) get a reward (e.g., remaining as it is, being in the elite set, and being treated differently), and the others may be punished (removed from the population set, or forced to be guided by elites). Accordingly, MAs act similar to a reinforcement learning (RL) algorithm. The policy is to select or be forced to undergo a single or a set of the heuristic operator(s) (as actions) for agents in each state of sequential evolution toward the optimal solution. Overall, in an MA, agents are not guided or supervised by a gradient but change their state on the landscape of an optimization problem according to a policy, set of actions (operators), and a reward function in a sequence of iterations, either randomly or by a model, to be more explainable (Sutton and Barto, 2018).

Maybe the initial study on MAs was conducted by Hooke and Jeeves (1961), who proposed the Pattern Search (Direct Search) algorithm. However, the term *meta-heuristic* was coined by Glover (1986). The most popular MAs from 1961 to 2021 (according to number of citations) are Genetic Algorithm (GA) (1975), Simulated annealing (SA) (1983), Tabu Search (TS) (1986), Greedy Random-

10.1016/j.ifacol.2022.09.488

ized Adaptive Search Procedure (GRASP) (1989), Genetic Programming (GP) (1990), Particle Swarm Optimization (PSO) (1995), Differential Evolution (DE) (1997), Harmony Search (HS) (2001), Bacterial Foraging Optimization (BFO) (2002), Iterated Local Search (ILS) (2003), Artificial Bee Colony (ABC) (2005), Big Bang-Big Crunch (BB-BC) (2006), Imperialist Competitive Algorithm (ICA) (2007), Firefly Algorithm (FA) (2008), Gravitational Search Algorithm (GSA) (2009), Bat Algorithm (BA) (2010), Brain Storm Optimization (BSO) (2011), Flower Pollination Algorithm (FPA) (2012), Dolphin Echolocation (DE) (2013), Grey Wolf Optimizer (GWO) (2014), Ant Lion Optimizer (ALO) (2015), Whale Optimization Algorithm (WOA) (2016), Salp Swarm Algorithm (SSA) (2017), Moth Search Algorithm (MSA) (2018), Harris Hawks Optimization (HHO) (2019), Slime Mould Algorithm (SMA) (2020), and Emperor penguin and Salp swarm Algorithm (ESA) (2021). It can be concluded that researchers have been interested in using a source of inspiration (e.g., nature, evolution, physics, game, psychology, and biology) to develop new MAs which can perform better in more test problems (also see (Hajipour et al., 2014; Fathollahi-Fard et al., 2020, 2018; Kashan et al., 2019; Amindoust et al., 2021)).

Regarding the discussed issues and the need for high-quality algorithms, emphasized since 1961 and revisited since 2006 (the industry 4.0 era), herein, the motivation is to propose a new MA, called HCO, which differs from its previous counterparts, both in structure, and the quality of the solutions. The HCO is built according to the inspiration from the natural shell searching and exchanging behavior of hermit crabs in their habitat. This algorithm only requires three inputs: population size (N), elitism number (E), and maximum number of iterations (I) (as a termination condition). The HCO is a population-based MA that starts with multiple initial random solutions, parallelizes the population to apply search operators, and aids search agents in adapting to the search space (i.e., landscape or response surface) through their evolution (exploration, exploitation, and escaping from local optima) by control parameters. The search operators are solitary (asynchronous) or social (synchronous) heuristics. The HCO is implemented on a set of well-known test problems compared with its counterparts.

2 HERMIT CRAB OPTIMIZER (HCO)

In this section, the HCO is developed, and its inspiration source is discussed.

Fig. 1. A hermit crab (*Paguroidea*).

2.1 Inspiration

Hermit crabs (*Paguroidea*), anomurans, and decapod crustaceans with approximately 1100 species. Irrespective of living as a hermit, these crabs are social creatures that usually live in groups or herds of 100 or more. The name *hermit* comes from the fact that they typically carry and live in a mobile shelter called *shell* as an exoskeleton (carapace) to protect their soft abdomen since they cannot generate it by themselves (see Fig. 1).

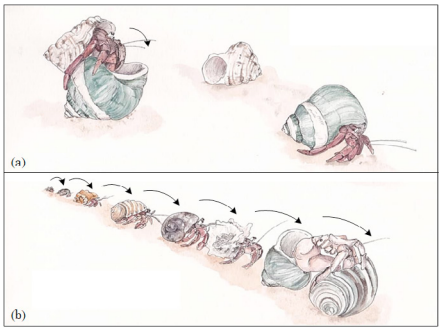

Fig. 2. Shell searching actions: (a) Asynchronous cue search (solitary search) and (b) Synchronous vacancy chain creation (social search).

A hermit crab usually walks around the habitat and samples visual and chemical cues emitted from empty gastropod shells, dying snails, or dead conspecifics in a solitary or asynchronous strategy (Fig. 2a). New shells enter the habitat when predators, disease, or environmental conditions kill gastropods or conspecifics. Odors of dead conspecifics signal already prepared homes for them (without a need for reshaping their structure). Therefore, these odors are more alluring than others. Sources that provide the most attractive cues are mostly followed. With the motivation to find a suitable shell, a swarm appears close to each odor source and tests the new shell for suitability. Even though hermit crabs encounter multiple shells throughout their lifetime, some may not meet their requirements. The solitary search seems inefficient because individual crabs may have a low chance of finding an empty shell that meets their needs (Gravel et al., 2004).

Hermit crabs that fail to find a new shell create a network to exchange shells in a social or asynchronous strategy (Fig. 2b). There may be situations in which some hermit crabs will find the desired shell too unsuitable (e.g., large), even after their solitary search tries. The hermit crabs (with smaller shells than the found one) usually wait and aggregate around a hermit crab (with a size closer to the found shell) to occupy its shell. Processes called *negotiation* and *aggression* are typical in this regard (Rotjan et al., 2010). In aggression, the dominant (*attacker*) (i.e., the bigger crab) exchanges its shell with the subordinate (*defender*) (i.e., the smaller crab). The attacker rapidly moves toward the defender and raps its shell, causing it to depart its house. Contrarily, all crabs will be aligned with the crab that finds the found shell suitable in a negotiation. In this process, the gigantic crab leaves its shell behind for the smaller one, as it may discover a more suitable shell from larger crabs. The social search strategy is more common and is called as *vacancy chain* process because of similarities with humans' behavior when getting promoted and other people acquiring their position (Rotjan et al., 2010).

2.2 Algorithmization

Herein, the shell searching behavior of hermit crabs as agents in the habitat (environment or landscape) is simulated to guide them in finding better shells as they grow. Next, the HCO is outlined by a pseudocode and flow chart.

2.2.1 Representation A state (population) matrix $\mathbf{P}$ of search agents in each iteration exists and is updated (if hermit crabs successfully find their desirable shell) or transferred to the next generation (otherwise). $\mathbf{P}$ is created as:

$$\mathbf{P}(i) = \begin{bmatrix} x_1^{1i} & \cdots & x_d^{1i} \\ \vdots & \ddots & \vdots \\ x_1^{ni} & \cdots & x_D^{Ni} \end{bmatrix}_{N\times D} = \begin{bmatrix} \overrightarrow{X}_1(i) \\ \vdots \\ \overrightarrow{X}_N(i) \end{bmatrix}_{N\times 1} \quad (1)$$

where N is the total number of search agents (population size) (indexed by n), and D is the total number of encoded values (features or dimensions) for a search agent (indexed by d). Accordingly, x denotes the location (state) of search agent n in dimension d. Notably, $\overrightarrow{X}$ is a feature vector representing the state of search agent n, and i is an indicator for iteration (at max equals I).

2.2.2 Initialization The initial population of search agents (i.e., elements of $\mathbf{P}(0)$) is generated randomly using the *pseudo-random number generator* of the programming language by which the HCO is coded:

$$x_d^{n0} = x_d^- + \alpha\left(x_d^+ - x_d^-\right), \quad 1 \le d \le D, 1 \le n \le N \quad (2)$$

where x_d^-, x_d^+ represent a lower and upper bound on values that d-th feature of each search agent n takes, and α is a pseudo random number in range $[0, 1]$. After initialization ($i = 0$), the main loop of the algorithm starts ($1 \le i \le I$).

2.2.3 Preprocessing Each of the search agents with features (x_d^{ni}) violating min/max boundaries (i.e., x_d^- or x_d^+) are fixed by setting their value equal to the closest bound. Three controlling parameters, namely a (total shells found), b (perceived risk of shell availability (Briffa and Bibost, 2009)), and c (distraction indicator or wind effect, impacting diffusion of odor molecules), are initiated as zero.

2.2.4 Evaluation Each search agent is associated with a value (e.g., objective or reward value) and featured by vector $\overrightarrow{X}$ decoded (or mapped) into the solution space. Therefore, we consider decoder a function, namely g, which results in a meaningful solution for the optimization problem. In the evaluation phase, $\mathbf{P}$ is merged with a new feature representing the objective values gained by search agents (as a reward):

$$\mathbf{\Pi}(i) = [\mathbf{P}(i), \mathbf{R}(i)] = \begin{bmatrix} \overrightarrow{X}_1(i) & g\left(\overrightarrow{X}_1(i)\right) \\ \vdots & \vdots \\ \overrightarrow{X}_N(i) & g\left(\overrightarrow{X}_N(i)\right) \end{bmatrix}_{N\times(1+1)} \quad (3)$$

where $\mathbf{\Pi}$ is a state-reward matrix, $\mathbf{P}$ is a state matrix showing the position of search agents, and $\mathbf{R}$ is a reward matrix gained by each agent in a state, all at iteration i. It should be emphasized that, by state, we mean the position of the search agents in the search space, which has a specific reward (objective values calculated by decoder function g). Notably, we do not store the actual solutions in the solution space, as we have a decoder, which is used to define the final solution for the user.

2.2.5 Comparison In this stage, rewards gained during the agents' new state evaluation are compared with the previous rewards in the earlier state. If the reward is better than the previous state (according to the direction of optimization), then the search agent updates its state and reward. This agent is labeled as *successful* ($f(n) = 0$) and the total shells found are summed by 1 ($a = a + 1$). The optimal solution (so far) is updated if an agent gains a reward higher than all agents' rewards in the previous state. Accordingly, the wind (distraction) indicator (c) is set to 1 (i.e., the wind is not distracting search agents from following best odor sources). If these cases do not occur, then the agent keeps its current state while being labeled as *failed* ($f(n) = 1$).

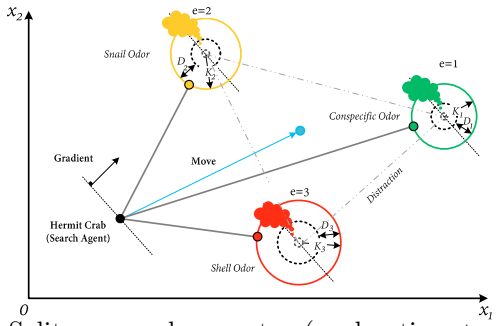

Fig. 3. Solitary search operator (exploration + exploitation).

2.2.6 Postprocessing In this stage, the perceived risk of shell availability (b) is calculated as:

$$b = 1 - a/N \quad (4)$$

2.2.7 Solitary search operator This stage is only pursued for those search agents with label *successful* (i.e., n does not count till N). Fig. 3 shows how a search agent updates its position (moves) in a 2D search space. As mentioned earlier, hermit crabs tempt to follow strong chemical cues in their habitat in a solitary search for shells. An operator should be defined to mimic this behavior, considering the diffusion of odor molecules from a point in the habitat. Accordingly, the solitary search operator is defined as follows:

$$\overrightarrow{X}_n(i+1) = \frac{1}{E}\sum_{e=1}^{E}\left(\overrightarrow{X}'_e(i) + \gamma_e\left|\delta_1\overrightarrow{X}'_e(i) - \overrightarrow{X}_n(i)\right|\right) \quad (5)$$

where $\overrightarrow{X}_n(i+1)$ and $\overrightarrow{X}_n(i)$ represents state (position) of n-th successful search agent in two sequential iterations, $\overrightarrow{X}'_e(i)$ is the perceived position of the E (elitism number) most dominant tempting odor sources in the habitat (indexed by e). Notably, these dominant odor sources are those *successful* search agents with highest reward. δ_1 is included to calculate the overall impact of wind (distraction) in the habitat, and γ_e is the specific distraction rate of the e-th odor source to follow. To calculate $\overrightarrow{X}'_e(i)$, we consider the following equation (random walk):

$$\overrightarrow{X}'_e(i) = \overrightarrow{X}_e(i) + \beta K_e(i)\mathbf{J}_{1\times D}, \quad 1 \le e \le E \quad (6)$$

where $\overrightarrow{X}_e(i)$ shows the actual position of dominant odor source e in the habitat, $K_e(i)$ the maximum distance that molecules of ordor source e diffuse in the habitat as hermit crabs get closer to odor sources in iterations, β a random number in range $[-1, 1]$, and $\mathbf{J}_{1\times D}$ a unit matrix. To calculate $K_e(i)$, the diffusion rate of a dominant odor molecule ($D_e(i)$) during a time interval (Δt) should be considered. The maximum travelable distance of this molecule in each direction is calculated by (Bird, 2002):

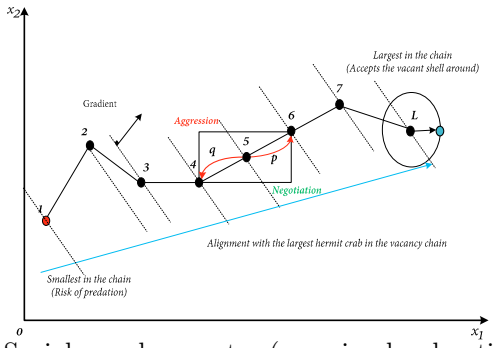

Fig. 4. Social search operator (escaping local optima).

$$K_e(i) = \sqrt{2D_e(i)\Delta t}, \quad 1 \le e \le E \tag{7}$$

where $D_e(i)$ and Δt are calculated by:

$$D_e(i) = \frac{\left|g\left(\overrightarrow{X}_e(i)\right) - g\left(\overrightarrow{X}_n(i)\right)\right|}{\left|g^+\left(\overrightarrow{X}_n(i)\right) - g^-\left(\overrightarrow{X}_n(i)\right)\right|}, \quad 1 \le e \le E \tag{8}$$

and:

$$\Delta t = i/I \tag{9}$$

The distraction rate of each hermit crab to follow a dominant source of odor is calculated as follows:

$$\gamma_e = (1 - i/I)\beta\,(2b-1)\,(1-\rho_e), \quad 1 \le e \le E \tag{10}$$

where:

$$\rho_e = \frac{\left|g\left(\overrightarrow{X}_E(i)\right) - g\left(\overrightarrow{X}_e(i)\right)\right|}{\left|g^+\left(\overrightarrow{X}_n(i)\right) - g^-\left(\overrightarrow{X}_n(i)\right)\right|}, \quad 1 \le e \le E \tag{11}$$

Finally, δ_1 shows the effect of wind in the habitat:

$$\delta_1 = c + 2\alpha\,(1-c) \tag{12}$$

It should be noted that after using the position of dominant odor sources, they are updated as follows (for $1 \le e \le E$):

if $\alpha_3 < 0.5$:

$$\overrightarrow{X}_e(i+1) = \overrightarrow{X}_e(i) + \alpha_1\left(\overrightarrow{X}^-(i) + \alpha_2\left(\overrightarrow{X}^+(i) - \overrightarrow{X}^-(i)\right)\right) \tag{13}$$

else:

$$\overrightarrow{X}_e(i+1) = \overrightarrow{X}_e(i) - \alpha_1\left(\overrightarrow{X}^-(i) + \alpha_2\left(\overrightarrow{X}^+(i) - \overrightarrow{X}^-(i)\right)\right) \tag{14}$$

If only one search agent exists for the solitary search process, or the size of solitary search agents is smaller than or equal to the elitism number (E), the positions of all solitary search members are updated similarly. Notably, all αs are random numbers in range [0,1].

2.2.8 Social search operator This stage is only pursued by those search agents with *failed* label (again, n does not count till N). Fig. 4 shows how a social search process updates hermit crabs' positions in a 2D search space. A transition matrix denoting the probability of shell exchange of a hermit crab with others in the vacancy chain, considering each hermit crab would prefer only a single hermit crab with the closest shell size (higher or lower sized than with its current shell), would be as (Weissburg et al., 1991):

$$\mathbf{W} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{R} & \mathbf{Q} \end{bmatrix} \tag{15}$$

where $\mathbf{I}$ is a 2×2 matrix corresponding to fixing (absorbing) positions of the smallest and largest hermit crab in the vacancy chain (regarding the obtained rewards between *failed* search agents), $\mathbf{Q}$ is a $(L-2) \times (L-2)$ matrix that corresponds to the shell exchanges between hermit crabs in the vacancy chain (sorted from largest to smallest according to the obtained rewards), $\mathbf{O}$ is a zeros matrix and finally, $\mathbf{R}$ is a matrix of transitions from transient hermit crabs to absorbing states. For instance, we may reach a transition matrix for hermit crabs in a vacancy chain as:

$$\mathbf{W} = \begin{array}{c} \\ 1 \\ 5 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 5 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ q & 0 & 0 & p & 0 \\ 0 & 0 & q & 0 & p \\ 0 & p & 0 & q & 0 \end{bmatrix} \end{array} \tag{16}$$

This phenomenon is similar to a Markov chain process. A random walk (a decision to choose one conspecific with larger (to negotiate) or smaller (to aggress) shell size) occurs. It is independent of how the hermit crab got to its current state in the first place and the probabilities p and q regardless of when a shell exchange takes place. Therefore each defines a conditional probability that $p(X_{n+1} = j|X_n = i)$. A move from state i to j is statistically independent of all the moves that led to i. Notably, all elements of $\mathbf{W}$ are positive, and the sum of elements in a row equals 1.

Finally, to mimic this stochastic shell exchange process, the social search operator for each hermit crab in the vacancy chain is:

$$\overrightarrow{X}_n(i+1) = \overrightarrow{X}'_n(i) + \beta\left|\delta_2\overrightarrow{X}_L(i) - \overrightarrow{X}'_n(i)\right| \tag{17}$$

where:

$$\overrightarrow{X}'_n(i) = p\overrightarrow{X}_{n+1}(i) + q\overrightarrow{X}_{n-1}(i)\,(i) \tag{18}$$

and

$$\delta_2 = c + 2\alpha\,(1-c) \tag{19}$$

Next, as the smallest hermit crab in the vacancy chain has a risk of being shell-less or being predated, and the largest one may look around for a suitable shell, the states of these two search agents are updated as follows:

$$\overrightarrow{X}_1(i+1) = \begin{cases} Reinitialize, \alpha_2 < 0.5 \\ (1-(2b-1))\,(1+\beta)\,\delta_2\overrightarrow{X}_L(i), \alpha_2 \ge 0.5 \end{cases} \tag{20}$$

and

$$\overrightarrow{X}_L(i+1) = \begin{cases} \overrightarrow{X}_L(i) + \beta\mathbf{J}_{1\times D}, \alpha_3 < 0.5 \\ \overrightarrow{X}_E(i) - \beta\left|\overrightarrow{X}_E(i) - \overrightarrow{X}_L(i)\right|, \alpha_3 \ge 0.5 \end{cases} \tag{21}$$

Note that E represents the optimal solution so far (the most dominant search agent (odor source), largest hermit crab, the search agent with the highest reward) (among successful search agents), but L represents the locally optimal solution (among failed search agents) in the vacancy chain.

The flowchart and pseudocode of HCO are available in Fig. 5 and Algorithm 1, respectively.

3 ANALYTICS AND INSIGHTS

Herein, HCO is implemented on five randomly chosen well-known but complex test problems (in terms of landscape), all of which have a minimization direction (Table 1). For these computational tests, $N = 60$, $E = 3$, and $I = 200$. Notably, parameter tuning is based on trial and error.

First, it is analyzed how HCO converges to an optimal or a near-optimal solution over iterations. The results are illustrated in Fig. 6. The 3D landscape considers only two

Table 1. Test problems.

Problem	Name	Fitness function	Dimension	Range	Optimal
F5	Rosenbrock	$\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+(x_i-1)^2\right]$	30	[-30,30]	0
F8	Schwefel (2.26)	$\sum_{i=1}^{n}-x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	30	[-512,512]	-12569.5
F13*	Penalized 2	$0.1\left(sin^2\left(3\pi x_1\right)+\sum_{i=1}^{n}(x_i-1)^2\left[1+sin^2\left(3\pi x_i+1\right)\right]+(x_n-1)^2\left[1+sin^2\left(2\pi x_n\right)\right]\right)+\sum_{i=1}^{n}u\left(x_i,5100,4\right)$	30	[-50,50]	0
F22	Shekel 7	$-\sum_{i=1}^{7}\left[(x-a_i)(x-a_i)^T+c_i\right]^{-1}$	4	[0,10]	-10.4029
O3	Cross-in-Tray	$-0.0001(\left\lvert\sin(x_1)\sin(x_2)\exp\left(\left\lvert 100-\sqrt{x_1^2+x_2^2}/\pi\right\rvert\right)\right\rvert+1)^{0.1}$	2	[-10,10]	-2.06261

* $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m x_i>a\\0-a<x_i<a\\k(-x_i-a)^m x_i<-a\end{cases}$

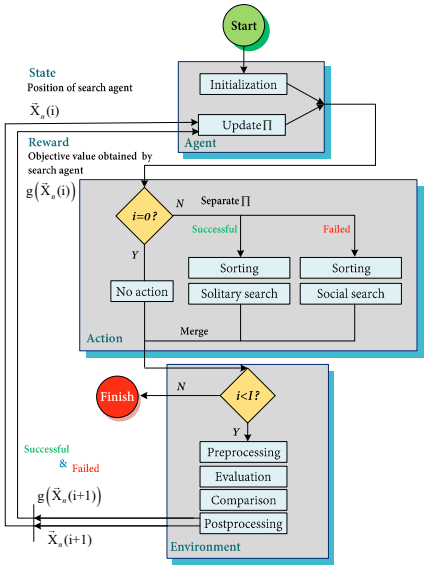

Fig. 5. Proposed HCO flowchart.

dimensions (x_1 and x_2). The 2D search trajectory shows how the search agents are distributed in the 3D landscape, considering stagnation in locally optimal solutions (i.e., it represents the last snapshot of search agents in the last iteration). The convergence curve indicates the impact of such stagnations. Besides, a new indicator called search efficiency is introduced, which shows the number of successful search agents throughout iterations.

Considerable positive aspects can be concluded when HCO is applied. First, it is seen that almost all search agents align with the agent with the highest reward value and have escaped (or are escaping) from locally optimum solutions. Second, the search efficiency indicator shows HCO preserves its search efficiency, even in the iterations close to the last one. It is an important aspect, as it is seen that still the MA can generate a new solution and can progress. Moreover, it indicates that although HCO looks to exploit the best solutions derived in an iteration, it values other optimal solutions too for exploitation. This feature is due to the inherent characteristic of these social creatures. Hermit crabs, who find a shell in solitary search, motivate others with successful shell exchange to move toward them; however, those who do not find a shell use vacancy chains to exchange shells, disregarding those who were successful in shell finding (disregarding immediate rewards). This means that HCO is an algorithm that is both greedy (exploit) and random (explore) since search agents learn to adapt to their habitat. Finally, considering the convergence curve and search efficiency together, it is observed that HCO fastly converges to the optimal solution (not the local one), which ensures deriving high-quality solutions in little time. Notably, the red dot shows the optimal solution reported by HCO.

Algorithm 1 Hermit Crab Optimizer (HCO).

— * *Initialization*:
$i \leftarrow 0$ (iteration indicator, max: I)
$\mathbf{P}(0) \leftarrow$ use encoder
$\mathbf{R}(0), \mathbf{B} \leftarrow +\infty$ (for min direction)
$\mathbf{\Pi}(0) \leftarrow [\mathbf{P}(0), \mathbf{R}(0)]$ (state-reward matrix)
$\mathbf{P}^!(0) \leftarrow \mathbf{P}(0)$ (clone of state matrix)
while $(i < I)$ **do**
— * *Preprocessing*:
  $a \leftarrow 0$ (total shells found)
  $b \leftarrow 0$ (perceived risk of shell availability)
  $c \leftarrow 0$ (indicator of distraction)
  fix $\mathbf{P}^!(i)$ if boundaries are violated
— * *Evaluation*:
  for $n = 1, \ldots, N$ **do**
    $g\left(\vec{X}^!_n(i)\right) \leftarrow$ use decoder
    fit $\leftarrow g\left(\vec{X}^!_n(i)\right)$
— * *Comparison*:
    if fit $< g\left(\vec{X}_n(i)\right)$ **then**
      $\mathbf{P}(i) \leftarrow \vec{X}^!_n(i)$
      $\mathbf{R}(i) \leftarrow g\left(\vec{X}^!_n(i)\right)$
      $\mathbf{\Pi}(i) \leftarrow [\mathbf{P}(i), \mathbf{R}(i)]$
      $a \leftarrow a + 1$
      $f(n) \leftarrow 0$ (indicator of failure)
      if fit $<$ **B then**
        $\mathbf{B} \leftarrow g\left(\vec{X}^!_n(i)\right)$
        $\mathbf{S} \leftarrow \vec{X}^!_n(i)$
        $c \leftarrow 1$ (indicator of distraction)
      end if
    else
      $f(n) \leftarrow 1$
    end if
  end for
— * *Postprocessing*:
  $b \leftarrow (1 - a/N)$
  $\mathbf{P}^c(i) \leftarrow$ *empty*
  $\mathbf{\Pi}^1(i) \leftarrow \mathbf{\Pi}(i)$ where $f(n) = 0$
  $\mathbf{\Pi}^2(i) \leftarrow \mathbf{\Pi}(i)$ where $f(n) = 1$
— * *Solitary search*:
  sort $\mathbf{\Pi}^1(i)$ regarding $\mathbf{R}(i)$
  for $n \in \mathbf{\Pi}^1(i)$ **do**
    $\vec{X}^!_n(i+1) \leftarrow$ Solitary search operator
  end for
— * *Social search*:
  sort $\mathbf{\Pi}^2(i)$ regarding $\mathbf{R}(i)$
  for $n \in \mathbf{\Pi}^2(i)$ **do**
    $\vec{X}^!_n(i+1) \leftarrow$ Social search operator
  end for
  $\mathbf{P}^!(i+1) \leftarrow$ merge $\vec{X}^!_n(i+1)$ from $\mathbf{\Pi}^1(i)$ & $\mathbf{\Pi}^2(i)$
  $i \leftarrow i + 1$
end while
return $[\mathbf{S}, \mathbf{B}]$

To test the accuracy and robustness of an MA, one should analyze the solutions reported by multiple times of implementation of an MA on the same optimization problem. Herein, HCO is compared with most popular and recent MAs, namely, PSO (Kennedy and Eberhart, 1995), ICA (Atashpaz-Gargari and Lucas, 2007), GWO (Mirjalili et al., 2014), and Equilibrium Optimizer (EO) (Faramarzi et al., 2020). Notably, the setting of the other

Table 2. Results of MAs on five test problems with 30 implementations.

Algorithm	PSO		ICA		GWO		EO		**HCO**	
Problem	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
F5	35.250	28.892	349.040	521.294	26.580	0.748	24.581	0.229	**0.000**	**0.000**
F8	-6969.768	830.473	-9748.967	750.391	-6118.410	0.094	-9860.201	355.369	**-12569.487**	**0.000**
F13	0.023	0.032	0.001	0.003	0.413	0.208	0.002	0.008	**0.000**	**0.000**
F22	-8.423	3.091	-10.051	1.316	-10.402	0.001	-10.049	1.326	**-10.403**	**0.000**
O3	-2.063	0.000	-2.063	0.000	-2.063	0.000	-2.063	0.000	**-2.063**	**0.000**

algorithms was not changed (as experimentally suggested by their developers), but it was ensured to have the same number of iterations (500) and populations (60). All MAs were implemented 30 times. The obtained results are shown in Table 2. As can be seen, HCO outperforms other MAs regarding both accuracy and robustness. By accuracy, we mean the closeness of the average solutions found to the real ones found by exact solvers. Besides, by robustness, we mean proximity of standard deviation to zero (as it means an MA has reached almost the same solution even if multiple times implemented, even if it has a stochastic nature). It can be concluded that HCO is more capable than both the famous MAs and the most recent ones. Based on such characteristics, and considering that these problems are not too complex (compared with problems such as traveling salesman and quadratic assignment problems), it is clearly concluded that HCO promises to derive better solutions than its previous famous MAs.

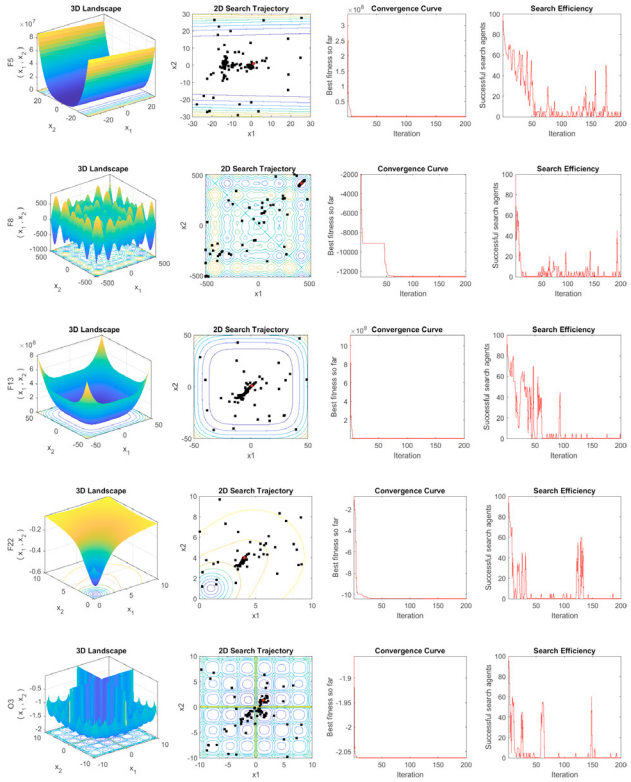

Fig. 6. Validation results by implementing HCO on test problems.

4 CONCLUSION

This paper proposed a novel nature-inspired meta-heuristic algorithm (MA), namely, Hermit Crab Optimizer (HCO), simulating swarm intelligence of hermit crabs and their solitary/social shell searching behavior in habitat. HCO is a straightforward parallel MA and a pseudo reinforcement learning (RL) algorithm. It requires only three inputs (i.e., population size, iteration number, and elitism number). It adapts to the landscape by learning from previous experiences (e.g., the perceived risk of shell availability and wind distraction). The validity, accuracy, and robustness of HCO were confirmed with computational experiments on multiple well-known test problems compared with prominent and recent MAs. In all tests, the HCO was found to be superior. As the HCO was only implemented on test problems, future research should extend and prove its applicability to other optimization, simulation, and machine learning problems.

REFERENCES

Amindoust, A., Ahwazian, A., Tavakkoli-Moghaddam, R., and Nikbakht, M. (2021). Search-in-forest optimizer: A bio-inspired meta-heuristic algorithm for global optimization problems. *Soft Computing (Accepted for publication)*.

Atashpaz-Gargari, E. and Lucas, C. (2007). Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. In *2007 IEEE Congress on Evolutionary Computation*, 4661–4667. IEEE.

Bird, R.B. (2002). Transport phenomena. *Applied Mechanics Reviews*, 55(1), R1–R4.

Briffa, M. and Bibost, A.L. (2009). Effects of shell size on behavioural consistency and flexibility in hermit crabs. *Canadian Journal of Zoology*, 87(7), 597–603.

Faramarzi, A., Heidarinejad, M., Stephens, B., and Mirjalili, S. (2020). Equilibrium optimizer: A novel optimization algorithm. *Knowledge-Based Systems*, 191, 105190.

Fathollahi-Fard, A.M., Hajiaghaei-Keshteli, M., and Tavakkoli-Moghaddam, R. (2018). The social engineering optimizer (seo). *Engineering Applications of Artificial Intelligence*, 72, 267–293.

Fathollahi-Fard, A.M., Hajiaghaei-Keshteli, M., and Tavakkoli-Moghaddam, R. (2020). Red deer algorithm (rda): a new nature-inspired meta-heuristic. *Soft Computing*, 24(19), 14637–14665.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5), 533–549.

Gravel, B.E., Wong, P.Y., Starks, P.T., and Pechenik, J.A. (2004). The use of artificial shells for exploring shell preference in the marine hermit crab pagurus longicarpus (say). In *Annales Zoologici Fennici*, 477–485. JSTOR.

Hajipour, V., Mehdizadeh, E., and Tavakkoli-Moghaddam, R. (2014). A novel pareto-based multi-objective vibration damping optimization algorithm to solve multi-objective optimization problems. *Scientia Iranica-Transaction E*, 21(6), 2368.

Hooke, R. and Jeeves, T.A. (1961). "direct search"solution of numerical and statistical problems. *Journal of the ACM (JACM)*, 8(2), 212–229.

Kashan, A.H., Tavakkoli-Moghaddam, R., and Gen, M. (2019). Find-Fix-Finish-Exploit-Analyze (F3EA) meta-heuristic algorithm: An effective algorithm with new evolutionary operators for global optimization. *Computers & Industrial Engineering*, 128, 192–218.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, 1942–1948vol.4. IEEE.

Mirjalili, S., Mirjalili, S.M., and Lewis, A. (2014). Grey Wolf Optimizer. *Advances in Engineering Software*, 69, 46–61.

Rotjan, R.D., Chabot, J.R., and Lewis, S.M. (2010). Social context of shell acquisition in coenobita clypeatus hermit crabs. *Behavioral Ecology*, 21(3), 639–646.

Sutton, R.S. and Barto, A.G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tafakkori, K. (2022). Artificially intelligent algorithms for optimization in Python. URL `https://www.supplychaindataanalytics.com/artificially-intelligent-algorithms-for-optimization-in-python`.

Weissburg, M., Roseman, L., and Chase, I.D. (1991). Chains of opportunity: a markov model for acquisition of reusable resources. *Evolutionary Ecology*, 5(2), 105–117.

Wolpert, D.H. and Macready, W.G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67–82.